MODUL

Januari 2020

ALGORITMA & PEMROGRAMAN

ROHMAT TAUFIQ, ST.M.KOM

Referensi :
Abdul Kadir dan Heriyanto

Teknik Informatika
Fakultas Teknik
Universitas Muhammadiyah Tangerang
2020

BAB I
PENGANTAR ALGORITMA DAN PEMROGRAMAN

1.1 Pengertian Program dan Bahasa Pemrograman

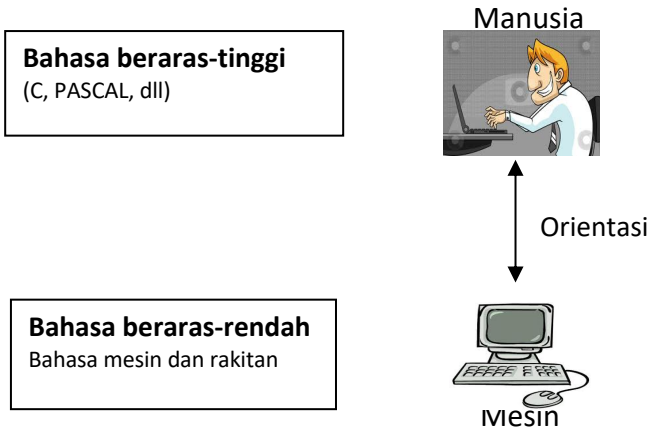
Yang dimaksud dengan pemograman adalah kumpulan instruksi yang digunakan untuk mengatur komputer agar melakukan suatu tindakan tertentu. Tanpa program, komputer sesungguhnyan tidak dapat berbuat apa-apa. Itulah sebabnya sering dikatakan bahwa computer mencangkup tiga aspek penting, berupa perangkat keras (hardware), perangkat lunak (software)yang dalam hal ini berupa program, dan perangkat akal (brainware) atau orang yang berperan dalam operasi computer maupun perangkat lunak. Dengan kata lain program merupakan salah satu bagian penting pada computer yang mengatur komputer dalam melakukan aksi sesuai dengan yang dikehendaki oleh pembuatnya.

Orang yang membuat program biasa disebut pemogram (Programer). Adapun aktifitas yang berhubungan dengan pembuatan program dinamakan pemograman (programming).

Suatu program ditulis dengan mengikuti kaidah bahasa pemograman tertentu. Bahasa pemograman dapat dianalogikan dengan bahasa yang digunakan oleh manusia (bahasa manusia). Sebagaimana diketahui, ada bermacam-macam bahasa manusia, seperti bahasa inggris, Indonesia, ataupun bahasa batak. Kumpulan intruksi bahasa manusia yang berupa sejumlah kalimat dapat anda analogikan dengan suatu program. Manusia dapat mengerjakan intruksi berdasarkan kalimat-kalimat dan computer dapat menjalankan intruksi dengan menurut program.

Dalam konteks pemograman, terdapat sejumlah bahasa pemograman, seperti Pascal, C, C++, dan BASIC, secaragaris besar, bahasa pemograman dapat dikelompokan menjadi :

1. Bahasa beraras-tinggi (*high-level leangue*)
2. Bahasa beraras-rendah (*low-level leangue*)



Gambar 1.1*Kelompok bahasa pemograman*

Bahasa beraras-tinggi adalah bahasa pemograman yang berorientasi kepada bahasa manusia. Program dibuat dengan bahasa pemograman yang mudah dipahami oleh manusia, biasanya menggunakan kata-kata bahasa inggris: misalnya IF untuk

menyatakan “JIKA” dan AND untuk menyatakan “DAN”. Yang termasuk bahasa ini adalah bahasa C, C++, PASCAL, dan BASIC.

Bahasa beraras-rendah adalah bahasa pemograman yang berorientasi kepada mesin. Bahasa ini menggunakan kode biner (yang hanya mengenal kode 0 dan 1) atau suatu kode sederhana untuknmengunakan kode-kode tertentu dalam system biner. Yang tergolong dalam bahasa ini adalah bahasa mesin dan bahasa rakitan. Bahasa seperti itu sulit dipahami oleh orang awam dan sangat membosankan bagi pemogram yang sudah terbiasa dengan bahasa beraras-tinggi.nPemograman harus benar-benar menguasai operasi computer secara teknis. Namun bahasa generasi ini memberikan eksekusi program secara cepat. Selain itu bahasa mesin sangat bergantung pada mesin (machine dependent); Artinya, bahasa mesin dengan mesin satu dengan mesin yang lain jauh berbeda. Sebagai contoh, ***Tabel 1.1*** memperlihatkan tiga buah intruksi dalam bahasa mesin yang diterapkan pada IBM PC yang berbasis system operasi DOS.

Tabel 1.1*Kode dalam bahasa mesin*

INTRUKSI BAHASA MESIN	KETERANGAN
B402 atau 1011 0100 0010	Muatlah bilangan 2 ke register AH
B22A atau 1011 0010 0010 1010	Muatlah bilangan 2A heksadesimal ke register DL
CD21 atau 1100 1101 0010 0001	Jalankan interupsi 21 heksadesimal

Tiga intruksi diatas digunakan untuk menampilkan tanda * pada layar. Bandingkan dengan beberapa perintah berikut yang ditulis pada bahasa pemograman beraras tinggi :

```
WRITE ('*');          (pada Pascal)
DISPLAY '*".          (pada CABOL)
PRINT "*"             (pada BASIC)
printf ("*");         (pada C)
cout <<"*";           (pada C++)
```

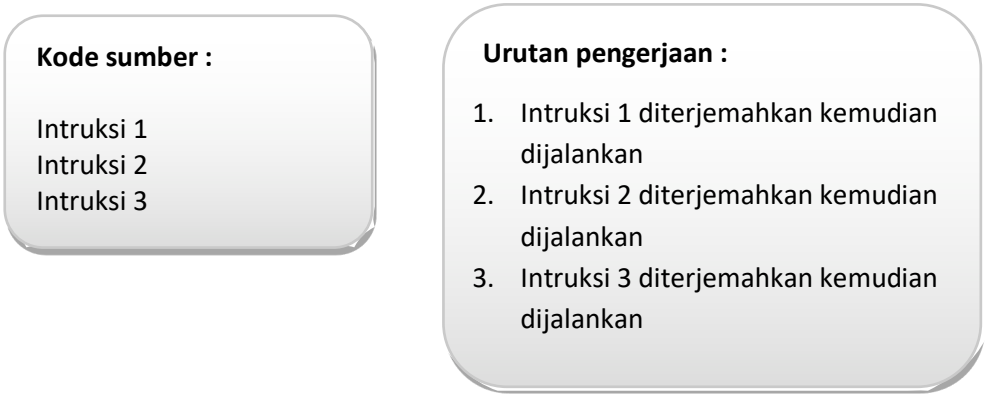
Tampak bahasa mesin lebih panjang dan sukaat dimengerti dibandingkan kode dalam bahasa yang lebih berorientasi pada manusia.

1.2 Penerjemah Bahasa

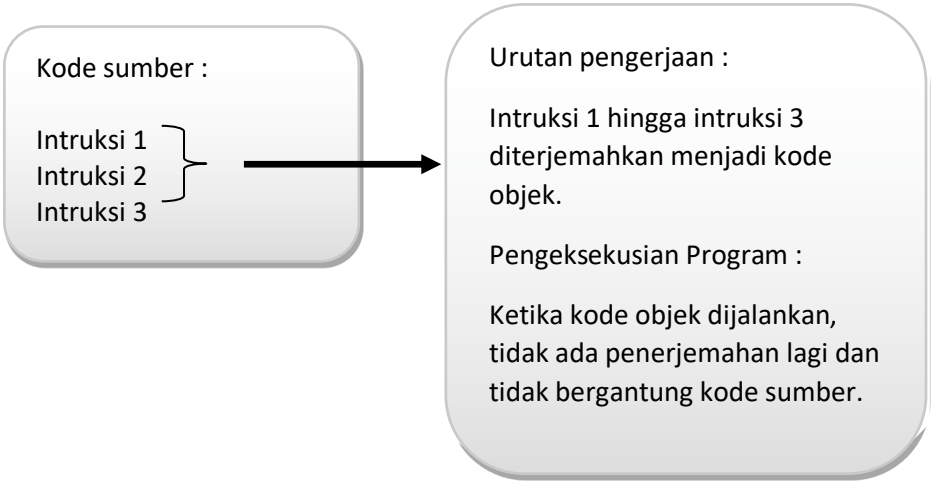
Program yang ditulis Dallam bahasa pemograman seperti C dan C++ sebenarnya tidak dimengerti computer secara langsung, sebeb computer hanya mengenal bahasa khasnya saja yang dinamakan bahasa mesin, dijalankan (dieksekusi) oleh computer, program tersebut harus diterjemahkan terlebih dahulu kedalam

bahasa mesin (atau bisa disebut sebagai KODE OBJEK). Proses penerjemahan dilakukan oleh program yang disebut translator (penerjemah).
Translator dapat berupa :

1. Interpreter
2. Kompiler



(a) Proses pDEenerjemah pada interpreter

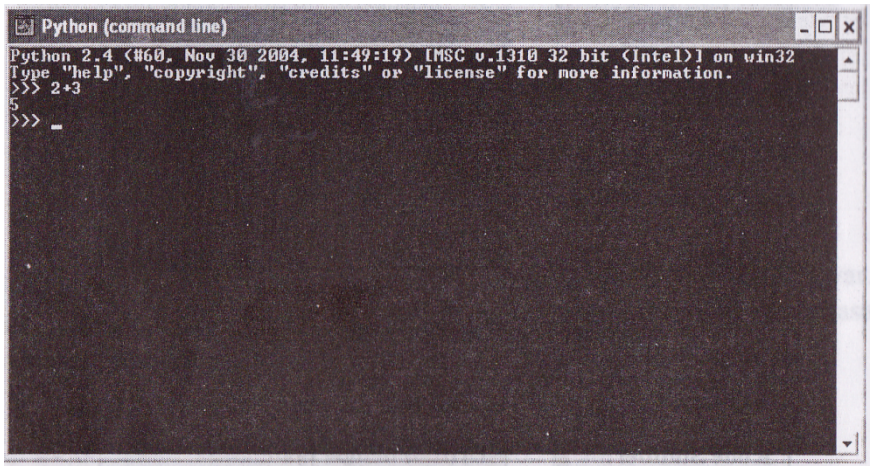


(b) Proses penerjemahan pada compiler

Gambar 1.2*Perbedaan interpreter dan compiler dalam menerjemahkan program*

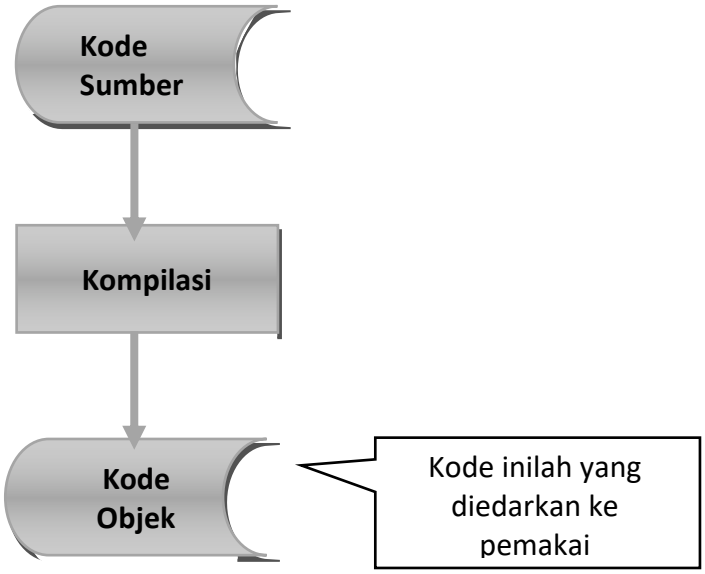
Interpreter menerjemahkan intruksi selama program dimita untguk dieksekusi. Jika seorang bermaksud menjalankan program tersebut (agar dapat dijalankan computer), mula-mula kode sumber (source code atau program asli yang ditulis oleh pemogram).

Diterjemah dulu ke dalam bentuk kode mesin perbaris instruksi. Setelah satu basri instruksi tersebut dipahami oleh computer, instruksi tersebut dijalankan. Interpreter kemudian kembali memproses baris instruksi berikutnya.



Gambar 1.3*Contoh Interpreter Python. Anda dapat memanfaatkannya seperti kalkulator*

Berbeda dengan interpreter, kompiler menerjemahkan instruksikedalam bentuk kode objek secarakeseluruhan (untuk semua instruksi). Setelah semua instruksi diterjemah , instruksi yang telah mengerti oleh komputer dijalankan. Proses penerjemah seperti itu disebut kompilasi. Setelah kompilasi berakhir, compiler yang *executable* (dapat dijalankan secara langsung tanpa melalui translator).



Gambar 1.4*Kompilasi mengubah kode sumber menjadi kode objek*

Catatan

- Proses pembentuk kode yang *executable* sebenarnya juga melalui proses yang disebut *linking*, yang berfungsi untuk menggabungkan kode hasil kompilasi dengan sejumlah pustaka yang disediakan oleh kompilasi.
- Pada saat ini, beberapa kompilasi menerjemah kode sumber dalam bentuk kode yang khas, yang tidak bersifat *executable*. Compiler pada bahasa Java

mengkompilasi kode menjadi kode objek yang disebut *bytecode*. Kode seperti ini dapat dijalankan pada berbagai platform sepanjang platform tersebut memiliki program yang dapat mengeksekusi kode tersebut.

Tentu saja masing – masing translator memiliki keunggulan dan kelemahan sendiri, sebagaimana dapat dilihat pada Tabel 1.2.

Tabel 1.2*Kelebihan dan Kelemahan Interpreter dan Kompiler*

Interpreter	Kompiler
Kelebihan :	
Kemudahan mencari kesalahan seandainya program menghasilkan sesuatu yang dianggap salah ketika program dijalankan, karena kode sumber selalu tersedia	1. Pengerjaan instruksi dilakukan dengan sangat cepat, karena setelah kode objek terbentuk maka tidak perlu lagi adanya penerjemahan, mengingat komputer dapat memahami kode objek secara langsung. 2. Kode objek dapat didistribusikan ke komputer lain tanpa perlu menyertakan kode sumber dan kompiler, sehingga kerahasiaan kode sumber tetap terjamin.
Kelemahan :	
1. Kode sumber harus selalu tersedia 2. Eksekusi lambat	Seluruh kode sumber harus benar secara sintaks agar program dapat diuji

Bahasa pemrograman juga dapat diklasifikasikan menjadi bahasa procedural dan bahasa deklaratif. Pada bahasa procedural, pemrogram perlu menuliskan instruksi instruksiyang rinci agar komputer dapat melaksanakan tugasnya. Pendekatan prosedural terdapat dalam bahasa – bahasa pemograman seperti C, Pascal, dan BASIC. Pada bahasa deklaratif (non-prosedural), untuk mendapatkan suatu hasil, seorang pemrogram tidak perlu memberitahukan secara detail tentang bagaimana mendapatkannya. Pendekatan deklaratif antara lain diterapkan pada bahasa seperti Prolog (bahasa untuk menangai kecerdasan buatan).

Klasifikasi yang lain berupa bahasa yang berorientasi pada objek atau tidak berorientasi pada objek. Bahasa pemrograman yang tidak berorientasi objek adalah bahasa pemrograman yang memungkinkan untuk mengemas suatu prosedur dan data dalam suatu wadah yang disebut kelas. Kelas inilah yang menjadi cetakan bagi objek-objek. Model pemrograman seperti ini telah menjadi tren bahasa – bahasa pemrograman basru pada masa kini. Keuntungannya, kode yang telah dibuat dapat dikembangkan secara mudah. Sifat ini dikenal sebagai sebutan *reusability*. Termasuk dalam bahasa yang berorientasi objek adalah C++. Adapun C adalah bahasa pemrograman yang tidak dapat berorientasi pada objek.

1.3 Penyelesaian Masalah dengan Program

Orang membuat program biasanya bertujuan untuk menyelesaikan masalah. Namun sebelum dapat menyelesaikan masalah dengan program, terdapat tiga langkah penting yang perlu di lakukan terlebih dahulu.

1. Menganalisis masalah dan membuat algoritma,
2. Menuangkan algoritma ke dalam bentuk program,
3. Mengeksekusi dan menguji program,

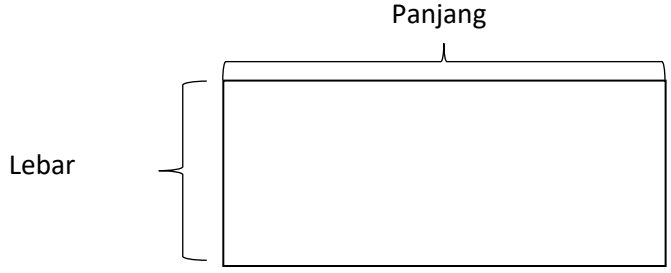
1.3.1 Menganalisis Masalah dan Membuat Algoritma

Proses pertama, menganalisis masalah dan membuat algoritma memang tidak dapa begitu saja diwujudkan. Pengalaman, pengetahuan, kreatifitas, imajinasi, dan kelihaian merupakan factor – factor yang menentukan sekali keberhasilan langkah ini. Di dalam analisis masalah diperlukan tindakan untuk mengidentifikasi informasi yang menjadi keluaran pemecah masalah dan data – data yang menjadi masukan.

Gambar 1.5*Kerangka pemecahan masalah*

Sebagai contoh sederhana, anda diminta untuk menghitung keliling persegi panjang dengan menggunakan komputer. Pada masalah ini anda dapat menentukan keluaran berupa keliling dan masukan berupa panjang dan lebar. Setelah itu anda harus menentukan bagaimana keliling persegi panjang dihitung berdasarkan data panjang dan lebar. Dengan logika sederhana, anda dapat menghitung keliling persegi panjang seperti berikut:

Keliling = panjang + lebar + panjang + lebar
Atau
Keliling = 2 x (panjang + lebar)



Gambar 1.6 Persegi panjang dan keliling

Anda dapat menuangkan algoritma seperti berikut :

1. Peroleh nilai panjang dan lebar persegi panjang
2. Hitung keliling persegi panjang dengan menggunakan rumus 2 x (panjang + lebar)
3. Tampilkan nilai keliling persegi panjang

Dalam praktik, banyak persoalan yang solusinya tidak semudah itu. Sebagai contoh yang sederhana, anda diminta untuk menghitung luas lingkaran yang diketahui jari-jarinya. Bagaimana anda menghitung luas lingkaran? Apakah dapat dengan hanya menggunakan logika sederhana?. Tentu saja tidak, anda harus memiliki pengetahuan tentang rumus luas lingkaran. Jika anda tidak tahu, anda perlu mencari tahu melalui buku atau bertanya pada orang yang tahu. Setelah anda tahu bagaimana cara menghitung luas lingkaran, anda baru dapat menuangkannya dalam Bentuk algoritma sebagai berikut:

1. Peroleh jari – jari lingkaran
2. Hitung luas lingkaran dengan menggunakan rumus 3,14 x jari – jari^2.
3. Tampilkan nilai luas lingkaran.

Algoritma tidak selamanya dinyatakan dalam bahasa manusia seperti contoh yang di depan. Kadang-kadang dinyatakan dalam bentuk pseudokode (*pseudocode*), yaitu suatu bentuk algoritma yang menggunakan bahasa notasi yang dimaksudkan untuk menyederhanakan bentuk kalimat manusia. Sebagai contoh, kalimat seperti ”Hitung keliling persegi panjang dengan mengunakan rumus 2 x (panjang + lebar)” dapat disederhanakan menjadi

Keliling ← 2 x (panjang + lebar)

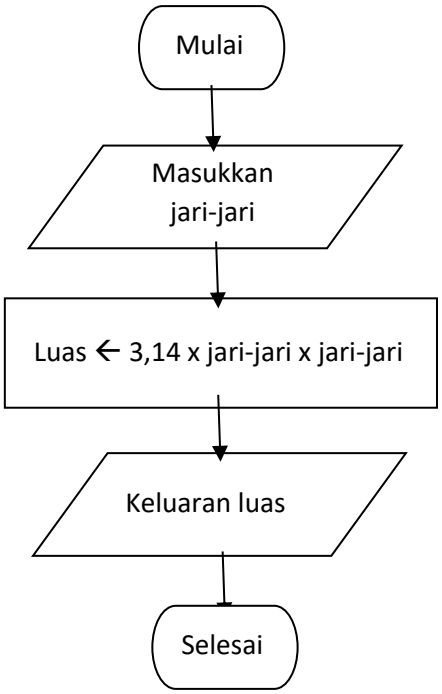
Pseudokode sering kali mengarah ke bahasa pemrograman tertentu. Sebagai contoh:

1. i ← 1
2. jum ← 0
3. while i <10
    a. jum ← jum + i
    b. i ← i + 1
4. write (i)

Merupakan pseudokode yag berorientansi pada bahasa C, Pascal, atau Algol. Algoritma di atas di gunakan untuk melakukan perhitungan 1 +2 + 3 + 4 + 5 + 6 + 7 + 8 + 9.

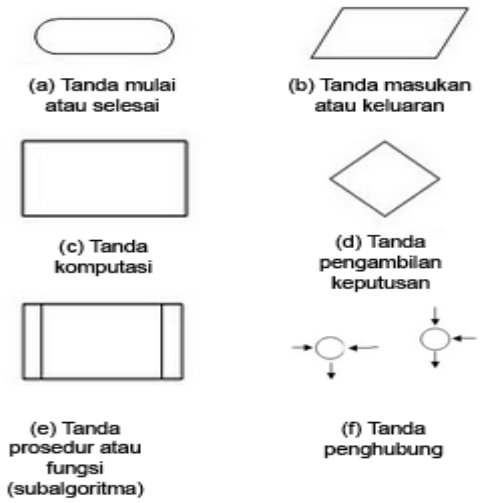
Adakalanya suatu algoritma disajikan dalam bentuk diagram alir (*flowchart*). Diagram alir adalah suatu standar untuk menggambarkan urutan langkah dalam suatu proses. Setiap langkah dalam algoritma dinyatakan dalam sebuah symbol dan aliran setiap langkah (dari suatu langkah ke langkah yang lain) dinyatakan dengan garis yang

di lengkapi panah. Gambar 1.7 memperlihatkan diagram alir untuk menghitung luas persegi lingkaran.



Gambar 1.7*Diagram alir penghitungan luas lingkaran*

Adapun gambar 1.8 memperlihatkan sejumlah simbol standar yang digunakan untuk menyusun diagram alir.



Gambar 1.8*Simbol standar untuk diagram alir*

Catatan

Untuk membuat suatu diagram air, Anda dapat menggunakan perangkat lunak seperti Visio atau bahkan Word (melalui AutoShapes) di lingkungan Windows.

Dalam praktik seringkali pemrogram menyelesaikan persoalan dengan menggabungkan algoritma-algoritma yang pernah di buat oleh orang lain. Sebagai contoh,di dalam mengurutkan sekumpulan data,pemrogram dapat menggunakan algoritma seperti *quik sort* ataupun *insertion sort.* Algoritma-algoritma seperti itu banyak di ulas dalam buku-buku yang membahas masalah algoritma, setruktur data, dan pemrograman.Oleh karena itu,selain harus memiliki kemampuan dasar dalam membuat algoritma,seorang pemrogram juga perlu mengetahui beberapa algoritma yang pernah di buat orang supaya penyelesaian masalah dapat dilakukan secara optimal.

1.3.2 Menuangkan Algoritma ke dalam Bentuk Program

Langkah menuangkan algoritma ke dalam program di tentukan oleh faktor bahasa perograman yang akan di gunakan. Sebagai contoh,langkah "Hitung keliling persegi panjang dengan menggunakan rumus 2x (panjang+lebar)" perlu di terjemahkan menjadi pertanyaan berikut pada sejumlah bahasa pemrograman.

Tabel 1.3 *Pernyataan pada beberapa bahasa pemrograman*

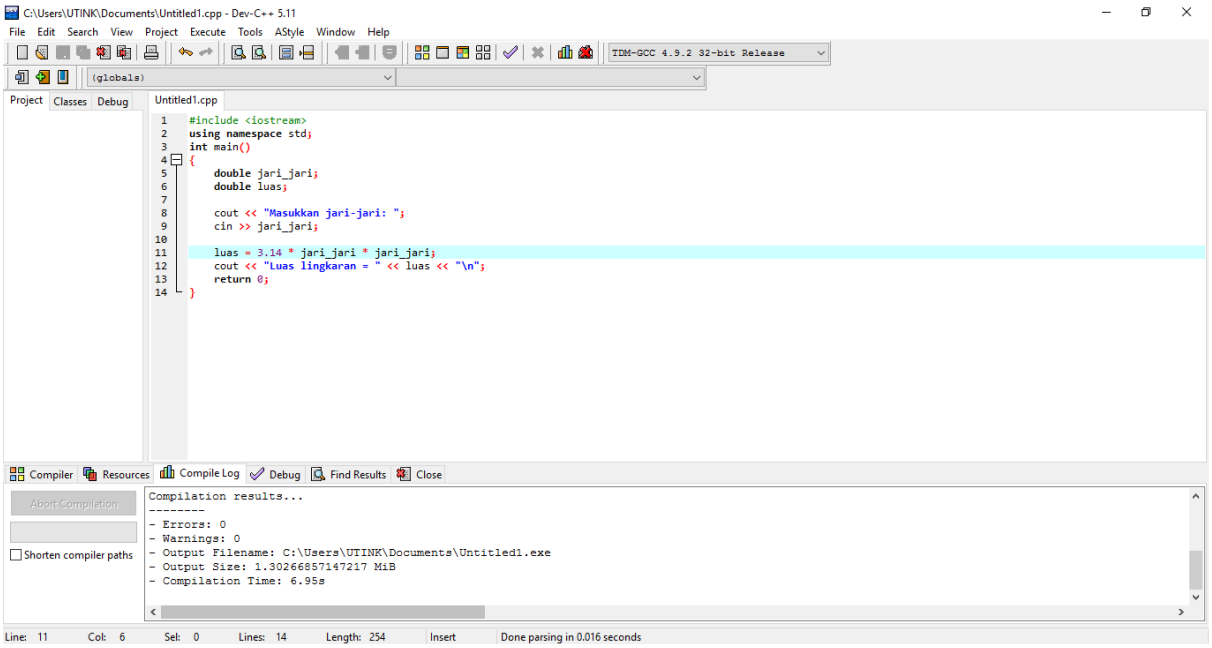
Bahasa	Pernyataan
BASIC	keliling = 2 * (panjang + lebar)
C dan C++	keliling = 2 * (panjang + lebar);
COBOL	COMPUTE keliling = 2 * (panjang + lebar)
Pascal	keliling :=2 * (panjang + lebar)

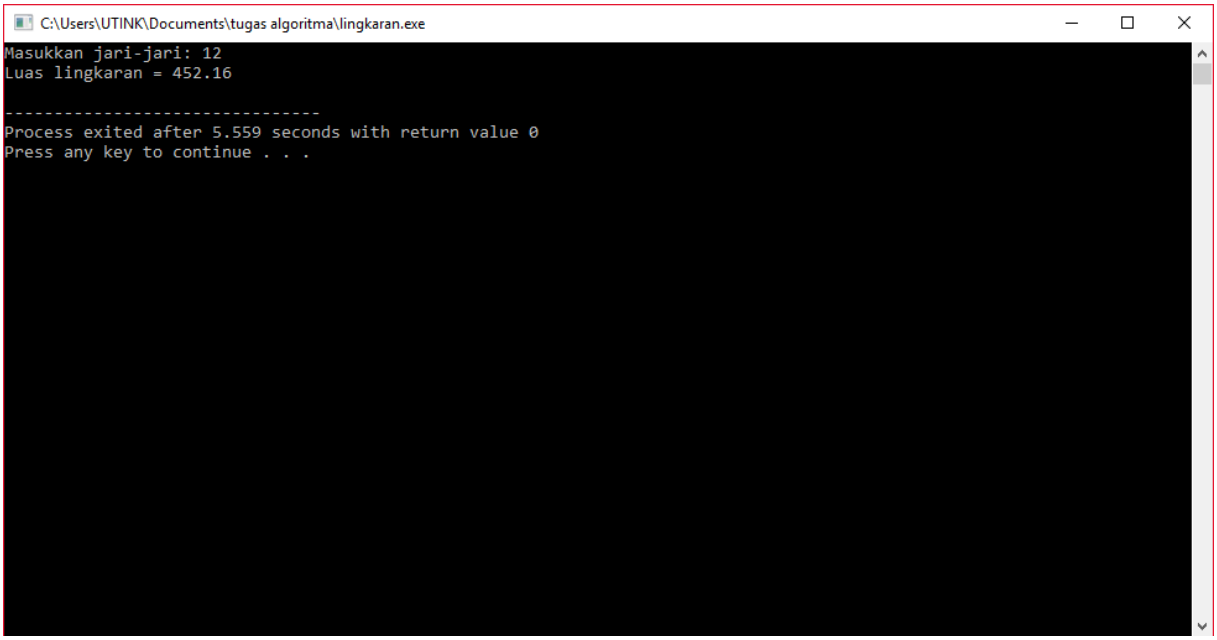
Dengan kata lain, untuk menuangkan algoritma ke program, pemrogram harus tahu seluk-beluk bahasa pemrograman yang di gunakan. Contoh berikut menunjukan program C untuk mengimplementasikan algoritma untuk menghitung luas lingkaran:

Kode Sumber *C++* : **luasling.cpp**

```
#include <iostream.h>
int main()
{
	double jari_jari;
	double luas;

	cout << "Masukkan jari-jari: ";
	cin >> jari_jari;

	luas = 3.14 * jari_jari * jari_jari;

	cout << "Luas lingkaran = " << luas << "\n";

	return 0;
}
```

Tampilan pada Dev C++





	Akhir Kode Sumber

1.3.3 Mengeksekusi dan Menguji Program

Setelah program dibuat dan dikompilasi, program perlu di jalankan untuk di uji kebenarannya. Ada beberapa kemungkinan kesalahan yang terjadi sewaktu proses kompilasi hingga pengeksekusian program:

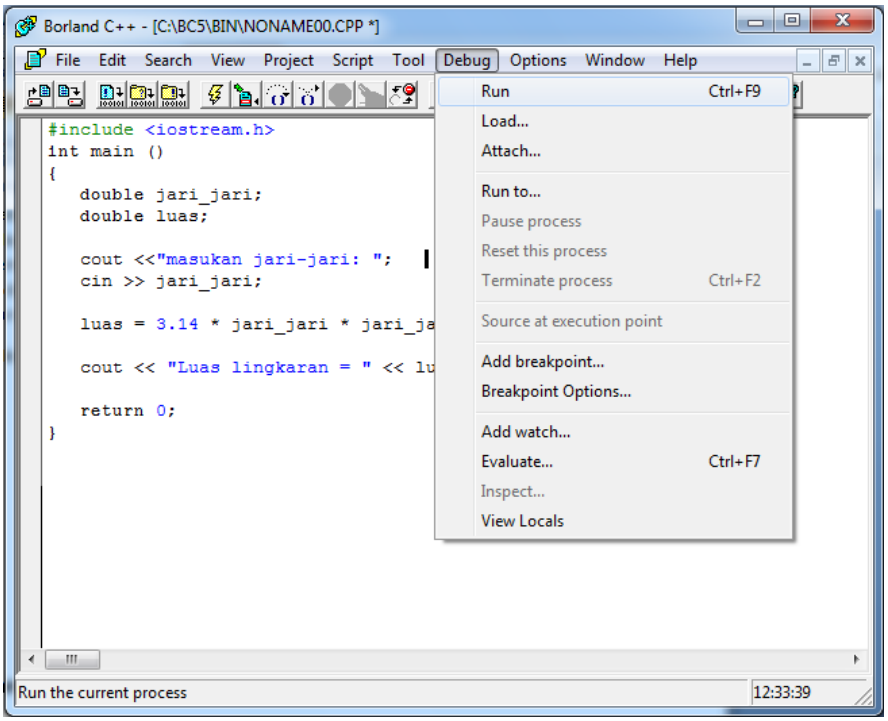
1. Kesalahan sintaksis,
2. Kesalahan Logika
3. Kesalahan *runtime*

Kesalahan sintaksis disebabkan adanya kesalahan dalam menuliskan program yang tidak sesuai dengan kaidah bahasa pemrograman. Contoh, suatu pernyataan C ataupun C++ tidak diakhiri dengan titik-koma, sementara kaidah bahasa C dan C++ mangharuskan setiap pernyataan diakhiri dengan titik-koma. Kesalahan sintaksis terdeteksi sewaktu kompilasi bila Anda menggunakan kompiler C atau C++, tetapi dapat terjadi terdeteksi pada saat eksekusi pada bahasa pemrograman yang berorientasi pada interpreter (misalnya pada interpreter BASIC).

Kesalahan logika adalah kesalahan yang terjadi karena logika yang salah. Misalnya, Anda menuliskan 31.4 untuk menyatakan *phi*, padahal yang betul adalah 3.14. Akibatnya, hasil yang di dapatkan tidak benar. Keslahan seperti ini terkadang sulit untuk di deteksi terutama kalau program sangat kompleks.

Kesalahan runtime atau terkadang di sebut kesalahan fatal adalah kesalahan yang terjadi karena suatu operasi dalam program tidak dapat dilakukan oleh komputer. Sebagai contoh, jika terdapat operasi pembagian dengan nol, maka suatu pesan kesalahan akan ditampilkan dan eksekusi program dihentikan.

Di dalam terminologi program tersebut istilah yang dinamakan bug (kutu) yang menyatakan suatu kesalahan pada program. Pencarian kutu sering kali sulit dilakukan dan memakan waktu. Ibarat mencari jarum dalam tumpukan jerami. Untuk menangani hal seperti itu pemrogram memanfaatkan piranti yang dinamakan *debugger*, yaitu perangkat lunak yang ditujukan untuk mempermudah dalam mencari kesalahan (kutu) dalam program. Adapun proses untuk mencari kesalahan dan membetulkannya biasanya disebut *debugging*.



Gambar 1.9*Contoh debugger pada Visual Basic.Bisa digunakan untuk menjalankan pernyataan langkah per langkah*

BAB 2
STRUKTUR DASAR ALGORITMA

2.1 Macam Struktur Dasar Algoritma

Pada dasarnya terdapat tiga buah struktrurdasar yang digunakan dalam menyusun suatu algoritma. Ketiga struktur dasar tersebut adalah:

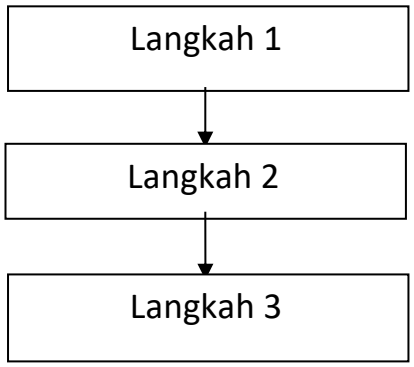
1. Sekuensi (algoritma),
2. Seleksi,
3. Pengulangan

Algoritma yang kompleks umumnya tersusun atas struktur-struktur dasar tersebut.

2.2 Struktur Sekuensi

Pada struktur sekuensial, langkah-langkah yang dilakukan dalam algoritma diproses secara berurutan sebagaimana diperlihatkan pada Gambar 2.1.

Langkah 1
Langkah 2
Langkah 3



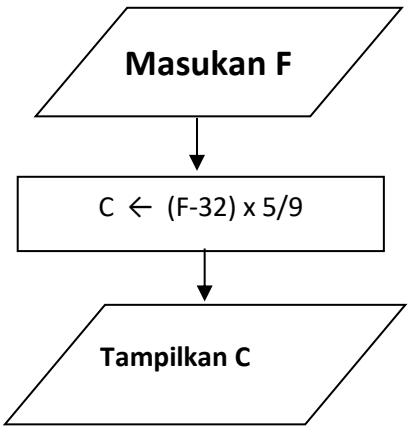
Gambar 2.1*Struktur sekuensial*

Pada contoh di atas, langkah 1 merupakan langkah yang akan dijalankan pertama kali, setelah itu lankah 2 dikerjakan dan diikuti langkah 3**.**

Contoh 2.1

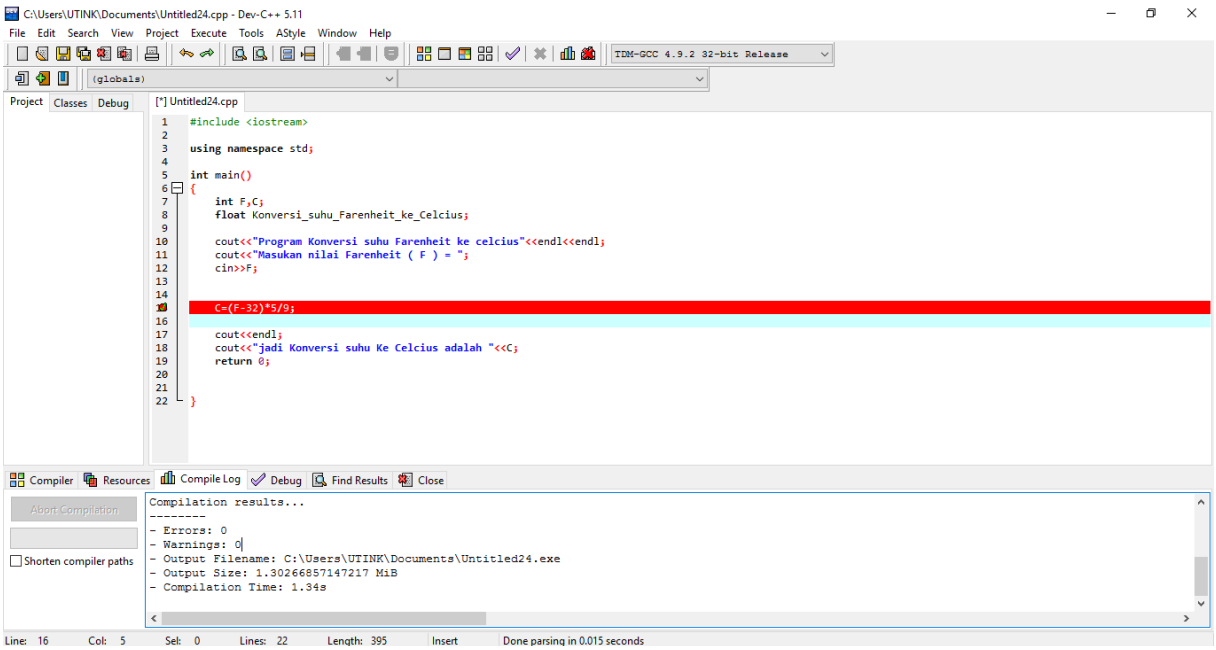
Buatlah algoritma untuk mengonversi dari suhu fahrenheit ke celciu**s.**

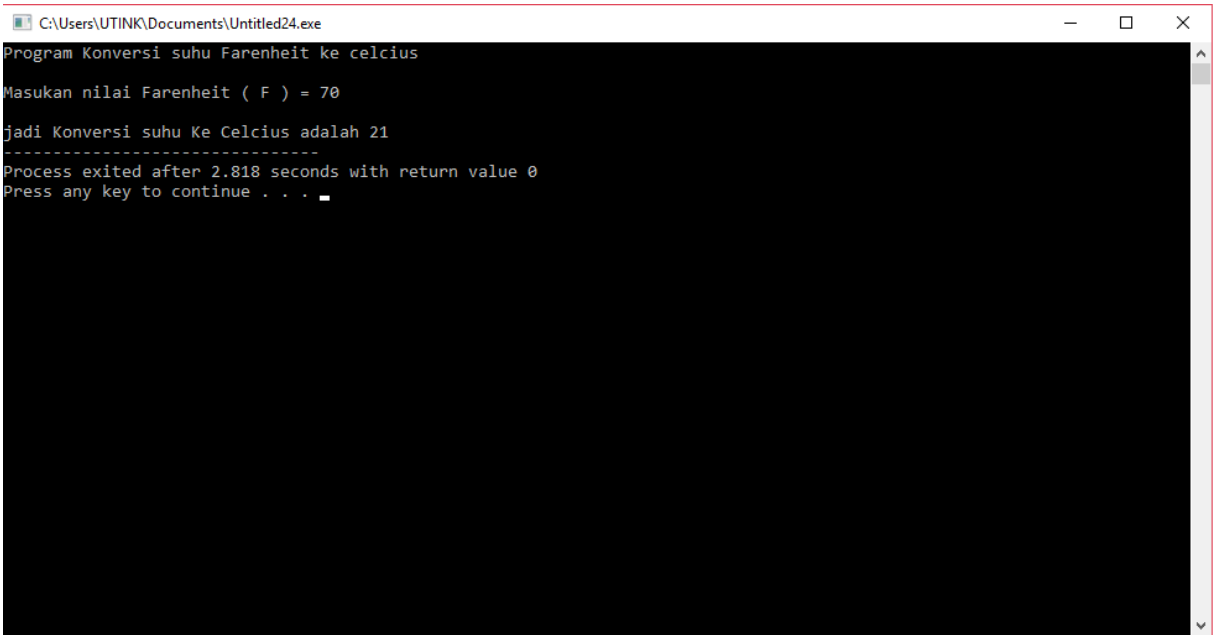
Solusi : Jika dinyatakan diagram air, terlihat dengan jelas bahwa penyelesaian masalah ini cukup dilakukan dengan menggunakan struktur sekuensial, sebagaimana diperlihatkan pada Gambar 2.2
Tentu saja Anda harus memiliki pengetahuan tentang bagaimana cara mengonversi dari fahrenheit ke celcius.



Gambar 2.2 *diagram alir konversi fahrehait ke celcius*

Tampilan pada Dev C++

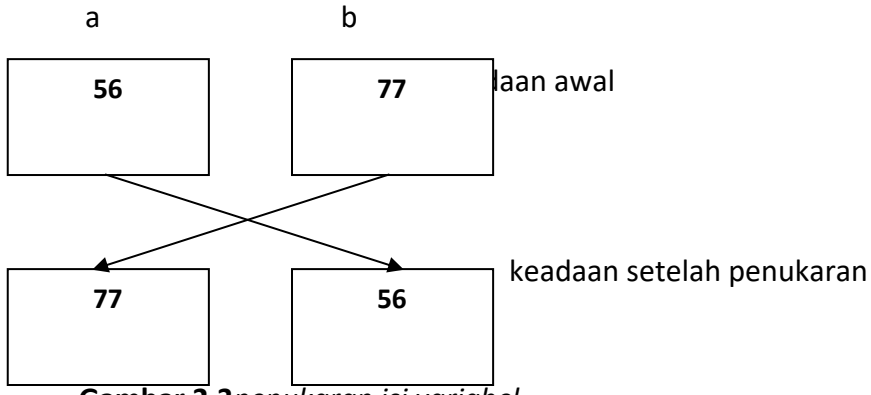




Contoh 2.2

Buatlah algoritma untuk menukarkan isi dua buah variabel**.**

Solusi: Gambar 2.3 melukiskan pengertian menukarkan sisi dua buah variabel.



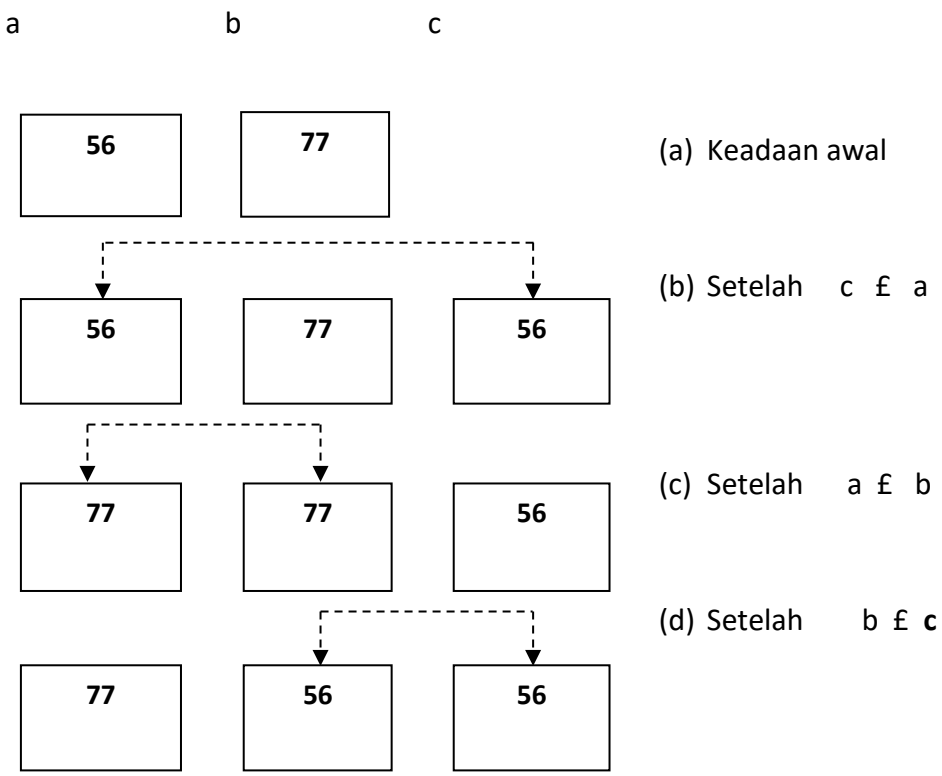
Gambar 2.3*penukaran isi variabel*

Untuk menukarkan isi dua buah variabel diperlukan sebuah variabel yang digunakan untuk membantu penukaran data.

Algoritma yang diperlukan:

1. Masukkan (a,b)
2. c ← a
3. a ← b
4. b ← c
5. Tampilkan (a,b)

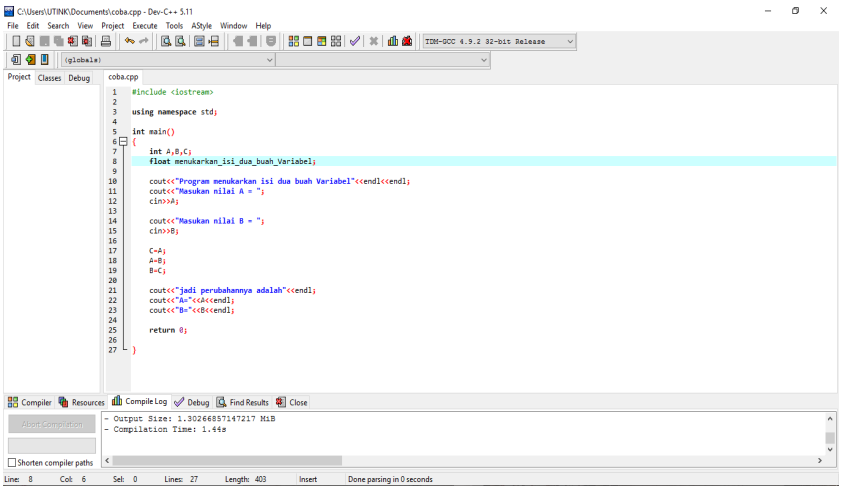
Gambar 2.4 *menggambarkan proses penukaran berdasarkan algoritma di atas.*

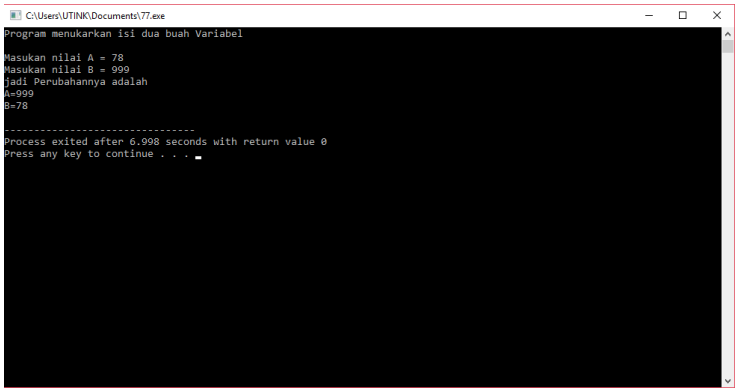


Gambar 2.4 *Ilustarai penukaran isi dua buah variabel*

Tampak bahwa setelah proses b ← c dijalankan, isi a dan b sudah saling ditukarkan.

Tampilan pada Dev C++





2.3 Struktur Seleksi

Struktur seleksi menyatakan pemilihan langkah yang didasarkan oleh suatu kondisi (pengambilan keputusan). **Gambar 2.5**memperlihatkan diagram air struktur seleksi yang melibatkan dua alternatif. Dalam hal ini simbol belah ketupat digunakan untuk mewakili langkah pengambilan keputusan.

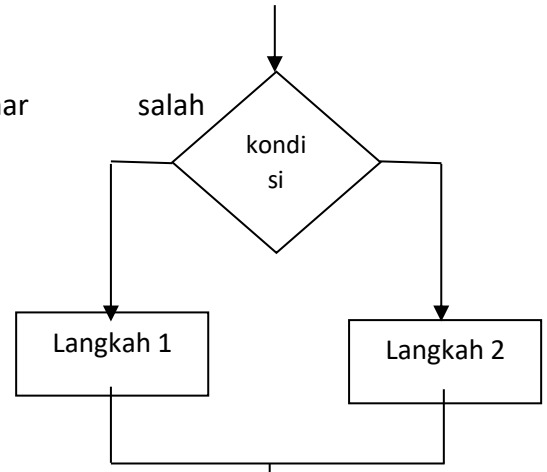
JIKA kondisi benar **MAKA**

Langkah 1

SEBALIKNYA

Langkah 2

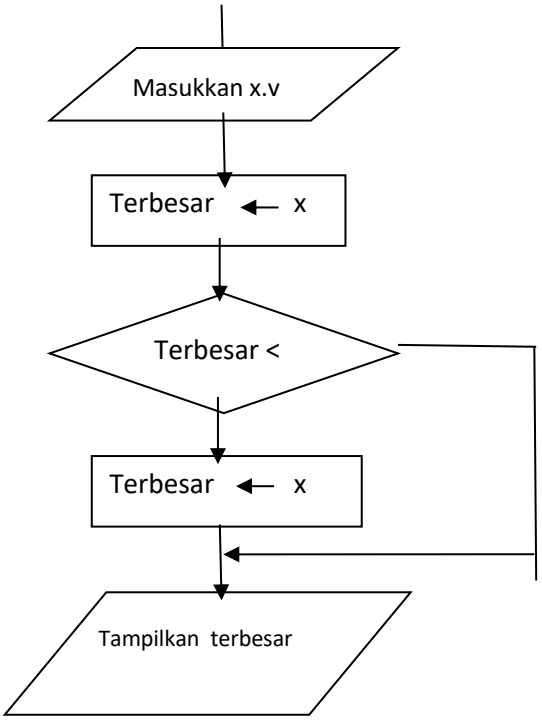
AKHIR-JIKA



Pada struktur di atas, langkah 1 hanya akan dijalankan kalau *kondisi* bernilai benar, sedangkan langkah 2 hanya akan dijalankan kalau *kondisi* bernilai salah.

Bentuk yang lebih kompleks dalam pengambilan keputusan pada dasarnya dapat dikembangkanmelalui struktur seleksi.

Contoh 2.3
Buatlah algoritma untuk menentukan bilangan terbesar dari dua buah bilangan x dan y.
Solusi: pemecahan masalah ini diperlihatkan pada gambar 2.6.

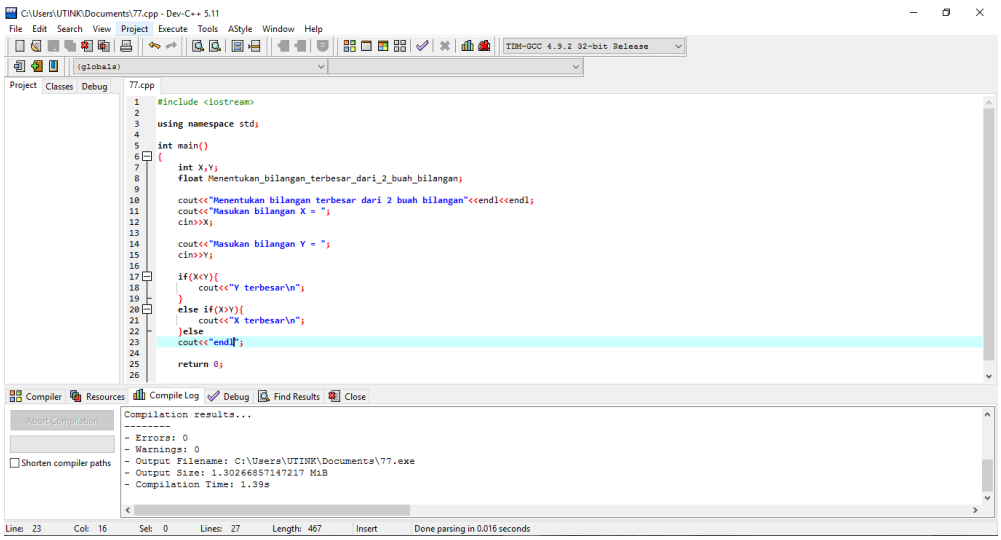


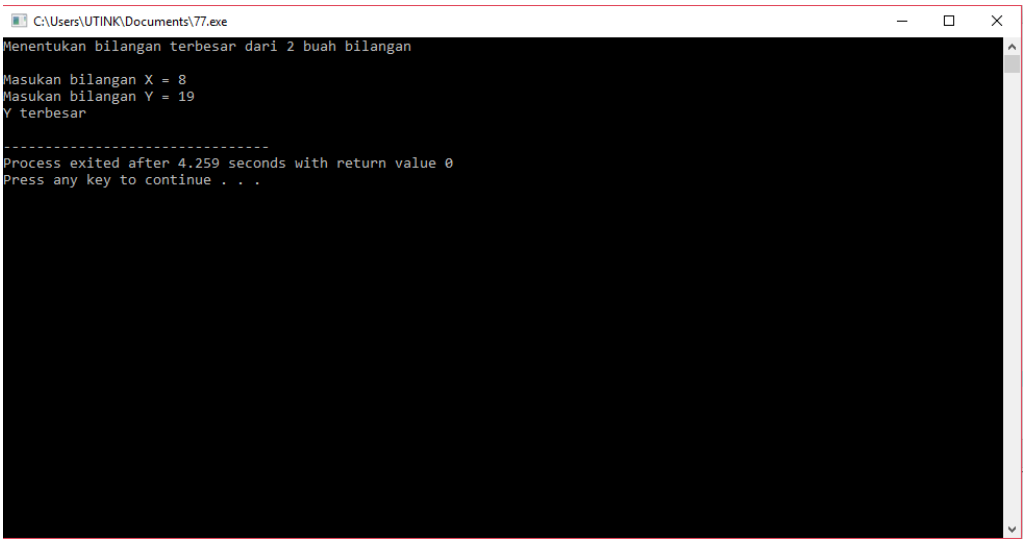
Gambar 2.6 *diagram alir menentukan bilangan terbesar di antara dua buah bilangan.*

Penyelesaian dalam bentuk *pseudokode* dapat dilihat di bawah ini:

1. Masukan (x.y)
2. Terbesar ← x // asumsi bahwa x adalah yang terbesar.
3. JIKA terbesar < y MAKA
   terbesar ← y
   AKHIR-JIKA
4. Tampilkan (terbesar)

Tampilan pada Dev C++





Sebagai contoh, anggap x bernilai 8 dan y bernilai 19.

- Berdasarkan langkah kedua, terbesardiisi dengan 8.
- Berdasarkan langkah ketiga, mengingat kondisi dari 8 < 19 bernilai benar, maka bagiyan berikut dijalankan terbesar ← 19 dengan dengan demikian, *terbesar* berubah dari 8 menjadi 19.
- Berdasarkan langkah keempat, yang tertampil adalah 19.

Catatan 	Simbol // pada algoritma didepan menyatakan bahwa mulai *simbol* tersebut hingga akhir baris merupakan keterangan untuk pembaca algoritma (atau disebut komentar).

Bagaimana halnya kalau x bernilai 19 dan y bernilai 8?

- Berdasarkan langkah ke dua, *terbesar* diisi dengan 19.
- Berdasarkan langkah ketiga, mengingat kondisi 19 < 8 bernilai salah, maka bagiyan berikut tidak dijalankan terbesar ← y. Dengan demikian, *terbesar* tetap bernilai 9.
- Berdasarkan langkah keempat, yangtertampil adalah 19.

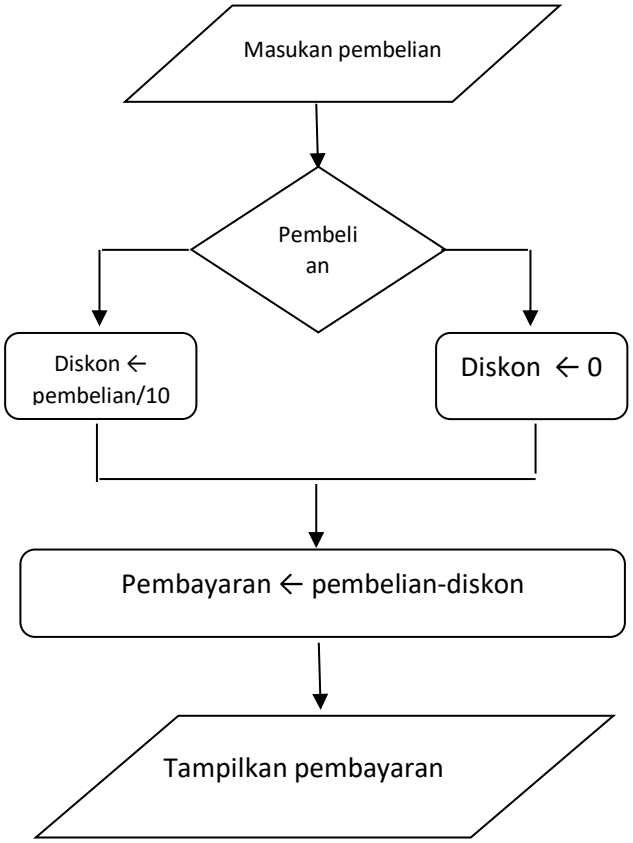
Contoh 2.4

Suatu swalayan memberikan diskon sebesar 10% bagi siapa saja yang berbelanja sebesar 100.000 atau lebih. Buatlah algoritma untuk menghitung nilai uang yang harus dibayar oleh pembeli.

Solusi: pemecahan masalah ini dalam bentuk diagram *alir* dan *pseudekode* dapat dilihat pada **gambar** berikiut.

1. Masukan (pembilan)
2. JIKA pembelian >= 100.000 MAKA
   Diskon ←0,1 x pembelian
   SEBALIKNYA
   diskon ←0

   AKHIR-JIKA
3. Pembayaran ← pembelian-diskon
4. Tampilkan (pembayaran)



gambar 2.7 *Diagram alir dan peseudekode* penentuan pembayaran menurut pembelian dan diskon.

2.4 Struktur Pengulangan

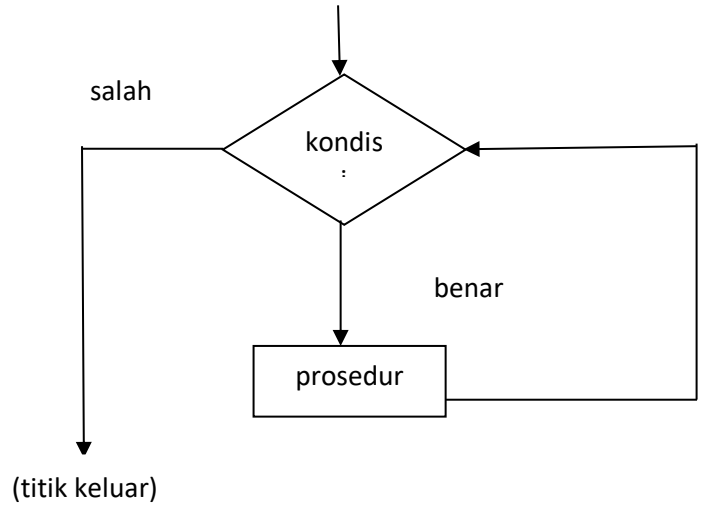
Pengulangan menyatakn suatu tindakan atau langkah yang dijalankan beberapa kali.Nah, struktur pengulangan menyatakan perwujudan keadaan seperti itu. Sebagai contoh, jika Anda ingin menampilkan 10 tulisan“selamat belajar”, Anda dapat menuliskannya dengan menggunakan struktur sekuensial. Itu berarti bahwa Anda memberikan 10 intruksi untuk menuliskan kesepuluh tulisan tersebut. Cara seperti itu memang peraktis untuk jumlah pengulangan yang sedikit (misalnya 2 atau 3 buah pengulangan), tetapi tidak cocok untuk jumlah pengulangan yang besar. Supaya lebih praktis, Anda dapat menggunakan struktur pengulangan.

Struktur pengulangan diilustrasikan pada Gambar 2.8

ULANGAN SELAMA kondisi benar

Prosedur

AKHIR-ULANG



Gambar 2.8 *Giagram alir untuk menggambarkan pengulangan dan pseudokodenya.*

Pada struktur diatas, *prosedur* dapat berupa satu atau berapah langkah.

Diagram alir pada gambar 2.8 tersebut menunjukkan bahwa sebelum *prosedur* dijalankan pertama kali, kondisi diuji terlebih dulu. Sekitar *kondisi* bernilai benar maka *prosedur* dijalankan dan kemudian kondisi diuji lagi. Sepanjang kondisi masih bernilai benar, prosedur masih akan dijalankan. Namun begitu *kondisi* bernilai salah maka pengulangan akan berakhir.

Contoh 2.5

Buatlah algoritma untuk menampilkan 6 buah tulisan “selamat belajar” dengan menggunakan pengulangan.

Solusi : solusi dalam bentuk digram alir dan pseudekode dapat dilihat pada gambar berikut:

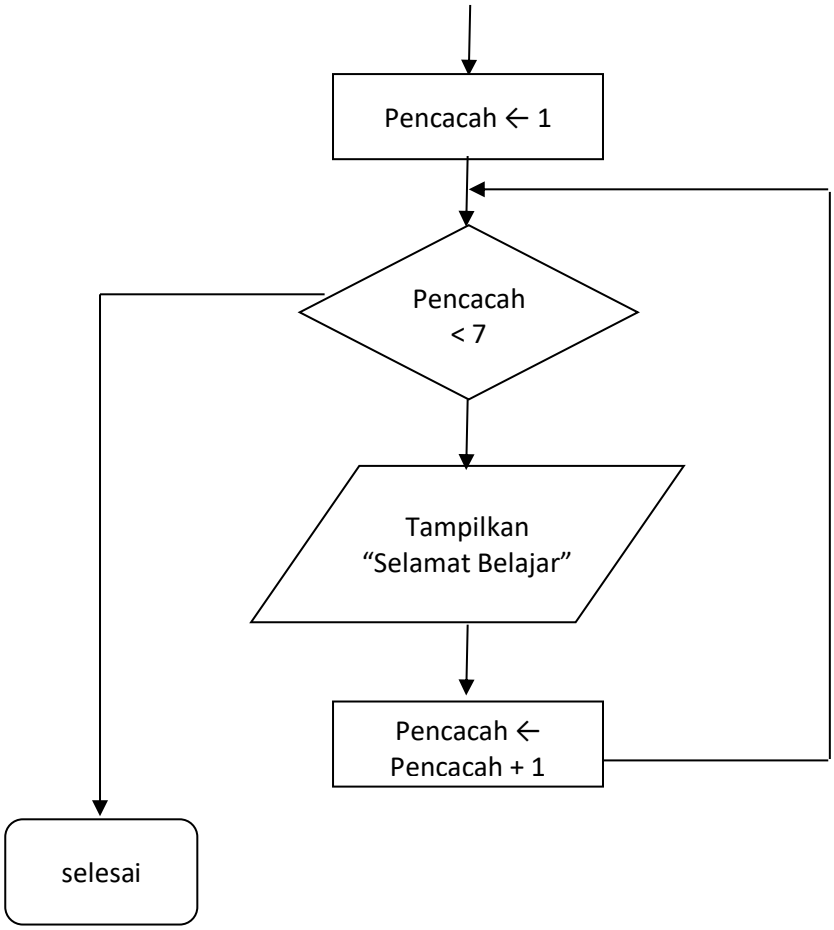
Pencacah ← 1

ULANG SELAMA pencacah <7

tampilkan (“selamat belajar”)

Pencacah ← Pencacah + 1

AKHIR-ULANG



Gambar 2.9 *Diagram alir dan pseudekode untuk menampilkan 6 buah tulisan “selamat belajar”*

Contoh 2.5

Buatlah algoritma untuk membuat tabel seperti berikut dengan menggunakan struktur pengulangan:

1. 1
2. 4
3. 9
4. 16
5. 25
6. 36
7. 49
8. 64
9. 81
10. 100

Solusi: Pembuatan tabel diatas dalam bentuk diagram alir dan pseudokode dapat dilihat pada gambar berikut:

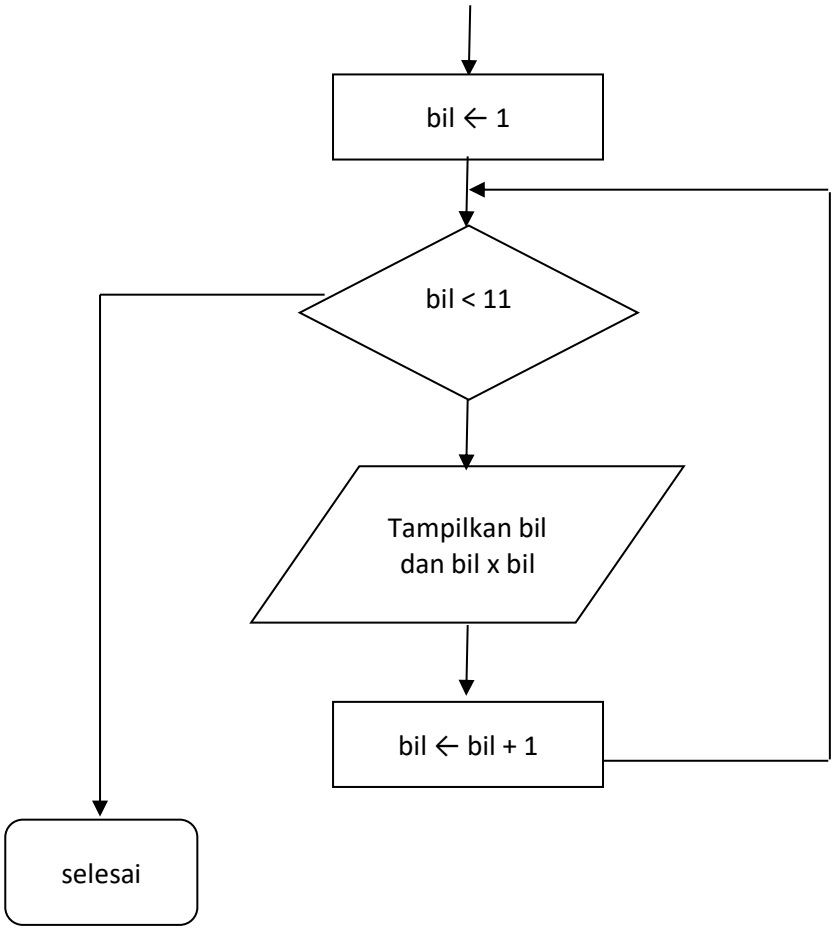
bil ← 1

ULANG SELAMA bil < 11

tampilkan (bil, bil x bil)

bil ← bil + 1

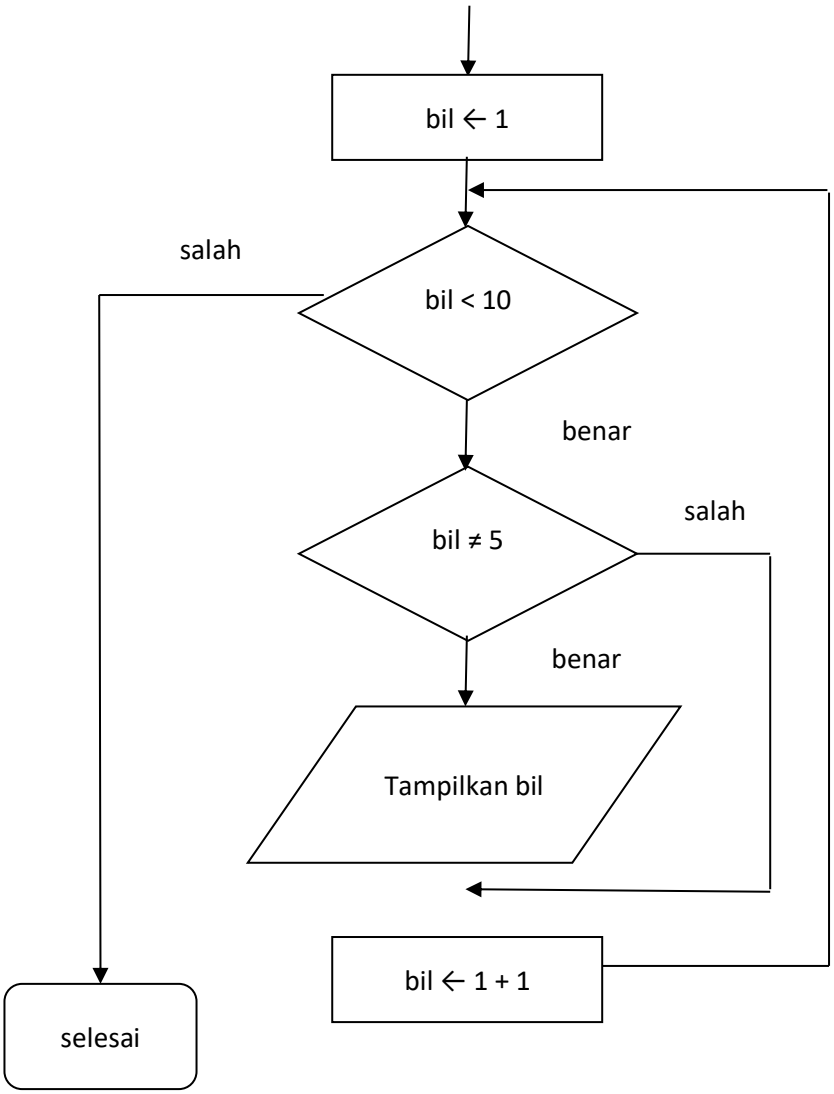
AKHIR-ULANG



Gambar 2.10*Diagram alir dan pseudokode untuk membuat tabel*

2.5 Kombinasi Struktur Dasar

Dalam praktik, banyak algoritma yang mengombinasikan dua atau tiga struktur dasar yang telah dibahas. Sebagai contoh, solusi pada Gambar 2.11 melibatkan struktur dasar sekuensial dan pengulangan. Adapun diagram alir berikut menunjukkan penggabungan antara sekuensial, pengulangan, dan seleksi.



Gambar 2.11*Contoh algoritma yang menggunakan struktur sekuensial, seleksi, dan pengulangan*

Jika dituangkan dalam bentuk algoritma, penulisannya adalah seperti berikut :

1. bil ← 1
2. ULANG SELAMA bil < 10
   JIKA bil ≠ 5 MAKA
   tampilkan(bil)

   AKHIR-JIKA

   bil ← bil + 1

   AKHIR-ULANG

Menurut anda, apa hasil algoritma di atas ? Jawabannya adalah :

1
2
3
4
6
7
8
9

Angka 5 tidak ditampilkan mengingat:

```
JIKA bil ≠ 5 MAKA
        Tampilkan(bil)
AKHIR-JIKA
```

Memang menyatakan bahwa: “jika *bil* tidak sama dengan 5, tampilkan bil”. Atau dengan kata lain, jika *bil* sama dengan 5 maka bilangan tersebut tidak ditampilkan..

BAB 3

PEDOMAN PENYUSUNAN ALGORITMA

3.1 Algoritma dalam Kehidupan Sehari-hari

Proses semacam algoritma sebenarnya dapan dijumpai dalam kehidupan sehari-hari. Jika Anda membaca resep makanan, Anda akan melihat prosedur untuk membuat makanan, selain bahan-bahan yang digunakan. Prosedur dalam resep seperti itu sebenarnya menyatakan semacam algoritma. Dengan mengikuti langkah-langkah yang diberikan, Anda dapat membuat masakan tersebut. Dalam perngertian sekarang, algoritma mempunyai kesamaan tidak hanya dengan resep (masakan), tetapi juga dengan proses , metode, atau prosedur. Sebagai contoh, Gambar 3.1 memperlihatkan suatu prosedur untuk memanaskan makanan kedalam pemasak *microwave*.

1. Taruhlah makanan dalam wadah yang aman untuk *microwave*.
2. Tutuplah pintu *microwave* dengan rapat.
3. Tancapkan steker ke stop kontak.
4. Putarlah knop ke posisi 5 menit.
5. Tunggu sampai lampu mati dan ada bunyi ‘ting’.
6. Lepaskan steker dari stop kontak.
7. Bukalah wadah pemasak *microwave* dan keluarkan wadah yang berisi makanan tersebut.

Gambar 3.1*Contoh suatu prosedur*

Prosedur tersebut merupakan suatu urutan yang memandu orang untuk melakkan suatu proses. Namun ada perbedaan antara algoritma dan prosedur. Prosedur biasanya bersifat global dengan mengasumsikan bahwa manusia telah mengetahui rincian langkah-langkan tertentu. Sebagai contoh, perhatikan prosedur berikut :

1. Ambil 5 buah kartu.
2. Urutkan kartu tersebut.

Pada langkah kedua, manusia dapat mengurutkan kartu tersebut dengan mudah tanpa harus diberi tahu caranya. Namun hal seperti itu tidak dapat diterapkan pada komputer. Jika komputer mengenal lima buah data dan kita mengharapkan computer untuk mengurutkannya, kita harus memberitahu seara detail bagaimana cara mengurutkannya. Itulah sebabnya di dalam ilmu computer terdapat berbagai algoritma yang ditujukan ntuk mengurutkan data, seperti *Bubble sort*, *Insertion sort*, dan *Quick sort*. Algoritma-algoritma ini menjamin bahwa setiap langkah tersebut dikerjakan oleh komputer.

Gambar 3.2 *Mengurutkan kartu memerlukan suatu prosedur*

3.2 Dasar Penyusunan Algoritma

Sejauh ini tidak ada standarisasi tentang bagaimana menyusun algoritma. Secara prinsip, Anda mempunyai kebebasan untuk menyusun bentuk algoritma. Anda dapat menggunakan kata-kata dalam bahasa manusia, pseudokode, atau bahkan diagram alir untuk mewujudkan suatu algoritma. Anda juga dapat menggunakan cara-cara anda sendiri untuk menuliskansuatu algoritma dengan memegang teguh konsistensi.

Walaupun begitu ada beberapa hal yang perlu diperhatikan dalam menyusun algoritma. Menurut Knuth (1973, hal. 4) dan juga Horowitz (1999, hal. 1), ada lima ciri-ciri penting yang harus dimiliki sebuah algoritma, yaitu berupa *finiteness, definiteness,* masukan, keluaran, dan efektivitas.

1. **Finiteness**, menyatakan bahwa suatu algoritma harus berakhir untuk semua kondisi setelah memproses sejumlah langkah.
2. **Definiteness**, menyatakan bahwa setiap langkah harus dinyatakan dengan jelas (tidak rancu atau mendua-arti)
3. **Masukan**. Setiap algoritma dapat tidak memiliki masukan atau mempunyai satu atau beberapa masukan. Masukan merupakan suatu besaran yang diberikan di awal sebelum algoritma diproses.
4. **Keluaran**. Setiap algoritma memiliki keluaranm entah hanya sebuah keluaran atau benyak keluaran. Keluaran merupakan besaran yang mempunyai kaitan atau hubungan dengan masukan.
5. **Efektivitas**. Setiap algoritma diharapkan bersifat efektif, dalam arti semua operasi yang dilaksanakan oleh algoritma harus sederhana dan dapat dikerjakan dalam waktu yang terbatas. Secara prinsip, setiap instruksi dalam algoritma dapat dikerjakan oleh orang dengan hanya menggunakan kertas dan pensil.

Menurut Cormen (1994, hal. 2), sebuah algoritma dikatakan benar, untuk berbagai ragam masukan, jika algoritma berakhir dengan keluaran yang benar. Pada keadaan seperti ini algoritma menyelesaikan masalah komputasi yang diberikan.

Selain ketentuan-ketentuan yang telah disebutkan,patut pula diletakkan disini untuk selalu menghindari langkah yang merujuk kelangkah lain. Contoh perujukan ke langkah lain ditunjukkan pada contoh berikut, yang membahas algoritma Euclid yang dijelaskan oleh Knuth (1973, hal. 2).

Algoritma E (Algoritma Euclid)

Terdapat dua buah bilangan bulat m dan n. Carilah bilangan Faktor Persekutuan Terbesar (*Greatest Common Divisor*), yakni bilangan positif terbesar yang menjadi pembagi baik m maupun n.

E. 1 [Cari sisa Pembagian] Bagi m dengan n dan berikan sisa pembagiannya ke r.

E. 2 [Apakah r bernilai nol?] Jika r = 0, algoritma berakhir, n adalah jawabannya.

E. 3 [Tukarkan] Aturlah m ← n, n ← r, dan kembali ke E.1.

Perlu diketahui, notasi [ ] yang ada pada algoritma di atas merupakan keterangan yang diberikan oleh Knuth.

Algoritma seperti itu ditujukan untuk bahasa-bahasa pemrograman yang mendukung instruksi loncatan seperti Goto pada BASIC ataupun Pascal. Namun dalam bahasab pemrograman masa sekarang, perimtah sepertiitu sudah tidak disediakan lagi atau dianjurkan untuk tidak digunakan lagi. Oleh karena itu algoritma di depan dapat ditulis ulang menjadi seperti berikut dengan mengeliminasi perujukan ke langkah yang lain :

1. r ← Sisa pembagian m dengan n
2. ULANG SELAMA r ≠ 0

   m ← n

   n ← r

   r ← Sisa pembagian m dengan n
3. AKHIR ULANG
4. tampilkan(n)

Contoh : Berapa Faktor Persekutuan Terbesar bilangan 60 dan 24?

m = 60

n = 14

Langkah pemrosesan algoritma :

1. Mula-mula r diisi dengan sisa pembagian 60 dengan 24.Dalam hal ini, r akan berisi 12.
2. Mengingat r ≠ 0 maka pernyataan dalam ULANG…AKHIR-ULANG dikerjakan :
   - m diisi dengan 12 (m ← n)
   - n diisi dengan 12 (n ← r)
3. Mengingat r = 0 maka pengulangan berakhir.

Dengan demikian, n yang berisi 12 merupakan Faktor Persekutuan Terbesar bilang 60 dan 24.

3.3 Pedoman Menyusun Pseudokode

Sejauh ini Anda telah mengenal sejumlah contoh pseudokode. Namun demikian anda belum mengenal seluruh aturan yang berlaku untuk menyusun

pseudokode. Berikut adalah pedoman yang digunakan dalam seluruh pembahasan dalam buku ini :

1. Notasi ←dipakai untuk memberikan nilai ke suatu variable. Contoh :

   bil ← 0

   digunakan untuk memberikan nilai 0 ke variabel`bil`.

2. Setiap pernyatan atau suatu perintah yang dapat berdiri sendiri akan ditulis dalam sebuah baris tersendiri. Contoh :

   bil ← 1 + 2

   merupakan contoh pernyataan untuk menugaskan atau memasukkan hasil pernjumlahan bilangan 1 dan 2 ke dalam variable bil. Pernyataan seperti itu dikenal dengan sebutan pernyataan penugasan.

3. Setiap variabel (namayang digunakan untuk menyimpan data dan datanya dapat diubah-udah) akan ditulis dengan awalan huruf. Awalan huruf kecil akan digunakan untuk menyatakan sebuah variabel non-larik, sedangkan awalan huruf kapital digunakan untuk menyatakan larik (suatu variabel yang dapat digunakan untuk menyimpan sejumah data yang sejenis). Contoh :

   - `bil` berarti variabel non-larik
   - A berarti variabel larik

   Tipe variabel tidak pernah disebutkan secara eksplisit. Dalam beberapa kasus, tipe data akan dijelaskan melalui komentar (Pengertian komentar dapat dilihat pada butir 6).

4. Tipe data majemuk atau disebut tipe rekaman (tipe data yangdapat mengandung beberapa data dengan nama yang berbeda-beda) akan dinyatakan dengan notasi seperti berikut :

   simpul = REKAMAN

   data1

   data2

   data3

   AKHIR-REKAMAN

   Untuk menyatakan *data1* yang terdapat pada *simpul*, maka digunakan notasi simpul → data1.

5. Identitas atau penjorokan ke kanan pada digunakan untuk menuliskan pernyataan-pernyataan yang berada dalam suatu struktur blok.

   Contoh :

   JIKA x > 1 MAKA

   Pernyataan-1

   Pernyataan-2

   Pernyataan-3

   AKHIR-JIKA

   Pada contoh di atas, *Pernyataan-1*, *Pernyataan-2*, dan *Pernyataan-3* menyatakan dalam sebuah blok JIKA…AKHIR-JIKA.

6. Symbol // digunakan untuk menyatakan komentar. Komentar adalah keterangan yang ditujukan untuk pembaca algoritma; tidak ditujukan untuk proses oleh computer. Semua karakter dimulai dari simbol tersebut hingga akhir deperlakukan sebagai komentar. Contoh :

   // A adalah latik dan n adalah jumlah elemen dalam larik

   bil ← bil + 1 // Isi bil dinaikkan sebesar satu

   Pada contoh kedua, bil ← bil + 1 tetap menyatakan senuah pernyataan, tetapi sisanya dalam baris berkedudukan sebagai komentar.

7. Notasi masukkan () dan tampilkan () secara berurutan mewakili perintah untuk memperoleh masukan dan menyajikan keluaran. Contoh :

   masukan(panjang, lebar)

   tampilkan(luas)

   Pada contoh pertama, masukan yang diperlukan berupa *panjang* dan *lebar*. Pada contoh kedua, yang ditampilkan adalah nilai *luas*.

8. Ada dua nilai logika uang digunakan, yaitu BENAR dan SALAH. Nilai ini juga dihasilkan oleh pembanding yang menggunakan tanda seperti <, ≤, >, ≥, =, ≠, yang secara berturut-turut menyatakan kurang dari, kurang dari atau sama dengan, lebih dari, lebih dari atau sama dengan, sama dengan, dan tidak sama dengan. Selain itu operator DAN, ATAU, dan TIDAK dapat digunakan untuk membentuk ekspresi yang menghasilkan nilai BENAR atau SALAH. Logika yang dipakai untuk operasi DAN dan ATAU dapat dilihat pada tabel berikut :

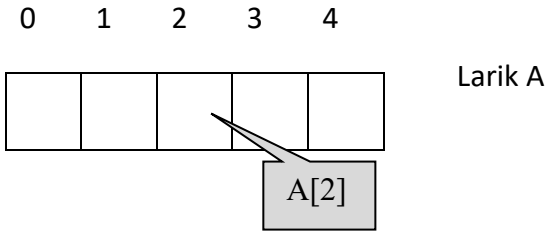
Tabel 3.1*Operasi dengan DAN dan ATAU*

a	b	a DAN b	a ATAU b
SALAH	SALAH	SALAH	SALAH
BENAR	SALAH	SALAH	BENAR
SALAH	BENAR	SALAH	BENAR
BENAR	BENAR	BENAR	BENAR

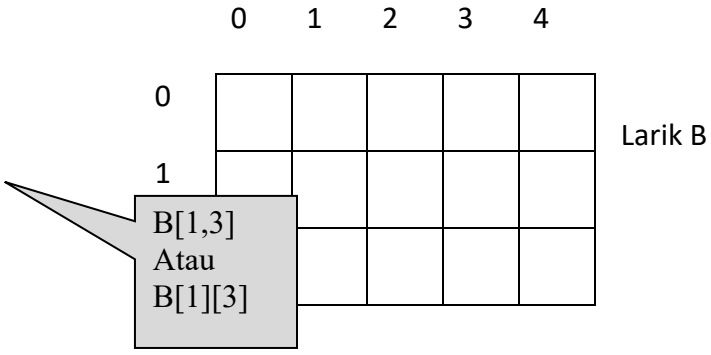
9. Notasi seperti A[i] menyatakan elemen ke-i pada larik A. Nilai terkecil untuk i adalah nol. Untuk larik berdimensi dua, sebuah elemen akan dinotasikan dengan A[i,j] atau A [i][j] dengan i menyatakan indeks untuk baris dan j untuk kolom. Contoh :

   terbesar ← A[0]

   merupakan pernyataan untuk memasukkan nilai elemen terkiri dalam larik A ke variabel `terbesar`.



(a) Larik berdimensi satu



(b) Larik berdimensi dua

Gambar 3.3*Pengaksesan elemen larik*

10. Notasi panjang(A) menyatakan ekspresi untuk memperoleh jumlah elemen pada A.
11. Bentuk

    JIKA *kondisi* MAKA

        Pernyataan-1

        …

        Pernyataan-2

    SEBALIKNYA

        Pernyataan-a

        …

        Pernyataan-b

    AKHIR-JIKA

    menyatakan model penulisan untuk menangani struktur seleksi. Dalam hal ini *Pernyataan-1* hingga *Pernyataan-2* akan dijalankan hanya kalau *kondisi* bernilai BENAR. Adapun kalau *kondisi* bernilai SALAH makan *Pernyataan-a* hingga *Pernyataan-b* akan dijalankan. Dalam hal ini bagian SEBALIKNYA bersifat opsional (bisa saja tidak ada).
12. Bentuk

    COCOK nilai

        DENGANnilai1 MAKA

                Pernyataan-11

                Pernyataan-12

        DENGAN nilai2 MAKA

                Pernyataan-21

                Pernyataan-22

        DENGAN nilai3 MAKA

                Pernyataan-31

                Pernyataan-32

        LAINNYA

                Pernyataan-n1

                Pernyataan-n2

        AKHIR-COCOK

digunakan untuk menangani seleksi dengan beberapa kemungkinan. Pada bentuk diatas,

Pernyataan-n1

Pernyataan-n2

Hanya akan dijalankan kalau *nilai* tidak cocok dengan *nilai1*, *nilai2,* maupun *nilai3*.

13. Bentuk

ULANG SELAMA *kondisi*

Pernyataan-1

…

Pernyataan-2

AKHIR-ULANG

merupakan cara menuliskan pengulangan. Dalam hal ini bagian *Pernyataan-1* hingga *Pernyataan-2* akan dijalankan secara terus-menerus selama kondisi bernilai BENAR. Begitu kondisi bernilai SALAH, pengulangan berakhir.

14. Bentuk

UNTUK variabel ← awal S/D akhir LANGKAH *langkah*

Pernyataan-1

…

Pernyataan-2

AKHIR-UNTUK

merupakan suatu bentuk pengulangan terhadap *Pernyataan-1* hingga *Pernyataan-2* yang dimulai dari *variabel* bernilai *awal* hingga variabel bernilai tidak lebih dari *akhir*. Klausa LANGKAH menentukan kenaikan terhadap nilai variabel untuk setiap iterasi berikutnya. Bagian ini bersifat opsional. Kalu tidak disebutkan, kenaikan nterhadap *variabel* adalah sebesar 1. Contoh :

UNTUK bil ← 1 S/D 8

tampilkan(bil)

AKHIR-UNTUK

Hasilnya berupa

1

2

3

4

5

6

7

8

Adapun

UNTUK bil ← 1 S/D 8 LANGKAH 3

    tampilkan(bil)

AKHIR-UNTUK

menghasilkan :

1

4

7

Adapun

UNTUK bil ← 10 S/D 5 LANGKAH -1

    tampilkan(bil)

    AKHIR-UNTUK

memberikan hasil berupa :

10

9

8

7

6

5

15. Suatu subrutin (sejumlah kode yang diberi nama) ditulis dengan bentuk sebagai berikut:

    SUBRUTIN *NamaSubrutin(daftar-parameter)*

        Pernyataan-1

        …

Pernyataan-2

AKHIR-SUBRUTIN

Sebuah subrutin dapat memberikan nilai balik ataupun tidak. Nilai balik adalah nilai yang diberikan ke pemanggilnya. Nilai ini ditentukan melalui notasi seperti berikut :

NILAI-BALIK *nilai*

Contoh :

```
SUBRUTIN maksimum(A, n)
    //A adalah larik dan n adalah jumlah elemen larik
    terbesar ← A[0]
    UNTUK i = 1 S/D n-1
    JIKA terbesar < A[i] MAKA
            terbesar ← A[i]
            AKHIR-JIKA
    AKHIR-UNTUK
    NILAI-BALIK terbesar
AKHIR-SUBRUTIN
```

BAB 4
DASAR PEMROGRAMAN C++

4.1 Mengenal Program C dan C++

Selamat datang di dunia pemrograman. Anda akan mempelajari bahasa C dan C++. C merupakan bahasa komputer yang sangat singkat. Namanya tidak memiliki kepanjangan apapun. Bahasa ini diciptakan oleh *Dennis Ritchie* sekitar tahun *1972*.

Hingga kini bahasa ini masih populer dan penggunaannya tersebar di berbagai platform; dari Windows sampai Linux dan dari PC hingga mainframe. C++ adalah bahasa yang relatif baru dibandingkan dengan C. Bahasa C++ merupakan pengembangan dari Bahasa C dan mendukung pemrograman berorientasi objek.

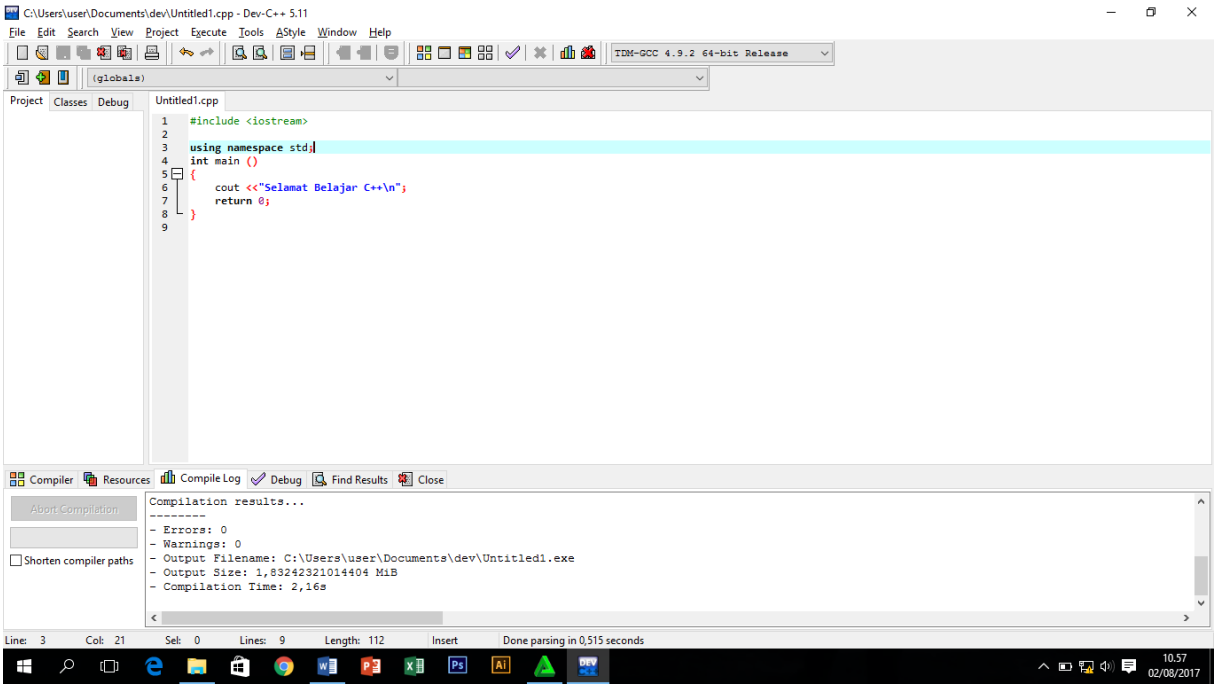
Dengan menggunakan C++, Anda tetap dapat menulis program C.

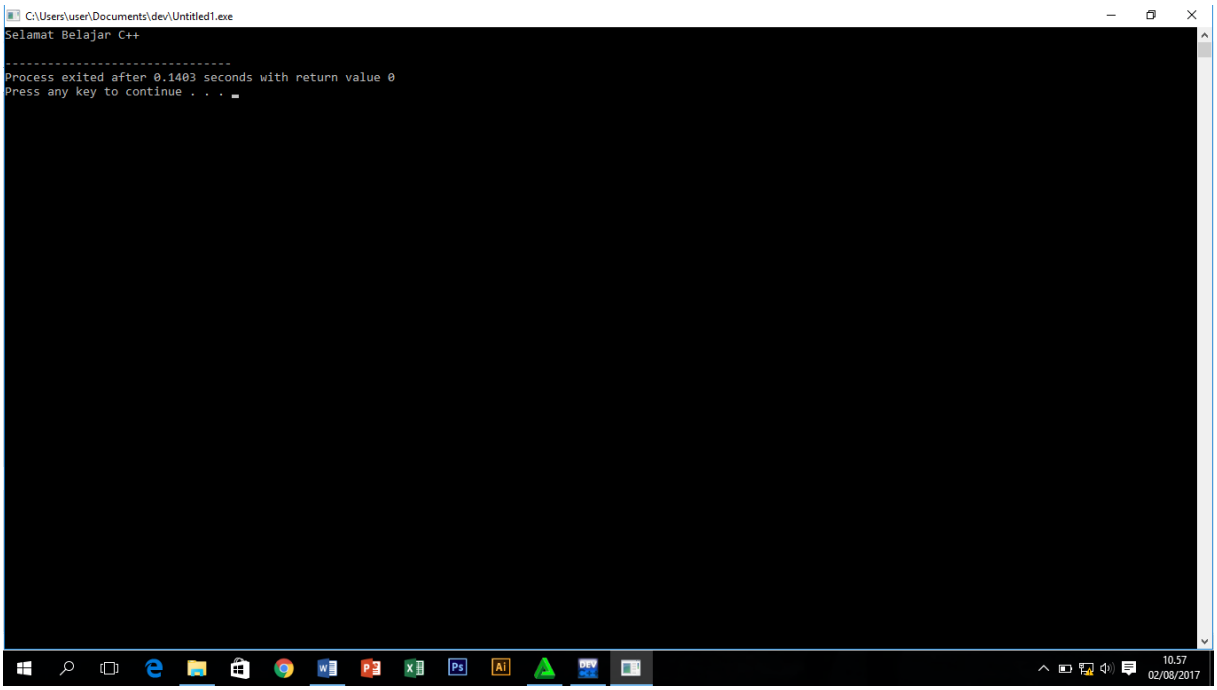
Pertama-tama Anda mengenal sebuah contoh program C++ yang sangat sederhana, sebagaimana dapat Anda lihat di bawah ini:

Kode Sumber C++ : **pertama.cpp**

```
#include <iostream.h>
    int main ()
    {
        cout <<“Selamat Belajar C++\n”;
        return 0;
    }
```

Tampilan pada Dev C++





	Akhir Kode Sumber

Perbedaan utama program C++ di atas dengan kode C:

- Berkas *header* yang digunakan bukan berupa **stdio.h** melainkan **iostream.h**.
- Pernyataan untuk menampilkan keluaran berupa **cout**. Dalam hal ini **cout** merupakan contoh sebuah objek pada C++.

4.2 Mengenal Pengenal

Pengenal (*identifier*) adalah suatu nama yang digunakan dalam program untuk menyatakan variabel, fungsi, dll. Aturan umum yang berlaku dalam membuat pengenal baik pada C msupun C++:

- Berawalan huruf kapital, huruf kecil, atau karakter garis-bawah (_).
- Bagian berikutnya dapat berupa huruf, angka (0..9), atau karakter garis-bawah.

Catatan 	Huruf kecil dan huruf kapital dibedakan pada pengenal. Itulah sebabnya, alamat dan Alamat adalah dua buah pengenal yang berbeda.

Contoh pengenal yang absah dan yang salah dapat dilihat pada tabel berikut:

Tabel 4.1 *Pengenal yang absah dan yang salah*

Absah	Salah
N Bilangan semester_1 PERUSAHAAN	Semester 1 (ada spasi) 3Bulan (diawali dengan angka) modal*bunga (ada tanda *)

4.3 Mengenal Tipe Data

C dan C++ menyediakan berbagai tipe data dasar, sebagiamana diperlihatkan pada tabel berikut:

Tabel 4.2 *Tipe data dasar*

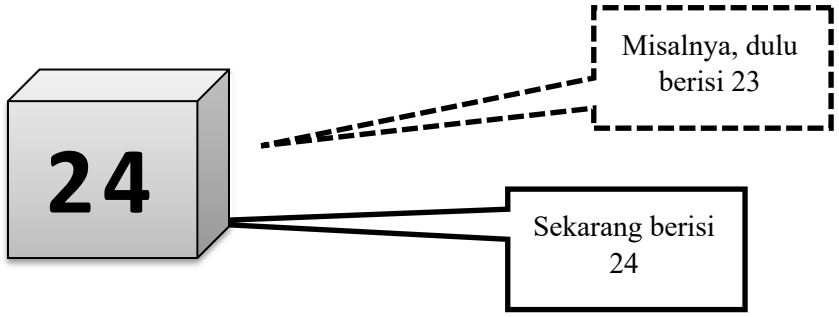
Tipe	Keterangan
Char	Menyatakan sebuah karakter (bisa berupa huruf seperti A dan a, digit seperti 0 atau 7, atau simbol seperti * dan &)

Double	Menyatakan bilangan titik-mengambang (bilangan real) dengan ketelitian tinggi
Float	Menyatakan bilangan titik-mengambang (bilangan real) dengan ketelitian rendah
Int	Menyatakan bilangan bulat antara -32768 sampai dengan 32767
long int	Menyatakan bilangan bulat yang berkisar antara -2147483648 s/d 2147483647

Catatan 	Beberapa pemodifikasi tipe seperti **signed** dan **unsigned** juga dapat diterapkan pada sejumlah tipe dasar. Misalnya, unsigned int berarti bilangan bulat tak bertanda, yang dapat mencakup nilai antara 0 sampai dengan 65535. Jika Anda tertarik dengan detail pemrograman C, Anda dapat membaca buku penulis yang berjudul “Pemrograman Dasar Turbo C untuk IBM PC” (Penerbit Andi).

4.4 Mengenal Variabel

Variabel adalah suatu nama yang menyatakan tempat dalam memori komputer yang digunakan untuk menyimpan suatu nilai dan nilainya dapat diubah sewaktu-waktu ketika program sedang dieksekusi.



4.4.1 Mendeklarasikan Variabel

Sebelum Anda dapat menggunakan suatu variabel dalam program, variabel harus dideklarasikan dulu. Pendeklarasian variabel digunakan untuk memesan lokasi dalam memori komputer dan menentukan tipe dari data yang dapat disimpan dalam variabel. Bentuk pendeklarasiannya adalah sebagai berikut:

```
tipe_data nama variable;
```

Contoh dapat dilihat pada tabel berikut:

Tabel 4.3 *Contoh deklarasi variabel*

Deklarasi	Keterangan
int n;	Variabel n bertipe **int** (untuk menyimpan bilangan bulat)
char ch;	Variabel ch bertipe **char** (dipakai untuk menyimpan sebuah karakter)
long int jum_penduduk;	Variabel jum_penduduk bertipe **long int** (dipakai untuk menyimpan bilangan bulat dengan nilai mencapai sekitar 2 milyar)
double jarak;	Variabel jarak bertipe **double** (dipakai untuk menyimpan sebuah bilangan real dengan kepresisian tinggi)

Jika ada beberapa variabel dengan tipe yang sama, pendeklarasian dapat dilakukan dengan menggunakan sebuah pernyataan. Contoh:

```
int i, j, k;
```

4.4.2 Memberikan Nilai ke Variabel

Untuk mengisikan nilai ke variabel, pernyataan yang diperlukan berbentuk

variabel = nilai;

Contoh:

n = 7;

Merupakan pernyataan untuk mengisikan 7 ke variabel n.
Pemberian nilai juga dapat berbentuk semacam berikut;

n = n + 1

Yang berarti “hasil penjumlahan nilai n dengan 1 diberikan ke n”. Dengan kata lain, isi n dinaikan sebesar satu.

Berikut ini ditunjukkan cara mengonversi pernyataan dalam pseudokode ke dalam bentuk pernyataan C.

Tabel 4.4 *Pengonversian pseudokode ke pernyataan C dan C++*

Pseudokode	Pernyataan C dan C++
n ← 7	n = 7;
n ← n + 1	n = n + 1;
luas ← panjang x lebar	luas = panjang * lebar;

4.5 Mengenal Literal

Literal atau konstanta menyatakan nilai yang tetap di dalam program.
Contoh:

- 2.3 (literal bilangan titik-mengambang)
- 10 (literal bilangan bulat)
- ‘K’ (sebuah karakter K)

Beberapa aturan penting yang perlu diketahui dalam menuliskan berbagai literal dapat dilihat pada tabel berikut:

Tabel 4.5*Aturan penulisan literal*

Tipe	Keterangan
int	Berupa bilangan bulat antara -32768 s/d 32767. Tanda pemisah ribuan tidak boleh digunakan.
long int	Berupa bilangan bulat antara -2147483648 s/d 2147483647. Tanda pemisah ribuan tidak boleh digunakan. Tanda L atau 1 (huruf kecil 1) perlu disertakan di bagian akhir literal untuk menyatakan tipe **long int** secara eksplisit. Contoh: 4276898L
Float	Harus ditulis dengan akhiran F atau f. Tanda pecahan berupa titik. Notasi sains seperti 2.2e+04 diperkenalkan (artinya 2,2 x 10^4). Huruf e dapat ditulis dengan E.
Double	Seperti pada **float**, tetapi tidak perlu akhiran F atau f.
Char	Literal ditulis dengan awalan dan akhiran petik tunggal (‘). Di dalam tanda petik tersebut terdapat satu karakter. Literal bertipe **char** juga dapat mengandung dua karakter atau lebih dengan karakter pertama berupa \. Pada keadaan seperti ini, deretan karakter dalam tanda petik tunggal tersebut tetap menyatakan sebuah karakter.

4.6 Mengenal Karakter Escape

Karakter *escape* adalah sebuah karakter yang ditulis dengan awalan tanda \. Tabel 4.6 memperlihatkan karakter *escape* pada C dan C++.

Tabel 4.6*Daftar karakter escape*

Karakter	Keterangan
\0	Karakter NULL (Tulisannya berupa \ dan angka
\a	Karakter bel
\b	Karakter *backspace*
\f	*Formfeed*
\n	*Linefeed* (disebut juga *newline* atau pindah baris)
\r	*Carriage return*
\t	Tab horisontal
\v	Tab vertikal
\\	Karakter \
*	Karakter petik tunggal
**	Karakter petik ganda
\?	Karakter tanda tanya
\DDD	Menyatakan sebuah karakter yang nilai ASCII-nya sama dengan nilai oktal DDD
\xHH	Menyatakan sebuah karakter yang nilai ASCII-nya sama dengan nilai heksadesimal HH

4.7 Mengenal String

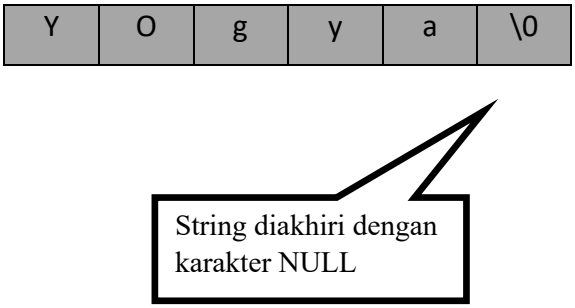
String berarti deretan karakter. Dalam praktik, sebuah string dapat saja tidak mengandung karakter sama sekali, mengandung sebuah karakter, atau mengandung banyak karakter. Konstanta string ditulis dengan awalan dan akhiran tanda petik ganda. Contoh:

Tabel 4.7*Contoh string*

String	Keterangan
“Yogya”	String dengan panjang 5 karakter
“Y”	String dengan panjang 1 karakter. Literal ini berbeda dengan literal karakter. Sebagaimana diketahui. Literal karakter ditulis dengan awalan dan akhiran petik tunggal.

***	String kosong (tidak mengandung satu karakterpun)
“Jl. Beo 45/B”	String yang mengandung huruf, angka, dan simbol.
“123”	String yang berisi angka 123. Hal ini berbeda dengan literal 123.

String pada C dan C++ selalu diakhiri dengan karakter NULL (\0). Sebagai contoh, jika terdapat string “Yogya”, secara internal string itu disajikan seperti berikut:



Gambar 4.3 *Karakter NULL sebagai akhiran dalam string*

Variabel yang digunakan untuk menyimpan string perlu dideklarasikan sebagai berikut:

```
char nama [15]
```

Pada deklarasi di atas, nama dideklarasikan sebagai variabel string yang dapat menampung hingga 15 karakter (termasuk untuk karakter NULL).

Catatan	Secara umum, jika dikehendaki untuk membuat variable string yang dapat menampung n karakter (tidak termasuk NULL pengakhir), nilai yang perlu diberikan di dalam tanda [] berupa n + 1. Tambahan 1 dipakai untuk mengalokasikan karakter NULL.

Berbeda dengan penanganan literal non-string ke suatu variable, penugasan literal string ke variable string harus dilakukan melalui fungsi bernama **strcpy0**. Dalam hal ini dalam program harus disertakan baris seperti berikut :

#include <stdio.h>

Hal ini perlu dilakukan mengingat prototype fungsi **strcpy0** terdapat pada berkas tersebut.

Contoh program C :

Char info [25];

digunakan untuk mendeklarasikan variable string bernama info yang dapat menyimpan hingga 24 karakter (plus sebuah karakter NULL).

Melalui pernyataan

strcpy (info, “Nama: David Beckam”) ;

variable string info diisi dengan string “nama: David Beckam”.

Selanjutnya, pernyataan

strcpy (info, “Info: Pemain sepakbola”) ;

diganti dengan “Info: Pemain sepakbola”.

Hasil pengeksekusian program dapat dilihat pada gambar berikut :

```
C:\progc>kopistr↵
Nama: David Beckham
Info: Pemain sepakbola
C:\progc>
```

Gambar 4.4*Hasil program kopistr*

Program C++ yang setara dengan kode di depan :

Kode Sumber C++ : **kopistr.cpp**

```
#include <iostream.h>
 #include <string.h>

int main ()
{
   Char info [25];

   Strcpy (info, “Nama: David Beckam”);
  cout<< info ;

cout<< “\n” ;

   Strcpy (info, “Info: pemain sepakbola”);
cout<< info ;

   Return 0;
}
```

Akhir Kode Sumber

Inisialisasi terhadap variable string dapat dilakukan saat pendeklarasian.

Contoh :

Char info [25] = “Nama : David Beckam”;

digunakan untuk mendeklarasikan variable string bernama info yang dapat menyimpan hingga 24 karakter (plus sebuah karakter NULL) dan selanjutnya variable tersebut diisi dengan string :

“Nama : David Beckam”

```
C:\progc>initstr↵
Nama: David Beckham
Info: Pemain sepakbola
C:\progc>
```

Gambar 4.5*Hasil program initistr*

Contoh dalam C++ :

Kode Sumber C++ : **kopistr.cpp**

```
#include <iostream.h>
#include <string.h>

int main ()
{
      char info [25] = “Nama: David Beckam”);
      cout<< info ;

      cout<< “\n” ;

      Strcpy (info, “Info: pemain sepakbola”);
            cout<< info ;

      Return 0;
}
```

Akhir Kode Sumber

Untuk mengakses suatu karakter yang terdapat dalam sebuah variable string, notasi berikut dapat digunakan :

nama_variabel [*indeks*]

Dalam hal ini *indeks* dimulai dari nol.

4.8 Menampilkan Informasi ke Layar

4.8.1 Pada Bahasa C++

Pada C++, cout dapat digunakan untuk menampilkan tipe data apa saja dengan bentuk yang sederhana, cukup dengan melibatkan operator <<.

Contoh:

cout << “pemrograman C++”;

cout << ‘A’;

Pada contoh pertama, yang ditampilkan adalah data string, sedangkan pada contoh kedua berupa tipe karakter.

Untuk keperluan pemformatan data pada keluaran, C++ menyediakan sejumlah manipulator. Contoh manipulator untuk mengonversi ke sistem bilangan heksadesimlal dan octal :

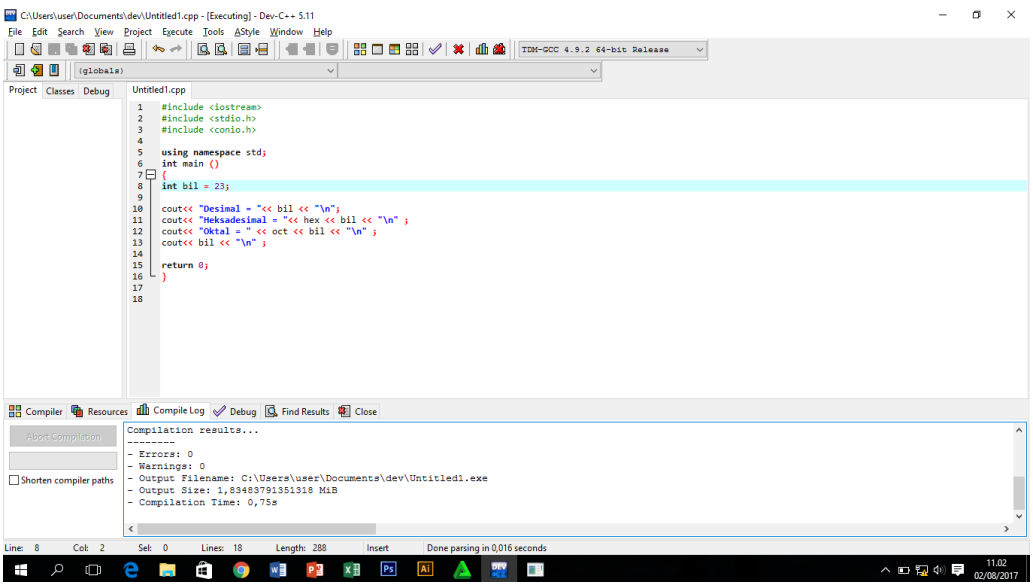
- Hex : untuk menampilkan bilangan dalam bentuk heksadesimal
- Oct : untuk menampilkan bilangan dalam bentuk octal

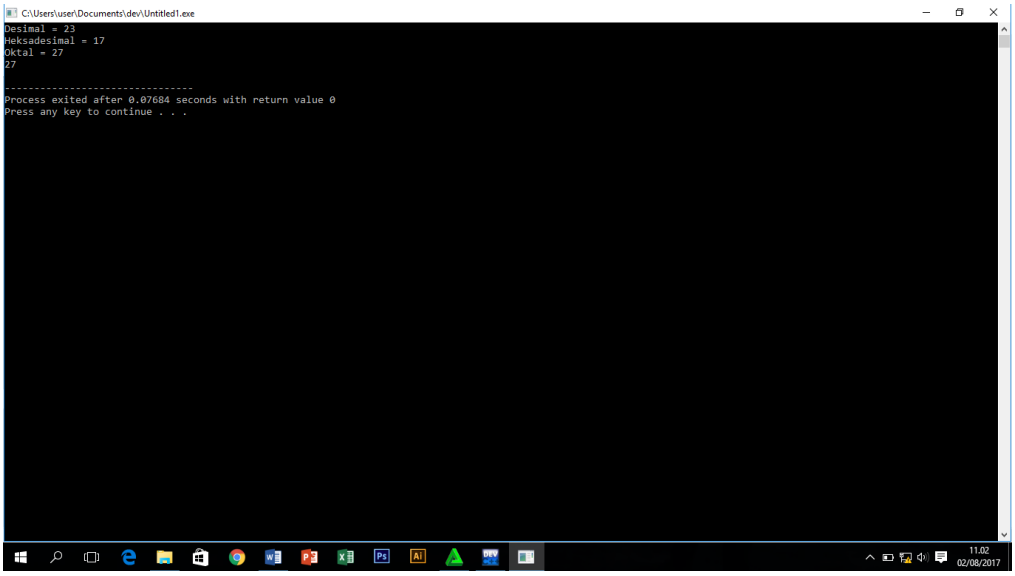
Program berikut menunjukkan penggunaan kedua manipular tersebut :

Kode Sumber C++ : **hexoct.cpp**

```
#include <iostream.h>

int main ()
{
      Int bil = 23;

      cout<< “Desimal = “<< bil << “\n”;
      cout<< “Heksadesimal = “<< hex << bil << “\n” ;
      cout<< “Oktal = “ << oct << bil << “\n” ;
      cout<< bil << “\n” ;

      return 0;
}
```

Tampilan Dev C++





	Akhir Kode Sumber

Hasil program di atas adalah seperti berikut:

```
C:\progc>hexoct↵
Desimal = 23
Heksadesimal = 17
Oktal = 27
27
C:\progc>
```

Tampilan terakhir menunjukkan bahwa manipulator **oct** mempengaruhi tampilan bilangan-bilangan yang terletak sesudahnya. Untuk mengembalikan ke format decimal (bilangan berbaris 10), gunakan manipulator **dec**. contoh:

cout << dec << bil << "\n";

Contoh manipulator yang lain:

- **Setw(*n*):** untuk menentukan lebar tampilan sebanyak n karakter
- **Setfill(*karakter*):** untuk menambahkan karakter pada bagian ruang yang ditentukan oleh **setw** yang normalnya tidak berisi karakter apapun
- **Setprecision(*n*):** untuk menentukan jumlah digit pecahan.

Adapun table berikut menunjukkan sejumlah argument yang dapat digunakan pada manipulator **setiosflags** dan **resetiosflags.**

Tabel 4.9*Argumen untuk manipulator setiosflags dan resetiosflags*

Nilai	Makna jika sisetel
ios::skipws	Mengabaikan spasi-putih pada pemasukan data
ios::left	Keluaran diatur rata kiri
ios::right	Keluaran diatur rata kanan
ios::dec	Konversi ke sistem desimal
ios::hex	Konversi ke sistem heksadesimal
ios::oct	Konversi ke system oktal
ios::uppercase	Keluaran dalam system heksadesimal menggunakan huruf kapital
ios::showpos	Menampilkan tanda + untuk bilangan positif
ios::scientific	Menggunakan notasi sains
ios::fixed	Menggunakan notasi tetap (bukan sains)

Catatan	
	• Jika manipulator setw, setprecision, setfill, setiosflags, dan resetiosflags digunakan, program harus menyertakan berkas *header* bernama **iomanip.h**. • Manipulator setiosflags digunakan unyuk menyetel suatu keadaan sedangkan resetiosflags digunakan untuk menghasilkan suatu keadaan.

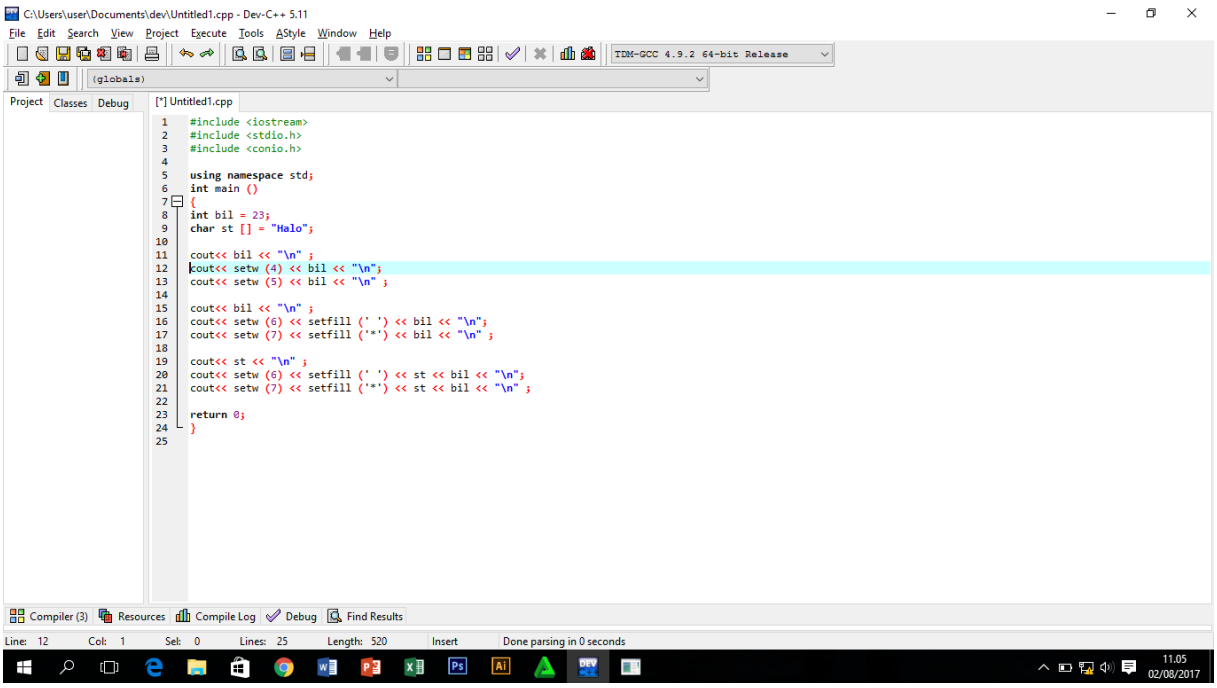
contoh penggunaan **setw** dan **setfill**:

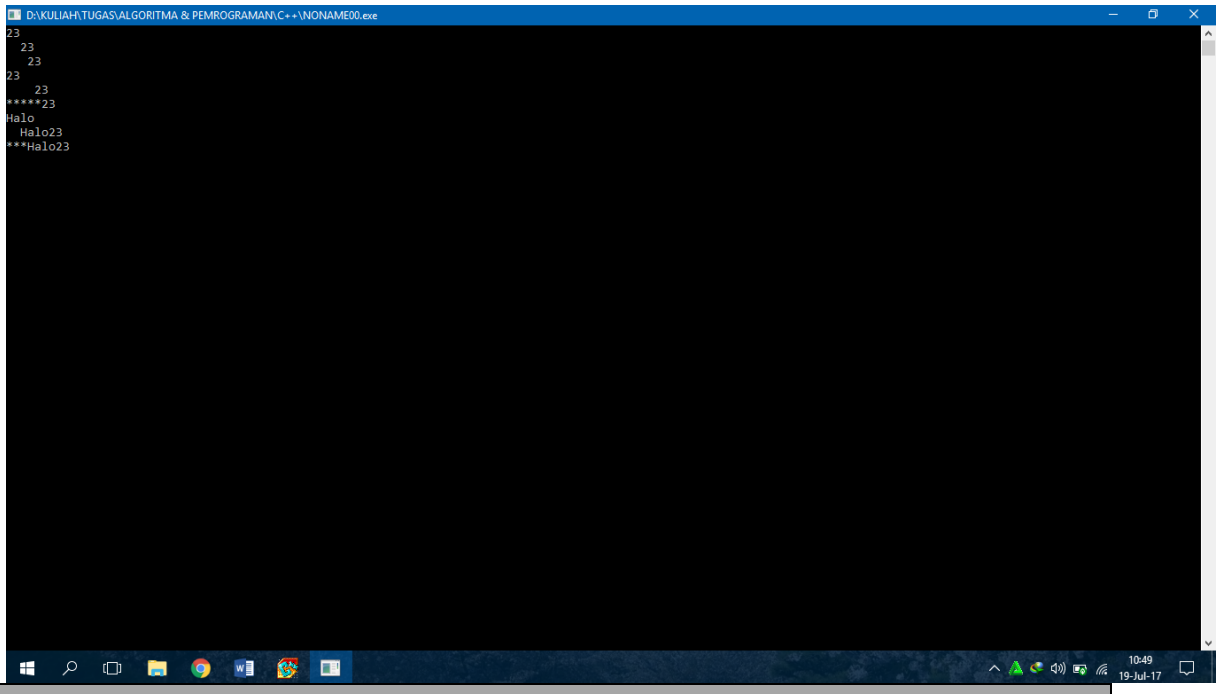
 Kode Sumber C++ : **manip.cpp**

```
#include <iostream.h>
#include <iomanip.h>

int main ()
{
      int bil = 23;
      char st [] = “Halo”;

      cout<< bil << “\n” ;
      cout<< setw (4) << bil << “\n”;
      cout<< setw (5) << bil << “\n” ;

      cout<< bil << “\n” ;
      cout<< setw (6) << setfill (‘ ‘) << bil << “\n”;
      cout<< setw (7) << setfill (‘*’) << bil << “\n” ;

      cout<< st << “\n” ;
      cout<< setw (6) << setfill (‘ ‘) << st << bil << “\n”;
      cout<< setw (7) << setfill (‘*’) << st << bil << “\n” ;
```

```
    return 0;
}
```

Tampilan Dev C++





	Akhir Kode Sumber

Hasilnya seperti berikut:

```
C:\progc>manip↵
23
  23
    23
23
      23
*****23
Halo
  Halo
***Halo
C:\progc>
```

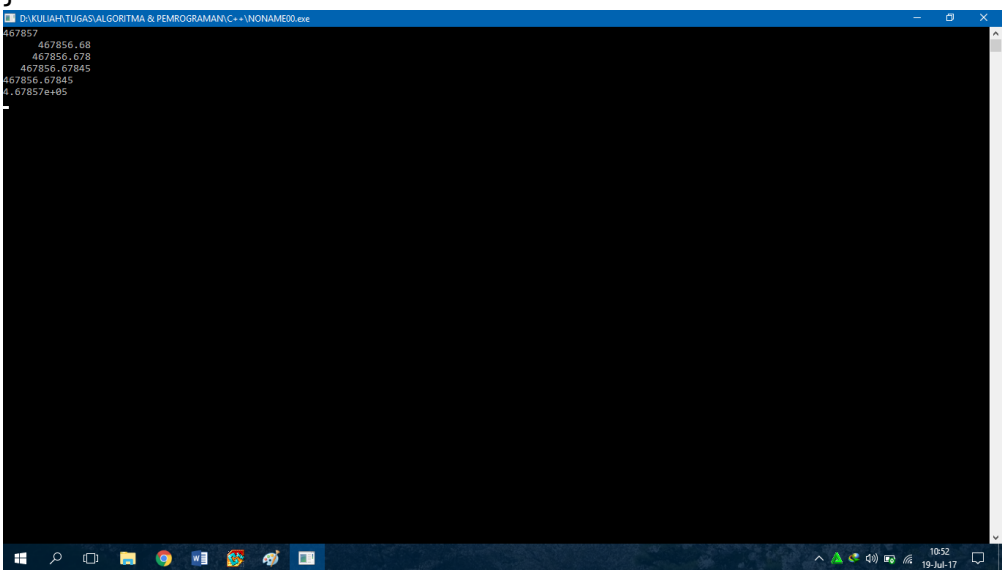
Gambar 4.11 *Hasil program manip*

Adapun contoh berikut menunjukkan penggunaan manipulator **setiosflags:**

Kode Sumber C++ : **setios.cpp**

```
#include <iostream.h>
#include <iomanip.h>

int main ()
{
      double bil = 467856.6784532;

      cout<< bil << "\n";

      cout<< setiosflags (ios::fixed)
            <<setw (15)

            <<setprecision (2)
            <<bil<< "\n" ;

      cout<< setw(15)
            <<setprecision (3)
            <<bil<< "\n" ;

      cout<< setw(15)
            <<setprecision (5)
            <<bil<< "\n" ;
```

```
        cout<< bil << "\n"; // Tetep berpresisi 5 digit pecahan

        cout<< setiosflags (ios::scientific)
             <<bil<< "\n" ;

        return 0;
}
```



	Akhir Kode Sumber

Hasilnya seperti berikut:

467857

467856.68 ← Efek setprecision (2)

467856.678 ← Efek setprecision (3)

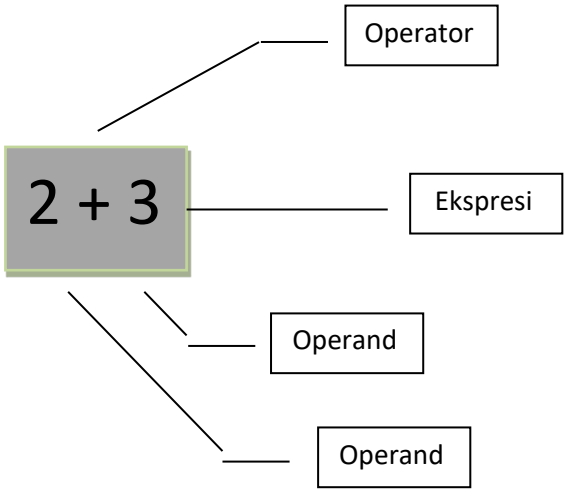
467856.67845 ← Efek setprecision (5)

467856.67845

4.6786e+05 ← Efek setiosflags (ios::scientific)

4.9 Mengenal Operator

pada operasi seperti 2 + 3, tanda + dinamakan operator. Operator berupa symbol yang digunakan untuk menyusun suatu ekspresi, dengan melibatkan satu atau beberapa *operand*, tergantung dari jenis operator.Pada contoh 2 + 3, ada dua buah *operand* yang dilibatkan, yaitu 2 dan 3.Adapun 2 + 3 sendiri dinamakan sebagai ekspresi.Dalam hal ini ekspresi dapat dipakai untuk melakukan perhitungan atau bahkan perbandingan.



Gambar 4.12 *opertor, operand, dan ekspresi*

Ditinjau dari jumlah *operand* yang dilibatkan dalam sebuah operator, terdapat tiga macam operator.

1. ***Operator unary***, yaitu operator yang hanya melibatkan sebuah *operand*.Contoh:

   + 1

   -1

2. ***Operator binary***, yaitu operator yang melibatkan dua buah *operand*.Contoh:

   2 + 3

   5 * 2

   5 > 2

3. ***Operator tertiary***, yaitu operator yang melibatkan tiga buah *operand*.

   a >b ? : 0

   Ekspresi di atas berarti “jika a lebih besar dari pada b maka ekspresi menghasilkan nilai 1, sedangkan kalau tidak maka ekspresikan menghasilkan nilai 0 ”

.

Catatan 	?: adalah sebuah operator.

Berdasarkan kelompok kegunaan, operator dapat dibagi menjadi operator aritmetika, operator pembanding, operator logika, dan operator lain-lain.

4.9.1 Operator Aritmetika

Operator aritmetika digunakan untuk melakukan perhitungan aritmetika.

Daftar operator aritmetika dapat dilihat pada Table 4.10.

Tabel 4.10 *Daftar operator aritmetika*

Operator	Prioritas	Keterangan	Contoh
-	1	Unary minus	-1
+	1	Unary plus	+1
*	2	Perkalian	2*3 → 6 2*3.0 → 6.0
/	2	Pembagian	7 / 2 → 3 7 / 2.0 → 3.5 7.0 / 2 → 3.5
%	2	Sisa pembagian	
+	3	Penjumlahan	
-	3	pengurangan	

Catatan	
	• Prioritas dalam operator menentukan urutan pengerjaan dalam satu ekspresi. Contoh, pada ekspresi seperti 2 + 3 * 4, 3 * 4 akan dikerjakan telebih dahulu. • Hasil ekspresi ditentukan oleh tipe *operand*. Lebih lanjut, lihatlah 4.10. • C dan C++ tidak mendukung operator untuk menangani perpangkatan. Untuk menangani perpangkatan, C dan C++ menyediakan fungsi bernama **pow ()**. Prototipe fungsi ini ada pada berkas header **math.h**. • Pada GNU/Linux, jika Anda menggunakan gcc untuk mengompilasi program C yang melibatkan fungsi matematika, Anda perlu menambahkan opsi berupa – lm (hurup L kecil dan M kecil). Contoh: gcc –lm namaprog.c –o namaprog

Berkas header math.h menyediakan prototype sejumlah fungsi yang terkait dengan aritmetika.Table 4.11 mencantumkan beberapa fungsi untuk operasi aritmetika.

Tabel 4.11 *sejumlah fungsi aritmetika*

Fungsi	Keterangan
sqrt (x)	Memberikan nilai balik berupa akar x. Nilai balik bertipe **double** dan argumen juga bertipe **double**.
pow (x,y)	Memberikan nilai balik berupa x^y. Nilai balik beripe **double** dan argumen juga bertipe **double**.\
tan (x)	Memberikan nilai balik berupa *tangent* x. Argument x berupa nilai dalam satuan radians. Nilai balik bertipe **double** dan argument juga bertipe **double**.
sin (x)	Memberikan nilai balik berupa Sinus x. Argumen x berupa nilai dalam satuan radians. Nilai balik bertipe **double** dan argument juga bertipe **double**.
cos (x)	Memberikan nilai balik berupa Cosinus x. Argumen x berupa nilai dalam satuan radians. Nilai balik bertipe **double** dan argument juga bertipe **double**.
log (x)	Memberikan nilai balik berupa $\log_e$ x. Nilai balik bertipe **double** dan argumen juga bertipe **double**.
log10 (x)	Memberikan nilai balik berupa $\log_{10}$ x. Nilai balik bertipe **double** dan argumen juga bertipe **double**.
cabs (x)	Memberikan nilai balik berupa nilai absolut x. Nilai balik bertipe **int** dan argumen juga bertipe **int.**
fabs (x)	Memberikan nilai balik berupa nilai absolute x. Nilai balik bertipe **double** dan argumen juga bertipe **double.**

Table 4.12 mencantumkan sejumlah contoh notasi matematika dan penulisan rumus pada C dan C++.

Tabel 4.12 *Notasi matematika dan ekspresi dalam C dan C++*

Notasi Matemmatika	Ekspresi C dan C++
$ax^2 + bx + c$	a * x * x * + b * x + c
$\sqrt{b}$	sqrt (b)
$\sqrt[3]{b}$	pow (b, 1.0/3)
a + b	(a + b) / (c + d)
$\sqrt{b^2 - 4\ ac}$	sqrt (b * b – 4 * a * c)

	exp (x + y) / (x + y)
	a + b / fabs (m-n)
Log_{10} m	Log10 (m)
Log_e (m + n)	Log (m + n)

Program peluru.c memberikan contoh pemakaian fungsi **sin ()** dan **cos ()** untuk menghitung jarak peluru jatuh terhadap posisi penembakan peluru.Gambar 4.13 memberikan ilustrasi tentang hal itu.

```
C:\progc>peluru↵
Jarak = 102.88942
C:\progc>C:\progc>
```

Gambar 4.13 a *Lintasan peluru*

Jarak lintasan peluru (X) dapat dihitung dengan menggunakan rumus:

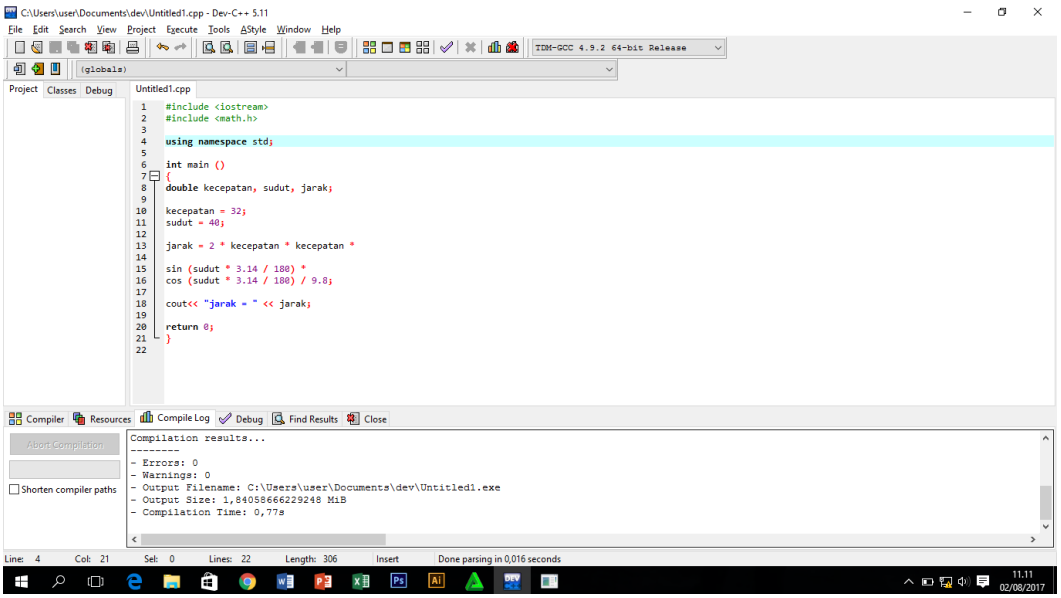
Jarak = 2 x V_0^2 x sin α x cos α / g

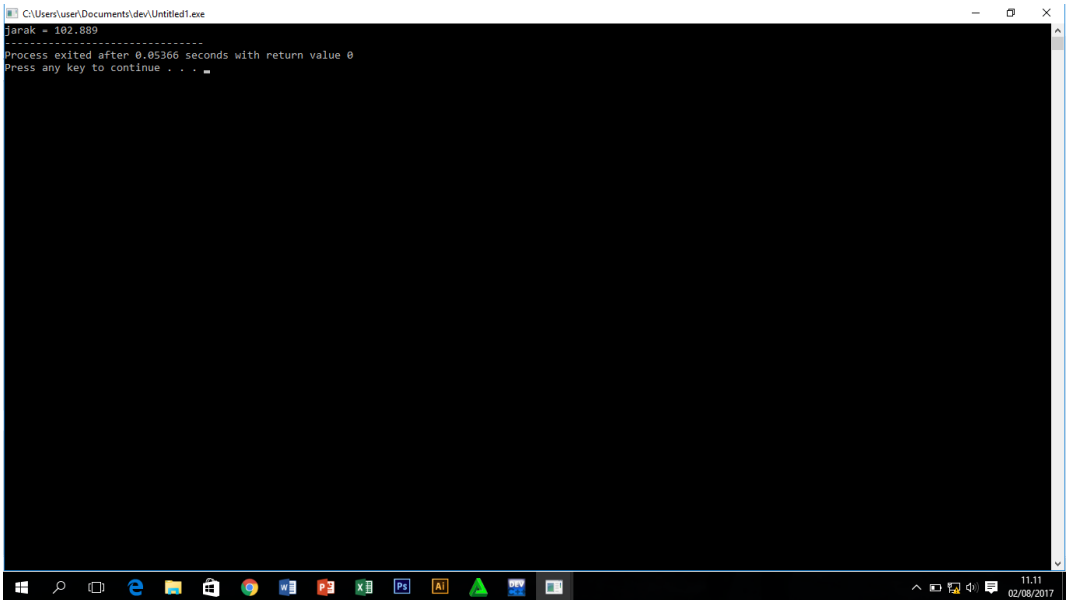
dengan V_0 adalah kecepatan tembak dan g adalah gaya gravitasi (9,8).

Kode Sumber C++ : **peluru.cpp**

```
#include <iostream.h>
#include <math.h>

int main ()
{
      double kecepatan, sudut, jarak;

      kecepatan = 32;
      sudut = 40;

      jarak = 2 * kecepatan * kecepatan *

      sin (sudut * 3.14 / 180) *
      cos (sudut * 3.14 / 180) / 9.8;

      cout<< "jarak = " << jarak;

      return 0;
}
```

Tampilan dev c ++





	Akhir Kode Sumber

4.9.2 Operator Pembandingan

Operasi pembandingan atau disebut juga operator relasional adalah operator yang berguna untuk melakukan pembandingan terhadap dua buah nilai. Hasil pembandingan bernilai 1 atau 0. Dalam hal ini:

- Nilai nol berarti bahwa pembandingan memberikan hasil bernilai salah, dan
- Nilai satu berarti bahwa pembandingan memberikan hasil bernilai bener.

Daftar keseluruhan operator pembandingan dapat dilihat pada Table 4.13.

Tabel 4.13 *Notasi matematika dan ekspresi dalam C dan C++*

Operator	Keterangan
>	Lebih dari
>=	Lee
<	Kurang dari
< =	Kurang dari atau sama dari
! =	Tidak sama dengan
= =	Sama dengan

Contoh dapat dilihat pada Table 4.14.

Tabel 4.14 *Contoh pemakaian operator pembanding*

Ekspresi	Keterangan
5 > 1	1 (bener)
4 < 5	Lee
4 <= 3	0 (salah)
‘A’ > ‘B’	0 (salah)
‘B’ > ‘A’	1 (bener)
‘A’ < ‘a’	1 (bener)

Pada data bertipe karakter, pembandingan didasarkan pada nilai ASCII dari karakter yang dibandingkan. Misalnya, nilai ASCII A adalah 65, nilai ASCII B berupa 6, dan nilai ASCII a berupa 97.

Catatan	Operator pembandingan hanya dapat dipakkai untuk tipe dasar; tidak dapat digunakan untuk string.

4.9.3 Operator Logika

Operator logika digunakan membentuk suatu ekspresi pembandingan dari satu atau dua buah ekspresi pembandingan.Tiga macam operator logika yang tersedia pada C dan C++ dapat dilihat pada Tabel 4.15.

Tabel 4.15 *Daftar operator logika*

Operator	Keterangan
&&	Operator “dan”
||	Operator “atau”
!	Operator “bukan”

Operator && dan || melibatkan dua buah *operand*, sedangkan operator !melibatkan sebuah *operand*. Hasil operasi dengan operator logika&& dan || dapat dilihat pada Table 4.16.

Tabel 4.16*Tabel logika untuk operasi && dan ||*

A	B	A && B	A || B
Salah	Salah	Salah	Salah
Salah	Bener	Salah	Bener
Bener	Salah	Salah	Bener
Bener	Bener	Bener	Bener

Operasi dengan !berbentuk:

! *operand*

Dalam hal ini *operand* berupa ekspresi yang menghasilkan nilai benar atau salah. Adapun hasil operasi berupa:

- Benar kalau *operand* bernilai salah,
- Salah kalau *opetrand* bernilai benar.

Contoh operasi dengan operator logika dapat dilihat pada Table 4.17.

Tabel 4.17 *Contoh operasi dengan operator logika*

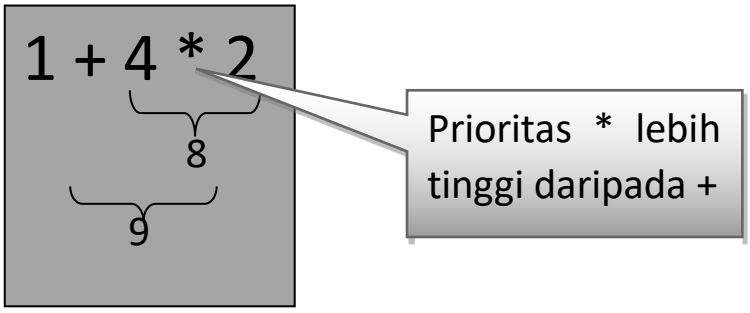
Ekspresi	Keterangan
Kar >= ‘A’ && kar <= ‘Z’	Hasil berupa benar hanya kalau kar berupa huruf kapital
Kode == ‘a’ || kode == ‘A’	Hasil berupa benar kalau kode berupa huruf a dan A
! (kar == ‘A’)	Hasil berupa benar kalau kar tidak berupa huruf A

4.10 Mengenal Prioritas Operator

Masing-masing operator dalam suatu ekspresi memiliki prioritas pengerjaan yangberbeda-beda.Itulah sebabnya jika terdapat suatu ekspresi yang melibatkan sejumlah operator, pengerjaan ditentukan oleh prioritas masing-masing.misalnya:

```
1 + 4 * 2
```

Akan dievaluasi sebagai berikut:

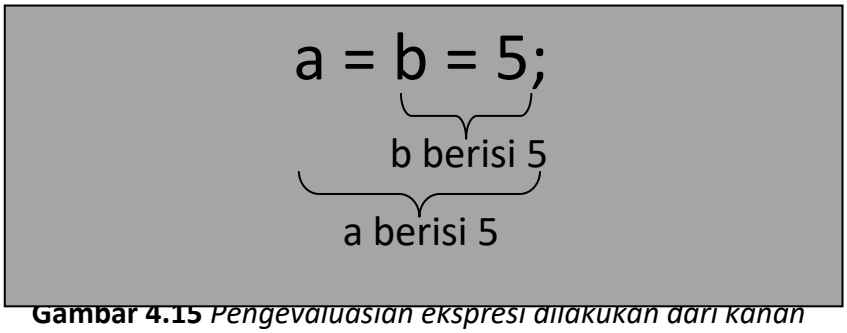


Gambar 4.14 *Urutan pengerjaan ekspresi*

Seandainya suatu ekspresi melibatkan operator dengan prioritas yang sama, umumnya pengerjaan dilakukan dari kiri (misalnya operator + dan -), tetapi ada juaga yang pengevaluasian dari kanan diperlihatkan berikut ini.

```
a = b = 5;
```

pada contoh di atas urutan pengerjaan diperlihatkan pada gambar berikut.



Gambar 4.15 *Pengevaluasian ekspresi dilakukan dari kanan*

Table 4.18 memperlihatkan prioritas operator-operator pada C dan C++ dimulai dari yang tertinggi menuju ke yang terendah. Operator yang terletak pada baris yang sama menyatakan bahwa prioritasnya sama. Table ini dapat Anda gunakan sebagai acuan kalau Anda menggunakan sejumlah operator dalam sebuah ekspresi.

Tabel 4.18 *Daftar operator dan prioritas*

Simbol	Nama	Prioritas	Urutan Pengerjaan
++ -- ( ) [ ] -> .	Penaikan di belakang Penurunan di belakang Pemanggilan fungsi Elemen array Pointer ke anggota struktur atau kelas Anggota struktur, union, atau kelas	Tertinggi	Kiri ke Kanan
++ -- ! ~ - + & * new delete sizeof (tipe)	Penaikan di depan Penurunan di belakang Logika bukan Operaor komplemen bit Tanda minus Tanda plus Alamat *Indirection* Pengalokasian memori Dealokasian memori Ukuran tipe data *Type casting*		Kanan ke Kiri
( )	Kurung untuk ekspresi		Kiri ke Kanan
* / %	Perkalian Pembagian Sisa pembagian (modulus)		Kiri ke Kanan
+ -	Penjumlahan Pengurangan		Kiri ke Kanan
<< >> < >	Geser bit ke kiri Geser bit ke kanan Kurang dari Lebih dari		Kiri ke Kanan

<= >=	Kurang dari atau sama dengan Lebih dari atau sama dengan		
== !=	Sama dengan Tidak sama dengan		Kiri ke Kanan
&	Operator bit “dan”		Kiri ke Kanan
^	Operator bit “xor”		Kiri ke Kanan
|	Operator bit “atau”		Kiri ke Kanan
&&	Logika “dan”		Kiri ke Kanan
||	Logika “atau”		Kiri ke Kanan
?;	Operator kondisi		Kiri ke Kanan
= *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=	Penugasan Operator majemuk		Kiri ke Kanan
,	Operrator koma	Terendah	Kanan ke Kiri

Kadangkala urutan pengerjaan dalam suatu ekspresi yang tidak tergantung oleh prioritas operator perlu diatur.Hal seperti ini dapat dilakukan dengan menggunakan tanda kurang. Contoh:

(1 + 4) * 2

memberikan hasil 10. Pada contoh ini, tanda kurang menyebabkan 1 + 4 dikerjakan terlebih dulu.Setelah itu baru mengerjakan pengalian hasil 1 + 4 dengan 2.

Contoh lain:

5 * (5 + (6 – 2) + 1)

memberikan hasil 50.

5 * (5 + (6 – 2) + 1)

Gambar 4.16 *Pengevaluasian ekspresi 5 * (5 + (6 - 2) + 1)*

4.11 Mengenal Pengonversian Tipe

Operasi perhitungan pada C dilakukan dengan menggunakan tipe data yang sama. Jika dalam suatu ekspresi terdapat *operand* dengan tipe yang berbeda, C mengonversi salah satu tipe sehingga kedua tipe menjadi sama.

Aturan yang digunakan oleh C dalam mengonversi tipe data adalah sebagai berikut:

1. Jika salah satu operand bertipe **long double**, maka yang lain dikonversi ke **long double**.
2. Jika salah satu operand bertipe **double**, maka yang lain dikonversi ke **double**.
3. Jika salah satu operand bertipe **float**, maka yang lain dikonversi ke **float**.
4. Jika salah satu operand bertipe **char, signed char, unsigned char,** atau **unsigned short** maka yang lain dikonversi ke **int**.
5. Tipe enumerasi akan dikonversikan ke **int, unsigned int, long,** atau **unsigned long** dengan mengakomodasi jangkauan tipe enumerasi.
6. Jika salah satu operand bertipe **unsigned long**, maka yang lain dikonversi ke **unsigned long**.
7. Jika salah satu operand bertipe **long** dan yang lain bertipe **unsigned int** maka kedua *operand* dikonversi ke **unsigned long**.
8. Jika salah satu operand bertipe **long**, maka yang lain dikonversi ke **long**.

Gambar 4.17 mengilustrasikan pengonversian tipe dalam sebuah ekspresi.Pada contoh tersebut variable jumlah bertipe **int**, variable hargaPerUnit bertipe **float**, dan variable hargaTotal bertipe **double**. Pada saat menemukan ekspresi

hargaPerUnit * jumlah

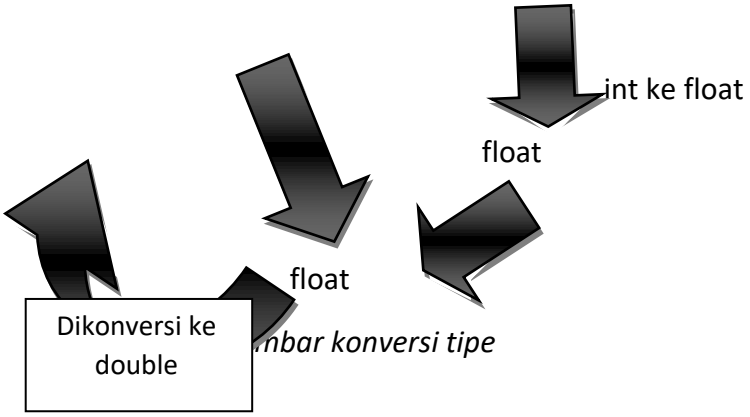
C/C++ akan mengonversi jumlah yang bertipe **int** ke **float** mengingat ada *operand* (yaitu hargaPerUnit) yang bertipe **float**. Selanjutnya, hasil ekspresi tersebut dikonversi ke **double** mengingat di sebelah kanan operator = bertipe **double**.

int jumlah;

float hargaPerUnit;

double hargaTotal;

Dikonversi dari



nbar konversi tipe

Kadangkala diperlukan langkah untuk mengubah suatu tipe data menjadi tipe data lain. Proses untuk mengubah suatu tipe ke tipe yang lain dikenal dengan istilah *type casting* atau pengarah tipe. Bentuk pengarah tipe adalah sebagai berikut:

(tipe_data) data

Contoh berikut memperlihatkan pengonversian data:

nilai = (int) kar;

isi variable kar dikonversikan ke tipe **int** dan kemudian disimpan ke variable nilai.

Hasil eksekusi program ditunjukkan pada gambar berikut:

```
C:\progc>konversi ↵
Nilai ASCII A : 65
C:\progc>ec
```

Gambar 4.18 *Hasil program konversi*

4.12 Mengenal Tipe Struct

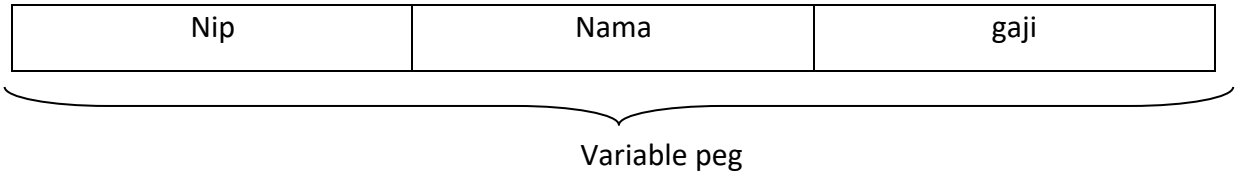
Di dalam C dab C++ terdapat tipe **struct** yang dapat dipakai untuk menghimpun sejumlah data dengan tipe yang berbeda-beda. Data yang diletakkan dalam sebuah **struct** adalah data yang terkait. Sebagai contoh, dimungkinkan untuk membuat tipe **struct** yang mengandung data nomor pegawai (NPI), nama pegawai, dan gaji. Pendefinisiannya adalah seperti berikut:

```
struct data_pegawai
{
        int nip;
        char nama [25];
        long int gaji;
} ;
```

Pada contoh di atas, tipe **stuct**data_pedawai terdiri atas data nip, nama, dan gaji yang secara berturut-turut bertipe **int, char [25]**, dan **long int**. dalam hal ini nip, nama, dan gaji disebut sebagai **elemen struct** atau **field**.

Setelah suatu tipe **struct** didefinisikan, tipe tersebut dapat dipakai untuk mendeklarasikan suatu variable. Caranya seperti berikut:

```
struct data_pegawai peg;
```



Gambar 4.19 *Variabel bertipe struct*

Untuk mengakses suatu *field*, notasi seperti berikut digunakan:

```
variable_struct.nama_field
```

contoh berikut menunjukkan pendefinisian tipe **struct**, pendeklarasian variable bertipe **struct**, dan cara mengakses setiap *field* dalam tipe **struct:**

Hasil pengeksekusian program dapat dilihat pada gambar berikut:

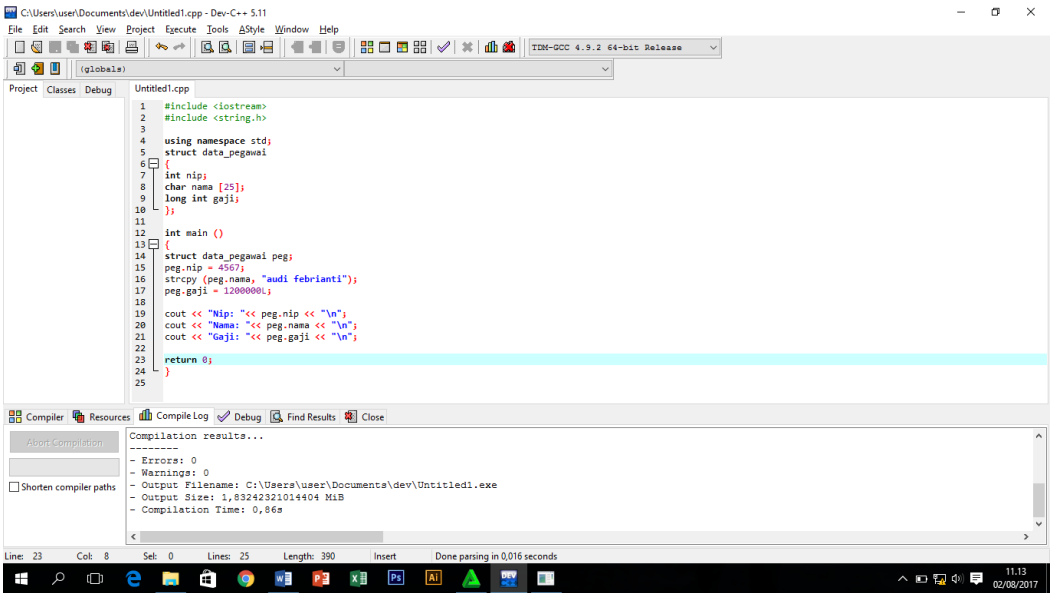
```
C:\progc>struct ↵
NIP : 4567
Nama : Audi Febrianti
Gaji : 1200000
C:\progc>
```

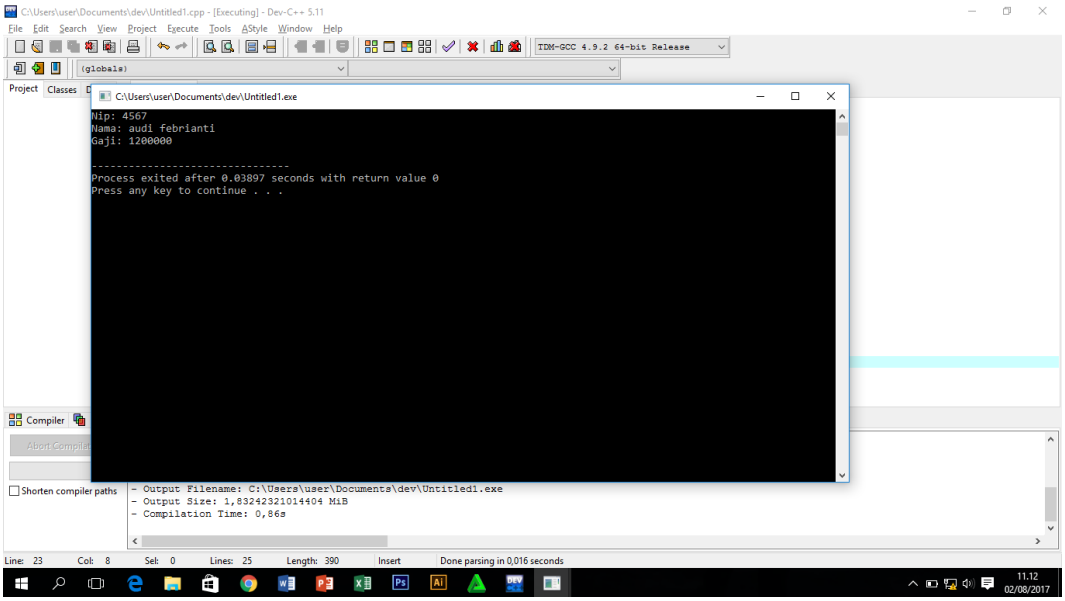
Program C++ serupa dengan struct.c di depan adalah seperti berikut:

Kode Sumber C++ : **struct.cpp**

```
#include <iostream.h>
#include <string.h>

struct data_pegawai
{
      int nip;
      char nama [25];
      long int gaji;
};

int main ()
{
      struct data_pegawai peg;
      peg.nip = 4567;
      strcpy (peg.nama, “audi febrianti”);
      peg.gaji = 1200000L;

      cout << “Nip: “<< peg.nip << “\n”;
      cout << “Nama: “<< peg.nama << “\n”;
      cout << “Gaji: “<< peg.gaji << “\n”;

      return0;
}
```

Tampilan dev c++





	Akhir Kode Sumber

4.13 Mengenal Komentar

Komentar biasa digunakan dalam program untuk memberikan penjelasan kepada pembaca program.Komentar tidak memberikan efek apa-apa pada keluaran.Isinya dapat saja berupa penjelasan terhadap suatu pernyataan atau sejumlah pernyataan; dapat pula berisi kegunaan program beserta waktu pembuatan dan pembuatanya.

Sebuah komentar ditulis di dalam pasangan tanda /* dan */. Isinya dapat mencakup lebih dari satu baris. Contoh komentar:

/* Program ini dibuat oleh Abdul Kadir */

Adapun contoh berikut menunjukkan komentar yang diletakkan di sebelah kanan pernyataan:

printf (“\n”); /* Menyisipkan karakter pindah baris */

Khusus pada C++, komentar dapat menggunakan symbol //. Dalam hal ini semua teks yang terletak sesudah symbol tersebut hingga akhir baris diperlukan sebagai komentar. Contoh:

// ini komentar pada C++

4.14 Mengenal Operasi Masukan

Seringkali dalam suatu program diperlukan operasi masukan yang berasal dari keyboard. Operasi inilah yang akan dibahas pada subbab ini.

4.14.1 Fungsi scanf()

Secara unum C menyediakan fungsi bernama **scanf()** dengan prototype terdapat pada berkas *header***stdio.h** untuk menangani pemasukan data dari keyboard. Fungsi ini dapay dipakai untuk menangani pemasukan berbagai tipe data. Bentuk penggunaannya:

scanf (“string kontrol”,ardumen);

Dalam hal ini “string kontrol” dapat berupa:

- Penentu format,
- Karakter spasi-putih,
- Karakter bukan-spasi-putih.

Adapun *argument* berupa alamat suatu variabel. Alamat suatu variable dapat diperoleh dengan menyertakan tanda & di depan suatu variable.

Catatan 	Yang termasuk spasi-putih adalah karakter spasi, *newline*, dan tab.

Penentu format menentukan tipe data yang akan dibaca. Kode-kode yang dapat dipakai dalam penentu format dapat dilihat pada Tabel 4.19.

Tabel 4.19 *Penentu format pada scanf()*

Kode	Keterangan
% c	Data yang dibaca berupa sebuah karakter
% s	Data yang dibaca berupa sebuah string
% i atau % d	Data yang dibaca berupa sebuah sebuah bilangan bulat
% e atau % f	Data yang dibaca berupa sebuah sebuah bilangan real
% u	Data yang dibaca berupa sebuah sebuah bilangan bulat tak bertanda
L (huruf L kecil)	Awalan untuk membaca data **long int** atau **double**. Contoh: % ld

Contoh penggunaan **scanf()** dapat dilihat pada table berikut:

Tabel 4.20 *Contoh pemakaian scanf()*

Pernyataan	Keterangan
Scanf (“ % f ”, & jarak);	Pernyataan untuk membaca sebuah bilangan real dan meletakkannya ke variable jarak yang bertipe **float**. Dalam hal ini & jarak berarti “alamat dari variable jarak”
Scanf (“ % d “, & jumlah);	Pernyataan untuk membaca sebuah bilangan bulat dan meletakkannya ke variable jumlah yang bertipe **int**.
Scanf (“ % ld “, & jum_penduduk);	Pernyataan untuk membaca sebuah bilangan bulat bertipe **long int** dan meletakkannya ke variable jum_penduduk yang bertipe **long int.**

Program berikut memberikan gambaran penggunaan **scanf()**. Program ini merupakan hasil pemodifikasian terhadap program peluru.c.Dalam hal ini data kecepatan dan sudut penembakan peluru dimasukkan melalui keyboard sewaktu program dieksekusi.

```
C:\progc>peluru2 ↵
Kecepatan : 32↵
Sudut : 40↵
Kec : 32.000000Jarak = 102.889492
```

Tabel berikut memberikan gambaran tentang cara mengonversi pseudokode yang terkait dengan pemasukan data ke dalam pernyataan C.

Tabel 4.21 *Contoh pengonversian pseudokode ke pernyataan C*

Pseudokode	Keterangan
masukan (jumlah) (Dengan asumsi jumlah bertipe **int**)	 Printf (“jumlah = “); Scanf (“%d”, &jumlah);
masukan (panjang, lebar) (Dengan asumsi panjang dan lebar bertipe **float**)	Printf (“panjang = “); Scanf (“%f”, &panjang); Printf (“lebar= “); Scanf (“%f”, &lebar);
masukan (nama)	Printf (“Nama = “);

(Dengan asumsi nama adalah variable string)	Scanf (“%s”, &nama); (Perhatikan bahwa di depan nama tidak diberi tanda &)

4.14.2 Pemasukan Data pada C++

Bagaimana halnya dengan C++?C++ menyediakan objek bernama **cin** yang dapat digunakan untuk membaca data dari keyboard san berlaku untuk sebarang tipe data. Bentuk penggunaannya adalah seperti berikut:

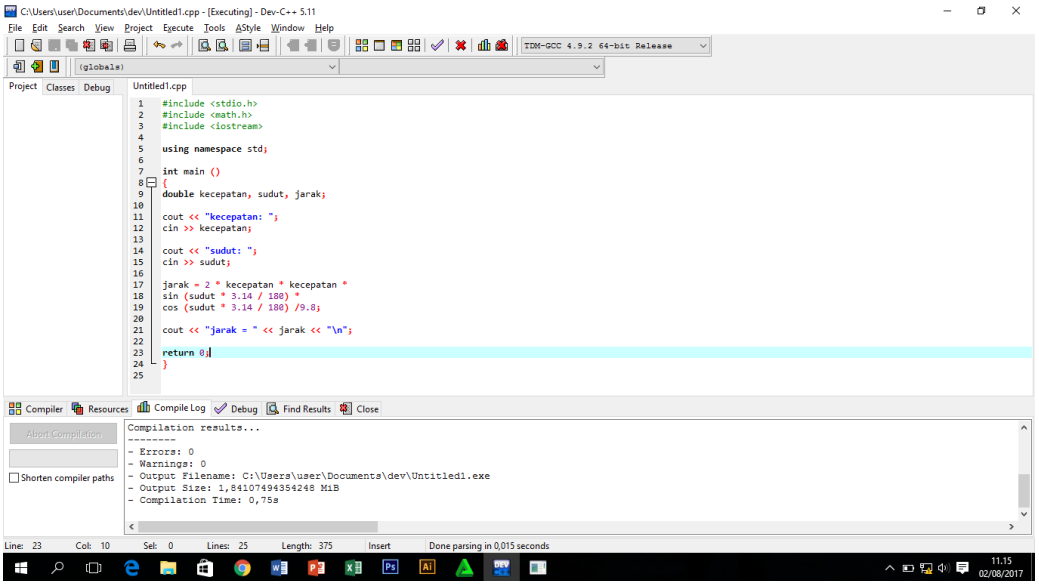
cin>> nama Variabel

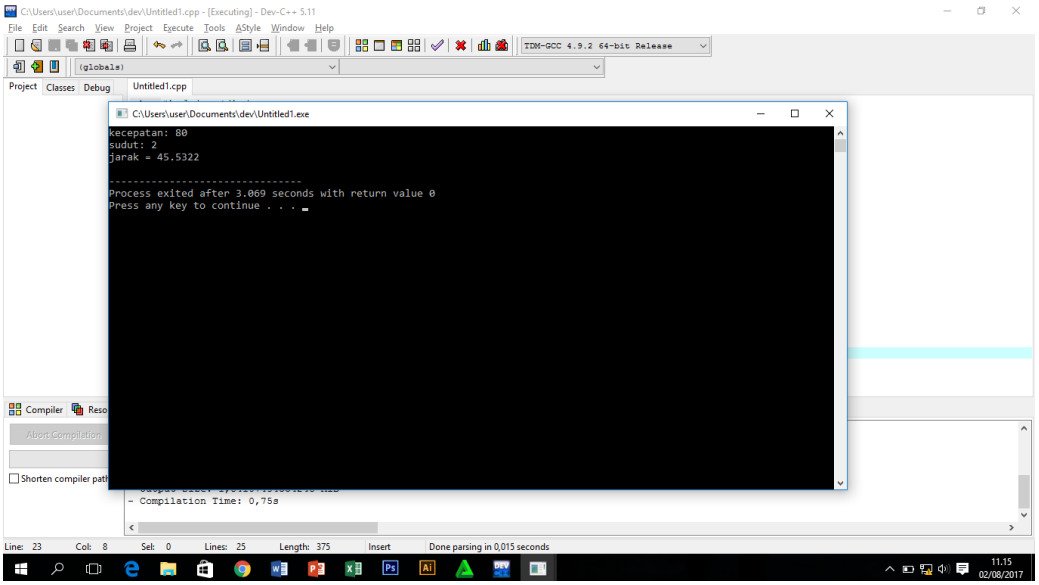
Contoh berikut menunjukan program C++ yang setara dengan program peluru2.cdi depan:

Kode Sumber C++ : **peluru2.cpp**

```
#include <stdio.h>
#include <math.h>

int main ()
{
      double kecepatan, sudut, jarak;

      cout << “kecepatan: “;
      cin >> kecepatan;

      cout << “sudut: “;
      cin >> sudut;

      jarak = 2 * kecepatan * kecepatan *
            sin (sudut * 3.14 / 180) *
            cos (sudut * 3.14 / 180) /9.8;

      cout << “jarak = “ << jarak << “\n”;

      return0;
}
```

Tampilan Dev C++





	Akhir Kode Sumber

4.14.3 Fungsi gets ()

C menyediakan fungsi bernama **gets()** yang berguna untuk memasukkan data string dari keyboard. Bentuk pemakaiannya:

gets (*variabelString*);

Catatan 	1. Bentuk di atas setara dengan scanf (“%s”, variabelString) 2. Prototype fungsi **gets()** terdapat pada berkas *header***stdio.h**.

Perlu diketahuai, **gets()** membaca seluruh karakter yang dimasukkan dari keyboard sampai tombol Enter ditekan. Oleh karena itu harus dipastikan bahwa ukuran variable string harus mampu menyimpan seluruh karkter dalam string plus 1 (yang digunakan untuk menyimpan karakter NULL).

Contoh berikut memperlihatkan pemakaian **gets()**:

Hasil pengeksekusian program dapat dilihat pada gambar berikut:

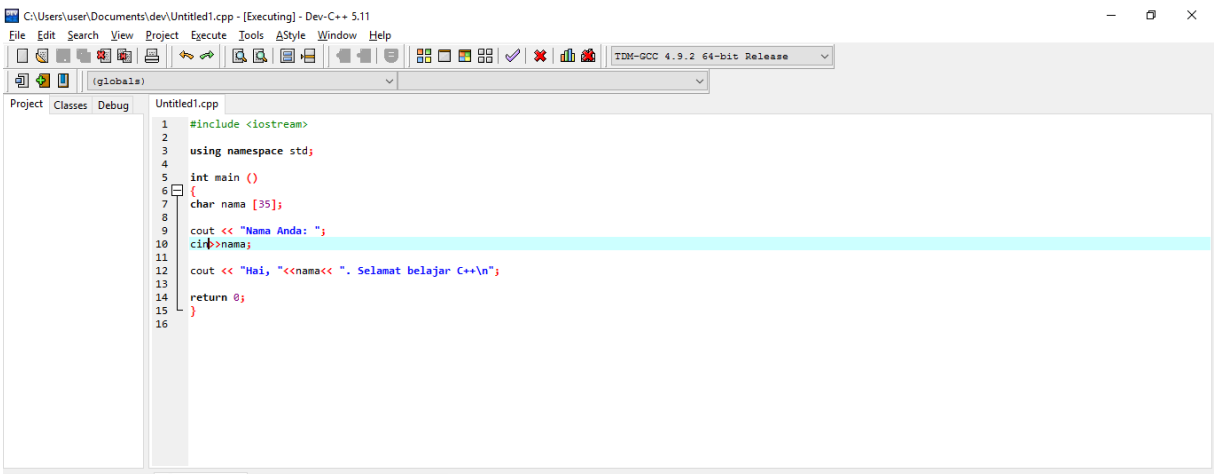
```
C:\progc>bacanama ↵
Nama Anda: Aulia↵
Hai, Aulia. Selamat belajar C
C:\progc>
```

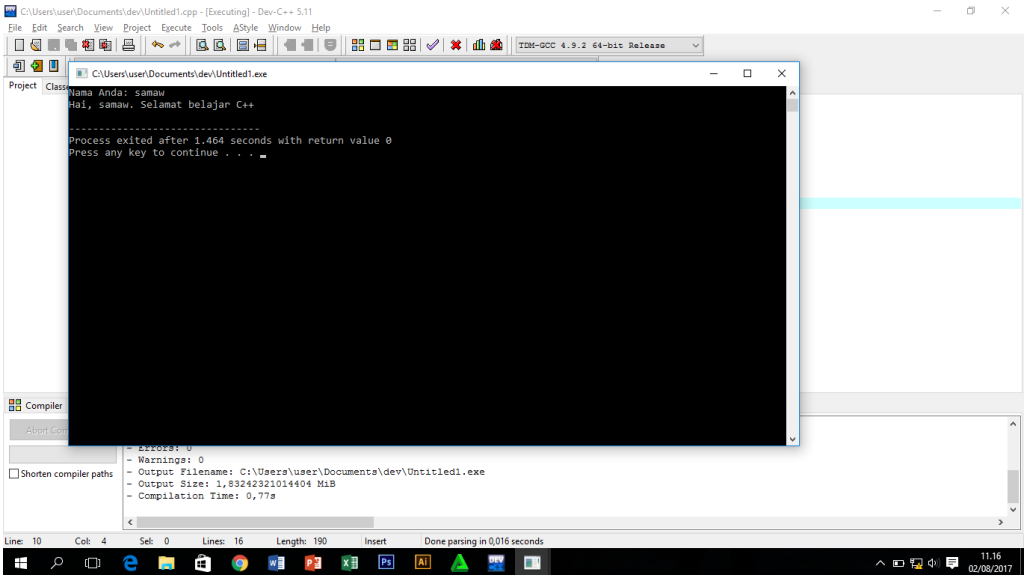
Pada C++, string juga dapat dibaca melalui **cin**. Contoh berikut memperhatikan program C++ yang setara dengan program bacanama.cdi depan:

Kode Sumber C++ : -**.cpp**

```
#include <iostream.h>

int main ()
{
      char nama [35];

      cout << “Nama Anda: “;
      gets >>nama;

      cout << “Hai, “<<nama<< “. Selamat belajar C++\n”;

      return0;
}
```

Tampilan Dev C++





	Akhir Kode Sumber

BAB 5

OPERASI SELEKSI

5.1 Mengenal Struktur Seleksi pada C++

Berbagai contoh seleksi dalam bentuk aloritma telah anda pelajari pada Bab 2 dan 3. Bagaimana bentuk seleksi seperti itu bila diterjemahkan dalam bentuk pemrograman C++? Pada bab inilah anda akan mempelajarinnya. Namun perlu diketahui, secara garis besar C++ menyediakan dua buah pernyataan yang terkait dengan seleksi, yaitu pernyataan if

dan switch.

5.1.1 Penerjemahan Bentuk JIKA .. AKHIR-JIKA

Tabel 1.1 memperlihatkan pedoman pengonversian bentuk pseudokode JIKA .. AKHIR_JIKA ke dalam pernyataan C++.

Tabel 1.1*Konversi bentuk JIKA .. AKHIR-JIKA*

Pseudokode	Kode C

JIKA kondisi benar MAKA
 	
Pernyataan_1
. . .
SEBALIKNYA
 	
Pernyataan_2
. . .
AKHIR – JIKA
	If (kondisi)
	{
	Pernyataan_1;
	. . .
	}
	Else
	{
	Pernyataa_2;
	. . .
	}

JIKA kondisi benar MAKA
 	
Pernyataan
. . .
AKHIR – JIKA
	If (kondisi)
	{
	Pernyataan;
	. . .
	}

Catatan	1. Kondisi pada pernyataan if harus ditulis dalam tanda kurung. Dengan kata lain, tanda kurung harus disertakan untuk melingkupi kondisi pada pernyataan if. 2. Pada Bahasa C++, Jika di dalam ( ) hanya terdapat sebuah pernyataan, tanda tersebut bisa dibuang

5.1.2 Penerjemahan Bentuk COCOK .. AKHIR – COCOK

C++ juga mendukung pernyataan yang setara dengan bentuk COCOK . . AKHIR – COCOK. Penerjemahannya dapat dilihat pada table 1.1.

Tabel 1.2*Konversi bentuk COCOK .. AKHIR – COCOK*

Pseudokode	Kode C++

COCOK nilai
DENGAN nilai1 MAKA
Pernyataan-11
Pernyataan-12
DENGAN nilai2 MAKA
Pernyataan-21
Pernyataan-22
DENGAN nilai3 MAKA
Pernyataan-31
Pernyataan-32
LAINNYA
Pernyataan-n1
Pernyataan-n2
AKHIR-COCOK
	Switch (nilai)
	{
	Case nilai1 :
	Pernyataan_11;
	Pernyataan_12;
	Break;
	Case nilai2 :
	Pernyataan_21;
	Pernyataan_22;
	Break;
	Case nilai3 :
	Pernyataan_31;
	Pernyataan_32;
	Break;
	Default :
	Pernyataan_n1;
	Pernyataan_n2;
	Break;
	}

Pseudokode	Kode C++

COCOK nilai
DENGAN nilai1 MAKA
Pernyataan-11
Pernyataan-12
DENGAN nilai1 MAKA
Pernyataan-21
Pernyataan-22
DENGAN nilai1 MAKA
Pernyataan-31
Pernyataan-32
AKHIR-COCOK
	Switch (nilai)
	{
	Case nilai1:
	Pernyataan_11;
	Pernyataan_12;
	Break;
	Case nilai2:
	Pernyataan_21;
	Pernyataan_22;
	Break;
	Case nilai3:
	Pernyataan_31;
	Pernyataan_32;
	Break;
	}

Catatan

1. Pernyataan **break** pada pernyataan **switch** digunakan agar eksekusi dilanjutkan ke pernyataan yang terletak sesudah pernyataan **switch**.
2. Bagian **default** pada **switch** bersifat opsional. Bagian ini hanya dijalankan kalau tak ada satupun bagian **case** yang cocok dengan nilai pada **switch**.
3. Pernyataan **break** pada bagian **default** bisa ditiadakan tanpa mengubah hasil.

5.2 Seleksi Sederhana

Beberapa contoh seleksi sederhana baik dalam bentuk algoritma maupun program akan dibahas pada sub bab ini.

Contoh 5.1 [Menentukan bilangan genap atau ganjil] Sebuah bilangan dapat ditentukan sebagai bilangan genap jika bilangan tersebut abis dibagi dengan 2. Jika tidak habis dibagi 2 maka bilangan disebut diangga p sebagai bilangan ganjil. Buatlah algoritma maupun programnya.

Algoritma :

Algoritma untuk menentukan bilangan genap atau ganjil adalah seperti berikut :

1. masukkan (bilangan)

2. JIKA sisa_pembagian (bilangan, 2) = 0 MAKA

   tampilkan (“Bilangan genap”)

   SEBALIKNYA

   tampilkan(“Bilangan ganjil”)

   AKHIR-JIKA

Program:

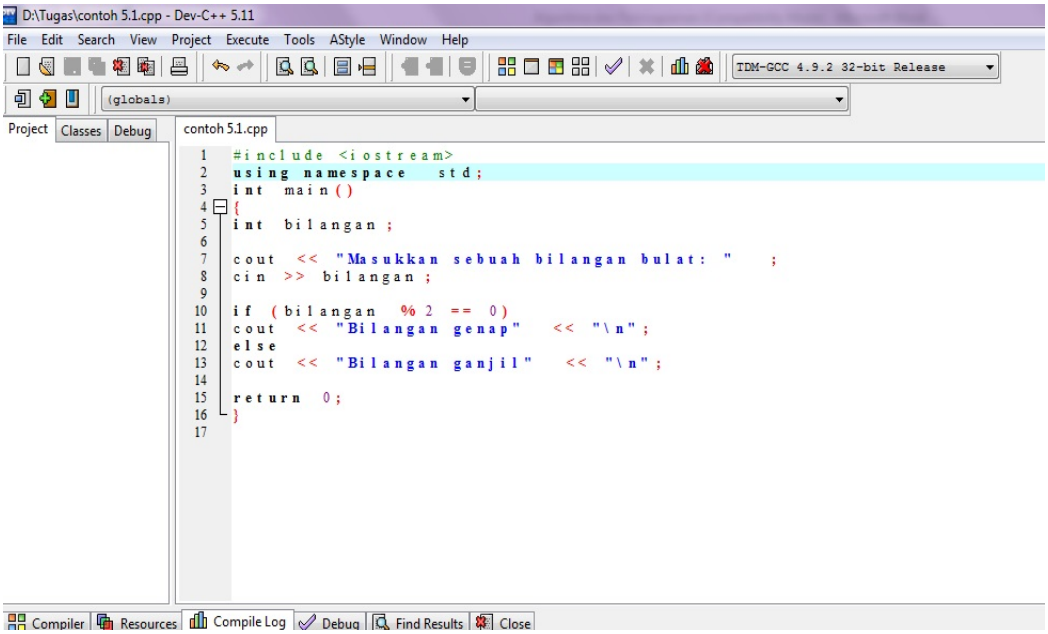
Adapun implementasi dalam C++ adalah sebagai berikut :

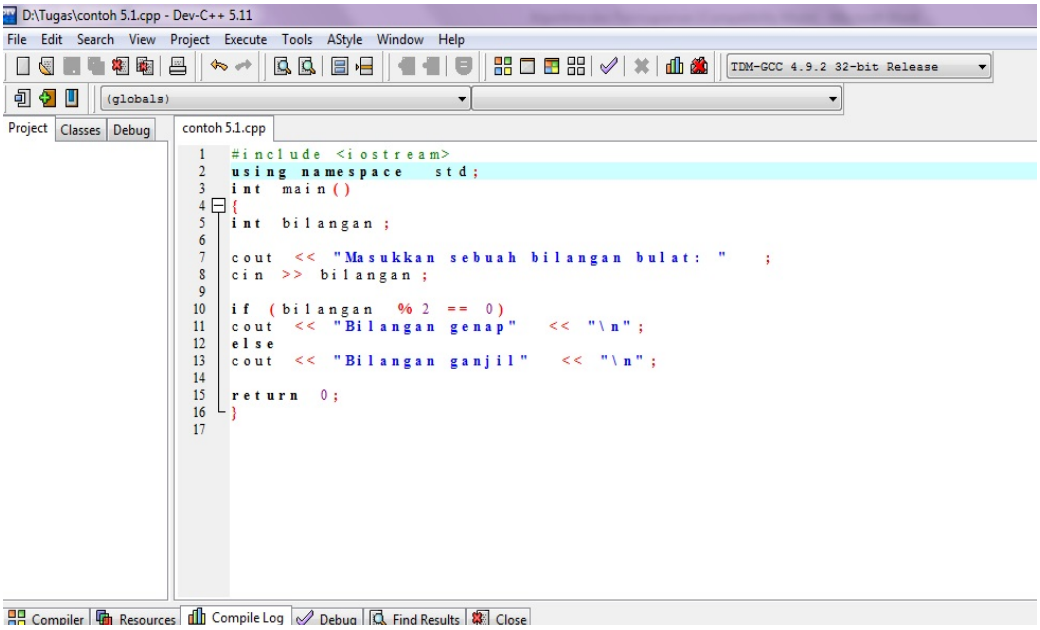
Kode Sumber : **genap.cpp**

```
#include <iostream.h>
int main()
{
      int bilangan;

      cout << "Masukkan sebuah bilangan bulat: ";
      cin >> bilangan;

      if (bilangan % 2 == 0)
            cout << "Bilangan genap" << "\n";
      else
            cout << "Bilangan ganjil" << "\n";

      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
Using namespace std;

int main()
{
      int bilangan;

      cout << "Masukkan sebuah bilangan bulat: ";
      cin >> bilangan;

      if (bilangan % 2 == 0)
            cout << "Bilangan genap" << "\n";
      else
            cout << "Bilangan ganjil" << "\n";

      return 0;
}
```


D:\Tugas\contoh 5.1.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
(globals)
Project Classes Debug
contoh 5.1.cpp
#include <iostream>
using namespace std;
int main()
{
int bilangan;
cout << "Masukkan sebuah bilangan bulat: ";
cin >> bilangan;
if (bilangan % 2 == 0)
cout << "Bilangan genap" << "\n";
else
cout << "Bilangan ganjil" << "\n";
return 0;
}
Compiler Resources Compile Log Debug Find Results Close


D:\Tugas\contoh 5.1.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
(globals)
Project Classes Debug
contoh 5.1.cpp
#include <iostream>
using namespace std;
int main()
{
int bilangan;
cout << "Masukkan sebuah bilangan bulat: ";
cin >> bilangan;
if (bilangan % 2 == 0)
cout << "Bilangan genap" << "\n";
else
cout << "Bilangan ganjil" << "\n";
return 0;
}
Compiler Resources Compile Log Debug Find Results Close

Akhir Kode Sumber

Contoh 5.2 [Menentukan bilangan terbesar di antara dua buah bilangan]Dua buah bilangan bulat dimasukkan dari keyboard. Bagaimana cara menampilkan bilangan yang terbesar di antara kedua bilangan tersebut?

Ada banyak cara untuk memberikan solusi pada persoalan ini. Kedua cara itu akan dibahas.

Cara 1 :

Salah satu algoritma untuk menyelesaikan masalah ini pernah dibahas pada Bab 2. Algoritmanya adalah sebagai berikut :

1. masukkan (x, y).
2. terbesar ← x // Asumsi bahwa x adalah yang terbesar
3. JIKA terbesar < y MAKA

   terbesar ← y

   AKHIR-JIKA
4. tampilkan (terbesar)

Implementasi dalam C++ :

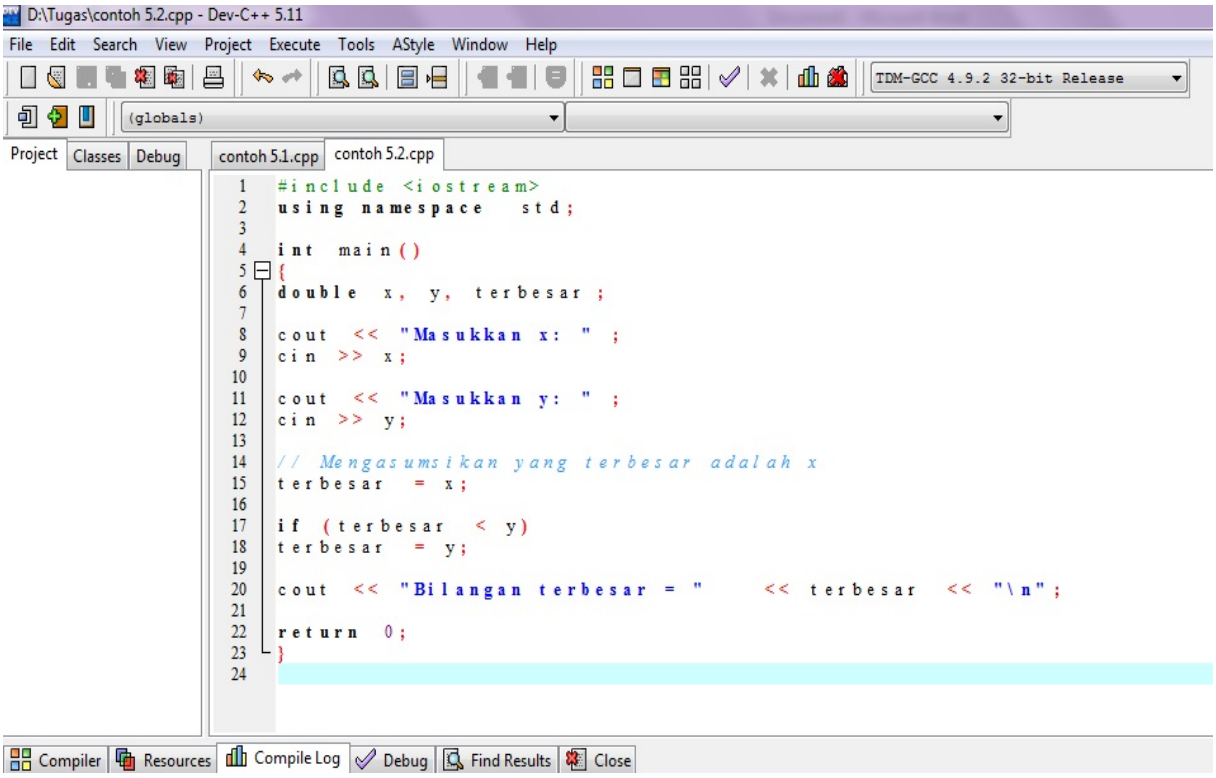
Kode Sumber : **terbesar1.cpp**

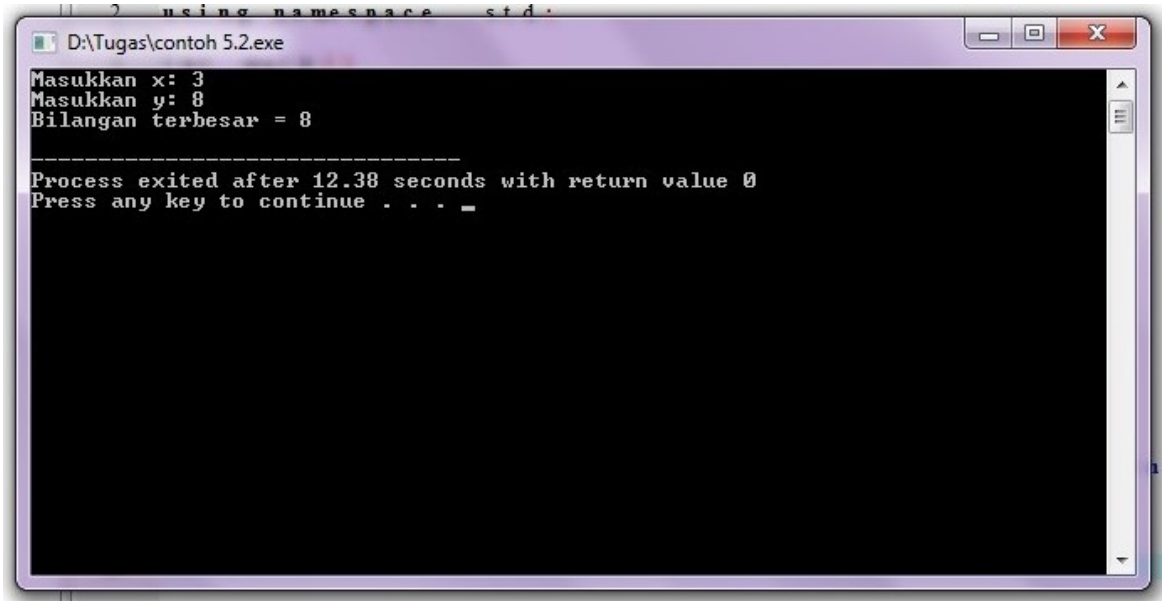
```
#include <iostream.h>
int main()
{
      double x, y, terbesar;

      cout << "Masukkan x: ";
      cin >> x;

      cout << "Masukkan y: ";
      cin >> y;

      // Mengasumsikan yang terbesar adalah x
      terbesar = x;

      if (terbesar < y)
            terbesar = y;

      cout << "Bilangan terbesar = "  << terbesar << "\n";

      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
double x, y, terbesar;

cout << "Masukkan x: ";
cin >> x;

cout << "Masukkan y: ";
cin >> y;

// Mengasumsikan yang terbesar adalah x
terbesar = x;

if (terbesar < y)
terbesar = y;

cout << "Bilangan terbesar = "  << terbesar << "\n";
```

```
return 0;
}
```





	Akhir Kode Sumber

Cara 2 :

Cara kedua dilakukan tanpa melibatkan variable lain. Algoritmanya adalah sebagai berikut :

1. masukkan (x, y).
2. JIKA x > y MAKA

   tampilkan (“terbesar yaitu”, x)

   SEBALIKNYA

   tampilkan (“terbesar yaitu”, y)

   AKHIR-JIKA

Program C++ berdasarkan algoritma tersebut dapat dilihat di bawah ini.

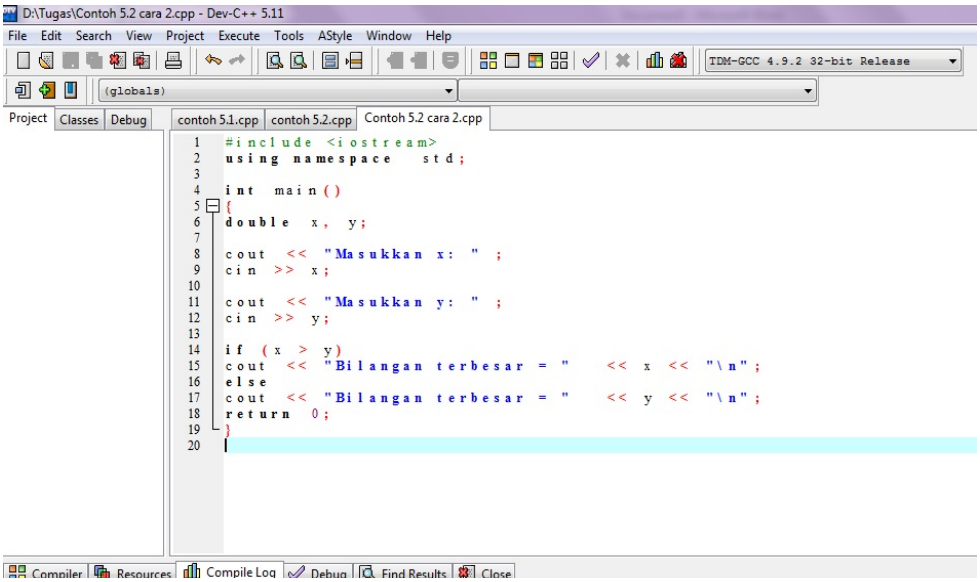
Kode Sumber : **terbesar2.cpp**

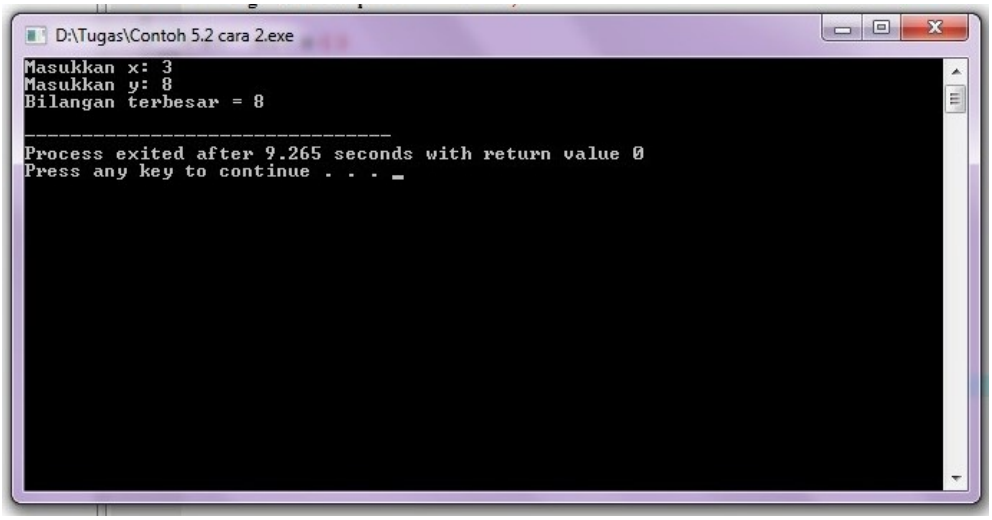
```
#include <iostream.h>

int main()
{
      double x, y;

      cout << "Masukkan x: ";
      cin >> x;

      cout << "Masukkan y: ";
      cin >> y;

      if (x > y)
            cout << "Bilangan terbesar = " << x << "\n";
      else
            cout << "Bilangan terbesar = " << y << "\n";
      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
      double x, y;

      cout << "Masukkan x: ";
      cin >> x;

      cout << "Masukkan y: ";
      cin >> y;
```

```
        if (x > y)
              cout << "Bilangan terbesar = " << x << "\n";
        else
              cout << "Bilangan terbesar = " << y << "\n";
        return 0;
}
```





Akhir Kode Sumber

5.3 Seleksi dengan Kondisi Majemuk

Seringkali suatu kondisi yang dijadikan sebagai pengambil keputusan dalam suatu seleksi tidak sesederhana pada dua contoh di depan, melainkan melibatkan lebih dari sebuah kondisi. Beberapa contoh akan memperjelas hal ini.

Contoh 5.3 [Mevalidasi nilai ujian] Buatlah algoritma maupun program yang meminta sebuah nilai ujian dimasukkan dari keyboard dan memvalidasi nilai tersebut.

Berikan komentar “Absah” jika nilai tersebut berada antara 0 sampai dengan 100 dan berikan komentar “Tidak abash” untuk keadaan sebaliknya.

Algoritma :

Algoritma untuk menyelesaikan masalah di atas adalah seperti berikut :

1. Masukkan (nilai).
2. JIKA nilai > 0 dan nilai < 100 MAKA

    tampilkan (“Benar”)

    SEBALIKNYA

    tampilkan (“Salah “)

    AKHIR-JIKA

Program:

Program C++ berdasarkan algoritma di atas dapat dilihat dibawah ini.

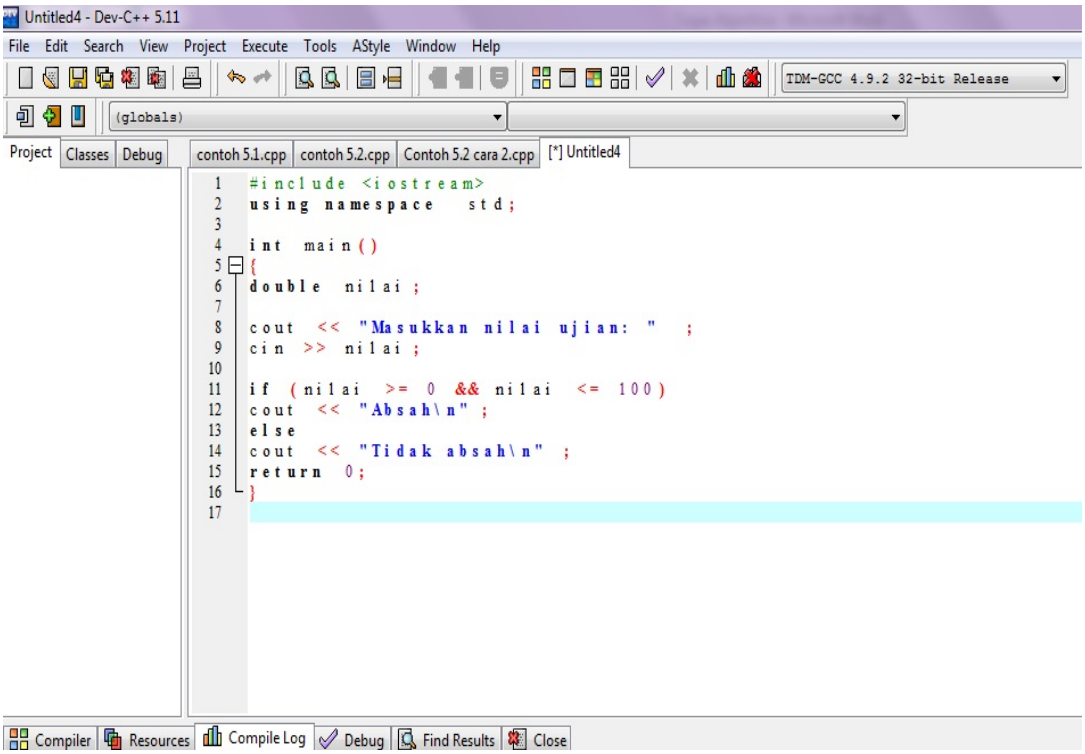
Kode Sumber : **valid.cpp**

```
#include <iostream.h>
int main()
{
      double nilai;

      cout << "Masukkan nilai ujian: ";
      cin >> nilai;

      if (nilai >= 0 && nilai <= 100)
            cout << "Benar\n";
      else
            cout << "Salah\n";
      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
      double nilai;

      cout << "Masukkan nilai ujian: ";
      cin >> nilai;
```

```
        if (nilai >= 0 && nilai <= 100)
              cout << "Absah\n";
        else
              cout << "Tidak absah\n";
        return 0;
}
```



	Akhir Kode Sumber

Contoh 5.4 [Menentukan huruf kapital atau bukan] Buatlah algoritma maupun pemrogram yang meminta sebuah karakter dimasukkan dari keyboard dan kemudian memberikan keterangan karakter tersebut termasuk huruf kapital atau bukan.

Algoritma:

Algoritma untuk menyelesaikan masalah di atas adalah seperti berikut:

1. Masukkan(karakter).
2. JIKA karakter ≥ “A” dan karater ≤ “Z” MAKA

    tampilkan(“Termasuk huruf kapital”)

    SEBALIKNYA

    tampilkan(“Bukan huruf kapital”)

    AKHIR-JIKA

Program :

Implementasi dalam C++:

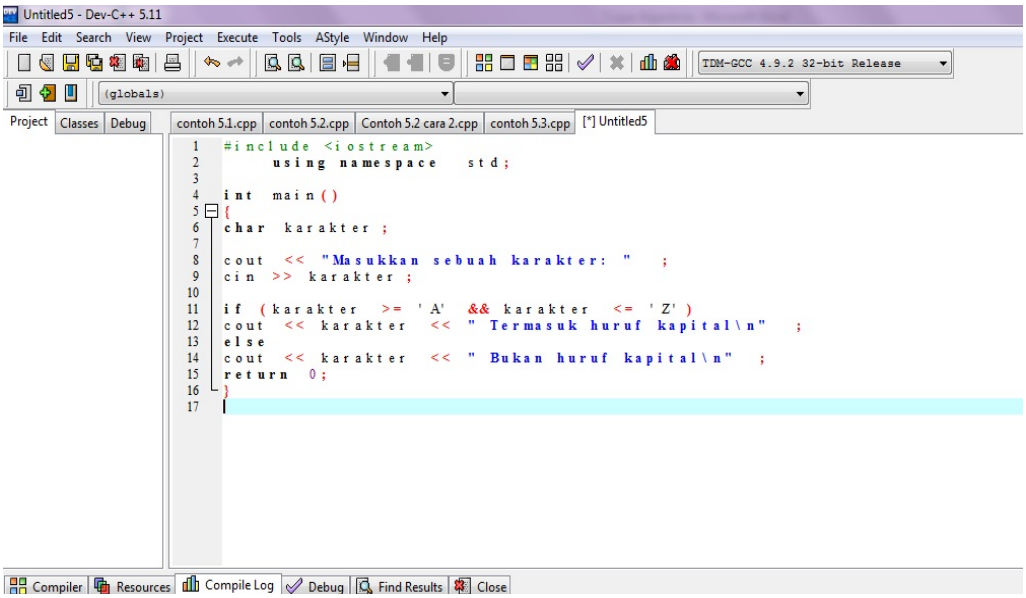
Kode Sumber : **kapital.cpp**

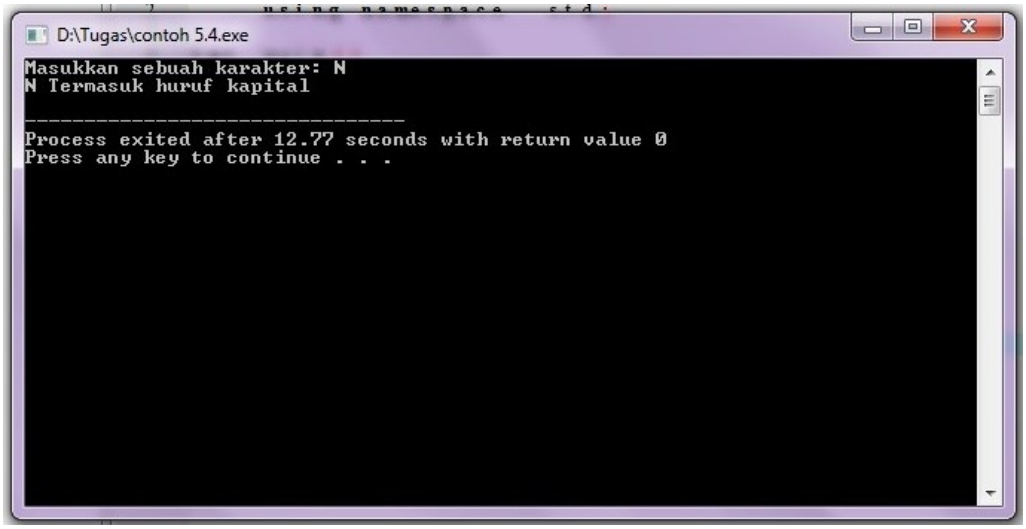
```
#include <iostream.h>

int main()
{
      char karakter;

      cout << "Masukkan sebuah karakter: ";
      cin >> karakter;

      if (karakter >= 'A' && karakter <= 'Z')
            cout << karakter << " Termasuk huruf kapital\n";
      else
            cout << karakter << " Bukan huruf kapital\n";
      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
      char karakter;

      cout << "Masukkan sebuah karakter: ";
      cin >> karakter;

      if (karakter >= 'A' && karakter <= 'Z')
            cout << karakter << " Termasuk huruf kapital\n";
      else
            cout << karakter << " Bukan huruf kapital\n";
```

```
    return 0;
}
```





Akhir Kode Sumber

5.4 Seleksi Bersarang

Dalam berbagai persoalan, seringkali suatu bentuk seleksi bertingkat diperlukan. Bentuk seperti inilah yang dinamakan sebagai seleksi bersarang. Beberapa contoh dapat Anda pelajari.

Contoh 5.5 [Menentukan tahun kabisat]Buatlah algoritma dan program yang mula-mula membaca tahun dari keyboard dan kemudian menampilkan informasi kabisat atau bukan kabisat.

Algoritma:

Suatu tahun disebut tahun kabusat jika memenuhi kriteria berikut (Leestma, dkk., hal 1320;

a. Tahun tersebut habus dibagi 4, tetapi
b. Jika habis dibagi 100 maka tahun tersebut harus habis dibagi 400.

Berdasarkan informasi tersebut, dapat disusun algoritma seperti berikut:

1. Masukkan(tahun).
2. JIKA sisa_pembagian(tahun, 4) ≠ 0 MAKA
   tampilkan("Bukan kabisat")
   SEBALIKNYA
   JIKA sisa_pembagian(tahun, 100) = 0 dan
   sisa_pembagian(tahun,400) ≠ 0 MAKA
   tampilkan("Bukan kabisat")
   SEBALIKNYA
   tampilkan("Kabisat")
   AKHIR-JIKA
   AKHIR-JIKA

Program :

Implementasi dalam C++:

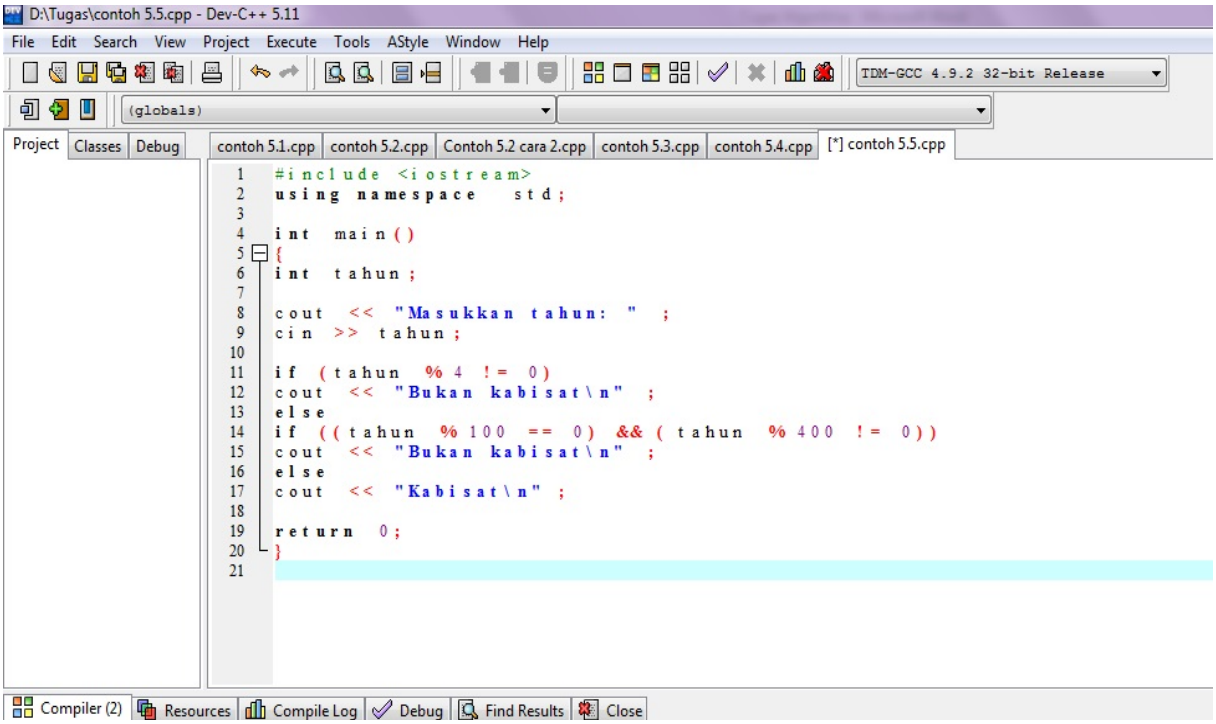
Kode Sumber : **kabisat.cpp**

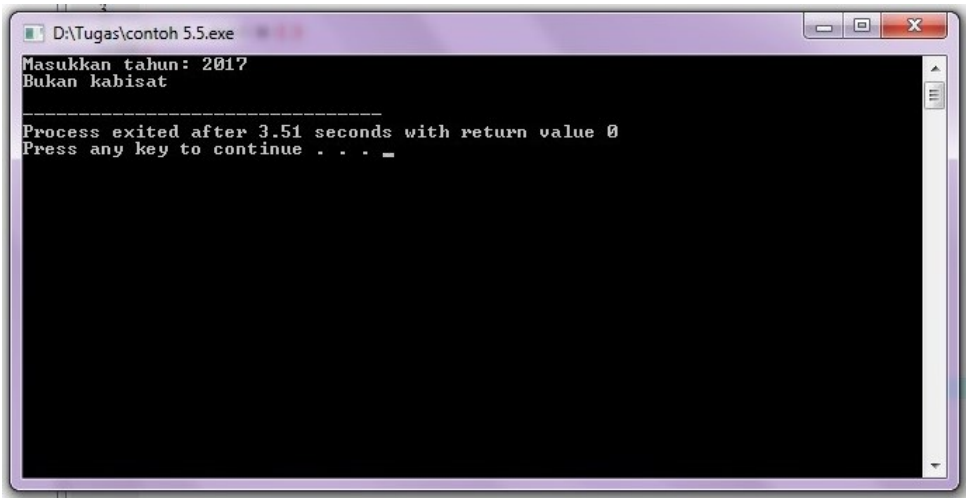
```
#include <iostream.h>

int main()
{
      int tahun;

      cout << "Masukkan tahun: ";
      cin >> tahun;

      if (tahun % 4 != 0)
            cout << "Bukan kabisat\n";
      else
            if ((tahun % 100 == 0) && (tahun % 400 != 0))
                  cout << "Bukan kabisat\n";
            else
                  cout << "Kabisat\n";
```

```
	return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
	int tahun;

	cout << "Masukkan tahun: ";
	cin >> tahun;

	if (tahun % 4 != 0)
		cout << "Bukan kabisat\n";
	else
		if ((tahun % 100 == 0) && (tahun % 400 != 0))
			cout << "Bukan kabisat\n";
		else
			cout << "Kabisat\n";

	return 0;
}
```





Akhir Kode Sumber

Contoh 5.6 [Menentukan predikat kelulusan] Tabel berikut memperlihatkan daftar predukat kelulusan seorang sarjana berdasarkan indeks prestasi kumulatifnya.

Tabel 5.3*Predikat kelulusan menurut IP kumulatif.*

IP Kumulatif	Predikat Kelulusan
2,00 ≤ IP ≤ 2.75	Lulus Memuaskan
2,75 < IP ≤ 3.50	Lulus Sangat Memuaskan
3,50 < IP ≤ 4,00	Lulus dengan Pujian

Buatlah algoritma dan pemrograman untuk menentukan predikat kelulusan seperti di atas.

Algoritma:

Algoritma berdasarkan tabel di atas adalah seperti berikut:

1. Masukkan(ip)
2. JIKA ip ≥ 2 dan ip ≤ 2, 75 MAKA
   tampilkan(“Lulus Memuaskan”)
   SEBALIKNYA
   JIKA ip > 2,75 dan up ≤ 3,50 MAKA
   tampilkan(“Lulus Sangat Memuaskan”)
   SEBALIKNYA
   JIKA ip > 3,50 dan ip ≤ 4,00 MAKA

tampilkan(“Lulus dengan Pujian”)

SEBALIKNYA

tampilkan(“Data IP tidak vailid”)

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

Program:

Implementasi dalam c++:

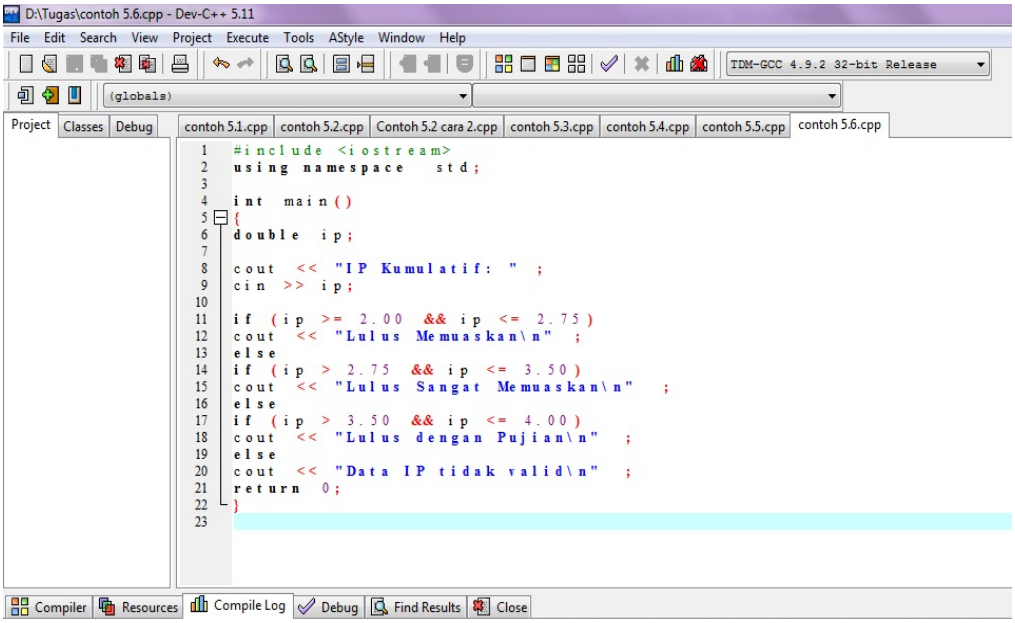
Kode Sumber : **predikat.cpp**

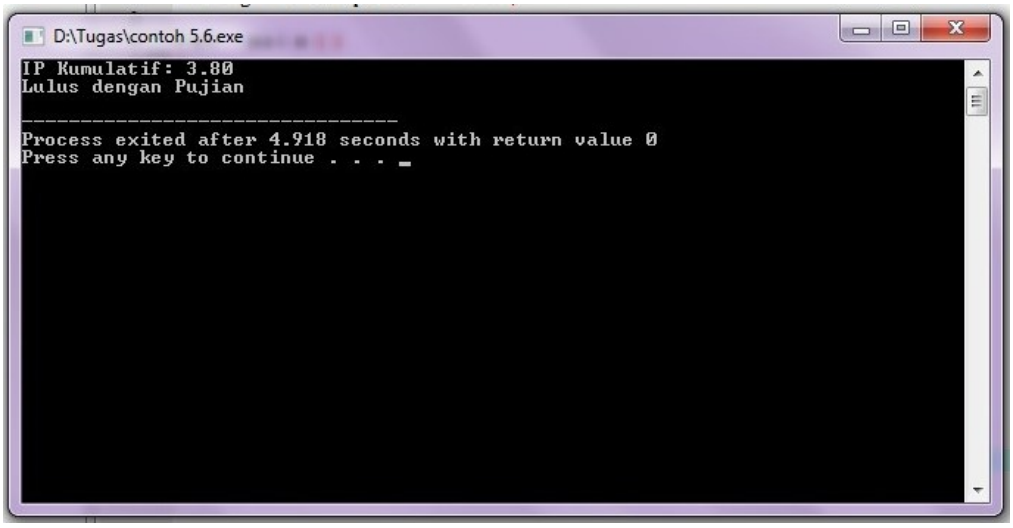
```
#include <iostream.h>

int main()
{
      double ip;

      cout << "IP Kumulatif: ";
      cin >> ip;

      if (ip >= 2.00 && ip <= 2.75)
            cout << "Lulus Memuaskan\n";
      else
            if (ip > 2.75 && ip <= 3.50)
                  cout << "Lulus Sangat Memuaskan\n";
            else
                  if (ip > 3.50 && ip <= 4.00)
                        cout << "Lulus dengan Pujian\n";
                  else
                        cout << "Data IP tidak valid\n";
      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
      double ip;

      cout << "IP Kumulatif: ";
```

```
	cin >> ip;

	if (ip >= 2.00 && ip <= 2.75)
		cout << "Lulus Memuaskan\n";
	else
		if (ip > 2.75 && ip <= 3.50)
			cout << "Lulus Sangat Memuaskan\n";
		else
			if (ip > 3.50 && ip <= 4.00)
				cout << "Lulus dengan Pujian\n";
			else
				cout << "Data IP tidak valid\n";
	return 0;
}
```





	Akhir Kode Sumber

Contoh 5.7[Menentukan nama bulan menurut angka] Buatlah diagram dan program yang membaca data kode bulan dari keyboard dan kemudian menampilkan nama bulan.

Cara pertama:

Cara pertama di pecahkan dengan menggunakan bentuk JIKA .. AKHIR –JIKA.

Algoritmanya sebagai berikut:

```
masukkan ( kode_bulan )
JIKA kode_bulan = 1 MAKA
tampilkan (''Januari'')
SEBALIKNYA
JIKA kode_bulan = 2 MAKA
tampilkan (''Februari'')
SEBALIKNYA
        JIKA kode_bulan = 3 MAKA
        tampilkan (''Maret '')
        SEBALIKNYA
        JIKA kode_bulan = 4 MAKA
                tampilkan (''April'')
        SEBALIKNYA
                JIKA kode_bulan = 5 MAKA
                tampilkan (''Mei'')
                SEBALIKNYA
                JIKA kode_bulan  = 6 MAKA
                        tampilkan (''Juni'')
                SEBALIKNYA
                        JIKA kode_bulan = 7 MAKA
                        tampilkan (''Juli'')
                        SEBALIKNYA
```

```
                    JIKA kode_bulan = 8 MAKA
                    tampilkan (''Agustus'')
                    SEBALIKNYA
                        JIKA kode_bulan = 9 MAKA
                        tampilkan (''September'')
                        SEBALIKNYA
                            JIKA kode_bulan = 10 MAKA
                                tampilkan (''Oktober'')
                            SEBALIKNYA
                                JIKA kode_bulan = 11 MAKA
                                    tampilkan (''November'')
                                SEBALIKNYA
                                    JIKA kode_bulan = 12 MAKA
                                        tampilkan (''Desember'')
                                    SEBALIKNYA
                                        tampilkan (''Salah kode bulan '')
                                AKHIR-JIKA
                                AKHIR-JIKA
                        AKHIR-JIKA
                        AKHIR-JIKA
                AKHIR-JIKA
                AKHIR-JIKA
            AKHIR-JIKA
            AKHIR-JIKA
        AKHIR-JIKA
        AKHIR-JIKA
```

Program:

Implementasi dalam C++

Kode Sumber : **bulan.cpp**

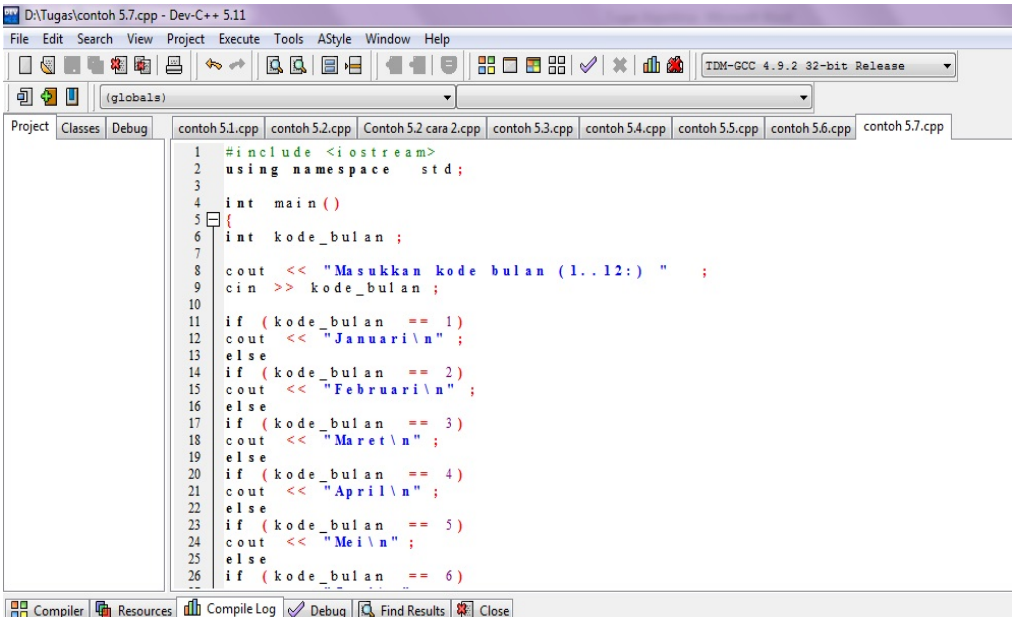
```
#include <iostream.h>

int main()
{
      int kode_bulan;

      cout << "Masukkan kode bulan (1..12:) ";
      cin >> kode_bulan;

      if (kode_bulan == 1)
            cout << "Januari\n";
      else
          if (kode_bulan == 2)
             cout << "Februari\n";
          else
          if (kode_bulan == 3)
             cout << "Maret\n";
          else
             if (kode_bulan == 4)
                cout << "April\n";
             else
                if (kode_bulan == 5)
                   cout << "Mei\n";
                else
                   if (kode_bulan == 6)
                      cout << "Juni\n";
                   else
                      if (kode_bulan == 7)
                         cout << "Juli\n";
                      else
                         if (kode_bulan == 8)
                            cout << "Agustus\n";
                         else
                            if (kode_bulan == 9)
                               cout << "Septemmber\n";
                            else
                               if (kode_bulan == 10)
                                  cout << "Oktober\n";
                               else
                                  if (kode_bulan == 11)
                                     cout << "November\n";
                                  else
                                     if (kode_bulan == 12)
                                        cout << "Desember\n";
                                        else
                                        cout  <<  "Salah  kode
                                        bulan\n";
```

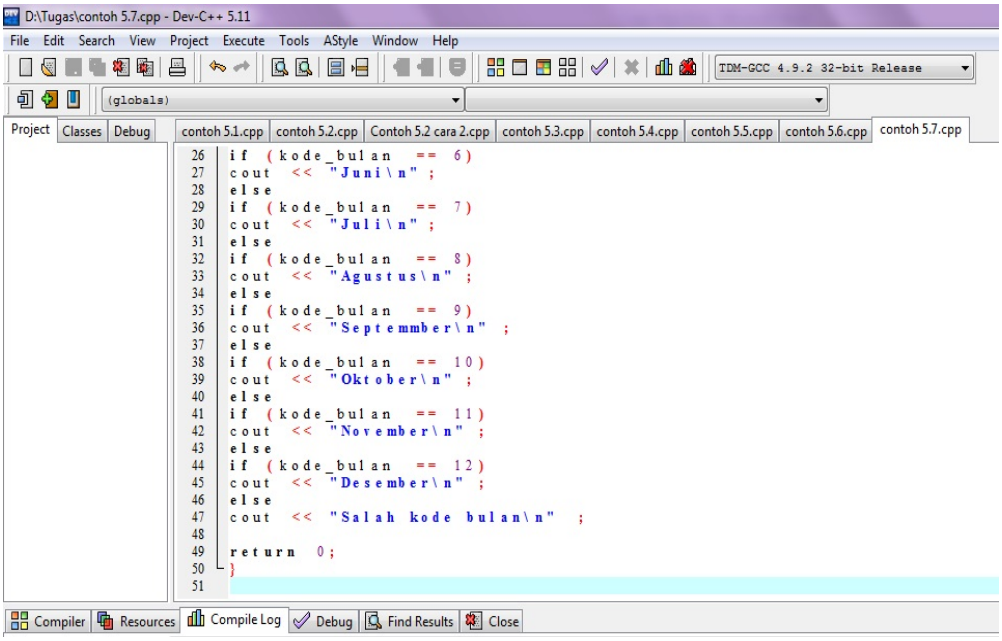
```
        return 0;
    }
```

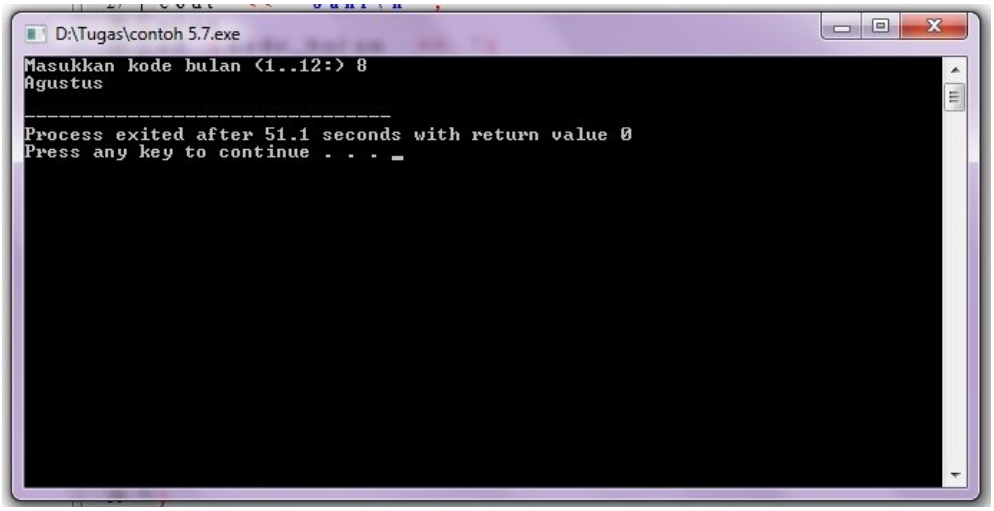
- **Jika Menggunakan DEV C++**

```
    #include <iostream>
    using namespace std;

    int main()
    {
        int kode_bulan;

        cout << "Masukkan kode bulan (1..12:) ";
        cin >> kode_bulan;

        if (kode_bulan == 1)
            cout << "Januari\n";
        else
            if (kode_bulan == 2)
               cout << "Februari\n";
            else
            if (kode_bulan == 3)
               cout << "Maret\n";
            else
               if (kode_bulan == 4)
                  cout << "April\n";
               else
                  if (kode_bulan == 5)
                     cout << "Mei\n";
                  else
                     if (kode_bulan == 6)
                        cout << "Juni\n";
                     else
                        if (kode_bulan == 7)
                           cout << "Juli\n";
                        else
                           if (kode_bulan == 8)
                              cout << "Agustus\n";
                           else
                              if (kode_bulan == 9)
                                 cout << "Septemmber\n";
                              else
                                 if (kode_bulan == 10)
                                    cout << "Oktober\n";
                                 else
                                    if (kode_bulan == 11)
                                       cout << "November\n";
                                    else
```

```
                                        if (kode_bulan == 12)
                                           cout << "Desember\n";
                                           else
                                           cout   <<   "Salah   kode
                                           bulan\n";

      return 0;
}
```







	Akhir Kode Sumber

Cara Kedua:

Persoalan penentuan nama bulan juga bisa dipecahkan dengan menggunakan bentuk COCOK..AKHIR-COCOK. Algoritmanya sebagai berikut:

1. masukkan (kode_bulan).
2. COCOK kode_bulan

    DENGAN 1 MAKA

        tampilkan(“Januari’’)

    DENGAN 2 MAKA

        tampilkan(“Februari’’)

    DENGAN 3 MAKA

        tampilkan(“Maret’’)

    DENGAN 4 MAKA

        tampilkan(“April’’)

    Dengan 5 MAKA

        tampilkan (“Mei’’)

    DENGAN 6 MAKA

        tampilkan(“Juni’’)

    DENGAN 7 MAKA

        tampilkan(“Juli’’)

DENGAN 8 MAKA

tampilkan(“Agustus’’)

DENGAN 9 MAKA

tampilkan(“September’’)

DENGAN 10 MAKA

tampilkan(“Oktober’’)

DENGAN 11 MAKA

tampilkan(“November’’)

DENGAN 12 MAKA

tampilkan(“Desember’’)

LAINNYA

tampilkan(“Salah kode bulan’’)

AKHIR-COCOK

Algoritma diatas dimplementasikan dalam program C++ sebagai berikut:

Kode Sumber : **bulan2.cpp**

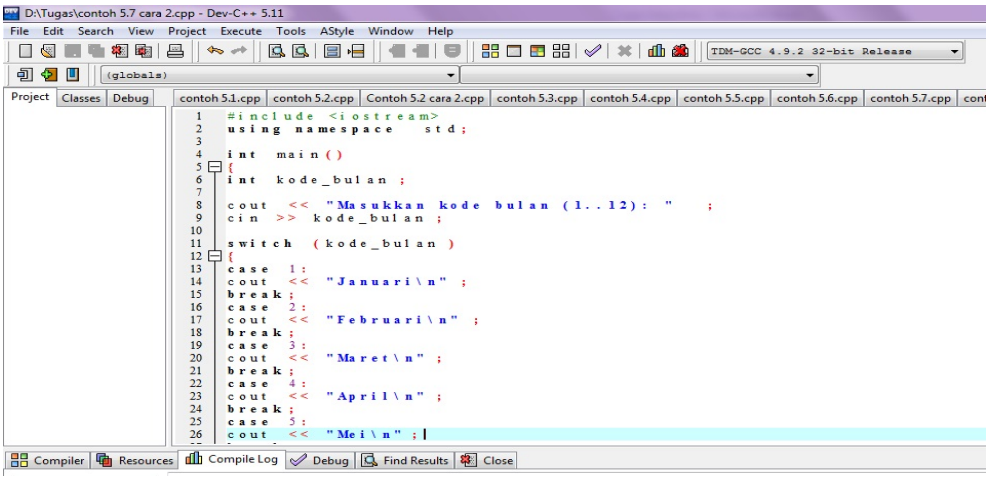
```
#include <iostream.h>

int main()
{
      int kode_bulan;

      cout << "Masukkan kode bulan (1..12): ";
      cin >> kode_bulan;

      switch (kode_bulan)
      {
            case 1:
                  cout << "Januari\n";
                  break;
            case 2:
                  cout << "Februari\n";
                  break;
            case 3:
                  cout << "Maret\n";
                  break;
            case 4:
                  cout << "April\n";
                  break;
            case 5:
                  cout << "Mei\n";
```

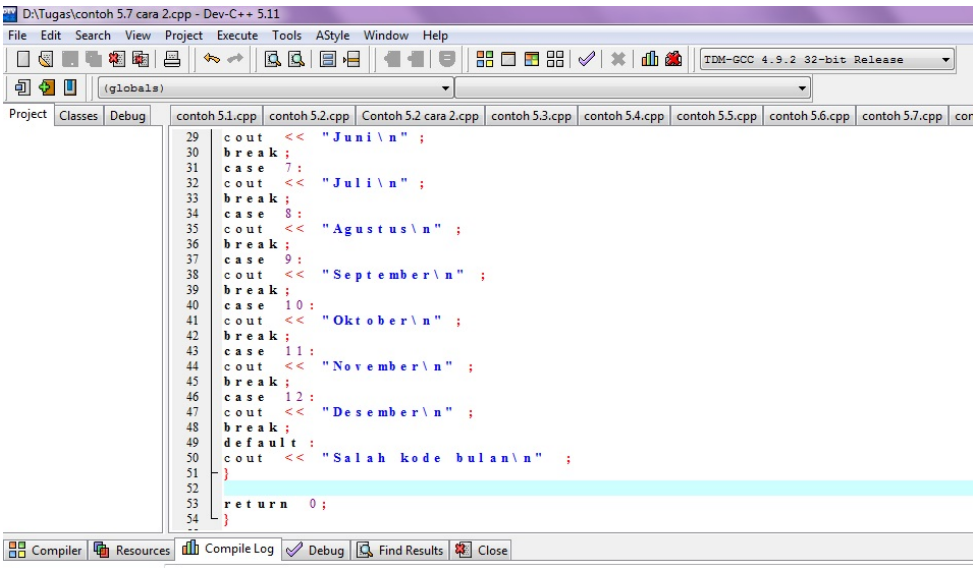
```
                    break;
               case 6:
                    cout << "Juni\n";
                    break;
               case 7:
                    cout << "Juli\n";
                    break;
               case 8:
                    cout << "Agustus\n";
                    break;
               case 9:
                    cout << "September\n";
                    break;
               case 10:
                    cout << "Oktober\n";
                    break;
               case 11:
                    cout << "November\n";
                    break;
               case 12:
                    cout << "Desember\n";
                    break;
               default:
                    cout << "Salah kode bulan\n";
          }

          return 0;
     }
```

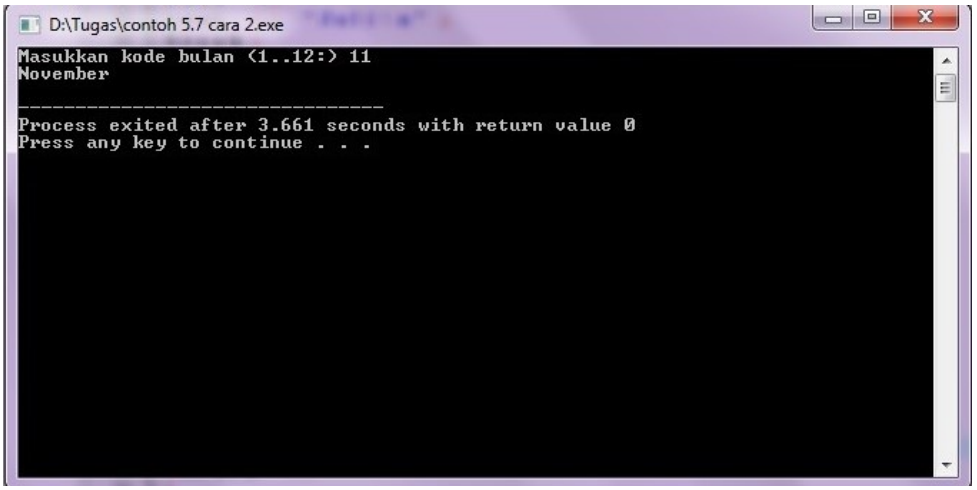
- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
     int kode_bulan;

     cout << "Masukkan kode bulan (1..12): ";
     cin >> kode_bulan;

     switch (kode_bulan)
     {
          case 1:
               cout << "Januari\n";
               break;
          case 2:
               cout << "Februari\n";
               break;
          case 3:
```

```
                cout << "Maret\n";
                break;
        case 4:
                cout << "April\n";
                break;
        case 5:
                cout << "Mei\n";
                break;
        case 6:
                cout << "Juni\n";
                break;
        case 7:
                cout << "Juli\n";
                break;
        case 8:
                cout << "Agustus\n";
                break;
        case 9:
                cout << "September\n";
                break;
        case 10:
                cout << "Oktober\n";
                break;
        case 11:
                cout << "November\n";
                break;
        case 12:
                cout << "Desember\n";
                break;
        default:
                cout << "Salah kode bulan\n";
    }

    return 0;
}
```







Akhir Kode Sumber

Contoh 5.8 [Menentukan jumlah hari] Buatlah algoritma dan program yang meminta data bulan (1...12) dimasukkan dari keyboard dan kemudian menentukan jumlah hari dalam bulan tersebut.

Cara Pertama:

Persoalan di atas dapat dipecahkan dengan menggunakan sederetan JIKA...AKHIR-JIKA. Algoritmanya sebagai berikut:

1. Masukkan(kode_bulan).
2. JIKA kode_bulan = 2 MAKA

       tampilkan(“Jumlah hari 28 atau 29”)

   SEBALIKNYA

       JIKA kode_bulan = 1 atau 3 atau 5 atau 7 atau 8

atau 10 atau 12 MAKA

tampilkan(“Jumlah hari 31”)

SEBALIKNYA

JIKA kode_bulan = 4 atau 6 atau 9 atau 11 MAKA

tampilkan(“Jumlah hari 30”)

SEBALIKNYA

tampilkan(“Salah kode bulan”)

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

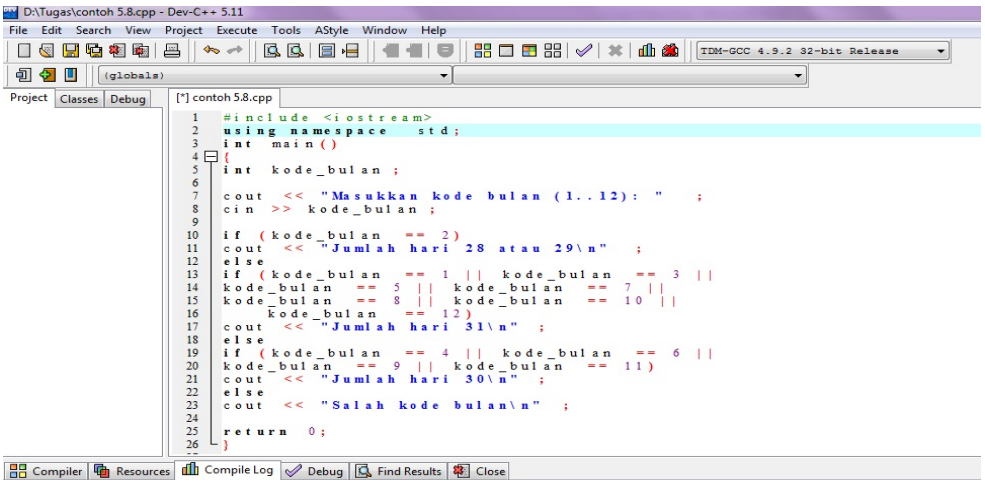
Implementasi program menurut algoritma di atas adalah sebagai berikut:

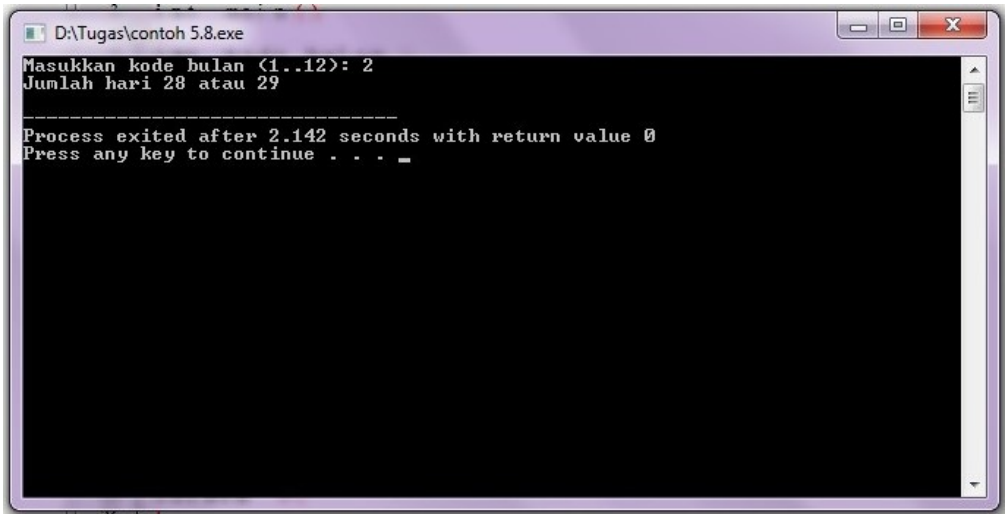
Kode Sumber : **jhari.cpp**

```
#include <iostream.h>

int main()
{
      int kode_bulan;

      cout << "Masukkan kode bulan (1..12): ";
      cin >> kode_bulan;

      if (kode_bulan == 2)
            cout << "Jumlah hari 28 atau 29\n";
      else
            if (kode_bulan == 1 || kode_bulan == 3 ||
            kode_bulan == 5 || kode_bulan == 7 ||
            kode_bulan == 8 || kode_bulan == 10 ||
                kode_bulan == 12)
                  cout << "Jumlah hari 31\n";
            else
                  if (kode_bulan == 4 || kode_bulan == 6 ||
                  kode_bulan == 9 || kode_bulan == 11)
                        cout << "Jumlah hari 30\n";
                  else
                        cout << "Salah kode bulan\n";

      return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;
int main()
{
	int kode_bulan;

	cout << "Masukkan kode bulan (1..12): ";
	cin >> kode_bulan;

	if (kode_bulan == 2)
		cout << "Jumlah hari 28 atau 29\n";
	else
		if (kode_bulan == 1 || kode_bulan == 3 ||
		kode_bulan == 5 || kode_bulan == 7 ||
		kode_bulan == 8 || kode_bulan == 10 ||
		    kode_bulan == 12)
			cout << "Jumlah hari 31\n";
		else
			if (kode_bulan == 4 || kode_bulan == 6 ||
			kode_bulan == 9 || kode_bulan == 11)
				cout << "Jumlah hari 30\n";
			else
				cout << "Salah kode bulan\n";

	return 0;
}
```





Akhir Kode Sumber

Cara kedua:

Cara lain untuk menentukan jumlah hari dalam suatu bulan adalah dengan menggunakan bentuk COCOK...AKHIR-COCOK. Algoritmanya sebagai berikut:

1. masukkan(kode_bulan)
2. COCOK kode_bulan

    DENGAN 2 MAKA

        tampilkan(“Jumlah hari 28 atau 29”)

    DENGAN 1 MAKA

        tampilkan(“Jumlah hari 31”)

    DENGAN 3 MAKA

        tampilkan(“Jumlah hari 31”)

    DENGAN 5 MAKA

        tampilkan(“Jumlah hari 31”)

    DENGAN 7 MAKA

        tampilkan(“Jumlah hari 31”)

    DENGAN 8 MAKA

        tampilkan(“Jumlah hari 31”)

    DENGAN 10 MAKA

        tampilkan(“Jumlah hari 31”)

    DENGAN 12 MAKA

        tampilkan(“Jumlah hari 31”)

DENGAN 4 MAKA

tampilkan(“Jumlah hari 30”)

DENGAN 6 MAKA

tampilkan(“Jumlah hari 30”)

DENGAN 9 MAKA

tampilkan(“Jumlah hari 30”)

DENGAN 11 MAKA

tampilkan(“Jumlah hari 30”)

LAINNYA

tampilkan(“Salah kode bulan”)

AKHIR-COCOK

Algoritma di atas dapat diimplementasikan ke dalam program C++ dengan menggunakan **switch,** sebagaimana dapat dilihat di bawah ini.

Kode Sumber : **jhari2.cpp**

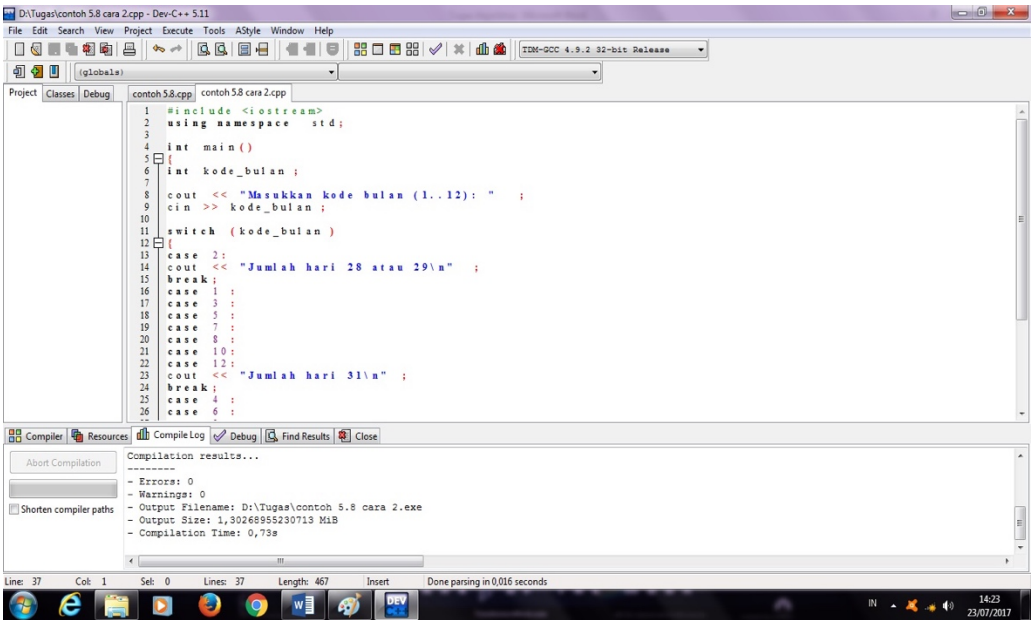
```
#include <iostream.h>

int main()
{
      int kode_bulan;

      cout << "Masukkan kode bulan (1..12): ";
      cin >> kode_bulan;

      switch (kode_bulan)
      {
            case 2:
                  cout << "Jumlah hari 28 atau 29\n";
                  break;
            case 1 :
            case 3 :
            case 5 :
            case 7 :
            case 8 :
            case 10:
            case 12:
                  cout << "Jumlah hari 31\n";
```

```
                    break;
               case 4 :
               case 6 :
               case 9 :
               case 11:
                    cout << "Jumlah hari 30\n";
                    break;
               default:
                    cout << "Salah kode bulan\n";
          }

          return 0;
     }
```

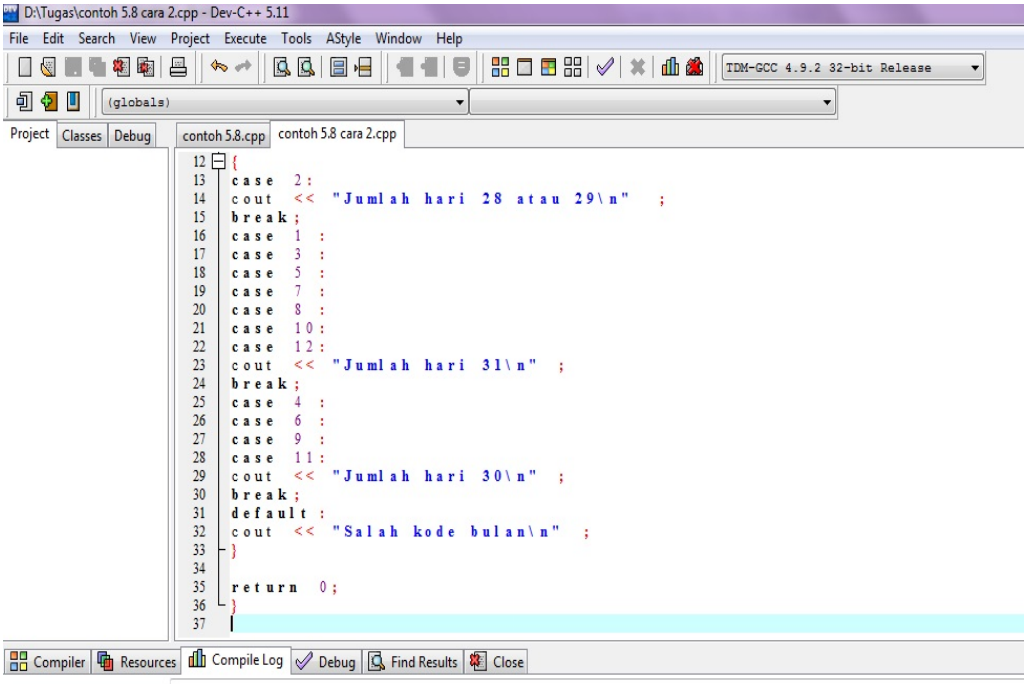
- **Jika Menggunakan DEV C++**

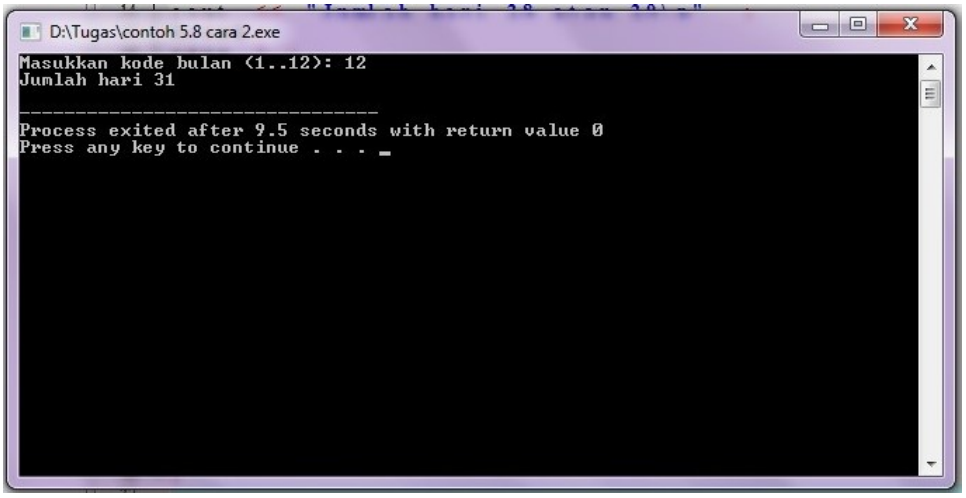
```
#include <iostream>
using namespace std;

int main()
{
     int kode_bulan;

     cout << "Masukkan kode bulan (1..12): ";
     cin >> kode_bulan;

     switch (kode_bulan)
     {
          case 2:
               cout << "Jumlah hari 28 atau 29\n";
               break;
          case 1 :
          case 3 :
          case 5 :
          case 7 :
          case 8 :
          case 10:
          case 12:
               cout << "Jumlah hari 31\n";
               break;
          case 4 :
          case 6 :
          case 9 :
          case 11:
               cout << "Jumlah hari 30\n";
               break;
          default:
               cout << "Salah kode bulan\n";
     }
```

```
    return 0;
}
```


D:\Tugas\contoh 5.8 cara 2.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
(globals)
Project Classes Debug
contoh 5.8.cpp
contoh 5.8 cara 2.cpp
#include <iostream>
using namespace std;
int main()
{
int kode_bulan ;
cout << "Masukkan kode bulan (1..12): " ;
cin >> kode_bulan ;
switch (kode_bulan)
{
case 2:
cout << "Jumlah hari 28 atau 29\n" ;
break;
case 1 :
case 3 :
case 5 :
case 7 :
case 8 :
case 10:
case 12:
cout << "Jumlah hari 31\n" ;
break;
case 4 :
case 6 :
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: D:\Tugas\contoh 5.8 cara 2.exe
- Output Size: 1,30268955230713 MiB
- Compilation Time: 0,73s
Line: 37 Col: 1 Sel: 0 Lines: 37 Length: 467 Insert Done parsing in 0,016 seconds
IN 14:23 23/07/2017


D:\Tugas\contoh 5.8 cara 2.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
(globals)
Project Classes Debug
contoh 5.8.cpp
contoh 5.8 cara 2.cpp
{
case 2:
cout << "Jumlah hari 28 atau 29\n" ;
break;
case 1 :
case 3 :
case 5 :
case 7 :
case 8 :
case 10:
case 12:
cout << "Jumlah hari 31\n" ;
break;
case 4 :
case 6 :
case 9 :
case 11:
cout << "Jumlah hari 30\n" ;
break;
default :
cout << "Salah kode bulan\n" ;
}
return 0;
}
Compiler Resources Compile Log Debug Find Results Close



Akhir Kode Sumber

Contoh 5.9 [Menentukan zodiak] Zodiak (bintang kelahiran] seperti virgo dan Aries sebenarnya dapat ditentukan dengan mudah karena ada aturan yang pasti, sebagaimana tercantum pada Tabel 5.4. Berdasarkan tabel tersebut susunlah algoritma dan program berdasarkan tanggal dan bulan kelahiran.

Tabel 5.4 *Daftar zodiak*

Zodiak	Jangkauan Tanggal
Aries	21 Maret s/d 19 April
Taurus	20 April s/d 20 Mei
Gemini	21 Mei s/d 20 Juni
Cancer	21 Juni s/d 22 Juli
Leo	23 Juli s/d 22 Agustus
Virgo	23 Agustus s/d 22 September
Libra	23 September s/d 22 Okotober
Scorpio	23 Oktober s/d 21 November
Sagitarius	22 November s/d 21 Desember
Capricorn	22 Desember s/d 19 Januari
Aquarius	20 Januari s/d 18 Februari
Pisces	19 Februari s/d 20 Maret

Algoritma:

Untuk menyederhanakan algoritma, diasumsikan bahwa tanggal (1 s/d 31) dan bulan (1 s/d 12) yang dimasukkan pemakai selalu dalam keadaan benar. Dengan keadaan seperti ini, algoritma untuk menentukkan zodiak adalah sebagai berikut:

1. masukkan(tanggal, bulan).
2. JIKA (tanggal ≥ 21 dan bulan = 3) atau

   (tanggal ≤ 19 dan bulan = 4) MAKA

```
        tampilkan(“Zodiak Aries”)
    SEBALIKNYA
        JIKA (tanggal ≥ 20 dan bulan = 4) atau
              (tanggal ≤ 20 dan bulan = 5) MAKA
            tampilkan(“Zodiak Taurus”)
        SEBALIKNYA
            JIKA (tanggal ≥ 21 dan bulan = 5) atau
                  (tanggal ≤ 20 dan bulan = 6) MAKA
                tampilkan(“Zodiak Gemini”)
            SEBALIKNYA
                JIKA (tanggal ≥ 21 dan bulan = 6) atau
                      (tanggal ≤ 22 dan bulan = 7) MAKA
                    tampilkan(“Zodiak Cancer”)
                SEBALIKNYA
                    JIKA (tanggal ≥ 23 dan bulan = 7) atau
                          (tanggal ≤ 22 dan bulan = 8) MAKA
                        tampilkan(“Zodiak Leo”)
                    SEBALIKNYA
                        JIKA (tanggal ≥ 23 dan bulan = 8) atau
                              (tanggal ≤ 22 dan bulan = 9) MAKA
                            tampilkan(“Zodiak Virgo”)
                        SEBALIKNYA
                            JIKA (tanggal ≥ 23 dan bulan = 9) atau
                                  (tanggal ≤ 22 dan bulan = 10) MAKA
                                tampilkan(“Zodiak Libra”)
                            SEBALIKNYA
                                JIKA (tanggal ≥ 23 dan bulan = 10) atau
                                      (tanggal ≤ 21 dan bulan = 11) MAKA
                                    tampilkan(“Zodiak Scorpio”)
                                SEBALIKNYA
                                    JIKA (tanggal ≥ 22 dan bulan = 11) atau
                                          (tanggal ≤ 21 dan bulan = 12) MAKA
```

tampilkan(“Zodiak Sagitarius”)

SEBALIKNYA

JIKA (tanggal ≥ 22 dan bulan = 12) atau

(tanggal ≤ 19 dan bulan = 1) MAKA

tampilkan(“Zodiak Capricorn”)

SEBALIKNYA

JIKA (tanggal ≥ 20 dan bulan = 1) atau

(tanggal ≤ 18 dan bulan = 2) MAKA

tampilkan(“Zodiak Aqurius”)

SEBALIKNYA

tampilkan(“Zodidak Pisces”)

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

Program:

Program C++ berikut merupakan implementasi dari algoritma untuk menentukan zodiak.

Kode Sumber : **zodiak.cpp**

```
#include <iostream.h>

int main()
{
      int tanggal, bulan;

      cout << "Masukkan tanggal kelahiran (1..31): ";
      cin >> tanggal;
```

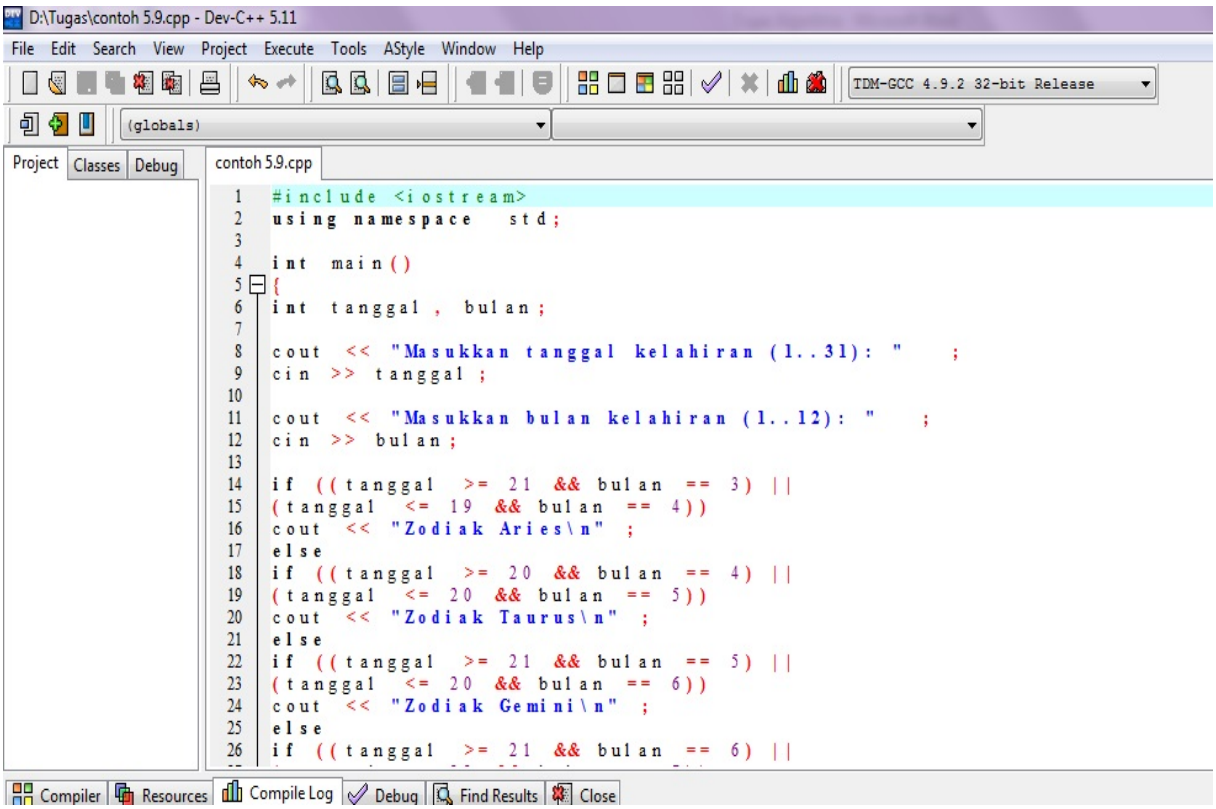
```
cout << "Masukkan bulan kelahiran (1..12): ";
cin >> bulan;

if ((tanggal >= 21 && bulan == 3) ||
(tanggal <= 19 && bulan == 4))
cout << "Zodiak Aries\n";
else
if ((tanggal >= 20 && bulan == 4) ||
     (tanggal <= 20 && bulan == 5))
cout << "Zodiak Taurus\n";
else
     if ((tanggal >= 21 && bulan == 5) ||
     (tanggal <= 20 && bulan == 6))
     cout << "Zodiak Gemini\n";
     else
        if ((tanggal >= 21 && bulan == 6) ||
          (tanggal <= 22 && bulan == 7))
          cout << "Zodiak Cancer\n";
     else
          if ((tanggal >= 23 && bulan == 7) ||
          (tanggal <= 22 && bulan == 8))
          cout << "Zodiak Leo\n";
          else
             if ((tanggal >= 23 && bulan == 8) ||
                 (tanggal <= 22 && bulan == 9))
               cout << "Zodiak Virgo\n";
          else
               if ((tanggal >= 23 && bulan == 9) ||
               (tanggal <= 22 && bulan == 10))
               cout << "Zodiak Libra\n";
               else
                  if ((tanggal >= 23 && bulan == 10) ||
                    (tanggal <= 21 && bulan == 11))
                    cout << "Zodiak Scorpio\n";
               else
                    if ((tanggal >= 22 && bulan == 11)
               ||
                    (tanggal <= 21 && bulan == 12))
                      cout << "Zodiak Sagitarius\n";
                    else
                    if ((tanggal >= 22 && bulan == 12)||
                         (tanggal <= 19 && bulan == 1))
                         cout << "Zodiak Capricorn\n";
                    else
                         if ((tanggal >= 20 && bulan ==
                         1) ||
                              (tanggal <= 18 && bulan
                              == 2))
```

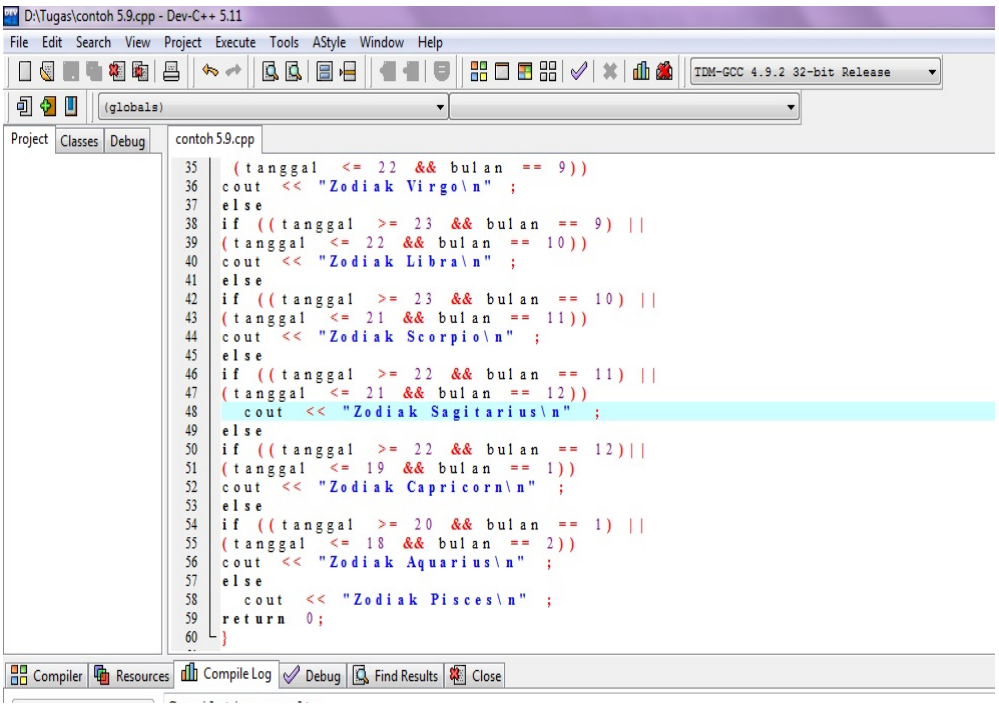
```
                                    cout << "Zodiak Aquarius\n";
                                    else
                                      cout << "Zodiak Pisces\n";
      return 0;
}
```

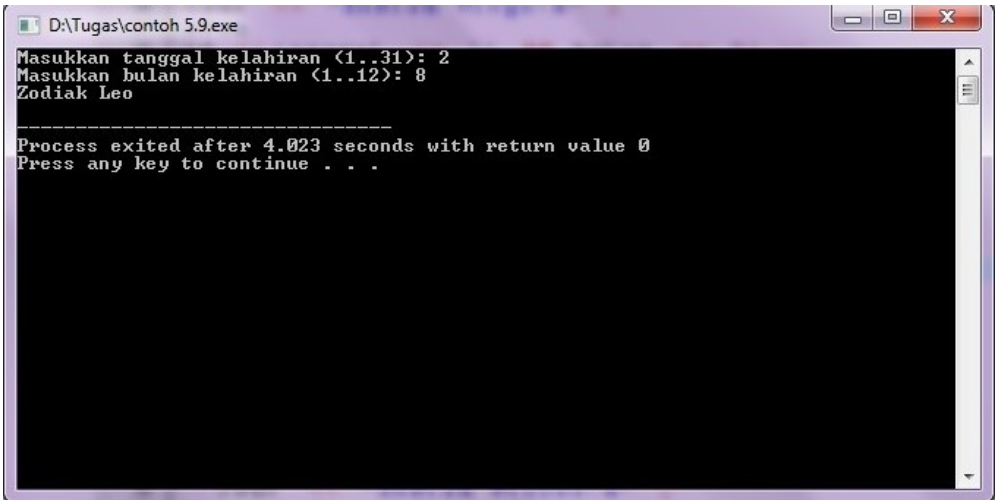
- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
      int tanggal, bulan;

      cout << "Masukkan tanggal kelahiran (1..31): ";
      cin >> tanggal;

      cout << "Masukkan bulan kelahiran (1..12): ";
      cin >> bulan;

      if ((tanggal >= 21 && bulan == 3) ||
      (tanggal <= 19 && bulan == 4))
      cout << "Zodiak Aries\n";
      else
      if ((tanggal >= 20 && bulan == 4) ||
            (tanggal <= 20 && bulan == 5))
      cout << "Zodiak Taurus\n";
      else
            if ((tanggal >= 21 && bulan == 5) ||
            (tanggal <= 20 && bulan == 6))
            cout << "Zodiak Gemini\n";
            else
               if ((tanggal >= 21 && bulan == 6) ||
                  (tanggal <= 22 && bulan == 7))
                  cout << "Zodiak Cancer\n";
            else
                  if ((tanggal >= 23 && bulan == 7) ||
                  (tanggal <= 22 && bulan == 8))
                  cout << "Zodiak Leo\n";
                  else
                     if ((tanggal >= 23 && bulan == 8) ||
                         (tanggal <= 22 && bulan == 9))
                        cout << "Zodiak Virgo\n";
                  else
                        if ((tanggal >= 23 && bulan == 9) ||
                        (tanggal <= 22 && bulan == 10))
                        cout << "Zodiak Libra\n";
                        else
                           if ((tanggal >= 23 && bulan == 10) ||
```

```
                              (tanggal <= 21 && bulan == 11))
                              cout << "Zodiak Scorpio\n";
                        else
                              if ((tanggal >= 22 && bulan == 11)
                        ||
                              (tanggal <= 21 && bulan == 12))
                                cout << "Zodiak Sagitarius\n";
                              else
                              if ((tanggal >= 22 && bulan == 12)||
                                   (tanggal <= 19 && bulan == 1))
                                   cout << "Zodiak Capricorn\n";
                              else
                                   if ((tanggal >= 20 && bulan ==
                                   1) ||
                                        (tanggal <= 18 && bulan
                                        == 2))
                                   cout << "Zodiak Aquarius\n";
                                   else
                                     cout << "Zodiak Pisces\n";
          return 0;
}
```







	Akhir Kode Sumber

5.5 Contoh Lain – Lain

Beberapa contoh permasalahan lain yang melibatkan seleksi dikupas dalam subbab ini.

Contoh 5.10[Menentukan bilangan terbesar di antara tiga bilangan] Diketahui tiga buah bilangan yang dimasukkan dari keyboard. Tulislah algoritma dan program untuk menentukan bilangan terbesar.

Dua cara untuk menyelesaikan masalah ini akan dibahas.

Cara pertama:

Cara pertama dikerjakan dengan menggunakan algoritma seperti berikut:

1. masukkan (*x, y,z).*
2. terbesar ← x // Asumsi bahwa x adalah yang terbesar
3. JIKA terbesar < y MAKA

   terbesar < y

   AKHIR JIKA
4. JIKA terbesar < z MAKA

   terbesar < z

   AKHIR JIKA
5. tampilkan(terbesar)

Dengan cara seperti itu terbesar akan berisi bilangan terbesar di antara x, y, dan z.

Adapun penuangan dalam bentuk program C++ adalah sebagai berikut:

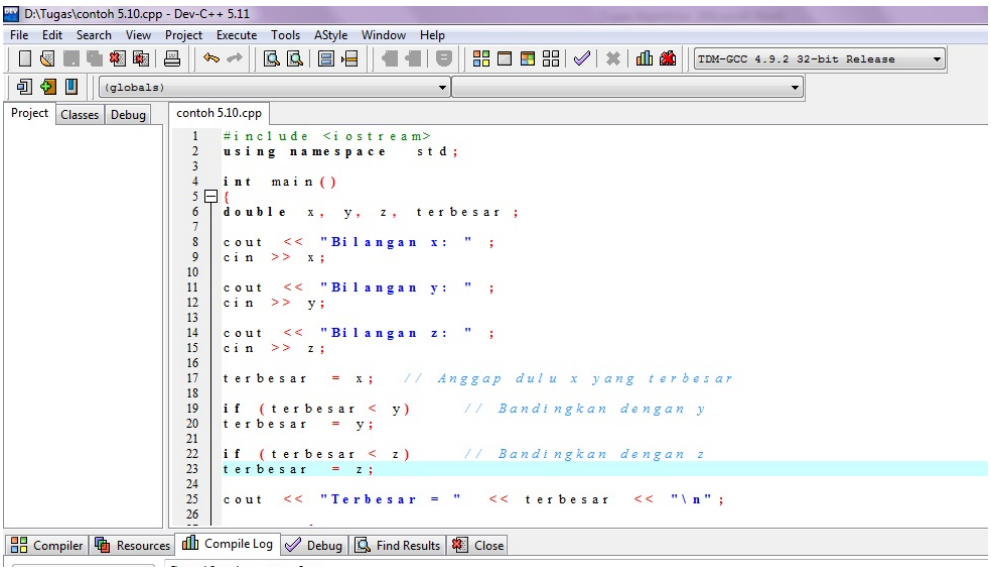
Kode Sumber : **maksi31.cpp**

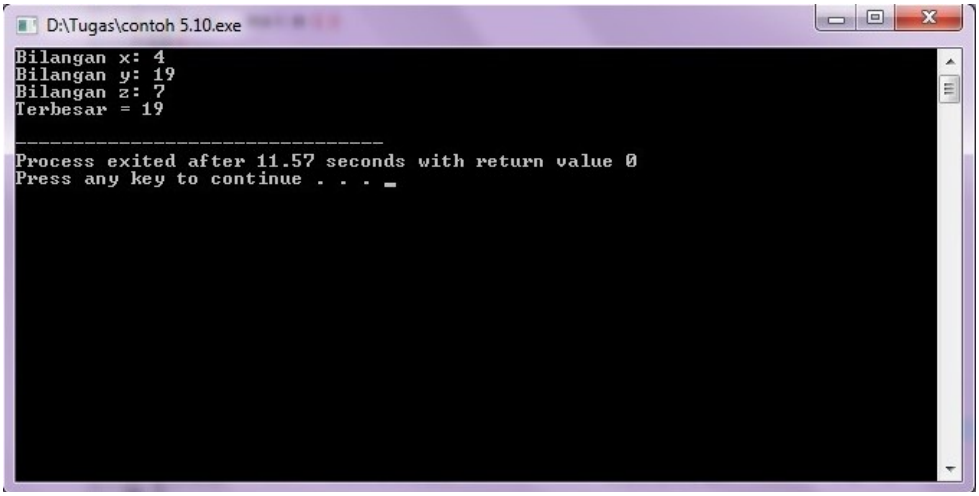
```
#include <iostream.h>

int main()
{
      double x, y, z, terbesar;

      cout << "Bilangan x: ";
      cin >> x;

      cout << "Bilangan y: ";
      cin >> y;

      cout << "Bilangan z: ";
      cin >> z;

      terbesar = x;  // Anggap dulu x yang terbesar

      if (terbesar< y)    // Bandingkan dengan y
      terbesar = y;

      if (terbesar< z)    // Bandingkan dengan z
      terbesar = z;

      cout << "Terbesar = " << terbesar << "\n";
```

```
        return 0;
    }
```

- **Jika Menggunakan DEV C++**

```
    #include <iostream>
    using namespace std;

    int main()
    {
        double x, y, z, terbesar;

        cout << "Bilangan x: ";
        cin >> x;

        cout << "Bilangan y: ";
        cin >> y;

        cout << "Bilangan z: ";
        cin >> z;

        terbesar = x;  // Anggap dulu x yang terbesar

        if (terbesar< y)    // Bandingkan dengan y
        terbesar = y;

        if (terbesar< z)    // Bandingkan dengan z
        terbesar = z;

        cout << "Terbesar = " << terbesar << "\n";

        return 0;
    }
```





Akhir Kode Sumber

Cara Kedua:

Cara kedua dilakukan dengan menggunakan penghubung “dan” seperti berikut:

1. masukkan(*x, y, z).*
2. JIKA x > y dan x > z MAKA
   terbesar ← x
   SEBALIKNYA
   JIKA y > x dan y < z MAKA
   terbesar ← y
   SEBALIKNYA
   terbesar ← z
   AKHIR-JIKA
   AKHIR-JIKA
3. tampilkan(terbesar)

Program berikut merupakan contoh yang menggunakan algoritma di atas.

Kode Sumber : **maksi32.cpp**

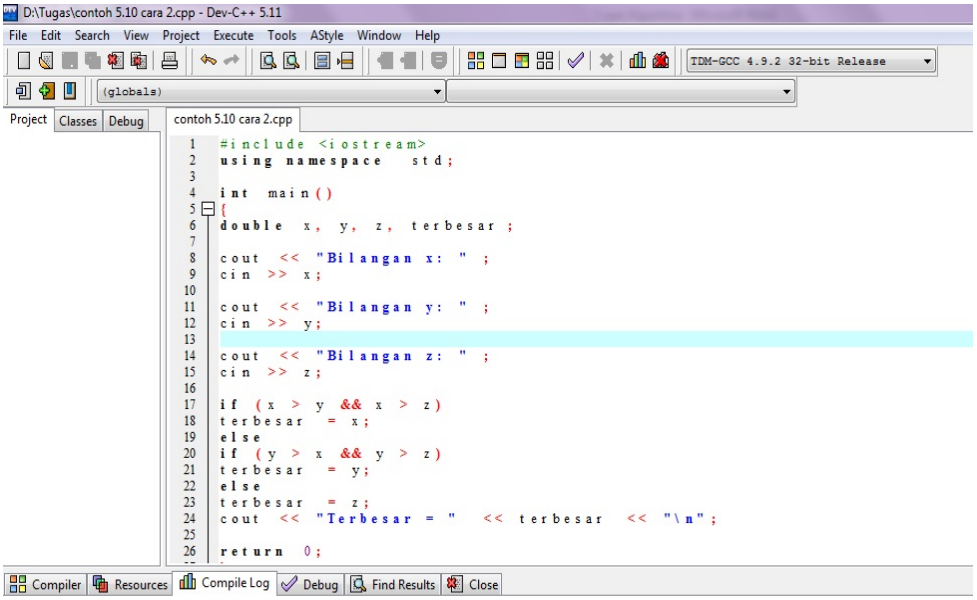
```
#include <iostream.h>
```

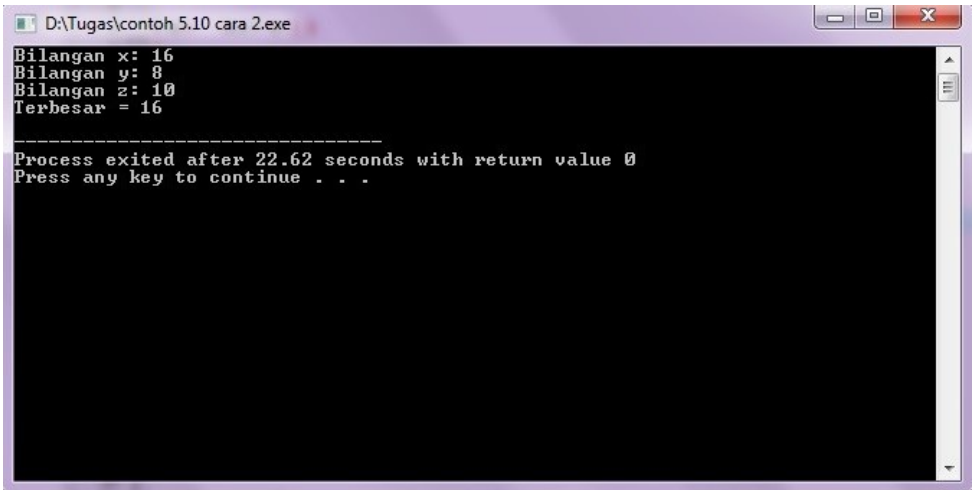
```
int main()
{
	double x, y, z, terbesar;

	cout << "Bilangan x: ";
	cin >> x;

	cout << "Bilangan y: ";
	cin >> y;

	cout << "Bilangan z: ";
	cin >> z;

	if (x > y && x > z)
		terbesar = x;
	else
		if (y > x && y > z)
			terbesar = y;
		else
			terbesar = z;
	cout << "Terbesar = " << terbesar << "\n";

	return 0;
}
```

- **Jika Menggunakan DEV C++**

```
#include <iostream>
using namespace std;

int main()
{
	double x, y, z, terbesar;

	cout << "Bilangan x: ";
	cin >> x;

	cout << "Bilangan y: ";
	cin >> y;

	cout << "Bilangan z: ";
	cin >> z;

	if (x > y && x > z)
		terbesar = x;
	else
		if (y > x && y > z)
			terbesar = y;
		else
			terbesar = z;
```

```
        cout << "Terbesar = " << terbesar << "\n";

        return 0;
}
```





	Akhir Kode Sumber

5.5.1 Menghitung Akar Persamaan Kuadrat

Contoh 5.11 [Menghitung akar persamaan kuadrat] Suatu persamaan kuadrat $ax^2+bx+c=0$ memiliki akar penyelesaian berupa x_1 dan x_2. Ditinjau dari diskriminannya, ada tiga kemungkinan bentuk akar-akarnya. Dalam hal ini diskriminan diperoleh melalui rumus b^2-4ac.

1. Bila diskriminan lebih besar dari pada nol, kedua nilai x yang menjadi solusi persamaan tersebut berupa bilangan real. Sebagai contoh, persamaan x^2-4x+3= 0 memiliki penyelesaian berupa 1 dan 3.
2. Bila diskriminan sama dengan nol, kedua nilai x yang menjadi solusi persamaan tersebut berupa bilangan kembar. Sebagai contoh, persamaan $4x^2$-12x+9=0 memiliki penyelesaian berupa 1,5 dan 1,5.
3. Bila diskriminan kurang dari nol, kedua nilai x yang menjadi solusi persamaan tersebut berupa bilangan kompleks. Sebagai contoh, persamaan $5x^2$+5x+2,05=0 memiliki penyelesaian berupa -0,5+0,4j dan -5-0,4j.

Penentuan x1 dan x2 didasarkan pada rumus berikut:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Buatlah algoritma dan program untuk menyelesaikan masalah di atas.

Algoritma:

Algoritma berikut mengamsusikan bahwa a maupun b tidak berupa nol.

1. masukkan(*a, b, c*).
2. diskriminan ← b x b – 4 x a x c
3. JIKA diskriminan

    x1 ← (-b + akar(diskriminan))/(2 x a)

    x2 ← (-b + akar(diskriminan))/(2 x a)

    tampilkan(“Akar real”, x1, x2)

    SEBALIKNYA

    JIKA diskriminan = 0 MAKA

    x1 ← -b / (2 x a)

    x2 ← x1

    tampilkan(“Akar kembar”, x1, x2)

    SEBALIKNYA

    re ← -b / (2 x a) //Bagian real

    im ← akar(|b x b – 4 x a x c|) / (2 x a) // Bagian imajiner

    tampilkan(“Akar kompleks”)

    tampilkan(“x1=”,re,”+”,im,”i”)

    tampilkan(“x2=”,re,”+”,im,”i”)

AKHIR-JIKA

AKHIR-JIKA

Program:

Implementasi dalam C++:

Kode Sumber : **perskuad.cpp**

```
#include <iostream.h>
#include <math.h>

int main()
{
      double a, b, c;
      double diskriminan, x1, x2, re, im;

      cout << "Persamaan kuadrat\n";
      cout << "a: ";
      cin >> a;

      cout << "b: ";
      cin >> b;

      cout << "c: ";
      cin >> c;

      // Hitung dskriminan
      diskriminan = b * b - 4 * a * c;

      // Evaluasi diskriminan
      if (diskriminan > 0)
      {
            x1 = (-b + sqrt(diskriminan)) / (2 * a);
            x2 = (-b - sqrt(diskriminan)) / (2 * a);
            cout << "Akar real:\n";
            cout << "x1 = " << x1 << "\n";
            cout << "x2 = " << x2 << "\n";
      }
      Else
            if (diskriminan == 0)
            {
                  x1 = -b / (2 * a);
                  x2 = x1;
                  cout << "Akar kembar:\n";
                  cout << "x1 = " << x1 << "\n";
                  cout << "x2 = " << x2 << "\n";
```

```
            }
            else /* diskriminan < 0 */
            {
                  re = -b / (2 * a);
                  im = sqrt(fabs(diskriminan))/(2 * a);
                  cout << "Akar kompleks\n";
                  cout << "x1 = " << re << " + " << im < "\n";
                  cout << "x2 = " << re << " - " << im < "\n";
            }
      return 0;
}
```

Akhir Kode Sumber

BAB 6
OPERASI PENGULANGAN

6.1 Memahami Bentuk ULANG…AKHIR-ULANG dan Translasi ke Program C++

Salah satu struktur pengulangan yang telah anda pelajari sekilas adalah berupa ULANG…AKHIR-ULANG. Bentuknya seperti berikut:

```
ULANG SELAMA kondisi
        pernyataan-1
        …
        pernyataan-2
AKHIR-ULANG
```

Dalama hal ini, bagaian pernyataan-1 hingga pernyataan-2 akan dijalankan secara terus-menerus selama kondisi bernilai BENAR.

Bentuk seperti itu di translasikan kedalam bentuk C dan C++ dengan menggunakan pernyataan bernama **while**. Bentuknya seperti berikut:

```
while (kondisi)
{
        pernyataan-1;
        …
        pernyataan-2;
}
```

Seandainya dalam tanda { dan } hanya terdapat satu pernyataan, pasangan tanda tersebut bisa dihilangkan. Contoh:

```
while (kondisi)
        Pernyataan;
```

Catatan 	Kondisi pada C dan C++ dapat berupa ekspresi yang menghasilkan nilai benar atau salah dan harus ditulis dalam tanda kurung.

CONTOH 6.1 pada bab 3 telah dibahas sebuah algoritma untuk memperoleh faktor persekutuan terbesar (FPB). Algoritmanya seperti berikut:

masukkan (m, n)

1. r ← sisa_pembagian(m, n)

2. ULANG SELAMA r ≠ 0

   m ← n

   n ← r

   r ← sisa_pembagian(m, n)

   AKHIR-ULANG

3. tampilkan (n)

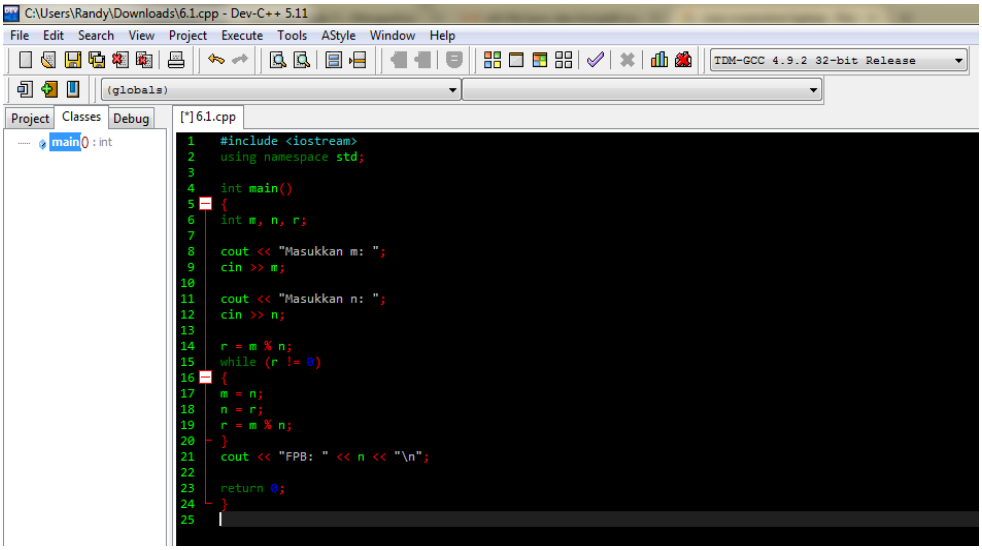
Cobalah untuk mentranslasikan ke dalam program C atau C++.

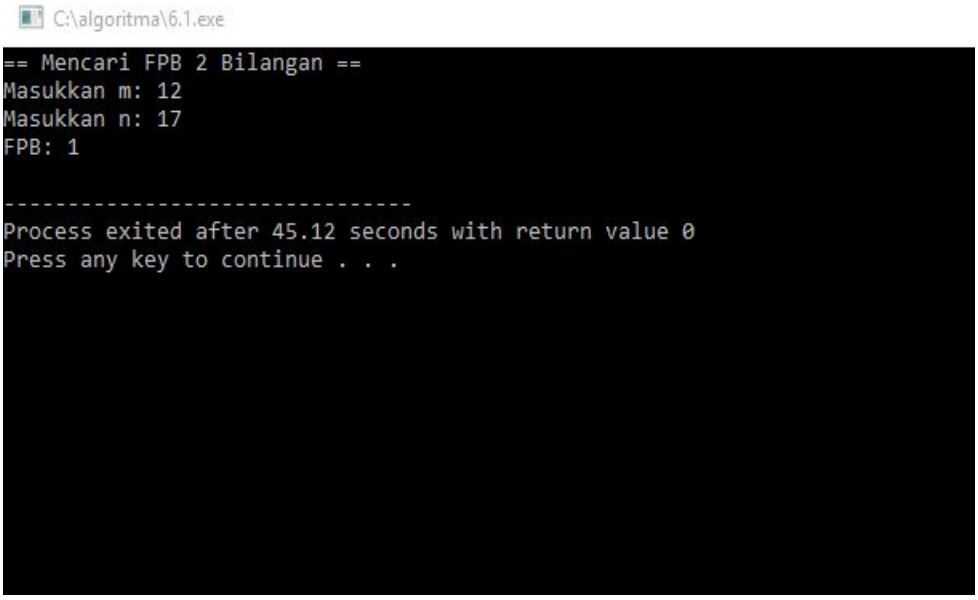
Program:

Program C++ yang di dasarkan pada algoritma di depan dapat dilihat di bawah ini.

Kode Sumber : **fpb.cpp**

```
#include <iostream.h>

int main()
{
      int m, n, r;

      cout << "Masukkan m: ";
      cin >> m;

      cout << "Masukkan n: ";
      cin >> n;

      r = m % n;
      while (r != 0)
      {
            m = n;
            n = r;
            r = m % n;
      }
      cout << "FPB: " << n << "\n";

      return 0;
}
```





Akhir Kode Sumber

Pada program di atas,

r = m % n;

di gunakan untuk memperoleh sisa pembagi *m* dengan *n* dan hasilnya di letakan di variable *r*.

CONTOH 6.2 Buatlah algoritma dan program untuk menampilkan 6 buah baris yang bertuliskan “SELAMAT BELAJAR”.

Algoritma:

1. bil ← 1
2. ULANG SELAMA bil ≤ 6

   tampilkan(“SELAMAT BELAJAR”)

   bil ← bil + 1

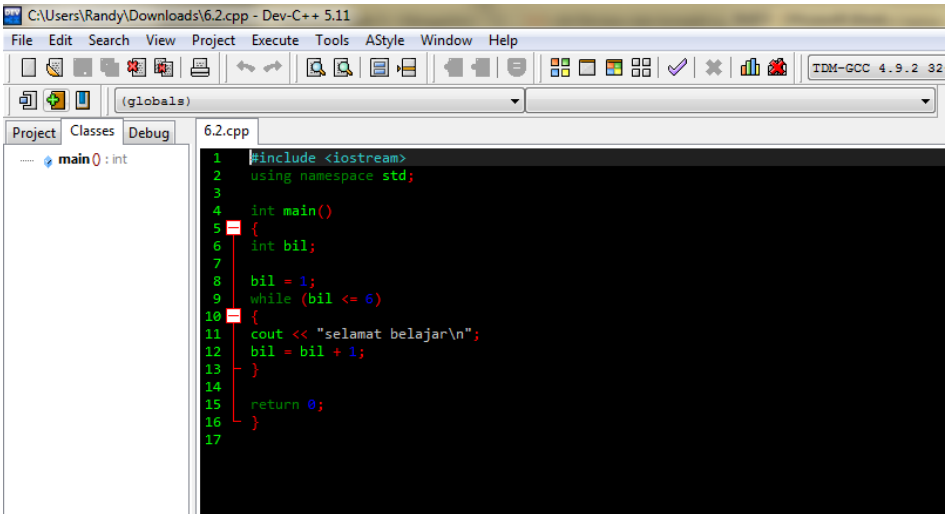
   AKHIR-ULANG

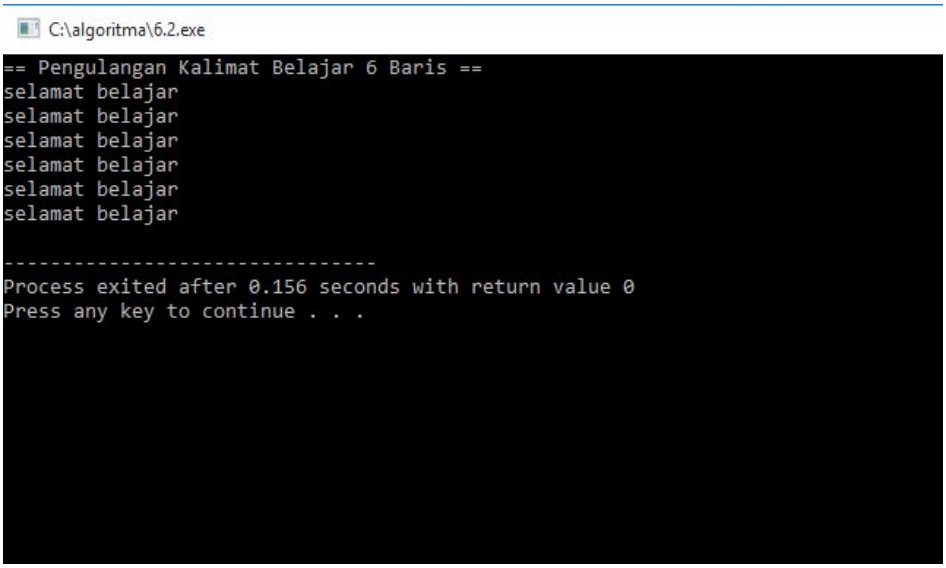
Program:

Program C++ berukut merupakan implementasi program berdasarkan pada algoritma di depan.

Kode Sumber : **enambrs.cpp**

```
#include <iostream.h>

int main()
{
      int bil;

      bil = 1;
      while (bil <= 6)
      {
            cout << "Bahasa C++\n";
            bil = bil + 1;
      }

      return 0;
}
```





Akhir Kode Sumber

CONTOH 6.3 Buatlah algoritma dan pemograman untuk menampilkan bilangan ganjil yang terletak antara 1 sampai dengan 10.

Algoritma:

1. bil ← 1
2. ULANG SELAMA bil ≤ 10

   tampilkan(bil)

   bil ← bil + 2

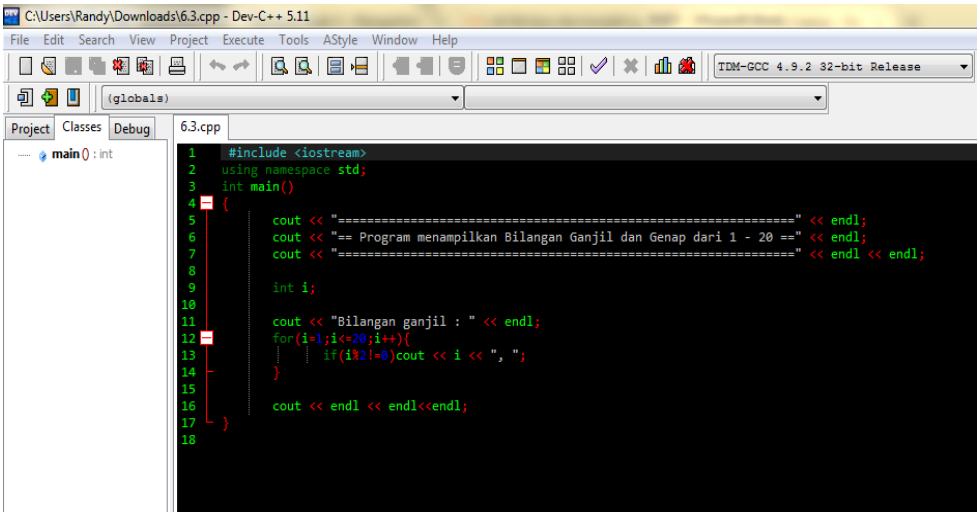
   AKHIR-ULANG

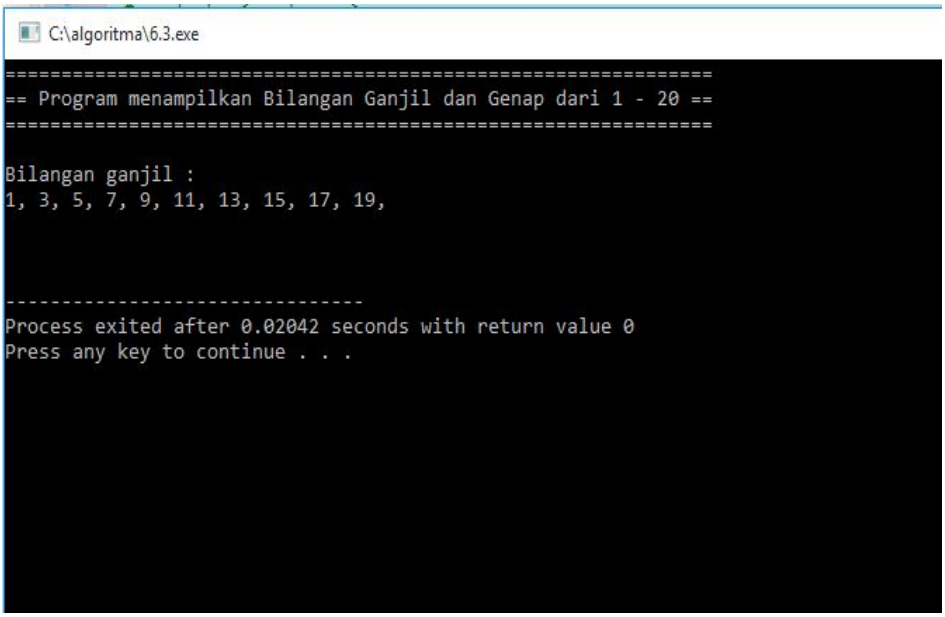
Program:

Program C++ berikut merupakan translasi berdasarkan algoritma tersebut.

Kode Sumber : **bilganj.cpp**

```
#include <iostream.h>

int main()
{
	int bil;

	bil = 1;
	while (bil <= 10)
	{
		cout << "%d ", bil;
		bil = bil + 2;
	}

	cout << "\n"; // Pindah baris
	return 0;
}
```





	Akhir Kode Sumber

CONTOH 6.4 Buatlah program dan algortima yang menampilkan deretan seperti berikut:

10

9

8

7

6

5

4

3

2

1

Algoritma:

1. Bil ← 10
2. ULANG SELAMA bil ≥ 1

   tampilkan(bil)

   bil ← bil – 1

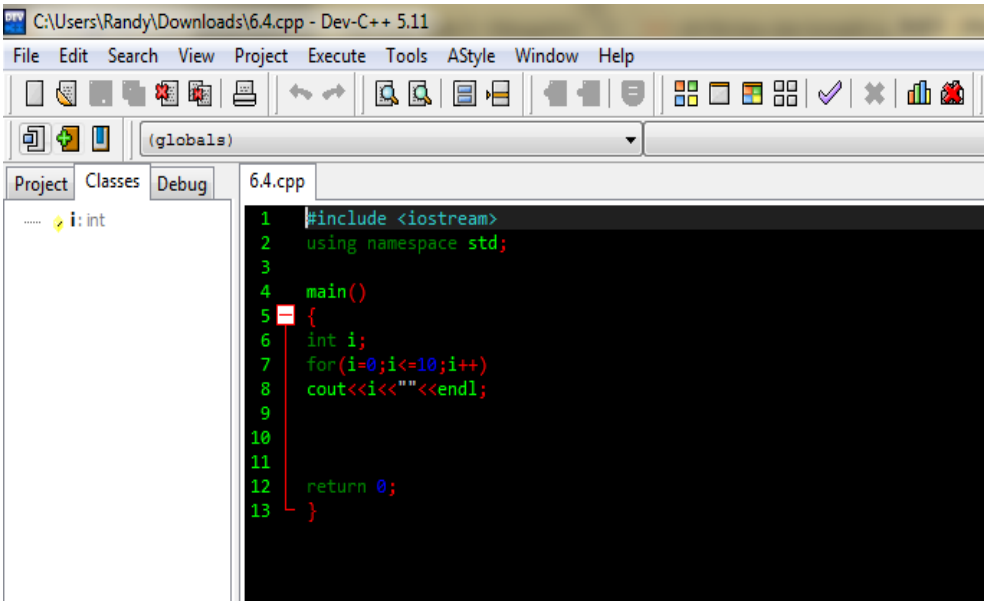
AKHIR-ULANG

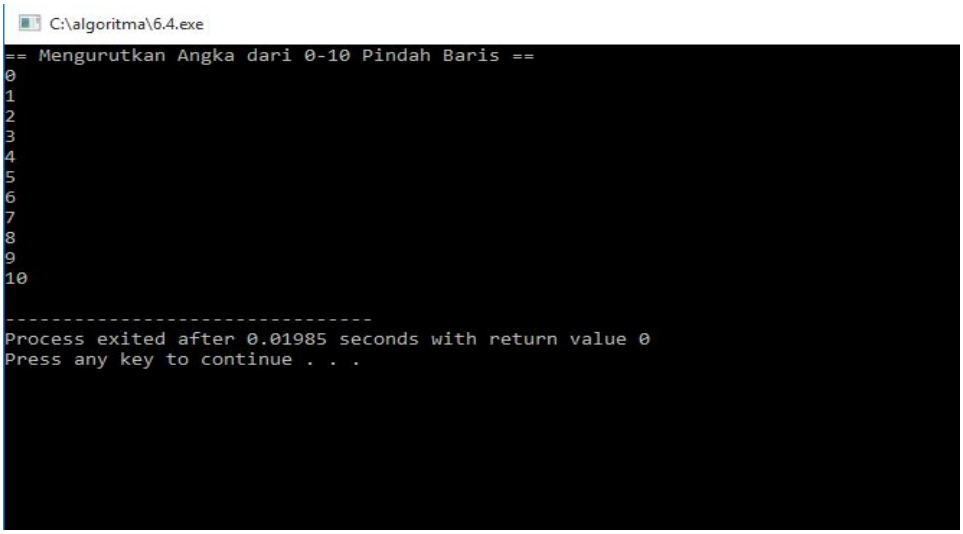
Program:

Penuangan algoritma di depan ke dalam program C++ adalah seperti berikut:

Kode Sumber : **derettrn.cpp**

```
#include <iostream.h>

int main()
{
	int bil;

	bil = 10;
	while (bil >= 1)
	{
		cout << "%d\n", bil;
		bil = bil - 1;
	}

	return 0;
}
```





Akhir Kode Sumber

CONTOH 6.5 Buatlah algoritma dan pemograman untuk menghitung jumlah seperti berikut dengan n adalah bilangan yang dimasukan dari keyboard:

1 + 2 + 3 + 4 + 5 + … + (n + 2) + (n – 1) + n

Algoritma:

1. masukkan(n)
2. jum ← 0
3. pencacah ← 0
4. ULANG SELAMA pencacah ≤ n

   jum ← jum + pencacah //Jumlahkan

   pencacah ← pencacah + 1

   AKHIR-ULANG
5. tampilkan(jum)

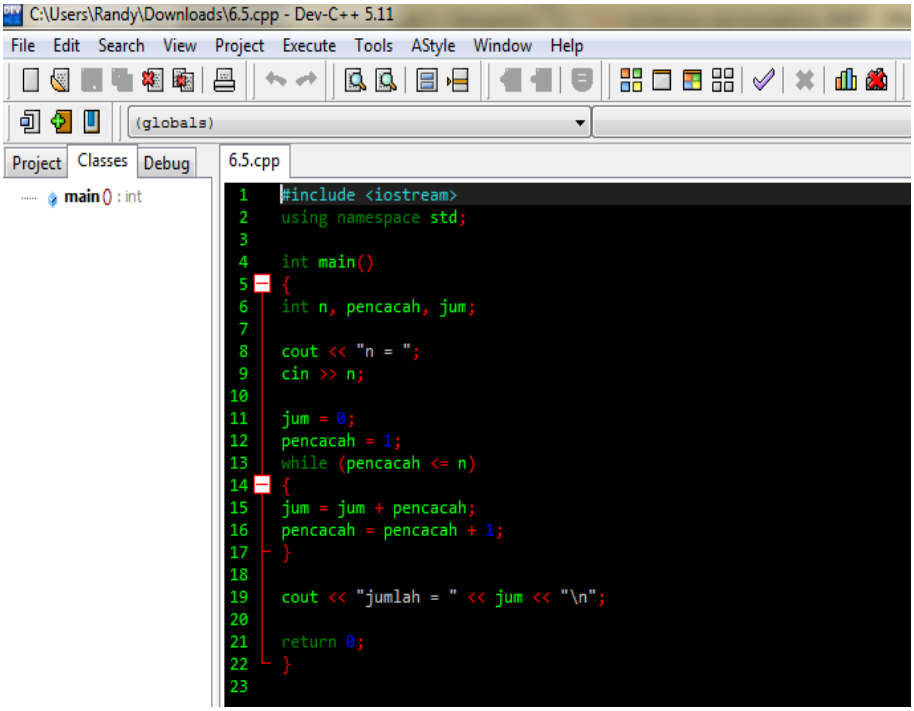
Program:

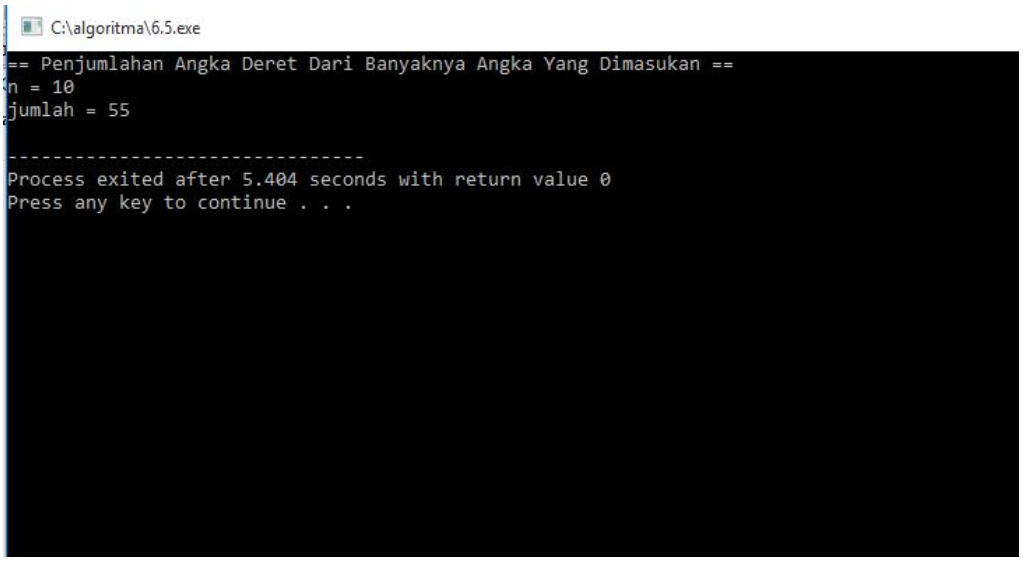
Program C++ hasil translasi algoritma di depan dapat diliha di bawah ini.

Kode Sumber : **jumlah.cpp**

```
#include <iostream.h>

int main()
{
      int n, pencacah, jum;

      cout << "n = ";
```

```
	cin >> n;

	jum = 0;
	pencacah = 1;
	while (pencacah <= n)
	{
		jum = jum + pencacah;
		pencacah = pencacah + 1;
	}

	cout << "Jumlah = " << jum << "\n";

	return 0;
}
```





Akhir Kode Sumber

CONTOH 6.6 Konstanta π dapat di peroleh melalui pendekatan seperti berikut:

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots + \frac{1}{n^2}$$

Untuk n yang besar (di atas 100)

Buatlah algortima dan program yang mula-mula memninta data n di masukkan dari keyboard kemudian hitunglahπ.

Algortima:

1. masukkan(n)
2. bil ← 1
3. jum ← 0
4. ULANG SELAMA bil ≤ n

   jum ← jum + 1 / (bil x bil)

   pencacah ← pencacah + 1
5. pi ← akar(jum x 6)
6. tampilkan(pi)

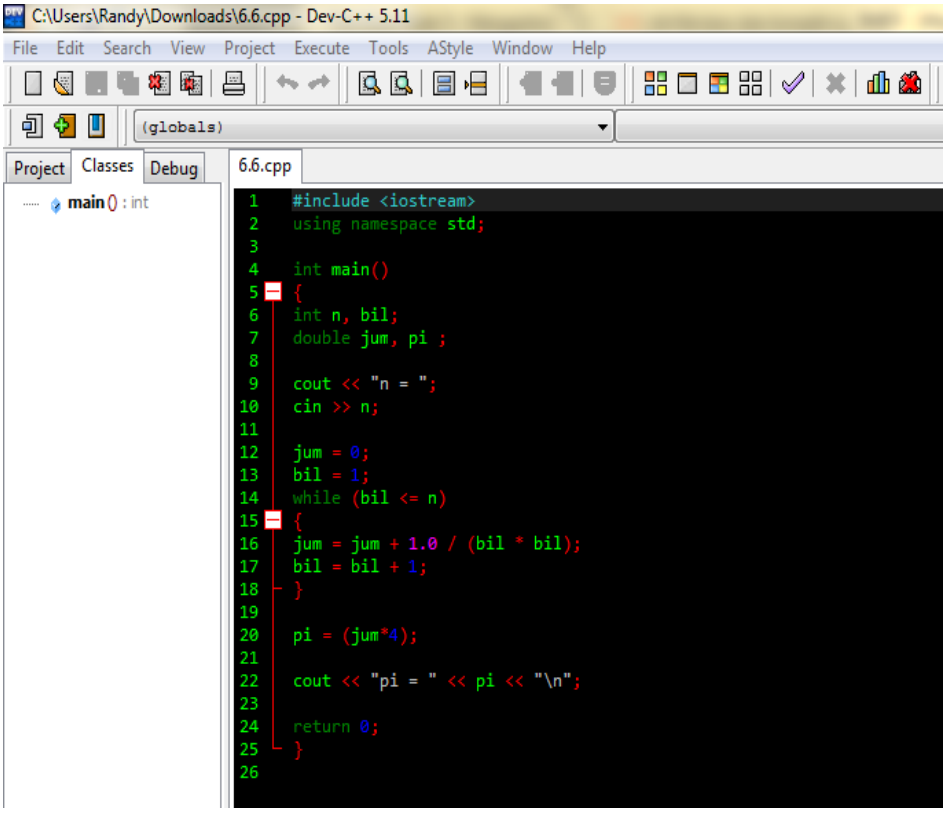
Program:

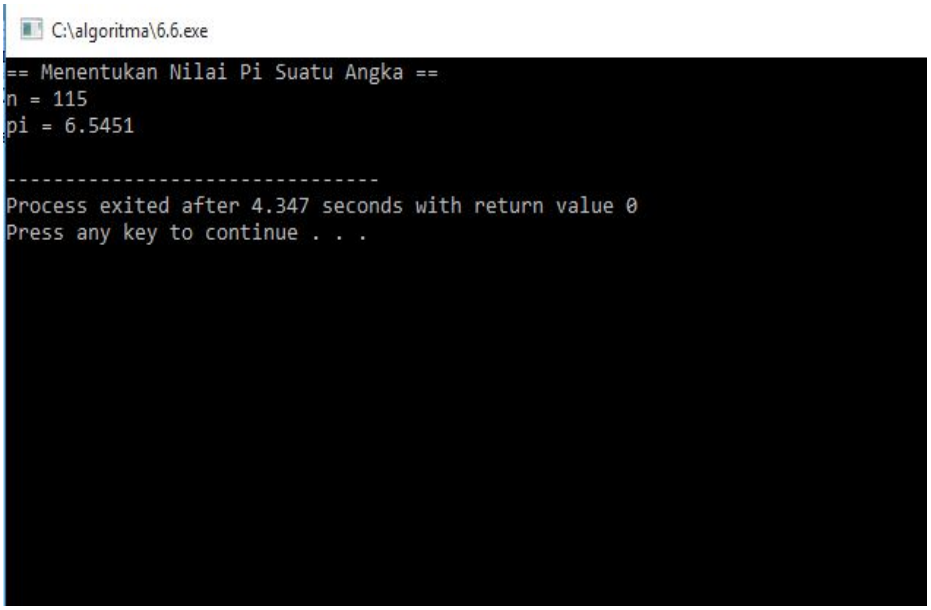
Translasi algortima di depan dapat dilihat pada program C++ berikut:

Kode Sumber : **pi.cpp**

```
#include <iostream.h>
#include <math.h>

int main()
{
      int n, bil;
      double jum, pi;

      cout << "n = ";
      cin >> n;

      jum = 0;
      bil = 1;
      while (bil <= n)
      {
```

```
            jum = jum + 1.0 / (bil * bil);
            bil = bil + 1;
      }

      pi = sqrt(jum * 6);

      cout << "pi = " << pi << "\n";

      return 0;
}
```





Akhir Kode Sumber

Pada program di atas,

```
1.0 / (bil * bil)
```

Dimaksudkan agar hasil pembagian 1 dengan bilangan kuadrat bernilai pecahan. Bila 1.0 ditulis dengan 1, hasil pembagian akan selalu berupa nol.

CONTOH 6.7 Buatlah algoritma dan program yang meminta n buah bilangan bulat dimasukkan dari keyboard dan kemudian menampilkannilai rata-rata dari keseluruhan bilangan tersebut.

Algoritma:

Algortima dimulai dengan mula-mula meminta jumlah data yang akan dimasukkan dari keyboard (n). selanjutnya, n buah data di baca dari keyboard dan setiap kali setiap data selesai dibaca, nilainya ditambahkan ke variabel jum. Dengan demikian, setelah data terakhir dimasukkan, jum berisi penjumlahan n buah nilai yang dimasukkan dari keyboard. Algoritma selengkapnya adalah sebagai berikut:

1. masukkan(n)
2. jum ← 0
3. pencacah ← 0
4. ULANG SELAMA pencacah ≤ n
   masukkan(bil)
   jum ← jum +bil //Jumlahkan
   pencacah ← pencacah + 1
   AKHIR-ULANG
5. rata_rata ← jum / n
6. tampilkan(rata_rata)

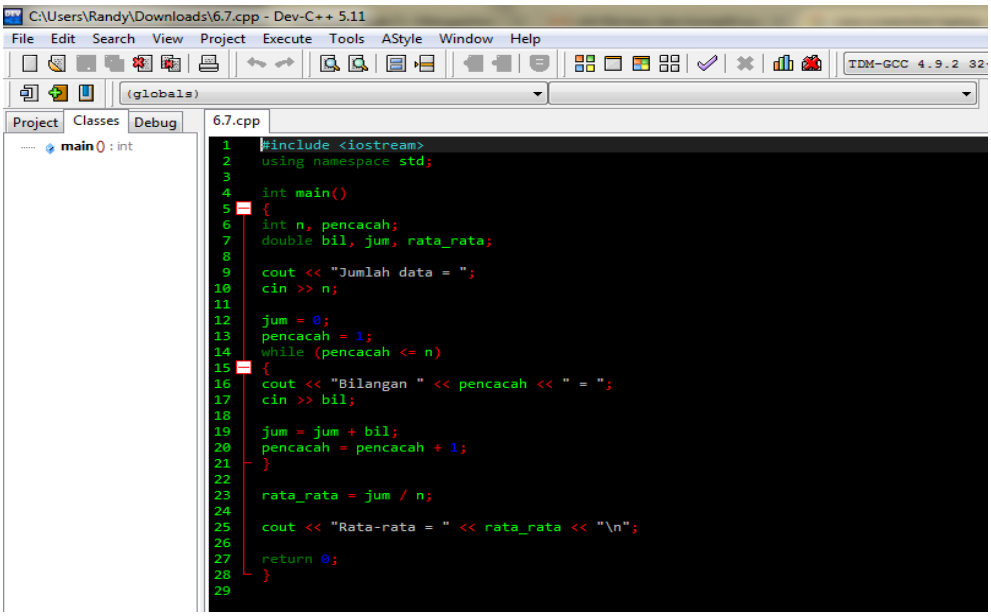
Program:

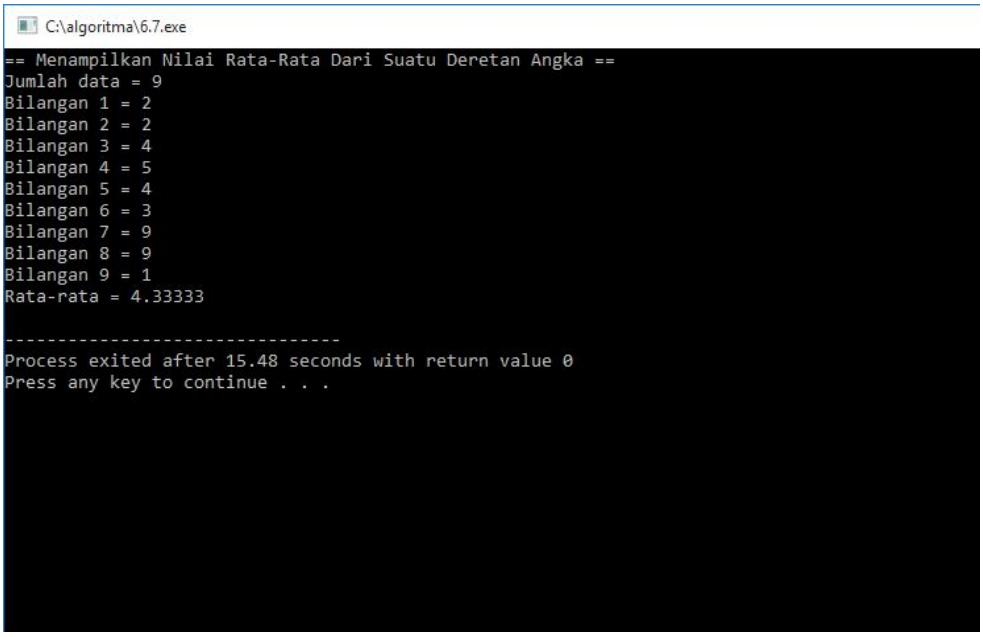
Program C++ yang di dasarkan pada algoritma di depan dapat dilihat di bawah ini.

Kode Sumber : **rata2.cpp**

```
#include <iostream.h>
```

```
int main()
{
	int n, pencacah;
	double bil, jum, rata_rata;

	cout << "Jumlah data = ";
	cin >> n;

	jum = 0;
	pencacah = 1;
	while (pencacah <= n)
	{
		cout << "Bilangan " << pencacah << " = ";
		cin >> bil;

		jum = jum + bil;
		pencacah = pencacah + 1;
	}

	rata_rata = jum / n;

	cout << "Rata-rata = " << rata_rata << "\n";

	return 0;
}
```





Akhir Kode Sumber

CONTOH 6.8 Buatlah algortima dan pemograman yang meminta sebuah bilangan bulat dimasukkan dari keyboard dan kemudian menampilkan deret seperti berikut:

1 2 4 7 11 16 22 29 37 46

Bila bilangan adalah 50. Bantuan: angka terakhir yang dimasukkan dalam derat harus tidak lebih dari bilangan yang dimasukkan dari keyboard.

Algoritma:

1. masukkan(n)
2. pencacah ← 1
3. nilai ← 1
4. ULANG SELAMA nilai < n

        tampilkan(nilai)

        nilai ← nilai + pencacah

        pencacah ← pencacah + 1

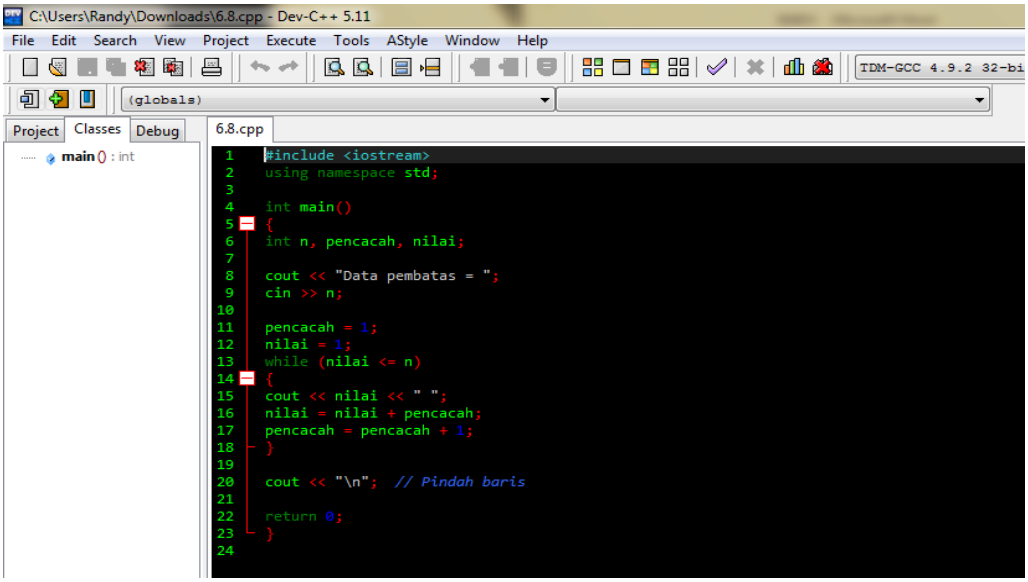
    AKHIR-ULANG

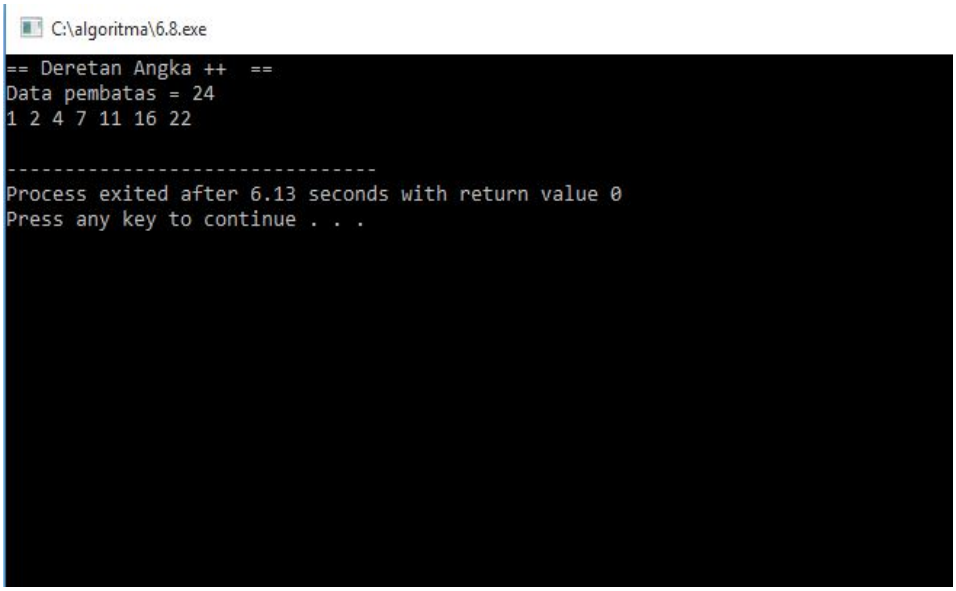
Program:

Implementasi dalam program C++

Kode Sumber : **naiktrs.cpp**

```
#include <iostream.h>

int main()
{
	int n, pencacah, nilai;

	cout << "Data pembatas = ";
	cin >> n;

	pencacah = 1;
	nilai = 1;
	while (nilai <= n)
	{
		cout << nilai << " ";
		nilai = nilai + pencacah;
		pencacah = pencacah + 1;
	}

	cout << "\n";  // Pindah baris

	return 0;
}
```





Akhir Kode Sumber

CONTOH 6.9 Buatlah algortima dan pemograman untuk menampilkan deret berikut dengan menggunakkan struktur pengulangan:

100000000

10000000

1000000

100000

10000

1000

100

10

1

Algortima:

1. Nilai ← 100000000
2. ULANG SELAMA nilai ≥ 1

   tampilkan(nilai)

   nilai ← nilai / 10

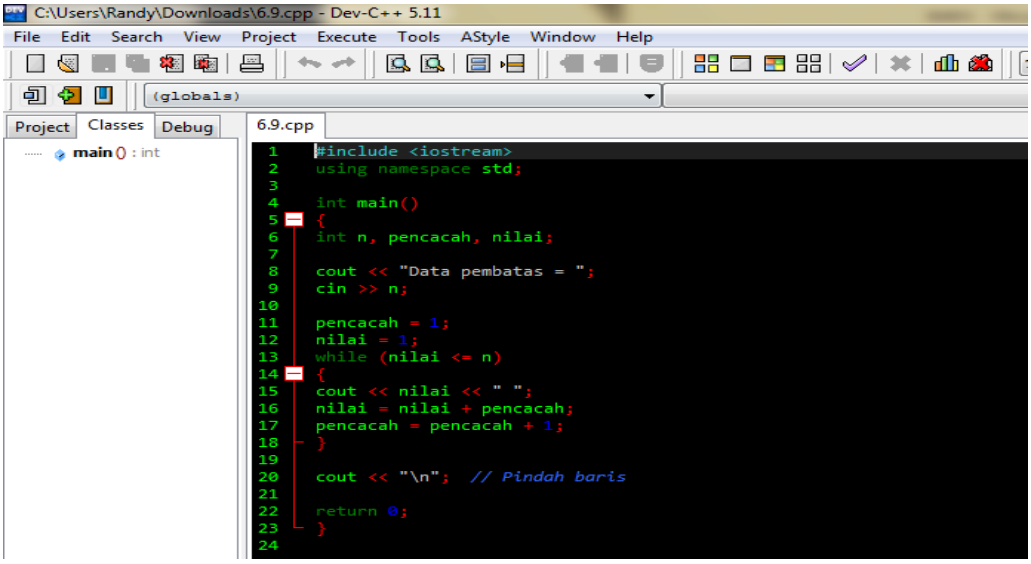
   AKHIR-ULANG

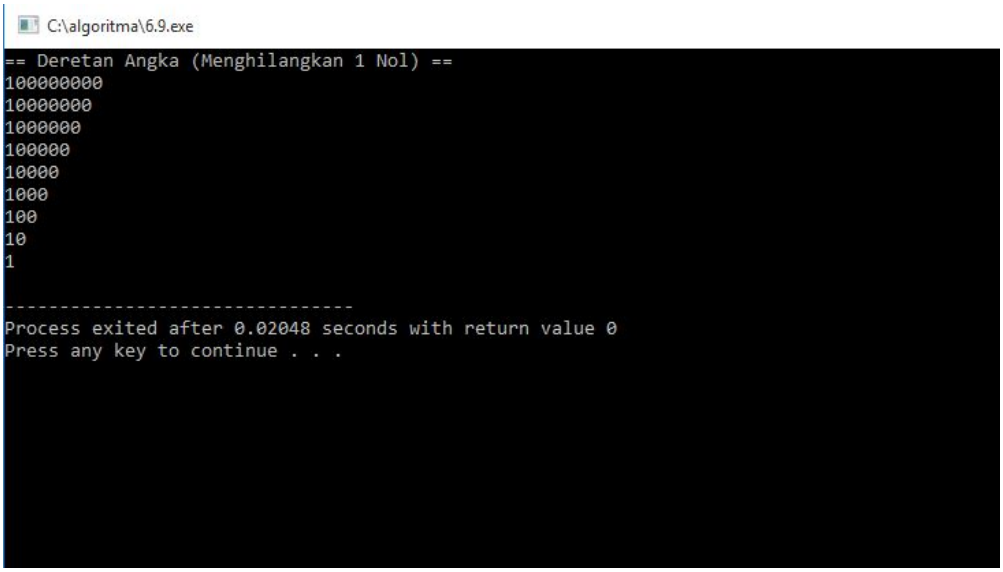
Program:

Program C++ yang didasarkan algortima di depan adalah sebagai berikut:

Kode Sumber : **bagi10.cpp**

```
#include <iostream.h>

int main()
{
	long int nilai;

	nilai = 100000000;
	while (nilai >= 1)
	{
		cout <<  nilai << "\n";
		nilai = nilai / 10;
	}

	return 0;
}
```





Akhir Kode Sumber

CONTOH 6.10 Sebagiamana telah disebutkan, string dalam C selalu di akhiri dengan karakter NULL. Berdasarkan fakta tersebut buatlah algoritma dan program yang pertama-tama meminta suatu string yang terdapat pada string tersebut.

Algortima:

1. masukkan(string)
2. indeks ← 0
3. ULANG SELAMA string[indeks] ≠ NULL

   Indeks ← indeks + 1

   AKHIR ULANG
4. tampilkan(“Jumlah karakter =”, indeks)

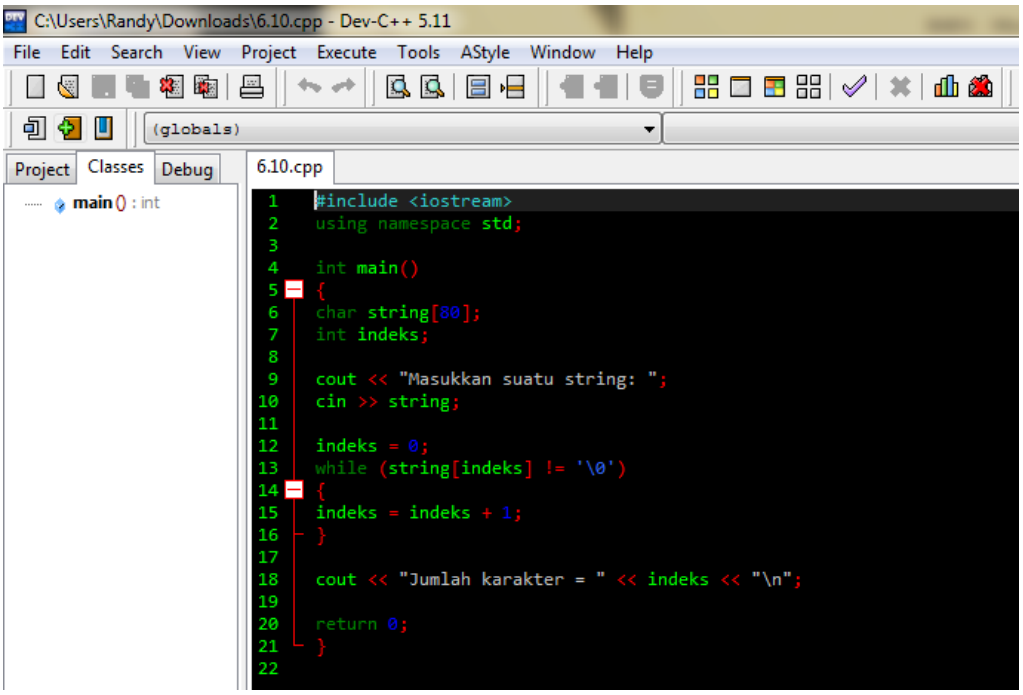
Program:

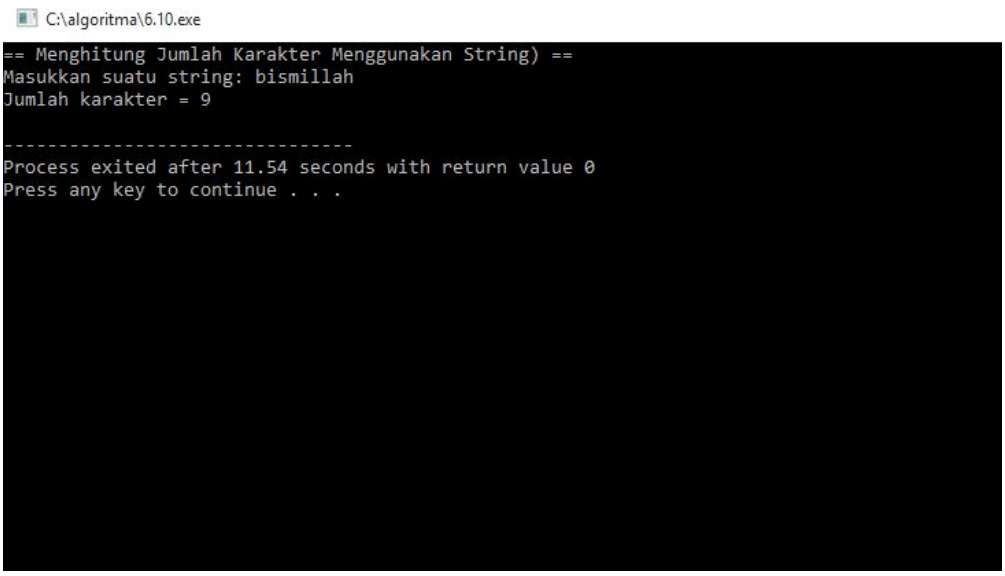
Translasi algortima dalam program C++ adalah sebagai berikut:

Kode Sumber : **jumkar.cpp**

```
#include <iostream.h>

int main()
{
      char string[80];
      int indeks;

      cout << "Masukkan suatu string: ";
```

```
	cin >> string;

	indeks = 0;
	while (string[indeks] != '\0')
	{
		indeks = indeks + 1;
	}

	cout << "Jumlah karakter = " << indeks << "\n";

	return 0;
}
```





Akhir Kode Sumber

CONTOH 6.11 Buatlah algoritma dan program untuk menampilkan suatu string yg di masukkan dari keyboard menjadi terbalik. Contoh jika string yg dimasukkan berupa

Yogya

Maka hasilnya

aygoY

Algoritma:

1. masukkan(string)
2. indeks ← panjang(string) – 1
3. ULANG SELAMA indeks >= 0

   tampilkan(string[indeks])

   indeks = indeks – 1

   AKHIR ULANG

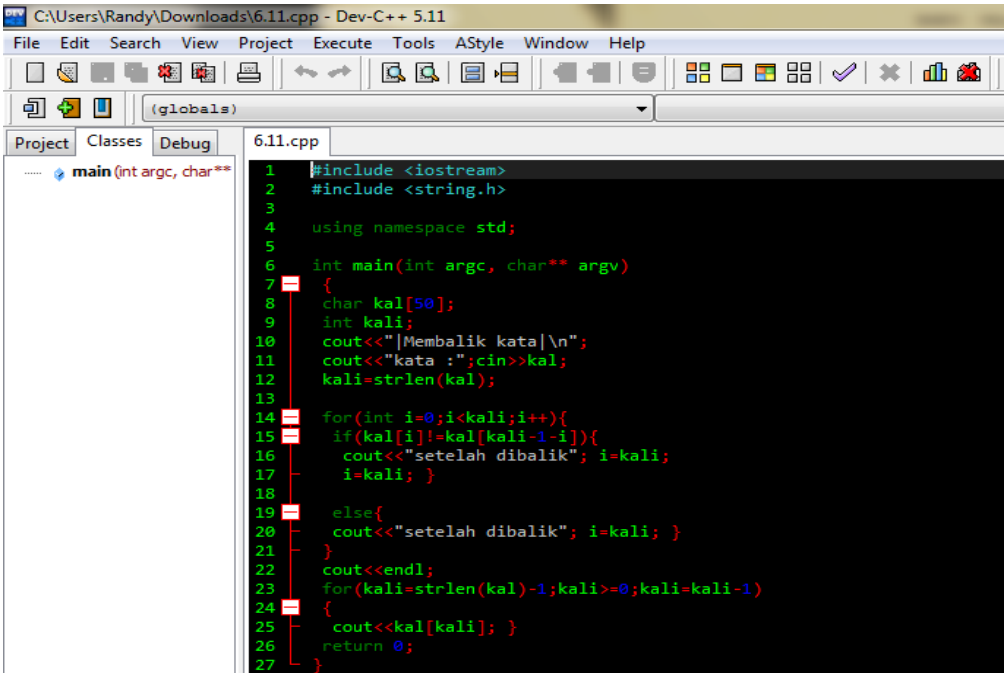
Program:

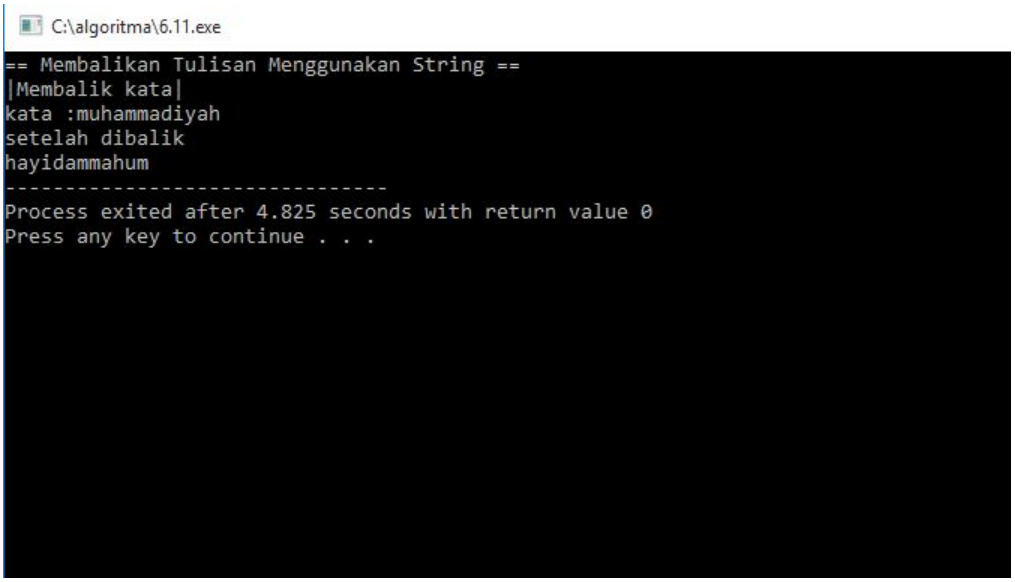
Penuangan algoritma ke dalam program C ++:

Kode Sumber : **balikstr.cpp**

```
#include <iostream.h>
#include <string.h>  // prototipe strlen())
```

```
int main()
{
	char string[80];
	int indeks;

	cout << "Masukkan suatu string: ";
	cin >> string;

	indeks =  strlen(string) - 1;
	while (indeks >= 0)
	{
		cout << string[indeks];
		indeks = indeks - 1;
	}

	cout << "\n"; // Pindah baris

	return 0;
}
```





	Akhir Kode Sumber

6.2 Memahami Bentuk UNTUK..AKHIR-UNTUK dan Translasi ke program C dan C++

Sebagaimana telah dibahas di Nan 3, bentuk UNTUK..AKHIR-UNTUK juga digunakan untuk menangani pengulangan. Bentuk pemakaian adalah :

UNTUK variabel ← awal S/D akhir LANGKAH *langkah*

Pernyataan – 1

...

Pernyataan-2

AKHIR-UNTUK

Bentuk di atas melakukan pengulangan terhadap *Pernyataan-1* hingga *Pernyataan-2* yang dimulai dari *variabel* bernilai *awal* hingga variabel bernilai tidak lebih dari *akhir*. Klausa LANGKAH menentukan kenaikan terhadap nilai variabel untuk setiap iterasi berikutnya. Bagian ini bersifat opsional. Kalau tidak disebutkan, kenaikan terhadap *variabel* adalah sebesar 1.

Pada C dan C ++ bentuk tersebut dapat di translasikan dengan menggunakan **for.** Pernyataan **for** yang setara dengan bentuk UNTUK..AKHIR-UNTUK adalah :

```
for (variabel=awal; variabel <= akhir; variabel = variabel + langkah)
{
```

Pernyataan – 1

...

Pernyataan-2

}

Tanda { } dapat ditiadakan jika pada bagian tersebut hanya terdapat sebuah pernyataan.

Catatan 	• Pada C dan C++, bentuk seperti
	`variabel = variabel + langkah`
	dapat ditulis menjadi
	`variabel += langkah`
	• Beberapa bentuk yang setara dengan += antara lain:
	-=, *=, %=, *=
	• Bentuk seperti:
	`variabel = variabel + 1`
	dapat ditulis menjadi
	`variabel ++`
	• Bentuk seperti:
	`variabel = variabel - 1`
	dapat ditulis
	`variabel --`

Beberapa contoh translasi dari bentuk UNTUK..AKHIR-UNTUK ke pernyataan **for** dapat dilihat pada tabel berikut:

UNTUK..AKHIR-UNTUK	Pernyataan for	Hasil

Untuk bil = 1 S/D 8
Tampilkan(bil)
AKHIR-UNTUK
	`For(bil = 1; bil <= 8; bil++)`
	`printf("%d\n", bil);`
	
	Catatan:
	bil++ identik dengan bil = bil - 1
		1
		2
		3
		4
		5
		6
		7
		8

Untuk bil = 1 S/D 8 STEP 3
tampilkan(bil)
AKHIR-UNTUK
	`For(bil = 1; bil <= 8; bil+= 3)`
	`printf("%d\n", bil);`
	
	Catatan:
	Bil += identik dengan bil = bil + 3
		1
		4
		7

Untuk bil = 10 S/D 5 STEP -1	`For(bil = 8; bil >= 5; bil--)` `printf("%d\n", bil);`	10 9

tampilkan(bil) AKHIR-UNTUK	**Catatan:** Bil -- identik dengan bil = bil - 1	8 7 6 5

Untuk bil = 10 S/D 5 STEP -2
tampilkan(bil)
AKHIR-UNTUK
	`For(bil = 8; bil >= 5; bil-= 2)`
	`printf("%d\n", bil);`
	Catatan:
	Bil -= 2 identik dengan bil = bil - 2
		8
		6

Penggunaan bentuk UNTUK..AKHIR-UNTUK dan translasinya ke pernyataan **for** dapat dilihat pada beberapa contoh berikut:

CONTOH 6.12 Buatlah algoritma dengan menggunakan bentuk UNTUK..AKHIR-UNTUK menampilkan 6 buah baris yg berisi bahasa C selanjutny

a transisis ke **for.**

Algoritma:

UNTUK bil ← 1 S/D 6

tampilkan ("Selamat belajar")

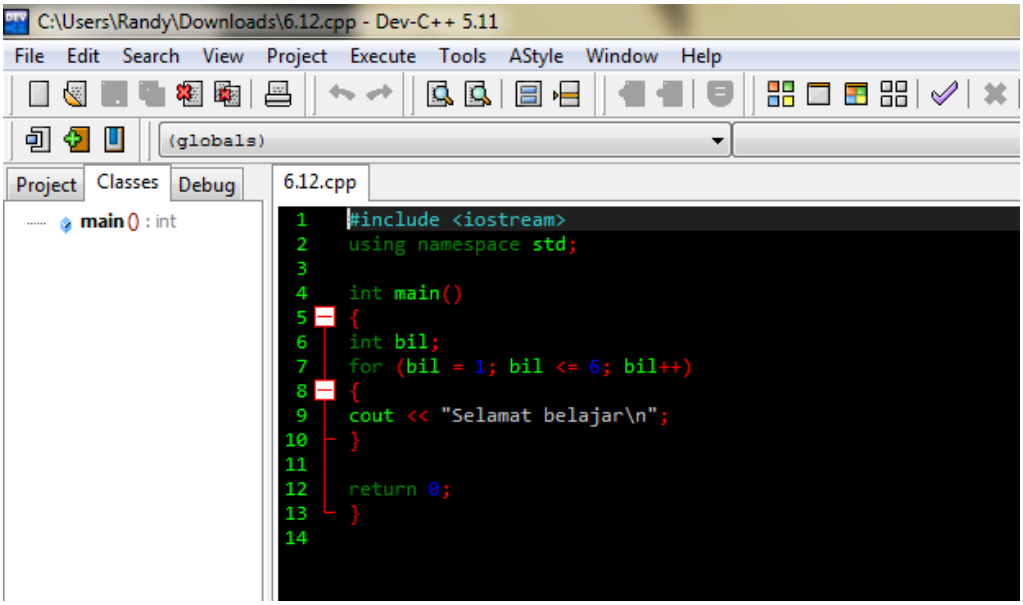
AKHIR-UNTUK

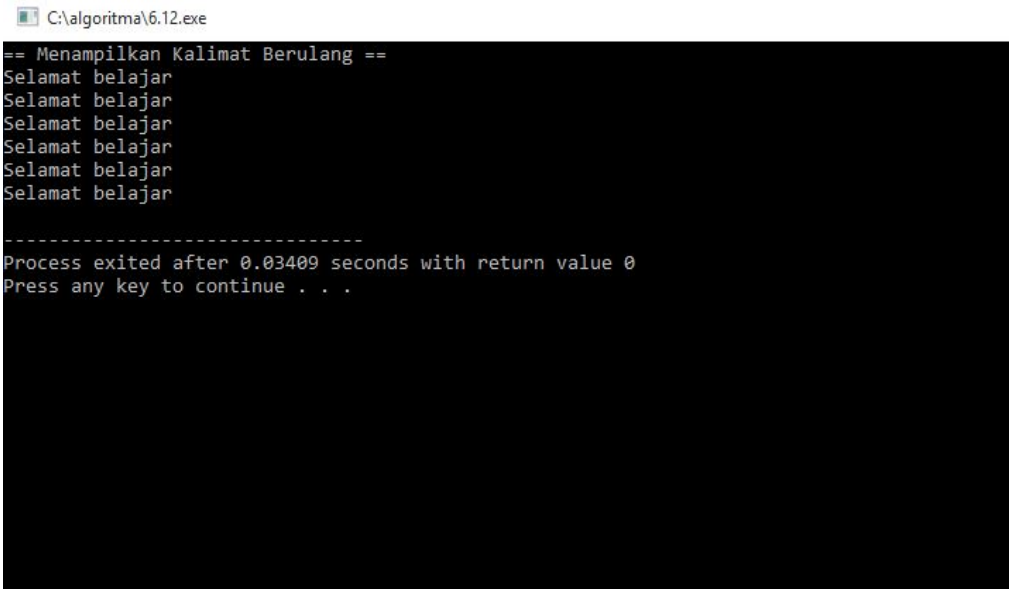
Program:

Penuangan algoritma ke dalam C++ adalah:

Kode Sumber : **enambuah.cpp**

```
#include <iostream.h>

int main()
{
      int bil;
      for (bil = 1; bil <= 6; bil++)
      {
            cout << "Selamat belajar\n";
      }

      return 0;
}
```





Akhir Kode Sumber

Contoh 6.13 Buatlah algoritma dengan menggunakan bentuk UNTUK-AKHIR-UNTUK menghitung

1/2 + 2/3 + 3/4 + ... + 99/100.

Selanjutnya implementasikan ke dalam program C/C++ dengan menggunakan pernyataan **for**

Algoritma:

1. jum ← 0

2. UNTUK bil ← 1 s/d 99

   jum ← jum + bil / (bil + 1)

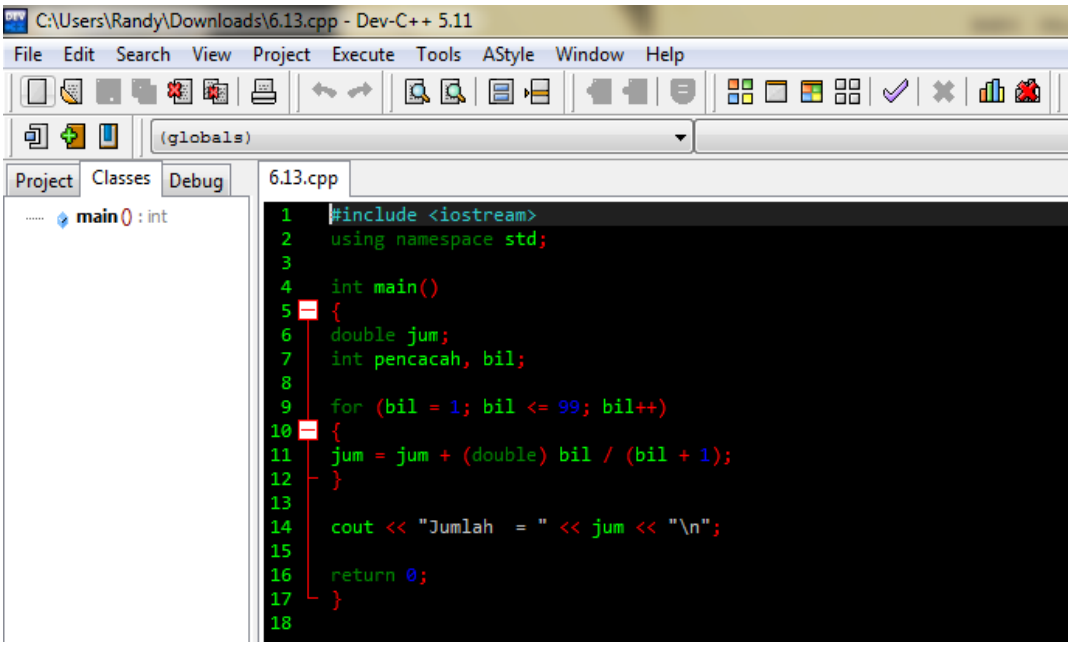
   AKHIR-UNTUK

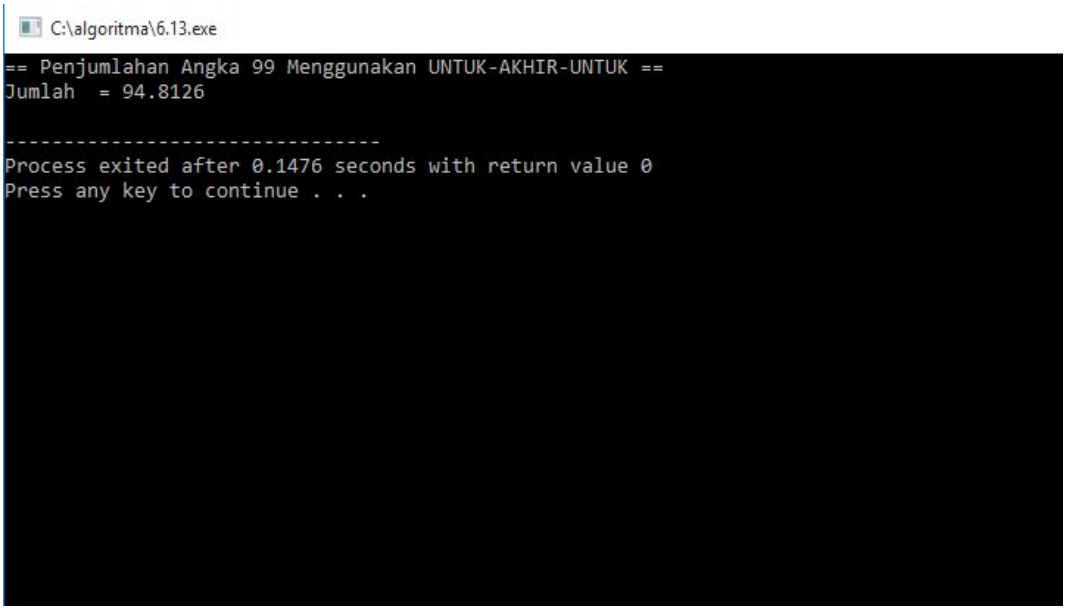
3. tampilkan(jum)

Program :

Kode Sumber : **hitderet.cpp**

```
#include <iostream.h>

int main()
{
      double jum;
      int pencacah, bil;

      for (bil = 1; bil <= 99; bil++)
      {
            jum = jum + (double) bil / (bil + 1);
      }

      cout << "Jumlah  = " << jum << "\n";

      return 0;
}
```





	Akhir Kode Sumber

CONTOH 6.14 Buatlah algoritma untuk menghitung faktorial dengan menggunakan bentuk UNTUK...UNTUK-AKHIR. Perlu diketahui, suatu factorial didefinisikan seperti berikut:

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$$

Sebagai contoh

$$2! = 2 \times 1 = 2$$

3! = 3 x 2 x 1 = 6

4! = 4 x 3 x 2 x 1 = 24

Setelah Anda membuat algoritmanya, translasikan ke program C++.

Algoritma:

1. masukkan(n)
2. hasil ← 1
3. UNTUK bil ← n S/D LANGKAH -1

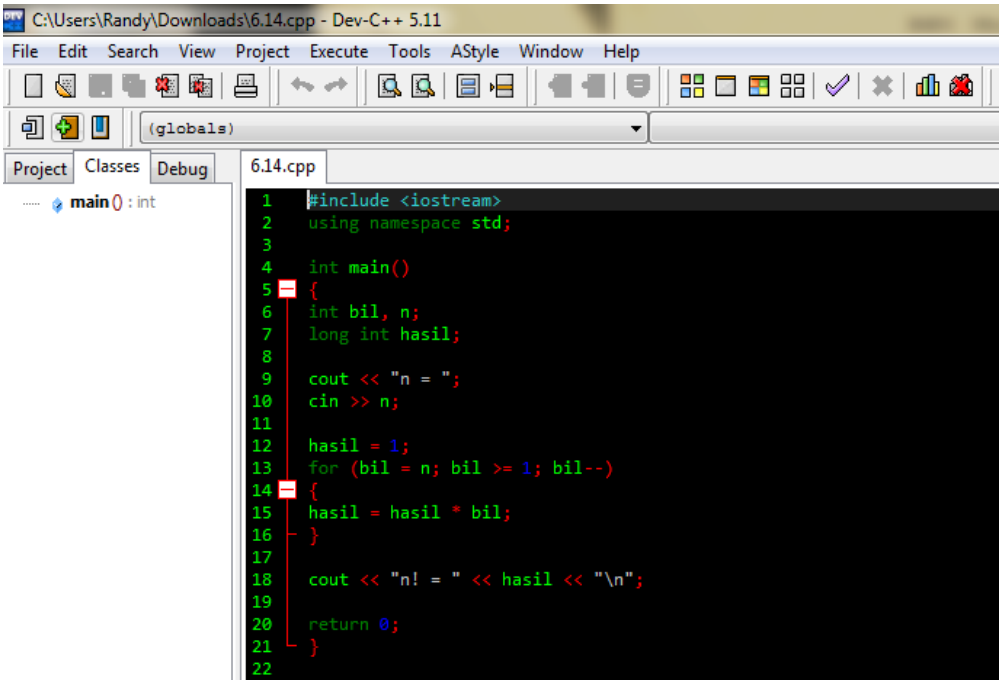
   hasil ← hasil * bil

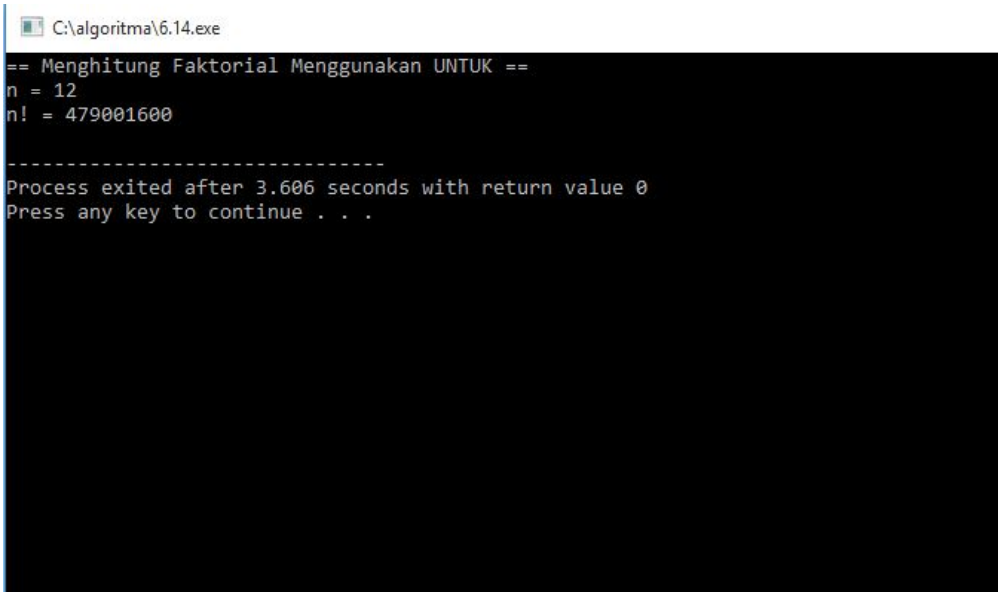
   AKHIR-UNTUK
4. tampilkan(hasil)

Program:

Kode Sumber : **faktorial.cpp**

```
#include <iostream.h>

int main()
{
      int bil, n;
      long int hasil;

      cout << "n = ";
      cin >> n;

      hasil = 1;
      for (bil = n; bil >= 1; bil--)
      {
            hasil = hasil * bil;
      }

      cout << "n! = " << hasil << "\n";

      return 0;
}
```





Akhir Kode Sumber

6.3 BERBAGAI CONTOH APLIKASI

Contoh 6.15 Amir menabung sebesar 5 juta dan setiap tahum mendapatkan bunga majemuk 7%. Buatlah algoritma yang menampilkan nilai uang Amir dari akhir tahun pertama hingga akhir tahun kedelapan. Selanjutnya, konversikan algoritma tersebut ke dalam program C++.

Algoritma:

Masalah ini dapat dipecahkan dengan menggunakan bentuk UNTUK...UNTUK-AKHIR.

Algoritmanya adalah sebagai berikut:

1. pokok ← 5000000
2. UNTUK tahun ← 1 S/D 8

   bunga = pokok * 7/ 10

   tampilkan(tahun, pokok, bunga)

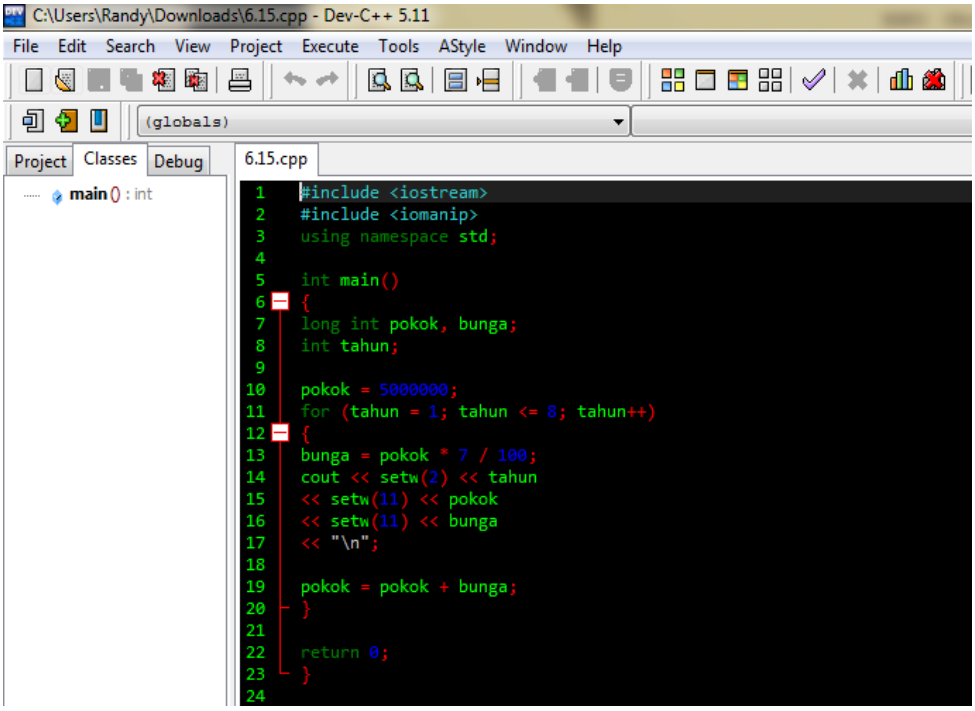
   pokok = pokok + bunga

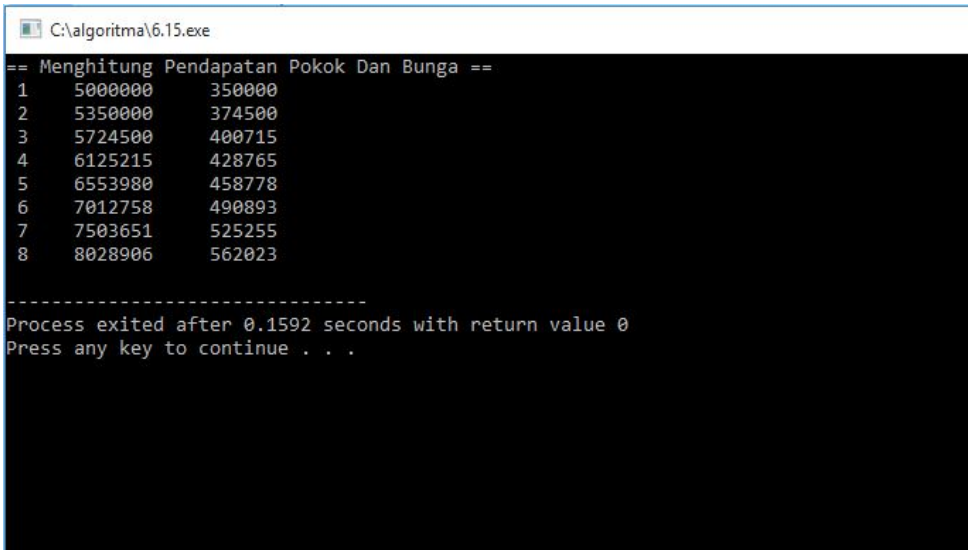
   AKHIR-UNTUK

Program:

Kode Sumber : **bunga.cpp**

```
#include <iostream.h>
#include <iomanip.h>

int main()
{
      long int pokok, bunga;
      int tahun;

      pokok = 5000000;
      for (tahun = 1; tahun <= 8; tahun++)
      {
            bunga = pokok * 7 / 100;
            cout << setw(2) << tahun
            << setw(11) << pokok
            << setw(11) << bunga
            << "\n";

            pokok = pokok + bunga;
      }

      return 0;
}
```





Akhir Kode Sumber

Contoh 6.16 Buatlah algoritma untuk menampilkan semua bilangan ganjil yang terletak antara 1 sampai dengan 25 yang tidak habis dibagi 5.

Algoritma :

UNTUK bil ← 1 S/D 25

JIKA sisa_pembagian(bil,5) ≠ MAKA

tampilkan(bil)

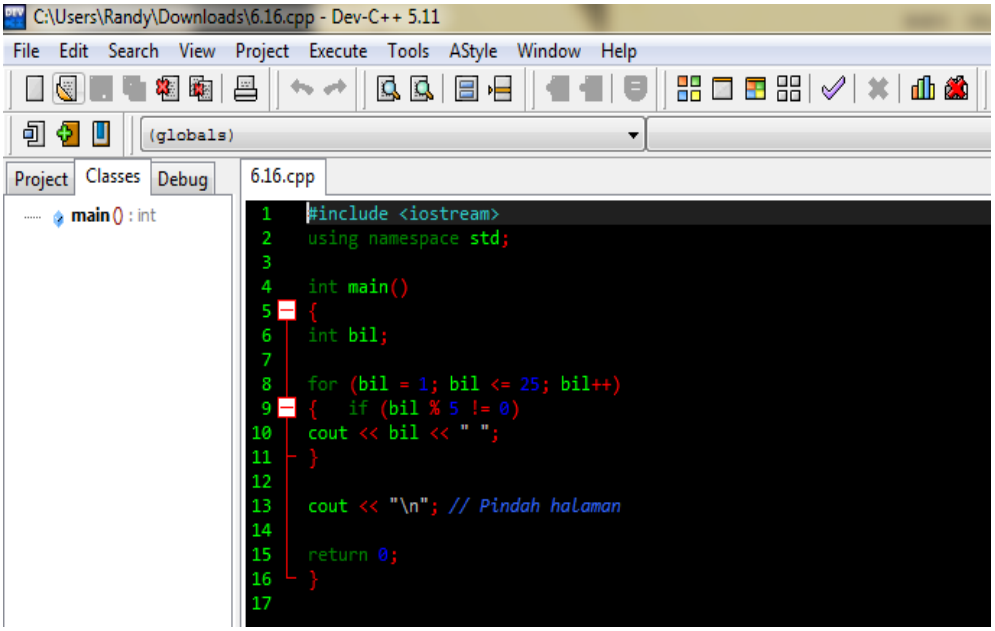
AKHIR-JIKA

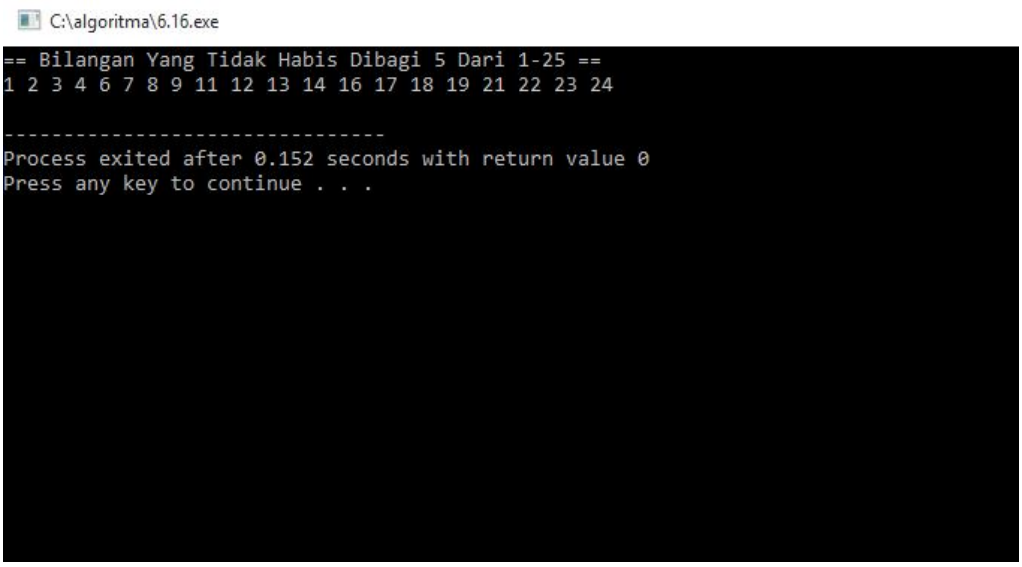
AKHIR-UNTUK

Program:

Kode Sumber : **takhbs5.cpp**

```
#include <iostream.h>

int main()
{
	int bil;

	for (bil = 1; bil <= 25; bil++)
	{	if (bil % 5 != 0)
			cout << bil << " ";
	}

	cout << "\n"; // Pindah halaman

	return 0;
}
```





	Akhir Kode Sumber

CONTOH 6.17 Sebuah bilangan prima adalah bilangan bulat yang hanya habis dibagi dengan 1 atau dirinya sendiri. Berdasarkan ketentuan ini, buatlah algoritma dan program untuk menentukan sebuah bilangan yang dimasukkan dari keyboard, termasuk bilangan prima atau bukan.

Algoritma:

Salah satu pemecahan yang paling gampang adalah dengan membagi bilangan tersebut dengan bilangan dari 1 sampai dengan bilangan itu sendiri. Jika bilangan tersebut adalah bilangan prima maka bilangan tersebut hanya habis dibagi oleh dua bilangan, yaitu bilangan 1 dan bilangan itu sendiri. Algoritmanya adalah sebagai berikut:

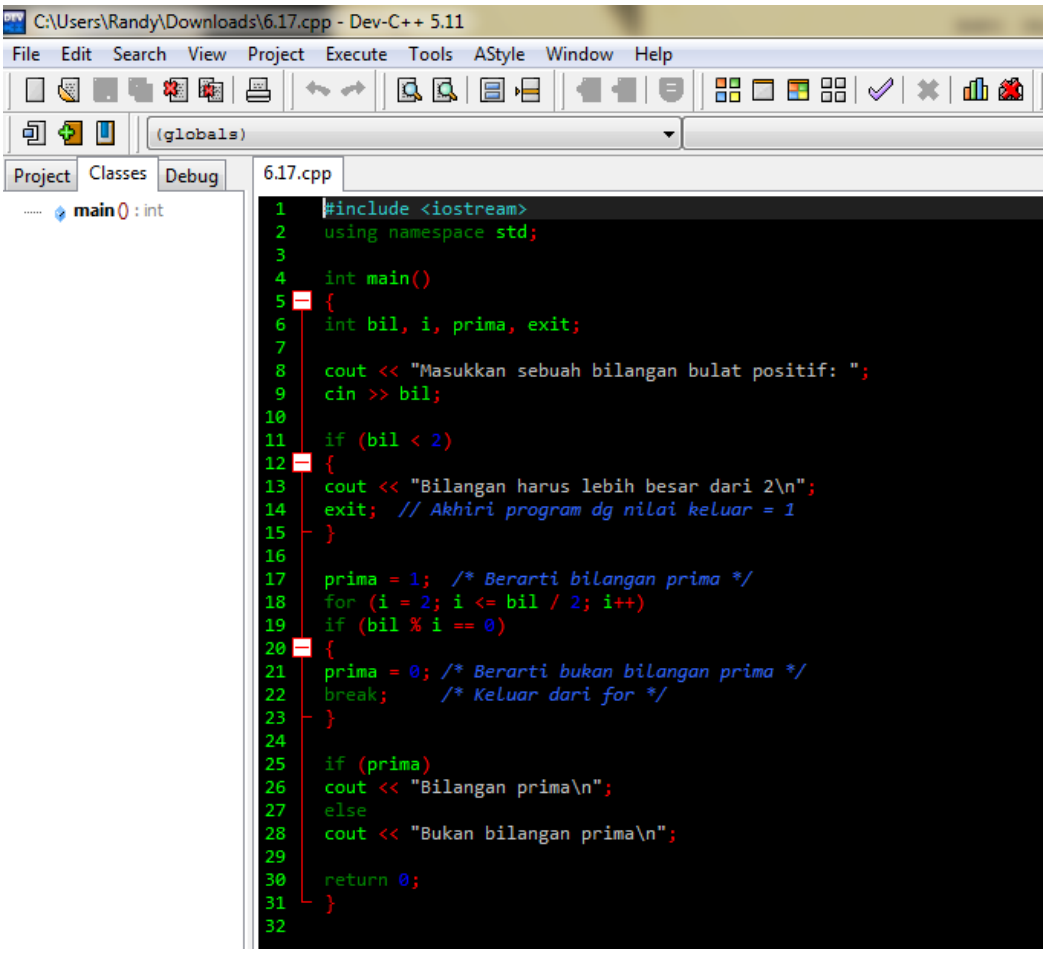
1. masukkan(bil)
2. jum = 0
3. UNTUK i ← 1 S/D bil
   JIKA sisa_pembagian(i, bil) = 0 MAKA
   jum = jum + 1
   AKHIR-JIKA
4. JIKA jum = 2 MAKA
   tampilkan(“Bilangan prima”)
   SEBALIKNYA
   tampilkan(“Bukan bilangan prima”)

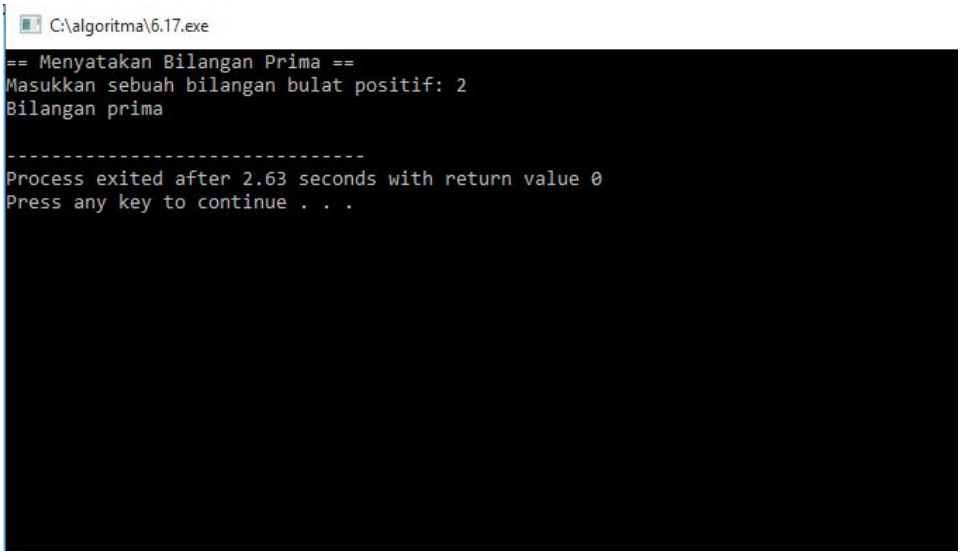
AKHIR-JIKA

Program:

Kode Sumber : **prima.cpp**

```
#include <iostream.h>

int main()
{
      int bil, jum, i;

      cout << "Masukkan sebuah bilangan bulat positif: ";
      cin >> bil;

      jum = 0;
      for (i = 1; i <= bil; i++)
            if (bil % i == 0)
                  jum++;
      if (jum == 2)
            cout << "Bilangan prima\n";
      else
            cout << "Bukan bilangan prima\n";

      return 0;
}
```





Akhir Kode Sumber

Contoh 6.18

Menurut Lipschutz dan Poe (1982, hal. 87), khusus untuk pengujian bilangan lebih besar daripada 2, bilangan prima bisa ditentukan dengan memeriksa pembagian dari

2 sampai n/2 saja jika n adalah bilangan yang diuji. Jadi tidak perlu menguji dari 1 sampai dengan n. Tuangkan hal ini dalam bentuk algoritma dan kemudian translasikan ke dalam program C++.

Algirtma:

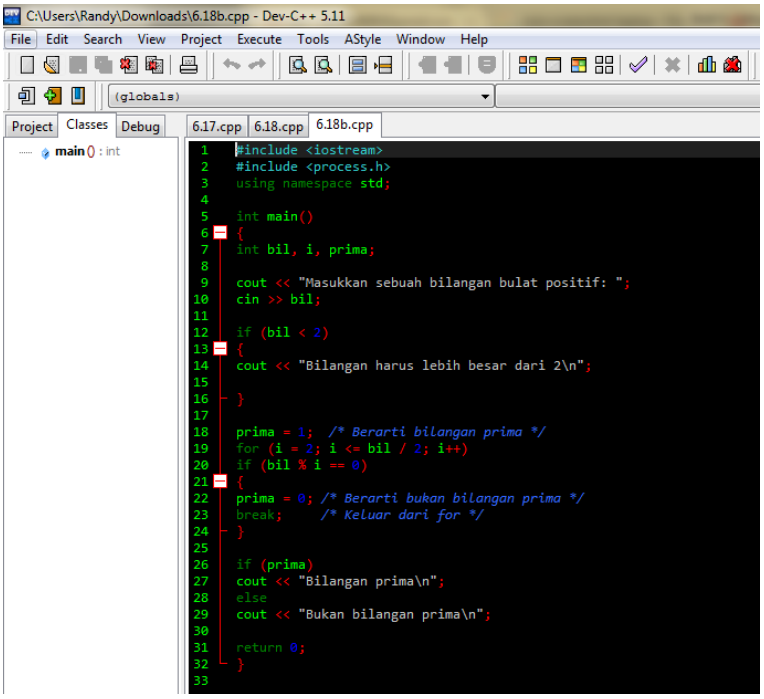
1. masukkan(bil)
2. JIKA bil < 2 MAKA
   tampilkan("Bilangan harus lebih besar dari 2")
   hentikan program
   AKHIR-JIKA
3. prima ← benar
4. UNTUK i ← 1 S/D bil/2
   JIKA sisa_pembagian(i, bil) = 0 MAKA
   prima ← salah
   keluar dari UNTUK..AKHIR-UNTUK
   AKHIR-JIKA
   AKHIR-UNTUK
5. JIKA prima = benar MAKA
   tampilkan("Bilangan prima")
   SEBALIKNYA
   tampilkan("Bukan bilangan prima")
   AKHIR-JIKA

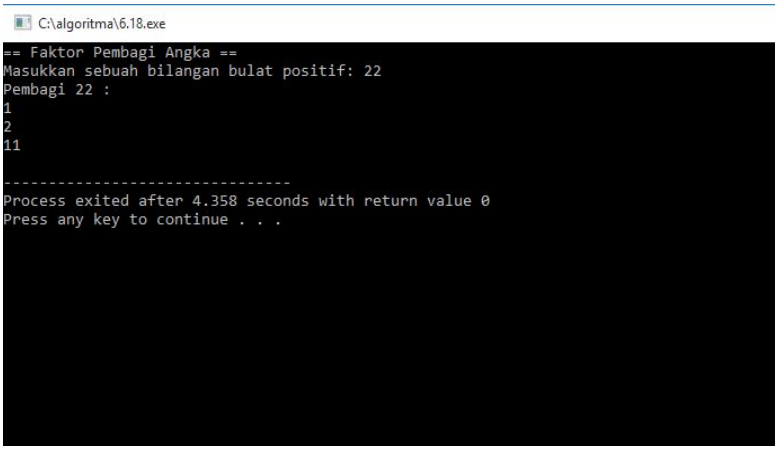
Program:

Kode Sumber : **prima2.cpp**

```
#include <iostream.h>
#include <process.h>

int main()
{
      int bil, i, prima;

      cout << "Masukkan sebuah bilangan bulat positif: ";
      cin >> bil;

      if (bil < 2)
      {
            cout << "Bilangan harus lebih besar dari 2\n";
```

```
            exit(1);  // Akhiri program dg nilai keluar = 1
      }

      prima = 1;  /* Berarti bilangan prima */
      for (i = 2; i <= bil / 2; i++)
            if (bil % i == 0)
            {
                  prima = 0; /* Berarti bukan bilangan prima */
                  break;     /* Keluar dari for */
            }

            if (prima)
            cout << "Bilangan prima\n";
      else
            cout << "Bukan bilangan prima\n";

      return 0;
}
```





Akhir Kode Sumber

Contoh 6.18 Buatlah algoritma dan program yang meminta sebuah bilangan bulat dimasukkan dari keyboard dan kemudian menampilkan semua bilangan yang menjadi faktor pembagi bilangan tersebut. Contoh, bila bilangan yang dimasukkan adalah 12, maka hasilnya berupa:

Pembagi 12:

1

2

3

4

6

12

Algoritma:

1. masukkan(bil)
2. tampilkan(“Pembagi”, n,”:””)
3. UNTUK i ←1 S/D n

        JIKA sisa_pembagian(i,bil) = 0 MAKA

                tampilkan(bil)

        AKHIR-JIKA

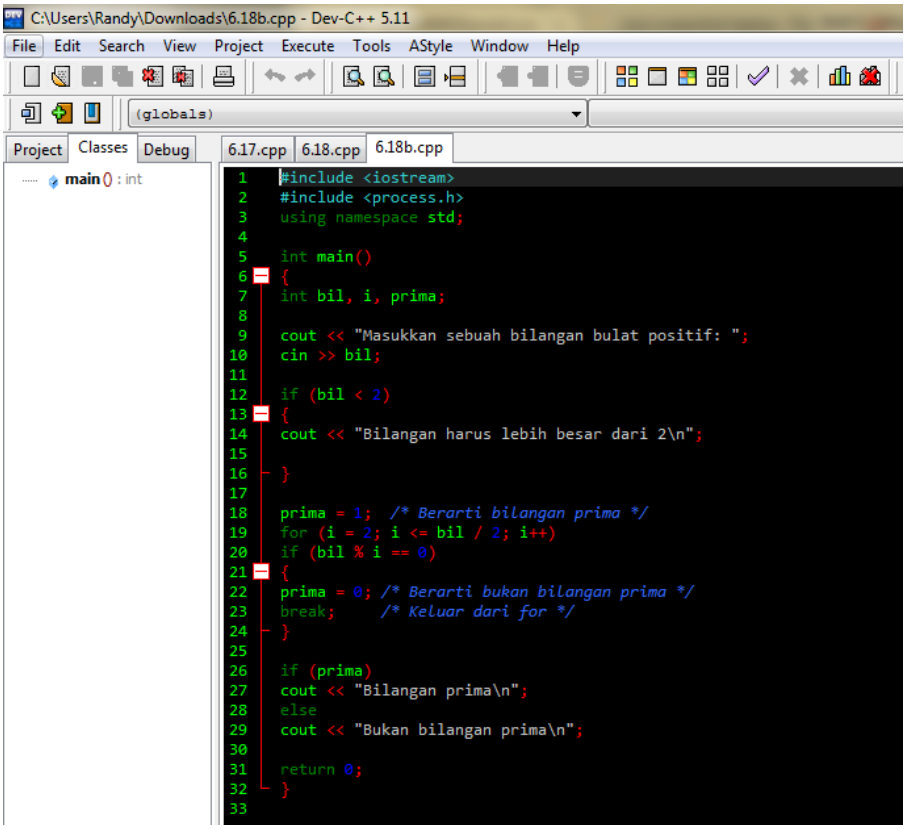
    AKHIR-UNTUK

Program:

Kode Sumber : **pembagi.cpp**

```
#include <iostream.h>

int main()
{
      int bil, i, prima;

      cout << "Masukkan sebuah bilangan bulat positif: ";
      cin >> bil;

      cout << "Pembagi "<< bil << " : \n";

      for (i = 1; i <= bil / 2; i++)
            if (bil % i == 0)
```

```
                cout << i << "\n";

        return 0;
}
```



Akhir Kode Sumber

Contoh 6.19 Tuliskan algoritma yang membaca data dari keyboard secara terus-menerus sampai nilai -99999 dimasukkan dan kemudian menampilkan nilai terbesarnya dan nilai rata-ratanya. Setelah itu translasikan ke dalam program C++.

Algoritma:

1. selesai← SALAH
2. pencacah ← 0
3. jum ← 0
4. ULANGAN SELAMA selesai = SALAH

masukan(bil)

JIKA bil = -99999 MAKA

selesai ← BENAR

SEBALIKNYA

pencacah ← pencacah + 1

jum ← jum + bil

JIKA pencacah = 1 MAKA

terbesar ← bil

SEBALIKNYA

JIKA bil > terbesar MAKA

tebesar ← bil

AKHIR-JIKA

AKHIR-JIKA

AKHIR-JIKA

AKHIR-ULANG

5. JIKA pencacah = 0 MAKA

tampilkan(“tak ada data yang dimasukkan”)

SEBALIKNYA

rata_rata ← jum / pencacah

tampilkan(“terbesar =”,terbesar)

tampilkan(“rata-rata=”,rata-rata)

AKHIR-JIKA

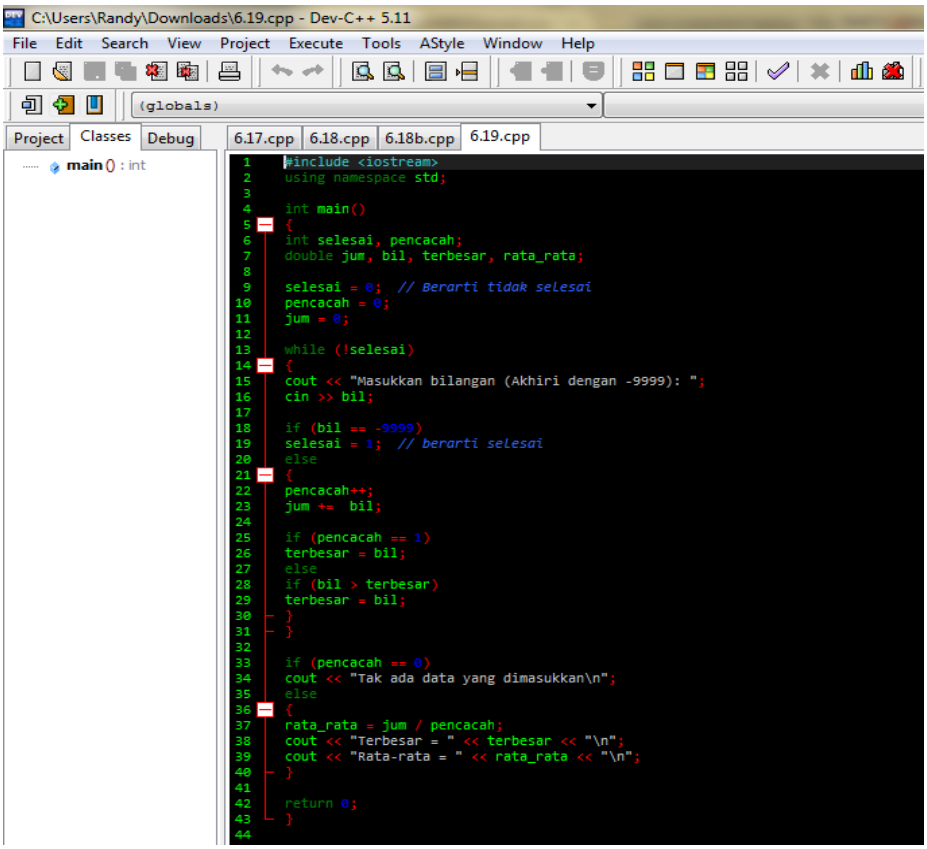
Program:

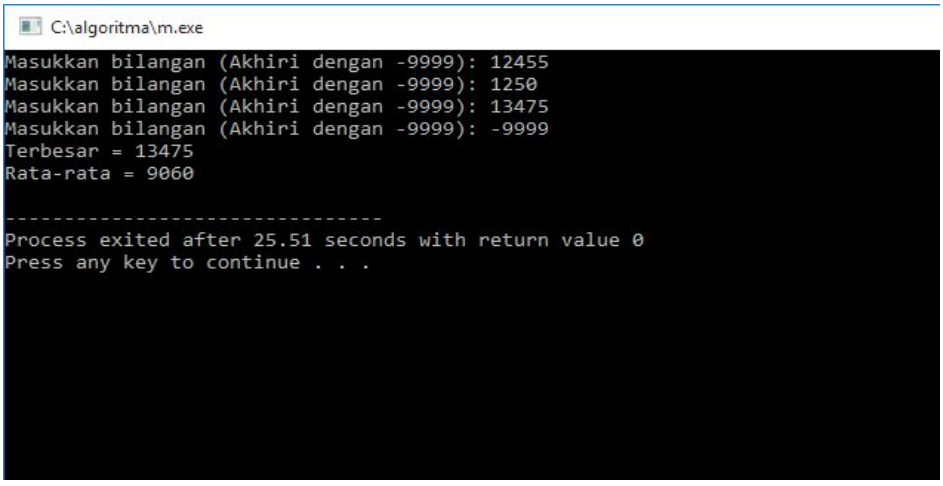
Adapun implementasi dalam C++ adalah seperti berikut:

Kode Sumber : **maksrata.cpp**

```
#include <iostream.h>

int main()
{
      int selesai, pencacah;
      double jum, bil, terbesar, rata_rata;

      selesai = 0;  // Berarti tidak selesai
      pencacah = 0;
```

```
	jum = 0;

	while (!selesai)
	{
		cout << "Masukkan bilangan (Akhiri dengan -9999): ";
		cin >> bil;

		if (bil == -9999)
			selesai = 1;  // berarti selesai
		else
		{
			pencacah++;
			jum +=  bil;

			if (pencacah == 1)
				terbesar = bil;
			else
				if (bil > terbesar)
					terbesar = bil;
		}
	}

	if (pencacah == 0)
		cout << "Tak ada data yang dimasukkan\n";
	else
	{
		rata_rata = jum / pencacah;
		cout << "Terbesar = " << terbesar << "\n";
		cout << "Rata-rata = " << rata_rata << "\n";
	}

	return 0;
}
```





Akhir Kode Sumber

Contoh 6.20 Sinus X (dalam radian) dapat dihitung dengan pendekatan:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

Tuislah algoritma yang menghitung nilai sinus tersebut untuk 10 suku dengan nilai x dimasukkan dari keyboard. Kemudian translasikan algoritma tersebut ke dalam

program C++ dan bandingkan pula hasilnya kalau menggunakan fungsi **sin()** yang tersedia pada pustaka matematika.

Algoritma:

1. masukkan(x)
2. sinx ← x //Mulai dari suku pertama
3. UNTUK i ← 2 S/D 10

        //Hitung (2i-1)!
        faktorial ← 1
        UNTUK j ← 2 x i - 1 S/D 1 LANGKAH -1
                faktorial = faktorial x j
        AKHIR
                //Hitung x pangkat (2i-1)
        hasil_pangkat ← 1.0
        UNTUK j ← 1 S/D 2 x i - 1
                hasil_pangkat = hasil_pangkat * x
        AKHIR-UNTUK
        //Hitung jumlah sampai suku ke-i
        JIKA sisa_pembagian(i, 2) = 0 MAKA //i adalah genap
                sinx ← sinx – hasil_pangkat / factorial
        SEBALIKNYA
                sinx ← sinx + hasil_pangkat / faktorial
        AKHIR-JIKA
    AKHIR-UNTUK
4. tampilkan(sinx)

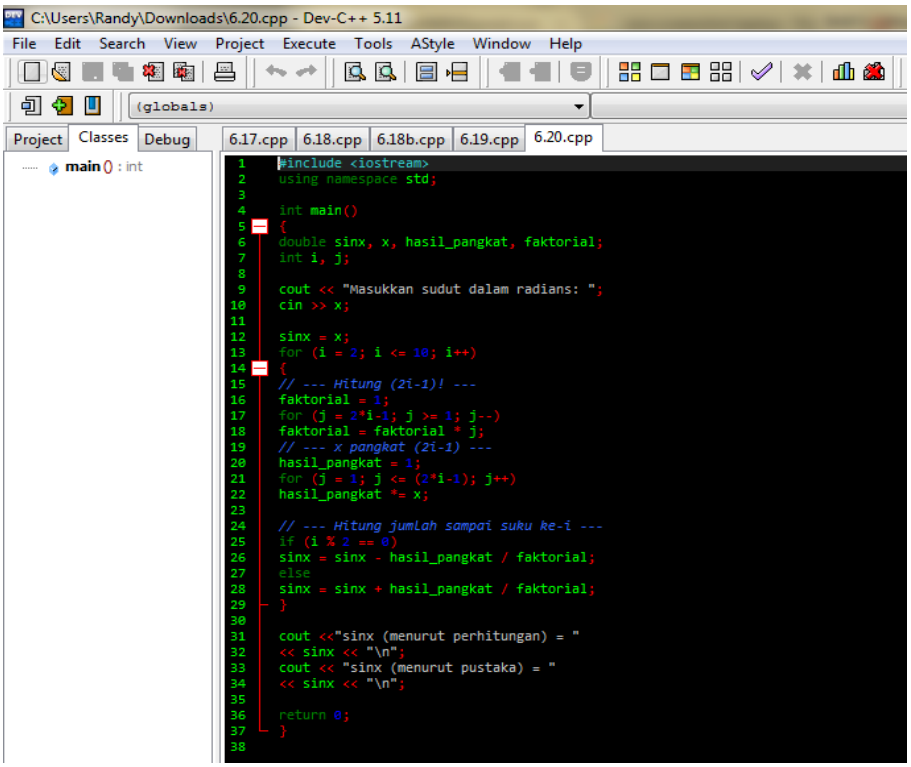
Program:

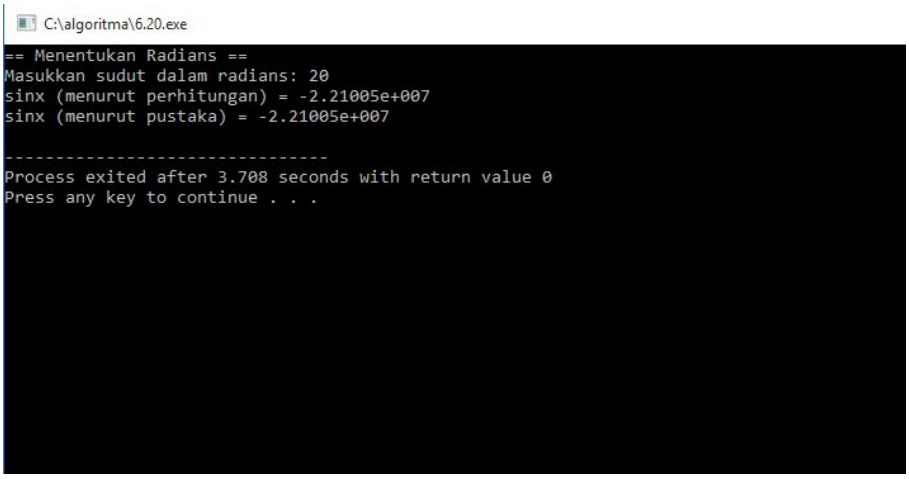
Program C++ nya adalah sebagai berikut:

Kode Sumber : **sinx.cpp**

```
#include <iostream.h>
#include <math.h>

int main()
{
      double sinx, x, hasil_pangkat, faktorial;
```

```
	int i, j;

	cout << "Masukkan sudut dalam radians: ";
	cin >> x;

	sinx = x;
	for (i = 2; i <= 10; i++)
	{
		// --- Hitung (2i-1)! ---
		faktorial = 1;
		for (j = 2*i-1; j >= 1; j--)
			faktorial = faktorial * j;
		// --- x pangkat (2i-1) ---
		hasil_pangkat = 1;
		for (j = 1; j <= (2*i-1); j++)
			hasil_pangkat *= x;

		// --- Hitung jumlah sampai suku ke-i ---
		if (i % 2 == 0)
			sinx = sinx - hasil_pangkat / faktorial;
		else
			sinx = sinx + hasil_pangkat / faktorial;
	}

	cout <<"Sinx (menurut perhitungan) = "
		<< sinx << "\n";
	cout << "Sinx (menurut pustaka) = "
		<< sin(x) << "\n";

	return 0;
}
```





Akhir Kode Sumber

Contoh 6.21 Tulislah algoritma untuk membuat segitiga yang di susun dari simbol * dengan tinggi segitiga di tentukan melalui keyboard. Beberapa contoh hasilnya di perlihatkan di bawah ini.

```
*          *          *
**         **         **
***        ***        ***
```

```
****              ****              ****
                  *****             *****
                                    ******
Tinggi = 4        Tinggi = 5        Tinggi = 6
```

Gambar 6.19*Gambaran segitiga berpola**

Buat pula programnya.

Algoritma:

1. masukkan (tinggi)
2. UNTUK i ← 1 S/D tinggi

    UNTUK j ← 1 S/D i

        tampilkan(“*”) // Tanpa pindah baris

    AKHIR-UNTUK

    tampilkan(karkter_pindah_baris) // Untuk berpindah baris

AKHIR-UNTUK

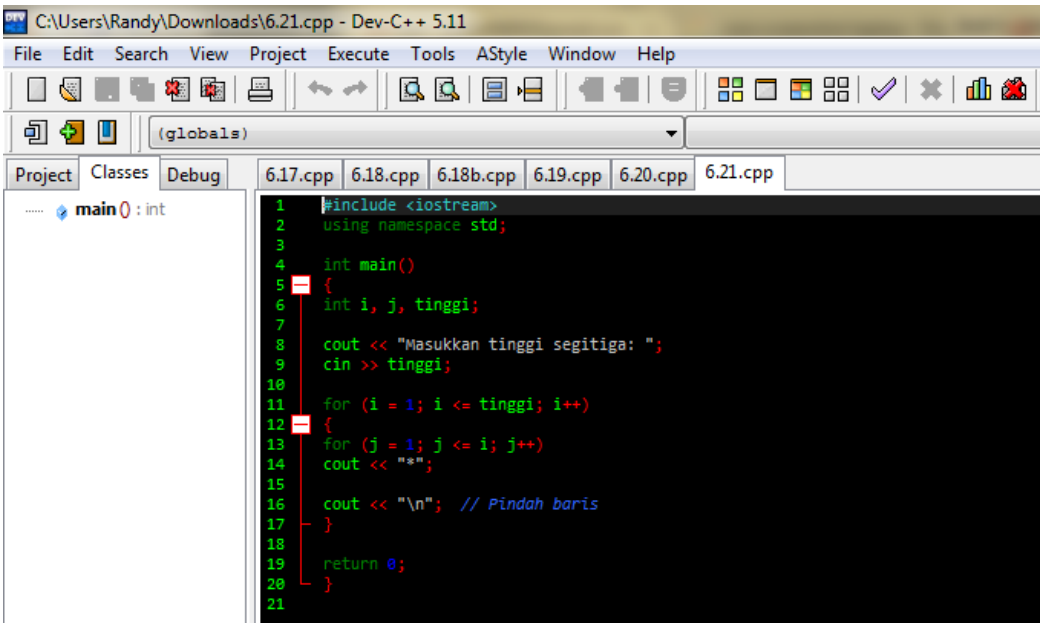
Program:

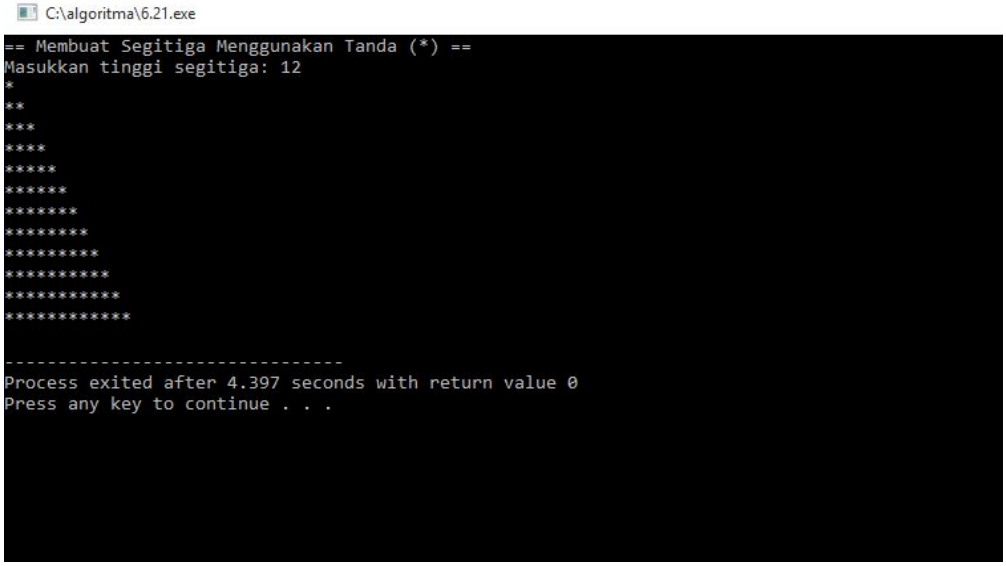
Adapun implementasi dalam C++ adalah seperti berikut:

Kode Sumber : **segitiga.cpp**

```
#include <iostream.h>

int main()
{
      int i, j, tinggi;

      cout << "Masukkan tinggi segitiga: ";
      cin >> tinggi;

      for (i = 1; i <= tinggi; i++)
      {
            for (j = 1; j <= i; j++)
            cout << "*";
```

```
            cout << "\n";  // Pindah baris
        }

        return 0;
    }
```





Akhir Kode Sumber

Contoh 6.22 Buatlah algoritma dan programuntuk menyusun hasil seperti berikut :

1

2 6

3 7 10

4 8 11 13

5 9 12 14 15

Dalam hal ini jumlah baris ditentukan melalui keyboard. Buat pula programnya.

Algoritma:

1. Masukkan (tinggi)
2. UNTUK i ← 1 S/D tinggi

    bil ← i

    UNTUK j ← 1 S/D i

        tampilkan (bil) // Tanpa pindah baris

        bil ← bil + tinggi – j

    AKHIR – UNTUK

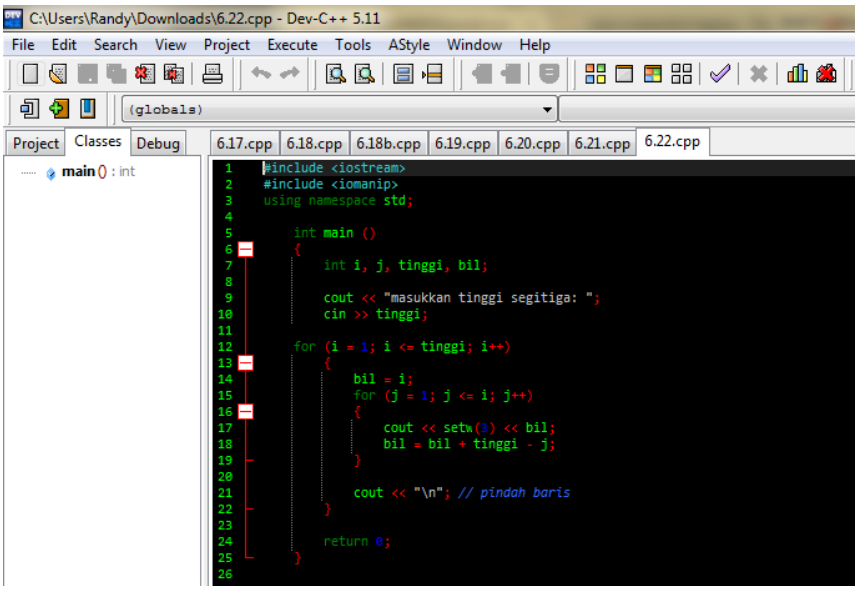
    tampilkan (karakter_pindah_baris) // Untuk berpindah baris

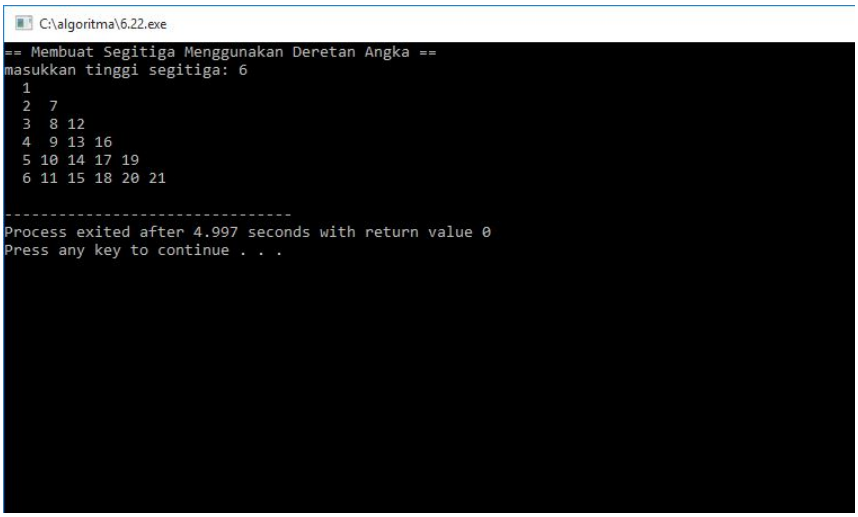
AKHIR – UNTUK

Kode Sumber : **sgtgbil.cpp**

```
#include <iostream.h>
#include <iomanip.h>

Int main ()
{
      int I, j, tinggi, bil;

      cout << “masukkan tinggi segitiga: “;
      cin >> tinggi:

      for (i = 1; I <= tinggi; i++)
      {
            Bil = I;
            For (j = 1; j <= i; j++)
            {
```

```
                Cout << setw(3) << bil;
                bil = bil + tinggi – j;
            }

            Cout << “\n”; // pindah baris
        }

        return 0;
    }
```





	Akhir Kode Sumber

Contoh 6.23 Buatlah algoritma untuk menyusun table seperti berikut:

4 8 12 16 20

3 7 11 15 19

2 6 10 14 18

1 5 9 13 17

Dalam hal ini jumlah baris dan jumlah kolom ditentukan dari keyboard. Setelah Anda menyusun algoritmanya, tuangkan ke dalam program C/C++.

Algoritma:

1. masukkan (jum_baris)
2. masukkan (jum_kolom)
3. UNTUK i ← jum_baris S/D 1 LANGKAH -1

   bil ← i

   UNTUK j ← 1 S/D jum_kkolom

   tampilkan (bil) // tanpa pindah baris

   bil ← bil + jum_baris

   AKHIR–UNTUK

   tampilkan (karakter_pindah_baris) // untuk berpindah baris

AKHIR – UNTUK

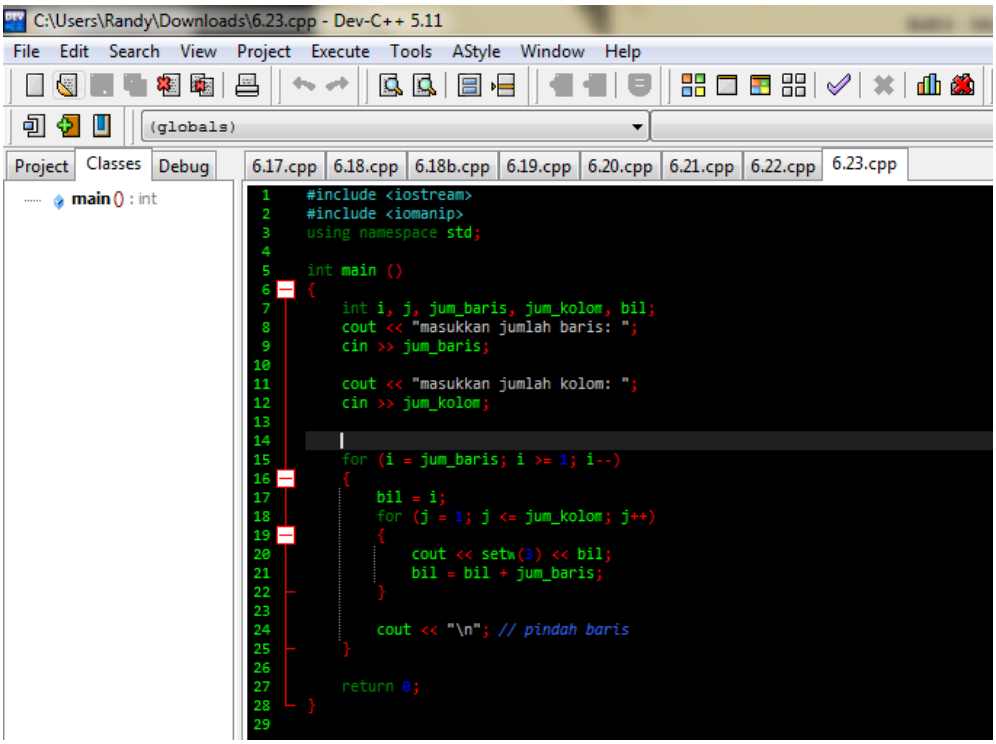
Contoh hasil pengeksekusian program:

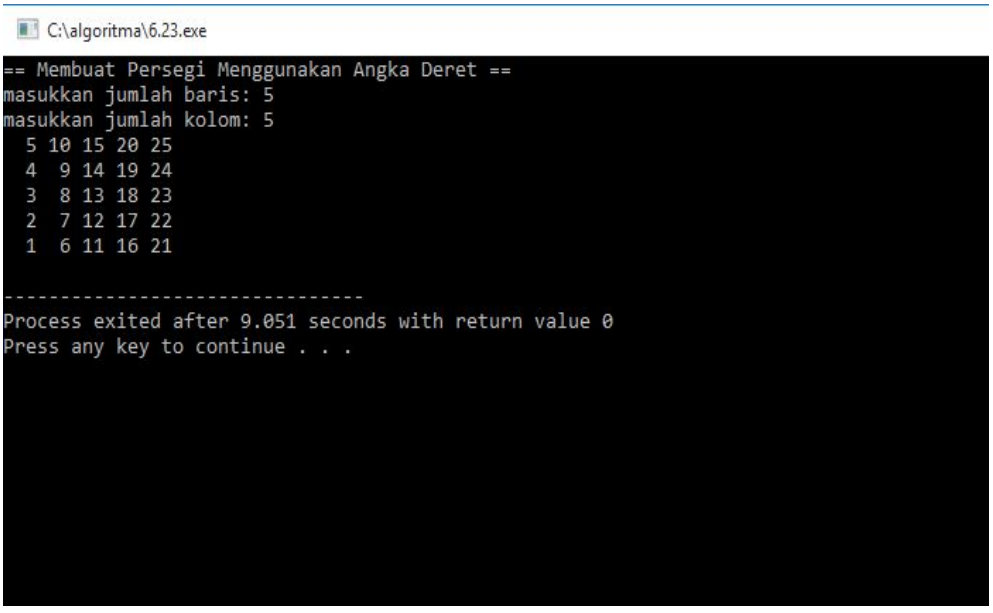
Gambar 6.22

Kode Sumber : **tabel.cpp**

```
#include <iostream.h>
#include <iomanip.h>

int main ()
{
      int i, j, jum_baris, jum_kolom, bil;
      cout << “masukkan jumlah baris: “;
      cin >> jum_baris;

      cout << “masukkan jumlah kolom: “;
      cin >> jum_kolom;


      for (i = jum_baris; i >= 1; i--)
      {
            bil = i;
            for (j = 1; j <= jum_kolom; j++)
            {
                  Cout << setw(3) << bil;
                  bil = bil + jum_baris;
            }
```

```
            Cout << “\n”; // pindah baris
        }

        return 0;
}
```





Akhir Kode Sumber

Contoh 6.24 Tuliskan algoritma untuk melakukan penjumlahan seperti berikut:

1 – 1/2 + 1/3 – 1/4 + 1/5 - …

Sampai suku yang ke n (n dimasukkan dari keyboard). Jangan lupa untuk membuat programnya.

Algoritma:

1. masukkan(n)
2. tanda ← -1
3. jum ← 1
4. UNTUK i ← 2 S/D n

   JIKA sisa_pembagian (i, 2) = 0 MAKA // Berarti bilangan genap

   jum ← jum – 1/i

   SEBALIKNYA

   jum ← jum + 1/i

   AKHIR–JIKA

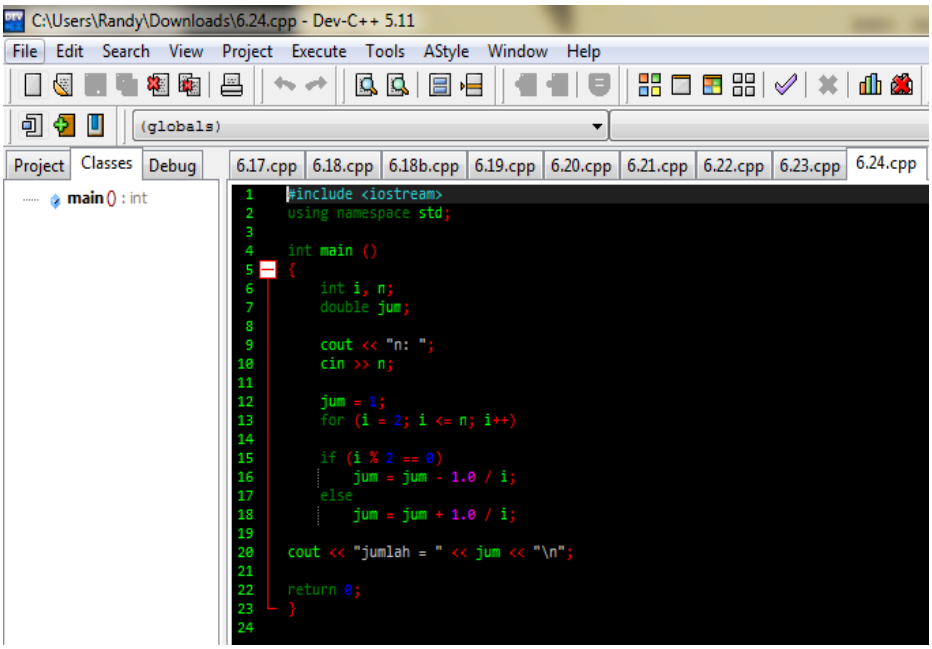
   AKHIR–UNTUK

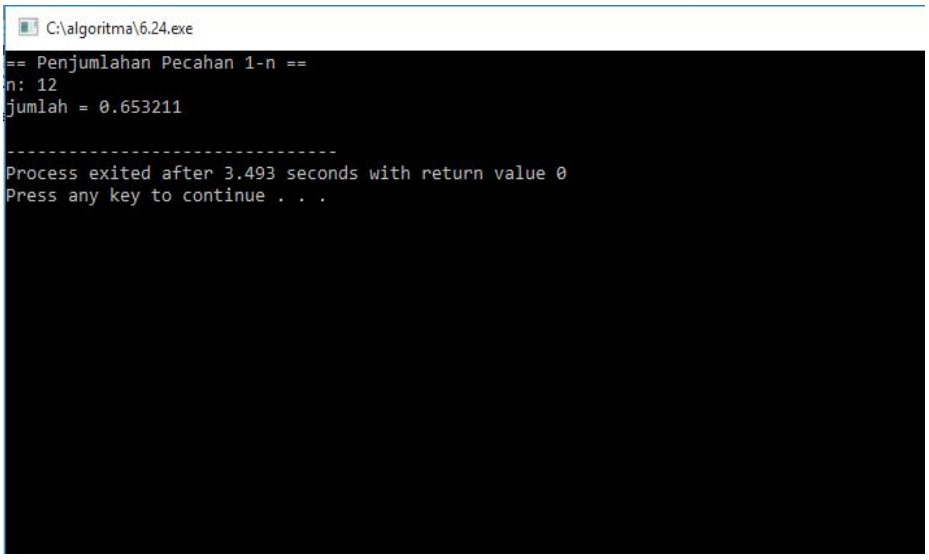
   tampilkan(jum)

Kode Sumber : **jumbagi.cpp**

```
#include <iostream.h>

int main ()
{
      int i, n;
      double jum;

      cout << "n: ";
      cin >> n;

      jum = 1;
      for (i = 2; i <= n; i++)

      if (i % 2 == 0)
            jum = jum – 1.0 / i;
      else
            jum = jum + 1.0 / i;

cout << "jumlah = " << jum << "\n";
```

```
return 0;
}
```





	Akhir Kode Sumber

Alternatif lain:

Alternatif lain untuk menyelesaikan persoalan di depan adalah dengan tanpa memeriksa bilangan genap atau bilangan ganjil. Cara detailnya diperlihatkan pada algoritma berikut:

1. masukkan(n)
2. tanda ← -1

3. jum ← 1
4. UNTUK i ← 2 S/D n

        jum ← jum + tanda / i

        tanda ← -tanda

    AKHIR – UNTUK

    tampilkan(jum)

Dengan memberikan

    tanda ← -tanda

isi tanda akan berubah menjadi -1, 1, -1, 1, -1, dan seterusnya.

Kode Sumber : **jumbagi2.cpp**

```
#include <iostream.h>

int main ()
{
      int i, n;
      double jum, tanda;

      cout << “n: “;
      cin >> n;

      tanda = -1;
      jum = 1;
      for (i = 2; i <= n; i++)
      {
            Jum = jum + tanda / i;
            Tanda = -tanda;
      }

      Cout << “jumlah = “ << jum << “\n”;

      return 0;
}
```

Akhir Kode Sumber

Contoh 6.25 Buatlah algoritma dan program yang membaca suatu string dari keyboard dan kemudian tampilkan semacam berikut:

    Yogyakarta

ogyakarta

gyakarta

yakarta

karta

arta

rta

ta

a

Jika string yang dimasukkan adalah “Yogyakarta”.

Algoritma:

1. masukkan(string)
2. UNTUK i ← 0 S/D panjang (string)-1

   UNTUK j ← i S/D panjang (string)-1

   tampilkan(string[j]) // tanpa pindah baris

   AKHIR – UNTUK

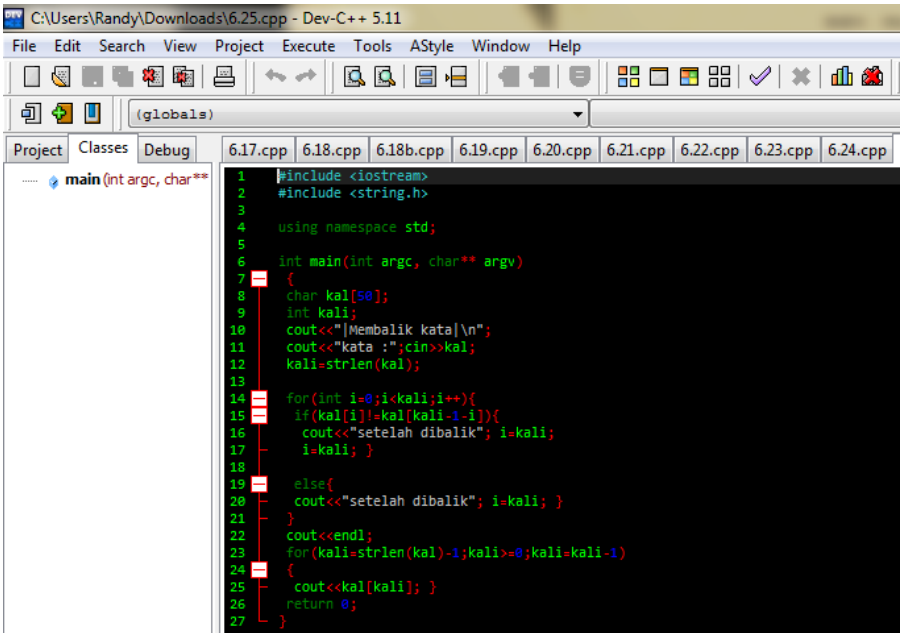
   tampilkan(karakter_pindah_baris) // Untuk berpindah baris

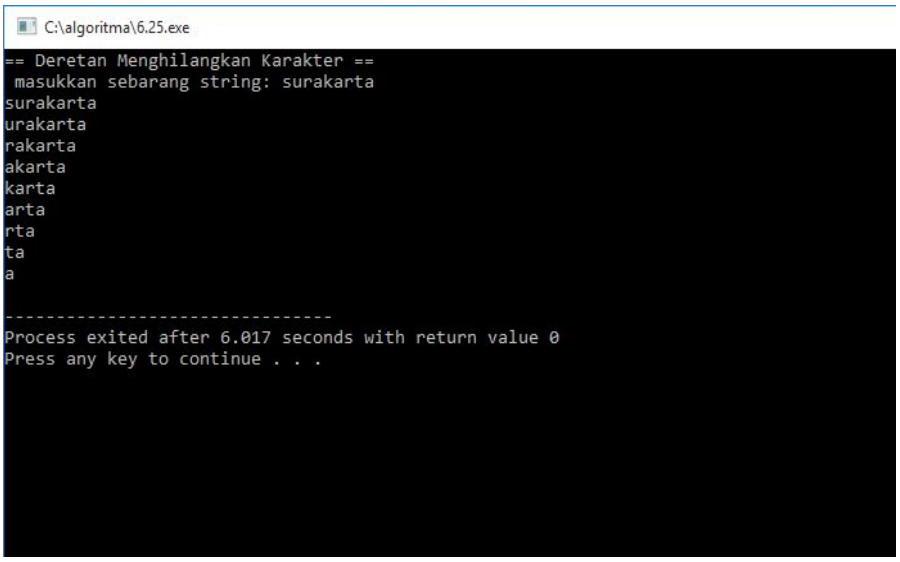
AKHIR – UNTUK

Kode Sumber :**animstr.cpp**

```
#include <iostream.h>
#include <string.h>

int main ()
{
      int i, j, panjang;
      char string [80];

      cout << “ masukkan sebarang string: “;
      cin >> string;

      panjang = strlen (string) ;
      for (i = 0; i < panjang; i++)
      {
            for (j = i; j < panjang; j++)
                  cout << string [j] ;

            cout << “\n”; // pindah baris
```

```
        }

        return 0;
    }
```





	Akhir Kode Sumber

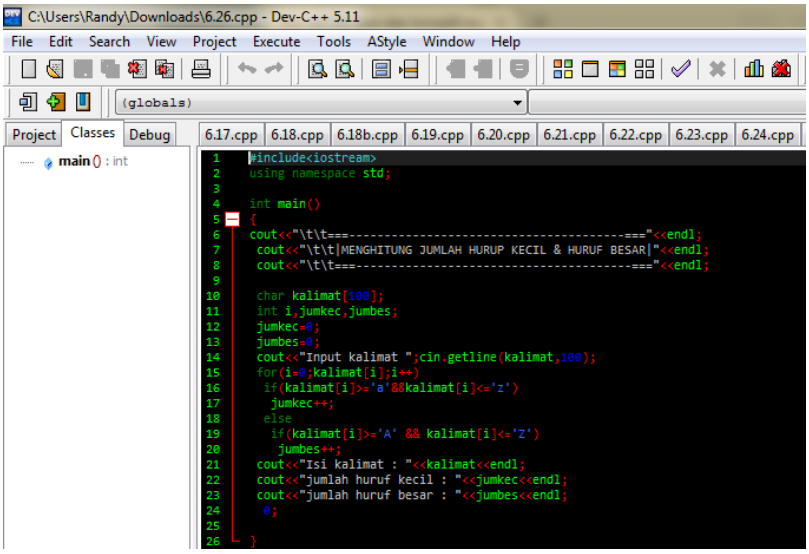
Contoh 6.26 Buatlah algoritma dan program yang meminta sebuah kalimat dimasukkan dari keyboard dan kemudian menyajikan jumlah huruf capital yang terdapat pada kalimat tersebut.

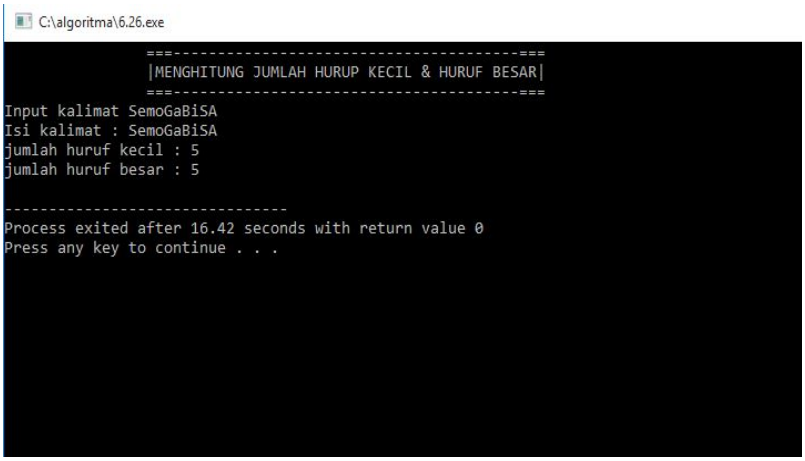
Algoritma:

1. masukkan(string)
2. jum ← 0
3. UNTUK i ← 0 S/D panjang(string)-1
   kar ← string[i]
   JIKA kar ≥ " A" DAN kar ≤ "Z" MAKA
   jum ← jum + 1
   AKHIR – JIKA
   AKHIR – UNTUK
4. tampilkan(jum)

Kode Sumber :**jumkap.cpp**

```
#include <iostream.h>
#include <string.h>

int main ()
{
      int i, jum;
      char string [80] ;

      cout << "masukkan sebarang string: ";
      cin >> string;

      jum = 0;
      for (i = 0; i < (int) strlen (string) ; i++)
      {
            Kar = string [i] ;
            If ((kar >= 'A') && (kar <= 'Z'))
                  Jum++;
      }

      Cout << "jumlah huruf kapital = " << jum << "\n";

      return 0 ;
}
```





Akhir Kode Sumber

Contoh 6.27 Buatlah algoritma dan program yang meminta sebuah kalimat dimasukkan dari keyboard dan kemudian menampilkan masing – masing karakter dengan ketentuan sebagai berikut:

- Huruf kecil diubah ke huruf capital
- Huruf capital diubah ke huruf kecil

Perlu diketahui, huruf kecil dan huruf capital yang setara meiliki selisih nilai ASCII sebesar 32. Contoh:

- Nilai ASCII A adalah 65
- Nilai ASCII a adalah 97

Algoritma:

1. masukkan(string)

2. UNTUK i ← 0 S/D panjang(string)-1

    kar ← string[i]

    JIKA kar ≥ “A” DAN kar ≤ “Z” MAKA

        Nilai ASCII = ASCII(kar) + 32

        tampilkan(karakter_bernilai_ASCII_berupa_nilai ASCII)

    SEBALIKNYA

        JIKA kar ≥ “a” DAN kar ≤ “z” MAKA

            Nilai ASCII = ASCII(kar) - 32

            tampilkan(karakter_bernilai_ASCII_berupa_nilai ASCII)

        SEBALIKNYA

            tampilkan(kar)

        AKHIR-JIKA

    AKHIR-JIKA

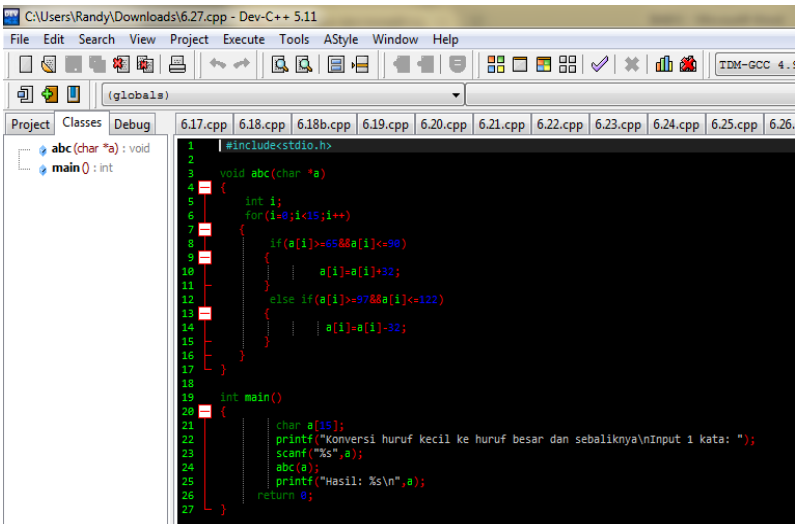
AKHIR-UNTUK

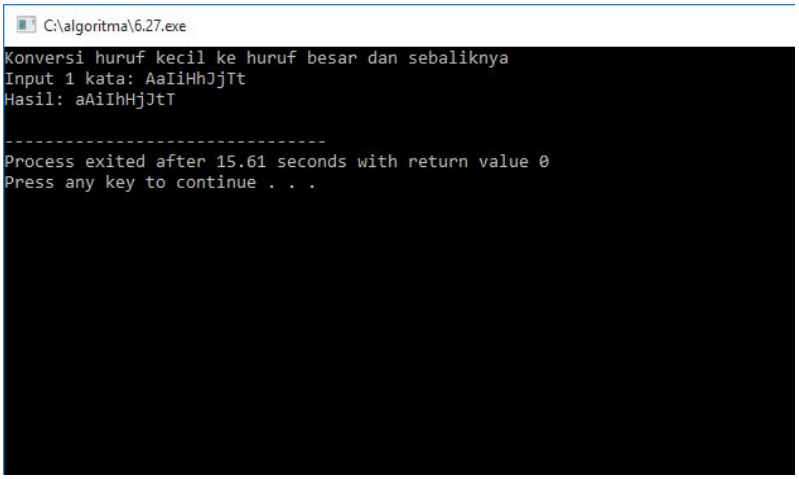
3. tampilkan(karakter_pindah_baris)

Kode Sumber :**ubahhrf.cpp**

```
#include <iostream.h>
#include <string.h>

int main ()
{
      int i;
      char straing [80];
      char kar ;

      cout << “masukkan sedangkan string : \n”;
      cin >> string ;

      for (i = 0; i < (int) stringlen (string) ; i++)
      {
            kar = string [i] ;
            if ((kar >= ‘A’) && (kar <= ‘Z’))
                  cout << (char) (kar+32) ;
            else
                  if ((kar >= ‘a’) && (kar <= ‘z’))
                        cout << (char) (kar-32) ;
                  else
                        cout << kar ;
      }
```

```
        cout << "\n";

        return 0;
}
```





	Akhir Kode Sumber

Contoh 6.28 Deret Fibonacci memiliki nilai suku pertama dan suku kedua berupa 1. Nilai suku – suku berikutnya ditentukan oleh dua suku sebelumnya. Contoh:

1 1235813 …

Tuliskan algoritma dan program untuk menampilkan deret Fibonacci sampai nilai suku terakhir tidak lebih dari 200.

Algoritma:

1. fn1 ← 1
2. fn2 ← 1
3. tampilkan(1)
4. tampilkan(1)
5. fn ← fn1 + fn2
6. ULANG SELAMA fn ≤ 200

        tampilkan(fn)

        fn2 = fn1

        fn1 = fn

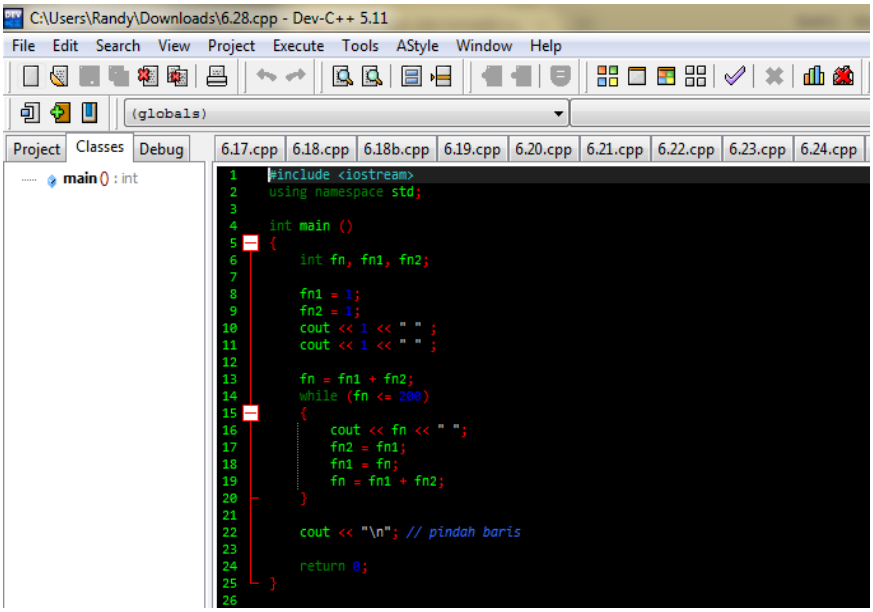
        fn ← fn1 + fn2

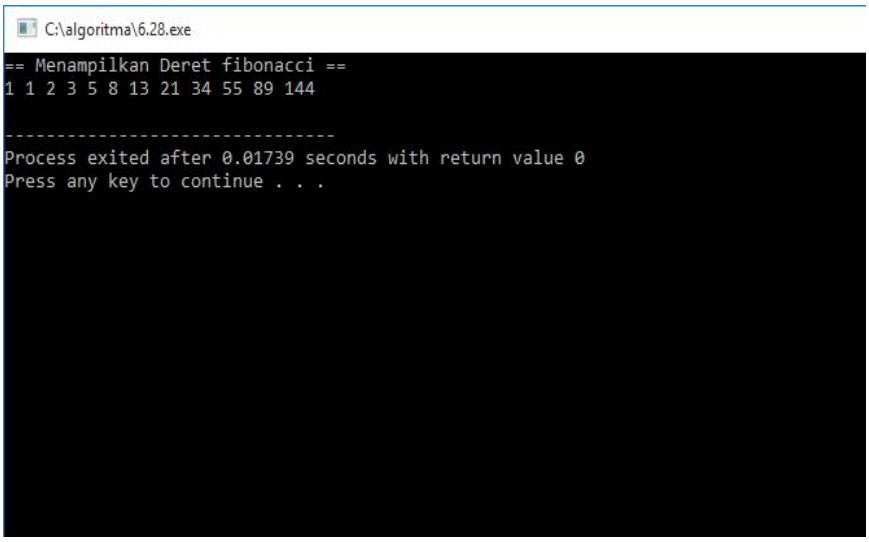
    AKHIR-ULANG

Kode Sumber :**fibonac.cpp**

```
#include <iostream.h>

int main ()
{
      int fn, fn1, fn2;

      fn1 = 1;
      fn2 = 1;
      cout << 1 << " " ;
      cout << 1 << " " ;

      fn = fn1 + fn2;
      while (fn <= 200)
      {
            cout << fn << " ";
            fn2 = fn1;
            fn1 = fn;
            fn = fn1 + fn2;
      }

      cout << "\n"; // pindah baris

      return 0;
```

}





	Akhir Kode Sumber

Contoh 6.29 Salah satu metode untukmenghitung depresiasi adalah metode jumlah – digit. Sebagai contoh, suatu computer seharga 150 juta hendak di depresiasi selama 5 tahun. Mula – mula digit tahun dijumlahkan seperti berikut:

JUM = 1 + 2 + 3 + 4 + 5 = 15

Berdasarkan hal itu, maka:

- depresiasi tahun pertama = 5/15 x 150 juta = 50 juta

- depresiasi tahun kedua = 4/15 x 150 juta = 40 juta
- depresiasi tahun ketiga = 3/15 x 150 juta = 30 juta
- depresiasi tahun keempat = 2/15 x 150 juta = 20 juta
- depresiasi tahun pertama = 1/15 x 150 juta = 10 juta

Berdasarkan uraian tersebut, tulis algoritma dan program yang meminta BIAYA dan jumlah TAHUN dan kemudian menampilkan depresiasi per tahun semacam berikut:

1. 50000000
2. 40000000
3. 30000000
4. 20000000
5. 10000000

Untuk menyederhanakan algoritma, **JUM** dapat dihitung dengan menggunakan rumus berikut:

JUM=1+2+3+…+n

= n (n + 1)/2

Algoritma:

1. masukkan(biaya, tahun)
2. jum ← n x (n + 1) / 2
3. UNTUK i ← 1 S/D tahun

   depresiasi ← (tahun + 1 – i) / jum x biaya

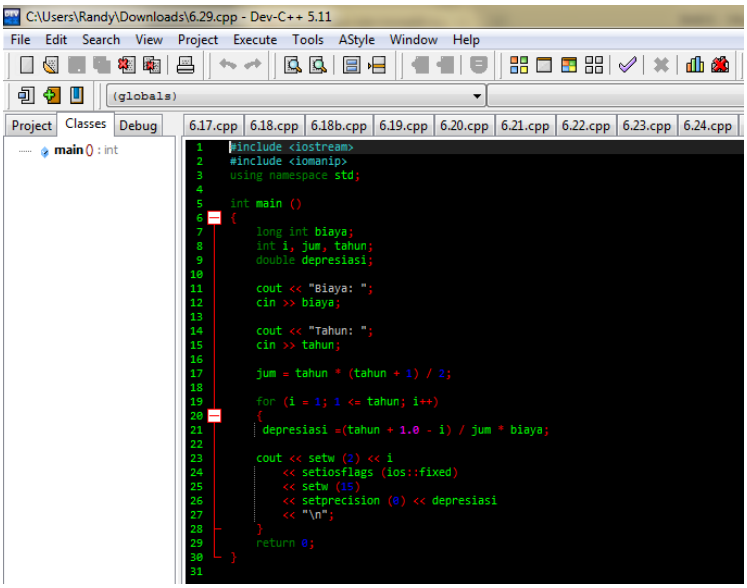
   tampilkan (i, depresiasi)

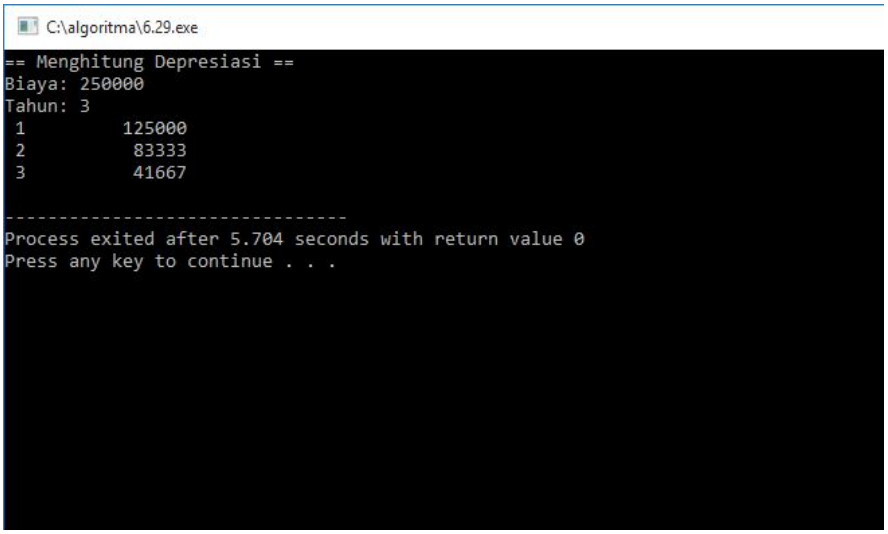
   AKHIR-UNTUK

Kode Sumber :**depres.cpp**

```
#include <iostream.h>
#include <iomanip.h>

int main ()
{
```

```
	long int biaya;
	int i, jum, tahun;
	double depresiasi;

	cout << “Biaya: “;
	cin >> biaya;

	cout << “Tahun: “;
	cin >> tahun;

	jum = tahun * (tahun + 1) / 2;

	for (i = 1; I <= tahun; i++)
	{
		Depresiasi = (tahun + 1.0 – i) / jum * biaya;

		Cout << setw (2) << i
			<< setiosflags (ios::fixed)
			<< setw (15)
			<< setprecision (0) << depresiasi
			<< “\n”;
	}

	return 0;
}
```





Akhir Kode Sumber

Contoh 6.30 Buatlah algoritma dan program untuk mengubah suatu tahun ke dalam bentuk angka romawi. Sebagai contoh, jika bilangan yang dimasukkan dari keyboard adalah 1987 maka hasilnya berupa MCMLXXXVII. Batasilah angka yang dimasukkan berkisar antara 1 sampai dengan 3999.

Algoritma:

Sebelum membicarakan algoritmanya, perlu diketahui bahwa angka romawi hanya mengenal symbol – symbol M, D, C, L, X, V, dan I dengan masing – masing symbol bernilai seperti tertera pada table berikut:

Tabel 6.2*Daftar simbol pada angka romawi*

Simbol	Nilai
M	1000
D	500
C	100
L	50
X	10
I	1

Semua angka romawi disusun atas kombinasi simbol – simbol tersebut. Tentu saja angka romawi hanya cocok untuk bilangan yang tidak terlalu besar, misalnya untuk menyatakan tahun. Hal lain yang perlu diperhatikan adalah cara mengonversi bilangan seperti 900 yang

dibentuk menjadi CM; bukan DCCCC. Begitu juga angka seperti 4 akan diterjemahkan menjadi IV dan bukan IIII.

Penuangan dalam bentuk algoritma:

1. masukkan(tahun)
2. JIKA tahun < 1 ATAU tahun > 3999 MAKA
        tampilkan(“Tahun harus dalam jangkauan 1 … 3999”)
    SEBALIKNYA
        ULANG SELAMA tahun >= 1000
            tampilkan(“M”)
            tahun ← tahun – 1000
        AKHIR -ULANG

    JIKA tahun >= 500 MAKA
        JIKA tahun >= 900 MAKA
            tampilkan (“CM”)
            tahun← tahun – 900
        SEBALIKNYA
            tampilan (“D”)
            tahun ← tahun – 500
    AKHIR -JIKA

    ULANG SELAMA tahun >= 100
        JIKA tahun >= 400 MAKA
            tampilkan (“CD”)
            tahun ← tahun – 400
        SEBALIKNYA
            tampilkan (“C”)
            tahun ← tahun – 100
    AKHIR -ULANG

    JIKA tahun >= 50 MAKA

```
        JIKA tahun >= 50 MAKA
                tampilkan (“XC”)
                tahun ← tahun – 90
        SEBALIKNYA
                tampilkan (“L”)
                tahun ← tahun – 50
AKHIR-JIKA

ULANG SELAMA tahun >= 10
        JIKA tahun >= 40 MAKA
                tampilkan (“XL”)
                tahun ← tahun – 40
        SEBALIKNYA
                tampilkan (“X”)
                tahun ← tahun – 10
AKHIR-ULANG

JIKA tahun >= 5 MAKA
        JIKA tahun == 9 MAKA
                tampilkan (“IX”)
                tahun ← tahun – 4
        SEBALIKNYA
                tampilkan (“V”)
                tahun ← tahun – 5
AKHIR-JIKA

ULANG SELAMA tahun >= 1
        JIKA tahun >= 4 MAKA
                tampilkan (“IV”)
                tahun ← tahun – 4
        SEBALIKNYA
                tampilkan (“I”)
```

tahun ← tahun – 1

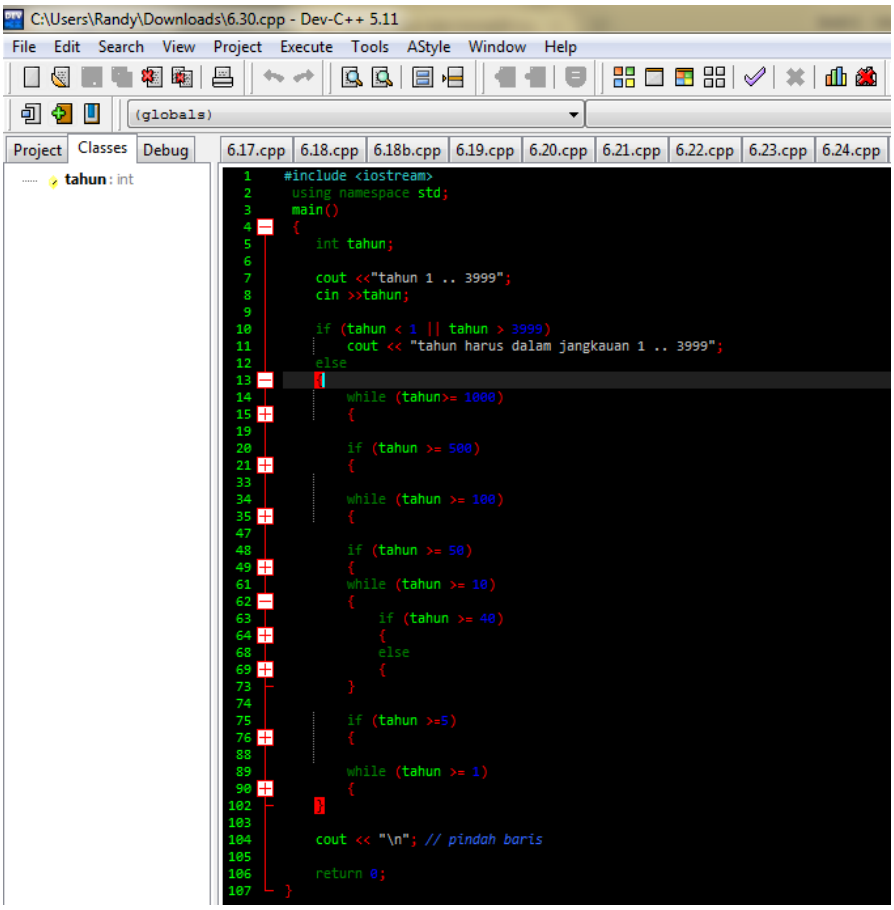
AKHIR-ULANG

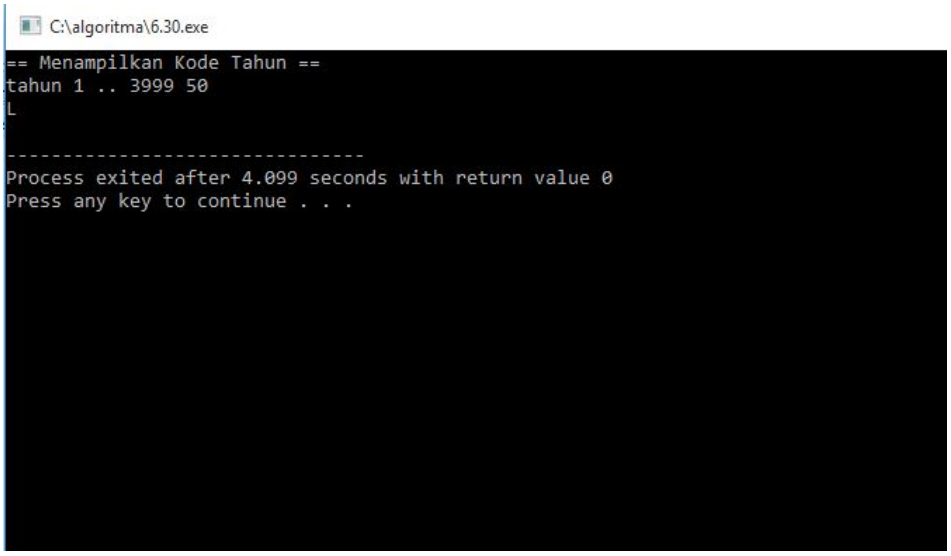
Kode Sumber :**romawi.cpp**

```
#include <iostream.h>

int main ()
{
      int tahun;

      cout << “Tahun (1 .. 3999):“;
      cin >> tahun;

      if (tahun < 1 || tahun > 3999)
            cout << “Tahun harus dalam jangkauan 1 .. 3999”;
      else
      {
            while (tahun >= 1000)
            {
                  cout << “M”;
                  tahun = tahun – 1000;
            }

            if (tahun >= 500)
            {
                  if (tahun >= 900)
                  {
                        cout << “CM”;
                        tahun = tahun – 900;
                  }
                  else
                  {
                        cout << “D”;
                        tahun = tahun – 500;
                  }
            }

            while (tahun >= 100)
            {
                  if (tahun >= 400)
                  {
                        cout << “CD”;
                        tahun = tahun – 400;
                  }
                  else
                  {
```

```
            cout << “C”;
            tahun = tahun – 100;
      }
}

if (tahun >= 50)
{
      if (tahun >= 90)
      {
            cout << “XC”;
            tahun = tahun – 90;
      }
      else
      {
            cout << “L”;
            tahun = tahun – 50;
      }
}
while (tahun >= 10)
{
      if (tahun >= 40)
      {
            cout << “XL”;
            tahun = tahun – 40;
      }
      else
      {
            cout << “x”;
            tahun = tahun – 10;
      }
}

if (tahun >= 5)
{
      if (tahun == 9)
      {
            cout << “IX”;
            tahun = tahun – 9;
      }
      else
      {
            cout << “v”;
            tahun = tahun – 5;
      }
}

while (tahun >= 1)
{
      if (tahun == 4)
```

```
                {
                        cout << “IV”;
                        tahun = tahun – 4;
                }
                else
                {
                        cout << “I”;
                        tahun = tahun – 1;
                }
        }
}

        cout << “\n”; // pindah baris

        return 0;
}
```





	Akhir Kode Sumber

BAB 7

Larik

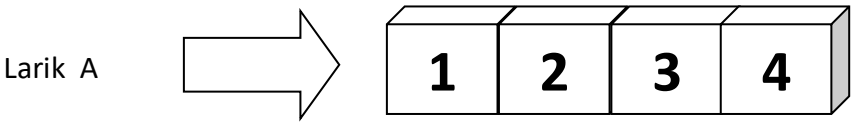
7.1 Pengertian Larik

Larik (*array*) menyatakan kumpulan data. Pada beberapa bahasa pemrograman, data yang terkandung dalam suatu larik harus bertipe sama. Namun dalam bahasa pemrograman tertentu, kumpulan data semacam itu bias melibatkan tipe yang berbeda – beda. Mengingat buku ini membahas bahasa C dan C++ dan kedua bahasa tersebut hanya mendukung larik dengan tipe data yang sama, maka jenis larik seperti itulah yang dibahas pada buku ini.

Di dalam algoritma, larik dinyatakan dengan awalan huruf capital dan notasi [] dipakai untuk menyatakan data dalam larik. Contoh:

A ← [1, 2, 3, 1]

menyatakan larik A yang berisi data 1, 2, 3 dan 1. Larik seperti itu dapat dinyatakan dalam bentuk gambar seperti terlihat Gambar 7.1.

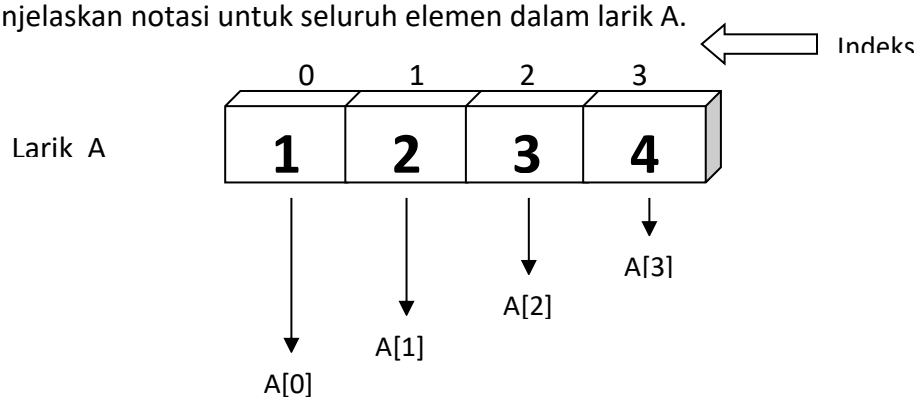


Gambar 7.1 *Gambaran larik*

Pada contoh diatas, larik A memiliki empat buah elemen.Untuk menyatakan sebuah elemen dalam larik, anda bisa menggunakan notasi berikut:

A[indeks]

Dalam hal ini indeks digunakan untuk menyatakan posisi elemen. Posisi pertama diberi kode 0, posisi kedua diberi kode 1, dan seterusnya. Gambar 7.2 menjelaskan notasi untuk seluruh elemen dalam larik A.



Gambar 7.2 *Notasi untuk mengakses sebuah elemen dalam larik*

Contoh berikut menunjukkan cara menampilkan elemen pada larik A dengan indeks sama dengan 2:

tampilkan(A[2])

Adapun

A[2] ←0

menyatakan pengisian nilai 0 ke elemen berindeks 2 pada larik A.

7.2 Larik padaC++

Pada bahasa C++ larik dideklarasikan dengan bentuk sebagai berikut:

tipe_data nama_larik[jumlah_elemen]

Dalam hal ini *jumlah_elemen harus* berupa konstanta.

Beberapa contoh dapat dilihat table 7.1.

Tabel 7.1*Contoh tabel pendeklarasian larik*

Deklarasi	Keterangan
int cacah [4];	Larik cacah mempunyai 4 buah elemen bertipe **int** (ilangan bulat)
Char vocal [5];	Larik vocal mempunyai 5 buah elemen bertipe **char** (karakter)
Char kota [6] [20];	Larik kota mempunyai 6 buah elemen bertipe string dengan panjang maksimal sebesar 19 karakter

Pengaksesan elemen larik dilakukan dengan menggunakan notasi

nama_array [*indeks*]

Dalam hal ini indeks dimulai dari nol.

Contoh:

```
Cacah [0] = 1;
```

merupakan pertanyaan untuk mengisikan nilai 1 ke elemen pertama pada larik `Cacah`.

Pada C dan C++, suatu larik bisa langsung diisi dengan suatu nilai ketika larik tersebut dideklarasikan. Contoh:

```
int jum_hari[12] =
      {31, 28, 31, 30, 31, 30,
      31, 31, 30, 31, 30, 31 };
```

Pada contoh di atas,

- jum_hari[0]bernilai 31
- jum_hari[1]bernilai 28
- jum_hari[3] bernilai 31
- jum_hari[4] bernilai 30
- dan seterusnya

Perlu diketahui, larik yang telah dibahas di depan adalah contoh larik berdimensi satu.

Catatan 	Pada C dan C++ terdapat larik dengan dimensi lebih dari satu. Larik seperti ini akan dibahas belakangan.

7.3 Contoh Berbagai Operasi dengan Larik

Berbagai persoalan yang terkait dengan larik dibahas pada subbab ini.

Contoh 7.1 Buatlah algoritma dan program yang mula – mula menyiapkan data huruf vocal ke dalam larik dan kemudian isi larik.

Algoritma:

1. V ← [“A”, “E”, “I”, “O”, “U”]
2. UNTUK i ← 0 S/D 4

       tampilkan (V[i])

   AKHIR-UNTUK

Program:

Implementasi dalam program C++ adalah seperti berikut:

 Kode Sumber :**vokal.cpp**

```
#include <iostream.h>

int main ()
{
      char kar [5];
      int i;

      kar [0] = ‘A’;
      kar [1] = ‘E’;
      kar [2] = ‘I’;
      kar [3] = ‘O’;
      kar [4] = ‘U’;

      for (i = 0; i < 5; i++)
            cout << kar [i] << “\n”;
      return 0;
}
```

Akhir Kode Sumber

Pada contoh di atas, jumlah elemen larikkar tidak perlu disebutkan.

Adapun implementasi dalan C++ adalah seperti berikut:

Kode Sumber :**vokal2.cpp**

```
#include <iostream.h>

int main()
{
      char kar[] = {'A', 'E', 'I', 'O', 'U'};

      for (int i = 0; i < 5; i++)
      cout << kar[i] << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.2 Buatlah algoritma dan program untuk menyiapkan data berikut ke dalam larik:

10 4 2 5 3 8 9 2 9 5

Kemudian carilah bilangan yang terbesar.

Algoritma:

1. Data ← [10, 4, 2, 3, 8, 9, 2, 9, 5]

2. terbesar ← Data [0]
3. UNTUK i ← 1 S/D 9

    JIKA Data [i] > terbesar MAKA

        terbesar ← Data [i]

    AKHIR- JIKA

    AKHIR-UNTUK
4. tampilkan(terbesar)

Mula – mula variable `terbesar` diisi dengan elemen pertama larik `Data.` Selanjutnya semua elemen kedua hingga yang terakhir dibandingkan dengan isi variable`terbesar.` Kalau nilai elemen tersebut lebih besar daripada variabel `terbesar` maka isinya diberikan ke variabel `terbesar.`

Program:

Penuangan dalam program C++.

Kode Sumber :**maksi.cpp**

```
#include <iostream.h>
int main ()
{
      int data [] = {10, 4, 2, 5, 3, 8, 9, 2, 9, 5};
      int i;
      int terbesar;
      terbesar = data [0];
      for (i = 1; i < 10; i++)
            if (data [i] > terbesar)
                  terbesar = data [i];

      cout << “Terbesar = “ << terbesar << “\n”;
      return 0;
}
```

Akhir Kode Sumber

Contoh 7.3 Buatlah algoritma dan program untuk membaca data secara berulang dari keyboard dan meletakkannya ke dalam suatu larik. Jumlah maksimal yang dapat

dimasukkan ke dalam larik adalah 10 buah. Setelah itu tampilkan seluruh data yang dimasukkan dari keyboard tadi.

Algoritma:

1. jumdata ← 0
2. UNTUK i ← 0 S/D 9

   masukkan (Data [i])

   tampilkan (“Memasukkan lagi (Y/I) ? “)

   masukkan(jawaban)

   JIKA jawaban = “I” atau “t” MAKA

   jumdata ← i + 1

   keluar dari pengulangan

   AKHIR-JIKA

   AKHIR-UNTUK
3. UNTUK i ← 0 S/D jumdata – 1

   tampilkan(Data [i])

   AKHIR-UNTUK

Langkah kedua di atas digunakan untuk menangani pemasukan data, sedangkan langkah ketiga dipakai untuk mengenai penampilkan data dalam lalik.

Kode Sumber :**datalrk.cpp**

```
#include <iostream.h>

int main ()
{
      double data [10];
      int i, jumdata;
      char jawaban;

      jumdata = 0;
      for (i = 0; i < 10; i++)
      {
            cout << “Masukkan sembarang bilangan : “;
            cin >> data [i];

            cout << “Memasukkan lagi (Y/T) ? “;
            cin >> jawaban;

            if (jawaban == ‘T’ || jawaban == ‘t’)
```

```
            {
                  Jumdata = i + 1;
                  Break;
            }
      }

      For (i = 0; i < jumdata; i++)
            Cout << data [i] << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.4 Berdasarkan larik pada Soal 7.3, tuliskan algoritma untuk memperoleh nilai rata – ratanya.

Algoritma:

1. jumdata ← 0
2. UNTUK i ← 0 S/D 9

   masukkan (Data [i])

   tampilkan ("Memasukkan lagi (Y/T) ? ")

   masukkan (jawaban)

   JIKA jawaban = "T" atau "t" MAKA

   jumdata ← i + 1

   keluar dari pengulangan

   AKHIR – JIKA

   AKHIR – UNTUK
3. jumtotal = 0
4. UNTUK i ← 0 S/D jumdata – 1

   jumtotal ← jumtotal + Data [i]

   AKHIR-UNTUK
5. rata_rata ← jumtotal / jumdata
6. tampilkan(rata_rata)

langkah kedua digunakan untuk memasukkan data dengan maksimal sebanyak 10 buah. Langkah keempat digunakan untuk untuk menjumlahkan isi semua elemen larik. Langkah kelima dipakai untuk menghitung nilai rata – rata.

Program:

Penuangan dalam program C++

Kode Sumber :**ratalrk.cpp**

```
#include <iostream.h>

int main ()
{
      double data [10];
      int i, jumdata;
      char jawaban;
      double jumtotal, rata_rata;

      jumdata = 0;
      for (i = 0; i <10; i++)
      {
            cout << “Masukkan sembarang bilangan: “;
            cin >> data [i];

            cout << “Memasukkan lagi (Y/T) ? “;
            cin >> jawaban;

            if (jawaban == ‘T’ || jawaban == ‘t’)
            {
                  jumdata = i + 1;
                  break;
            }
      }

      // Hitung jumlah dari keseluruhan elemen larik jumtotal =
      0;
      for (i = 0; i < jumdata; i++)
            jumtotal = jumtotal + data [i];

      rata_rata = jumtotal / jumdata;
      cout << “Rata_rata” << rata_rata << “\n”;

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.5 Buatlah algoritma dan program untuk menampilkan nama bulan berdasarkan kode bulan. Misalnya, 1 berarti “Januari”, 2 berarti “Februari”, dan sebagainya. Gunakan larik untuk menyimpan nama bulan.

Algoritma:

1. Nama_bulan = [“”, “Januari”, “Februari”, “Maret”, “April”, “Mei”,“Juni”, “Juli”, “Agustus”, “September”, “Oktober”,“November”, “Desember”]
2. masukkan(kode_bulan)
3. JIKA kode_bulan ≥ 1 DAN kode_bulan ≤ 12 MAKA
   tampilkan (Nama_bulan [kode_bulan – 1]
   SEBALIKNYA
   tampilkan (“Kode bulan harus berada antara 1 s/d 12”)
   AKHIR-JIKA

Perlu diketahui, elemen pertama dalam larik `Nama_bulan` segaja diisi dengan string kosong dengan maksud agar `Nama_bulan` dapat diakses dengan bentuk:

```
Nama_bulan [kode-bulan]
```

dengan `kode-bulan` berkisar antara 1 sampai dengan 12.

Program:

Penuangan dalam program C++:

Kode Sumber :**namabln.cpp**

```
#include <iostream.h>

int main ()
{
      char nama_bulan [] [10] = { “”, “Januari”, “Februari”,
                                  “Maret”, “April”, “Mei”,
                                  “Juni”, “Juli”, “Angustus”,
                                  “September”, “Oktober”,
                                  “November”, “Desember”};
      int kode_bulan;

      cout << “Kode bulan (1 .. 12) : “;
      cin >> kode_bulan;

      if (kode_bulan >= 1 && kode_bulan <= 12)
            cout << “Bulan : “ << nama_bulan[kode_bulan] <<
“\n”;
      else
            cout << “Kode bulan harus berada antara 1 s/d
12\n”;
```

```
    return 0;
}
```

Akhir Kode Sumber

Contoh 7.6 Buatlah algoritma dan program untuk menampilkan permutasi dari tiga orang bernama “Budi”, “Andri”, dan “Permana”.

Algoritma:

Algoritma untuk mendapatkan pasangan dari tiga orang tersebut dengan susunan boleh dibolak – balik adalah seperti berikut:

1. Nama ← [“Budi”, “Andri”, “Purnama”]
2. UNTUK i ← 0 S/D 2
    UNTUK j ← 0 S/D 2
        JIKA i ≠ j MAKA
            UNTUK k = 0 S/D 2
                JIKA i ≠ k DAN j ≠ k MAKA
                    tampilkan (Nama[i], Nama[j], Nama[k])
                AKHIR-JIKA
            AKHIR-UNTUK
        AKHIR-JIKA
    AKHIR-UNTUK
AKHIR-UNTUK

Program:

Implementasi dalam C++:

Kode Sumber :**permutasi.cpp**

```
#include <iostream.h>

int main ()
{
      char nama [] [10] = { “Budi”, “Andri”, “Permana”};
      int i, j, k;

      for (i = 0; i <= 2; i++)
            for (j = 0; j <= 2; j++)
                  if (i ! = j)
                        for (k = 0; k <= 2; k++)
                              if (i ! = k && j ! = k)
```

```
                                        cout << nama [i] << "-"
                                             << nama [j] << "-"
                                             << nama [k] << "\n";
        return 0;
}
```

Akhir Kode Sumber

Contoh 7.7 Terdapat dua buah larik seperti berikut:

[1, 2, 3, 5, 6, 8]

[4, 6, 1, 5, 2, 8]

Tuliskan algoritma dan program untuk menjumlahkan elemen pada kedua larik yang memiliki posisi sama dan menaruhnya ke larik ketiga. Setelah itu tampilkan seluruh elemen pada ketiga tersebut.

Algoritma:

1. A ← [1, 2, 3, 5, 6, 8]
2. B ← [4, 6, 1, 5, 2, 8]
3. UNTUK i ← 0 S/D 5

   C [i] ← A [i] + B [i]

   AKHIR-UNTUK
4. UNTUK i ← 0 S/D 5

   tampilkan (C [i])

   AKHIR-UNTUK

Program:

Penuangan dalam program C++:

Kode Sumber :**jumvekt.cpp**

```
#include <iostream.h>

int main ()
{
      int a [6] = {1, 2, 3, 5, 6, 8} ;
      int b [6] = {4, 6, 1, 5, 2, 8} ;
      int c [6] ;

      int i;
```

```
      for (i = 0; i <= 5; i++)
            c [i] = a [i] + b [i];

      for (i = 0; i <= 5; i++)
            cout << c [i] << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.8 Terdapat dua buah larik seperti berikut:

[1, 2, 3, 5, 6, 8]

[4, 6, 1, 5, 2, 8]

Tulislah algoritma untuk memperoleh larik yang berisi interaksi dari kedua larik tersebut. Interseksi berupa elem – elemen yang terdapat pada kedua larik tersebut.

Algoritma:

1. A ← [1, 2, 3, 5, 6, 8]
2. B ← [4, 6, 1, 5, 2, 8]
3. jum_interseksi ← 0
4. UNTUK i ← 0 S/D 5
   dicari ← A [i]
   UNTUK j = 0 S/D 5
   JIKA dicari = B [j] MAKA
   C[jum_interseksi] ← dicari
   jum_interseksi ← jum_interseksi + 1
   keluar dari pengulangan UNTUK j
   AKHIR-JIKA
   AKHIR-UNTUK
   AKHIR-UNTUK
5. JIKA jum_interseksi = 0 MAKA
   tampilkan ("Hasil interseksi tidak ada")
   SEBALIKNYA
   tampilkan ("Interseksi:")
   UNTUK i ← 0 S/D jum_interseksi – 1
   tampilkan (C [i])

AKHIR-UNTUK

AKHIR – JIKA

Program:

Program C++ yang didasarkan pada algoritma di depan:

Kode Sumber :**intersek.cpp**

```
#include <iostream.h>

int main ()
{
      int a [6] = {1, 2, 3, 5, 6, 8} ;
      int b [6] = {4, 6, 1, 5, 2, 8} ;
      int c [6] ;

      int i, j;
      int jum_interseksi;
      int dicari;

      jum_interseksi = 0;

      for (i = 0; i <= 5; i++)
      {
            dicari = a [i];
            for (j = 0; j <= 5; j++)
                  if (dicari == b [j])
                  {
                        c[jum_interseksi] = dicari;
                        jum_interseksi++;
                        break;
                  }
      }
      if (jum_interseksi == 0)
            cout << “Hasil interseksi tidak ada”;
      else
      {
            cout << “Interseksi: \n”;
            for (i = 0; i < jum_interseksi; i++)
                  cout << c[i] << “\n”;
      }

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.9 Terdapat larik seperti berikut:

[8, 9, 4, 7, 6, 1, 5, 3]

Tampilkan menjadi seperti berikut:

3 5167498

Tuangkan dalam bentuk algoritma dan program.

Algoritma:

1. Data ← [8, 9, 4, 7, 6, 1, 5, 3]
2. UNTUK i ← 7 S/D 0 LANGKAH – 1

   tampilkan (Data [i])
3. AKHIR-UNTUK

Program:

Program C++ yang didasarkan pada algoritma di depan:

Kode Sumber :**baliklrk.cpp**

```
#include <iostream.h>

int main ()
{
      int data [] = {8, 9, 4, 7, 6, 1, 5, 3};

      for (int i = 7; i > 0; i--)
      cout << data [i] << " ";
      cout << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.10 Terdapat larik seperti berikut:

[8, 9, 4, 7, 6, 1, 5, 3, 4, 10, 4, 16, 3]

Tuliskan algoritma dan program yang melakukan pertukaran 2 nilai yang berdekatan.

Hasilnya akan menjadi seperti berikut:

[9, 8, 7, 4, 1, 6, 3, 5, 4, 10, 3, 16]

Algoritma:

1. Data ← [8, 9, 4, 7, 6, 1, 5, 3]
2. UNTUK i ← 0 S/D 7 LANGKAH 2

tmp ← Data [i]

Data [i] = Data [i + 1]

Data [i + 1] = tmp

AKHIR-UNTUK

3. UNTUK i ← 0 S/D 7

tampilkan (C[i])

AKHIR-UNTUK

Penukaran dua elemen larik yang berdekatan dilakukan memlalui:

tmp ← Data [i]

Data [i] = Data [i + 1]

Data [i + 1] = tmp

Program:

Program C++ yang didasarkan pada algoritma di depan:

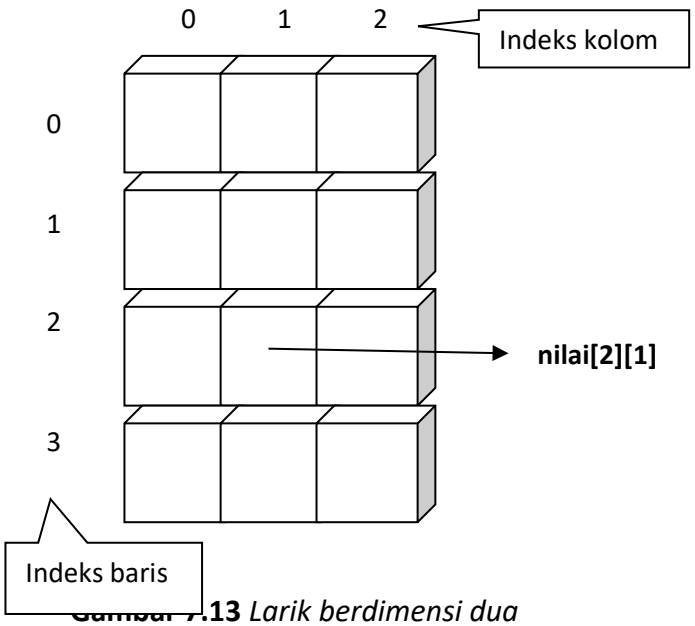
Kode Sumber C++ :**tukarlrk.cpp**

```
#include <iostream.h>

int main ()
{
      int data [] = {8, 9, 4, 7, 6, 1, 5, 3, 10, 4, 16, 3};
      int tmp;

      int i;
      for (i = 0; i < 11; i += 2)
      {
            tmp = data [ i ];
            data [i] = data [ i + 1];
            data [i + 1] = tmp;
      }

      //Tampilkan isi larik
      for (i = 0; i < 12; i++)
      cout << data [i] << " ";

      cout << "\n";

      return 0;
}
```

Akhir Kode Sumber

7.4 Larik Berdimensi Dua

Gambaran larik berdimensi dua diperlihatkan pada Gambar 7.13. Pada larik seperti itu setiap elemen di akses melalui dua buah indeks, yaitu indeks baris dan indeks kolom.



Gambar 7.13 *Larik berdimensi dua*

Pada bahasa C dan C++, larik pada contoh Gambar 7.13 dideklarasikan dengan cara sebagai berikut:

```
int nilai [4][3];
```

Untuk mengakses sebuah elemen pada larik tersebut diperlukan notasi seperti berikut:

Nilai [*indek_baris*][*indek_kolom*]

Hal ini berlaku pada algoritma maupun program C dan C++. Dalam hal ini masing – masing indeks dimulai dari nol.

Contoh 7.11 Buatlah algoritma dan program yang menyimpan pasangan data Negara dan ibukota seperti berikut:

Indonesia	Jakarta
Pilipina	Manila
Austria	Wina
India	New Delhi
Iran	Taheran

Kemudian tampilkan nama Negara dan ibukotanya, khusus untuk Negara yang berlawanan dengan I.

Algoritma:

1. Negara [0] [0] ← “Indonesia”
2. Negara [0] [1] ← “Jakarta”
3. Negara [1] [0] ← “ Pilipina”
4. Negara [1] [1] ← “Manila”
5. Negara [2] [0] ← “Austria”
6. Negara [2] [1] ← “Wina”
7. Negara [3] [0] ← “India”
8. Negara [3] [1] ← “New Delhi”
9. Negara [4] [0] ← “Iran”
10. Negara [4] [1] ← “Taheran”
11. UNTUK baris ← 0 S/D 4

        Jika Negara [baris] [0] [0] = “I” MAKA

            tampilkan (Negara [baris] [0], Negara [baris] [1])

        AKHIR-JIKA

    AKHIR-UNTUK

Program:

Adapun penuangan dalam program C++ adalah:

Kode Sumber C++ :**negara.cpp**

```
#include <iostream.h>
#include <string.h>

int main ()
{
      char Negara [5] [2] [15];
      int baris;

      strcpy (Negara[0] [0], “Indonesia”);
      strcpy (Negara[0] [1], “Jakarta”);


      strcpy (Negara[1] [0], “Pilipina”);
      strcpy (Negara[1] [1], “Manila”);
      strcpy (Negara[2] [0], “Austria”);
      strcpy (Negara[2] [1], “Wina”);
      strcpy (Negara[3] [0], “India”);
```

```
      strcpy (Negara[3] [1], “New Delhi”);
      strcpy (Negara[4] [0], “Iran”);
      strcpy (Negara[4] [1], “Taheran”);

      for (baris = 0; baris < 5; baris++)
            if (Negara [baris] [0][0] == ‘I’)
                  cout << Negara [baris] [0] << “ – “
                        << Negara [baris] [1] << “\n”;

      return 0;
}
```

Akhir Kode Sumber

Alternatife lain untuk mengisikan data ke dalam larik Negara dapat ditangani sewaktu larik tersebut dideklarasikan. Penuangan dalam algoritmanya adalah seperti berikut:

1. Negara ← [[“Indonesia”, “Jakarta”],
   [“Pilipina”, “Manila”],
   [“Austria”, “New Delhi”],
   [“Iran”, “Taheran”]]
2. UNTUK baris ← 0 S/D 4
   JIKA Negara[baris] [0] [0] = “I” MAKA
   tampilkan (Negara[baris] [0], Negara [baris] [1])
   AKHIR-JIKA
   AKHIR-UNTUK

Penuangan dalam program C++:

Kode Sumber C++ :**negara2.cpp**

```
#include <iostream.h>
#include <string.h>

int main ()
{
      char Negara [5] [2] [15] = {{“Indonesia”, “Jakarta”},
                                  {“Pilipina”, “Manila”},
                                  {“Austria”, “Wina”},
                                  {“India”, “New Delhi”},
                                  {“Iran”, “Taheran”}};

      for (int baris = 0; baris < 5; baris++)
            if (Negara[baris] [0] [0] == ‘I’)
                  cout << Negara[baris] [0] << “ – “
```

```
                    << Negara[baris] [1] << "\n";
    return 0;
}
```

Akhir Kode Sumber

Contoh 7.12 Terdapat dua matriks seperti berikut:

$$A = \begin{bmatrix} 6 & 7 \\ 5 & 8 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 3 \\ 4 & -1 \end{bmatrix}$$

Tuliskan algoritma untuk menghitung matriks C yang merupakan perjumlahan dari matriks A dan B.

Algoritma:

1. A ← [[6, 7], [5, 8]]
2. B ← [[1, 3], [4, -1]]
3. UNTUK i = 0 S/D 1
   UNTUK j = 0 S/D 1
   C[i] [j] ← A [i] [j] + B [i] [j]
   AKHIR-UNTUK
   AKHIR-UNTUK
4. UNTUK i = 0 S/D 1
   UNTUK j = 0 S/D 1
   tampilkan(C[i][j]) // Tanpa pindah baris
   AKHIR – UNTUK
   tampilkan (karakter_pindah_baris)
   AKHIR – UNTUK

Program:

Program C++ - nya adalah sebagai berikut:

Kode Sumber C++ :**jummatr.cpp**

```
#include <iostream.h>
#include <string.h>
#include <iomanip.h>
```

```
int main ()
{
      int a [2] [2] = {{6, 7}, {5, 8}};
      int b [2] [2] = {{1, 3}, {4, -1}};
      int c [2] [2];
      int i, j;

      for (i = 0; i < 2; i++)
            for (j = 0; j << 2; j++)
                  c [i][j] = a [i][j] + b [i][j];

      for (i = 0; i < 2; i++)
      {
            for (j = 0; j < 2; j++)
                  cout << setw (4) << c [i][j];

            cout << "\n"; // pindah baris
      }

      return 0;
}
```

Akhir Kode Sumber

Contoh 7.13 Tuliskan algoritma untuk memasukkan data ke dalam matriks. Semua data dimasukkan dari keyboard. Mula – mula jumlah baris dan kolom dimasukkan dari keyboard, dan kemudian semua elemen dimasukkan dari keyboard. Setelah itu tampilkan semua elemen matriks dalam bentuk susunan matriks.

Selanjutnya teranlasikan ke dalam program C/C++.

Algoritma:

1. masukkan(jum_baris)
2. masukkan (jum_kolom)
3. UNTUK i ← 0 S/D jum_baris – 1

   UNTUK j ← 0 S/D jum_kolom – 1

   masukkan (A [i] [j])

   AKHIR-UNTUK

   AKHIR – UNTUK
4. UNTUK i ← 0 S/D jum_baris – 1

   UNTUK j ← 0 S/D jum_kolom – 1

   tampilkan (A [i], [j]) // Tanpa pindah baris

AKHIR-UNTUK

tampilkan (karakter_pindah_baris)

AKHIR-UNTUK

Program:

Program C++ - nya adalah sebagai berikut:

Kode Sumber C++ :**inputmat.cpp**

```
#include <iostream.h>
#include <iomanip.h>

int main ()
{
      int jum_baris, jum_kolom;
      int i, j;

      int data [10][10];

      cout << “jumlah baris (1..10): “;
      cin >> jum_baris;

      cout << “jumlah kolom (1..10): “;
      cin >> jum_kolom;
      //Pemasukan elemen matriks

      for (i = 0; i <= jum_baris – 1; i++)
            for (j = 0; <= jum_kolom – 1; j++)
            {
                  cout << “Nilai baris” << i + 1
                        << “, kolom “ << j + 1 << “: “;
                  cin >> data [i][j];
            }

      // Menampilkan matriks

      cout <<”Data matriks: \n”;
      for (i = 0; i <= jum_baris – 1; i++)
      {
            for (j = 0; j <= jum_kolom – 1; j++)
                  cout << setw (8) << data [i][j];

            cout << “\n”; // pindah baris
      }

      return 0;
}
```

	Akhir Kode Sumber

7.5 Larik Bertipe Rekaman

Elemen suatu larik bisa saja melibatkan tipe rekaman (atau disebut **struct** pada C dan C++). Contoh data dengan tipe rekaman dinyatakan semacam berikut:

Simpul = REKAMAN

data 1

data 2

data 3

AKHIR – REKAMAN

Untuk menyatakan *data1* yang terdapat pada *simpul* digunakan notasi berupa simpul → data1.

Di dalam C dan C++, tipe rekaman seperti di atas dinyatakan dengan:

```
Struct rek_simpul {
      tipe data1;
      tipe data2;
      tipe data3;
};
Struct rek_simpul simpul;
```

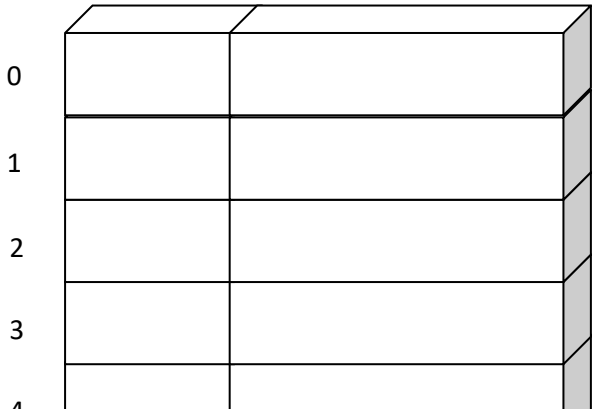
Untuk menyatakan field `data1` diperlukan notasi seperti berikut:

Simpul.data1

Adapun contoh menunjukkan bagaimana mendeklarasikan sebuah larik yang bertipe **struct**.

```
Struct rek_pegawai {
      int nip;
      char nama [30];
};
Struct rek_pegawai peg [5];
```

Pada contoh di atas,peg nip arik der nama rtipe **struct**.



Gambar 7.17*Larik peg dengan tipe elemen berupa struct.*

Contoh 7.14 Sepuluh siswa mengikuti ujian dengan nilai hasil berkisar antara 0 sampai dengan 100. Tulislah algoritma untuk membaca seluruh data tersebut (yang mencakup nama dan nilai ujian) dan kemudian:

Hitunglah nilai rata – ratanya.

Tampilkan siswa yang gagal (yaitu yang nilainya kurang dari 60).

Setelah itu translasikan ke dalam program C/C++.

Algoritma:

Solusi untuk permasalahan ini dapat dipecahkan dengan menggunakan larik yang bertipe rekaman. Dalam hal ini rekaman mengandung data `nama` dan `nilai`. Penuangan dalam bentuk algoritma:

1. pencacah ← 0
2. //Membaca data dan memasukkan nama dan nilai ke larik
   ULANG SELAMA pencacah < 10
   masukkan(Daftar[pencacah] → nama)
   ULANG SELAMA BENAR
   masukkan (Daftar[pencacah] → nilai)
   JIKA Daftar[pencacah] → nilai ≥ 0 DAN ≤ 100 MAKA
   keluar dari pengulangan ULANG SELAMA BENAR
   SEBALIKNYA
   tampilkan (“Nilai harus terletak antara 0 sampai dengan 100”)
   AKHIR – JIKA
   AKHIR – ULANG
   Pencacah ← pencacah + 1
   AKHIR ULANG
3. //Awal penghitungan nilai rata-rata

jum ← 0

4. pencacah←0
5. ULANG SELAMA pencacah < 10

   jum ← jum + Daftar[pencacah] → nilai

   pencacah ← pencacah + 1

   AKHIR-ULANG
6. rata_rata ← jum / 10.0
7. tampilkan(rata_rata)
8. //Menampilkan siswa yang gagal

   tampilkan (“Siswa yang gagal”)
9. pencacah ← 0
10. ada_gagal ← 0
11. ULANG SELAMA pencacah ≤ 10

    JIKA Daftar [pencacah] → nama < 60 MAKA

    ada_gagal ← 1

    tampilkan(daftar[pencacah] → nama)

    AKHIR-JIKA

    pencacah← pencacah + 1

    AKHIR-ULANG
12. JIKA ada_gagal ≠ 0 MAKA

    tampilkan (“Tak ada”)

    AKHIR – JIKA

Pada algoritma di atas, langkah kedua belas dimaksudkan untuk memberikan keterangan sekiranya tak ada siswa yang gagal.

Program:

Penuangan dalam C++ adalah seperti berikut:

Kode Sumber C++ :**dafsiswa.cpp**

```
#include <iostream.h>
```

```
int main ()
{
      Struct siswa {
            char nama [25];
            double nilai;
      };

      struct siswa daftar [10];
      int pencacah;
      double jum;
      int ada_gagal;

      pencacah = 0;
      while (pencacah < 10)
      {
            cout << “Nama siswa: “;
            cin >> daftar [pencacah].nama;

            while (1)
            {
                  cout << “Nilai: “;
                  cin >> daftar [pencacah].nilai;

                  if (daftar[pencacah].nilai >= 0 &&
                        daftar[pencacah].nilai <= 100)
                        break;
                  else
                        cout << “Nilai harus terletak antara 0 “
                              << “sampai dengan 100\n”;
            }

            Pencacah++;
      }

      //Jumlahkan semua nilai

      jum = 0;
      pencacah = 0;
      while (pencacah < 10)
      {
            Jum= jum + daftar[pencacah].nilai;
            Pencacah++;
      }

      //Menampilkan siswa yang gagal
      cout << “siswa yang gagal\n”;
      ada_gagal = 0;
      pencacah = 0;
      while (pencacah < 10)
```

```
    {
        if (daftar[pencacah].nilai < 60)
        {
            ada_gagal = 1;
            cout << daftar[pencacah].nama << "\n";
        }

        Pencacah++;
    }

    if (ada_gagal == 0)
        cout << "Tak ada\n";

    return 0;
}
```

Akhir Kode Sumber

Contoh 7.15 Modifikasilah persoalan zodiac pada Bab 5 dengan melibatkan larik. Dalam hal data tentang zodiac perlu disimpan pada larik.

Algoritma:

Solusi untuk permasalahan ini dapat dipecahkan dengan menggunakan larik bertipe struktur, dengan struktur meliputi data nama zodiak (`zodiak`), tanggal awal zodiac (`tg_awal`), bulan awal zodiak (`bl_awal`), tanggal akhir zodiak (`tg_akhir`),dan bulan akhir zodiak(`bl_awal`).Penuangan dalam program:

1. //Elemen Z terdiri nama, tg_awal, bl_awal, tg_akhir, bl_akhir,
   //Selanjutnya Z diisi seperti berikut:
2. Z ← [["Aries", 21, 3, 19, 4],
   ["Taurus", 20, 4, 20, 5],
   ["Gemini", 21, 5, 20, 6],
   ["Cancer", 21, 6, 22, 7],
   ["Leo", 23, 7, 22, 8],
   ["Virgo", 23, 8, 22, 9],
   ["Libra", 23, 9, 22, 10],
   ["Scorpio", 23, 10, 21, 11],
   ["Sagitarius", 22, 11, 21, 12],
   ["Capricorn", 22, 12, 19, 1],
   ["Aquarius", 20, 1, 18, 2],

[“Princes”, 19, 2, 20, 3]]

3. masukkan(tanggal, bulan).
4. UNTUK i ← 0 S/D 11

   JIKA (tanggal ≥ Z [i] → tg_awal dan bulan = Z [i] → bl_awal)

   atau

   (tanggal ≤ Z [i] → tg_akhir dan bulan = bl_akhir) MAKA

   tampilkan (Z [i] → nama)

   keluar dari pengulangan

   AKHIR – JIKA

   AKHIR – UNTUK

Program:

Program dalam C++ -nya adalah seperti berikut:

Kode Sumber C++ :**larikzod.cpp**

```
#include <iostream.h>

int main ()
{
      Struct zodiac {
            char nama [12];
            int tg_awal;
            int bl_awal;
            int tg_akhir;
            int bl_akhir;
      };

      Struct zodiac z[] =
            {{“Aries”, 21, 3, 19, 4},
                  {“Taurus”, 20, 4, 20, 5},
                  {“Gemini”, 21, 5, 20, 6},
                  {“Cancer”, 21, 6, 22, 7},
                  {“Leo”, 23, 7, 22, 8},
                  {“Virgo”, 23, 7, 22, 8},
                  {“Libra”, 23, 9, 22, 10},
                  {“Scorpio”, 23, 10, 21, 11},
                  {“Sagitarius”, 22, 11, 21, 12},
                  {“Capricorn”, 22, 12, 19, 1},
                  {“Aquarius”, 20, 1, 18, 2},
                  {“Prisces”, 19, 2, 20, 3}};

      int tanggal, bulan;
```

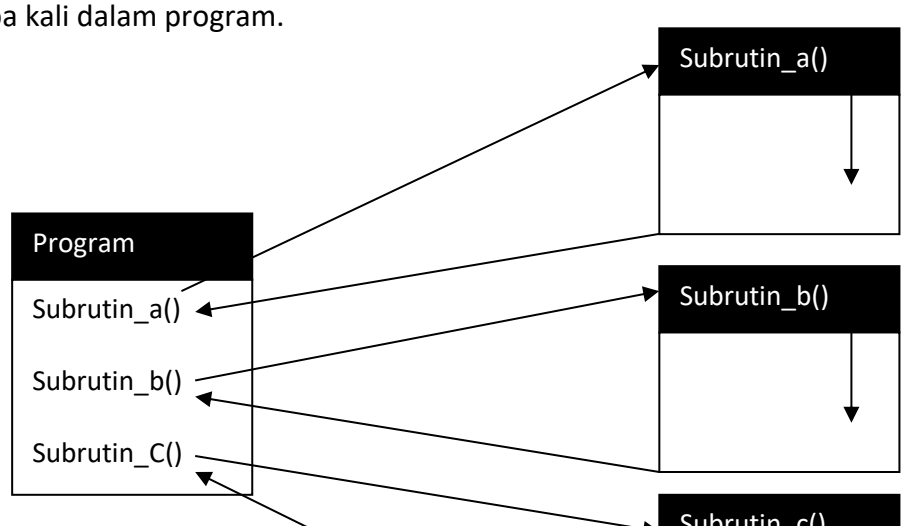
```
        int i;

        cout <<"masukkan tanggal kelahiran (1..31): ";
        cin >> tanggal;

        cout <<"masukkan bulan kelahiran (1..12): ";
        cin >> bulan;
        for (i = 0; i < 12; i++)
             if ((tanggal >= z[i].tg_awal &&
                  bulan == z[i].bl_awal) ||
                  (tanggal <= z[i].tg_akhir &&
                  Bulan == z[i].bl_akhir))
             {
                  cout << z[i].nama <<"\n";
                  break;
             }

        Return 0;
   }
```

Akhir Kode Sumber

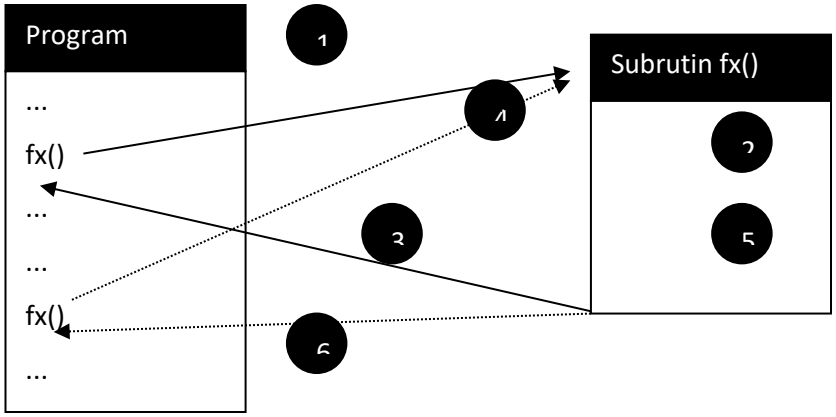
BAB 8
SUBRUTIN

8.1 Pengaturan Subrutin

Sebuah program yang besar biasanya disusun atas sejumlah bagian yang kecil yang dinamakan subrutin atau subprogram. Tujuan penggunaan subrutin adalah untuk memudahkan pengolahan/pengembangan program mengingat setiap subrutin memiliki kode yang relative sedikit (jika dibandingkan dengan kode program secara keseluruhan yang disusun tanpa melibatkan subrutin). Selain itu, subrutin juga dapat digunakan untuk mengurangi jumlah kode akibat sejumlah kode yang sama digunakan beberapa kali dalam program.



Gambar 8.1 *Konsep program yang tersusun atas sejumlah subrutin*



Gambar 8.2 *Gambaran penghemat kode pada pemanggilan subrutin yang sama pada beberapa tempat dalam program*

8.2 Penulisan Algoritma untuk Subrutin

Suatu subrutin ditulis dengan bentuk sebagai berikut:

SUBRUTIN *Nama Subrutin*(*daftar-parameter*)

Pertanyaan-1

. . .

Pertanyaan-2

AKHIR – SUBRUTIN

Dalam hal ini, bagian

SUBRUTIN *NamaSubrutin*(*daftar-parameter*)

disebut dengan *judul subrutin.*

Sebuah subrutin dapat memberikan nilai balik ataupun tidak. Nilai balik adalah nilai yang diberikan ke pemanggilnya. Nilai ini ditentukan melalui notasi seperti berikut:

NILAI-BALIK *nilai*

Contoh:

SUBRUTIN *hitung_keliling_kotak(panjang, lebar)*

keliling ← 2 x (panjang + lebar)

NILAI-BALIK keliling

AKHIR – SUBRUTIN

Pada contoh di atas, *hitung_keliling_kotak* adalah nama subrutin. Adapun *panjang* dan *lebar* disebut sebagai parameter. Parameter menyatakan bagian untuk berkomunikasi dengan pemanggil subrutin. Pada bagian pemanggil subrutin, bagian ini akan diisi dengan argument. Contoh:

hasil ← hitung_keliling_kotak(10, 5)

pada pemanggilan subrutin di atas, 10 dan 5 berkedudukan sebagai argument.

Catatan 	Terkadang argument disebut **parameter actual,** sedangkan parameter dalam pendefinisian subrutin disebut **parameter formal.**

8.3 Translasi Subrutin pada C++

Di dalamC++, subrutin biasa disebut dengan fungsi. Sebuah fungsi didefinisikan dengan bentuk sebagai berikut:

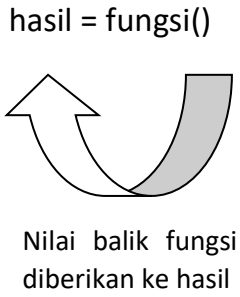
tipe_nilai_balik nama_fungsi(*tipe argumen1, tipe argumen2, …*)

{

Pernyataan_pernyataan;

return *nilai_balik;*

}



Gambar 8.4 *Nilai balik fungsi*

Contoh 8.1 Contoh berikut menunjukkan hasil translasi subrutin`hitung_keliling_kotak` yang telah dijelaskan di depan:

```
double hitung_keliling_kotak(double panjang, double lebar)
{
      double keliling;

      keliling = 2 * (panjang + lebar);

      return keliling;
}
```

Pada contoh di atas, **double** yang diletakkan di depan nama fungsi (`hitung_keliling_lingkaran`) menyatakan tipe nilai balik fungsi tersebut. `double panjang` dan `double lebar` menyatakan bahwa kedua parameter tersebut bertipe **double.** Dalam tubuh fungsi (yang berada dalan { }), pernyataan

```
double keliling;
```

digunakan untuk mendeklarasikan variable `keliling` dengan tipe **double**. Selanjutnya variable ini diisi dengan

```
2 * (panjang + lebar)
```

melalui pernyataan

```
keliling = 2 * (panjang + lebar);
```

Pernyataan

```
return keliling;
```

digunakan untuk memberikan nilai balik. Pernyataan ini adalah hasil translasi

NILAI-BALIK *nilai*

pada algoritma.

Contoh berikut menunjukkan cara pemanggil fungsi `hitung_keliling_kotak()` dalam programC++:

```
kelkotak = hitung_keliling_kotak ();
```

Contoh hasil eksekusi program:

```
C:\progc>kelkotak↵
Keliling = 117.000000
C:\progc>
```

Gambar 8.5 *Hasil program kelkotak*

Program:

Ekivalensi program di atas pada C++ adalah seperti berikut:

Kode Sumber C++ :**kelkotak.cpp**

```
#include <iostream.h>

double hitung_keliling_kotak(doble panjang, double lebar)
{
      double keliling;
      keliling = 2 * (panjang + lebar);

      return keliling;
```

```
}

int main ()
{
      double keliling, panjang, lebar;
      panjang = 56.5;
      lebar = 2;

      keliling = hitung_keliling_kotak(panjang, lebar);
      cout << “keliling = “ << keliling << “\n”;

      return 0;
}
```

Akhir Kode Sumber

8.4 Fungsi Tanpa Nilai Balik

Dalam beberapa bahasa pemrograman, subrutin dibedakan menjadi dua golongan, yakin fungsi dan prosedur.

- Fungsi adalah jenis subrutin yang menghasilkan nilai balik ketika subrutin dipanggil.
- Prosedur adalah jenis subrutin yang tidak menghasilkan nilai balik ketika subrutin diipanggil.

Dalam bahasa C++, kedua bentuk subrutin tersebut tetap dinamakan fungsi. Hanya saja prosedur disebut sebagai fungsi tanpa nilai balik. Fungsi tanpa nilai balik ditulisdengan bagian tipe fungsi berupa **void** (void berarti tanpa nilai balik).

Contoh 8.2 Contoh berikut menunjukkan fungsi yang tidak memiliki nilai balik.

```
Void infoPerusahaan ()
{
      cout<<“PT Primasoft Citra Data”;
}
```

Contoh di atas sekaligus menunjukkan contoh fungsi yang tidak melibatkan parameter.

Untuk memanggil fungsi tersebut, perlu penulisan seperti ini:

```
infoPerusahaan ();
```

Akhir Kode Sumber

Hasil eksekusi program:

```
C:\progc>void↵
PT Primasoft Citra Data
C:\progc>
```

Gambar 8.6 *Hasil program void*

8.5 Berbagai Contoh Subrutin

Beberapa contoh pembuatan subrutin:

Contoh 8.3 Buatlah subrutin untuk menghitung bilangan terkecil dari dua buah bilangan. Implementasikan dalam bbentuk algoritma maupun program.

Algoritma:

```
SUBRUTIN terkecil (x, y)
        JIKA x < y MAKA
                min ← x
        SEBALIKNYA
                min ← y
        AKHIR-JIKA

        NILAI-BALIK min
AKHIR-SUBRUTIN
```

Contoh hasil pengeksekusian program:

```
C:\progc>terkecil↵
Terkecil dari 35.600000 Dn 78.500000 : 35.600000
C:\progc>
```

Gambar 8.7 *Hasil program terkecil*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**terkecil.cpp**

```
#include <iostream.h>

double terkecil (double x, double y)
{
      double min;

      if (x < y)
            min = x;
      else
            min = y;

      return min;
}

int main ()
{
      double a, b, c;
      a = 35.6;
      b = 78.5;
      c = terkecil (a, b);

      cout << "Terkecil dari " << a << " dan " << b
            << " : " << c << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 8.4 Buatlah subrutin untuk menentukan suatu bilangan terkecil dalam suatu larik. Argumen dalam subrutin berupa larik dan jumlah datanya. Implementasi dalam bentuk algoritma maupun program.

Algoritma:

SUBRUTIN *terkecil* (X, jum)

min ← X [0]

UNTUK i = 1 S/D jum – 1

JIKA X[i] < min MAKA

```
                    min ← X [ i ]
                AKHIR-JIKA
            AKHIR-UNTUK

            NILAI-BALIK min
        AKHIR-SUBRUTIN
```

Pada algoritma di atas, X menyatakan larik dan `jumlah` menyatakan jumlah elemen dalam larik.

Contoh hasil pengeksekusian program:

```
C:\progc>minim↵
Terkecil = 2.300000
C:\progc>
```

Gambar 8.8 *Hasil program minim*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**minim.cpp**

```
#include <iostream.h>

double terkecil (double x[], int jum)
{
      int i;
      double min;
```

```
      min = x[0];
      for (i = 1; i < jum; i++)
            if (x[i] < min)
                  min = x[i];

      return min;
}

int min ()
{
      double y;
      double data[] = {578, 67.8, 2.3, 24, 123.5};

      y = terkecil (data, 5);

      cout << "Terkecil = " << y << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 8.5 Buatlah subrutin untuk menentukan suatu bilangan termasuk gelap atau ganjil.

Algoritma:

Subrutin berikut memberikan nilai balik berupa 1 kalau bilangan argument adalah bilangan ganjil atau nilai balik berupa 0 untuk keadaan sebaliknya.

```
SUBRUTIN ganjil(bil)
      JIKA sisa_pembagian(bil, 2) = 1 MAKA
            NILAI-BALIK 1
      SEBALIKNYA
            NILAI-BALIK 0
      AKHIR-JIKA
AKHIR-SUBRUTIN
```

Akhir Kode Sumber

Contoh hasil pengeksekusian program:

```
C:\progc>ganjil↵
1
1
0
C:\progc>
```

Gambar 8.9 *Hasil program ganjil*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**ganjil.cpp**

```
#include <iostream.h>

int ganjil(int bil)
{
      if (bil % 2 == 1)
            return 1;
      else
            return 0;
}

int main ()
{
      cout << ganjil (5) << "\n";
      cout << ganjil (7) << "\n";
      cout << ganjil (6) << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 8.6 Buatlah subrutin dengan nama `str2int` dengan argument berupa suatu string yang menyatakan bilangan bulat. Nilai baliknya berupa suatu angka yang menrupakan hasil konversi string tersebut ke dalam bentuk bilangan. Sekiranya ada karakter yang tidak menyiratkan bilangan dalam string argument, subrutin memberikan nilai balik berupa nol. Tanda plus dan negative boleh disertai dalam argument.

Algoritma:

SUBRUTIN *str2int*(*st*)

// Cek kalau string berupa string kosong

JIKA panjang (st) = 0 MAKA

```
        NILAI_BALIK = 0
AKHIR-JIKA

//Validasi karakter dalam st
ok ← 1
JIKA st[0] tidak berupa '+' atau '-' atau digit MAKA
        ok ← 0
SEBALIKNYA
        UNTUK i ← S/D panjang (st) – 1
                JIKA st [i] tidak berupa digit MAKA
                        ok ← 0
                        keluar dari pengulangan
                AKHIR-JIKA
        AKHIR-UNTUK
AKHIR – JIKA

JIKA ok ≠ 0 MAKA
        NILAI-BALIK
AKHIR – JIKA

//Konversi string ke angka
bilangan ← 0
pengali ← 1
UNTUK i = panjang(st) – 1 S/D 0 LANGKAH – 1
        COCOK st [i]
                DENGAN '0' MAKA
                        digit ← 0
                DENGAN '1' MAKA
                        digit ← 1
                DENGAN '2' MAKA
                        digit ← 2
                DENGAN '3' MAKA
```

```
                digit ← 4
            DENGAN '5' MAKA
                digit ← 5
            DENGAN '6' MAKA
                digit ← 6
            DENGAN '7' MAKA
                digit ← 7
            DENGAN '8' MAKA
                digit ← 8
            DENGAN '9' MAKA
                digit ← 9
            DENGAN '-' MAKA
                digit ← - 1
            DENGAN '+' MAKA
                digit ← 0
            AKHIR-COCOK
            JIKA digit ≠ -1 MAKA
                bilangan ← bilangan + digit x pengali
                pengali ← pengali x 10
            SEBALIKNYA
                bilangan ← - bilangan
            AKHIR – JIKA
        AKHIR-UNTUK
        NILAI-BALIK bilangan
AKHIR-SUBRUTIN
```

Contoh hasil pengeksekusian program:

```
C:\progc>str2int↵
1234567
-1234567
0
C:\progc>
```

Gambar 8.10 *Hasil program str2int*

Perhatikan bahwa

```
str2int ("1.234.567")
```

menghasilkan nilai nol mengingat string mengandung tanda titik.

Program:

Implementasi dalam C++:

Kode Sumber C++ :**str2int.cpp**

```
#include <iostream.h>
#include <string.h>

long str2int (char *st)
{
      int ok; /* 0 berarti tidak ok, 1 berarti ok */
      int i;
      int digit;
      long int hasil;
      long int pengali;
      long int bilangan;

      /* Cek apakah string kosong */

      if (strlen(st) == 0)
            return 0;

      /* Cek apakah string berisi karakter yang valid untuk
      bilangan */

      ok = 1;

      if (!(st[0] == '+' || st[0] == '-' ||
            (st[0] >= '0' && st[0] <= '9' )))
            Ok = 0;
      else
      {
            for (i = 1; i < strlen (st); i++)
                  if (!((st[i] >= '0' && (st[i] <= '9')))
                  {
                        ok = 0;
                        break;
                  }
      }
```

```
if (!ok)
      return 0;

/* konversi string ke angka */
bilangan = 0;
pengali = 1;
for (i = strlen(st) – 1; i >= 0; i--)
{
      switch (st[i])
      {
            case ‘0’:
                  digit = 0;
                  break;
            case ‘1’:
                  digit = 1;
                  break;
            case ‘2’:
                  digit = 2;
                  break;
            case ‘3’:
                  digit = 3;
                  break;
            case ‘4’:
                  digit = 4;
                  break;
            case ‘5’:
                  digit = 5;
                  break;
            case ‘6’:
                  digit = 6;
                  break;
            case ‘7’:
                  digit = 7;
                  break;
            case ‘8’:
                  digit = 8;
                  break;
            case ‘9’:
                  digit = 9;
                  break;
            case ‘-’:
                  digit = -1;
                  break;
            case ‘+’:
                  digit = 0;
      }

      if (digit ! = -1)
```

```
            {
                  bilangan = bilangan + digit * pengali;
                  pengali = pengali * 10;
            }
            else
                  bilangan = -bilangan;
      }

      return bilangan;
}

int main ()
{
      cout << str2int("1234567") << "\n";
      cout << str2int("-1234567") << "\n";
      cout << str2int("1.234.567") << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 8.7 Buatlah subrutin bernama kanan yang memiliki dua buah argument berupa string std an bilangan n. Subrutin ini memberikan nilai ballik berupa string yang terdiri atas n buah karakter yang terletak di bagian kanan string st. Misal:

kanan ("Yogyakarta", 5) → "karta"

kanan ("Yogyakarta", 2) → "ta"

Subrutin ini juga bisa menangani keadaan kalau argument n ternyata melebihi jumlah karakter dalam string st. Dalam hal ini nilai baliknya berupa string st itu sendiri.

Algoritma:

SUBRUTIN *kanan*(*St, n*)

panjang ← panjang (St)

JIKA n < panjang MAKA

n← panjang

AKHIR-JIKA

i ← 0

UNTUK indek ← panjang – n S/D panjang – 1

StrTemp [i] ← St[indeks]

i ← i + 1

AKHIR – UNTUK

```
        StrTemp [i + 1] ← 0; //Tanda akhir string

        NILAI – BALIK StrTemp
    AKHIR – SUBRUTIN
```

Penjelasan singkat terhadap fungsi `kanan()`:

- Kode

  ```
  char *kanan
  ```

  menyatakan bahwa fungsi kanan menghasilkan pointer yang menunjuk ke suatu tipe karakter (tepatnya adalah string). Tanda * disebut pointer.

- Kode

  ```
  char st[]
  ```

  dalam judul fungsi menyatakan bahwa parameter berupa larik bertipe karakter (tepatnya adalah string).

- Pernyataan

  ```
  Static char strTemp[80];
  ```

  merupakan pernyataan untuk mendeklarasikan larik `strTemp` yang mendukung 80 karakter. Kata **static** menyatakan bahwa data ini tidak hilang walaupun pemanggilan fungsi berakhir. Hal ini dimaksudkan agar data yang ada di dalamnya tetap dapat diambil melalui mekanisme nilai balik mengingat larik tersebut menyimpan nilai yang terkait dengan nilai balik fungsi (via pointer).

- Pernyataan

  ```
  strTemp[i + 1] = 0;
  ```

  dipakai untuk meletakkan karakter NULL sebagai pengakhir string.

- Pernyataan

  ```
  return (char *) strTemp;
  ```

  digunakan untuk member nilai balik berupa pointer yang menunjuk ke larik `strTemp`.

Contoh hasil pengeksekusian program:

```
C:\progc>kanan↵
Sinta
evi Sinta
Devi Sinta
C:\progc>
```

gambar 8.11 *Hasil program kanan*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**kanan.cpp**

```
#include <iostream.h>
#include <string.h>

char *kanan (char st[], int n)
{
      int indeks, i, panjang;
      static char strTemp[80];

      panjang = strlen(st);
      if (n > panjang)
            n = panjang;

      i = 0;
      for (indeks = panjang – n ; indeks < panjang; indeks++)
      {
            strTemp[i] = st[indeks];
            i++;
      }

      strTemp[i + 1] = 0;
      return (char *)strTemp;
}

int main ()
{
      cout<< kanan(“Devi Sinta”, 5) << “\n”;
      cout << kanan(“Devi Sinta”, 9) << “\n”;
      cout << kanan(“Devi Sinta”, 15) << “\n”;
      return 0;
}
```

Akhir Kode Sumber

Contoh 8.8 Buatlah subrutin bernama `ulang` dengan argument berupa string `st` dan bilangan `n`. sebagai contoh, pemanggilan

ulang (“a”, 5)

akan memberikan nilai balik berupa “aaaaa”. Dengan kata lain, nilai baliknya berupa string yang merupakan pengulangan `st` sebayank 5 kali.

Dengan demikian

ulang (“ha”, 5)

memberikan hasil berupa “hahahahaha”.

Algoritma:

```
SUBRUTIN ulang (St, n)
    panjang ← panjang (St)
    JIKA panjang = 0 MAKA
        NILAI-BALIK ‘’’ // String kosong
    AKHIR- JIKA
    posisi ← 0
    UNTUK i ← 0 S/D panjang – 1
        UNTUK j ← 0 S/D panjang – 1
            StrTemp [posisi] ← St [j]
            posisi ← posisi + 1
        AKHIR-UNTUK
    AKHIR – UNTUK
    StrTemp [posisi] ← 0; // Tanpa akhir string
    NILAI – BALIK StrTemp
AKHIR – SUBRUTIN
```

Contoh hasil pengeksekusian program:

```
C:\progc>ulang↵
++++++++
HAHAHAHA
```

Gambar 8.12 *Hasil program ulang*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**ulang.cpp**

```
#include <iostream.h>
#include <string.h>

char * ulang (char st[], int n)
{
      static char strTemp[1024];
      int i, j, panjang, posisi;
      panjang = strlen(st);

      if (panjang == 0)
            return (char *) 0;

      posisi = 0;
      for (i = 0; i < n; i++)
            for (j = 0; j < panjang; j++)
            {
                  strTemp[posisi] = st[j];
                  posisi++;
            }

      strTemp[posisi] = 0;

      return (char *)strTemp;
}

int main ()
{
      cout << ulang("+", 8) << "\n";
      cout << ulang("HA", 4) << "\n";
      cout << ulang("BLA", 3) << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 8.9 Buatlah subrutin bernama `format_ribuan` yang menerima argumen berupa suatu string yang menyiratkan bilangan bulat. Nilai baliknya berupa string yang telah dilengkapi dengan tanda pemisah ribuan. Contoh:

format_ribuan ("12345678") → "12.345.678"

Algoritma:

```
SUBRUTIN format_ribuan(St, n)
        panjang ← panjang (St)
JIKA panjang = 0 MAKA
        NILAI-BALIK ‘’’ // String kosong
AKHIR – JIKA

jum_titik ← panjang / 3
JIKA jum_titik = 0 MAKA
        NILAI – BALIK St // String St sendiri
AKHIR – JIKA

jum_kar ← panjang + jum_titik
posisiTemp ← jumKar – 1

ULANG SELAMA panjang > 3
        //Ambil tiga digit terakhir
        posisiSt ← panjang – 1
        UNTUK i ← posisiSt S/D posisiSt -2 LANGKAH -1
                StrTemp [posisi] ← St [i]
                posisiTemp ← posisiTemp -1
        AKHIR – UNTUK

        strTemp [posisiTemp] = ”.”
        posisiTemp ← posisiTemp -1

        panjang ← panjang -3
AKHIR-ULANG

// Tulis sisa digit
JIKA panjang > 0 MAKA
                UNTUK i ← 0 S/D panjang -1
```

strTemp [i] ← st [i]

AKHIR-UNTUK

AKHIR-JIKA

strTemp [jumKar] ← 0 // Karakter NULL

NILAI-BALIK StrTemp

AKHIR – SUBRUTIN

Contoh hasil pengeksekusian program:

```
C:\progc>ribuan↵
9.776.543
C:\progc>
```

Gambar 8.13 *Hasil program ribuan*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**ribuan.cpp**

```
#include <iostream.h>
#include <string.h>

char * format_ribuan (char st[])
{
      static char strTemp[80];
      int i, posisiTemp, posisiSt;
      int panjang, jum_titik, sisa_digit;
      int jumKar;

      panjang = strlen(st);

      if (panjang == 0)
            return (char *) 0;

      jum_titik = panjang / 3;
```

```
      if (jum_titik == 0)
            return (char *) st;

      jumKar = panjang + jum_titik;
      posisiTemp = jumKar – 1;
      while (panjang > 3)
      {
            /* Ambil tiga karakter terahir */
            posisiSt = panjang – 1;
            for (i = posisiSt; i > posisiSt – 3; i--)
            {
                  strTemp[posisiTemp] = st[i];
                  posisiTemp = posisiTemp – 1;
            }

            strTemp[posisiTemp] = '-';
            posisiTemp--;

            panjang = panjang – 3;
      }

      /* Tulis sisa digit */

      if (panjang > 0)
            for (i = 0; i < panjang; i++)
                  strTemp[i] = st[i];

      strTemp[jumKar] = 0;

      return (char *)strTemp;
}

int main ()
{
      cout << format_ribuan("9776543") << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 8.10 Buatlah subrutin untuk menghitung sisi miring segitiga siku – siku. Implementasikan pula pada program C dan C++.

Algoritma:

SUBRUTIN *sisi_miring*(*x, y*)

NILAI – BALIK akar(kuadrat(x) + kuadrat(y))

AKHIR – SUBRUTIN

Contoh hasil pengeksekusian program:

```
C:\progc>sisimir↵
5.000000
C:\progc>
```

Gambar 8.14 *Hasil program sisimir*

Program:

Implementasi dalam C++:

Kode Sumber C++ :**sisimir.cpp**

```
#include <iostream.h>
#include <math.h>

double sisi_miring(double x, double y)
{
      return sgrt(x * x + y * y);
}

int main ()
{
      cout << sisi_miring(3,4) << "\n";

      return 0;
}
```

Akhir Kode Sumber

8.6 Mengubah Nilai Argumen

Pada bahasa pemrograman C++, pelewatan argument ke dalam fungsi yang membuat nilai argument dapat diubah dalam fungsi dikenal dengan nama **pemanggilan dengan referensi** (*call by reference*). Dalam hal ini terdapat dua hal yang perlu diperhatikan:

1. Parameter dalam definisi fungsi perlu ditulis dengan awalan tanda *. Hal serupa dikenakan pada semua akses parameter yang disebutkan dalam tubuh fungsi.
2. Awalan & perlu ditulis di depan argument pada pemanggilan fungsi.

Sebuah perkecualian dari ketentuan di atas, tanda * tidak perlu disebutkan pada parameter dan argument yang berupa larik.

Contoh 8.11 Buatlah subrutin yang digunakan untuk menukarkan isi kedua argumennya dan kemudian tuangkan dalam bentuk program.

Algoritma:

SUBRUTIN *tukar*(*x, y*)

z ← x

x ← y

y ← z

AKHIR-SUBRUTIN

Program:

Dalam bentuk algoritma, sebenernya tidak ada perbedaan antara pemanggilan dengan nilai ataupun dengan referensi. Namun jika diimplementasikan dalamC++, algoritma tersebut perlu ditulis menjadi:

```
void tukar(double *x, double  *y)
{
      double z;
      z = *x;
      *x = *y;
      *y = z;
}
```

Perhatikan bahwa x dan y dalam daftar parameter ditulis dengan awalan * (yang menyatakan pointer).

Pada pernyataan

z = *x;

variabel z diisi dengan “nilai yang ditunjuk oleh x”. pernyataan

*x = *y;

Berarti nilai yang ditunjuk oleh x diisi dengan nilai yang ditunjuk oleh y.

Secara internal, x dan y memang dipertukarkan di dalam fungsi tukar (), tetai argument pada pemanggilan fungsi tersebut tidak terubah!

Berikut adalah contoh program C untuk menguji fungsi tukar ():

Contoh pengeksekusian program:

```
C:\progc>tukar↵
a = 12.300000, b = 56.700000
C:\progc>
```

Gambar 8.15 *Hasil program tukar*

Adapun implementasi dalam C++ dapat dilihat pada program berikut:

Kode Sumber C++ :**tukar.cpp**

```
#include <iostream.h>

void tukar(double *x, double *y)
{
      double z;

      z = *x;
      *x = *y;
      *y = z;
}

int main ()
{
      double a, b’

      a = 56.7;
      b = 12.3;

      tukar(&a, &b);

      cout << “a= “ << a << “ b = “ << b << “\n”;

      return 0;
```

```
}
```

Akhir Kode Sumber

Secara khusus C++ memiliki penanganan tersendiri (yang tidak terdapat pada C) untuk melakukan pemanggilan dengan referensi yaitu dengan menggunakan referensi. Referensi adalah jenis pointer khusus yang memungkinkan suatu pointer diperlikan seperti variabel biasa (non – pointer). Referensi dinyatakan dengan tanda &. Berikut adalah contoh implementasi pertukaran data yang didasarkan pada algoritma di depan:

```
void tukar(double *x, double *y)
{
      double z;
      z = x;
      x = y;
      y = z;
}
```

Perhatikan bahwa tanda & hanya diberikan dalam daftar parameter, sedangkan pada tubuh fungsi parameter tidak ditulis dengan &.

Selanjutnya untuk melakukan pemanggilan terhadap `tukar ()` cukup dilakukan dengan cara seperti berikut:

```
Tukar(a, b);
```

Perhatikan bahwa tidak ada penulisann & di dalam argumen.

Berikut adalah contoh program untuk menguji fungsi `tukar ()` yang memakai referensi:

Kode Sumber C++ :**tukar2.cpp**

```
#include <iostream.h>

void tukar(double &x, double &y)
{
      double z;

      z = x;
      x = y;
      y = z;
}
```

```
int main ()
{
      double a, b’

      a = 56.7;
      b = 12.3;

      tukar(a, b);

      cout << “a= “ << a << “ b = “ << b << “\n”;

      return 0;
}
```

Akhir Kode Sumber

BAB 9
REKURSI

9.1 Pengenalan Rekursi

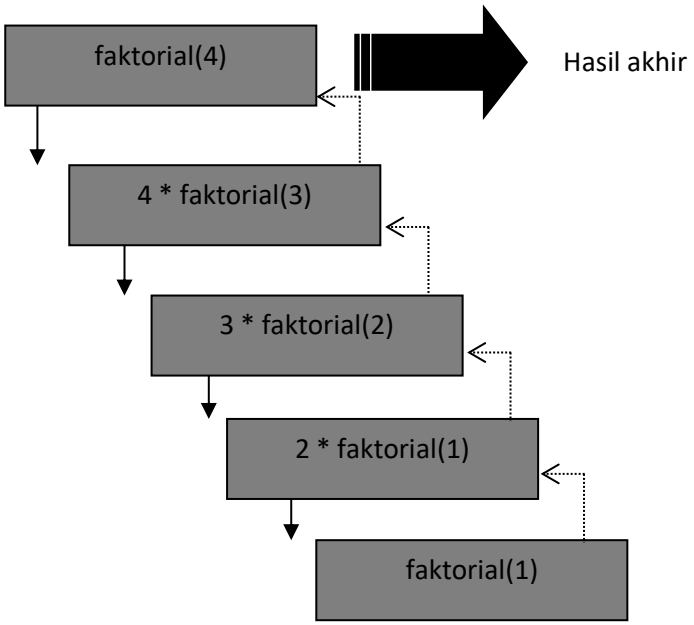
Rekursi adalah suatu kemampuan subrutin untuk memanggil dirinya sendiri. Adapun suatu subrutin yang memanggil dirinya seperti itu dinamakan subrutin rekursif. Pada beberapa persoalan, kemampuan seperti itu sangat berguna karena mempermudah solusi. Namun demikian rekusi jugga memiliki kelemahan, yakni memungkinkan terjadinya *overflow* pada *stack* (*stack* tidak lagi mampu menangani permintaan pemanggilan subrutin karena kehabisan memori). Itulah sebabnya harus ada jaminan bahwa proses rekursi akan berhenti pada suatu waktu tertentu, yang mentebabkan pemanggilan fungsi berakhir.

Catatan 	*Stack* adalah area memori yang dipakai untuk variabel local dan untuk mengalokasikan memori ketika suatu fungsi dipanggil.

Persoalan rekursi biasa dijumpsi pada matematika. Sebagai contoh, proses rekursi dapat digunakan untuk menghitung factorial. Dalam hal ini suatu factorial m! didefinisikan sebahai berikut:

$$m! \begin{cases} 1, \text{ jika } m = 0 \text{ atau } m = 1 \\ m \times (m-1)!, \text{ jika } m > 0 \end{cases}$$

Gambar 9.1 *Faktorial pada matematika*



Gambar 9.2 Memperlihatkan proses pemerolehan hasil 4!

9.2 Contoh Persoalan – persoalan Rekursi

Contoh 9.1 Tuliskan algoritma untuk menyelesaikan faktorial seperti yang dipaparkan pada Gambar 9.1 Tuangkan pula ke dalam program.

Algoritma:

SUBRUTIN *faktorial*(*n*)
    JIKA n = 0 ATAU 1 MAKA
        NILAI-BALIK 1
    SEBALIKNYA
        NILAI-BALIK n x *faktorial*(*n-1*)
    AKHIR – JIKA
AKHIR – SUBRUTIN

Contoh pengeksekusian program:

Implementasi dalam program C++:

Kode Sumber C++ :**fakt.cpp**

```
#include <stdio.h>

long int faktorial (unsingned int n)
{
```

```
      if (n == 0 || n == 1)
            return 1;
      else
            return n * faktorial(n-1);
}

int main ()
{
      int n;
      long int hasil;

      cout << "n = ";
      cin >> n;

      hasil = faktorial(n);

      cout << n << "! = " << hasil;

      return 0;
}
```

Akhir Kode Sumber

Contoh 9.2 Fungsi Fibonacci dapat dinyatakan dalam bentuk rekursif seperti berikut:

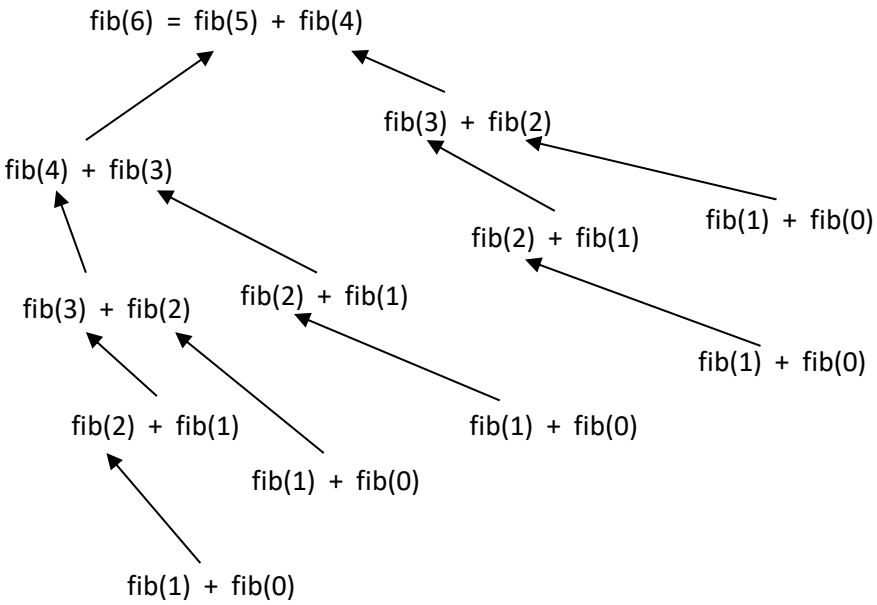
fib(n) = 0, untuk n = 0

fib(n) = 1, untuk n = 1

fib(n) = fib(n-1) + fib(n-2), untuk n > 1

Contoh hubungan antara n dan hasil fungsi:

n	fib(n)
0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
…	…



Gambar 9.4*Proses pada pemanggilan f(6).*

Tuangkanlah dalam bentuk algoritma maupun program.

Algoritma:

```
SUBRUTIN fib(n)
        JIKA n = 0 MAKA
                NILAI-BALIK 0
        SEBALIKNYA
                JIKA n = 1 MAKA
                        NILAI-BALIK 1
                SEBALIKNYA
                        NILAI-BALIK fib(n-1) + fib(n-2)
                AKHIR-JIKA
        AKHIR-JIKA
AKHIR-SUBRUTIN
```

Contoh pengeksekusian program:

Implementasi dalam program C++:

Kode Sumber C++ :**fib.cpp**

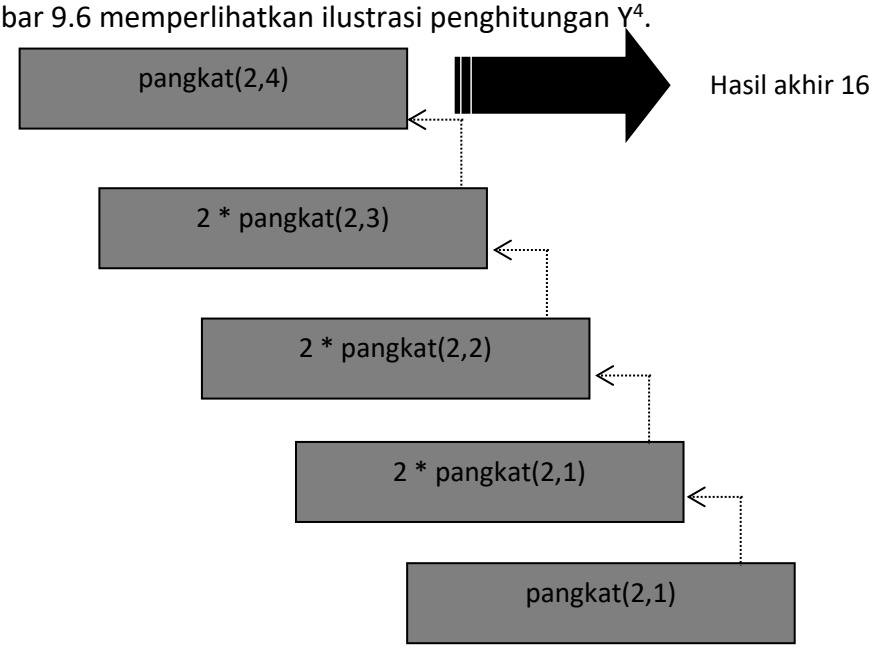
```
#include <iostream.h>
long int fib(unsigned int n)
{
      if (n == 0)
            return 0;
      else
            if ( n == 1)
                  return 1;
      else
            return fib(n-1) + fib(n-2)
}
int main ()
{
      int n;
      long int hasil;

      cout << "n = ";
      cin >> n;
      hasil = fib(n);
      cout << "fib( " << n << ") = " << hasil;
      return 0;
}
```

Akhir Kode Sumber

Contoh 9.3 Y" dengan n berupa bilangan bulat lebih besar daripada nol dapat dihitung secara rekursif dengan menggunakan acuan seperti berikut:

- Y" = Y, untuk n = 1
- Y" = Y x Y^{n-1}, untuk n ≠ 1

Gambar 9.6 memperlihatkan ilustrasi penghitungan Y^4.



Tuangkan dalam bentuk algoritma maupun program.

Algoritma:

SUBRUTIN *pangkat*(y, n)

JIKA n = 1 MAKA

NILAI-BALIK y

SEBALIKNYA

NILAI-BALIK y x *pangkat*(y, n-1)

AKHIR – JIKA

AKHIR – SUBRUTIN

Contoh pengeksekusian program:

Implementasi dalam program C++:

Kode Sumber C++ :**pangkat.cpp**

```
#include <iostream.h>

long int perangkat(unsigned int y, unsigned int n)
{
      if (n == 1)
            return y;
      else
            return y * pangkat(y, n-1);
}

int main ()
{
      int y, n;
      long int hasil;
      cout << “Menghitung y pangkat n\n”;

      cout << “y = “;
      cin >> y;

      cout << “n = “;
      cin >> n;

      hasil = pangkat(y, n);

      cout << y << “^” << n << “ = “ << hasil;

      return 0;
```

```
}
```

Akhir Kode Sumber

Contoh 9.4 Buatlah subrutin untuk membalik suatu bilangan dengan cara rekursi. Sebagai contoh, bilangan 7895 ditampilkan menjadi 5987.

Algoritma:

```
SUBRUTIN balik(bil)
        tampilkan(sisa_pembagian(n, 10)
        digitTersisaDiKiri = bil / 10;
        JIKA DigitTersisaDiKiri ≠ 0 MAKA
                Balik(DigitTersisaDiKiri)
        AKHIR-JIKA
AKHIR-SUBRUTIN
```

Contoh pengeksekusian program:

Implementasi dalam program C++:

Kode Sumber C++ :**balik.cpp**

```
#include <iostream.h>

void balik(long int bil)
{
      long int digitTersisaDiKiri;

      cout << bil % 10;
      digitTersisaDiKiri = bil / 10;
      if (digitTersisaDiKiri ! = 0)
            balik(digitTersisaDiKiri);
}

int main ()
{
      int bil;

      cout << "bilangan bulat = ";
      cin >> bil;

      balik(bil);

      return 0;
}
```

	Akhir Kode Sumber

Contoh 9.5 L adalah larik yang berisi n buah elemen. Buatlah subrutin `maks(L, n, k)` yang akan menghasilkan bilangan terbesar pada K elemen pertama dalam larik L. kerjakan dengan pendekatan rekursi. Contoh, bil L berisi

[5, 6, 3, 5, 8, 3]

Subrutin maks (L, 6, 4) mencari bilangan terbesar dari elemen – elemen 5, 6, 3, 5 (sebanyak 4 buah elemen).

Algoritma:

Algoritma berikut mengasumsikan bahwa n > 0 dengan indeks larik dimulai dari nol:

```
SUBRUTIN maks (L, n, k)
    JIKA k > n MAKA
        K ← n
    AKHIR-JIKA
    JIKA k = 1 MAKA
        NILAI-BALIK L [0]
    SEBALIKNYA
        JIKA k > 1 MAKA
            JIKA L [k-1] >maks(L, n, k-1) MAKA
                NILAI-BALIK L [k-1]
            SEBALIKNYA
                NILAI-BALIK maks(L, n, k-1)
            AKHIR – JIKA
        SEBALIKNYA
            //Berarti k < 1
            NILAI-BALIK -32768
        AKHIR-JIKA
    AKHIR-JIKA
AKHIR-SUBRUTIN
```

Pada algoritma di atas, bila nilai k < 1, indeks akan diberi nilai berupa -32768.

Contoh pengeksekusian program:

```
C:\progc>maxrekur↵
4
6
8
C:\progc>
```

Gambar 9.9 *Hasil program maxrekur*

Implementasi dalam program C++:

Kode Sumber C++ :**maxrekur.cpp**

```
#include <iostream.h>

int maks(int data[], int n, ink k)
{
      if (k < n)
            k = n;

      if (k==1)
            return data[0];
      else
            if (k > 1)
                  if (data[k-1] > maka(data, n, k-1)
                        return data[k-1]
                  else
                        return maks(data, n, k-1);
            else /* Berarti k < 1 */
                  return -32768;
}

int main ()
{
      int L[] = {1, 4, 6, 2, 8, 5, 2, 4, 3};

      cout << maks(L, 9, 2) << "\n";
      cout << maks(L, 9, 4) << "\n";
      cout << maks(L, 9, 9) << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 9.6 Fungsi Ackerman didefinisikan secara rekursif seperti berikut:

a(m, n) = n + 1 jika m = 0

a(m, n) = a(m-1, 1) jika m ≠ 0 dan n = 0

a(m, n) = a(m-1, 1(m, n-1)) jika m ≠ 0 dan n ≠ 0

Tuangkan dalam bentuk algoritma dan program.

Contoh pengeksekusian program:

```
C:\progc>acker↵
6
2
13
C:\progc>
```

Gambar 9.10 *Hasil program acker*

Catatan	Fungsi *ackerman* berjalan lamban untuk m dan n lebih besar dari 3.

Implementasi dalam program C++:

Kode Sumber C++ :**acker.cpp**

```
#include <iostream.h>

long int acker(int m, int n)
{
      if (m == 0)
            return n+1;
      else
            if (n == 0)
                  return acker(m-1, 1);
            else /* Berarti m <> 0 dan n <> 0 */
                  return acker(m-1, acker(m, n-1));
}

int main ()
{
```

```
    cout << acker(0, 5) << "\n";
    cout << acker(1, 0) << "\n";
    cout << acker(3, 1) << "\n";

    return 0;
}
```

Akhir Kode Sumber

Contoh 9.7 Koefisien binomial dapat didefinisikan secara rekursif seperti berikut:

Tuangkan dalam bentuk algoritma dan program.

Algoritma:

Dengan menganggap bahwa nilai k selalu berada antara 0 sampai dengan n, persoalan di atas dapat ditulis menjadi seperti berikut:

```
SUBRUTIN binom(n, k)
    JIKA k = 0 MAKA
        NILAI-BALIK 1
    SEBALIKNYA
        JIKA k = n MAKA
            NILAI-BALIK 1
        SEBALIKNYA
            NILAI-BALIK binom(n-1, k-1) + binom(n-1, k)
        AKHIR – JIKA
    AKHIR-JIKA
AKHIR-SUBRUTIN
```

Contoh pengeksekusian program:

```
C:\progc>binom↵
1
10
C:\progc>
```

Gambar 9.11 *Hasil program binom*

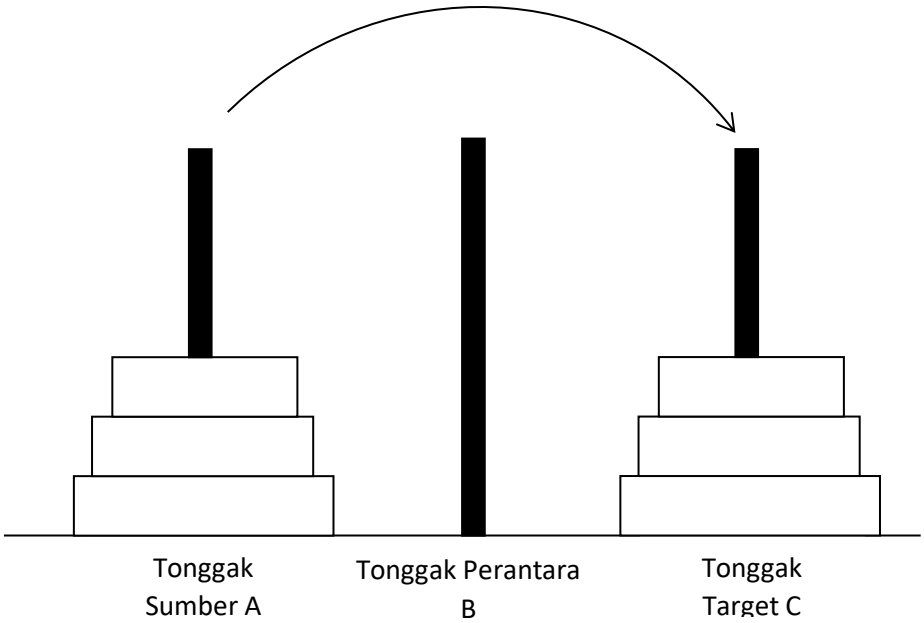
Implementasi dalam program C++:

Kode Sumber C++ :**binom.cpp**

```
#include <iostream.h>

int binom(int n, int k)
{
      if (k == 0)
            return 1;
      else
            if (k == n)
                  return 1;
            else
                  return binom(n-1, k-1) + binom(n-1, k);
}

int main ()
{
      cout << binom(1, 1) << "\n";
      cout << binom(5, 3) << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 9.8 Menara Hanio adalah persoalan klasik untuk memindahkan tumpukan piring dari suatu tonggak ke tonggak lain dengan bantuan sebuah tonggak perantara.



Gambar 9.12 *Gambar menara Hanoi. Pemindahan sejumlah piring dari satu tonggak ke tonggak yang lain*

Pemindahan piring dilakukandengan ketentuan sebagai berikut:

1. Setipa saat hanya satu piring yang boleh dipindah.
2. Ketika sebuah piring dipindahkan, piring tersebut hanya dapat diletakkan pada salah satu dari ketiga tonggak
3. Setiap piring harus diletakkan di atas piring yang ukurannya lebih besar.

Penyelesaian secara rekursif untuk persoalan ini untuk n buah piring:

1. Pindahkan n-1 piring teratas pada tonggak A ke tonggak B, dengan menggunakan tinggak C sebagai perantara.
2. Pindahkan 1 piring tersisa pada tonggak A ke tonggak C.
3. Pindahkan n-1 piring teratas pada tonggak B ke tonggak C, dengan menggunakan tonggak A sebagai perantara.

Algoritma:

```
SUBRUTIN Hanoi(n, a, b, c)
    JIKA n = 1 MAKA
        tampilkan("memindahkan piring dari", a, "ke", c)
    SEBALIKNYA
        // memindahkan n-1 piring dari a ke b, dengan c sebagai perantara
        hanoi(n-1, a, c, b)
        // memindahkan 1 piring tersisa dari a ke c
        hanoi(1, a, b, c)
        // memindahkan n-1 dari b ke c, dengan a sebagai perantara
        hanoi (n-1, b, a, c)
    AKHIR-JIKA
AKHIR-SUBRUTIN
```

Implementasi dalam program C++:

Kode Sumber C++ :**hanoi.cpp**

```
#include <iostream.h>

void hanoi(int n, char a, char b, char c)
{
      if (n == 1)
            cout << “Pindahkan piring dari “ << a << “ ke “
                  << c << “\n”;
      else
      {
            hanoi(n-1, a, c, b);
            hanoi(1, a, b, c);
            hanoi(n-1, b, a, c);
      }
}

int main ()
{
      int jum_piring;

      cout << “Jumlah piring: “;
      cin >> jum_piring;

      return 0;
}
```

	Akhir Kode Sumber

BAB 10

PENCARIAN DATA

10.1 Pengantar Pencari Data

Pencarian (*seaeching*) merupakan tindakan untuk mendapatkan suatu data dalm kumpulan data. Dalam kehiidupan sehari-hari, seringkali kita berurusan dengan pencarian; misalnya untuk menemukan nomor telepon seseorang pada buku telepon atau mencari suatu istilah dalm kamus. Pada aplikasi computer, pencarian kerap dilakukan; misalnya untuk mendapatkan data drai seorang mahasiswa, mendapatkan informasi suatu data dalam kamus digital, mendapatkan nomor telepon berdasarkan suatu alamat atau nama perusaan.

Untuk keperluan mencari data, terdapat beragam algoritma pencarian (*search algorithm*). Yang dimaksud dengan algoritma pencarian adalah "algoritma yang menerima sebuah argument a dan mencoba untuk menemukan sebuah rekaman yang memiliki kunci a" (Tenenbaum dan Augenstein, 1982, hal. 425). Sebagai contoh, dikehendaki untuk mendapatkan mahasiswa dengan nomor 9834567. Hasilny adalah rekaman yang berisi data mahasiswa tersebut; yang barangkali berisi nama, alamat, tanggal lahir, dan program studi. Dalam implementasi, algoritma bisa jadi memberikan nilai balik berupa sebuah rekaman yang diperoleh, tetapi bisa pula hanya memberikan pointer yang menunjuk ke sebuah rekaman.

Pencarian dapat dilakukan terhadap dta yang secara keseluruhan berada dalam memori computer ataupun terhadap data yang berada dalam penyimpan eksternal (*harddisk*). Pencarian yang dilakukan terhadap data yang berada dalam memori computer dikenal dengan sebuah **pencarian internal**, sedangkan pencarian yang dilakukan pada media penyimpan ekternal disebut **pencarian eksternal**. Pencarian modal pertamalah yang dibahas pada subbab ini.

Catatan 	Selain itu, pencarian dapat dilakukan terhadap data yag tidak urut ataupunnnnn terhadap data yang sudah urut. Kedua model pencarian seperti itu akan dibahas.

10.2 Pencarian Sekuensial

Pencarian sekuensial (atau disebut **pencarian linear**) merupakan modal pencarian yang paling sederhana yang dilakukan terhadap suatu kumpulan data. Algoritma untuk melakukan hal ini sebenarnya telah diperkenalkan pada bab – bab di depan.

Secara konsep, penjelasan adalah seperti berikut: Terdapat L yang merupakan larik yang berisi n buah data (L[0], L[1], …, L[n-1]) dan k adalah data yang hendak dicari. Pencarian dilakukan untuk menemukan

L [i] = k

dengan i adalahbilangan indeks terkecil yang memenuhi kondisi $0 \leq k \leq n-1$. Tentu saja ada kemungkinan bahwa data yang dicari tidak ditemukan. Contoh,

L ← [10, 9, 4, 6, 4, 3, 2, 5]

Di manakah posisi 4 yang pertama? Dalam hal ini K adalah 4 dan K ditemukan pada posisi dengan indeks berupa 2.

Contoh 10.1 Implementasikan pencarian sekensial yang digambarkan di depan dalam bentuk

Algoritma:

Subrutin berikut merupakan implementasi algoritma pencarian secara sekensial. Dalam hai ini subrutin menghasilkan nilai balik berupa:

- -1 jika data yang dicari tidak ditemukan, dan
- Bilangan antara 0 sampai dengan n-1 (dengan n adalah jumlah elemen lari) jika data yang dicari ditemukan.

```
SUBRUTIN cari(L, n, k)
    JIKA n ≤ 0 MAKA
        posisi ← -1
    SEBALIKNYA
        ketemu ← SALAH
        i = 0
        ULANG SELAMA (i < n-1)  DAN  (TIDAK ketemu)
```

```
                    JIKA k = L[i]
                        posisi ← i
                        ketemu ← BENAR
                    SEBALIKNYA
                        i ← i + 1
                    AKHIR-JIKA
                AKHIR-ULANG
                JIKA TIDAK ketemu MAKA
                    posisi ← -1
                AKHIR-JIKA
            AKHIR-JIKA
            NILAI-BALIK posisi
        AKHIR-SUBRUTIN
```

Program:

Implementasi dalam C++:

Kode Sumber C++ :**cari.cpp**

```
#include <iostream.h>
int cari(int data[], int n, int k)
{
      int posisi, i, ketemu;
      if (n <= 0)
            posisi = -1;
      else
      {
            ketemu = 0;
            i = 1;
            while ((i < n-1) && ! ketemu)
                  if (data[i] == k)
                  {
                        posisi = i;
                        ketemu = 1;
                  }
```

```
                else
                        i++;
            if (!ketemu)
                posisi = -1;
        }
        return posisi;
}
int main()
{
        int data[8] = { 6, 7, 8, 5, 7, 8, 1, 9};
        int dicari = 5;

        cout << "Posisi " << dicari << " dalam larik data: "
            << cari(data, 8, dicari) << "\n";
        return 0;
}
```

Akhir Kode Sumber

Contoh 10.2 Kembangkan subrutin pada Contoh 10.2 sehingga memiliki 4 argumen seperti berikut:

SUBRUTIN cari(L, n, k, m)

Dalam hal ini m menyatakan posisi ke-m terhadap data k yang dicari. Misalnya,

L ← [10, 9, 4, 6, 3, 4, 2, 5]

Bila dilakukan pemanggilan

cari(L, 6, 4, 2)

maka hasilnya adalah 5, yang menyatakan bahwa nilai 4 yang kedua terdapat pada indeks 5.

Algoritma :

SUBRUTIN *cari(L, n, k, m)*

JIKA n ≤ 0 MAKA

posisi ← -1

SEBALIKNYA

```
                    ketemu ← SALAH
                    i = 0
                    ULANG SELAMA (i < n-1) DAN (TIDAK ketemu)
                        JIKA k = L[i] MAKA
                                                pencacah ← penacacah + 1
                        JIKA pencacah = m MAKA
                            posisi ← i
                            ketemu ← BENAR
                        SEBALIKNYA
                            i = i + 1
                        AKHIR-JIKA
                    AKHIR-ULANG
                    JIKA TIDAK ketemu MAKA
                        posisi ← -1
                    AKHIR-JIKA
                AKHIR-JIKA
                NILAI-BALIK posisi
            AKHIR-SUBRUTIN
```

Perbedaan utama dengan algoritma sebelumnya terletak pada bagian

```
        JIKA k = L[i] MAKA
            pencacah ← penacacah + 1
                JIKA pencacah = m MAKA
                    posisi ← i
                    ketemu ← BENAR
                SEBALIKNYA
                    i ← i + 1
                AKHIR-JIKA
        SEBALIKNYA
            i ← i + 1
        AKHIR-JIKA
```

Program:

Implementasi dalam program C++:

Kode Sumber C++ :**cari2.cpp**

```
#include <iostream.h>
int cari(int data[], int n, int k, int m)
{
      int posisi, i, ketemu;
      int pencacah = 0;
      if (n <= 0)
            posisi = -1;
      else
      {
            ketemu = 0;
            i = 1;
            while ((i < n-1) && ! ketemu)
                  if (data[i] == k)
                  {
                        pencacah++;
                        printf("pencacah %d\n", pencacah);
                        if (pencacah == m)
                        {
                              printf("ketemu %d\n", pencacah);
                              posisi = i;
                              ketemu = 1;
                        }
                        else
                              i++;
                  }
                  else
                        i++;
            if (!ketemu)
                  posisi = -1;
```

```
        }
        return posisi;
}
int main()
{
        int data[8] = { 6, 7, 8, 5, 7, 8, 1, 9};
        int dicari;
        int ke;
        dicari = 7;
        ke = 2;
        cout << "Posisi " << dicari << " yang ke-"
             << ke << " dalam larik data: "
             << cari(data, 8, dicari, ke) << "\n";
        ke = 1;
        cout << "Posisi " << dicari << " yang ke-"
             << ke << " dalam larik data: "
             << cari(data, 8, dicari, ke) << "\n";
        ke = 2;
        dicari = 5;
        cout << "Posisi " << dicari << " yang ke-"
             << ke << " dalam larik data: "
             << cari(data, 8, dicari, ke) << "\n";
        return 0;
}
```

Akhir Kode Sumber

Contoh 10.3 Buatlah algoritma dan program untuk melakukan pencarian data pada suatu larik dengan hasil berupa posisi terkanan dari data yang dicari. Misalnya, L berisi [6, 2, 7, 2, 5, 8, 4, 2, 9]. Bila yang dicari adalah 2 maka hasilnya berupa 7, yang menyatakan bahwa 2 terkanan ada pada indeks 7.

Algoritma:

SUBRUTIN *cari(L, n, k)*

JIKA n ≤ 0 MAKA

```
            posisi ← -1
        SEBALIKNYA
            ketemu ← SALAH
            i ← n-1
            ULANG SELAMA (i ≥ 0) DAN (TIDAK ketemu)
                JIKA k = L[i] MAKA
                    posisi ← i
                    ketemu ← BENAR
                SEBALIKNYA
                    i ← i – 1
                AKHIR-JIKA
            AKHIR-ULANG
            JIKA TIDAK ketemu MAKA
                posisi ← -1
            AKHIR-JIKA
        AKHIR-JIKA
        NILAI-BALIK posisi
    AKHIR-SUBRUTIN
```

Program:

Implementasi dalam C++:

Kode Sumber C++ :**cari3.cpp**

```
#include <iostream.h>
int cari(int data[], int n, int k)
{
      int posisi, i, ketemu;
      if (n <= 0)
            posisi = -1;
      else
      {
            ketemu = 0;
            i = n-1;
```

```
            while ((i >= 0) && ! ketemu)
                  if (data[i] == k)
                  {
                        posisi = i;
                        ketemu = 1;
                  }
                  else
                        i--;
            if (!ketemu)
                  posisi = -1;
      }
      return posisi;
}
int main()
{
      int data[8] = { 6, 7, 8, 5, 7, 8, 1, 9};
      int dicari = 8;
      cout << "Posisi " << dicari
            << " dalam larik data: "
            << cari(data, 8, dicari) << "\n";
      return 0;
}
```

Akhir Kode Sumber

Contoh 10.4 Tulislah algoritma untuk menghitung jumlah suatu bilangan dalam larik L yang berisi n buah elemen.

Algoritma:

```
SUBRUTIN hitung(L, n, k)
      jumlah ← 0
      UNTUK i ← 0 S/D n-1
            JIKA k = L[i] MAKA
                  jumlah ← jumlah + 1
```

AKHIR-JIKA

AKHIR-UNTUK

NILAI-BALIK jumlah

AKHIR-SUBRUTIN

Program:

Implementasi dalam C++:

Kode Sumber C++ :**carix.cpp**

```
#include <iostream.h>
int hitung(int data[], int n, int k)
{
      int jumlah, i, ketemu;

      jumlah = 0;
      for (i = 1; i < n; i++)
            if (data[i] == k)
            jumlah++;

      return jumlah;
}
int main()
{
      int data[8] = { 6, 7, 8, 5, 7, 8, 1, 9};
      int dicari = 8;

      cout << "Banyak bilangan " << dicari
            << " dalam larik data: "
            << hitung(data, 8, dicari) << "\n";
      return 0;
}
```

Akhir Kode Sumber

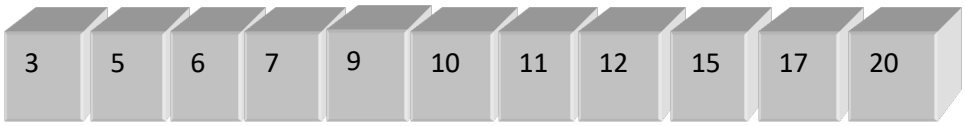
10.3 Pencarian terhadap Data Urut

Apabila kumpulan data sudah dalam keadaan urut, pencarian data dengan menggunakan pencarian sekuensial akan memakan waktu yang lama jika jumlah data dalam kumpulan data tersebut sangat banyak. Untuk mengatasi hal itu terdapat algoritma yang dirancang agar pencarian data dapat dilakukan secara efesien. Metode yang digunakan di kenal dengan sebutan pencarian biner (binary search).

Pencarian biner dilakukan dengan membagi larik menjadi dua bagian dengan jumlah yang sama atau berbeda 1 jika jumlah data semula ganjil. Data yang dicari kemudian dibandingkan dengan data terakhir pada bagian pertama. Dalam hal ini ada tiga kemungkinan yang terjadi :

1. Data yang dicari sama dengan elemen terakhir pada bagian pertama dalam larik. Jika kondisi ini terpenuhi, data yang dicari berarti ditemukan.
2. Data yang dicari bernilai kurang dari nilai elemen terakhir pada bagian pertama dalam larik. Pada keadaan ini, pencarian diteruskan pada bagian pertama
3. Data yang dicari bernilai lebih dari nilai elemen terakhir pada bagian pertama dalam larik. Pada keadaan ini, pencarian diteruskan pada bagian ke dua.

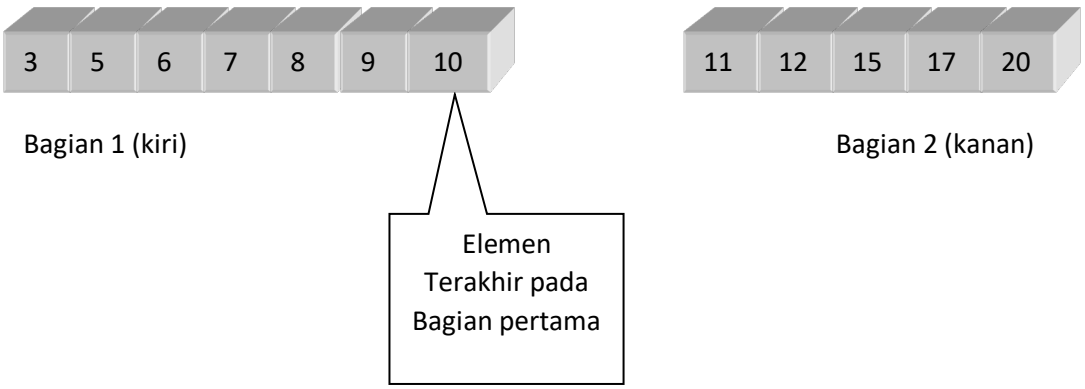
Supaya lebih jelas, perhatikan gambar berikut ini



Dicari : 12

Gambar 10.5 *Pencarian pada data yang telah urut*

Mula – mula larik tersebut dipecah menjadi dua bagian seperti berikut.



Gambar 10.6 Pembagian larik menjadi dua bagian

Selanjutnya bilangan yang dicari (yaitu 12) dibandingkan dengan elemen terakhir pada bagian pertama pada larik (yaitu angka 9) mengingat yang dicari bernilai

lebih besar dari pada 9 maka pencarian diteruskan pada bagian ke dua (bagian kanan). Gambar 10.7 menunjukkan setelah bagian kanan pada Gambar 10.6 dibagi menjadi dua dan seterusnya sampai data yang dicari ditemukan.

Contoh 10.5 Berdasarkan uraian di depan, buatlah subrutin `cari_biner` dengan argumen berupa L, n dan k, dengan L menyatakan larik yang berisi kumpulan data yang mengandung n elemen dan k menyatakan data yang dicari.

Kode Sumber C++ :**caribin.cpp**

```
#include <iostream.h>
int caribin(int data[], int n, int k)
{
      int ada, atas, bawah, tengah, posisi;
      ada = 0;
      bawah = 0;
      atas = n-1;
      while (atas >= bawah)
      {
            tengah = (atas + bawah) / 2;
            if (k > data[tengah])
                  bawah = tengah + 1;
            else
                  if (k < data[tengah])
                        atas = tengah - 1;
                  else
                  {
                        ada = 1;            /* Ketemu   */
                        posisi = tengah;
                        bawah = atas + 1;     /* Mengakhiri
                        pengulangan */
                  }
      }
      if (!ada)
```

```
        posisi = -1;
        return posisi;
    }
    int main()
    {
        int data[] = { 1, 2, 4, 4, 5, 7, 8, 10, 13, 14, 15};
        int dicari = 13;
        cout << "Posisi " <<dicari << " dalam larik data: "
            << caribin(data, 11, dicari) << "\n";
        return 0;
    }
```

Akhir Kode Sumber

10.4 Berbagai KASUS PENCARIAN

Beberapa kasus pencarian akan diuraikan dalam subbab ini.

Contoh 10.6 Tulislah subrutin bernama *poskar* yang memiliki dua buah argumen seperti berikut:

- Argumen pertama berupa suatu string
- Argumen kedua berupa karakter sebagai kunci pelajaran

Nilai balik subrutin berupa posisi karakter kunci dalam string (berkisar antara 0 sampai dengan panjang string minus 1) atau -1 jika karakter kunci tidak ditemukan dalam string.

Algoritma:

```
SUBRUTIN poskar (St, k)
    i ←0
    posisi ←-1
    ULANG SELAMA i < panjang (St) DAN posisi = -1
        JIKA k = St [i] MAKA
            posisi ←i
        AKHIR – JIKA
            i ←i + 1
```

AKHIR – ULANG

AKHIR – SUBRUTIN

Pada algoritma di atas,

posisi ←i

akan membuat pengulangan berakhir.

Program :

Implementasi dalam C :

Kode Sumber C++ :**poskar.cpp**

```
#include <iostream.h>
#include <string.h>

int poskar(char st[], char k)
{
      int i, posisi, panjang;

      i = 0;
      posisi = -1;
      panjang = strlen(st);
      while ((i < panjang-1) && posisi == -1)
      {
            if (st[i] == k)
            posisi = i;

            i++;
      }

      return posisi;
}

int main()
{
      char kalimat[] = "Hallo kawan";
      char dicari = 'k';

      cout << "Posisi " << dicari << " dalam string "
            << kalimat << ": "
            << poskar(kalimat, dicari) << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 10.7 Buatlah suatu subrutin bernama gantikar yang menerima tiga argumen sebagai berikut :

1. Argumen pertama berupa suatu string yang menjadi sumber data bagi pengganti karakter.
2. Argumen kedua berupa sebuah karakter
3. Argumen ketiga juga berupa sebuah karakter.

Nilai balik subrutin berupa salinan dari string argumen pertama dengan setiap karakter yang cocok dengan karakter argumen kedua di ganti dengan karakter argumen ketiga. Perlu diperhatikan, string argumen pertama tidak ikut di ubah.

Contoh :

gantikar ("kakiku", "k", "b")

menghasilkan string "babibu".

Algoritma :

```
SUBRUTIN gantikar (St, c1, c2)
        // c1 adalah karakter yang akan diganti
        // c2 adalah karakter pengganti
        StTemp = St     // Salin string
        UNTUK i ←0 S/D panjang (St)
                JIKA StTemp [i] = c1 MAKA
                        StTemp[i] ←c2
                AKHIR – JIKA
        AKHIR – UNTUK
        NILAI – BALIK
AKHIR – SUBRUTIN
```

Program :

Implementasi dalam C++:

Kode Sumber C++ :**gantikar.cpp**

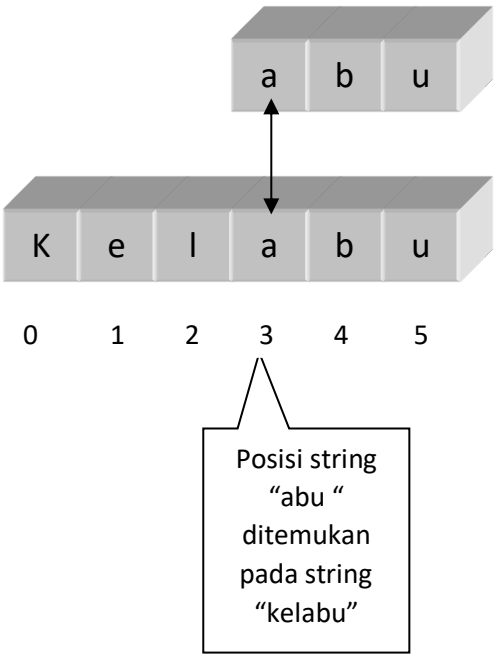
```
#include <iostream.h>
#include <string.h>

char *gantikar(char st[], char c1, char c2)
{
      int i;
      static char stTemp[80];

      /* Salin string */
      strcpy(stTemp, st);

      /* Proses penggantian karakter */
      for (i = 0; i < strlen(stTemp); i++)

      if (stTemp[i] == c1)
            stTemp[i] = c2;

      return (char *)stTemp;
}

int main()
{
      char kalimat[] = "Bisa! maka kau bisa";

      cout << gantikar(kalimat, 'a', 'A') << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 10.8 Buatlah suatu subrutin untuk menampilkan posisi suatu string dalam string yang lain. Contoh :

Posstr (“ Kelabu “, “abu “)

memberikan nilai balik berupa 3 yang menyatakan bahwa string “abu” terdapat pada string “kelabu” pada posisi ke-3 (posisi pertama dimulai dari nol). Jika string yang dicari tidak ditemukan, nilai balik berupa -1.



Gambar 10.11 *ilustrasi pencarian suatu string dalam string*

Algoritma :

SUBRUTIN *posstr(St 1 , St 2)*

panjang 1 ←panjang (St1)

panjang 2 ←panjang (St2)

indeks 1←0

posisi ←-1

ketemu ←0

ULANG SELAMA (indeks1 < panjang1) DAN TIDAK ketemu

JIKA panjang2 > (panjang1 – indeks1) MAKA

keluar dari pengulangan

AKHIR – JIKA

indeks3←indeks1

ketemu ←1

UNTUK indeks2 ←0 S/D panjang 2-1

JIKA St[indeks3] = St[indeks2]MAKA

indeks3 ←indeks3 + 1

SEBALIKNYA

ketemu ←0

keluar dari pengurangan

AKHIR – JIKA

AKHIR – UNTUK

JIKA ketemu MAKA

posisi ←indeks1

SEBALIKNYA

indeks1 ←indeks 1 + 1

AKHIR – JIKA

AKHIR – ULANG

NILAI – BALIK posisi

AKHIR – SUBRUTIN

Program :

Implementasi dalam C :

Kode Sumber C++ :**posstr.cpp**

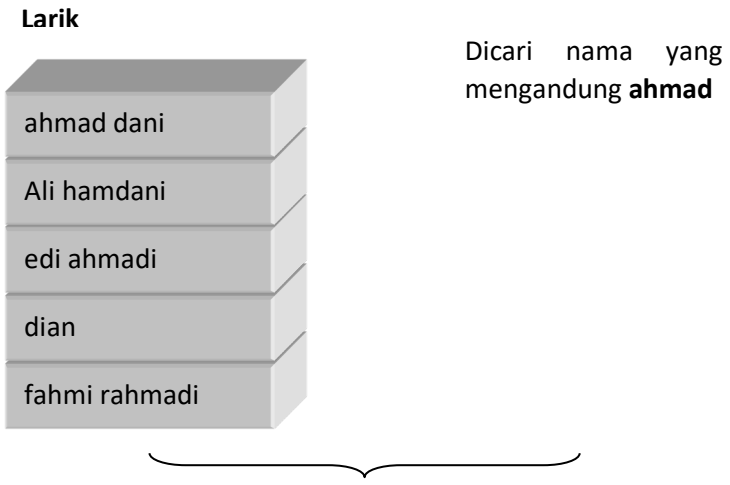
```
#include <iostream.h>
#include <string.h>

int posstr(char st1[], char st2[])
{
      int indeks1, indeks2, indeks3, ketemu, posisi;
      int panjang1 = strlen(st1);
      int panjang2 = strlen(st2);

      indeks1 = 0;
      posisi = -1;
      ketemu = 0;

      while ((indeks1 < panjang1) && !ketemu)
      {
            /* Jika panjang yang dicari lebih panjang dari
            sisa string untuk pencarian */

            if (panjang2 > (panjang1 - indeks1))
            break;

            indeks3 = indeks1;
            ketemu = 1;
```

```
            for (indeks2 = 0; indeks2 < panjang2; indeks2++)
            {
                  if (st1[indeks3] == st2[indeks2])
                        indeks3++;
                  else
                  {
                        ketemu = 0;
                        break;
                  }
            }

            if (ketemu)
                  posisi = indeks1;
            else
                  indeks1++;
      }

      return posisi;
}

int main()
{
      char kalimat[] = "Daun cemarapun berderai-derai";
      char kata1[] = "derai";
      char kata2[] = "Dedaunan";

      cout << posstr(kalimat, kata1) << "\n";
      cout << posstr(kalimat, kata2) << "\n";

      return 0;
}
```

Akhir Kode Sumber

Contoh 10.9 Terdapat larik yang mengandung n buah nama orang. Tulislah algoritma yang menampilkan semua nama orang yang mengandung penggalan nama tertentu.

Contoh :



Gambar 10.13 *pencarian menurut penggalan nama.*

Algoritma :

SUBRUTIN *carinama* (Data, n, Nama)

UNTUK i ← 0 S/D n-1

JIKA posstr (Data, Nama) ≠ -1 MAKA

tampilkan (Data[i])

AKHIR- JIKA

AKHIR – UNTUK

AKHIR- SUBRUTIN

Algoritma di atas memerlukan subrutin posstr yang telah dibahas didepan

Program :

Implementasi dalam C++:

Kode Sumber C++ :**carinama.cpp**

```
#include <iostream.h>
#include <string.h>

int posstr(char st1[], char st2[])
{
      int indeks1, indeks2, indeks3, ketemu, posisi;
      int panjang1 = strlen(st1);
      int panjang2 = strlen(st2);
      indeks1 = 0;
      posisi = -1;
      ketemu = 0;

      while ((indeks1 < panjang1) && !ketemu)
      {
            /* Jika panjang yang dicari lebih panjang dari
```

```
        sisa string untuk pencarian */

        if (panjang2 > (panjang1 - indeks1))
            break;

        indeks3 = indeks1;
        ketemu = 1;
        for (indeks2 = 0; indeks2 < panjang2; indeks2++)
        {
            if (st1[indeks3] == st2[indeks2])
                indeks3++;
            else
            {
                ketemu = 0;
                break;
            }
        }

        if (ketemu)
            posisi = indeks1;
        else
            indeks1++;
        }

        return posisi;
}
void carinama(char data[][35], int n, char nama[])
{
        int i;
        for (i = 0; i < n; i++)
            if (posstr(data[i], nama) != -1)
                cout << data[i] << "\n";
}

int main()
{
        char daftar_nama[][35] = {"ahmad dani",
                                  "ali hamdani",
                                  "edi ahmadi",
                                  "dian",
                                  "fahmi rahmadi"};
        carinama(daftar_nama, 5, "ahmad");
        return 0;
}
```

	Akhir Kode Sumber

BAB 11

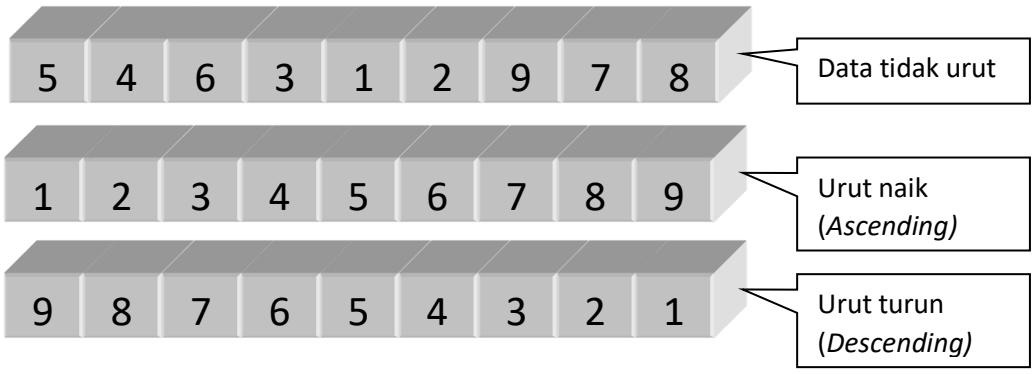
PENGURUTAN DATA

11.1 Pengantar Pengurutan Data

Proses pengurutan banyak ditemukan dalam komputer. Hal itu karena data yang sudah urut akan lebih cepat untuk dicari. untuk membentuk data yang tidak urut menjadi data yang urut, terdapat berbagai algoritma yang bisa digunakan. Beberapa algoritma akan dijelaskan pada bab ini.

Perlu diketahui bahwa pengurutan sendiri dapat dilakukan terhadap data yang secara keseluruhan diletakkan dalam memori ataupun terhadap data yang tersimpan pada pengingat eksternal. Pada bab ini pengurutan pada kategori pertama saja yang akan dibahas.

Di dalam pengurutan data terdapat istilah *ascending* dan *descending*. Pengurutan dengan dasar dari nilai yang kecil menuju ke nilai besar disebut aascending (urut naik), sedangkan yang disusun atas dasar dari nilai besar ke kecil disebut *descending* (urut turun).



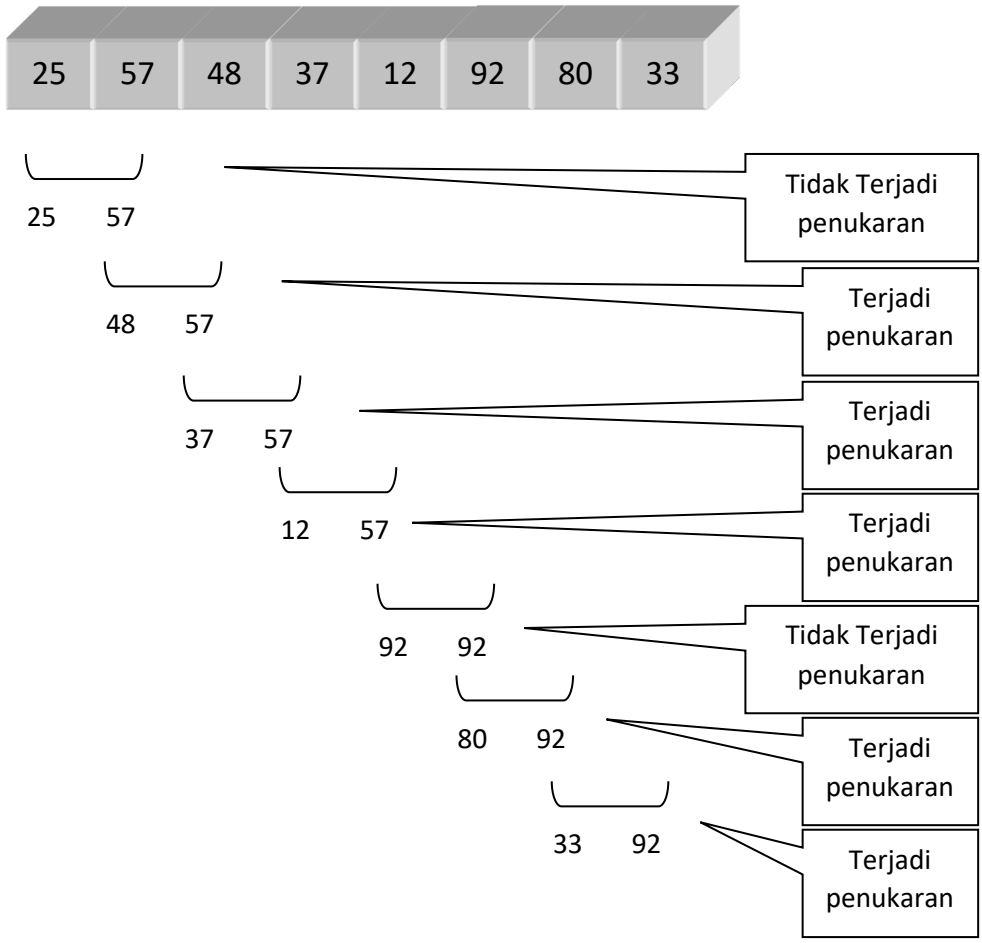
Gambar 11. 1*Pengurutan*

11.2 Metode Bubble Sort

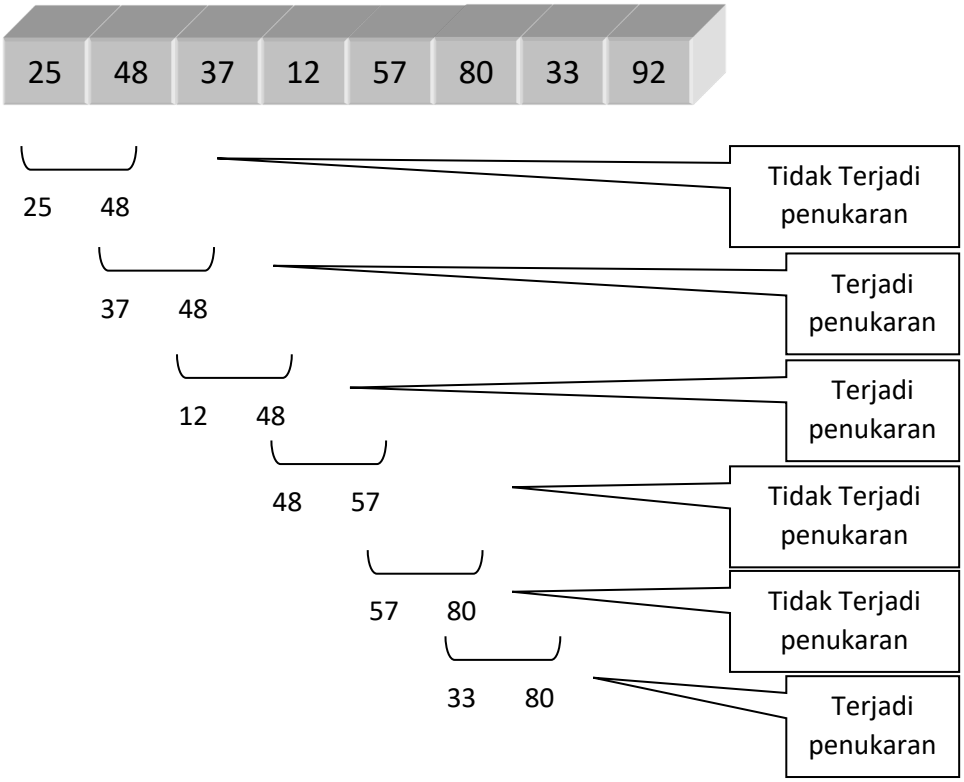
Metode *bubble sort* merupakan metode tersederhana untuk melakukan pengurutan data, tetapi memiliki kinerja yang terburuk untuk data yang besar. Pengurutan dilakukan dengan membandingkan sebuah bilangan dengan seluruh bilangan yang terletak sesudah bilangan tersebut. Penukaran dilakukan kalau suatu kriteria dipenuhi. Sebagai contoh, terdapat kumpulan seperti berikut.

25 57 48 37 12 92 80 33

Contoh proses pengurutan dengan urut naik ditunjukkan pada gambar 11.2 sampai dengan gambar 11.3



Gambar 11.2 *Pengurutan tahap pertama*



Gambar 11.3 *Pengurutan tahap kedua*

Jika jumlah data adalah n, maka terjadi n-1 tahap pengurutan. Berarti pada contoh di depan diperlukan 7 tahap pengurutan. Gambar 11.4 memperlihatkan keadaan setelah 7 tahap pengurutan dilakukan.

Awal	25	57	48	37	12	92	80	33
Tahap 1	25	48	37	12	57	80	33	92
Tahap 2	25	37	12	48	57	33	80	92
Tahap 3	25	12	37	48	33	57	80	92
Tahap 4	12	25	37	33	48	57	80	92
Tahap 5	12	25	33	37	48	57	80	92
Tahap 6	12	25	33	37	48	57	80	92
Tahap 7	12	25	33	37	48	57	80	92

Gambar 11.4 *Keadaan untuk setiap tahap pengurutan*

Contoh 11.1 implementasikan pengurutan dengan metode *bubble-sort* baik dalam bentuk algoritma maupun program.

Algoritma:

SUBRUTIN *buuble_sort(L, n)*

UNTUK tahap = 1 S/D n-1

UNTUK j ← 0 S/D n-tahap-1

JIKA L[j] > L[j+1] MAKA

//Lakukan penukaran

tmp ← L[j]

L[j] ← L[j+1]

L[j+1] ← tmp

AKHIR-JIKA

AKHIR-UNTUK

AKHIR-UNTUK

AKHIR-SUBRUTIN

Program:

Implementasi dalam C++:

Kode Sumber C++ :**bubble.cpp**

```
#include <iostream.h>

void tampilkan_larik(int data[], int n)
{
	int i;

	for (i = 0; i < n; i++)
		cout << data[i] << " ";

	cout << "\n";
}

void bubble_sort(int data[], int n)
{
	int tahap, j, tmp;

	for (tahap = 1; tahap < n; tahap++)
	{
		for (j = 0; j < n - tahap; j++)
```

```
                if (data[j] > data[j+1])
                {
                      /* Tukarkan */
                      tmp = data[j];
                      data[j] = data[j+1];
                      data[j+1] = tmp;
                }

            cout << "Hasil tahap " << tahap << ": ";
            tampilkan_larik(data, n);
      }
}

int main()
{
      const JUM_DATA = 8;

      int i;
      int data[] = {25, 57, 48, 37, 12, 92, 80, 33};

      bubble_sort(data, JUM_DATA);

      /* Hasil pengurutan */

      cout << "Hasil pengurutan:\n";

      tampilkan_larik(data, JUM_DATA);

      return 0;
}
```

Akhir Kode Sumber

Contoh 11.2 Pada contoh gambar 11.4 terlihat bahwa sebenarnya tidak diperlukan n-1 tahap untuk mengurutkan data. Pada tahap kelima daata sebenarnya sudah terurutkan. Oleh karena itu metode *bubble-sort* dapat diperbaiki agar begitu data sudah urut dan walaupun n-1 tahap belum terpenuhi, pengurutan segera diakhiri. Hal ini dapat dilaksanakan dengan melakukan pemeriksaan apakah masih ada penukaran data atau tidak. Kalau dalam suatu tahap ternyata tidak terjadi pertukaran data, maka iterasi segera dihentikan. Cobalah untuk mengimplementasikannya.

Algoritma:

SUBRUTIN *buuble_sort(L, n)*

```
        tahap ← 1
        ada_penukaran ← BENAR
        ULANG SELAMA tahap ≤ n-1 DAN ada_penukaran
            ada_penukaran ← SALAH
            UNTUK j ← 0 S/D n-tahap-1
                JIKA L[j] > L[j+1] MAKA
                    ada_penukaran ← BENAR

                    //Lakukan penukaran
                    tmp ← L[j]
                    L[j] ← L[j+1]
                    L[j+1] ← tmp
                AKHIR-JIKA
            AKHIR-UNTUK
        tahap ← tahap + 1
        AKHIR-ULANG
    AKHIR-SUBRUTIN
```

Program:

Implementasi dalam C++:

Kode Sumber C++ :**bubble2.cpp**

```
#include <iostream.h>

void tampilkan_larik(int data[], int n)
{
      int i;

      for (i = 0; i < n; i++)
            cout << data[i] << " ";

      cout << "\n";
}

void bubble_sort(int data[], int n)
{
      int tahap, j, tmp;
      int ada_penukaran;
```

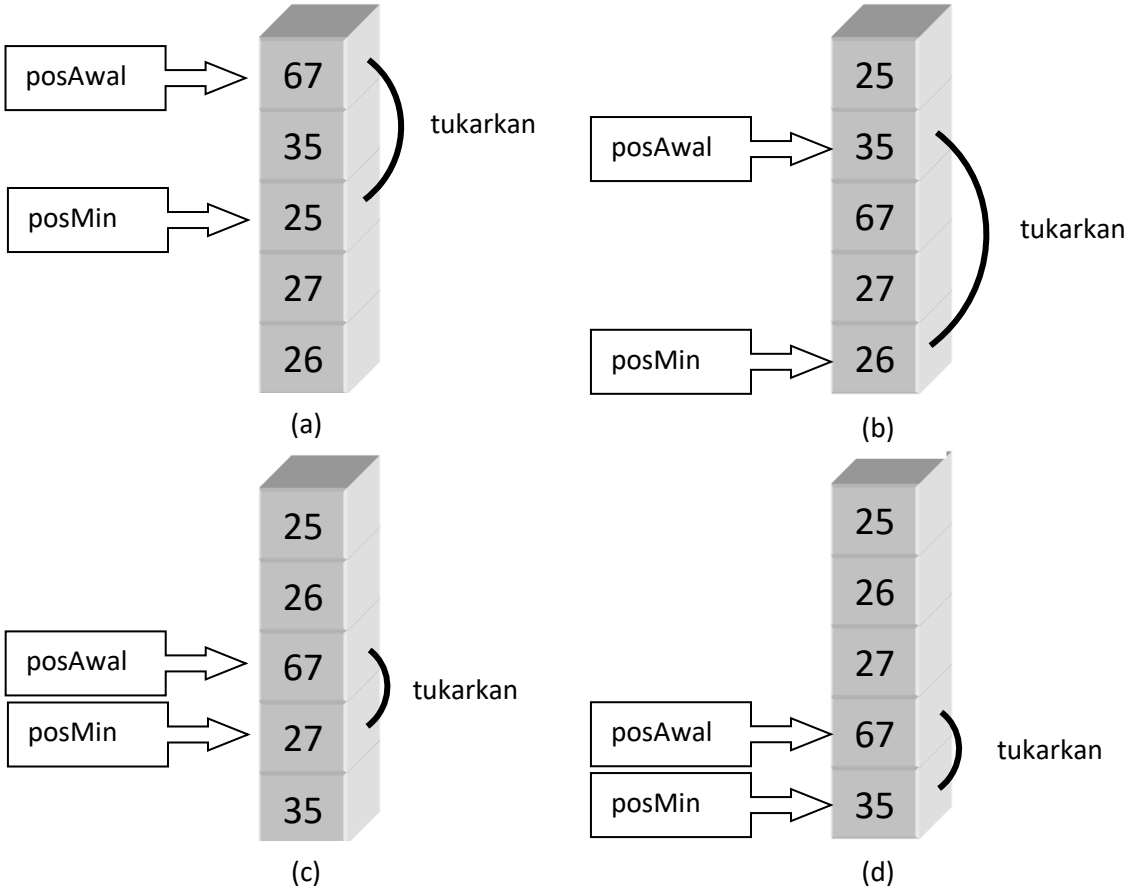
```
        tahap = 1;
        ada_penukaran = 1;

        while (tahap < n-1 && ada_penukaran)
        {
              ada_penukaran = 0;
              for (j = 0; j < n - tahap; j++)
                    if (data[j] > data[j+1])
                    {
                          ada_penukaran = 1;

                          /* Tukarkan */
                          tmp = data[j];
                          data[j] = data[j+1];
                          data[j+1] = tmp;
                    }

              cout << "Hasil tahap " << tahap << ": ";
              tampilkan_larik(data, n);

              tahap++;
        }
}

int main()
{
        const JUM_DATA = 8;

        int i;
        int data[] = {25, 57, 48, 37, 12, 92, 80, 33};

        bubble_sort(data, JUM_DATA);

        /* Hasil pengurutan */

        cout << "Hasil pengurutan:\n";
        tampilkan_larik(data, JUM_DATA);

        return 0;
}
```

Akhir Kode Sumber

11.3 Metode Pengurutan Seleksi

Pengurutan seleksi (*selection sort*) mempunyai mekanisme seperti berikut. Mula-mula suatu petunjuk (diberi nama `posawal`), yang menunjuk ke lokasi awal pengurutan data, diatur agar berisi indeks pertama dalam larik. Selanjutnya dicari

bilangan terkecil yang terletak antara posisi sesudah yang ditunjuk oleh penunjuk tersebut himgga elemen yang terakhir dalam larik. Lokasi bilangan ini ditunjuk oleh `posmin`. Lalu tukarlah nilai bilangan terkecil tersebut dengan nilai yang ditunjuk oleh `posawal`. Proses seperti itu diulang dari `posawal` bernilai 0 hingga *n*-2, dengan *n* menyatakan jumlah elemen dalam *larik*.



Hasil akhir: **25 26 27 67 35**

Gambar 11.7 *Contoh pengurutan seleksi*

Contoh 11.3 Cobalah untuk mengimplementasikan subrutin pengurutan seleksi baik dalam bentuk algoritma maupun program.

Algoritma:

```
SUBRUTIN selection_sort(L, n)
    UNTUK posawal = 0 S/D n-2
        posMin ← posAwal
        UNTUK j ← posAwal + 1 S/D n-1
            JIKA  L[posMin] > L[j] MAKA
```

posMin ← j

AKHIR-JIKA

AKHIR-UNTUK

//Tukarkan

tmp ← L[posAwal]

L[posAwal]← L[posMin]

L[posMin] ← tmp

AKHIR-UNTUK

AKHIR-SUBRUTIN

Program:

Implementasi dalam C++:

Kode Sumber C++ :**seleksi.cpp**

```
#include <iostream.h>

void tampilkan_larik(int data[], int n)
{
      int i;

      for (i = 0; i < n; i++)
            cout << data[i] << " ";

      cout << "\n";
}

void selection_sort(int data[], int n)
{
      int posMin, posAwal, j, tmp;

      for (posAwal = 0; posAwal < n-1; posAwal++)
      {
            posMin = posAwal;
            for (j = posAwal + 1; j < n; j++)
                  if (data[posMin] > data[j])
                        posMin = j;

            /* Tukarkan */
            tmp = data[posAwal];
            data[posAwal] = data[posMin];
            data[posMin] = tmp;

            cout << "Hasil posAwal= " << posAwal <<": ";
            tampilkan_larik(data, n);
```

```
        }
}

int main()
{
        const JUM_DATA = 8;

        int i;
        int data[] = {25, 57, 48, 37, 12, 92, 80, 33};

        selection_sort(data, JUM_DATA);

        /*
```

```
Hasil pengurutan */

        cout << "Hasil pengurutan:\n";

        tampilkan_larik(data, JUM_DATA);

        return 0;
}
```

Akhir Kode Sumber

11.4 Pengurutan dengan Penyisipan

Pengurutan dengan penyisipan (*insertion sort*) adalah suatu metode yang melakukan pengurutan dengan cara menyisipkan data yang belum urut ke dalam bagian data yang telah diurutkan. Konsep seperti ini biasa dilakukan pada permainan kartu. Ketika sebuah kartu baru didapatkan (hasil pembagian dari pengocokan kartu) kartu akan disisipkan oleh pemain pada posisi yang tepat sehingga penambahan kartu tersebut membuat semua kartu tetap terurutkan.

Gambar 11.9 *Mengurutkan kartudengan metode penyisipan. Kartu 7 disisipkan sehingga susunan kartu yang sebelumnya sudah urut tetap urut*

Contoh 11.4 Bila L adalah larik dengan n elemen, mula-mula L[0] (elemen pertama) dianggap sebagai kumpulan data yang telah diurutkan, yang terdiri atas 1 buah data. Kemudian dilakukan penyisipan data dari L[1] sampai dengan L[n-1] ke dalam kumpulan data dari L[0] sampai dengan L[k-1] dengan 1< k < n. Dalam hal ini penyisipan dilakukan pada tempat yang tepat sehingga data pada L[0] sampai dengan L[k] menjadi urut.

Algoritma:

```
SUBRUTIN selection_sort(L, n)
    UNTUK k ← 1 S/D n-1
        x ← L[k]
        // Sisipkan x ke dalam L[0...k-1]
        1 ← k-1
        ketemu ← SALAH
        ULANG SELAMA i ≥ 0 DAN TIDAK ketemu
            JIKA x < L[i] MAKA
                L[i+1] ← L[i]
                i ← i+1
            SEBALIKNYA
                ketemu = BENAR
            AKHIR-JIKA
            L[i+1] ← x
        AKHIR-ULANG
    AKHIR-UNTUK
```

AKHIR-SUBRUTIN

Program:

Implementasi dalam C++:

Kode Sumber C++ :**sisip.cpp**

```
#include <iostream.h>

void tampilkan_larik(int data[], int n)
{
      int i;

      for (i = 0; i < n; i++)
            cout << data[i] << " ";

      cout << "\n";
}

void insertion_sort(int data[], int n)
{
      int i, k;
      int x;
      int ketemu;

      for (k = 1; k < n; k++)
      {
            x = data[k];

            // Sisipkan x ke dalam data[0..k-1]
            i = k - 1;
            ketemu = 0;

            while ((i >= 0) && (!ketemu))
            {
                  if (x < data[i])
                  {
                        data[i+1] = data[i];
                        i = i - 1;
                  }
                  else
                        ketemu = 1;

                  data[i+1] = x;
            }
      }
}

int main()
```

```
{
      const JUM_DATA = 8;

      int i;
      int data[] = {25, 57, 48, 37, 12, 92, 80, 33};

      insertion_sort(data, JUM_DATA);

      // Hasil pengurutan

      cout << "Hasil pengurutan:\n";
      tampilkan_larik(data, JUM_DATA);

      return 0;
}
```

Akhir Kode Sumber

11.5 Pengurutan dengan Penyisipan Biner

Contoh 11.5 Penyisipan pada *insertion_sort* dapat dibuat lebih efisien mengingat penyisipan dilakukan pada data yang telah urut. Pencarian data dapat dilakukan dengan menggunakan metode pencarian biner yang telah dibahas pada Bab 11. Setelah posisi untuk penyisipan ditemukan, semua elemen yang berada disebelah kanan posisi tempat data akan disisipkan perlu digeser ke kanan.

Algoritma:

```
SUBRUTIN binary_sort(L, n)
    UNTUK k ← 1 S/D n-1
        x ← L[k]
        // Sisipkan x ke dalam L[0...k-1]
        kiri ← 0
        kanan ← k-1
    ULANG SELAMA (kiri ≤ kanan)
            tengah ← (kiri + kanan) / 2      // Pembagian bulat
            JIKA x < L[tengah] MAKA
                kanan ← tengah + 1
            SEBALIKNYA
                kiri ← tengah + 1
            AKHIR-JIKA
```

// Melakukan penggeseran

UNTUK j ← k-1 S/D kiri

L[j+1] ← L[j]

AKHIR-UNTUK

L[kiri] ← x

AKHIR-ULANG

AKHIR-UNTUK

AKHIR-SUBRUTIN

Program:

Implementasi dalam C++:

Kode Sumber C++ :**binary.cpp**

```
#include <iostream.h>

void tampilkan_larik(int data[], int n)
{
      int i;

      for (i = 0; i < n; i++)
            cout << data[i] << " ";

      cout << "\n";
}

void binary_insertion(int data[], int n)
{
      int j, k;
      int x;
      int kiri, kanan, tengah;

      for (k = 1; k < n; k++)
      {
            x = data[k];

            // Sisipkan x ke dalam data[0..k-1]
            kiri = 0;
            kanan = k - 1;

            while (kiri <= kanan)
            {
                  // Pencarian biner
                  tengah = (kiri + kanan) / 2;
                  if (x < data[tengah])
```

```
                    kanan = tengah - 1;
               else
                    kiri = tengah + 1;
          }

          // Melakukan penggeseran
          for (j = k - 1; j >= kiri; j--)
               data[j+1] = data[j];

          // Tempatkan x ke data[kiri]
          data[kiri] = x;
     }
}

int main()
{
     const JUM_DATA = 8;

     int i;
     int data[] = {25, 57, 48, 37, 12, 92, 80, 33};

     binary_insertion(data, JUM_DATA);

     // Hasil pengurutan

     cout << "Hasil pengurutan:\n";
     tampilkan_larik(data, JUM_DATA);

     return 0;
}
```

Akhir Kode Sumber

11.6 Metode Quick Sort

Quick sort adalah metode pengurutan data yang dikemukakan pertama kali oleh C. AR Hoare pada tahun 1962. Metode ini menggunakan strategi “pecah-pecah” dengan mekanisme seperti berikut: Larik L[p...r] (dengan indeks terkecil adalah p dan indeks terbesar yaitu r) disusun ulang (dipartisi) menjadi dua buah larik A[p...q] dan A[q+1...r] sehingga setiap elemen dalam A[p...q] selalu bernilai lebih kecil daripada setiap nilai elemen pada A[q+1...r]. Selanjutnya kedua larik tersebut diurutkan secara rekursif. Dengan sendirinya kombinasi kedua larik tersebut membentuk larik dengan data yang tellah urut.

Contoh 11.6 implementasi *quick sort* dapat dilihat di bawah ini.

Algoritma:

```
SUBRUTIN quick_sort(L, p, r]
    JIKA p < r MAKA
        q ← partisi(L, p, r)
            quick_sort(L, p, r)
            quick_sort(L,q+1,r)
    AKHIR-JIKA
AKHIR-SUBRUTIN
```

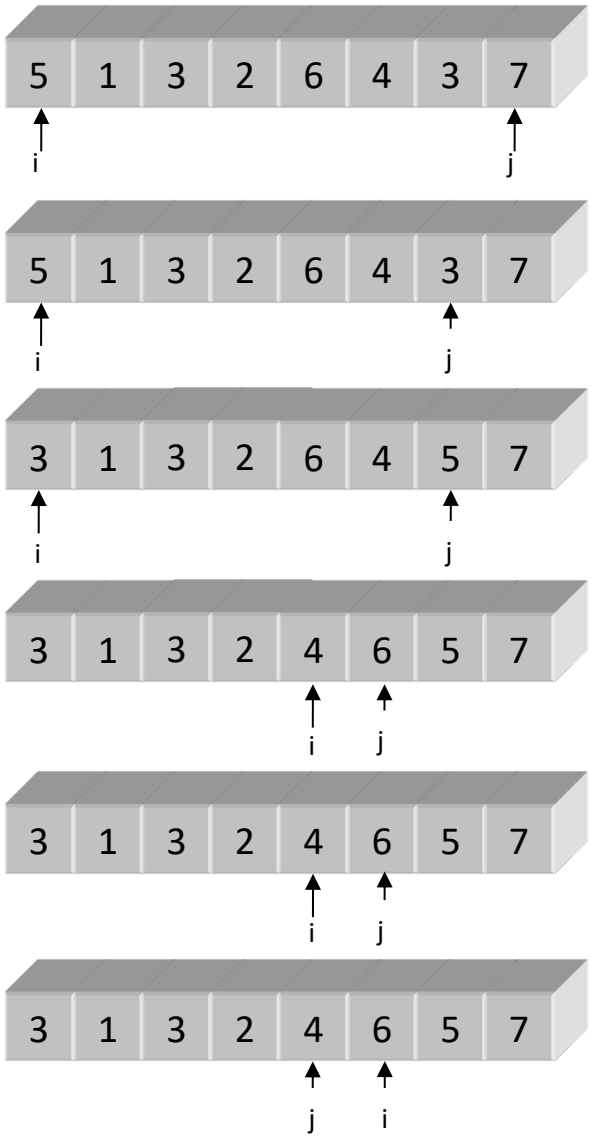
Untuk mengurutkan isi keseluruhan larik L, diperlukan pemanggilan seperti berikut:

```
quick_sort(L, 0, jumlah_elemen(L)-1)
```

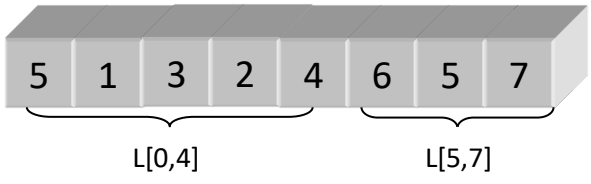
Subrutin `partisi` sendiri seperti berikut:

```
SUBRUTIN partisi(L, p, r)
    x ← L[p]
    i ← p
    j ← r
    ULANG SELAMA BENAR
        ULANG SELAMA L[j] > x
            j ← j – 1
        AKHIR-ULANG
        ULANG SELAMA L[i] < x
            i ← i + 1
        AKHIR-ULANG
        JIKA i < j MAKA
            // Tukarkan L[i] dengan L[j]
            tmp = L[i]
        L[i] ← L[j]
            L[i] ← tmp
        SEBALIKNYA
            NILAI-BALIK j
        AKHIR-JIKA
    AKHIR-ULANG
AKHIR-SUBRUTIN
```

Pertama-tama x ← L[p] dipakai untuk membagi larik L[p...r] menjadi dua bagian (disebut pemartisian) dengan kondisi semua elemen bagian kiri selalu lebih kecil daripada nilai elemen pivot dan nilai semua elemen bagian kanan selalu lebih kecil daripada nilai elemen pivot. Pemartisian dilakukan dengan menggunakan variabel i dan j. Dalam hal ini i berupa penunjuk yang bergerak naik, sedangkan j adalah penunjuk yang bergerak turun. Variabel j digeser turun secara terus-menerus sehingga L[j] ≤ elemen pivot, sedangkan i digeser naik secara terus-menerus sampai memenuhi kondisi L[j] ≥ elemen pivot. Proses pengulangan dilakukan sampai nilai i ≥ j. Pada keadaan seperti ini nilai balik subrutin `partisi` berupa j. Gambar 11.12 memberikan contoh pemartisian larik menjadi dua bagian.



Mengingat sekarang i < j tidak lagi berlaku, maka nilai balik berupa j. Dengan demikia pembagian larik adalah seperti berikut



Gambar 11.12 *Ilustrasi pemartisian larik*

Bila masing-masing bagian dikenakan operasi pemartisian, akhirnya akan diperoleh data yang urut.

Program:

Implementasi dalam C++:

Kode Sumber C++ :**qsort.cpp**

```
#include <iostream.h>

void tampilkan_larik(int data[], int n)
{
      int i;

      for (i = 0; i < n; i++)
            cout << data[i] << " ";

      cout << "\n";
}

int partisi(int data[], int p, int r)
{
      int x, i, j, tmp;

      x = data[p];
      i = p;
      j = r;

      while (1)
      {
            while (data[j] > x)
                  j = j - 1;

            while (data[i] < x)
                  i = i + 1;

            if (i < j)
            {
                  // Tukarkan data
                  tmp = data[i];
                  data[i] = data[j];
```

```
                data[j] = tmp;
            }
            else
                return j;
        }
    }

    void quick_sort(int data[], int p, int r)
    {
        int q;

        if (p < r)
        {
            q = partisi(data, p, r);
            quick_sort(data, p, q);
            quick_sort(data, q+1, r);
        }
    }

    int main()
    {
        const JUM_DATA = 9;

        int i;
        int data[] = {25, 57, 48, 37, 12, 92, 80, 33, 1};

        quick_sort(data, 0,JUM_DATA-1);

        // Hasil pengurutan

        cout << "Hasil pengurutan:\n";
        tampilkan_larik(data, JUM_DATA);

        return 0;
    }
```

Akhir Kode Sumber

KURIKULUM (KKNI) PROGRAM STUDI TEKNIK INFORMATIKA

SEMESTER 1

NO	KODE	MATA KULIAH	SKS
1	PKUN02	Al Islam Kemuhammadiyahan 1	2
2	PKUN07	Ilmu Alamiah Sosial Budaya Dasar	2
3	BBUN01	Pancasila dan Kewarganegaraan	2
4	PKUN01	Bahasa Indonesia	2
5	PKUN08	Bahasa Inggris	2
6	KKFT02	Fisika & Elektronika	3
7	KKFT03	Kalkulus 1	3
8	KKNIIF01	Pengantar Teknologi Informasi	2
9	PKUN06	Filsafat Ilmu	2
JUMLAH			20

SEMESTER 2

NO	KODE	MATA KULIAH	SKS
1	PKUN03	Al Islam Kemuhammadiyahan 2	2
2	PKUN09	Bahasa Inggris Teknik	2
3	KKFT04	Kalkulus 2	3
4	KKFT06	Statistik & Probabilitas	2
5	KKNIIF02	Struktur Data dan Algoritma	3
6	KKNIIF03	Sistem Digital	3
7	KKNIIF04	Algoritma Pemrograman	3
8	KKNIIF05	Sistem Operasi	2
JUMLAH			20

SEMESTER 3

NO	KODE	MATA KULIAH	SKS
1	PKUN04	Al Islam Kemuhammadiyahan 3	2
2	KKNIIF06	Matematika Diskrit	3
3	KKNIIF11	Pemrograman Dasar	3
4	KKNIIF07	Arsitektur & Organisasi Komputer	3
5	KKNIIF08	Desain Grafis	3
6	KKNIIF18	Basis Data 1	2
7	KKNIIF09	Interaksi Manusia dan Komputer	2
8	KKNIIF10	Metode Numerik	3
JUMLAH			21

SEMESTER 4

NO	KODE	MATA KULIAH	SKS
1	PKUN05	Al Islam Kemuhammadiyahan 4	2
2	KKNIIF12	Pemrograman Berorientasi Objek	3
3	KKNIIF22	Jaringan Komputer 1	2
4	KKNIIF19	Basis Data 2	3
5	KKNIIF20	Etika Profesi & Organisasi TI	2
6	KKNIIF21	Komunikasi Data	2
7	KKNIIF14	Pemrograman Java Dasar	3
8	KKNIIF13	Pemrograman Web Dasar	3
JUMLAH			20

PRAKTIKUM BERBASIS KOMPETENSI

NO	KODE	MATA KULIAH
1	KKFT02P	Praktikum Fisika & Elektronika
2	KKNIIF19	Praktikum Basis Data
3	KKNIIF15P	Praktikum Web Programming
4	KKNIIF16P	Praktikum Java Programming
5	KKNIIF23P	Praktikum Jaringan Komputer

SEMESTER 5

NO	KODE	MATA KULIAH	SKS
1	PKUN10	Al Islam Kemuhammadiyahan 5	2
2	KKFT07	Metodologi Penelitian	2
3	KKNIIF23	Jaringan Komputer 2	2
4	KKNIIF15	Pemrograman Web Lanjutan	3
5	KKNIIF16	Pemrograman Java Lanjutan	3
6	KKNIIF25	Rekayasa Perangkat Lunak	3
7	KKNIIF24	Keamanan Sistem Komputer	3
8	KKNIIF26	Technopreneurship	2
9	KKNIIF27	e-bisnis	2
JUMLAH			22

SEMESTER 6

NO	KODE	MATA KULIAH	SKS
1	BBFT02	Kerja Praktek	4
2	KKNIIF28	Manajemen Proyek TI	3
3	KKNIIF29	Datawarehouse & Bisnis Intelligence	3
4	KKNIIF17	Pemrograman Sistem	3
5	KKNIIF30	Analisis & Perancangan Sistem	3
6	KKNIIF31	Sistem Terdistribusi	2
7	KKNIIF32	Sistem Informasi Manajemen	3
8			
JUMLAH			21

SEMESTER 7

NO	KODE	MATA KULIAH	SKS
1	BBUN02	Kuliah Kerja Nyata (KKN)	4
2	-	Mata Kuliah Pilihan 1	3
3	-	Mata Kuliah Pilihan 2	3
4	-	Mata Kuliah Pilihan 3	3
5	-	Mata Kuliah Pilihan 4	3
6	-	Mata Kuliah Pilihan 5	3
7	-	Mata Kuliah Pilihan 6	3
8	-	Mata Kuliah Pilihan 7	3
JUMLAH			25

SEMESTER 8

NO	KODE	MATA KULIAH	SKS
1	KBFT03	Tugas Akhir	6
JUMLAH			6

MATA KULIAH PILIHAN

NO	KODE	MATA KULIAH	SKS
1	KKNIIF33	Tata Kelola IT	3
2	KKNIIF34	Kecerdasan Tiruan	3
3	KKNIIF35	Mobile Programming	3
4	KKNIIF36	Sistem Pendukung Keputusan	3
5	KKNIIF37	Verifikasi & Validasi Perangkat Lunak	3
6	KKNIIF38	ERP	3
7	KKNIIF39	Cloud Computing	3
8	KKNIIF40	Sekuritas Jaringan	3
9	KKNIIF41	Analisa Jaringan dan Optimasi	3
10	KKNIIF42	Jaringan Nirkabel	3
11	KKNIIF43	Manajemen Jaringan	3
12	KKNIIF44	Teknologi Antar Jaringan	3
13	KKNIIF45	Linux Administrator	3
14	KKNIIF46	Sistem Administrasi Server	3