

ALGORITMA & PEMROGRAMAN (C++)



Disusun sebagai modul mata kuliah Algoritma dan Pemrograman
pada Semester Gasal 2016/2017

**PROGRAM STUDI KOMPUTERISASI AKUNTANSI
AMIK BSI PONTIANAK
2016**

KATA PENGANTAR

Puji syukur penulis panjatkan kehadiran Allah SWT, yang telah memberikan rahmat dan hidayahnya sehingga modul **praktikum Algoritma Pemrograman** ini dapat terselesaikan dengan baik. Selanjutnya modul ini disusun untuk memberikan gambaran bagi mahasiswa yang mempelajari program **Algoritma Pemrograman** dengan menggunakan metode cepat karena modul ini disertai contoh kasus, sehingga lebih memudahkan mahasiswa dalam memahami bahasa pemrograman.

Tak lupa penulis mengucapkan banyak terima kasih kepada semua pihak yang telah membantu dengan tenaga dan pikirannya, terima kasih juga kepada rekan-rekan instruktur, dosen dan semuanya yang tidak bisa disebutkan satu persatu, yang selalu mendukung penulis sehingga modul ini sehingga dapat selesai sesuai yang kita inginkan semua.

Penulis menyadari masih banyak kekurangan dalam penyusunan modul ini. Untuk itu saran dan kritik yang membangun sangat penulis harapkan guna perbaikan dan pengembangan modul ini kedepan.

Akhir kata penulis berharap semoga modul praktikum **Algoritma Pemrograman** ini dapat dipergunakan sebaik-baiknya dan dapat dijadikan referensi untuk mahasiswa umum yang ingin mempelajarinya.

Pontianak, September 2016

Penyusun

DAFTAR ISI

Cover	1
Kata Pengantar	2
Daftar isi	3
Pertemuan 1 Pengenalan Bahas C++	5
Pertemuan 2 Tipe Data, Variabel dan Konstanta, Fungsi masukan dan Fungsi keluaran	9
Pertemuan 3 Operator Bahasa C++ dan Fungsi manipulasi string dan Konversi String	10
Pertemuan 4 Penyeleksian Kondisi	12
Pertemuan 5 Perulangan	18
Pertemuan 6 Array	25
Pertemuan 7 Pointer da File Header	28
Pertemuan 8 <i>UTS</i>	29
Pertemuan 9 <i>Fungsi</i>	32
Pertemuan 10 Structure	40
Pertemuan 11 OOP dan Karakteristik OOP	44
Pertemuan 12 Struktur Enum Union Bit-Field dan Typedef	49
Pertemuan 13 Rekursi	50
Pertemuan 14 Review Materi	51
Pertemuan 15-16 Presentasi UAS Project	93

Pengenalan Bahasa C++



1.1. Sejarah Singkat

1.1.1. Sekilas Perkembangan Bahasa C

Bahasa C dikembangkan di Bell lab pada tahun 1972 ditulis pertama kali oleh Brian W. Kernighan dan Denies M. Ricthie merupakan bahasa turunan atau pengembangan dari bahasa B yang ditulis oleh Ken Thompson pada tahun 1970 yang diturunkan oleh bahasa sebelumnya, yaitu BCL. Bahasa C, pada awalnya dirancang sebagai bahasa pemrograman yang dioperasikan pada sistem operasi UNIX.

Bahasa C merupakan bahasa pemrograman tingkat menengah yaitu diantara bahasa tingkat rendah dan tingkat tinggi yang biasa disebut dengan **Bahasa Tingkat Menengah**. Bahasa C mempunyai banyak kemampuan yang sering digunakan diantaranya kemampuan untuk membuat perangkat lunak, misalnya dBASE, Word Star dan lain-lain.

1.1.2. Sekilas Tentang C++

Pada tahun 1980 seorang ahli yang bernama Bjarne Stroustrup mengembangkan beberapa hal dari bahasa C yang dinamakan "C with Classes" yang pada mulanya disebut "a better C" dan berganti nama pada tahun 1983 menjadi C++ oleh Rick Mascitti, dibuat di Laboratorium Bell, AT&T.

Pada C++ ditambahkan konsep-konsep baru seperti class dengan sifat-sifatnya yang disebut dengan Object Oriented Programming (OOP), yang mempunyai tujuan utamanya adalah membantu dan mengelola program yang besar dan kompleks.

1.1.3. Perbedaan Antara Bahasa C Dengan C++

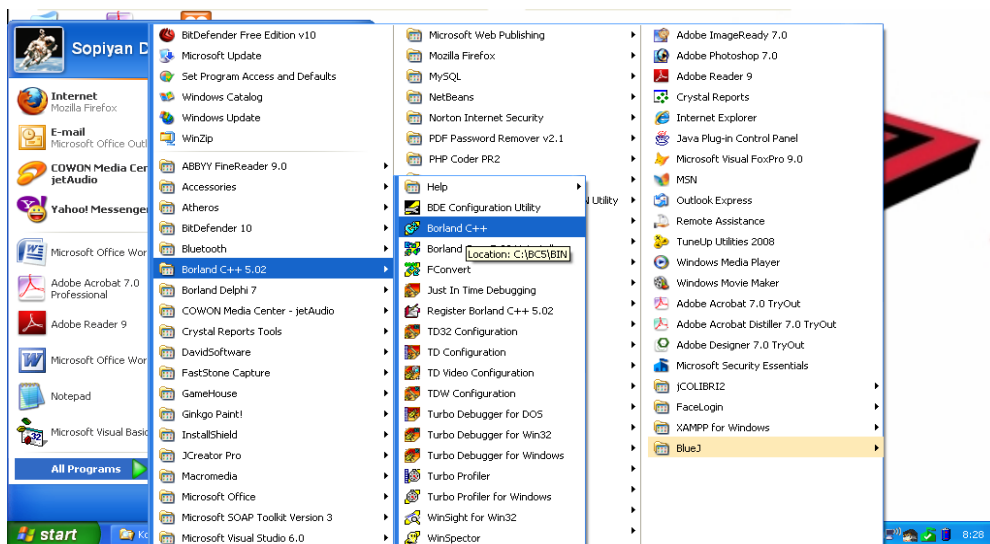
Bahasa C merupakan bahasa pemrograman prosedural, di mana penyelesaian atas suatu masalah dilakukan dengan membagi-bagi masalah tersebut ke dalam sub-sub masalah yang lebih kecil. Sedangkan C++ merupakan bahasa pemrograman yang memiliki sifat Object Oriented Programming (OOP). Untuk menyelesaikan masalah, C++ melakukan langkah pertama dengan mendefinisikan class-class yang merupakan a.-class yang dibuat sebelumnya sebagai abstraksi dari objek-objek fisik. Class tersebut berisi keadaan objek, anggota-anggotanya, dan kemampuan dari objeknya. Setelah beberapa class dibuat, masalah dipecahkan menggunakan class.

1.2. Pengenalan IDE Borland C++

IDE merupakan singkatan dari Integrated Development Environment, merupakan Lembar kerja terpadu untuk pengembangan program. IDE dari Borland C++, dapat digunakan untuk:

1. Menulis Naskah Program.
2. Mengkompilasi Program (*Compile*)
3. Melakukan Pengujian Program (*Debugging*)
4. Mengaitkan Object dan Library ke Program (*Linking*)
5. Menjalankan Program (*Running*)

Untuk mengaktifkan aplikasi Borland C++, lakukanlah langkah-langkah berikut ini:
Klik tombol Start □ pilih Program □ Borland C++ 5.02 □ klik Borland C++

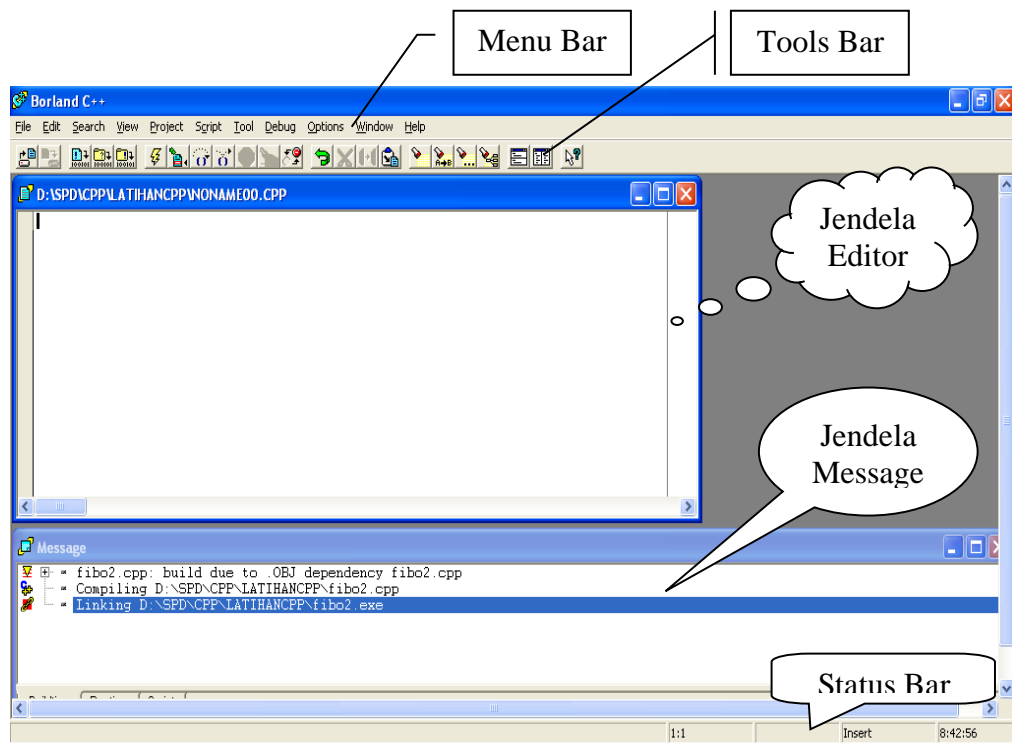


Gambar 1.1.
Menu Untuk Mengaktifkan Program Borland C++

Berikut IDE dari Borland C++, seperti gambar dibawah ini:



Gambar 1.2.
Layar Pembuka Borland C++



Gambar 1.3.
IDE Borland C++ 5.02

IDE pada Borland C++, terbagi menjadi 4 (empat) bagian, yaitu:

a. Baris Menu (Menu Bar)

Menu utama terdiri dari; File, Edit, Search, Run, Compile, Debug, Project, Options, Window dan Help.

b. Baris Peralatan (Tools Bar)

Baris yang menampilkan shortcuts (icons) untuk mempermudah pengguna dalam pembuatan program-program C++, seperti icon open, save, compiler, run dan lain-lain.

c. Jendela Editor

Tempat untuk pengetikan program dan membuat program. Jika pertama kali anda membuat program, nama file jendela editor adalah NONAME00.CPP

d. Jendela Message

Tempat untuk menampilkan pesan-pesan pada proses kompilasi dan link program. Jika ada kesalahan syntax program maupun varibel dan objek, maka akan diberikan pesan kesalahannya yang kemudian dapat didouble klik pada pesan tersebut untuk mendapatkan petunjuk di baris yang mana terdapat kesalahannya.

e. Baris Status (Status Bar)

Baris yang akan menampilkan keterangan-keterangan pada saat mengaktifkan menu bar dan sub menu serta keterangan-keterangan lain (seperti petunjuk baris dan kolom, waktu yang sedang berjalan).

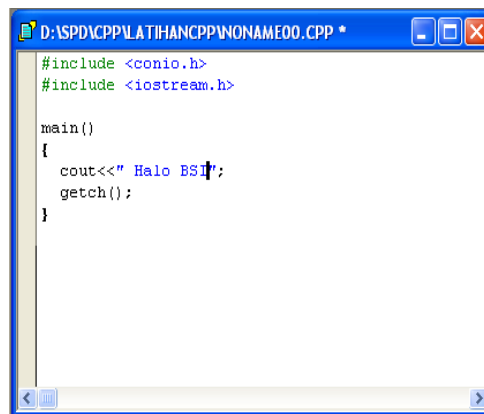
1.3. Struktur Program C++

Struktur program C++, sama seperti struktur program C yang terdahulu. Struktur program C++ terdiri sejumlah blok fungsi, setiap fungsi terdiri dari satu atau beberapa pernyataan yang melaksanakan tugas tertentu.

Bentuk Umum:

```
#include <file-header>
main()
{
    pernyataan;
}
```

Contoh-1



```
D:\ISPD\CPP\LATIHAN\CPP\NOMAME00.CPP *
#include <conio.h>
#include <iostream.h>

main()
{
    cout<<" Halo BS!";
    getch();
}
```

Penjelasan :

1. **#include <file-header>** merupakan preprocessor pada C++ untuk pemanggilan file header yang memuat beberapa perintah-perintah dari C++ (contoh, apabila ingin menggunakan perintah cout maka harus menggunakan file header iostream.h)
2. **main()** merupakan awal mula dari blok program utama
3. **tanda { dan }** sebagai punctuator yang awal blok program hingga akhir blok program
4. **cout** merupakan perintah keluaran pada C++
5. **getch();** apabila ditempatkan sebelum funtuator }, maka berfungsi sebagai penahan dari tampilan hasil

1.4. Model Memori

Borland C++, mempunyai enam model memori untuk program dan data. Model-model memori tersebut adalah:

a. Model Tiny

Model memori yang menyediakan jumlah memori untuk program dan data tidak lebih dari 64 Kb.

b. Model Small

Model memori yang menyediakan jumlah memori untuk masing-masing program dan data tidak lebih dari 64 Kb.

c. Model Medium

Model memori yang menyediakan jumlah memori untuk program tidak lebih dari 64 Kb dan data tidak lebih dari 64 K.

d. Model Compact

Model memori yang menyediakan jumlah memori untuk program lebih dari 64 Kb dan data tidak lebih dari 64 K.

e. Model Large

Model memori yang menyediakan jumlah memori untuk program dan data lebih dari 64 K.

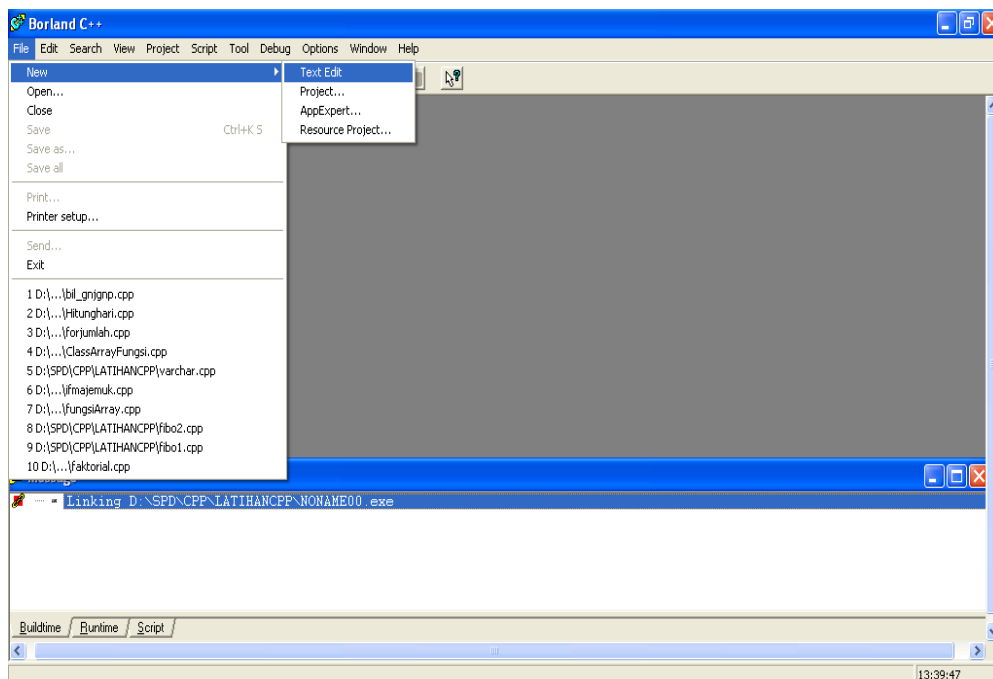
d. Model Huge

Model memori yang menyediakan jumlah memori untuk menyimpan satu jenis data.

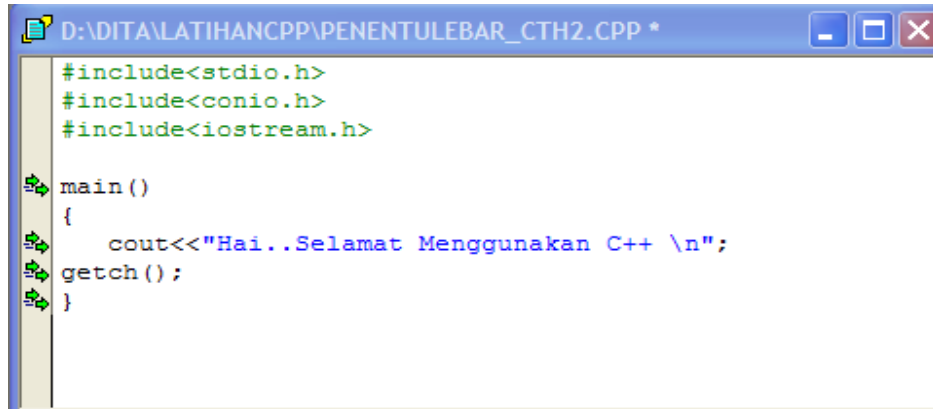
1.5. Membuat File Editor

File Editor merupakan file kode program yang dapat dikompilasi, kemudian dijalankan untuk menampilkan hasilnya yang mempunyai ekstensi file **.CPP**.

Cara mengaktifkannya : Klik Menu File Klik New Text Edit



Gambar 1.5 Cara Menampilkan Text Edit



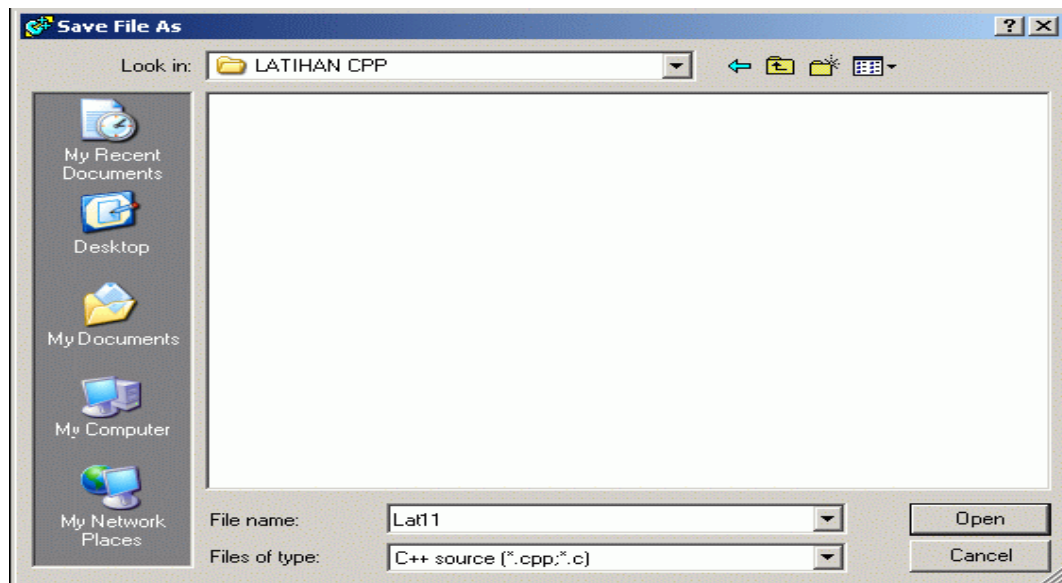
Gambar 1.6 Jendela Text Edit

1.6. Menyimpan File Editor

Setelah selesai mengetikkan naskah program yang baru pada jendela Text Edit, maka selanjutnya disimpan dengan cara :

- a. Kik Menu **F**ile **S**ave
- b. Menekan HotKey **Ctrl + KS**.

Selanjutnya tampil jendela Save File As, seperti dibawah ini :



Gambar 1.7. Jendela Save File As

Pada Borland C++ 5.02 terdapat tiga cara menyimpan file editor, diantaranya yaitu :

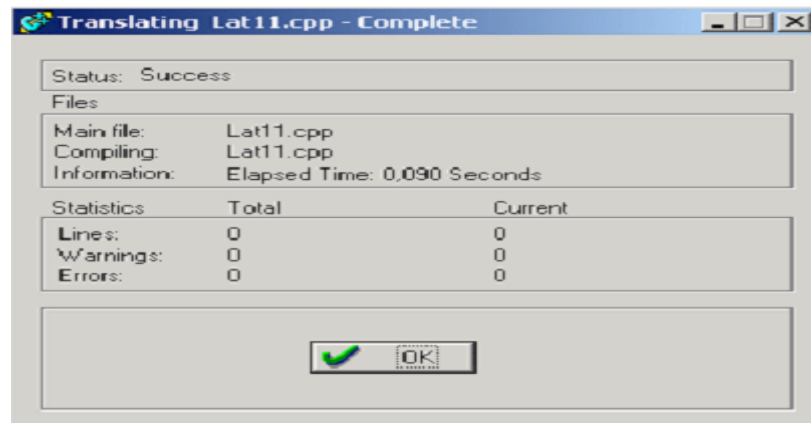
- Save** Digunakan untuk menyimpan File Program pada jendela yang sedang aktif kedalam disk. Hotkey yang ada bisa gunakan untuk menyimpan dengan menekan tombol **Ctrl + KS**.
- Save As** Digunakan untuk menyimpan File Program pada jendela yang sedang aktif kedalam disk dengan nama file yang berbeda.
- Save All** Digunakan untuk menyimpan semua File Program pada jendela yang sedang aktif kedalam disk.

1.7. Menterjemahkan Program

Proses Compile merupakan suatu proses menterjemahkan program dari bahasa manusia kedalam bahasa yang dimengerti oleh komputer yaitu bahasa mesin, yaitu dengan cara :

- Kik Menu **Debug** → **Compile**
- Menekan HotKey Alt + F9

Selanjutnya tampil kotak dialog Compile, seperti dibawah ini :



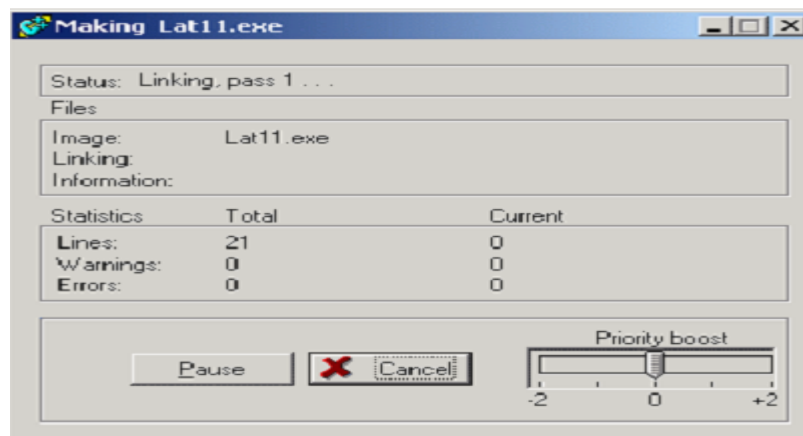
Gambar 1.8 Kotak Dialog Compile

1.8. Menjalankan Program

Proses Run merupakan suatu proses menterjemahkan program, melakukan proses linking, membuat file eksekusi (.exe) dan sekaligus menjalankan program, yaitu dengan cara:

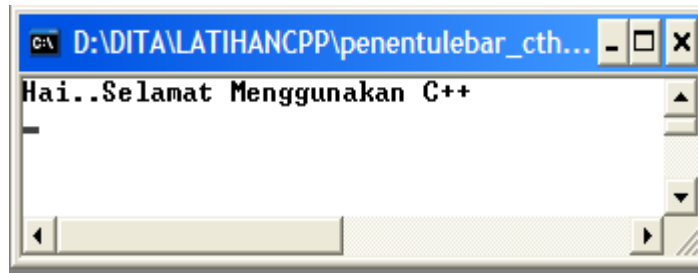
- Kik Menu **Debug** → **Run**
- Menekan HotKey Ctrl + F9

Selanjutnya tampil kotak dialog Run, seperti dibawah ini :



Gambar 1.9 Kotak Dialog Run

Setelah proses menterjemahkan program, proses linking, selanjutnya tampil hasil seperti gambar 1.10 dibawah ini :

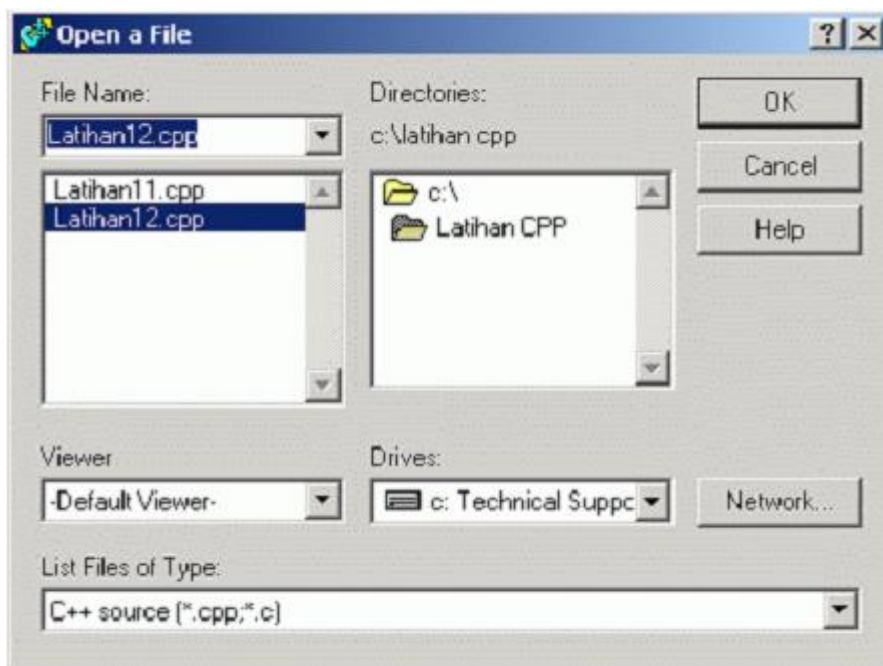


Gambar 1.10 Contoh Hasil Keluaran Program

1.9. Membuka File Editor

Penjelasan Membuka atau memanggil file editor yang sudah pernah dibuat, dengan cara : Klik Menu **File** → **Open**

Selanjutnya tampil Jendela Open, seperti dibawah ini :

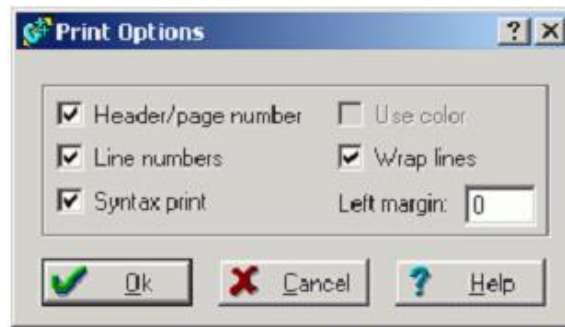


Gambar 1.11 Jendela Open File pada C++

1.10. Mencetak File Editor

Penjelasan Mencetak file program pada jendela yang sedang aktif dengan cara Klik **File** → **Print**

Selanjutnya tampil Jendela Print Option, seperti dibawah ini :



Gambar 1.12 Jendela Print Option

1.11. Keluar dari Borland C++ 5.02

Keluar dari Aplikasi Borland C++ 5.02, dengan cara **File** → **Exit**

Tipe Data, Variabel Konstanta, Fungsi Masukan Dan Fungsi Keluaran



2.1. Pengenalan Tipe Data

Borland C++ memiliki 7 tipe data dasar dan 3 tipe tambahan, diantaranya:

Tabel 2.1. Tipe Data

Tipe Data	Ukuran Memori	Jangkauan Nilai	Jumlah Digit
Char	1 Byte	-128 s.d 127	
Int	2 Byte	-2,147,435,648 s.d 2,147,435,647	
Short	2 Byte	-32768 s.d 32767	
Long	4 Byte	-2,147,435,648 s.d 2,147,435,647	
Float	4 Byte	3.4×10^{-38} s.d $3.4 \times 10^{+38}$	5 – 7
Double	8 Byte	1.7×10^{-308} s.d $1.7 \times 10^{+308}$	15 – 16
Long Double	10 Byte	3.4×10^{-4932} s.d $1.1 \times 10^{+4932}$	19

Tipe Data Tambahan, yang dimiliki oleh Borland C++, adalah :
Unsigned digunakan bila data yang digunakan hanya data yang positif saja.

Tabel 2.2. Tipe Data Tambahan Tambahan

Tipe Data	Jumlah Memori	Jangkauan Nilai
Unsigned Integer	2 Byte	0 – 65535
Unsigned Character	1 Byte	0 – 255
Unsigned Long Integer	4 Byte	0 – 4,294,967,295

2.2. Konstanta

Konstanta adalah suatu nilai yang sifatnya tetap. Secara garis besar konstanta dapat dibagi menjadi dua bagian, yaitu:

- Konstanta Bilangan
- Konstanta Teks

A. Konstanta Bilangan

Dalam hal ini konstanta bilangan dibagi menjadi tiga kelompok, antara lain:

1. Konstanta Bilangan Bulat (*Integer*).
Adalah bilangan yang tidak mengandung nilai desimal. Ini merupakan nilai default pada konstanta bilangan.
Contoh : 1, 2, 3, 100
2. Konstanta Desimal Berpresisi Tunggal (*Floating Point*)
Konstanta Floating Point, mempunyai bentuk penulisan, yaitu :
 - Bentuk Desimal (contoh : 5.57)
 - Bentuk Eksponensial / Bilangan Berpangkat (contoh : $4.22e3 \rightarrow 4.22 \times 10^3$)s
3. Konstanta Desimal Berpresisi Ganda (*Double Precision*)
Konstanta Double Precision, pada prinsipnya sama seperti Konstanta Floating Point, tetapi Konstanta Double Precision mempunyai daya tampung data lebih besar.

B. Konstanta Teks

Dalam hal ini konstanta teks dibagi menjadi dua kelompok, antara lain;

1. Data Karakter (*Character*).
Data karakter hanya terdiri dari sebuah karakter saja yang diapit oleh tanda kutip tunggal ('). Data karakter dapat berbentuk abjad (*huruf besar atau kecil*), angka, notasi atau simbol.
Contoh : 'Y' 'y' '9' '&' dan lain-lain.
2. Data Teks (*String*).
Data String merupakan rangkaian dari beberapa karakter yang diapit oleh tanda kutip ganda (").
Contoh : "Virusland", "Jakarta", "AMIK BSI", "Y" dan lain-lain.

C. Deklarasi Konstanta

Bentuk deklarasi konstanta diawali dengan reserved word *const*.

Bentuk penulisannya :

```
const nama_konstanta = nilai konstanta;
```

atau

```
const tipe_data nama_konstanta = nilai konstanta;
```

Contoh: `const x = 89;`
`const float phi = 3.14;`



Pada deklarasi konstanta bilangan desimal (floating point) harus diikutsertakan model dari tipe datanya.

2.3. Variabel

Adalah suatu tempat menampung data atau konstanta dimemori yang mempunyai nilai atau data yang dapat berubah-ubah selama proses program.

Dalam pemberian nama variabel, mempunyai ketentuan-ketentuan antara lain :

1. Tidak boleh ada spasi (contoh : gaji bersih) dan dapat menggunakan tanda garis bawah (_) sebagai penghubung (contoh : gaji_bersih).
2. Tidak boleh diawali oleh angka dan menggunakan operator aritmatika.

Variabel, dibagi menjadi dua jenis kelompok, yaitu :

- Variabel Numerik
- Variabel Teks

A. Variabel Numerik

Variabel numerik ini dibagi menjadi menjadi 3 (tiga) macam :

1. Bilangan Bulat atau Integer
2. Bilangan Desimal Berpresisi Tunggal atau Floating Point.
3. Bilangan Desimal Berpresisi Ganda atau Double Precision.

B. Variabel Text

1. Character (Karakter Tunggal)
2. String (Untuk Rangkaian Karakter)

C. Deklarasi Variabel

Adalah proses memperkenalkan variabel kepada Borland C++ dan pendeklarasian tersebut bersifat mutlak karena jika tidak diperkenalkan terlebih dahulu maka Borland C++ tidak menerima variabel tersebut.

Deklarasi Variabel ini meliputi tipe variabel, seperti integer atau character dan nama variabel itu sendiri. Setiap kali pendeklarasian variabel harus diakhiri oleh tanda titik koma (;).

Tabel 2.3. Deklarasi Variabel

TIPE VARIABEL	SIMBOL DEKLARASI
Integer	Int
Floating Point	Float
Double Precision	Double
Karakter	Char
Unsigned Integer	unsigned int
Unsigned Character	unsigned char
Long Integer	long int
Unsigned Long Integer	unsigned long int

Bentuk penulisannya :

```
Tipe data nama_variabel;
```

Contoh Deklarasi char nama_mahasiswa[20];
char grade;
float rata_rata ;
int nilai;

2.4. Perintah Keluaran

Perintah standar output yang disediakan oleh Borland C++, diantaranya adalah :

- printf()
- puts()
- putchar()
- cout()

2.4.1 printf()

Fungsi *printf()* merupakan fungsi keluaran yang paling umum digunakan untuk menampilkan informasi kelayar.

`printf("string-kontrol", argumen-1, argumen-2,`

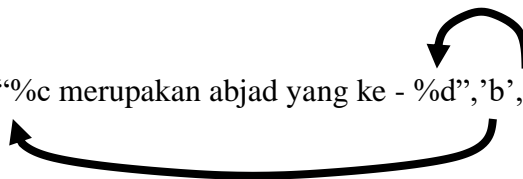
String-Kontrol dapat berupa keterangan yang akan ditampilkan pada layar beserta penentu format. Penentu format dipakai untuk memberi tahu kompilernya mengenai jenis data yang dipakai dan akan ditampilkan.

Argumen ini dapat berupa variabel, konstanta dan ungkapan.

Tabel 2.4. Penentu Format Printf()

TIPE DATA	Penentu Format Untuk <i>printf()</i>
Integer	%d
Floating Point	
Bentuk Desimal	%f
Bentuk Berpangkat	%e
Bentuk Desimal dan Pangkat	%g
Double Precision	%lf
Character	%c
String	%s
Unsigned Integer	%u
Long Integer	%ld
Long Unsigned Integer	%lu
Unsigned Hexadecimal Integer	%x
Unsigned Octal Integer	%o

`printf("%c merupakan abjad yang ke - %d", 'b', 2);`

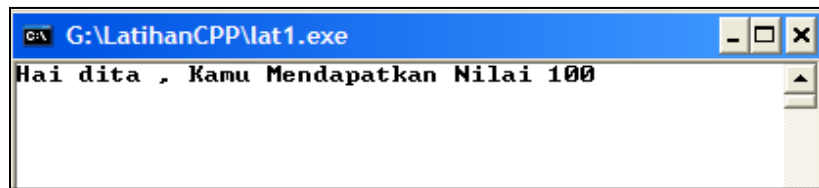


Contoh-1

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

main( )
{
    char nama ="dita";
    int nilai = 100;
    clrscr();

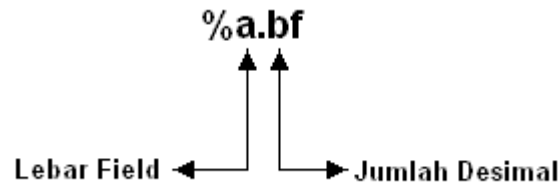
    printf("hai %s, Kamu mendapatkan Nilai %i" , nama, nilai);
}
```



Gambar 2. 1 Hasil Contoh 1

a. Penggunaan Penentu Lebar Field

Bila ingin mencetak atau menampilkan data yang bertipe data FLOAT atau pecahan, tampilan yang tampak biasanya kurang bagus. Hal tersebut dapat diatur lebar field-nya dan jumlah desimal yang ingin dicetak. Berikut bentuk penulisannya:

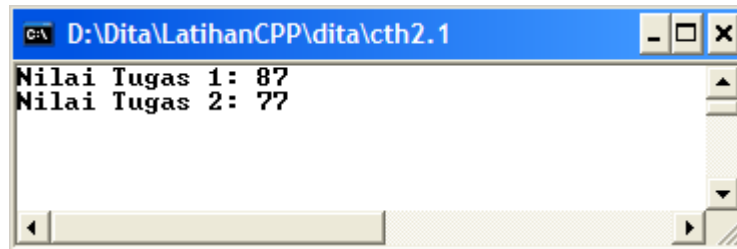


Contoh-2

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

main( )
{
    float a1=87.5, a2=77.50;
    clrscr();
    printf("Nilai Tugas 1: %2f \n",a1);
    printf("Nilai Tugas 2: %2f ",a2);
    getch();
}
```

Output yang akan dihasilkan, jika tidak menggunakan penentu lebar field adalah:

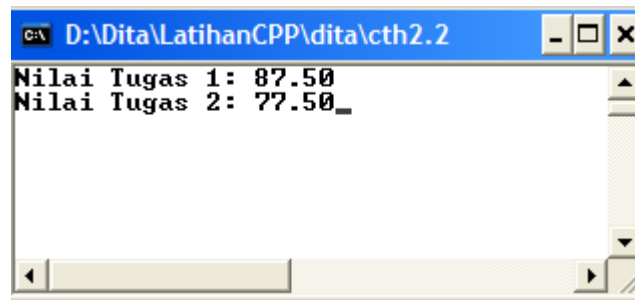


Gambar 2. 2 Hasil Contoh 2

Contoh-3

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main( )
{
float a1=87.5, a2=77.50;
clrscr();
printf("Nilai Tugas 1: %2.2f \n",a1);
printf("Nilai Tugas 2: %2.2f",a2);
getch( );
}
```

Output yang akan dihasilkan, jika menggunakan penentu lebar field adalah



Gambar 2. 3 Hasil Contoh 3

b. Penggunaan Escape Sequences.

Escape Sequences menggunakan notasi “ \ ” (back slash) jika karakter terdapat notasi “\” ini sebagai karakter “escape” (menghindari).
 Beberapa Escape Sequences lainnya antara lain :

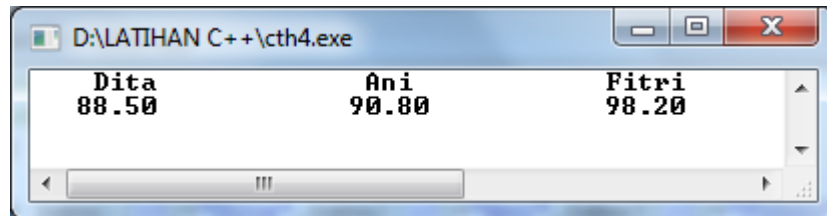
Tabel 2.5. Escape Sequences

ESCAPE SEQUENCES	PENGERTIAN
\b	Backspace
\f	Formfeed
\n	Baris Baru
\r	Carriage Return
\t	Tab (default = 8 karakter)
\'	Tanda kutip tunggal (')
\"	Tanda Kutip Ganda (")

\\	Backslash
\xaa	Kode ASCII dalam hexadecimal. (aa menunjukkan angka ASCII ybs)
\aaa	Kode ASCII dalam octal. (aaa menunjukkan angka ASCII ybs)

Contoh-4

```
#include<stdio.h>
#include<conio.h>
main()
{
char nama1[10]="Dita",nama2[10]="Ani",nama3[10]="Fitri";
float a = 88.5, b = 90.8, c = 98.2;
clrscr();
printf("%8s\t %7s\t %8s ", nama1,nama2,nama3);
printf("\n%8.2f \t %8.2f \t %8.2f ", a,b,c);
getch();
}
```



Gambar 2. 4 Hasil Contoh 4

2.4.2. puts()

Perintah *puts()* sebenarnya sama dengan *printf()*, yaitu digunakan untuk mencetak string ke layar. *puts()* berasal dari kata **PUT STRING**.

Perbedaan antara *printf()* dengan *puts()* adalah :

Tabel 2.6. Perbedaan fungsi *puts()* dengan *printf()* untuk tipe data string

<i>printf()</i>	<i>puts()</i>
Harus menentukan tipe data untuk data string, yaitu %s	Tidak Perlu penentu tipe data string, karena fungsi ini khusus untuk tipe data string.
Untuk mencetak pindah baris, memerlukan notasi ‘ \n ‘	Untuk mencetak pindah baris tidak perlu notasi ‘ \n ‘ , karena sudah dibeikan secara otomatis.

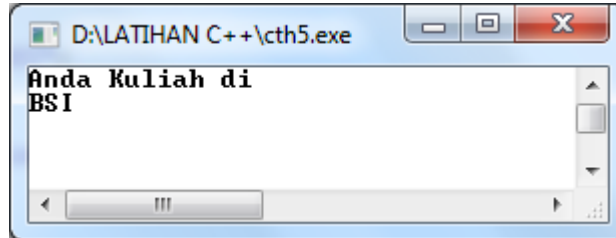
Contoh-5

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

main()
{
```

```
char nama[5] = "BSI";  
clrscr();  
  
puts("Anda Kuliah di ");  
puts(a);  
}
```

Output yang akan dihasilkan, dari program contoh-5 diatas adalah :



Gambar 2. 5 Hasil Contoh 5

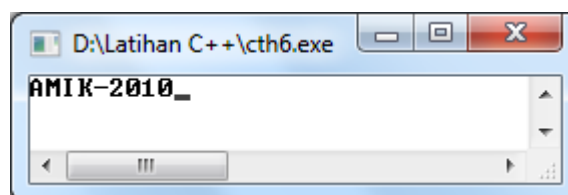
2.4.3. putchar()

Perintah *putchar()* digunakan untuk menampilkan **sebuah karakter** ke layar. Penampilan karakter tidak diakhiri dengan pindah baris.

Contoh-5 `#include <stdio.h>`
 `#include <conio.h>`

```
main()  
{  
  clrscr( );  
  putchar('B');  
  putchar('S');  
  putchar('I');  
  putchar('-');  
  putchar('2');  
  putchar('0');  
  putchar('0');  
  putchar('9');  
  getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah :



Gambar 2. 6 Hasil Contoh 6

2.4.4. cout

Fungsi *cout* merupakan sebuah object didalam Borland C++ digunakan untuk menampilkan suatu data kelayar. Untuk menggunakan fungsi cout ini, harus menyertakan file header **iostream.h** .

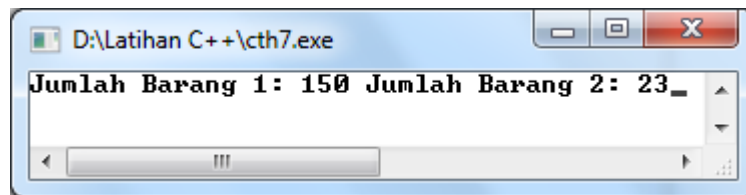
```
Contoh-7      #include <stdio.h>
               #include <conio.h>
               #include <iostream.h>

               main( )
               {
                 int jumbar1=150, jumbar2=23;

                 clrscr();
                 cout<<"Jumlah Barang 1: "<<jumbar1;
                 cout<<" Jumlah Barang 2: "<<jumbar2;

                 getch();
               }
```

Output yang akan dihasilkan, dari program contoh-7 diatas adalah :



Gambar 2. 7 Hasil Contoh 7

2.4.5. Fungsi Manipulator

Manipulator pada umumnya digunakan untuk mengatur tampilan layar, untuk menggunakan manipulator ini file header yang harus disertakan file header **iomanip.h** . Ada beberapa fungsi manipulator yang disediakan oleh Borland C++, antara lain.

- endl
- end
- flush()
- dec()
- hex()
- oct()
- setbase()
- setw()
- setfill()
- setprecision()
- setosflags()

Berikut akan dibahas beberapa fungsi manipulator, diantaranya:

a. endl

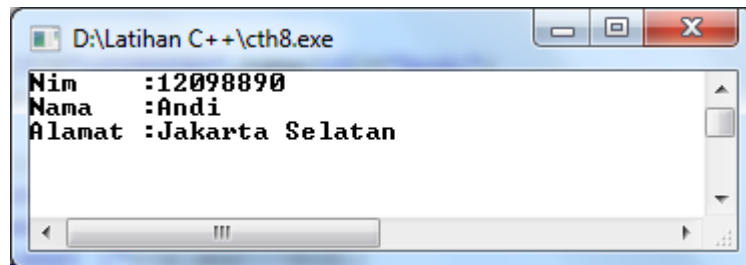
endl merupakan suatu fungsi manipulator yang digunakan untuk menyisipkan karakter NewLine atau mengatur pindah baris. Fungsi ini sangat berguna untuk piranti keluaran berupa file di disk. File header yang harus disertakan adalah file header **iostream.h** .

Contoh-8

```
# include<stdio.h>
# include<conio.h>
# include<iostream.h>
main( )
{
char nim[9]="12098890", nama[15]="Andi";
char alamat[20]="Jakarta Selatan";

clrscr( );
cout<<"Nim   :"<<nim<<endl;
cout<<"Nama  :"<<nama<<endl;
cout<<"Alamat :"<<alamat<<endl;
getch( );
}
```

Output yang akan dihasilkan, dari program contoh-8 diatas adalah :



Gambar 2. 8 Hasil Contoh 8

b. ends

ends merupakan suatu fungsi manipulator yang digunakan untuk menambah karakter null (nilai ASCII NOL) kederetan suatu karakter. Fungsi ini akan berguna untuk mengirim sejumlah karakter kefile didisk atau modem dan mangakhirinya dengan karakter NULL.. File header yang harus disertakan adalah file header **iostream.h** .

Contoh-9

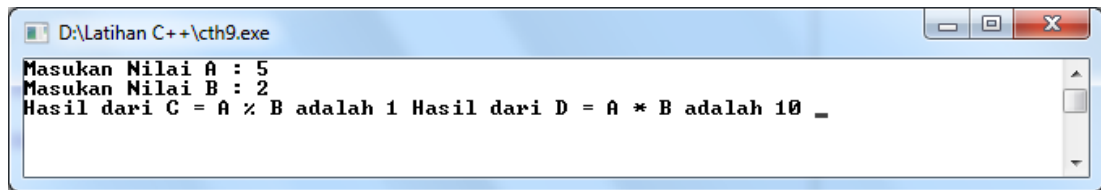
```
# include <stdio.h>
# include <conio.h>
# include <iostream.h>
main( )
{
int a, b, c, d;
clrscr( );
cout<<"Masukan Nilai A : "; cin>>a;

cout<<"Masukan Nilai B : "; cin>>b;
c = a % b;
d = a * b;

cout<<"Hasil dari C = A % B adalah "<<c<<ends;
```

```
cout<<"Hasil dari D = A * B adalah "<<d<<ends;  
getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-9 diatas adalah :



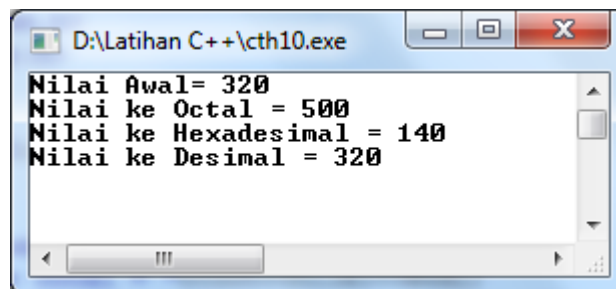
Gambar 2. 9 Hasil Contoh 9

c. dec, oct dan hex

dec, oct dan hex merupakan suatu fungsi manipulator yang digunakan untuk menampilkan data dalam bentuk desimal(bilangan berbasis 10), oktal(bilangan berbasis 8) dan hexadesimal(bilangan berbasis 16). File header yang harus disertakan adalah file header **iomanip.h** .

```
Contoh-10 # include <stdio.h>  
# include <conio.h>  
# include <iostream.h>  
# include <iomanip.h>  
main( )  
{  
int nilai = 320;  
clrscr( );  
  
cout<<"Nilai Awal= "<<nilai<<endl;  
cout<<"Nilai ke Octal = "<<oct<<nilai<<endl;  
cout<<"Nilai ke Hexadesimal = "<<hex<<nilai<<endl;  
cout<<"Nilai ke Desimal = "<<dec<<nilai<<endl;  
  
getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-10 diatas adalah:



Gambar 2. 10 Hasil Contoh 10

d. `setprecision()`

Fungsi `setprecision()` merupakan suatu fungsi manipulator yang digunakan untuk mengatur jumlah digit desimal yang ingin ditampilkan. Fungsi ini biasa pada fungsi `cout()`, file header yang harus disertakan adalah file header `iomanip.h`.

```

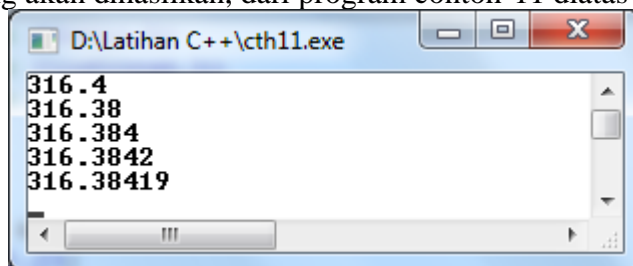
Contoh-11    # include <conio.h>
                # include <iostream.h>
                # include <iomanip.h>

                main( )
                {
                    float a,b,c;
                    a = 25.23;
                    b = 12.54;
                    clrscr( );

                    c = a * b;

                    cout<<setiosflags(ios::fixed);
                    cout<<setprecision(1)<<c<<endl;
                    cout<<setprecision(2)<<c<<endl;
                    cout<<setprecision(3)<<c<<endl;
                    cout<<setprecision(4)<<c<<endl;
                    cout<<setprecision(5)<<c<<endl;
                    getch( );
                }
    
```

Output yang akan dihasilkan, dari program contoh-11 diatas adalah:



Gambar 2. 11 Hasil Contoh 11

e. `setbase()`

`setbase()` merupakan suatu fungsi manipulator yang digunakan untuk konversi bilangan Octal, Decimal dan Hexadecimal. File header yang harus disertakan file header `iomanip.h`.

Bentuk penulisannya :

`setbase(base bilangan);`

Base bilangan merupakan base dari masing-masing bilangan, yaitu :

- Octal = basis 8
- Decimal = basis 10
- Hexadecimal = basis 16

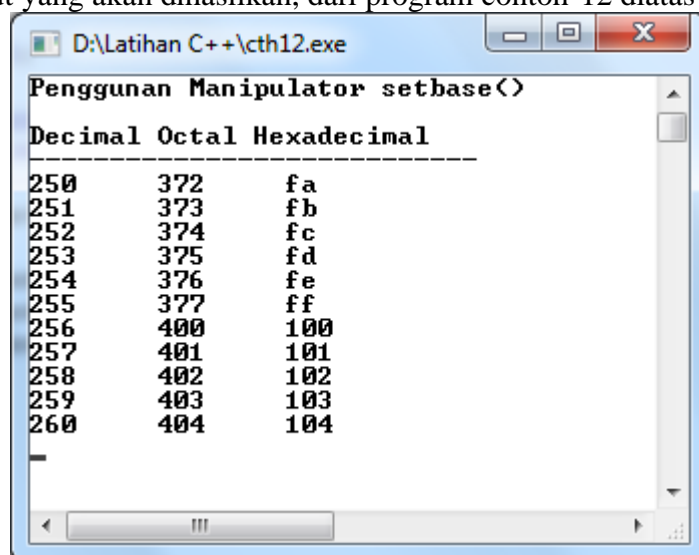
Contoh-12 //Penggunaan Manipulator `setbase()`

```

#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
main( )
{
int a = 250;
clrscr( );
cout<<"Peggungan Manipulator setbase()"<<"\n\n";
cout<<"Decimal Octal Hexadecimal"<<endl;
cout<<"-----"<<"\n";
for(a=250;a<=260;a++)
{
cout<<setbase(10)<<a<<" ";
cout<<setbase(8)<<a<<" ";
cout<<setbase(16)<<a<<endl;
}
getche( );
}

```

Output yang akan dihasilkan, dari program contoh-12 diatas adalah:



Gambar 2.12. Hasil Contoh-12

f. setw()

setw() merupakan suatu fungsi manipulator yang digunakan untuk mengatur lebar tampilan dilayar dari suatu nilai variabel. File header yang harus disertakan file header **iomanip.h** .

Bentuk penulisannya :

setw(int n);

n = merupakan nilai lebar tampilan data, integer.

Contoh-13 // Penggunaan Manipulator setw()

```

#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>

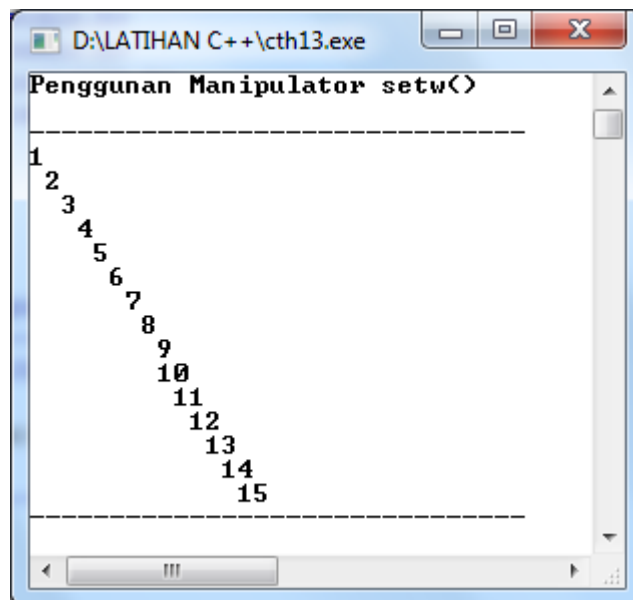
```

```

main( )
{
int a;
clrscr( );
cout<<"Peggungan Manipulator setw()"<<"\n\n";
cout<<"-----"<<"\n";
for(a=1;a<=15;a++)
{
cout<<setw(a)<<a<<endl;
}
cout<<"-----"<<"\n";
getche( );
}

```

Output yang akan dihasilkan, dari program contoh-13 diatas adalah:



Gambar 2.13. Hasil Contoh-13

g. setfill()

setfill() merupakan suatu fungsi manipulator yang digunakan untuk menampilkan suatu karakter yang ditelakan didepan nilai yang diatur oleh fungsi **setw()**. File header yang harus disertakan file header **iomanip.h** .

Bentuk penulisannya :

setfill(charakter);

Contoh-14 // penggunaan setfill dan setw()

```

#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
main( )
{

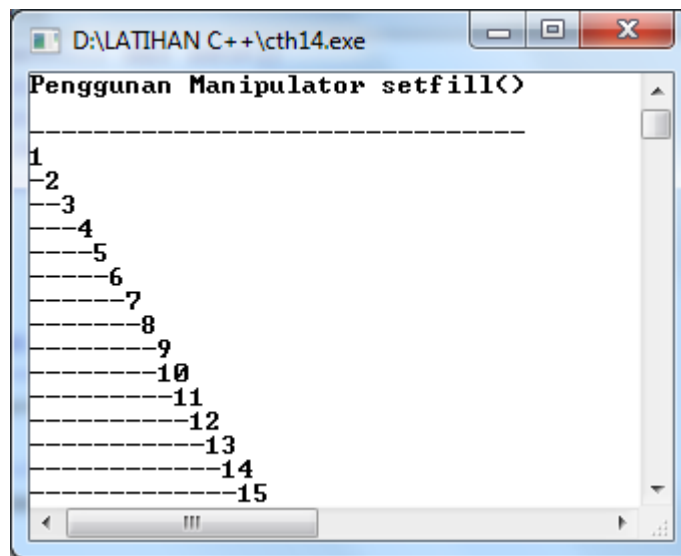
```

```

int a;
clrscr( );
cout<<"Penggunaan Manipulator setfill()"<<"\n\n";
cout<<"-----"<<"\n";
for(a=1;a<=15;a++)
{
cout<<setfill('-');
cout<<setw(a)<<a<<endl;
}
getche( );
}

```

Output yang akan dihasilkan, dari program contoh-14 diatas adalah:



Gambar 2.14. Hasil Contoh-14

h. *setiosflags()*

Fungsi *setiosflags()* merupakan suatu fungsi manipulator yang digunakan untuk mengatur sejumlah format keluaran data.. Fungsi ini biasa pada fungsi **cout()**, file header yang harus disertakan file header **iosmanip.h**.

Ada beberapa format keluaran untuk fungsi **setiosflags()**, antara lain.

1. Tanda Format Perataan Kiri dan Kanan

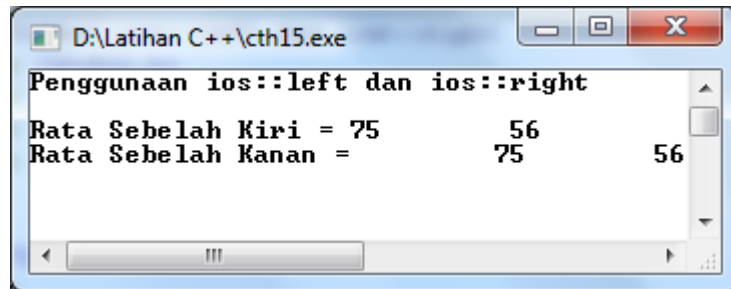
Terdapat dua buah tanda format yang digunakan untuk perataan kiri dan kanan, pengaturan terhadap lebar variabel untuk perataan kiri dan kanan ini melalui fungsi **setw()**.

- **ios::left** digunakan untuk mengatur perataan sebelah kiri
- **ios::right** digunakan untuk mengatur perataan sebelah kanan

Contoh-15 //tanda format **ios::left** dan **ios::right**
include <stdio.h>

```
# include <conio.h>
# include <iostream.h>
# include <iomanip.h>
main( )
{
int a = 75, b = 56;
clrscr( );
cout<<"Penggunaan ios::left dan ios::right\n\n";
cout<<"Rata Sebelah Kiri = ";
cout<<setiosflags(ios::left)<<setw(10)<<a;
cout<<setiosflags(ios::left)<<setw(10)<<b;
cout<<endl;
cout<<"Rata Sebelah Kanan = ";
cout<<setiosflags(ios::right)<<setw(10)<<a;
cout<<setiosflags(ios::right)<<setw(10)<<b;
getche( );
}
```

Output yang akan dihasilkan, dari program contoh-15 diatas adalah:



Gambar 2.15. Hasil Contoh-15

2. Tanda Format Keluaran Notasi Konversi

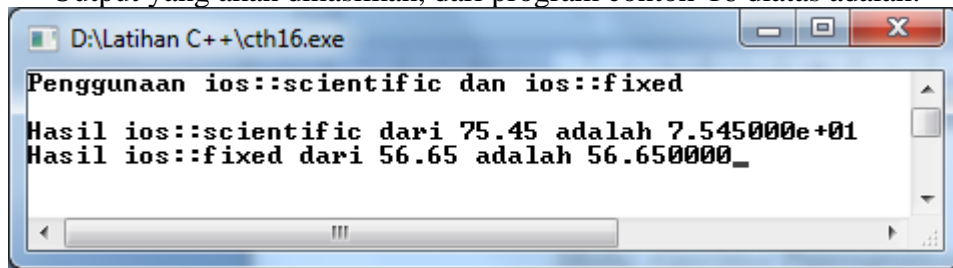
Tanda format yang digunakan untuk keluaran Notasi, yaitu:

- **ios::scientific** digunakan untuk mengatur keluaran dalam bentuk notasi eksponensial.
- **ios::fixed** digunakan untuk mengatur keluaran dalam bentuk notasi desimal.

Contoh-16

```
//tanda format ios::scientific dan ios::fixed
# include <stdio.h>
# include <conio.h>
# include <iostream.h>
# include <iomanip.h>
main( )
{
clrscr( );
cout<<"Penggunaan ios::scientific dan ios::fixed\n";
cout<<"\nHasil ios::scientific dari 75.45 adalah ";
cout<<setiosflags(ios::scientific)<<75.45<<endl;
cout<<"Hasil ios::fixed dari 56.65 adalah ";
cout<<setiosflags(ios::fixed)<<56.65;
getche( );
}
```

Output yang akan dihasilkan, dari program contoh-16 diatas adalah:



Gambar 2.16. Hasil Contoh-16

3. Tanda Format Konversi Dec, Oct dan Hex

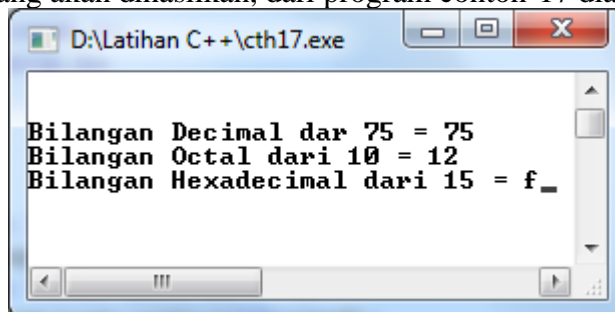
Terdapat tiga macam tanda format yang digunakan untuk konversi keluaran dalam basis Decimal, Octal dan Hexadecimal, yaitu:

- **ios::dec** digunakan untuk mengatur keluaran dalam konversi basis desimal.
- **ios::oct** digunakan untuk mengatur keluaran dalam konversi basis oktal.
- **ios::hex** digunakan untuk mengatur keluaran dalam konversi basis heksadesimal.

Contoh-17

```
//tanda format ios::dec, ios::oct, ios::hex
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
main()
{
clrscr();
cout<<"\n\n";
cout<<"Bilangan Decimal dar 75 = ";
cout<<setiosflags(ios::dec)<<75<<endl;
cout<<"Bilangan Octal dari 10 = ";
cout<<setiosflags(ios::oct)<<10<<endl;
cout<<"Bilangan Hexadecimal dari 15 = ";
cout<<setiosflags(ios::hex)<<15;
getche();
}
```

Output yang akan dihasilkan, dari program contoh-17 diatas adalah:



Gambar 2.17. Hasil Contoh-17

4. Tanda Format Manipulasi Huruf Hexadecimal

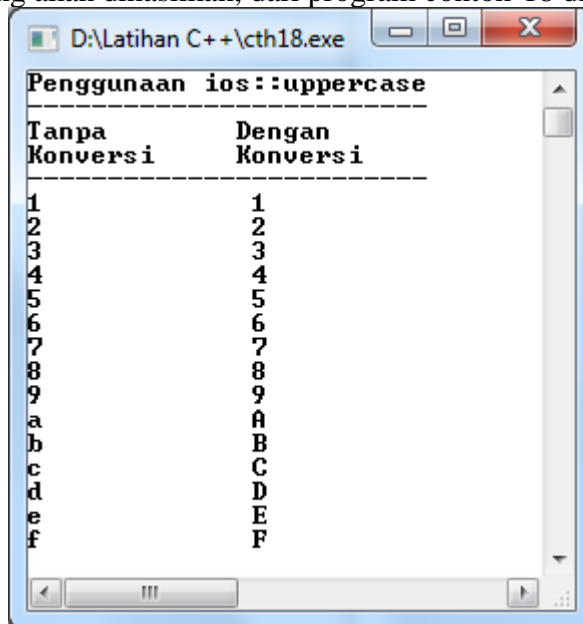
Untuk keperluan memanipulasi atau mengubah huruf pada notasi hexadecimal dengan menggunakan tanda format:

- **ios::uppercase** digunakan untuk mengubah huruf pada notasi huruf hexadecimal.

Contoh-18

```
//tanda format ios::uppercase
# include <stdio.h>
# include <conio.h>
# include <iostream.h>
# include <iomanip.h>
main( )
{
int a;
clrscr( );
cout<<"Penggunaan ios::uppercase\n";
cout<<"-----\n";
cout<<"Tanpa Dengan \n";
cout<<"Konversi Konversi \n";
cout<<"-----\n";
for (a=1; a<=15; a++)
cout<<hex<<a<<endl;
for (a=1; a<=15; a++)
{
gotoxy(15,a+5);
cout<<setiosflags(ios::uppercase)<<hex<<a<<endl;
}
getche( );
}
```

Output yang akan dihasilkan, dari program contoh-18 diatas adalah:



Gambar 2.18 Hasil Contoh-18

5. Tanda Format Keluaran Dasar Bilangan Hexadecimal dan Octal

Untuk keperluan menampilkan dasar bilangan Hexadecimal dan Oktal dengan menggunakan tanda format:

- **ios::showbase** digunakan untuk menampilkan tanda **0x** (nol-x) diawal pada tampilan bilangan hexadecimal dan **0** (nol) diawal pada tampilan bilangan decimal.

```
Contoh-19    //tanda format ios::showbase
               # include <stdio.h>
               # include <conio.h>
               # include <iostream.h>
               # include <iomanip.h>
               main( )
               {
               int a;
               clrscr( );
               cout<<"Penggunaan ios::showbase\n";
               cout<<"-----\n";
               cout<<"Decimal Hexadecimal Oktal \n";
               cout<<"-----\n";
               cout<<setiosflags(ios::showbase);
               for (a=1; a<=15; a++)
               {
               gotoxy(4,a+5);
               cout<<dec<<a<<endl;
               }
               for (a=1; a<=15; a++)
               {
               gotoxy(15,a+5);
               cout<<hex<<a<<endl;
               }

               for (a=1; a<=15; a++)
               {
               gotoxy(25,a+5);
               cout<<oct<<a<<endl;
               }
               cout<<"-----\n";
               getch( );
               }
```

Output yang akan dihasilkan, dari program contoh-19 diatas adalah:

Decimal	Hexadecimal	Okta1
1	0x1	01
2	0x2	02
3	0x3	03
4	0x4	04
5	0x5	05
6	0x6	06
7	0x7	07
8	0x8	010
9	0x9	011
10	0xa	012
11	0xb	013
12	0xc	014
13	0xd	015
14	0xe	016
15	0xf	017

Gambar 2.19. Hasil Contoh-19

6. Tanda Format Menampilkan Titik Desimal

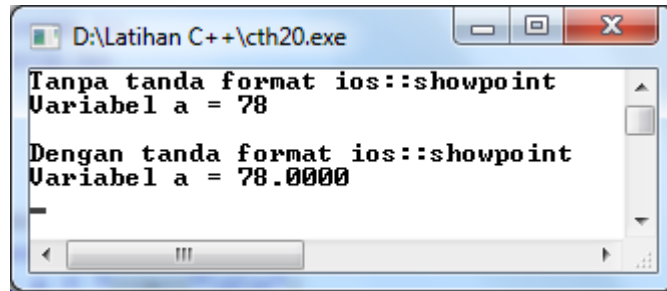
Untuk keperluan menampilkan titik desimal dengan menggunakan tanda format:

- **ios::showpoint** digunakan untuk menampilkan titik desimal pada bilangan yang tidak mempunyai titik desimal pada tipe data float atau double.

Contoh-20

```
//tanda format ios::showpoint
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
main()
{
double a = 78;
clrscr();
//-> tanpa tanda format ios::showpoint
cout<<"Tanpa tanda format ios::showpoint"<<endl;
cout<<"Variabel a = "<<a<<"\n\n";
//-> dengan tanda format ios::showpoint
cout<<"Dengan tanda format ios::showpoint"<<endl;
cout<<setiosflags(ios::showpoint);
cout<<"Variabel a = "<<a<<endl;
getche();
}
```

Output yang akan dihasilkan, dari program contoh-20 diatas adalah:



Gambar 2.20. Hasil Contoh-20

7. Tanda Format Menampilkan Simbol Plus (+)

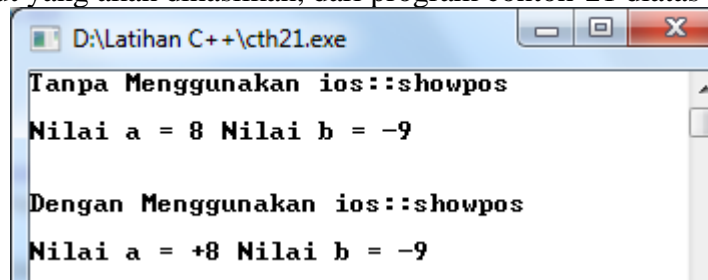
Untuk keperluan menampilkan simbol Plus (+) pada bilangan genap dengan menggunakan tanda format:

- **ios::showpos** digunakan untuk menampilkan simbol plus (+) pada variabel yang memiliki nilai bilangan positif.

```

Contoh-21 //tanda format ios::showpos
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
main( )
{
int a = 8, b = -9;
clrscr( );
cout<<"Tanpa Menggunakan ios::showpos"<<"\n\n";
cout<<"Nilai a = "<<a<<" Nilai b = "<<b<<endl;
cout<<"\n\n";
cout<<setiosflags(ios::showpos);
cout<<"Dengan Menggunakan ios::showpos"<<"\n\n";
cout<<"Nilai a = "<<a<<" Nilai b = "<<b<<endl;
getche();
}
    
```

Output yang akan dihasilkan, dari program contoh-21 diatas adalah:



Gambar 2.21. Hasil Contoh-21

2.5. Perintah Masukan

Perintah standar input yang disediakan oleh Borland C++, diantaranya adalah:

- **scanf()**
- **gets()**
- **cout()**
- **getch()**
- **getche()**

2.5.1. scanf()

Fungsi *scanf()* digunakan untuk memasukkan berbagai jenis data. Bentuk Umum dari fungsi ini adalah:

```
scanf("penentu format", &nama-variabel);
```

simbol **&** merupakan pointer yang digunakan untuk menunjuk kealamat variabel memori yang dituju.

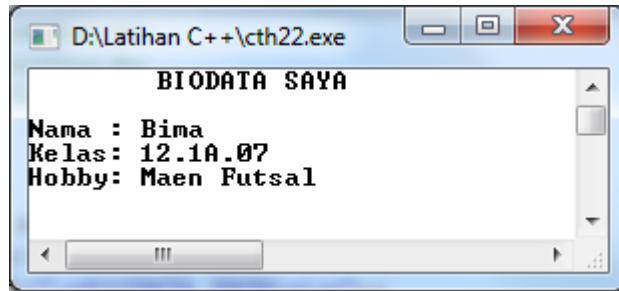
Tabel 2.7. Penentu Format scanf()

TIPE DATA	Penentu Format Untuk <i>scanf()</i>
Integer	%d
Floating Point	
Bentuk Desimal	%e atau %f
Bentuk Berpangkat	%e atau %f
Double Precision	%lf
Character	%c
String	%s
Unsigned Integer	%u
Long Integer	%ld
Long Unsigned Integer	%lu
Unsigned Hexadecimal Integer	%x
Unsigned Octal Integer	%o

```
Contoh-22  #include<stdio.h>
            #include<conio.h>
            #include<iostream.h>

            main( )
            {
            char nama[15],kelas[20],hobby[15];
            clrscr( );
            printf("\tBIODATA SAYA\n");
            printf("Nama : ");scanf("%s",&nama);
            printf("Kelas: ");scanf("%s",&kelas);
            printf("Hobby: ");scanf("%s",&hobby);
            getch( );
            }
```

Output yang akan dihasilkan, dari program contoh-22 diatas adalah:



Gambar 2. 22 Hasil Contoh 22

2.5.2. gets()

Fungsi *gets()* digunakan untuk memasukkan data string. Bentuk Umum dari fungsi ini adalah:

gets (nama-variabel-array) ;

Perbedaan antara *scanf()* dengan *gets()* adalah:

Tabel 2.8. Perbedaan scanf() dengan gets()

<i>scanf()</i>	<i>gets()</i>
Tidak dapat menerima string yang mengandung spasi atau tab dan dianggap sebagai data terpisah	Dapat menerima string yang mengandung spasi atau tab dan masing dianggap sebagai satu kesatuan data.

Contoh-23

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

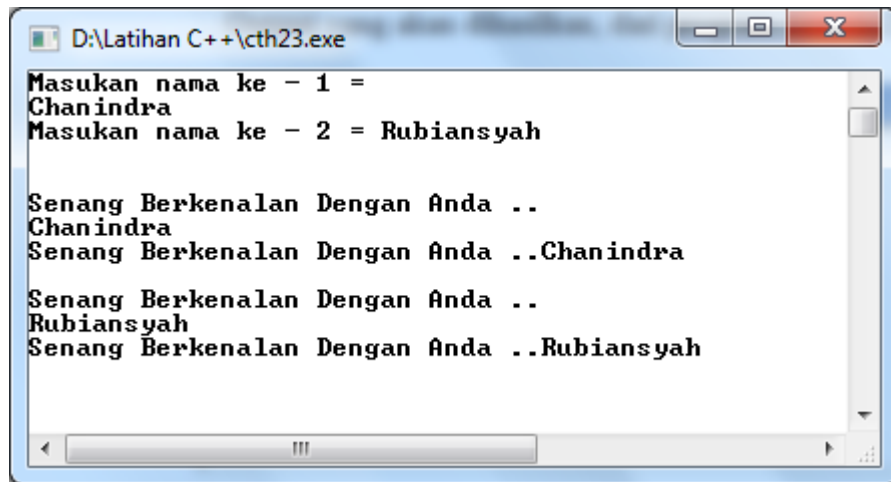
main( )
{
    char nm1[20];
    char nm2[20];

    clrscr( );

    puts("Masukan nama ke - 1 = ");
    gets(nm1);
    printf("Masukan nama ke - 2 = ");
    scanf("%s",&nm2);
    printf("\n\n");
    puts("Senang Berkenalan Dengan Anda ..");
    puts(nm1);
    printf("Senang Berkenalan Dengan Anda ..%s", nm1);
    printf("\n\n");
    puts("Senang Berkenalan Dengan Anda ..");
    puts(nm2);
    printf("Senang Berkenalan Dengan Anda ..%s", nm2);
    getch( );
}
```

}

Output yang akan dihasilkan, dari program contoh-23 diatas adalah:



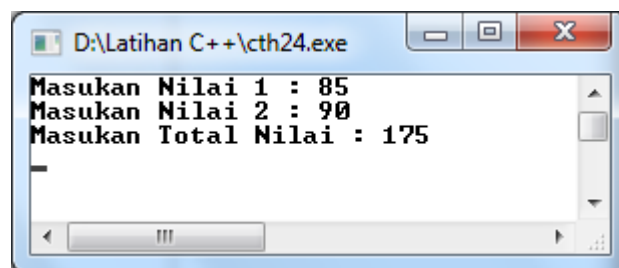
Gambar 2. 12 Hasil Contoh 23

2.5.3. cin

Fungsi *cin* merupakan sebuah object didalam C++ digunakan untuk memasukkan suatu data. Untuk menggunakan fungsi cin ini, harus menyertakan file header **iostream.h** .

```
Contoh-24    # include <stdio.h>
               # include <conio.h>
               # include <iostream.h>
               main( )
               {
                 int nilai1,nilai2, total;
                 clrscr( );
                 cout<<"Masukan Nilai 1 : ";
                 cin>>nilai1;
                 cout<<"Masukan Nilai 2 : ";
                 cin>>nilai2;
                 total=nilai1+nilai2;
                 cout<<"Masukan Total Nilai : "<<total<<endl;
                 getch( );
               }
```

Output yang akan dihasilkan, dari program contoh-24 diatas adalah:



Gambar 2. 13 Hasil Contoh 24

2.5.4. `getch()`

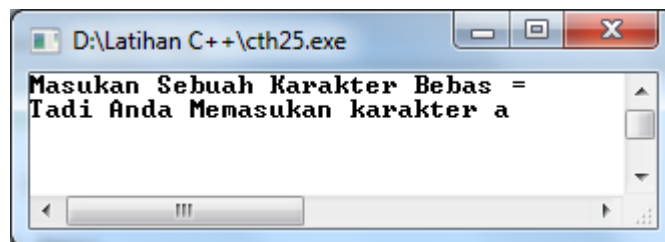
Fungsi `getch()` (*get character and echo*) dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukkan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan tidak akan ditampilkan di layar. File header yang harus disertakan adalah **conio.h**.

```
Contoh-25      # include <stdio.h>
                  # include <conio.h>

                  main( )
                  {
                    char kar;

                    clrscr( );
                    printf("Masukan Sebuah Karakter Bebas = ");
                    kar = getch( );
                    printf("\nTadi Anda Memasukan karakter %c", kar);
                    getch( );
                  }
```

Output yang akan dihasilkan, dari program contoh-25 diatas adalah:



Gambar 2. 14 Hasil Contoh 25

2.5.5. `getche()`

Fungsi `getche()` dipakai untuk membaca sebuah karakter dengan sifat karakter yang dimasukkan tidak perlu diakhiri dengan menekan tombol ENTER, dan karakter yang dimasukan ditampilkan di layar. File header yang harus disertakan adalah **conio.h**.

```
Contoh-26      # include <stdio.h>
                  # include <conio.h>

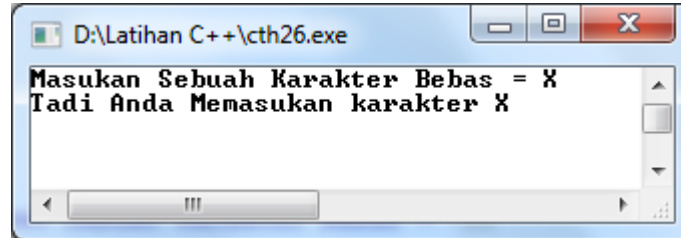
                  main( )

                  {
                    char kar;
                    clrscr( );


                    printf("Masukan Sebuah Karakter Bebas = ");
                    kar = getche( );
```

```
printf("\nTadi Anda Memasukan karakter %c", kar);  
getch ();  
}
```

Output yang akan dihasilkan, dari program contoh-26 diatas adalah:



Gambar 2. 15 Hasil Contoh 26

 Kedua fungsi ini dapat digunakan untuk menahan tampilan hasil program yang di eksekusi agar tidak langsung kembali ke listing program tanpa menekan tombol ALT – F5. Karena fungsi *getch()* merupakan fungsi masukkan, jadi sebelum program keluar harus menginputkan satu buah karakter.

2.6. Tugas 1

Kerjakan tampilan Berikut : dengan menggunakan deklarasi variabel dan sintaks input(*gets,scanf,cin*) dan output(*printf&cout*)



Operator Bahasa C++ Dan Fungsi Manipulasi String serta Konversi String



Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program, yang digunakan untuk melakukan sesuatu operasi atau manipulasi, seperti penjumlahan, pengurangan dan lain-lain.

Operator mempunyai sifat sebagai berikut:

- **Unary**
Sifat Unary pada operator adalah hanya melibatkan sebuah operand pada suatu operasi aritmatik
Contoh : -5
- **Binary**
Sifat Binary pada operator adalah melibatkan dua buah operand pada suatu operasi aritmatik
Contoh : $4 + 8$
- **Ternary**
Sifat Ternary pada operator adalah melibatkan tiga buah operand pada suatu operasi aritmatik
Contoh :
 $(10 \% 3) + 4 + 2$

3.1. Operator Aritmatika

Operator untuk operasi aritmatika yang tergolong sebagai operator binary adalah:

Tabel 3.1. Operator Aritmatika

Operator	Keterangan	Contoh
*	Perkalian	$4 * 5$
/	Pembagian	$8 / 2$
%	Sisa Pembagian (mod)	$5 \% 2$
+	Penjumlahan	$7 + 2$
-	Pengurangan	$6 - 2$

Operator yang tergolong sebagai operator *Unary*, adalah:

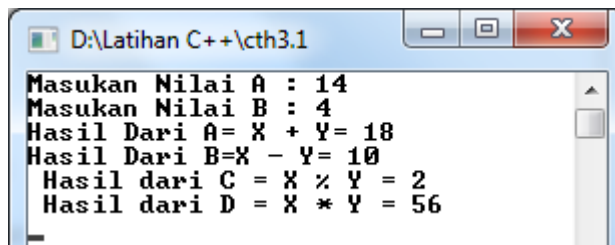
Tabel 3.2. Operator Unary

Operator	Keterangan	Contoh
+	Tanda Plus	-4
-	Tanda Minus	+6

Contoh-1

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
main( )
{
    int x,y, a=0, b=0, c = 0, d = 0;
    clrscr( );
    cout<<"Masukan Nilai A : "; cin>>x;
    cout<<"Masukan Nilai B : "; cin>>y;
    a=x+y; c = x% y;
    b=x-y; d = x * y;
    printf("Hasil Dari A= X + Y= %i \n", a);
    printf("Hasil Dari B=X - Y= %i \n",b)
    cout<<" Hasil dari C = X % Y = "<<c<<endl;
    cout<<" Hasil dari D = X * Y = "<<d<<endl;
    getch( );
}
```

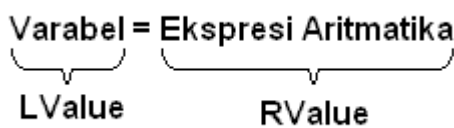
Output yang akan dihasilkan, dari program contoh-1 diatas adalah :



Gambar 3. 1 Hasil Contoh 1

3.1.1. Ekspresi Aritmatika

Bentuk penulisan ekspresi aritmatika dikaitkan dengan pernyataan pemberi nilai. Bentuk Umum :



- Variabel, dikenal dengan sebutan LValue (*Left Value*)
- Ekspresi Aritmatika dikenal dengan sebutan RValue (*Right Value*)

- Tanda “ = “, dikenal dengan sebagai *Operator Pemberi Nilai (Assignment Operator)*.



LValue harus selalu berupa variabel tunggal. Bila LValue bukan berupa variabel, maka akan tampil pesan kesalahan ***LValue required in function ...***

RValue dapat berupa konstanta, variabel lain maupun suatu ekspresi atau rumus aritmatika.

3.1.2. Hierarki Operator Aritmatika.

Didalam suatu ekspresi aritmatika, selalu menjumpai beberapa operator aritmatika yang berbeda yang dapat digunakan secara bersamaan. Urutan operator aritmatika sebagai berikut :

Tabel. 3.3. Tabel Hierarki Operator Aritmatika

Operator	Keterangan
* atau /	Tingkatan operator sama, penggunaannya tergantung letak, yang didepan didahulukan
%	Sisa Pembagian
+ atau -	Tingkatan operator sama, penggunaannya tergantung letak, yang didepan didahulukan

Contoh $A = 8 + 2 * 3 / 6$

Langkah perhitungannya :

$$A = 8 + 6 / 6 \rightarrow (6 / 6 = 1)$$

$$A = 8 + 1$$

$$A = 9$$

Tingkatan operator ini dapat diabaikan dengan penggunaan tanda kurung “(“ dan “)”.

Contoh :

$$A = (8 + 2) * 3 / 6$$

Langkah perhitungannya :

$$A = 10 * 3 / 6$$

$$A = 30 / 6$$

$$A = 5$$

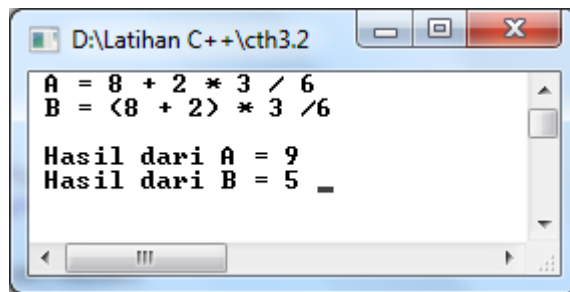
Contoh-2 `#include <stdio.h>`
 `#include <conio.h>`
 `#include <iostream.h>`

```

main( )
{
  int a, b;
  clrscr( );
  a = 8 + 2 * 3 / 6;
  b = (8 + 2) * 3 / 6;
  cout<<" A = 8 + 2 * 3 / 6"<<endl;
  cout<<" B = (8 + 2) * 3 /6"<<endl;
  cout<<endl;
  cout<<" Hasil dari A = "<<a<<endl;
  printf(" Hasil dari B = %i ",b);
  getch( );
}

```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah :



Gambar 3. 2 Hasil Contoh 2

3.2. Operator Pemberi Nilai Aritmatika

Sebelumnya kita telah mengenal operator pemberi nilai (*assignment operator*) yaitu tanda “ = “. Sebagai contoh penggunaan operator pemberi nilai $A = A + 1$

Dari penulisan ekspresi diatas, Borland C++ dapat menyederhanakan menjadi $A += 1$

Notasi “ += “ ini dikenal dengan operator pemberi nilai aritmatika. Ada beberapa operator pemberi nilai aritmatka diantaranya:

Tabel. 3.4. Tabel Operator Pemberi Nilai Aritmatika

Operator	Keterangan
*=	Perkalian
/=	Pembagian
%=	Sisa Pembagian
+=	Penjumlahan
-=	Pengurangan

3.3. Operator Penambah dan Pengurang

Masih berkaitan dengan operator pemberi nilai, Borland C++ menyediakan operator penambah dan pengurang. Dari contoh penulisan operator pemberi nilai sebagai penyederhanaannya dapat digunakan operator penambah dan pengurang.

Tabel. 3.5. Tabel Operator Penambah dan Pengurang

Operator	Keterangan
++	Penambahan
--	Pengurangan

$A = A + 1$ atau $A = A - 1$; disederhanakan menjadi $A ++$ atau $A --$
 Operator “ ++ “ atau “ -- “ dapat diletakan didepan atau di belakang variabel.

- Keterangan :**
1. Penambahan: menambahkan 1 ke nilai variable, prefix ($++A$) atau postfix ($A ++$)
 2. Pengurangan: mengurangi 1 ke nilai variabel, prefix ($--A$) atau postfix ($A --$)

Kedua bentuk penulisan operator ini mempunyai arti yang berbeda.

- **Jika diletakan didepan variabel**, maka proses penambahan atau pengurangan akan dilakukan sesaat sebelum atau langsung pada saat menjumpai ekspresi ini, sehingga nilai variabel tadi akan langsung berubah begitu ekspresi ini ditemukan, *sedangkan*
- **Jika diletakan dibelakang variabel**, maka proses penambahan atau pengurangan akan dilakukan setelah ekspresi ini dijumpai atau nilai variabel akan tetap pada saat ekspresi ini ditemukan.

Contoh **Penambahan:**
`int x=5;`
`y = ++x;`
 (nilai saat ini: y = 6, x=6)
`int x=5;`
`y = x++;`
 (nilai saat ini : y = 5, x=6)

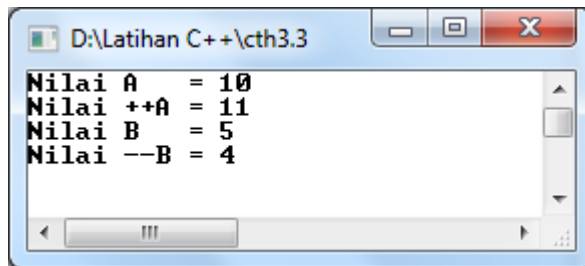
Pengurangan:
`int x=5;`
`y = --x;`
 (nilai saat ini: y = 4, x=4)
`int x=5;`
`y = x--;`
 (nilai saat ini: y = 5, x=4)

Contoh-3 `/* Penggunaan Notasi Didepan Variabel*/`
`#include <stdio.h>`
`#include <conio.h>`

```
main( )
{
  int a = 10, b = 5;
  clrscr( );

  printf("Nilai A = %d", a);
  printf("\nNilai ++A = %d", ++a);
  printf("\nNilai B = %d", b);
  printf("\nNilai --B = %d", --b);
  getch( );
}
```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah:



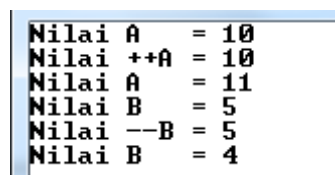
Gambar 3. 3 Hasil Contoh 3

Contoh-4

```
/* Penggunaan Notasi Dibelakang Variabel*/
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main( )
{
  int a = 10, b = 5;
  clrscr( );

  printf("Nilai A = %d", a);
  printf("\nNilai ++A = %d", a++);
  printf("\nNilai A = %d", a);
  printf("\nNilai B = %d", b);
  printf("\nNilai --B = %d", b--);
  printf("\nNilai B = %d", b);
  getch( );
}
```

Output yang akan dihasilkan, dari program contoh-4 diatas adalah:



Gambar 3. 4 Hasil Contoh 4

3.4. Operator Relasi

Operator Relasi digunakan untuk membandingkan dua buah nilai. Hasil perbandingan operator ini menghasilkan nilai numerik 1 (*True*) atau 0 (*False*).

Tabel. 3.5. Tabel Operator Relasi

Operator	Keterangan
==	Sama Dengan (bukan pemberi nilai)
!=	Tidak Sama dengan
>	Lebih Dari
<	Kurang Dari
>=	Lebih Dari sama dengan
<=	Kurang Dari sama dengan

Contoh-5

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

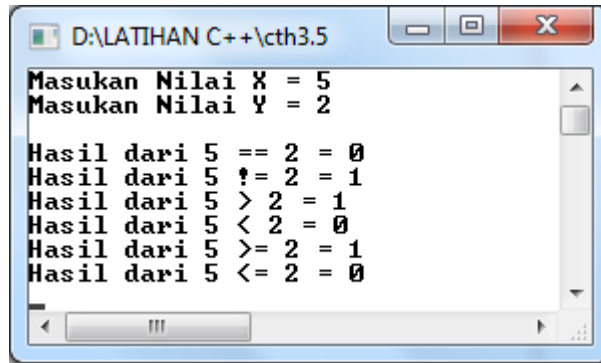
main( )
{
    float a, b, c, d, e, f, x, y;
    clrscr( );

    cout<<"Masukan Nilai X = "; cin>>x;
    cout<<"Masukan Nilai Y = "; cin>>y;

    a = x == y;
    b = x != y;
    c = x > y;
    d = x < y;
    e = x >= y;
    f = x <= y;

    cout<<endl;
    cout<<"Hasil dari "<<x<<" == "<<y<<" = "<<a<<endl;
    cout<<"Hasil dari "<<x<<" != "<<y<<" = "<<b<<endl;
    cout<<"Hasil dari "<<x<<" > "<<y<<" = "<<c<<endl;
    cout<<"Hasil dari "<<x<<" < "<<y<<" = "<<d<<endl;
    cout<<"Hasil dari "<<x<<" >= "<<y<<" = "<<e<<endl;
    cout<<"Hasil dari "<<x<<" <= "<<y<<" = "<<f<<endl;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-5 diatas adalah:



Gambar 3. 5 Hasil Contoh 5

3.5. Operator Logika

Operator Relasi digunakan untuk menghubungkan dua buah operasi relasi menjadi sebuah ungkapan kondisi. Hasil dari operator logika ini menghasilkan nilai numerik 1 (*True*) atau 0 (*False*).

Tabel. 3.5. Tabel Operator Relasi

Operator	Keterangan
&&	Operator Logika AND
	Operator Logika OR
!	Operator Logika NOT

3.5.1. Operator Logika AND

Operator logika AND digunakan untuk menghubungkan dua atau lebih ekspresi relasi, akan dianggap BENAR, bila semua ekspresi relasi yang dihubungkan bernilai BENAR.

Tabel Logika And

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

Contoh : Ekspresi Relasi-1 $\rightarrow A + 4 < 10$

Ekspresi Relasi-2 $\rightarrow B > A + 5$

Ekspresi Relasi-3 $\rightarrow C - 3 \geq 4$

Penggabungan ketiga ekspresi relasi diatas menjadi;

$$A+4 < 10 \ \&\& \ B>A+5 \ \&\& \ C-3 \geq 4$$

Jika nilai $A = 3$; $B = 3$; $C = 7$, maka ketiga ekspresi tersebut mempunyai nilai:

- Ekspresi Relasi-1 $\rightarrow A + 4 < 10 \rightarrow 3 + 4 < 10 \rightarrow$ BENAR
- Ekspresi Relasi-2 $\rightarrow B > A + 5 \rightarrow 3 > 3 + 5 \rightarrow$ SALAH

- Ekspresi Relasi-3 $\rightarrow C - 3 \geq 4 \rightarrow 7 - 3 \geq 4 \rightarrow$ BENAR
Dari ekspresi relasi tersebut mempunyai nilai BENAR, maka
 $A+4 < 10 \ \&\& \ B>A+5 \ \&\& \ C-3 \geq 4 \rightarrow$ SALAH = 0

Contoh-6

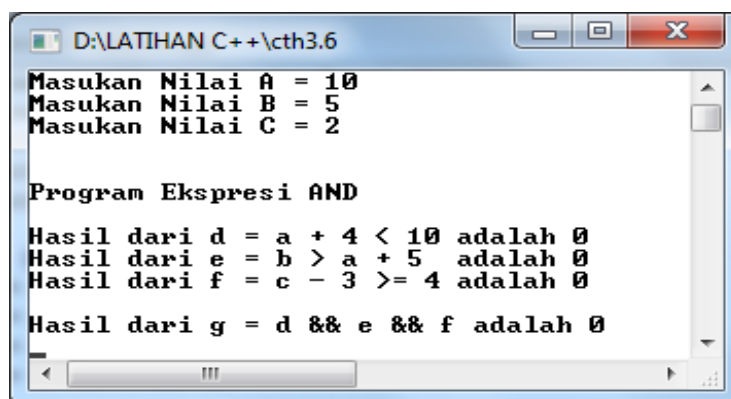
```
/* Penggunaan Operasi Logika AND */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main( )
{
    float a, b, c, d, e, f, g, h;
    clrscr();
    cout<<"Masukan Nilai A = "; cin>>a;
    cout<<"Masukan Nilai B = "; cin>>b;
    cout<<"Masukan Nilai C = "; cin>>c;

    // Proses

    d = a + 4 < 10;
    e = b > a + 5;
    f = c - 3 >= 4;
    g = d && e && f;

    cout<<endl<<endl;
    cout<<"Program Ekspresi AND"<<endl<<endl;
    cout<<"Hasil dari d = a + 4 < 10 adalah " <<d<<endl;
    cout<<"Hasil dari e = b > a + 5 adalah " <<e<<endl;
    cout<<"Hasil dari f = c - 3 >= 4 adalah " <<f;
    cout<<endl<<endl;
    cout<<"Hasil dari g = d && e && f adalah " <<g;
    cout<<endl;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah:



Gambar 3. 6 Hasil Contoh 6

3.5.2. Operator Logika OR

Operator logika OR digunakan untuk menghubungkan dua atau lebih ekspresi relasi, akan dianggap BENAR, bila salah satu ekspresi relasi yang dihubungkan bernilai BENAR dan bila semua ekspresi relasi yang dihubungkan bernilai SALAH, maka akan bernilai SALAH.

Tabel Logika Or

A	B	A B
T	T	T
T	F	T
F	T	T
F	F	F

Contoh Ekspresi Relasi-1 $\rightarrow A + 4 < 10$
 Ekspresi Relasi-2 $\rightarrow B > A + 5$
 Ekspresi Relasi-3 $\rightarrow C - 3 > 4$

Penggabungan ketiga ekspresi relasi diatas menjadi;
 $A + 4 < 10 \parallel B > A + 5 \parallel C - 3 > 4$

Jika nilai $A = 3$; $B = 3$; $C = 7$, maka ketiga ekspresi tersebut mempunyai nilai:

- Ekspresi Relasi-1 $\rightarrow A + 4 < 10 \rightarrow 3 + 4 < 10 \rightarrow$ BENAR
- Ekspresi Relasi-2 $\rightarrow B > A + 5 \rightarrow 3 > 3 + 5 \rightarrow$ SALAH
- Ekspresi Relasi-3 $\rightarrow C - 3 > 4 \rightarrow 7 - 3 > 4 \rightarrow$ SALAH

Dilihat ekspresi diatas salah satu ekspresi tersebut mempunyai nilai BENAR, maka ekspresi tersebut tetap bernilai BENAR.

$$A + 4 < 10 \parallel B > A + 5 \parallel C - 3 > 4 \rightarrow \text{BENAR} = 1$$

Contoh-7 /* Penggunaan Operasi Logika OR */

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

main()
{
    float a, b, c, d, e, f, g, h;
    clrscr();
    cout<<"Masukan Nilai A = "; cin>>a;
    cout<<"Masukan Nilai B = "; cin>>b;
    cout<<"Masukan Nilai C = "; cin>>c;

    d = a + 5 > 10;
    e = b > 5 + a ;
    f = c - 4 <= 7;
    g = d || e || f;

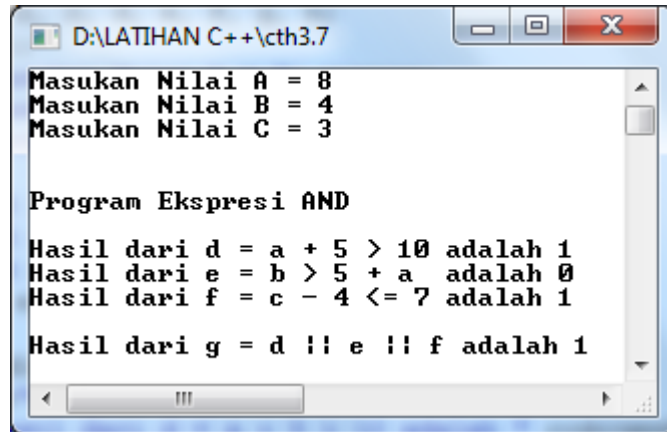
    cout<<endl<<endl;
```

```

cout<<"Program Ekspresi AND"<<endl<<endl;
cout<<"Hasil dari d = a + 5 > 10 adalah " <<d<<endl;
cout<<"Hasil dari e = b > 5 + a adalah " <<e<<endl;
cout<<"Hasil dari f = c - 4 <= 7 adalah " <<f;
cout<<endl<<endl;
cout<<"Hasil dari g = d || e || f adalah " <<g;
cout<<endl;
getch();
}

```

Output yang akan dihasilkan, dari program contoh-7 diatas adalah :



Gambar 3.7 Hasil Contoh 7

Contoh-8

```

/* Penggunaan Operasi Logika AND OR*/
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
float a, b, c, d, e, f, g, h;

clrscr();
cout<<"Masukan Nilai A = "; cin>>a;

cout<<"Masukan Nilai B = "; cin>>b;
cout<<"Masukan Nilai C = "; cin>>c;

// Proses
d = a + 4 < 10;
e = b > a + 5;
f = c - 3 >= 4;
g = d && e && f;

cout<<endl<<endl;
cout<<"Program Ekspresi AND / OR"<<endl<<endl;
cout<<"Hasil dari d = a + 4 < 10 adalah " <<d<<endl;
cout<<"Hasil dari e = b > a + 5 adalah " <<e<<endl;

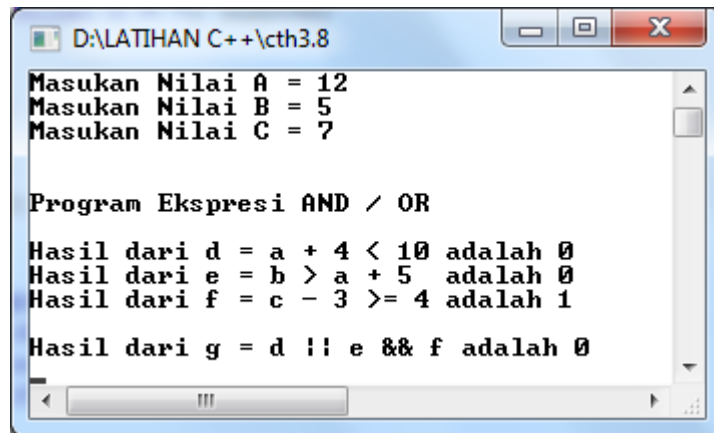
```

```

        cout<<"Hasil dari f = c - 3 >= 4 adalah " <<f;
        cout<<endl<<endl;
        cout<<"Hasil dari g = d || e && f adalah " <<g;
        cout<<endl;
        getch();
    }

```

Output yang akan dihasilkan, dari program contoh-8 diatas adalah :



Gambar 3. 8 Hasil Contoh 8

3.5.3. Operator Logika NOT

Operator logika NOT akan memberikan nilai kebalikkan dari ekspresi yang disebutkan. Jika nilai yang disebutkan bernilai BENAR maka akan menghasilkan nilai SALAH, begitu pula sebaliknya.

Contoh :

Ekspresi Relasi → $A + 4 < 10$

Penggunaan Operator Logika NOT diatas menjadi;

$$!(A+4 < 10)$$

Jika nilai $A = 3$; maka ekspresi tersebut mempunyai nilai:

- Ekspresi Relasi-1 → $A + 4 < 10 \rightarrow 3 + 4 < 10 \rightarrow \text{BENAR}$

Dilihat ekspresi diatas salah satu ekspresi tersebut mempunyai nilai BENAR dan jika digunakan operator logika NOT, maka ekspresi tersebut akan bernilai SALAH

$$!(A+4 < 10) \rightarrow !(\text{BENAR}) = \text{SALAH} = 0$$

Contoh-9

```

/* Penggunaan Operasi Logika NOT */
#include <stdio.h>
#include <conio.h>
#include<iostream.h>

main( )
{

```

```

int a, b, c;
clrscr();

cout<<"Masukan Nilai A  = ";
cin>>a;

/* Proses */
b = (a + 4 < 10);
c = !(b);

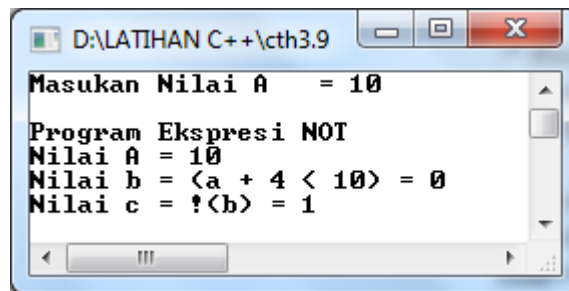
cout<<endl<<"Program Ekspresi NOT "<<endl;

cout<<"Nilai A = "<<a<<endl;
cout<<"Nilai b = (a + 4 < 10) = "<<b<<endl;
cout<<"Nilai c = !(b) = "<<c;

getch();
}

```

Output yang akan dihasilkan, dari program contoh-8 diatas adalah :



Gambar 3. 9 Hasil Contoh 9

3.6. Operator Bitwise

Operator Bitwise digunakan untuk memanipulasi data dalam bentuk bit. Borland C++ menyediakan enam buah operator bitwise.

Tabel. 3.6. Tabel Operator Bitwise

Operator	Keterangan
~	Bitwise NOT
<<	Bitwise Shift Left
>>	Bitwise Shift Right
&	Bitwise AND
^	Bitwise XOR
	Bitwise OR

3.6.2. Operator Bitwise >> (Shift Right)

Operator Bitwise Shift Right digunakan untuk menggeser sejumlah bit kanan.

Contoh :

```
0000000011001001 = 201
      ||||| → digeser 1 bit ke kanani
000000001100100 = 100
```

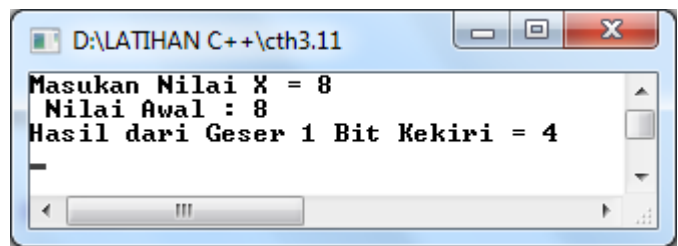
Dibagian kanan disisipkan 0, sebanyak bit yang digeser

Contoh-11

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
void main( )
{
    int x;
    clrscr( );
    cout<<"Masukan Nilai X = ";
    cin>>x;
    cout<<" Nilai Awal : "<<x<<endl;
    x = x >> 1;

    cout<<"Hasil dari Geser 1 Bit Kekiri = "<<x<<endl;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-11 diatas adalah :



Gambar 3. 11 Hasil Contoh 11

3.6.3. Operator Bitwise & (And)

Operator Bitwise & (And) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar (1) jika semua operand yang digabungkan bernilai benar (1). Berikut dapat dilihat ilustrasi untuk membandingkan bit dari 2 operand.

Tabel. 3.7. Tabel Operator Bitwise And

Bit Operand 1	Bit Operand 2	Hasil Operand
0	0	0
0	1	0
1	0	0
1	1	1

Contoh :

```

11001001 = 201
01100100 = 100
-----
01000000 = 64
    
```

AND

Contoh-12

```

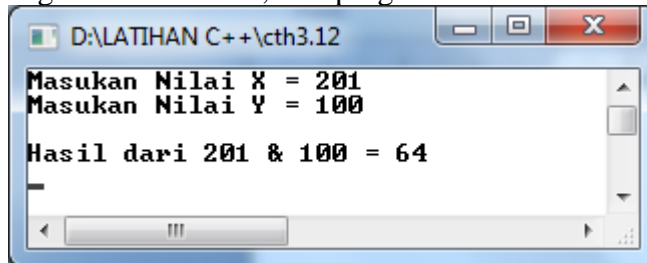
#include<iostream.h>
#include<stdio.h>
#include<conio.h>

void main( )
{
    int a, x, y;
    clrscr();

    cout<<"Masukan Nilai X = ";
    cin>>x;
    cout<<"Masukan Nilai Y = ";
    cin>>y;
    a = x & y;

    cout<<'\n';
    cout<<"Hasil dari "<<x<<" & "<<y<<" = "<<a<<endl;
    getch();
}
    
```

Output yang akan dihasilkan, dari program contoh-12 diatas adalah :



Gambar 3. 12 Hasil Contoh 12

3.5.4. Operator Bitwise | (Or)

Operator Bitwise | (Or) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar jika ada salah satu operand yang digabungkan ada yang bernilai benar (1). Berikut dapat dilihat ilustrasi untuk membandingkan bit dari 2 operand.

Tabel. 3.8. Tabel Operator Bitiwise Or

Bit Operand 1	Bit Operand 2	Hasil Operand
0	0	0
0	1	1
1	0	1
1	1	1

Contoh :

$$\begin{array}{r}
 11001001 = 201 \\
 01100100 = 100 \\
 \hline
 11101101 = 237
 \end{array}
 \quad \text{OR}$$

Contoh-13

```

#include<iostream.h>
#include<stdio.h>
#include<conio.h>

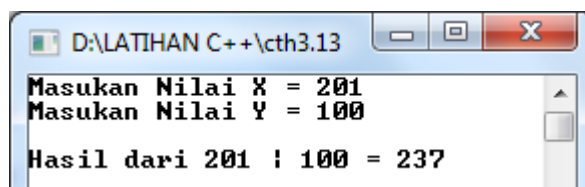
void main( )
{
    int a, x, y;
    clrscr();

    cout<<"Masukan Nilai X = ";
    cin>>x;
    cout<<"Masukan Nilai Y = ";
    cin>>y;

    a = x | y;

    cout<<'\n';
    cout<<"Hasil dari "<<x<<" | "<<y<<" = "<<a<<endl;
    getch();
}
    
```

Output yang akan dihasilkan, dari program contoh-13 diatas adalah :



Gambar 3. 13 Hasil Contoh 13

3.6.5. Operator Bitwise ^ (eXclusive Or)

Operator Bitwise ^ (XOr) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar (1) jika dari dua bit yang dibandingkan hanya sebuah bernilai benar (1). Berikut dapat dilihat ilustrasi untuk membandingkan bit dari 2 operand.

Tabel. 3.9. Tabel Operator Bitwise XOr

Bit Operand 1	Bit Operand 2	Hasil Operand
0	0	0
0	1	1
1	0	1
1	1	0

Contoh :

```

11001001 = 201
01100100 = 100
----- XOR
10101101 = 137
    
```

Contoh-14

```

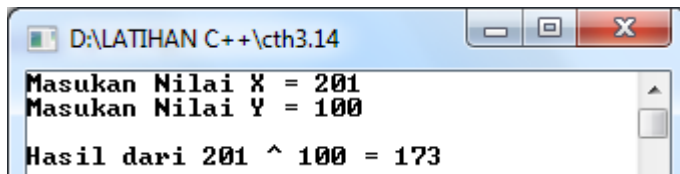
#include<iostream.h>
#include<stdio.h>
#include<conio.h>

void main( )
{
    int a, x, y;
    clrscr( );
    cout<<"Masukan Nilai X = ";
    cin>>x;
    cout<<"Masukan Nilai Y = ";
    cin>>y;

    a = x ^ y;

    cout<<"\n";
    cout<<"Hasil dari "<<x<<" ^ "<<y<<" = "<<a<<endl;
    getch( );
}
    
```

Output yang akan dihasilkan, dari program contoh-14 diatas adalah:



3.6.5. Operator Bitwise (Not) Gambar Hasil Contoh-14

Operator Bitwise ~ (Not) digunakan membalik nilai bit dari suatu operand. Berikut dapat dilihat ilustrasi untuk membandingkan bit dari 2 operand.

Tabel. 3.10. Tabel Operator Bitwise Not

Bit Operand	Hasil
0	1
1	0

Contoh :

```

00001000 = 8
|||||||
11110111 = 247 = -9
    
```

Contoh-15

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

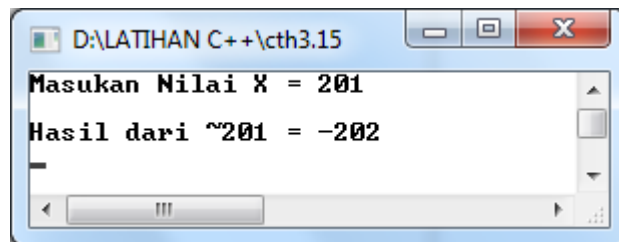
void main()
{
    int a, x, y;
    clrscr();
    cout<<"Masukan Nilai X = ";
    cin>>x;

    a = ~x;

    cout<<'\n';

    cout<<"Hasil dari ~" <<x<<" = " <<a<<endl;
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-15 diatas adalah :



Gambar 3. 14 Hasil Contoh 15

3.7. Operasi String

Operasi string selalu dijumpai didalam bahasa pemrograman, dikarenakan hampir semua bahasa pemrograman menggunakan manual *inputnya* adalah string, terutama pada pemrograman visualisasi. Pada bab ini akan dibahas beberapa perintah dan fungsi string.

3.7.1. Fungsi Manipulasi String

Borland C++ menyediakan beberapa fungsi yang digunakan untuk keperluan memanipulasi string, diantaranya:

a. Fungsi `strcat()`

Fungsi ini digunakan untuk menambahkan string sumber kebagian akhir dari string tujuan. File header yang harus disertakan adalah **string.h** dan **ctype.h**

Bentuk Penulisan :

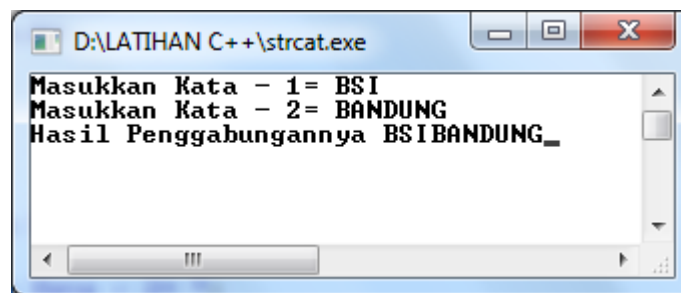
```
strcat(tujuan, sumber);
```

```

Contoh-16   #include <string.h>
                #include <ctype.h>
                #include <iostream.h>

                main()
                {
                    char a1[20];
                    char a2[20];
                    clrscr();
                    cout<<"Masukkan Kata - 1= ";
                    cin>>a1;
                    cout<<"Masukkan Kata - 2= ";
                    cin>>a2;
                    strcat(a1, a2);
                    cout<<"Hasil Penggabungannya "<<a1;
                    getch();
                }
    
```

Output yang akan dihasilkan, dari program contoh-16 diatas adalah:



Gambar 3.15 Hasil Contoh-16

b. Fungsi strcmp()

Fungsi ini digunakan untuk membandingkan string pertama dengan string kedua. Hasil dari fungsi ini bertipe data integer (int). File header yang harus disertakan adalah **string.h**

Bentuk Penulisan :

```
var_int = strcmp(str1, str2);
```

```

Contoh-17   #include <string.h>
                #include <iostream.h>
                #include <conio.h>
                main()
                {
                    char a1[ ] = "BSI";
                    char a2[ ] = "Bsi";
                    char b1[ ] = "BSI";
                    clrscr();
                    cout<<"Hasil Perbandingan "<<a1<<" dan "<<a2<<"->";
    
```

```
cout<<strcmp(a1,a2)<<endl;
cout<<"Hasil Perbandingan "<<a2<<" dan "<<a2<<"->";
cout<<strcmp(a2,a1) <<endl;
cout<<"Hasil Perbandingan "<<a1<<" dan "<<b1<<"->";
cout<<strcmp(a1,b1) <<endl;
getch();
}
```

Output yang akan dihasilkan, dari program contoh-17 diatas adalah:

Gambar 3.1.6. Hasil Contoh-17

c. Fungsi strcpy()

Fungsi ini digunakan untuk menyalin string asal ke-variabel string tujuan, dengan syarat string tujuan harus mempunyai tipe data dan dan ukuran yang sama dengan string asal. File header yang harus disertakan adalah **string.h**.

Bentuk Penulisan :

```
strcpy(tujuan, asal);
```

Contoh-18

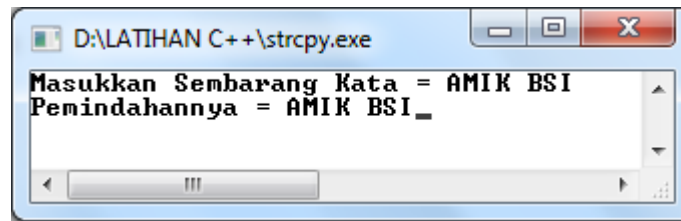
```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <iostream.h>
```

```
main()
{
    char huruf[20];
    char pindah[20];
    clrscr();

    cout<<"Masukkan Sembarang Kata = ";
    gets(huruf);

    /* Proses */
    strcpy(pindah, huruf);
    cout<<"Pemindahannya = "<<pindah;
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-18 diatas adalah:



Gambar 3.17 Hasil Contoh-18

d. Fungsi strlen()

Fungsi ini digunakan untuk memperoleh banyaknya karakter dalam string. File header yang harus disertakan adalah **string.h**

Bentuk Penulisan :

strlen(str);

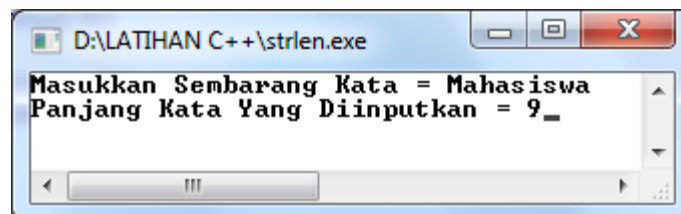
Contoh-19

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

main( )
{
    char huruf[20];
    char pindah[20];
    clrscr();

    cout<<"Masukkan Sembarang Kata = ";
    gets(huruf);
    cout<<"Panjang Kata Yang Diinputkan = ";
    cout<<strlen(huruf);
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-19 diatas adalah:



Gamabr 3.18 Hasil Contoh-19

e. Fungsi strrev()

Fungsi ini digunakan untuk membalik letak urutan pada string. String urutan paling akhir dipindahkan ke urutan paling depan dan seterusnya. File header yang harus disertakan adalah **string.h**

Bentuk Penulisan :

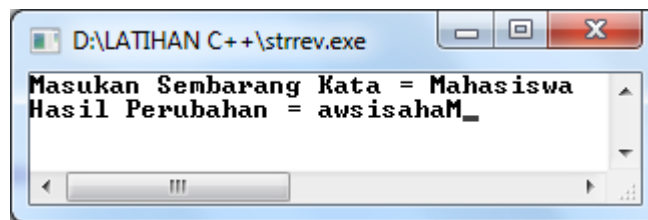
```
strrev(str);
```

```
Contoh-20 #include <stdio.h>
#include <conio.h>
#include <string.h>
#include <iostream.h>

main()
{
    char kata[20];
    clrscr();
    cout<<"Masukan Sembarang Kata = ";
    gets(kata);

    strrev(kata);
    cout<<"Hasil Perubahan = "<<kata;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-20 diatas adalah:



Gambar 3.19 Hasil Contoh-20

3.7.2. Fungsi Konversi String

Borland C++ 5.02 menyediakan beberapa fungsi yang digunakan untuk keperluan konfersi string.

a. Fungsi atof()

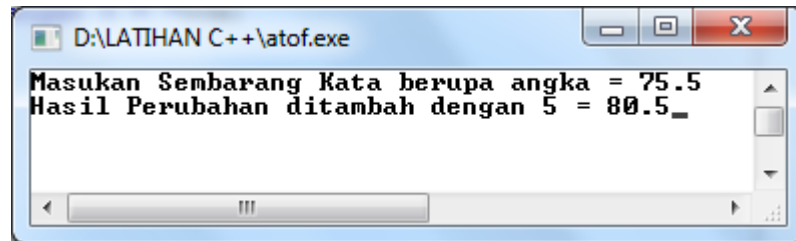
Fungsi ini digunakan untuk mengubah string (teks) angka menjadi bilangan numerik float. File header yang harus disertakan adalah **math.h**

```
Contoh-21 #include <stdio.h>
#include <conio.h>
#include <math.h>
#include <iostream.h>

main()
{
    char kata[20];
    float angka, a, b;
    clrscr( );
    cout<<"Masukan Sembarang Kata berupa angka = ";
```

```
    gets(kata);
    angka = atof(kata);
    a = angka + 5;
    cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-21 diatas adalah:



Gambar 3.20 Hasil Contoh-21

b. Fungsi atoi()

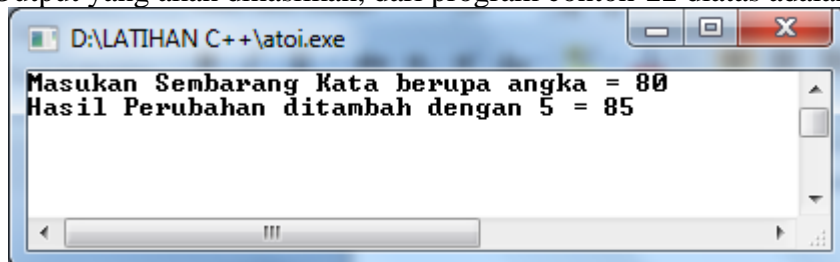
Fungsi ini digunakan untuk mengubah string (teks) angka menjadi bilangan numerik integer. File header yang harus disertakan adalah **stdlib.h**

```
Contoh-22    #include <stdio.h>
               #include <conio.h>
               #include <stdlib.h>
               #include <iostream.h>
```

```
main( )
{
    char kata[20];
    float angka, a, b;
    clrscr( );

    cout<<"Masukan Sembarang Kata berupa angka = ";
    gets(kata);
    angka = atoi(kata);
    a = angka + 5;
    cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-22 diatas adalah :



Gambar 3-21. Hasil Contoh-22

c. Fungsi atol()

Fungsi ini digunakan untuk mengubah string (teks) angka menjadi bilangan numerik long integer. File header yang harus disertakan adalah **stdlib.h**

```

Contoh-23   #include <stdio.h>
                #include <conio.h>
                #include <stdlib.h>
                #include <iostream.h>

                main()
                {
                    char kata[20];
                    float angka, a, b;

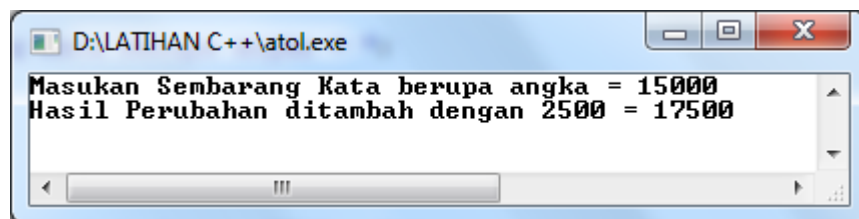
                    clrscr( );

                    cout<<"Masukan Sembarang Kata berupa angka = ";
                    gets(kata);

                    angka = atol(kata);
                    a = angka + 2500;

                    cout<<"Hasil Perubahan ditambah dengan 2500 = "<<a;
                    getch( );
                }
    
```

Output yang akan dihasilkan, dari program contoh-23 diatas adalah:



Gambar 3.22. Hasil Contoh-23

d. Fungsi strlwr()

Fungsi ini digunakan untuk mengubah setiap huruf kapital (huruf besar) dalam string menjadi huruf kecil. File header yang harus disertakan adalah **string.h**
Bentuk Penulisan :

```
strlwr(str);
```

```

Contoh-24   #include <stdio.h>
                #include <conio.h>
                #include <string.h>
                #include <iostream.h>
    
```

```

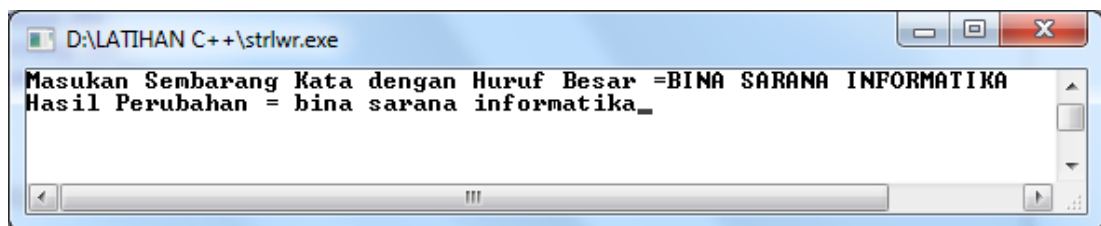
main()
{
    char kata[20];
    clrscr();

    cout<<"Masukan Sembarang Kata dengan Huruf Besar =";
    gets(kata);
    strlwr(kata);

    cout<<"Hasil Perubahan = "<<kata;
    getch();
}

```

Output yang akan dihasilkan, dari program contoh-24 diatas adalah:



Gambar 3.23. Hasil Contoh-24

e. Fungsi `strupr()`

Fungsi ini digunakan untuk mengubah setiap huruf kecil dalam string menjadi huruf kapital (huruf besar). File header yang harus disertakan adalah **string.h**

Bentuk Penulisan :

```
strupr(str);
```

Contoh-25

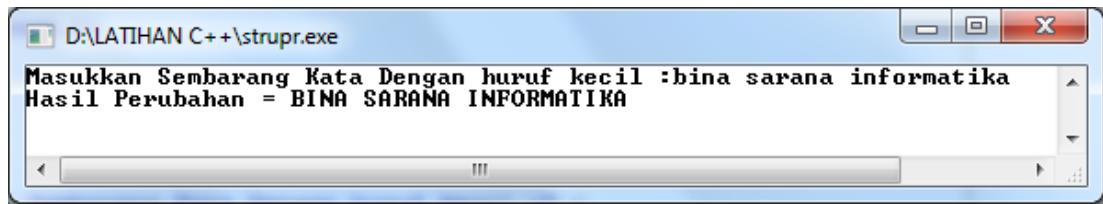
```

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <iostream.h>

main()
{
    char kata[20];
    clrscr();
    cout<<"Masukkan Sembarang Kata dengan Huruf Besar: ";
    gets(kata);
   strupr(kata);
    cout<<"Hasil Perubahan : "<<kata<<endl;
    getch();
}

```

Output yang akan dihasilkan, dari program contoh-25 diatas adalah :



Gambar 3-24 Hasil Contoh-25

3.8. Tugas 2

1. Tentukan apa hasil numerik dari ekspresi relasi dan logika dibawah ini. Diberikan nilai A = 3; B = 6 ; C = 2 ; K = 5; L = 4; M = 3

- a. $D = (4 + 2 > A \ \&\& \ B - 2 > 3 + 2 \ \parallel \ B + 2 \leq 6 + 2)$
- b. $D = K + 5 < M \ \parallel \ (C * M < L \ \&\& \ 2 * M - L > 0)$
- c. $D = L + 5 < M \ \parallel \ C * K < L \ \&\& \ 2 * K - L > 0$
- d. $D = A * 4 \leq 3 * M + B$
- e. $D = K + 10 > A \ \&\& \ L - 2 > 4 * C$

2. Dari program dibawah ini, bagaimanakah keluaran yang dihasilkan

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

main( )
{
    int a = 21;
    clrscr( );
    printf("\n Nilai a = %d",a);
    printf("\n Nilai a++ = %d",a++);
    printf("\n Nilai ++a = %d",++a);
    printf("\n Nilai --a = %d",--a);
    printf("\n Nilai a = %d",a);
    a+=3;
    printf("\n Nilai a = %d",a);
    printf("\n Nilai ++a = %d",++a);
    printf("\n Nilai a++ = %d",a++);
    printf("\n Nilai --a = %d",--a);
    printf("\n Nilai a-- = %d",a--);

    getch( );
}
```

3. Buatlah program untuk menghitung panjang kata berikut ini:
Akademi Manajemen Informatika dan Komputer Bina Sarana Informatika

4. Buatlah program untuk membalik kata berikut ini :
Akademi Manajemen Informatika dan Komputer Bina Sarana Informatika

Menjadi seperti berikut :

akitamrofni anaraS aniB retupmoK and akitamrofni nemejanaM imedakA

5. Buatlah program untuk menggabungkan dua buah string
Kalimat1 = Manajemen
Kalimat2 = Informatika

Menjadi seperti berikut:

ManajemenInformatika

6. Diberikan kalimat string berikut :
Kalimat1 = "35.6"
Kalimat2 = "12.5"

Kemudian kedua kalimat diatas dihitung menjadi perhitungan :

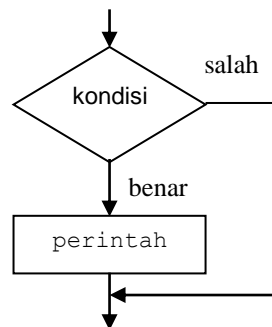
- a. Perkalian
- b. Pembagian
- c. Penambahan
- d. Pengurangan
- e. Mencari sisa hasil pembagian

Penyeleksian Kondisi

Pernyataan Percabangan digunakan untuk memecahkan persoalan untuk mengambil suatu keputusan diantara sekian pernyataan yang ada. Untuk keperluan pengambilan keputusan, Borland C++ menyediakan beberapa perintah antara lain.

4.1. Pernyataan IF

Pernyataan *if* mempunyai pengertian, “ *Jika kondisi bernilai benar, maka perintah akan dikerjakan dan jika tidak memenuhi syarat maka akan diabaikan*”. Dari pengertian tersebut dapat dilihat dari diagram alir berikut:



Gambar 4.1. Diagram Alir IF

Bentuk umum dari pernyataan *if*

```
if (kondisi)
pernyataan;
```

Penulisan *kondisi* berada di dalam tanda kurung kurawal jika pemakaian *if* diikuti dengan pernyataan majemuk, bentuk penulisannya sebagai berikut :

```
if (kondisi)
{
    pernyataan;
    .....
}
```

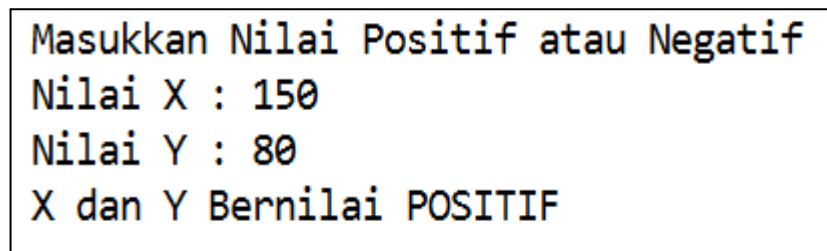
Contoh-1

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main( )
{
    float x,y;
    clrscr( );

    cout<<" Masukkan Nilai Positif atau Negatif "<<endl;
    cout<<" Nilai X : ";cin>>x;
    cout<<" Nilai Y : ";cin>>y;

    if ((x >= 0 ) && (y>=0))
        cout<<"X dan Y Bernilai POSITIF "<<endl;
    getch( );
}
```

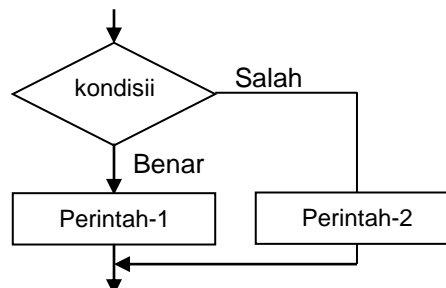
Output yang akan dihasilkan, dari program contoh-1 diatas adalah:



Gambar 4. 2 Hasil Contoh 1

4.1.1. Pernyataan IF - ELSE

Pernyataan *if* mempunyai pengertian, “*Jika kondisi bernilai benar, maka perintah-1 akan dikerjakan dan jika tidak memenuhi syarat maka akan mengerjakan perintah-2*”. Dari pengertian tersebut dapat dilihat dari diagram alir berikut:



Gambar 4.3 Diagram Alir if-else

Bentuk umum dari pernyataan *if-else*

```
if (kondisi)
    perintah-1;
else
    perintah-2;
```

Perintah-1 dan perintah-2 dapat berupa sebuah pernyataan tunggal, pernyataan majemuk atau pernyataan kosong. Jika pemakaian *if-else* diikuti dengan pernyataan majemuk, bentuk penulisannya sebagai berikut :

```
if (kondisi)
{
    perintah-1;
    ...
}
else
{
    perintah-2;
    ...
}
```

Contoh

Menentukan besarnya diskon dan bonus dari jumlah beli, dengan kriteria :

- jika jumlah beli > 15 maka diskon 5% dan bonusnya : payung
- selain itu tidak dapat diskon dan bonus

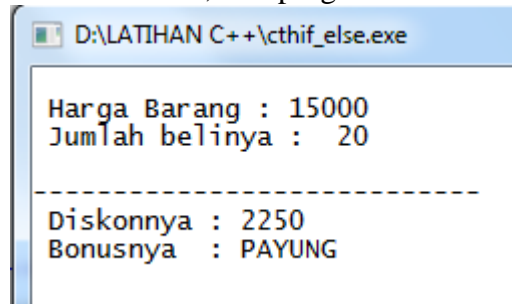
```
contoh-2    #include <stdio.h>
              #include <conio.h>
              #include <iostream.h>

              main()
              {
                float jumbel,hrg;
                char bonus[15];
                float diskon;

                clrscr( );
                puts("");
                cout<<" Harga Barang : " ; cin>> hrg;
                cout << " Jumlah belinya : " ; cin>> jumbel;
                if(jumbel>=15)
                { diskon = 0.15 * hrg ;
                  strcpy(bonus, "PAYUNG");
                }
                else
                { diskon =0 ;
```

```
strcpy(bonus, "Tidak Dapat");
}
cout<<"\n-----"<<endl;
cout<<" Diskonnya : "<<diskon<<endl;
cout<<" Bonusnya : "<<bonus<<endl;
getch();
}
```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah :



Gambar 4.4 Hasil Contoh 2

4.1.2. Pernyataan NESTED IF

Nested if merupakan pernyataan if berada didalam pernyataan if yang lainnya. Bentuk penulisan pernyataan Nested if adalah :

```
if(syarat)
{
    if(syarat)
        ... perintah;
    else
        ... perintah;
}
else
{
    if(syarat)
        ... perintah;
    else
        ... perintah;
}
```

Contoh

Suatu perusahaan menjual pakaian dengan ketentuan sebagai berikut:

- Jika kode baju=1 maka Merk Baju = H&R, dengan ukuran baju=S, maka harganya 45000, Jika ukuran baju=M, maka harganya 60000, selain itu harganya = 0.
- Jika kode baju=2 maka Merk Baju = Adidas, dengan ukuran baju=S, maka harganya 65000, Jika ukuran Baju=M, maka harganya 75000, selain itu harganya = 0.
- Selain kode baju diatas, maka salah kode

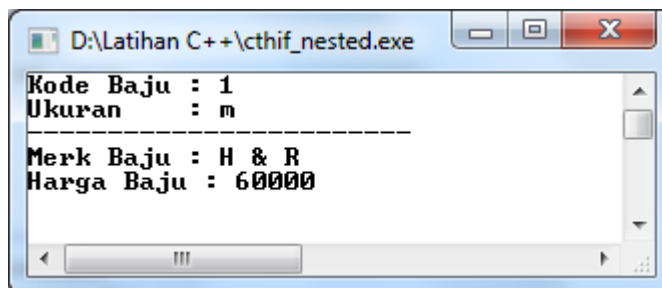
Contoh-3 #include<stdio.h>
 #include<conio.h>

```
#include<iostream.h>

main( )
{
    char kode,ukuran,merk[15];
    long harga=0;
    clrscr( );
    cout<<"Kode Baju : ";cin>>kode;
    cout<<"Ukuran  : ";cin>>ukuran;

    if (kode == '1')
    {
        strcpy(merk,"H & R");
        if (ukuran== 'S' || ukuran == 's')
            harga=45000;
        else if (ukuran== 'M' || ukuran == 'm')
            harga=60000;
        else
            harga = 0;
    }
    else if (kode=='2')
    {
        strcpy(merk," Adidas");
        if (ukuran=='S' || ukuran == 's')
            harga=65000;
        else if (ukuran== 'M' || ukuran == 'm')
            harga=75000;
        else
            harga = 0;
    }
    else
        cout<<"Salah Kode Baju"<<endl;
    cout<<"-----"<<endl;
    cout<<"Merk Baju : "<<merk<<endl;
    cout<<"Harga Baju : "<<harga<<endl;
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah :



```
D:\Latihan C++\cthif_nested.exe
Kode Baju : 1
Ukuran : m
-----
Merk Baju : H & R
Harga Baju : 60000
```

Gambar 4. 5 Hasil Contoh 3

4.1.3. Pernyataan IF – ELSE Majemuk

Bentuk dari *if-else* bertingkat sebenarnya serupa dengan *nested if*, keuntungan penggunaan *if-else* bertingkat dibanding dengan *nested if* adalah penggunaan bentuk penulisan yang lebih sederhana.

Bentuk Umum Penulisannya:

```

if (syarat)
{
    ... perintah;
    ... perintah;
}
else if (syarat)
{
    ... perintah;
    ... perintah;
}
else
{
    ... perintah;
    ... perintah;
}
    
```

Contoh

Suatu perusahaan memberikan komisi kepada para salesman dengan ketentuan sebagai berikut:

- Bila salesman dapat menjual barang hingga Rp. 200.000 ,- , akan diberikan uang jasa sebesar Rp. 10.000 ditambah dengan uang komisi Rp. 10% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 200.000 ,- , akan diberikan uang jasa sebesar Rp. 20.000 ditambah dengan uang komisi Rp. 15% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 500.000 ,- , akan diberikan uang jasa sebesar Rp. 30.000 ditambah dengan uang komisi Rp. 20% dari pendapatan yang diperoleh hari itu.

Contoh-4

```

#include<stdio.h>
#include<conio.h>
#include<iostream.h>

main( )
{
    float pendapatan, jasa=0, komisi=0, total=0;
    clrscr( );
    cout<<"Pendapatan Hari ini Rp. ";
    cin>>pendapatan;

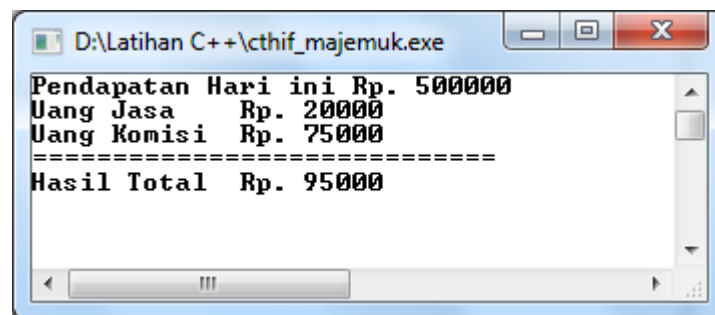
    if (pendapatan >= 0 && pendapatan <= 200000)
    
```

```

{
    jasa=10000;
    komisi=0.1*pendapatan;
}
else if(pendapatan<=500000)
{
    jasa=20000;
    komisi=0.15*pendapatan;
}
else
{
    jasa=30000;
    komisi=0.2*pendapatan;
}
/* menghitung total */
total = komisi+jasa;
cout<<"Uang Jasa Rp. "<<jasa<<endl;
cout<<"Uang Komisi Rp. "<<komisi<<endl;
cout<<"====="<<endl;
cout<<"Hasil Total Rp. "<<total<<endl;
getch( );
}

```

Output yang akan dihasilkan, dari program contoh-4 diatas adalah :



Gambar 4. 6 Hasil Contoh 4

4.2. Pernyataan switch - case

Bentuk dari *switch - case* merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif penyelesaian. Pernyataan *switch - case* ini memiliki kegunaan sama seperti *if - else* bertingkat, tetapi penggunaannya untuk memeriksa data yang bertipe **karakter** atau **integer**. Bentuk penulisan perintah ini sebagai berikut:

```
switch (ekspresi integer atau karakter )
{
    case konstanta-1 :
        ... perintah;
        ... perintah;
        break;
    case konstanta-2 :
        ... perintah;
        ... perintah;
        break;
    default :
        ... perintah;
        ... perintah;
}
```

Setiap pilihan akan dijalankan jika syarat nilai konstanta tersebut dipenuhi dan **default** akan dijalankan jika semua cabang di atasnya tidak terpenuhi. Pernyataan *break* menunjukkan bahwa perintah siap keluar dari *switch*. Jika pernyataan ini tidak ada, maka program akan diteruskan ke pilihan-pilihan yang lainnya.

Contoh-5

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

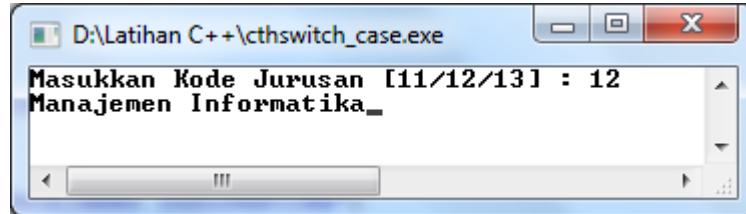
main( )
{
    int kode;
    clrscr( );
    cout<<"Masukkan Kode Jurusan [11/12/13] : ";
    cin>>kode;

    switch(kode)
    {
        case 11 :
            cout<<"Komputerisasi Akuntansi";
            break;
        case 12 :
            cout<<"Manajemen Informatika";
            break;

        case 13 :
            cout<<"Tehnik Komputer";
            break;
        default:
            cout<<"Anda Salah Memasukan kode";
            break;
    }
}
```

```
    }  
    getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-5 diatas adalah :

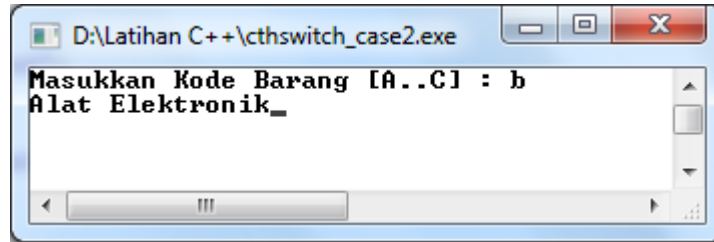


Gambar 4.7 Hasil Contoh 5

Contoh-6

```
#include<stdio.h>  
#include<conio.h>  
#include<iostream.h>  
  
main( )  
{  
    char kode;  
    clrscr( );  
    cout<<"Masukkan Kode Barang [A..C] : ";  
    cin>>kode;  
  
    switch(kode)  
    {  
        case 'A' :  
        case 'a' :  
            cout<<"Alat Olah Raga";  
            break;  
        case 'B' :  
        case 'b' :  
            cout<<"Alat Elelektronik";  
            break;  
        case 'C' :  
        case 'c' :  
            cout<<"Alat Masak";  
            break;  
  
        default:  
            cout<<"Anda Salah Memasukan kode";  
            break;  
    }  
    getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah :



Gambar 4.8 Hasil Contoh 6

4.3. Tugas 3

1. Buatlah program untuk menghitung nilai rata-rata dari seorang siswa, dengan ketentuan sebagai berikut :
 - Nama Siswa, Nilai Pertandingan I, Nilai Pertandingan II, Nilai Pertandingan III diinput.
 - Nilai Rata-rata merupakan hasil dari Nilai Pertandingan I, II dan III dibagi dengan 3.
 - Ketentuan Juara
 - Jika nilai rata-rata yang dihasilkan lebih besar dari 80, maka menjadi Juara I
 - Jika nilai rata-rata yang dihasilkan lebih besar dari 75, maka menjadi Juara II
 - Jika nilai rata-rata yang dihasilkan lebih besar dari 65, maka menjadi Juara III
 - Selain itu tidak juara
 - Tampilan yang diinginkan sebagai berikut:

Layar Masukkan

PROGRAM HITUNG NILAI RATA-RATA

Nama Siswa :
Nilai Pertandingan I :
Nilai Pertandingan II :
Nilai Pertandingan III :

Layar Keluaran

Siswa yang bernama ...
Memperoleh nilai rata-rata ... dan menjadi juara ke-... dari hasil perlombaan yang diikutinya.

2. Buatlah program untuk menghitung nilai akhir seorang siswa dari kursus yang diikutinya. Dengan ketentuan sebagai berikut :
 - Nama Siswa, Nilai Keaktifan, Nilai Tugas dan Nilai Ujian diinput.

- Proses yang dilakukan untuk mendapatkan nilai murni dari masing-masing nilai, adalah
- Nilai Murni Keaktifan = Nilai Keaktifan dikalikan dengan 20%.
- Nilai Murni Tugas = Nilai Tugas dikalikan dengan 30%
- Nilai Murni Ujian = Nilai Ujian dikalikan dengan 50%
- Nilai Akhir adalah Nilai Murni Keaktifan + Nilai Murni Tugas + Nilai Murni Ujian
- Ketentuan Nilai Huruf
 - Jika nilai Akhir yang dihasilkan lebih besar dari 80, maka mendapat grade “A”
 - Jika nilai Akhir yang dihasilkan lebih besar dari 70, maka mendapat grade “B”
 - Jika nilai Akhir yang dihasilkan lebih besar dari 56, maka mendapat grade “C”
 - Jika nilai Akhir yang dihasilkan lebih besar dari 46, maka mendapat grade “D”
 - Selain itu mendapat grade “E”
- Tampilan yang diinginkan sebagai berikut :

Layar Masukkan

PROGRAM HITUNG NILAI AKHIR

Nama Siswa : ...
Nilai Keaktifan : ...
Nilai Tugas : ...
Nilai Ujian : ...

Layar Keluaran

Siswa yang bernama
Dengan Nilai Persentasi Yang dihasilkan.
Nilai Keaktifan * 20% : ...
Nilai Tugas * 30% : ...
Nilai Ujian * 50% : ...

Jadi Siswa yang bernama ... memperoleh nilai akhir sebesar dengan grade ...

3. PT. DINGIN DAMAI, memberi gaji pokok kepada karyawan kontraknya sebesar Rp. 300,000 perbulan, dengan memperoleh tunjangan-tunjangan sebagai berikut :

- Tunjangan Jabatan

Golongan	Persentase
1	5%
2	10%
3	15%

Logikanya : Jika seorang karyawan tersebut dengan golongan 3, maka mendapatkan tunjangan sebesar 15% * Rp. 300,000

Perulangan

Operasi perulangan selalu dijumpai didalam bahasa pemrograman, disini akan dibahas beberapa perintah perulangan diantaranya.

5.1. Pernyataan for

Perulangan yang pertama adalah *for*. Bentuk umum pernyataan *for* sebagai berikut :

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )
```

Bila pernyataan didalam *for* lebih dari satu maka pernyataan-pernyataan tersebut harus diletakan didalam tanda kurung.

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )  
{  
    pernyataan / perintah;  
    pernyataan / perintah;  
    pernyataan / perintah;  
}
```

Kegunaan dari masing-masing argumen *for* diatas adalah :

- **Inisialisasi:** merupakan bagian untuk memberikan nilai awal untuk variabel-variabel tertentu.
- **Syarat Pengulangan:** memegang kontrol terhadap pengulangan, karena bagian ini yang akan menentukan suatu perulangan diteruskan atau dihentikan.
- **Pengubah Nilai Pencacah:** mengatur kenaikan atau penurunan nilai pencacah.

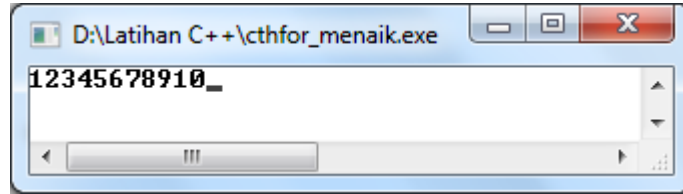
Contoh :

Sebagai contoh program untuk mencetak bilangan dari 1 hingga 10 secara menaik, secara menurun dan menampilkan bilangan ganjil, sebagai berikut:

```
Contoh-1      /* ----- */  
                /* Program for - bilangan naik */  
                /* ----- */  
                #include<stdio.h>  
                #include<conio.h>  
                #include<iostream.h>  
                main( )  
                {  
                    int a;  
                    clrscr( );  
                    for(a = 1; a <= 10; ++a)
```

```
cout<<a;
getch( );
}
```

Output yang akan dihasilkan, dari program contoh-1 diatas adalah :

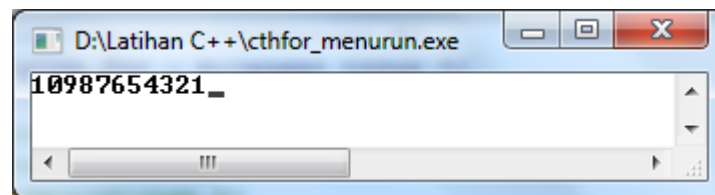


Gambar 5. 1 Hasil Contoh 1

Contoh-2

```
/* ----- */
/* Program for - bilangan turun */
/* ----- */
#include <stdio.h>
#include <conio.h>
#include<iostream.h>
main( )
{
    int a;
    clrscr( );
    for(a = 10; a >= 1; --a)
        cout<<a;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah :



Gambar 5. 2 Hasil Contoh 2

Contoh-3

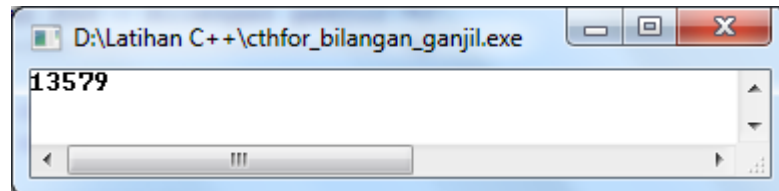
```
/* ----- */
/* Program for - bilangan ganjil */
/* ----- */

#include <stdio.h>
#include <conio.h>
#include<iostream.h>

main( )
{
    int a;
```

```
clrscr();  
for(a = 1; a <= 10; a+=2)  
    cout<<a;  
  
    getch();  
}
```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah :



Gambar 5. 3 Hasil Contoh 3

5.1.1. Pernyataan nested - for

Pernyataan Nested *for* adalah suatu perulangan *for* didalam perulangan *for* yang lainnya. Bentuk umum pernyataan *Nested for* sebagai berikut :

```
for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )  
{  
    for ( inisialisasi; syarat pengulangan; pengubah nilai pencacah )  
    {  
        pernyataan / perintah;  
    }  
}
```



Didalam penggunaan nested-for, perulangan yang di dalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang diluar diselesaikan.

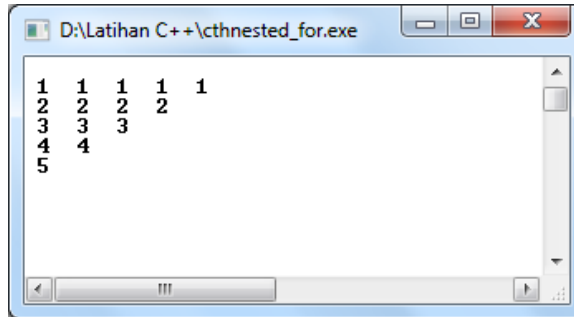
Contoh-4

```
/* ----- */  
/* Program for - Nested for */  
/* ----- */  
#include<stdio.h>  
#include<conio.h>  
  
main()  
  
{  
    int a, b;  
    clrscr();  
  
    for(a = 1; a <= 5; a++)  
    {
```

```

printf("\n");
    for(b = a; b <= 5; b++)
        printf(" %d ",a);
    }
getch();
}
    
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah :



Gambar 5. 4 Hasil Contoh 5

5.1.2. Perulangan Tidak Berhingga

Perulangan tak berhingga merupakan perulangan (loop) yang tak pernah berhenti atau mengulang terus, hal ini sering terjadi disebabkan adanya kesalahan penanganan kondisi yang dipakai untuk keluar dari loop.

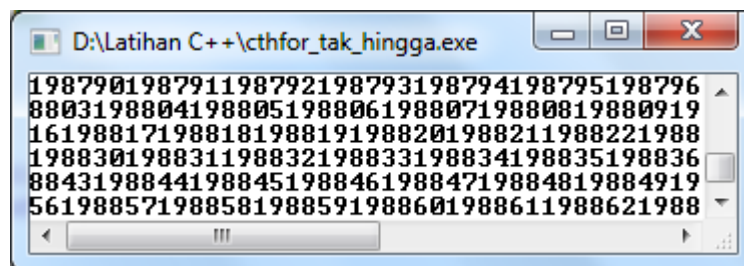
Sebagai contoh, jika penulisan perintah sebagai berikut:

Contoh-5

```

/* ----- */
/* Program for Tdk Berhingga */
/* ----- */
#include<stdio.h>
#include<conio.h>
main( )
{
    int bil;
    clrscr( );
    for (bil = 60; bil >=10; bil++)
        printf("%d", bil);
    getch( );
}
    
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah :



Gambar 5. 4 Hasil Contoh 4

Pada pernyataan ini tidak akan berhenti untuk menampilkan bilangan menurun, kesalahan terjadi pada pengubah nilai pencacah, seharusnya penulisan yang benar berupa

bil - -

Akan tetapi yang ditulis adalah :

bil ++

Oleh karena kondisi $bil \geq 1$ selalu bernilai benar (karena bil bernilai 6), maka pernyataan

printf("%d", bil);

akan terus dijalankan.

Jika terjadi hal semacam ini, untuk menghentikan proses yang terus menerus semacam ini dengan menekan tombol **CTRL – PAUSE** atau **CTRL – BREAK**.

5.2. Pernyataan goto

Pernyataan *goto* merupakan instruksi untuk mengarahkan eksekusi program ke-pernyataan yang diawali dengan suatu label. Label merupakan suatu pengenal (*identifier*) yang diikuti dengan tanda titik dua (:). Bentuk pemakaian *goto* sebagai berikut:

goto label;

Contoh Penggunaan *goto*, dapat dilihat pada program berikut:

```
Contoh-6      /* ----- */
               /* Program dengan pernyataan goto */
               /* ----- */
               #include<iostream.h>
               #include<stdio.h>
               #include<conio.h>

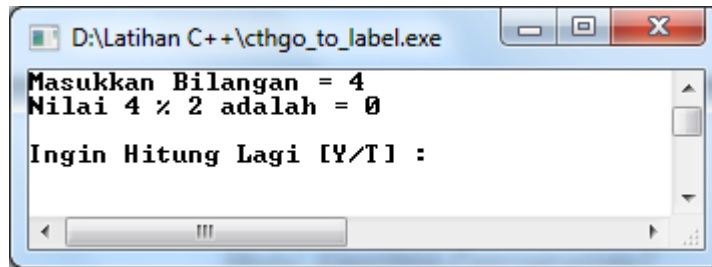
               main( )
               {
                 int a, b;
                 char lagi;
                 atas: // deklarasi label
                 clrscr( );

                 cout<<>"Masukkan Bilangan = "; cin<<a;
                 b = a % 2;
                 printf("Nilai %d %% 2 adalah = %d",a, b);
                 printf("\n\nIngin Hitung Lagi [Y/T] : ");
                 lagi = getche( );

                 if (lagi == 'Y' || lagi == 'y')
                   goto atas; // penggunaan label

                 getch( );
               }
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah :



Gambar 5. 5 Hasil Contoh 6

5.3. Pernyataan *while*

Pernyataan perulangan *while* merupakan instruksi perulangan yang mirip dengan perulangan *for*. Bentuk perulangan *while* dikendalikan oleh syarat tertentu, yaitu perulangan akan terus dilaksanakan selama syarat tersebut terpenuhi. Bentuk umum perulangan *while*, sebagai berikut:

```
while ( syarat )  
    Pernyataan / perintah ;
```

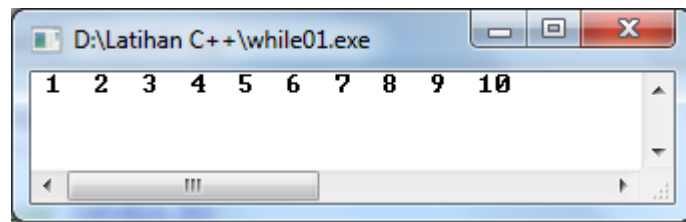
Bentuk umum perulangan *while*, dengan lebih dari perintah / pernyataan, sebagai berikut:

```
while ( syarat )  
{  
    Pernyataan / perintah ;  
    Pernyataan / perintah ;  
}
```

Contoh-7

```
/* ----- */  
/* Program while01.cpp */  
/* ----- */  
  
#include <stdio.h>  
#include <conio.h>  
main()  
{  
    int bil=1;  
    clrscr();  
    while(bil<=10)  
    {  
        printf(" %d ",bil);  
        ++bil;  
    }  
    getch();  
}
```

Output yang akan dihasilkan, dari program contoh-9 diatas adalah:

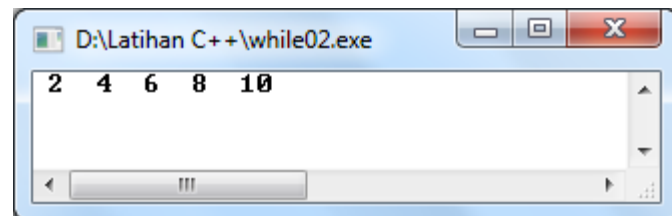


Gambar 5. 6 Hasil Contoh 7

Contoh-8

```
/* ----- */
/* Program while02.cpp */
/* ----- */
#include <stdio.h>
#include <conio.h>
main( )
{
    int bil=2;
    clrscr( );
    while(bil<=10)
    {
        printf(" %d ",bil);
        bil+=2;
    }
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-10 diatas adalah :



Gambar 5. 7 Hasil Contoh 8

5.4. Pernyataan do - while

Pernyataan perulangan *do - while* merupakan bentuk perulangan yang melaksanakan perulangan terlebih dahulu dan pengujian perulangan dilakukan dibelakang.

Bentuk umum perulangan *do - while*, sebagai berikut :

```
do
    pernyataan / perintah ;
while ( syarat );
```

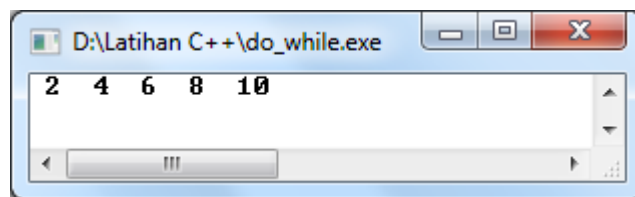
Bentuk umum perulangan *do - while*, dengan lebih dari perintah / pernyataan, sebagai berikut:

```
do
{
    Pernyataan / perintah ;
    Pernyataan / perintah ;
}
while ( syarat );
```

Contoh-9

```
/* ----- */
/* Program do - while */
/* ----- */
#include <stdio.h>
#include <conio.h>
main( )
{
    int bil=2;
    clrscr( );
do
    {
        printf(" %d ",bil);
        bil+=2;
    }
    while(bil<=10);
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-9 diatas adalah :



Gambar 5. 8 Hasil Contoh 9

5.5. Pernyataan break

Pernyataan *break* telah dibahas pada pernyataan pengambilan keputusan *switch*. Pernyataan *break* ini berfungsi untuk keluar dari struktur *switch*. Selain itu pernyataan *break* berfungsi keluar dari perulangan (*for*, *while* dan *do-while*). Jika pernyataan *break* dikerjakan, maka eksekusi akan dilanjutkan ke pernyataan yang terletak sesudah akhir dari badan perulangan (*loop*).

Contoh-10

```
/* ----- */
/* Program do - while dengan break */
/* ----- */
```

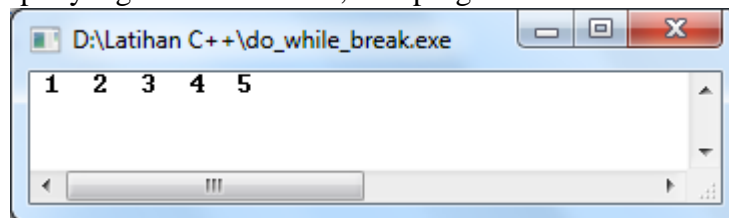
```

#include <stdio.h>
#include <conio.h>
main()
{
    int bil = 1;
    clrscr( );

    do
    {
        if (bil >= 6)
            break;
        printf(" %d ",bil);
    }
    while(bil++);
    getch( );
}

```

Output yang akan dihasilkan, dari program contoh-10 diatas adalah :



Gambar 5. 9 Hasil Contoh 10

Contoh-11

```

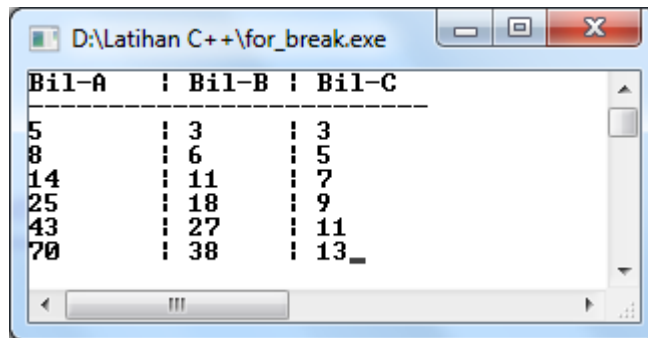
/* ----- */
/* Perulangan FOR dengan break; */
/* ----- */

#include <stdio.h>
#include <conio.h>

main( )
{
    int a=3, b=2, c=1, bil;
    clrscr( );
    printf("Bil-A | Bil-B | Bil-C\n");
    printf("-----");
    for(bil=1; bil<=10; ++bil)
    {
        a+=b; b+=c; c+=2;
        printf("\n%d \t| %d \t| %d",a, b, c);
        if(c==13)
            break;
    }
    getche( );
}

```

Output yang akan dihasilkan, dari program contoh-11 diatas adalah :



Gambar 5. 10 Hasil Contoh 11

5.6. Pernyataan continue

Pernyataan *continue* digunakan untuk mengarahkan eksekusi ke iterasi (proses) berikutnya pada loop yang sama, dengan kata lain mengembalikan proses yang sedang dilaksanakan ke-awal loop lagi, tanpa menjalankan sisa perintah dalam loop tersebut.

Contoh-12

```

/* ----- */
/* Perulangan FOR dengan coninue */
/* ----- */

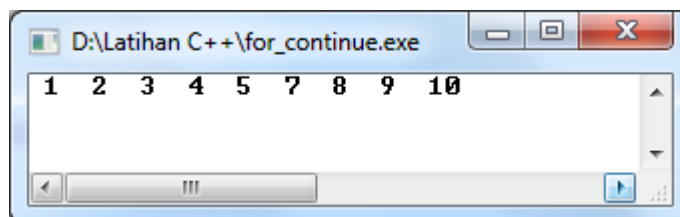
#include <stdio.h>
#include <conio.h>

main()
{
    int bil;
    clrscr();

    for(bil=1; bil<=10; ++bil)
    {
        if(bil==6)
            continue;
        printf(" %d ",bil);
    }
    getch();
}

```

Output yang akan dihasilkan, dari program contoh-12 diatas adalah :



Gambar 5. 11 Hasil Contoh 12

5.7. Tugas 4

Buatlah beberapa program seperti petunjuk berikut :

1. Buatlah program untuk menghitung 10 deret bilangan genap dengan hasilnya :
 $2 + 4 + 6 + 8 + 10 + 12 + 14 + 16 + 18 + 20 = 110$
2. Buatlah program untuk menghitung 10 deret bilangan ganjil dengan hasilnya :
 $1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 + 19 = 100$
3. Buatlah program untuk menghitung penjumlahan deret bilangan genap membentuk segitiga siku dengan hasilnya :
$$\begin{array}{r} 2 \\ 2 + 4 \\ 2 + 4 + 6 \\ 2 + 4 + 6 + 8 \\ 2 + 4 + 6 + 8 + 10 \end{array} \quad \begin{array}{r} = 2 \\ = 6 \\ = 12 \\ = 20 \\ = 30 \end{array}$$
4. Buatlah program untuk menghitung perkalian deret bilangan ganjil membentuk segitiga siku dengan hasilnya :
$$\begin{array}{r} 1 \\ 1 * 3 \\ 1 * 3 * 5 \\ 1 * 3 * 5 * 7 \\ 1 * 3 * 5 * 7 * 9 \end{array} \quad \begin{array}{r} = 1 \\ = 3 \\ = 15 \\ = 105 \\ = 945 \end{array}$$



Array

Variabel Larik atau lebih dikenal dengan ARRAY adalah Tipe terstruktur yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe sama. Suatu Array mempunyai jumlah komponen yang banyaknya tetap. Banyaknya komponen dalam suatu larik ditunjukkan oleh suatu indek untuk membedakan variabel yang satu dengan variabel yang lainnya.

Variabel array dalam Borland C++, dapat digolongkan menjadi dua buah dimensi:

- Array Berdimensi Satu.
- Array Berdimensi Dua

6.1. Array Berdimensi Satu

Sebelum digunakan, variabel array perlu dideklarasikan terlebih dahulu. Cara mendeklarasikan variabel array sama seperti deklarasi variabel yang lainnya, hanya saja diikuti oleh suatu indek yang menunjukkan jumlah maksimum data yang disediakan.

Bentuk Umum pendeklarasian array:

Tipe_Data Nama_Variabel[Ukuran]

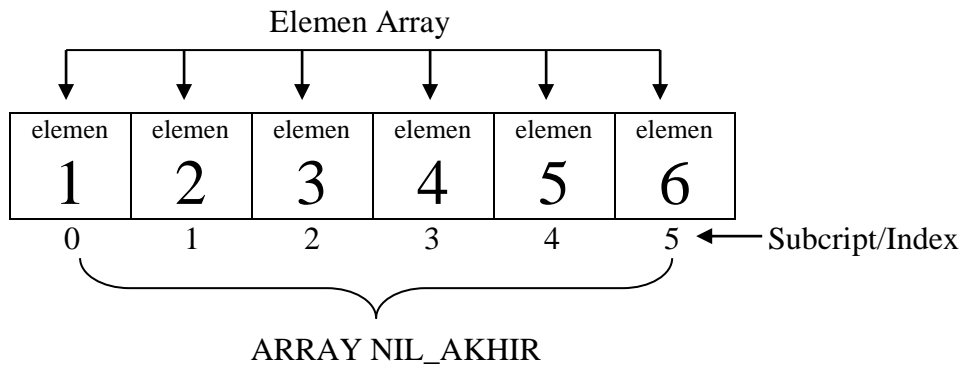
Keterangan :

- Type Data : Untuk menyatakan type data yang digunakan.
- Ukuran : Untuk menyatakan jumlah maksimum elemen array.

Contoh Pendeklarasian Array

`float Nil_Akhir[6];`

Suatu array dapat digambarkan sebagai kotak panjang yang berisi kotak-kotak kecil didalam kotak panjang tersebut.



Subscript atau Index array pada C++, selalu dimulai dari Nol (0)

6.1.1. Inisialisasi Array Berdimensi Satu

Inisialisasi adalah memberikan nilai awal terhadap suatu variabel. Bentuk pendefinisian suatu array dapat dilihat dari contoh berikut :

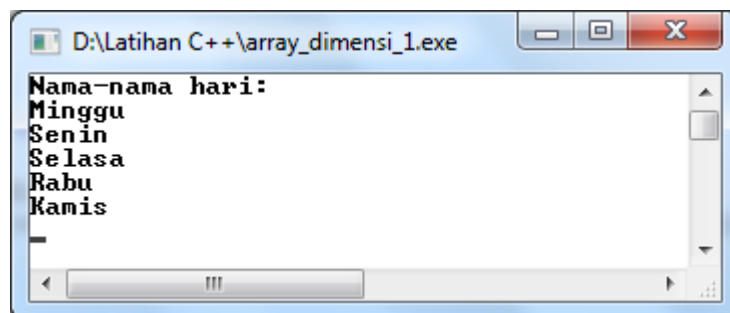
Tipe_data nama_array[jml_elemen] = { nilai array };

```

Contoh-1      /* ----- */
                /* Inisialisasi Array Dimensi 1 */
                /* ----- */
                #include <conio.h>
                #include <iostream.h>

                main()
                {
                    char hari[7][10] =
                    { "Minggu", "Senin", "Selasa", "Rabu", "Kamis", "jum'at", "Sabtu" };
                    clrscr();
                    cout<<"Nama-nama hari:"<<endl;
                    cout<<hari[0]<<endl<<hari[1]<<endl<<hari[2]<<endl<<hari[3]
                    <<endl<<hari[4]<<endl;
                    getch();
                }
    
```

Output yang akan dihasilkan, dari program contoh-1 diatas adalah:



Gambar 8.1. Hasil Contoh-1

6.1.2. Mengakses Array Berdimensi Satu

Suatu array, dapat diakses dengan menggunakan subscript atau indexnya. Bentuk umum pengaksesan dengan bentuk :

Nama_Array[Subscript/Index]

Contoh Nil_Akhir[3];
 Nil_Akhir[1];
 Nil_Akhir[0];

Contoh-2

```

/* ----- */
/* Program Array Satu Dimensi */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip.h>

main()
{
    int i;
    char nama[5][20];
    float nilai1[5];
    float nilai2[5];
    float hasil[5];
    clrscr();

    for(i=1;i<=2;i++)
    {
        cout<<"Data Ke - "<<i<<endl;
        cout<<"Nama Siswa : "; gets(nama[i]);
        cout<<"Nilai MidTest : "; cin>>nilai1[i];
        cout<<"Nilai Final : "; cin>>nilai2[i];
        hasil[i] = (nilai1[i] * 0.40)+ (nilai2[i] * 0.60);
        cout<<endl;
    }

    cout<<"-----" <<endl;
    cout<<"No. Nama Siswa Nilai Nilai ";
    cout<<"Hasil" <<endl;
    cout<<" MidTest Final ";
    cout<<"Ujian" <<endl;
    cout<<"-----" <<endl;
    for(i=1;i<=2;i++)
    {
        cout<<setiosflags(ios::left)<<setw(4)<<i;
        cout<<setiosflags(ios::left)<<setw(10)<<nama[i];
        cout<<setprecision(2)<<" " <<nilai1[i];
        cout<<setprecision(2)<<" " <<nilai2[i];
    }

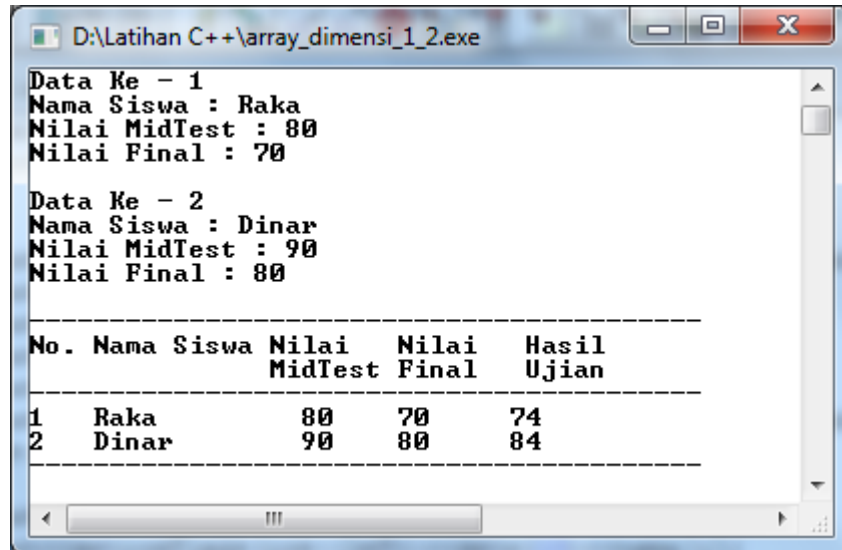
```

```

        cout<<setprecision(2)<<" " <<hasil[i]<<endl;
    }
    cout<<"-----" <<endl;
    getch();
}

```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah:



Gambar 8.2. Hasil Contoh-2

6.2. Array Berdimensi Dua

Array dimensi dua tersusun dalam bentuk baris dan kolom, dimana indeks pertama menunjukkan baris dan indeks kedua menunjukkan kolom. Array dimensi dua dapat digunakan seperti pendataan penjualan, pendataan nilai dan lain sebagainya. Bentuk Umum pendeklarasian array :

Tipe_Data Nama_Variabel[index-1][index-2]

Keterangan:

- Type Data : Untuk menyatakan type data yang digunakan.
- Index-1 : Untuk menyatakan jumlah baris
- Index-2 : Untuk menyatakan jumlah kolom

Contoh Pendeklarasian Array

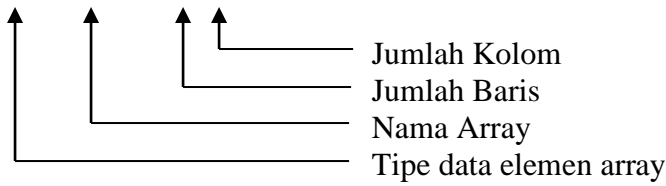
Sebagai contoh pendeklarasian yang akan kita gunakan adalah pengolahan data penjualan, berikut dapat anda lihat pada tabel berikut :

Data Penjualan Pertahun

No	Tahun Penjualan		
	2001	2002	2003
1	150	159	230
2	100	125	150
3	210	125	156

Jika anda lihat dari tabel 7.1 diatas maka dapat dituliskan kedalam array dimensi dua berikut:

int data_jual[3][3];



6.2.1. Inisialisasi Array Berdimensi Dua

Inisialisasi adalah memberikan nilai awal terhadap suatu variabel. Bentuk pendefinisian suatu array dapat dilihat dari contoh berikut:

```

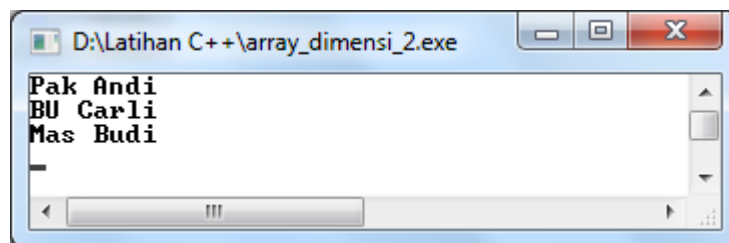
Contoh-3      #include <conio.h>
                  #include <iostream.h>

                  main()
                  {
                    char nama[2][3][10]={{ "Pak","BU","Mas"},
                                          {"Andi","Budi","Carli"} };

                    clrscr();

                    cout<<nama[0][0]<<ends<<nama[1][0]<<endl;
                    cout<<nama[0][1]<<ends<<nama[1][2]<<endl;
                    cout<<nama[0][2]<<ends<<nama[1][1]<<endl;
                    getch();
                  }
    
```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah:



Gambar 8.3. Hasil Contoh-3

6.2.3. Mengakses Array Berdimensi Dua

Suatu array, dapat diakses dengan menggunakan subscript atau indexnya. Contoh pengaksesan dengan bentuk sebagai berikut:

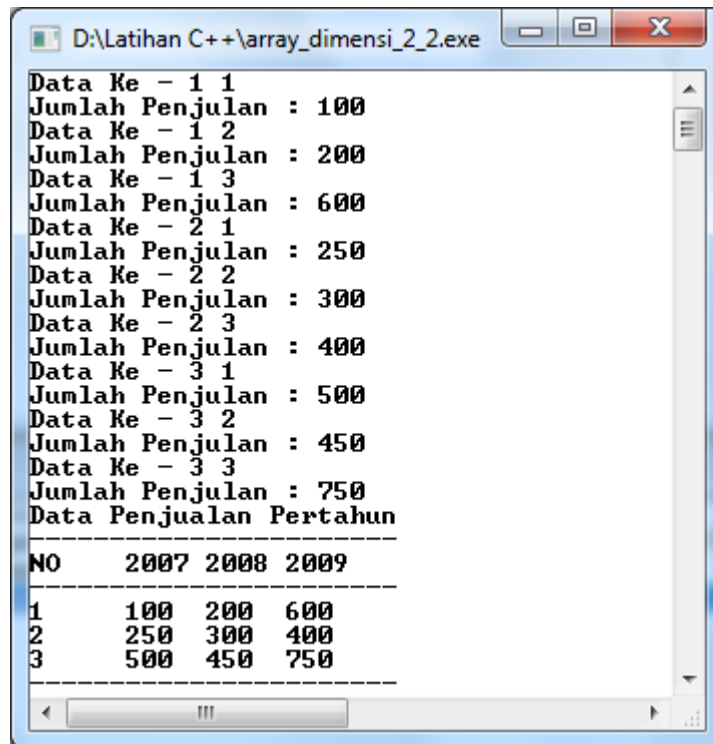
```

Contoh-4      /* ----- */
                  /* Array Dimensi 2 */
                  /* ----- */

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

```
main()
{
    int i, j;
    int data_jual[4][4];
    clrscr();
    for(i=1;i<=3;i++)
    {
        for(j=1;j<=3;j++)
        {
            cout<<"Data Ke - "<<i<<" "<<j<<endl;
            cout<<"Jumlah Penjualan : ";
            cin>>data_jual[i][j];
        }
    }
    cout<<"Data Penjualan Pertahun"<<endl;
    cout<<"-----"<<endl;
    cout<<"NO    2007 2008 2093"<<endl;
    cout<<"-----"<<endl;
    for(i=1;i<=3;i++) {
        cout<<setiosflags(ios::left)<<setw(5)<<i;
        for(j=1;j<=3;j++)
        {
            cout<<setiosflags(ios::right)<<setw(4);
            cout<<data_jual[i][j];
            cout<<" ";
        }
        cout<<endl;
    }
    cout<<"-----"<<endl;
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-4 diatas adalah:



Gambar 8.4. Hasil Contoh-4

6.2.24. Sorting dalam C++

a. Selection Sort

Tehnik pengurutan dengan cara pemilihan elemen atau proses kerja dengan memilih elemen data terkecil untuk kemudian dibandingkan & ditukarkan dengan elemen pada data awal, dst sampai seluruh elemen shg akan menghasilkan pola data yg telah *disort*.

Prinsip kerja dari teknik ini adalah sbb :

1. Pengecekan dimulai data ke-1 sampai dengan data ke-n
2. Tentukan bilangan dengan Index terkecil dari data bilangan tersebut
3. Tukar bilangan dengan Index terkecil tersebut dengan bilangan pertama (I = 1) dari data bilangan tersebut
4. Lakukan langkah 2 dan 3 untuk bilangan berikutnya (I= I+1) sampai didapatkan urutan yg optimal.

Contoh-5 /* ----- */
 /* Selection sort dengan C++ */
 /* ----- */

```

#include<conio.h>
#include<stdio.h>
int main() {
int i, j, iMin;
int n, Urut;
int Tmp, code;
int Arr[100];

printf("\nInputkan banyak data yang akan diurutkan : ");
scanf("%i", &n);
Urut = 1;
for(i = 0; i < n; i++) {
printf("Masukan data ke %i : ", i + 1);
scanf("%i", &Arr[i]);
}
for(i = 0; i < n - 1; i++) {
iMin = i;
for(j = Urut; j < n; j++) {
if(Arr[j] < Arr[iMin]) {
iMin = j;
if(Arr[i] != Arr[iMin]) {
Tmp = Arr[i];
if(Arr[i] > Arr[iMin]) {
Arr[i] = Arr[iMin];
Arr[iMin] = Tmp;
}
}
}
}
Urut = Urut + 1;
}
printf("\nSetelah Pengurutan\n");
for(i = 0; i < n; i++) {
printf("Elemen ke %i : %i\n", i + 1, Arr[i]);
}
getch();
}for(i = 0; i < n - 1; i++) {
iMin = i;
for(j = Urut; j < n; j++) {
if(Arr[j] < Arr[iMin]) {
iMin = j;
if(Arr[i] != Arr[iMin]) {
Tmp = Arr[i];
if(Arr[i] > Arr[iMin]) {
Arr[i] = Arr[iMin];
Arr[iMin] = Tmp;
}
}
}
}
}
}
Urut = Urut + 1;

```

```
}
printf("\nSetelah Pengurutan\n");
for(i = 0; i < n; i++) {
printf("Elemen ke %i : %i\n", i + 1, Arr[i]);
}
getch(); }
```

b. Buble Sort

Prinsip Kerja dari *Bubble Sort* adalah :

1. Pengecekan mulai dari data ke-1 sampai data ke-n
2. Bandingkan data ke-n dengan data sebelumnya (n-1)
3. Jika lebih kecil maka pindahkan bilangan tersebut dengan bilangan yg ada didepannya (sebelumnya) satu persatu (n-1,n-2,n-3,...dst)
4. Jika lebih besar maka tidak terjadi pemindahan
5. Ulangi langkah 2 dan 3 s/d sort optimal.

```
#include<conio.h>
#include<stdio.h>
int main() {
int i, j, iMin;
int n, Urut;
int Tmp, code;
int Arr[100];
printf("\nInputkan banyak data yang akan diurutkan : ");
scanf("%i", &n);
for(i = 0; i < n; i++) {
printf("Masukan data ke %i : ", i + 1);
scanf("%i", &Arr[i]);
}
for(i = 1; i < n; i++) {
for(j = 0; j < n - 1; j++) {
if(Arr[j] > Arr[j + 1]) {
Tmp = Arr[j];
Arr[j] = Arr[j + 1];
Arr[j + 1] = Tmp;
}
}
}
printf("\nSetelah Pengurutan\n");
for(i = 0; i < n; i++) {
printf("Elemen ke %i : %i\n", i + 1, Arr[i]);
}
getch();
}
```

c. Insertion Short

Prinsip dasar Insertion adalah secara berulang-ulang menyisipkan / memasukan setiap elemen. ke dlm posisinya / tempatnya yg benar.

1. Prinsip Kerja *Insertion Sort* adalah
2. Pengecekan mulai dari data ke-1 sampai data ke-n

3. Bandingkan data ke-I (I = data ke-2 s/d data ke-n)
4. Bandingkan data ke-I tersebut dengan data sebelumnya (I-1), Jika lebih kecil maka data tersebut dapat disisipkan ke data awal sesuai dgn posisi yg seharusnya
5. Lakukan langkah 2 dan 3 untuk bilangan berikutnya (I= I+1) sampai didapatkan urutan yg optimal.

```
#include "conio.h"
#include "stdio.h"

int main( ) {

int i, j, iMin;
int n, Urut;
int Tmp, code;
int Arr[100];
printf("\nInputkan banyak data yang akan diurutkan : ");
scanf("%i", &n);
for(i = 0; i < n; i++)
{
printf("Masukan data ke %i : ", i + 1);
scanf("%i", &Arr[i]);
}
for(i = 1; i < n; i++)
{
Tmp = Arr[i];
j = i - 1;
while(Arr[j] >= Tmp && j > 0) {
Arr[j + 1] = Arr[j];
j = j - 1;
}
if(Tmp >= Arr[j]) {
Arr[j + 1] = Tmp;
} else {
Arr[j + 1] = Arr[j];
Arr[j] = Tmp;
}
}
printf("\nSetelah Pengurutan\n");
for(i = 0; i < n; i++) {
printf("Elemen ke %i : %i\n", i + 1, Arr[i]);
}
getch();
}
```

8.4. TUGAS 5

1. Sebuah perusahaan ayam goreng dengan nama “**GEROBAK FRIED CHICKEN**” yang telah lumayan banyak pelanggannya, ingin dibantu dibuatkan program untuk membantu kelancaran usahanya.
 “**GEROBAK FRIED CHICKEN**” mempunyai daftar harga ayam sebagai berikut :

Kode JenisPotong Harga

```
-----
D   Dada Rp. 2500
P   Paha Rp. 2000
S   Sayap Rp. 1500
-----
```

Buatlah programnya dengan ketentuan:

- Setiap pembeli dikenakan pajak sebesar 10% dari pembayaran.
- Banyak Jenis, Jenis Potong dan Banyak Beli diinput.
- Tampilan yang diinginkan sebagai berikut:

Layar Masukkan

GEROBAK FRIED CHICKEN

```
-----
Kode JenisPotong  Harga
-----
```

```
D   Dada      Rp. 2500
P   Paha      Rp. 2000
S   Sayap     Rp. 1500
-----
```

```
Banyak Jenis : ... <diinput>
Jenis Ke - ... <proses counter>
Kode Potong [D/P/S] : ... <diinput>
Banyak Potong : ... <diinput>
<<Terus berulang tergantung Banyak Jenis>>
```

Layar Keluaran

GEROBAK FIRED CHICHEN

```
-----
No. Jenis      Harga  Bayak  Jumlah
   Potong      Satuan Beli   Harga
-----
```

```
... ..      ....   ....   Rp ....
... ..      ....   ....   Rp ....
-----
```

```
Jumlah Bayar Rp ....
Pajak 10%   Rp ....
Total Bayar Rp ....
```

2. PT. STAY COOL, memberikan Honor tetap kepada karyawan kontraknya sebesar Rp. 700,000,- per bulan, dengan memperoleh tunjangan-tunjangan sebagai berikut:

- Tunjangan Jabatan

Golongan	Persentase
1	5%
2	10%
3	15%

- Honor Lembur Jumlah jam kerja normal dalam satu bulan sebanyak 240 Jam Kerja. Honor lembur diberikan jika jumlah jam kerja sebih dari 240 jam, maka kelebihan jam kerja tersebut dikalikan dengan honor lembur perjam sebesar Rp. 2,500 untuk setiap kelebihan jam kerja dalam satu bulannya.
- Pendapatan Bersih = Honor tetap + Tunjangan Jabatan+Honor lembur – Pajak 10%
- Total Gaji yang dikeluarkan = akumulasi dari pendapatan bersih karyawan
- Tampilan yang diinginkan sebagai berikut :

Layar Masukkan dan Keluaran

**Program Hitung Honor Karyawan Kontrak
PT. STAY COOL**

Masukkan Jumlah Karyawan : ... <diinput>
 Karyawan Ke - ... <proses counter>
 Nama Karyawan : ... <di input>
 Golongan (1/2/3) : ... <di input>
 Jumlah Jam Kerja : ... <di input>

<<Terus berulang tergantung Jumlah Karyawan>>

PT. STAY COOL

No.	Nama Karyawan	Tunjangan Jabatan	Honor Lembur	Pajak	Pendapatan Bersih
...
...

Total Gaji yang dikeluarkan Rp.



Pointer Dan File Header

Merupakan sebuah variabel yang berisi alamat dari variabel lain. Suatu pointer dimaksudkan untuk menunjukan ke suatu alamat memori sehingga alamat dari suatu variabel dapat diketahui dengan mudah.

7.1. Operator Pointer

Terdapat dua macam operator pointer yang disediakan oleh Borland C++:

1. Operator dereference (&)
2. Operator reference (*)

7.1.1. Operator Dereference (&)

Didalam mendeklarasikan suatu variabel harus pada lokasi yang pasti didalam penggantian memori. Pada umumnya kita tidak dapat menentukan dimana variabel akan ditempatkan. Terkadang secara otomatis dilakukan oleh kompiler dan sistem operasi yang sedang aktif, tetapi sesekali sistem operasi akan memberikan banyak alamat yang kita tidak mengetahui dimana variabel ditempatkan. Hal ini dapat dilakukan dengan memberikan suatu identifer “&” (*ampersand sign*) didepan nama variabel, operator ini biasa disebut dengan “*address of*” atau operator alamat.

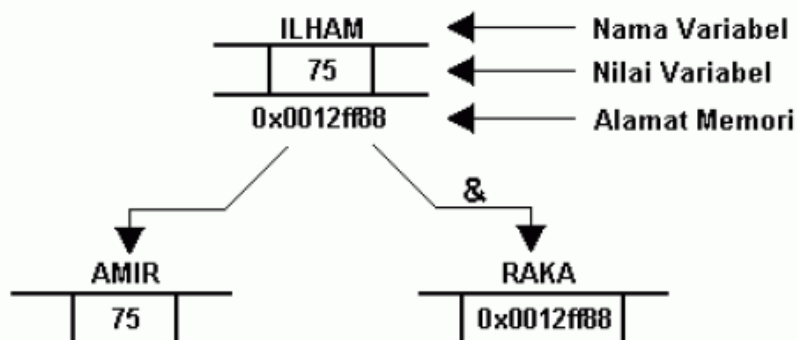
Dengan menggunakan operator dereference (&) ini, suatu variabel akan menghasilkan alamat lokasi memori.

Sebagai contoh ILHAM ditempatkan pada memori dengan alamat 0x0012ff88 dan dideklarasikan sebagai berikut:

ILHAM = 75;

AMIR = ILHAM; // AMIR sama dengan ILHAM (75)

RAKA = &ILHAM; // RAKA sama dengan Address Of ILHAM (0x0012ff88)



Gambar 7.1. Diagram Penggunaan Opeator Dereference

7.1.2. Operator Reference (*)

Dengan menggunakan operator anda dapat mengakses secara langsung nilai yang terdapat didalam variabel yang berpointer, hal ini dapat dilakukan dengan menambahkan identifier asterisk (*), agar dapat menterjemahkan nilai sebenarnya dari suatu variabel. Operator ini biasa disebut dengan *“value pointed by”*.

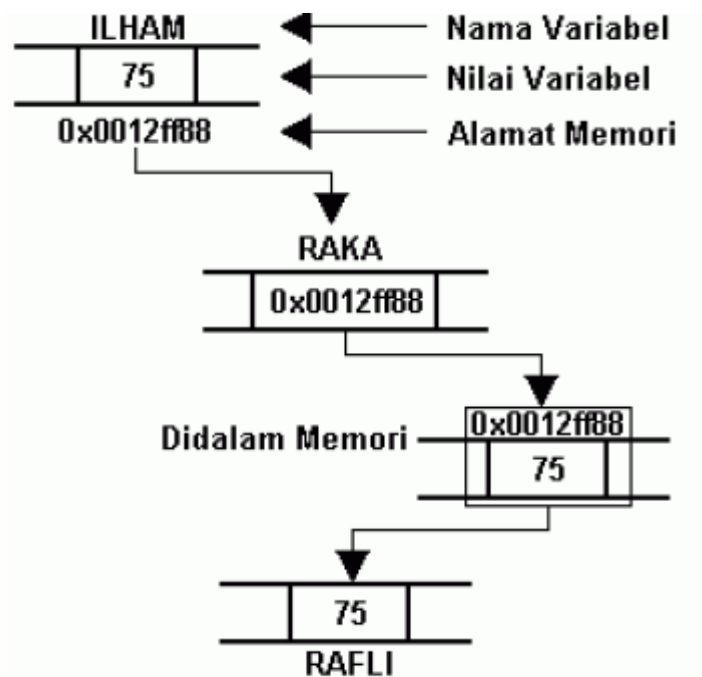
Dengan menggunakan operator reference (*) ini, menghasilkan nilai yang berada pada suatu alamat memori

Sebagai contoh ILHAM ditempatkan pada memori dengan alamat 65524 dan dideklarasikan sebagai berikut:

ILHAM = 75;

RAKA = &ILHAM; // RAKA sama dengan Address Of ILHAM (0x0012ff88)

RAFLI = *RAKA; // RAFLI sama dengan value pointed by RAKA(75)



Gambar 7.2 Diagram Penggunaan Opeator Reference

7.2. Deklarasi Pointer Pada Konstanta

Suatu pointer dapat dideklarasikan secara konstanta atau secara tetap tidak dapat diubah. Untuk mendeklarasikan pointer secara konstanta dengan memberikan kata const didepan nama konstanta.

Bentuk penulisan:

tipe_data * const nama_konstanta;

Contoh-1

```
//-----//
//Pendeklarasian Pointer Konstanta //
//-----//
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
void main()
{
    char *const nama = "Borland C++";
```

```

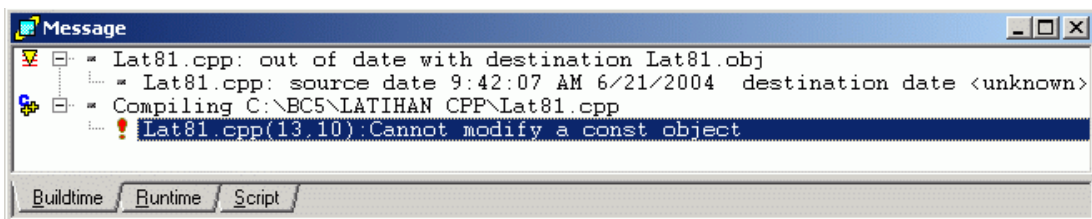
clrscr( );
cout<<"Nama Program = "<<nama<<endl;
nama = "Visual C++";
cout<<"Nama Program = "<<nama<<endl;
getch( );
}

```

Pada program diatas, terdapat kesalahan dan tidak dapat dijalankan, penyebabnya pada pernyataan **nama = "Borland C++"**; Karena variabel **nama**, merupakan pointer konstanta, yaitu tidak dapat diubah-ubah. Pesan Kesalahan Yang Tampil adalah:

Cannot modify a const object

Error Message yang akan dihasilkan, dari program contoh-1 diatas adalah:



Gambar 7.3 Error Message Contoh-1

7.3. Deklarasi Pointer Pada Variabel

Karena keahlian dari pointer untuk menunjuk secara langsung kesuatu nilai, memeriksa satu persatu data yang memiliki pointer pada saat variabel tersebut pertama kali dideklarasikan.

Bentuk penulisan:

```
tipe_data *nama_variabel;
```

Contoh-2

```

//-----//
//Penggunaan Pointer Dereference //
//-----//
#include<stdio.h>
#include<conio.h>
#include<iostream.h>

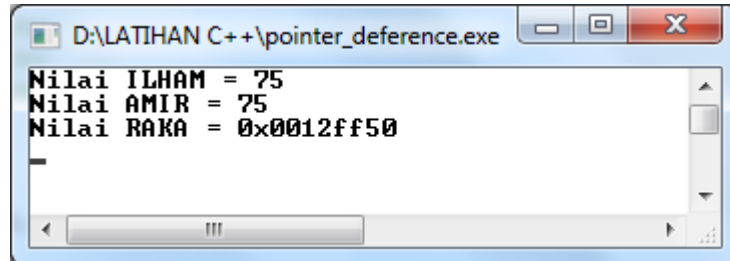
main( )
{
    int ilham, amir, *raka;
    clrscr( );
    ilham = 75;
    amir = ilham;

    raka = &ilham;
    cout<<"Nilai ILHAM = "<<ilham<<endl;
    cout<<"Nilai AMIR = "<<amir<<endl;
}

```

```
        cout<<"Nilai RAKA = "<<raka<<endl;
        getch();
    }
```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah:



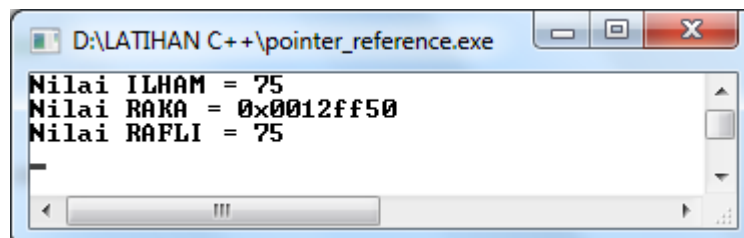
Gambar 7.4 Hasil Contoh-2

Contoh-3

```
//-----//
//Penggunaan Pointer Reference //
//-----//
#include<stdio.h>
#include<conio.h>
#include<iostream.h>

main()
{
    int ilham, *raka, rafli;
    clrscr();
    ilham = 75;
    raka = &ilham;
    rafli = *raka;
    cout<<"Nilai ILHAM = "<<ilham<<endl;
    cout<<"Nilai RAKA = "<<raka<<endl;
    cout<<"Nilai RAFLI = "<<rafli<<endl;
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah:



Gambar 7.5 Hasil Contoh-3

7.4. Pointer Pada Pointer

Tidak terbatas menunjuk alamat dari suatu variabel, pointer dapat pula menunjuk ke pointer lainnya. Didalam pendeklarasiannya, hanya menambahkan pointer reference (*) pada variabel yang akan ditunjuk.

Sebagai contoh :

```
char ilham;
char *raka;      //pointer ke variabel
char **amir;    //pointer pada pointer
ilham = '75';
raka = &ilham;
amir = &raka;
```



Gambar 7.6 Diagram Penggunaan Pointer Pada Pointer

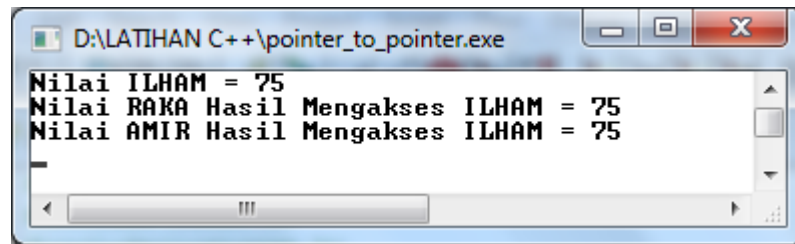
Contoh-5

```
// -----//
//Penggunaan Pointer to Pointer //
//-----//
#include<stdio.h>
#include<conio.h>
#include<iostream.h>

main()
{
    int ilham;
    int *raka; //pointer ke variabel
    int **amir; //pointer pada pointer
    clrscr( );

    ilham = 75;
    cout<<"Nilai ILHAM = "<<ilham<<endl;
    //-> Penugasan Ke Alamat Memori
    raka = &ilham;
    amir = &raka;
    cout<<"Nilai RAKA Hasil Mengakses ILHAM = ";
    cout<<*raka<<endl;
    cout<<"Nilai AMIR Hasil Mengakses ILHAM = ";
    cout<<**amir<<endl;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-5 diatas adalah:



Gambar 7.7 Hasil Contoh-5

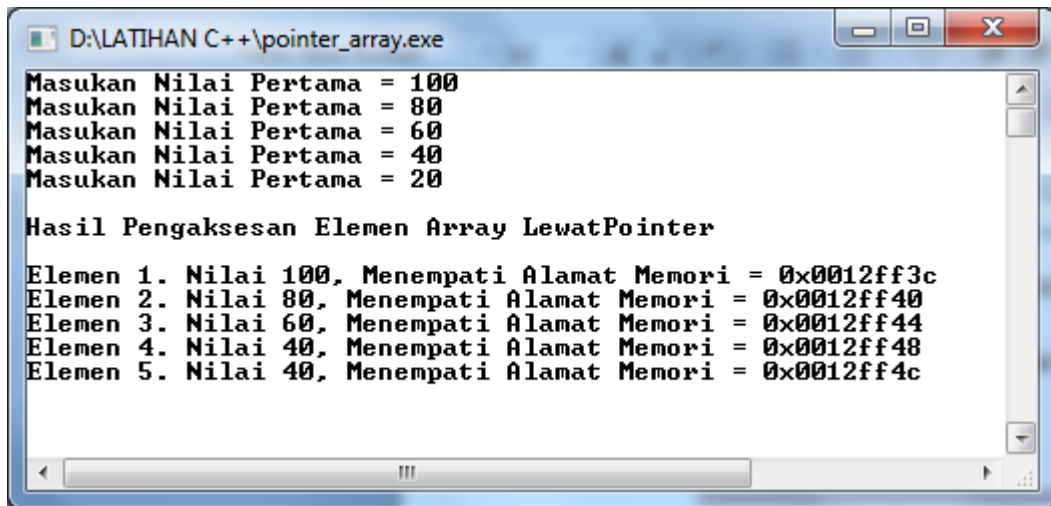
7.5. Pointer pada Array

Konsep Array diantaranya adalah banyak loncatan dari pointer satu ke pointer yang lain. karena secara internal array juga menyatakan alamat, yaitu pengenalan array sama dengan alamat pada elemen pertama, pada array. Sebagai contoh sederhana dapat anda lihat pada contoh program berikut:

Contoh-6

```
//-----//
//Penggunaan Pointer to Array //
//-----//
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main( )
{
    int i;
    int nilai[5];
    int *ptrnilai;
    ptrnilai = nilai;
    for(i=1;i<=5;i++)
    {
        cout<<"Masukan Nilai Pertama = ";
        cin>>nilai[i];
    }
    cout<<endl;
    cout<<"Hasil Pengaksesan Elemen Array Lewat";
    cout<<"Pointer";
    cout<<endl<<endl;
    for(i=1;i<=5;i++)
    {
        cout<<"Elemen "<<i<<". Nilai "<<nilai[i];
        cout<<", Menempati Alamat Memori = ";
        cout<<&ptrnilai[i];
        cout<<endl;
    }
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah:



Gambar 7.8 Hasil Contoh-6

7.6. Pointer pada String

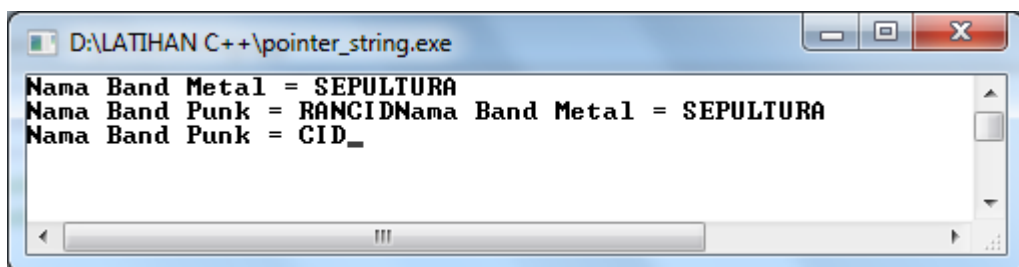
Pointer pada string dapat anda lihat pada contoh program berikut:

```

Contoh-7      /* ----- */
                 /* Pointer pada String */
                 /* ----- */
                 #include <iostream.h>
                 #include <conio.h>

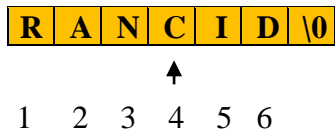
                 main()
                 {
                   char band_metal[ ] = "SEPULTURA";
                   char *band_punk = "RANCID";
                   cout<<"Nama Band Metal = "<<band_metal<<endl;
                   cout<<"Nama Band Punk = "<<band_punk;
                   band_punk+=3; //menambah nilai penunjuk / pointer
                   cout<<"Nama Band Metal = "<<band_metal<<endl;
                   cout<<"Nama Band Punk = "<<band_punk;
                   getch();
                 }
    
```

Output yang akan dihasilkan, dari program contoh-7 diatas adalah:



Gambar 7.9 Hasil Contoh-7

Pada program diatas, terdapat perubahan nilai pointer `band_punk` , yang di tunjukkan oleh penambahan nilai pointer pada `band_punk+=3`, secara default, pembacaan dilakukan mulai dari pointer pertama, karena sudah terjadi penambahan dengan 3, maka pembacaan berpindah ke alamat ke.4, sehingga tercetak kata CID.



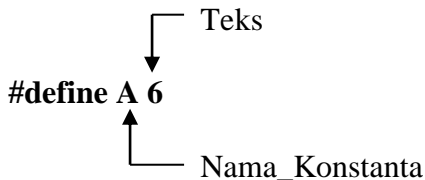
7.7. Preprocessor Directives

Preprocessor directive merupakan suatu perintah yang termasuk kedalam program, tetapi bukanlah instruksi dari program itu sendiri, tetapi untuk preprocessor. Preprocessor ini dijalankan secara otomatis oleh kompiler, ketika didalam proses penterjemahan (*Compile*) program berlangsung, didalamnya membuat nilai pembuktian pertama dan menterjemahkan code program didalam kode objek. Didalam penggunaan preprocessor directive selalu dimulai dengan tanda : # Ada beberapa preprocessor directive, diantaranya adalah:

7.7.1. # define

Digunakan untuk mendefinisikan suatu nilai tertentu kepada suatu nama konstanta. Bentuk umum dari preprocessor directive **#define** ini adalah:
#define nama_konstanta teks

Contoh :



Dalam pendeklarasian preprocessor directive **#define**, **Nama_Konstanta** sebaiknya ditulis dengan menggunakan huruf besar, guna untuk membedakannya dengan nama_variabel. Sedangkan **Teks** merupakan suatu nilai yang diberikan pada nama_konstanta. Teks dapat berupa:

- Numerik contoh: #define PI 3.14
- Karakter contoh: #define HURUF 'B'
- String contoh: #define JABATAN "INSTRUCTOR"
- Pernyataan contoh: #define CETAK ("Borland C++")
- Fungsi Sederhana contoh: #define LUAS_KUBUS (n*n)

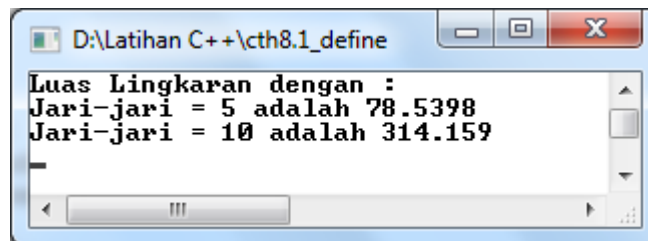
Setelah `#define` ditentukan didalam program cukup dituliskan nama_konstantanya saja. `#define` akan mengganti semua nama konstanta tadi dengan teksnya sebelum proses kompilasi dimulai.

Contoh-1

```
/* ----- */
/* Program Penggunaan #define */
/* ----- */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#define PI 3.141592
#define L(n) PI*n*n

main( )
{
    clrscr();
    cout<<"Luas Lingkaran dengan : "<<endl;
    cout<<"Jari-jari = 5 adalah "<<L(5)<<endl;
    cout<<"Jari-jari = 10 adalah "<<L(10)<<endl;
    getch();
}
```

Output yang akan dihasilkan, dari program contoh-1 diatas adalah:



```
D:\Latihan C++\cth8.1_define
Luas Lingkaran dengan :
Jari-jari = 5 adalah 78.5398
Jari-jari = 10 adalah 314.159
```

Gambar 8.1 Hasil Contoh-1

Contoh-2

```
/* ----- */
/* Program Penggunaan #define */
/* ----- */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#define awal {
#define akhir }
#define mulai() main()
#define cetak cout
#define masuk cin
#define hapus() clrscr()
#define tahan() getch()
#define LS_KUBUS (sisi*sisi)

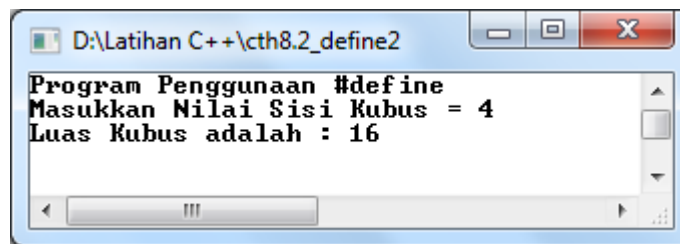
mulai()
awal
```

```

int sisi, ls_kubus;
hapus( );
cetak<<"Program Penggunaan #define"<<endl;
cetak<<"Masukkan Nilai Sisi Kubus = ";
masuk>>sisi;
ls_kubus = LS_KUBUS;
cetak<<"Luas Kubus adalah : "<<ls_kubus;
tahan( );
akhir

```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah:



Gambar 8.2. Hasil Contoh-2

7.7.2. # include

Preprocessor #include telah dibahas pada bab sebelumnya, yaitu berfungsi untuk memasukkan atau menyertakan file-file header kedalam program yang akan dibuat. Dalam penulisan #include ada dua bentuk penulisan :

#include "nama_file_header"

atau

#include <nama_file_header>

Pada bentuk penulisan #include mempunyai arti yang berbeda, yaitu:

- **#include "nama_file_header"**

“Pertama kali compiler akan mencari file header yang disebutkan pada directori yang sedang aktif dan apa bila tidak ditemukan akan mencari pada directori dimana file header tersebut berada “.

- **#include <nama_file_header>**

“Pertama kali compiler akan mencari file header yang disebutkan pada directori yang ada file headernya, kecuali pada directori yang sedang aktif.

7.7.3. # if - #endif

Preprocessor #if - #endif digunakan untuk mengkompilasi jika pernyataan kondisi pada #if bernilai benar, jika tidak maka, diabaikan. Pernyataan kondisi berupa ekspresi konstanta yang dideklarasikan dengan #define.

Bentuk Penulisan #if ekspresi-konstanta
 pernyataan;
 #endif

Contoh-3

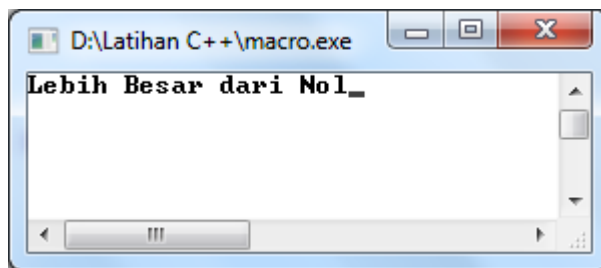
```

/*****
Penggunaan #if - #endif

```

```
*****/  
#include <conio.h>  
#include <stdio.h>  
#define N 4  
main( )  
{  
#if N > 0  
    printf("Lebih Besar dari Nol");  
#endif  
getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah:



Gambar 8.3. Hasil Contoh-3

7.7.4. # if - #else - #endif

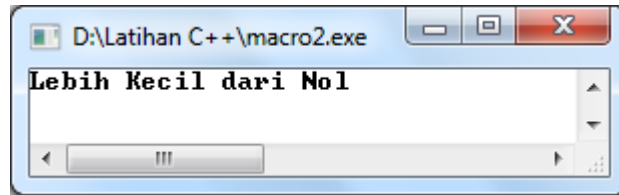
Preprocessor #if - #else -#endif digunakan untuk mengkompilasi jika pernyataan kondisi pada #if bernilai benar, jika #if bernilai salah maka, pernyataan #else dikompilasi. Pernyataan kondisi berupa ekspresi konstanta yang dideklarasikan dengan #define.

Bentuk Penulisan #if ekspresi-konstanta
 Pernyataan-1;
#else
 Pernyataan-2;
#endif

Contoh-4

```
/******  
Penggunaan #if - #else - #endif  
*****/  
#define N -4  
main( )  
{  
#if N > 0  
    printf("Lebih Besar dari Nol");  
#else  
    printf("Lebih Kecil dari Nol");  
#endif  
}
```

Hasil dari program contoh-4 diatas adalah:



Gambar 8.4. Hasil Contoh-4

7.7.5. # elif

Preprocessor **#elif** digunakan untuk mengkompilasi dari pernyataan bertingkat. Dalam hal ini **#elif** sama halnya seperti **#elseif**, merupakan kombinasi dari **#if** dan **#else**. Perintah dijalankan sesuai dengan kondisi yang telah ditentukan, Hasil hanya dapat dijalankan sesuai dengan ketentuan yang benar. Bentuk **#elif** diikuti oleh ekspresi-konstanta.

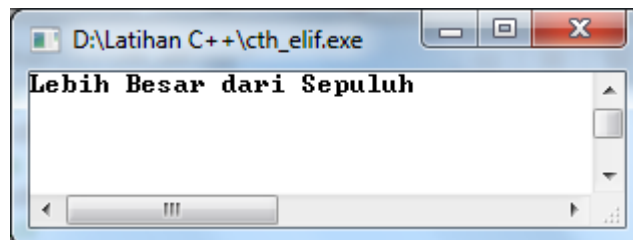
Bentuk Penulisan

```
#if ekspresi-konstanta-1
    Pernyataan-1;
#elif ekspresi-konstanta-2
    Pernyataan-2;
# elif ekspresi-konstanta-n
    Pernyataan-n;
#endif
```

Contoh-5

```
/******
Penggunaan #elif
*****/
#define N 12
main( )
{
    #if N > 10
        printf("Lebih Besar dari Sepuluh");
    #elif N == 10
        printf("Sama Dengan Sepuluh ");
    #else N < 10
        printf("Lebih Kecil dari Sepuluh");
    #endif
}
```

Hasil dari program contoh-5 diatas adalah:



Gambar 8.5. Hasil Contoh-5

7.7.6. #undef

Preprocessor **#undef** digunakan untuk menghilangkan nilai yang telah didefinisikan dari daftar definisi.

Contoh-6

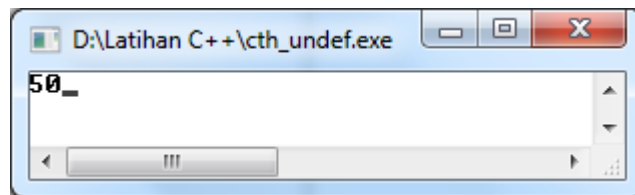
```

/*****
Penggunaan #undef
*****/
#include<iostream.h>
#define LEBAR_MAKS 100
#if LEBAR_MAKS>200
#undef LEBAR_MAKS //--> menghilangkan LEBAR_MAKS
#define LEBAR_MAKS 200
#elif LEBAR_MAKS <50
#undef LEBAR_MAKS //--> menghilangkan LEBAR_MAKS
#define LEBAR_MAKS 50
#else
#undef LEBAR_MAKS //--> menghilangkan LEBAR_MAKS
#define LEBAR_MAKS 50
#endif

main( )
{
char str[LEBAR_MAKS];
cout<<LEBAR_MAKS;
}

```

Hasil dari program contoh-6 diatas adalah:



Gambar 8.6. Hasil Contoh-6

7.7.7. # ifdef - # ifndef

Preprocessor #ifdef dan #ifndef memberikan bagian dari program yang akan dikompilasi, hal ini dapat dilakukan jika sudah konstanta didefinisikan pada bagian #define, hal ini merupakan parameter yang khusus yang harus terdefinisi.

Bentuk umum penulisan sebagai berikut:

#ifdef

nama-konstanta pernyataan;

#endif

Penjelasan: Jika nama-konstanta terdefinisi maka, pernyataan akan dijalankan, jika nama-konstanta tidak terdefinisi maka, pernyataan akan diabaikan.

Contoh-7

```

/--> Penggunaan #ifdef dan #ifndef
#include<iostream.h>
#define ANAK1 "ILHAM"
#define ANAK2 "HADIANSYAH"

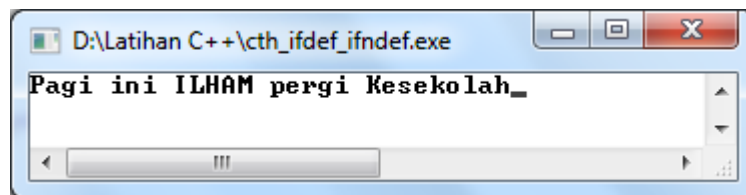
```

```

main( )
{
    #ifdef ANAK1
        cout<<"Pagi ini "<<ANAK1<<" pergi Kesekolah";
    #else
        #ifndef ANAK2
            cout<<"Hari ini "<<ANAK1<<" "<<ANAK2;
            cout<<" pergi Kesekolah";
        #else
            cout<<"Pagi ini "<<ANAK2<<"pergi Kesekolah";
        #endif
    #endif
    getch();
}

```

Hasil dari program contoh-7 diatas adalah:



Gambar 8.7. Hasil Contoh-7

7.8. Pembuatan File Header

File Header adalah suatu file dengan akhiran **.h** . File ini sebenarnya berisikan deklarasi fungsi dan definisi konstanta. Selain file-file header standar yang disediakan oleh C++, kita dapat juga membuat file header sendiri, dengan cara yang sama seperti membuat file editor. Yang harus diperhatikan pada saat menyimpan file header yang telah dibuat harus digunakan akhiran **.h** . Berikut contoh file header standar yang disediakan oleh Borland C++.

```

/* types.h
   Types for dealing with time.
   Copyright (c) Borland International 1987,1988
   All Rights Reserved.
*/

#ifndef      TIME_T
#define      TIME_T
typedef long time_t;
#endif

```

Sebagai latihan berikut ini akan dibuatkan suatu file header sendiri yang akan digunakan pada file editor.

Buatlah program file heder dibawah ini, kemudian simpan dengan nama : **atur.h**, pada folder kerja anda folder include.

Contoh-8

```
/* atur.h
   contoh pembuatan file header untuk
   pengaturan.
*/
#define awal {
#define akhir }
#define mulai( ) main( )
#define cetak cout
#define tampil printf
#define masuk scanf
#define hapus( ) clrscr( )
#define jika if
#define warna textcolor
#define tahan getche( )
```

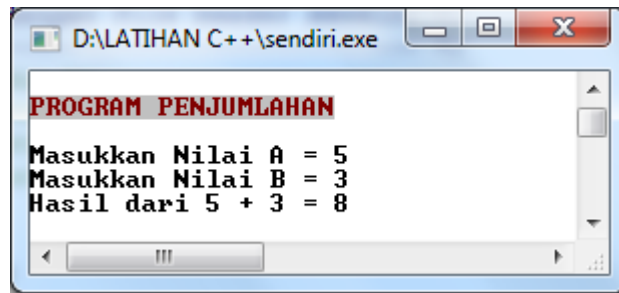
Setelah disimpan pada direktori C:\BC5\INCLUDE\... , selanjutnya Compile file **atur.h**.

Buatlah program dibawah ini, kemudian gunakan file header yang sudah anda buat dan simpan dengan nama : **sendiri.cpp**

Contoh-9

```
/* ----- */
/* program dengan file header sendiri */
/* ----- */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include"atur.h"
mulai( )
awal
    int a, b, c;
    hapus( );
    warna(4);
    tampil("\nPROGRAM PENJUMLAHAN\n");
    cout<<endl;
    cout<<"Masukkan Nilai A = ";
    cin>>a;
    cout<<"Masukkan Nilai B = ";
    cin>>b;
    c=a+b;
    cout<<"Hasil dari "<<a<<" + "<<b<<" = "<<c;
    tahan;
akhir
```

Hasil dari program contoh-9 diatas adalah:



Gambar 8.9. Hasil Contoh-9

7.9. Latihan

1. Buatlah program menghitung pangkat dua serta pangkat tiga dari sebuah bilangan bulat dengan makro. Sebagai input adalah bilangan itu sendiri, sedangkan sebagai output adalah pangkat dua serta pangkat tiga dari bilangan bulat tersebut.

2. Buatlah program menghitung luas dan keliling lingkaran. Proses berada didalam file header, nama file header yang diinginkan : **lingkaran.h**

Tampilan Yang Diinginkan:

Masukkan Nilai Jari-jari : ... <di-input>

Luas Lingkaran : ... < hasil proses >

Keliling Lingkaran : ... < hasil proses >

3. Buatlah program menghitung nilai akhir perkuliahan pada suatu matakuliah, dengan ketentuan sebagai berikut:
 - Nilai Absensi * 10 %
 - Nilai Tugas * 20 %
 - Nilai U.T.S * 30 %
 - Nilai U.A.S * 40 %

Untuk proses penilaian dilakukan didalam file header dan simpan nama file header tersebut **hitnilai.h**.

Tampilan yang diinginkan:

Program Hitung Nilai Akhir Mata Kuliah

Masukkan Nilai Absensi :<di-input>

Masukkan Nilai Tugas :<di-input>

Masukkan Nilai U.T.S :<di-input>

Masukkan Nilai U.A.S :<di-input>

Nilai Murni Absensi = <data-inputan> * 10% = <hasil-proses>

Nilai Murni Tugas = <data-inputan> * 20% = <hasil-proses>

Nilai Murni U.T.S = <data-inputan> * 30% = <hasil-proses>

Nilai Murni U.A.S = <data-inputan> * 40% = <hasil-proses>

Nilai Akhir yang diperoleh sebesar : <hasil-proses>

UTS



Mahasiswa melakukan Ujian Tengah Semester

Fungsi

Fungsi (*Function*) merupakan blok dari kode yang dirancang untuk melaksanakan tugas khusus. Kegunaan dari fungsi ini adalah untuk:

- Mengurangi pengulangan penulisan program yang berulang atau sama.
- Program menjadi lebih terstruktur, sehingga mudah dipahami dan dapat lebih dikembangkan.

Fungsi-fungsi yang sudah kita kenal sebelumnya adalah fungsi *main()*, yang bersifat mutlak, karena fungsi ini program akan dimulai, sebagai contoh yang lainnya fungsi *printf()*, *cout()* yang mempunyai tugas untuk menampilkan informasi atau data kelayar dan masih banyak lainnya.

9.1. Struktur Fungsi

Sebuah fungsi sederhana mempunyai bentuk penulisan sebagai berikut:

```
nama_fungsi(argumen)
{
... pernyataan / perintah;
... pernyataan / perintah;
... pernyataan / perintah;
}
```

Keterangan:

- **Nama fungsi**, boleh dituliskan secara bebas dengan ketentuan, tidak menggunakan spasi dan nama-nama fungsi yang mempunyai arti sendiri.
- **Argumen**, diletakan diantara tanda kurung “()” yang terletak dibelakang nama fungsi. Argumen boleh diisi dengan suatu data atau dibiarkan kosong.
- **Pernyataan / perintah**, diletakan diantara tanda kurung ‘{ }’.

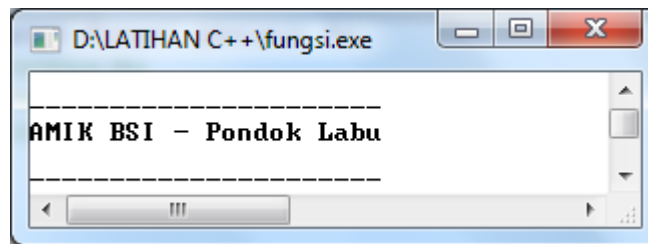
Pada pemanggilan sebuah fungsi, cukup dengan menuliskan nama fungsinya.

Contoh pembuatan fungsi sederhana

```
Contoh-1      /* pembuatan fungsi garis() */
               #include<conio.h>
               #include<stdio.h>
               #include<iostream.h>
               garis()
               {
                 printf("\n-----\n");
```

```
    }  
  
    /* program utama */  
    main( )  
    {  
        clrscr( );  
        garis( ); //memanggil fungsi garis  
        cout<<"AMIK BSI - Pondok Labu"<<endl;  
        garis( ); //memanggil fungsi garis  
        getch( );  
    }  
}
```

Output yang akan dihasilkan, dari program contoh-1 diatas adalah:



Gambar 9.1. Hasil Contoh-1

9.2. Prototipe dan Parameter Fungsi

Prototipe merupakan uraian dari blok fungsi yang dapat digunakan untuk mendeklarasikan ke kompiler mengenai:

- Tipe data keluaran dari fungsi.
- Jumlah parameter yang digunakan
- Tipe data dari masing-masing parameter yang digunakan.

Prototipe fungsi dituliskan di atas blok program utama dan diakhiri dengan tanda qualifier titik koma (;), sedangkan blok program fungsi yang mengandung perintah-perintah atau pernyataan-pernyataan dari program berada di bawah blok program utama yang memiliki keuntungan sebagai berikut:

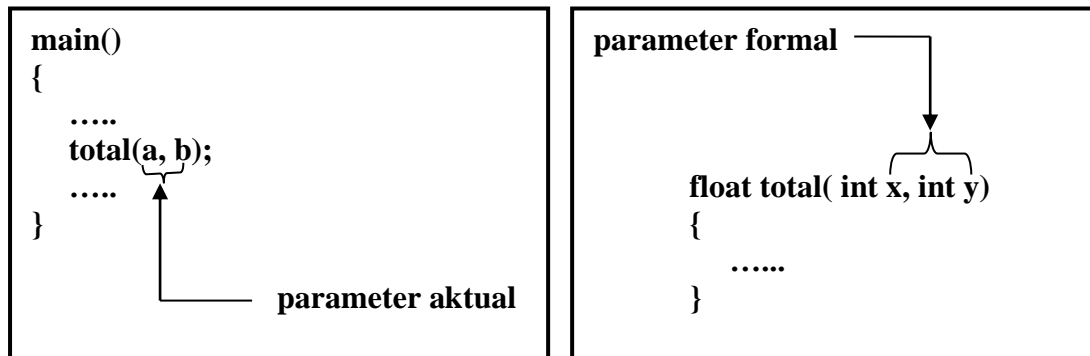
- Kompiler akan melakukan konversi antara tipe parameter dalam definisi dan parameter fungsi.
- Jika jumlah parameter yang digunakan dalam definisi fungsi dan pada saat pemanggilan fungsi berbeda atau tidak sama, maka akan menunjukkan kesalahan.

Sedangkan yang dimaksud dengan parameter pada fungsi adalah suatu pendefinisian nilai-nilai dari objek-objek yang dideklarasikan pada bagian argumen di fungsi. Nilai-nilai pada objek-objek tersebut didapat dari variabel-variabel yang berada pada program utama.

Terdapat dua macam para parameter fungsi, yaitu :

- **Parameter formal** adalah variabel yang terdapat pada daftar parameter yang berada didalam definisi fungsi.
- **Parameter Aktual** adalah variabel yang digunakan pada pemanggilan suatu fungsi.

Bentuk penulisan Parameter Formal dan Parameter Aktual.



Contoh penggunaan prototipe fungsi dan parameter fungsi sebagai berikut:

Contoh-2

```
//Penggunaan Prototipe pada Fungsi
#include <conio.h>
#include <iostream.h>
#include <string.h> //untuk strcpy

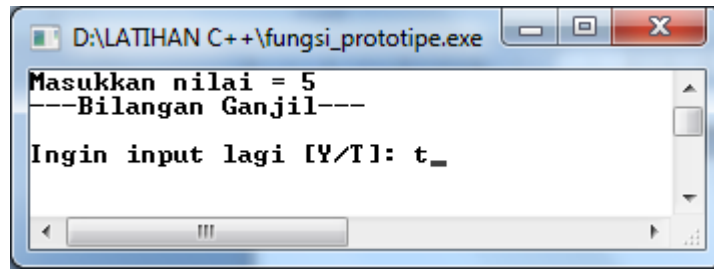
char coment (char ket[30],int n); //prototipe fungsi

main( )
{
    char lagi,c[30];
    int i;
    atas:
    clrscr( );
    {
        cout<<"Masukkan nilai = ";cin>>i;
        coment(c,i); //paramater formal
        cout<<c;
        cout<<"\n\nIngin input lagi [Y/T]: ";cin>>lagi;
    }
    if (lagi=='Y' || lagi=='y')
        goto atas;
    else
        getch( );
}

//blok program fungsi dengan parameter aktual
char coment (char ket[30],int n)
{
    int a;
    a=n%2;
    if (a==1)
        strcpy(ket,"---Bilangan Ganjil---");
    else
        strcpy(ket,"---Bilangan Genap---");
}
```

}

Output yang akan dihasilkan, dari program contoh-2 diatas adalah:



Gambar 9.2. Hasil Contoh-2

Di dalam bahasa C++ ada dua cara untuk melewati nilai-nilai parameter ke dalam fungsi dari nilai-nilai variabel, yaitu:

9.2.1. Pemanggilan dengan nilai (*Call by Value*)

Pada pemanggilan dengan nilai yaitu nilai dari parameter aktual akan dimasukkan ke parameter formal. Dengan cara ini nilai parameter aktual tidak dapat berubah, walaupun nilai dari parameter formal berubah. Berikut contoh pemanggilan dengan nilai dapat dilihat pada contoh berikut:

Contoh-3

```

/* ----- */
/* Penggunaan Call By Value */
/* Program Tambah Nilai */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

tambah(int m, int n);

main( )
{
    int a, b;
    a = 5;
    b = 9;
    clrscr( );
    cout<<"Nilai Sebelum Fungsi Digunakan ";
    cout<<"\na = "<<a<<" b = "<<b;
    tambah(a,b);
    cout<<"\nNilai Setelah Fungsi Digunakan";
    cout<<"\na = "<<a<<" b = "<<b;
    getch( );
}

tambah(int m, int n)
{
    m+=5;
    n+=7;
    cout<<"\n\n Nilai di dalam Fungsi Tambah( )";
}

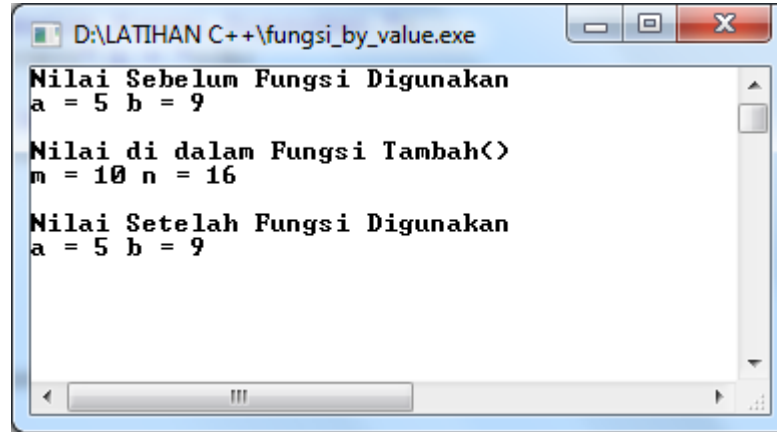
```

```

        cout<<"\n m = "<<m<<" n = "<<n;
        cout<<endl;
    }

```

Output yang akan dihasilkan, dari program contoh-3 diatas adalah :



Gambar 9.3. Hasil Contoh-3

9.2.2. Pemanggilan dengan Referensi (*Call by Reference*)

Pemanggilan dengan referensi merupakan pemanggilan nilai suatu parameter di dalam fungsi ke parameter actual yang disimpan pada alamat memori dengan menggunakan **pointer**. Cara ini dapat dipakai untuk mengubah isi suatu parameter aktual dengan melaksanakan perubahan nilai dari suatu parameter yang dilakukan di dalam fungsi.

Contoh-4

```

/* ----- */
/* Penggunaan Call By Reference */
/* Program Tambah Nilai */
/* ----- */

#include<conio.h>
#include<stdio.h>
#include<iostream.h>

tambah(int *c, int *d); // deklarasi prototype fungsi

// program utama
main( )
{
    int a, b;
    a = 4;
    b = 6;
    clrscr( );
    cout<<" Nilai Sebelum Pemanggilan Fungsi";
    cout<<"\n a = "<<a<<" b = "<<b;
    tambah(&a,&b);
    cout<<endl;
    cout<<"\n Nilai Setelah Pemanggilan Fungsi";
    cout<<"\n a = "<<a<<" b = "<<b;

```

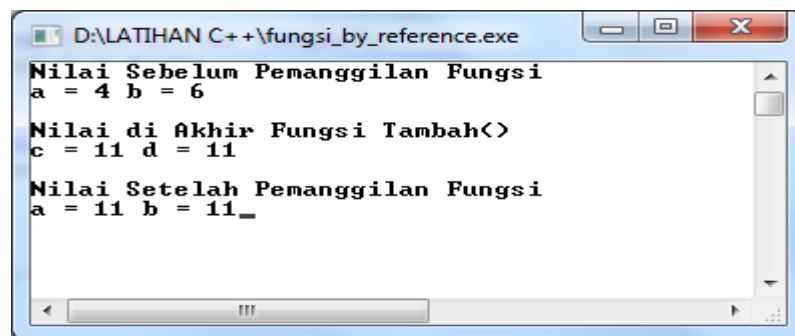
```

        getch( );
    }

    //blok program fungsi
    tambah(int *c, int *d)
    {
        *c+=7;
        *d+=5;
        cout<<endl;
        cout<<"\n Nilai di Akhir Fungsi Tambah()";
        cout<<"\n c = "<<*c<<" d = "<<*d;
    }

```

Output yang akan dihasilkan, dari program contoh-4 diatas adalah:



Gambar 9.4. Hasil Contoh-4

9.3. Pernyataan *return()*.

Digunakan untuk mengirimkan nilai atau nilai dari suatu fungsi kepada fungsi yang lain yang memanggilmnya. Pernyataan *return()* diikuti oleh argumen yang berupa nilai yang akan dikirimkan. Syarat utama dalam pembuatan fungsi return adalah nama fungsi yang dapat mengembalikan nilai, dikarenakan nilai hasil dari parameter formal akan disimpan pada nama fungsi. Contoh pemakaian pernyataan *return()* dapat dilihat pada contoh berikut:

```

Contoh-5      /* Pernyataan Return pd Fungsi */
                #include <conio.h>
                #include <iostream.h>
                #include <stdio.h>

                float luas (int r) //fungsi luas lingkaran
                {
                return(3.14*r*r); }

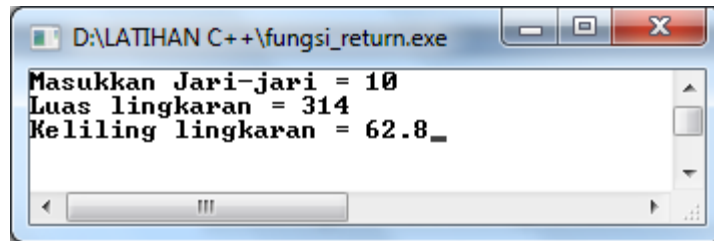
                float kel (int r) //fungsi kel lingkaran
                {
                return(3.14*2*r); }

                main( )
                {

```

```
int j;  
clrscr( );  
cout<<"Masukkan Jari-jari = ";cin>>j;  
cout<<"Luas lingkaran = "<<luas(j)<<endl;  
cout<<"Keliling lingkaran = "<<kel(j);  
getch( );  
}
```

Output yang akan dihasilkan, dari program contoh-5 diatas adalah:



Gambar 9.5. Hasil Contoh-5

9.4. Jenis Variabel

Jenis Variabel pada C++ ini sangat berguna didalam penulisan suatu fungsi agar penggunaan didalam penggunaan suatu variabel tidak salah. Terdapat beberapa jenis variabel yaitu:

- Variabel Lokal.
- Variabel Eksternal atau Global.
- Variabel Statis.

9.4.1. Variabel Lokal

Variabel Lokal adalah variabel yang dideklarasikan didalam fungsi dan hanya dikenal oleh fungsi yang bersangkutan. Variabel lokal biasa disebut dengan *Variabel Otomatis*.

Contoh-6

```
/* ----- */  
/* Variabel Lokal */  
/* ----- */  
#include<conio.h>  
#include<stdio.h>  
#include<iostream.h>  
  
lokal( );  
  
main( )  
{  
    int a = 15;  
    clrscr( );  
  
    cout<<"Pemanggilan Variabel Lokal"<<endl;  
    cout<<"\nNilai didalam fungsi main() = : "<<a;
```

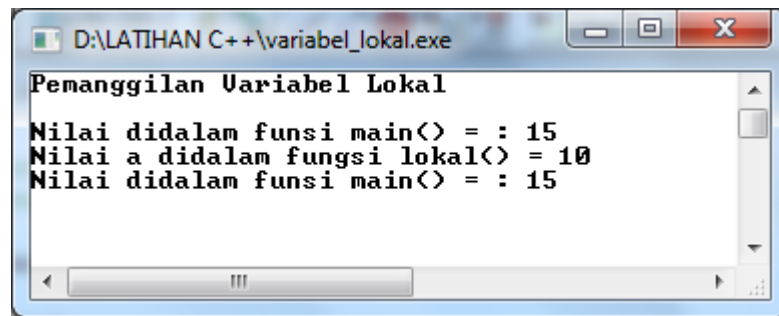
```

lokal();
cout<<"\nNilai didalam fungsi main() = : "<<a;
getch();
}

lokal()
{
    int a = 10;
    cout<<"\nNilai a didalam fungsi lokal() = "<<a;
}

```

Output yang akan dihasilkan, dari program contoh-6 diatas adalah:



Gambar 9.6. Hasil Contoh-6

9.4.2. Variabel Eksternal

Variabel Eksternal adalah variabel yang dideklarasikan diluar fungsi yang bersifat **global** yang artinya dapat digunakan bersama-sama tanpa harus dideklarasikan berulang-ulang. Untuk pendeklarasian variabel eksternal ini, diluar dari fungsi main(), yang selama ini pendeklarasian variabel selalu didalam fungsi main().

Contoh-7

```

/* ----- */
/* Variabel Eksternal atau Global */
/* ----- */
#include<conio.h>

#include<stdio.h>
#include<iostream.h>

int a = 6; //--> deklarasi variabel eksternal

void lokal ();
void main ()
{
    clrscr();
    cout<<"Penggunaan Variabel Eksternal"<<endl;
    cout<<"\nNilai didalam fungsi main() = : "<<a;
    lokal (); //--> pemanggilan fungsi lokal
}

```

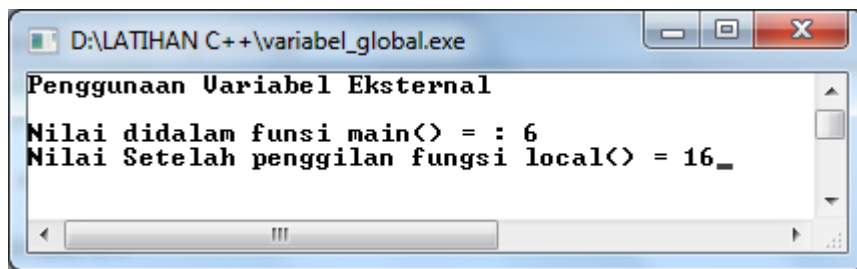
```

        cout<<"\nNilai Setelah panggilan fungsi lokal() = ";
        cout<<a;
        getch();
    }

    void lokal( )
    {
        a+=10;
    }

```

Output yang akan dihasilkan, dari program contoh-7 diatas adalah:



Gambar 9.7. Hasil Contoh-7

9.4.3. Variabel Statis

Variabel Statis dapat berupa variabel local atau variabel eksternal Sifat variabel statis ini mempunyai sifat antar lain.

- Jika variabel statis bersifat local, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan.
- Jika variabel statis bersifat eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama ditempat variabel statis dideklarasikan.
- Jika tidak ada inisialisasi oleh pemrograman secara otomatis akan diberikan nilai awal nol.

Suatu variabel statis diperoleh dengan menambahkan kata-kunci *static* didepan penentu tipe data variabel.

Contoh-8

```

/* ----- */
/* Penggunaan Variabel Statis */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
walah( ); //--> prototipe fungsi walah

main( )
{
    int k = 5;

```

```

clrscr();
walah();
walah();
cout<<"\nNilai K didalam fungsi main() = "<<k;
getch();
}

walah()
{
static int k; // --> deklarasi variabel statis
k += 4;
cout<<"\nNilai K didalam fungsi() = "<<k;
}

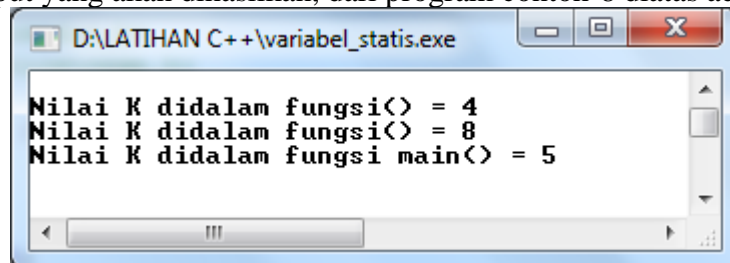
```



Hal ini terlihat bahwa :

- Pada pada prototipe fungsi walah() tidak terdapat nilai awal, maka secara otomatis variabel k = 0.
- Pada pemanggilan fungsi walah() pertama, tercetak nilai variabel k = 4, didapat dari k=0+4.
- Pada pemanggilan fungsi walah() kedua, tercetak nilai variabel k = 8, didapat dari k=4+4, karena nilai k yang terbaru adalah 4.
- Pada pencetakan k didalam fungsi main(), adalah 5, karena variabel k, didalam fungsi main() bersifat lokal.

Output yang akan dihasilkan, dari program contoh-8 diatas adalah :



Gambar 9.8. Hasil Contoh-8

9.5. Fungsi inline

Fungsi inline (*inline function*) digunakan untuk mengurangi lambatnya eksekusi program dan mempercepat eksekusi program terutama pada program yang sering menggunakan atau memanggil fungsi yang berlebih. terutama program-program yang menggunakan pernyataan perulangan proses seperti for, while dan do – while. Inline function dideklarasikan dengan menambahkan kata kunci **inline** didepan tipe data.

Contoh-9

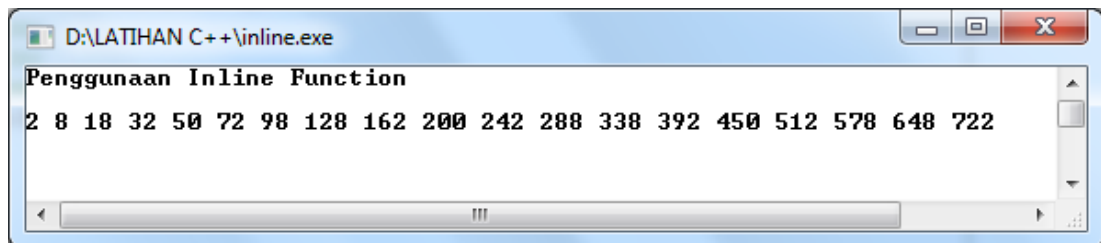
```

/* ----- */
/* Penggunaan inline function */
/* ----- */
#include<conio.h>

```

```
#include<stdio.h>
#include<iostream.h>
inline int kali(int i, int j)
{
return(i * j);
}
main( )
{
int k;
clrscr( );
for(k = 1; k < 20; k++)
cout<<kali(k, k*2)<<" ";
getch( );
}
```

Output yang akan dihasilkan, dari program contoh-9 diatas adalah:



Gambar 9.9. Hasil Contoh-9

Contoh-10

```
/* ----- */
/* Penggunaan inlide function */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

inline static void cplusplus( )
{

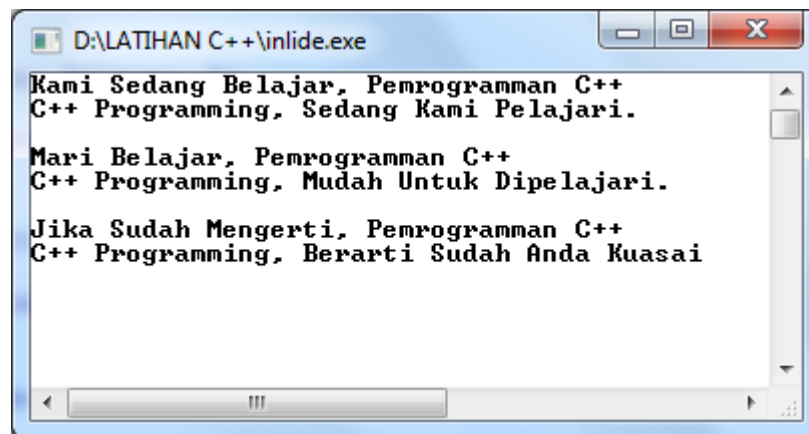
cout << "Pemrogramman C++\n";
cout << "C++ Programming, ";
}
int main( )
{
{
cout << "Kami Sedang Belajar, ";
cplusplus( );
cout << "Sedang Kami Pelajari.\n\n";
}
{
cout << "Mari Belajar, ";
cplusplus( );
}
```

```

cout << "Mudah Untuk Dipelajari.\n\n";
}
{
cout << "Jika Sudah Mengerti, ";
cplusplus( );
cout << "Berarti Sudah Anda Kuasai";
}
getche( );
}

```

Output yang akan dihasilkan, dari program contoh-10 diatas adalah :



Gambar 9.10. Hasil Contoh-10

9.6. Function Overloading

Function Overloading adalah mendefinisikan beberapa fungsi, sehingga memiliki nama yang sama tetapi dengan parameter yang berbeda. Dapat diartikan bahwa fungsi yang overload berarti menyediakan versi lain dari fungsi tersebut. Salah satu kelebihan dari C++ adalah Overloading. s Sebagai contoh membentuk fungsi yang sama dengna tipe yang berbeda-beda dan dibuatkan pula nama fungsi yang berbeda-beda pula.

Contoh-11

```

/* ----- */
/* Penggunaan function overloading */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

int hitung(int b);
long hitung(long c);
float hitung(float d);

void main( )
{

```

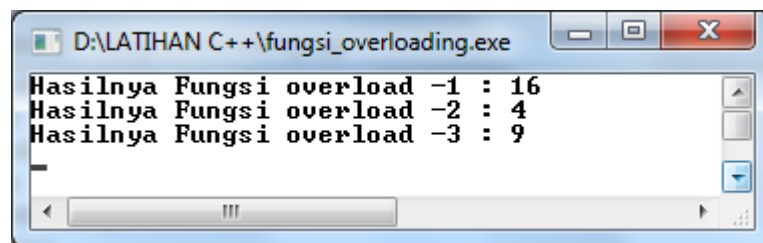
```
clrscr();
cout<< "Hasilnya Fungsi overload -1 : ";
cout<<hitung(4)<<endl;
cout<< "Hasilnya Fungsi overload -2 : ";
cout<<hitung(2)<<endl;
cout<< "Hasilnya Fungsi overload -3 : ";
cout<<hitung(3)<<endl;
getch();
}

int hitung(int b)
{
    return(b*b);
}

long hitung(long c)
{
    return(c*c);
}

double hitung(double d)
{
    return(d*d);
}
```

Output yang akan dihasilkan, dari program contoh-11 diatas adalah :



Gambar 9.11. Hasil Contoh-11

9.7 Prosedur

Prosedur adalah sederetan instruksi algoritmik yang diberi nama, dan akan menghasilkan efek neto yang terdefinisi. Prosedur menyatakan suatu aksi dalam konsep algoritma yang dibicarakan pada cerita “Mengupas kentang”. Dimana contoh ini dari aksi yang dilakukan oleh Ibu Tati yang mengupas kentang untuk mempersiapkan makan malam sebagai berikut. Pernyataan ini mencakup hal yang luas ruang lingkungannya, misalnya :

- 1) Apakah kentangnya harus dibeli dulu atau sudah ada di dapur ?

2) Apakah yang dimaksud dengan mengupas kentang untuk makan malam berarti sampai dengan kentang terhidang ?

Ketika kentangnya terhidang, jadi sup, digoreng atau direbus saja ? Maka kita harus membatasi dengan jelas keadaan awal yang menjadi titik tolak mengupas kentang dan keadaan akhir yang ingin dicapai supaya dapat “merencanakan” efek neto yang diinginkan. Sehingga hal tersebut dapat ditentukan:

- Initial state (I.S. keadaan awal), T0, adalah kentang sudah ada di kantong kentang, yang ditaruh di rak di dapur, di mana ibu Tati akan mengupasnya
- Final state (F.S. keadaan akhir), T1, kentang dalam keadaan terkupas di panci, siap untuk dimasak dan kantong kentangnya harus dikembalikan ke rak lagi.

Pengandaian yang lain adalah bahwa persediaan kentang di ibu selalu cukup untuk makan malam. Penambahan kentang ke dapur di luar tinjauan masalah ini. Ini adalah contoh bagaimana kita menentukan batasan dari persoalan yang akan diprogram. Suatu kejadian dapat dipandang sebagai urutan dari beberapa kejadian, berarti dapat diuraikan dalam beberapa (sub) aksi yang terjadi secara sekuensial. Dengan sudut pandang ini makan efek kumulatifnya sama dengan efek neto dari seluruh kejadian. Dikatakan bahwa kejadian tersebut dianggap sebagai sequential process atau disingkat proses.

Mendefinisikan (membuat spesifikasi) prosedur berarti menentukan nama prosedur serta parameternya (jika ada), dan mendefinisikan keadaan awal (Initial State, I.S.) dan keadaan akhir (Final State, F.S.) dari prosedur tersebut. Prosedur didefinisikan (dituliskan spesifikasinya) dalam kamus. Cara penulisan spesifikasi : prosedur diberi nama, dan parameter formal (jika ada) yang juga diberi nama dan dijelaskan tipenya.

Secara sederhana, dapat diartikan bahwa sebuah prosedur yang terdefinisi “disimpan” di tempat lain, dan ketika “dipanggil” dengan menyebutkan namanya “seakan-akan” teks yang tersimpan di tempat lain itu menggantikan teks pemanggilan. Pada saat itu terjadi asosiasi parameter (jika ada). Dengan konsep ini, maka I.S dan F.S dari prosedurlah yang menjamin bahwa eksekusi program akan menghasilkan efek neto yang diharapkan. Jadi, setiap prosedur harus :

- a. Didefinisikan (dibuat spesifikasinya) dan dituliskan kode programnya
- b. Dipanggil, pada saat eksekusi oleh prosedur lain atau oleh program utama

9.8 Parameter Prosedur

Prosedur tanpa parameter memanfaatkan nilai dari nama-nama yang terdefinisi pada kamus global. Pemakaiannya biasanya harus “hati-hati”, apalagi jika teks program sudah sangat besar dan implementasinya menjadi banyak file. Prosedur berparameter dirancang, agar sepotong kode yang sama ketika eksekusi dilakukan, dapat dipakai untuk nama parameter yang berbeda-beda. Nama parameter yang dituliskan pada definisi/spesifikasi prosedur disebut sebagai parameter formal. Sedangkan parameter yang dituliskan pada pemanggilan prosedur disebut sebagai parameter aktual.

Parameter formal adalah nama-nama variabel (list nama) yang dipakai dalam mendefinisikan prosedur, dan membuat prosedur tersebut dapat dieksekusi dengan nama-nama yang berbeda ketika dipanggil. Parameter formal adalah list nama yang akan dipakai pada prosedur, yang nantinya akan diasosiasikan terhadap nama variabel lain pada saat pemanggilan. Sesuai dengan ketentuan nilainya, ada tiga type parameter formal:

- ✓ parameter Input, yaitu parameter yang diperlukan prosedur sebagai masukan untuk melakukan aksi yang efektif.
- ✓ parameter Output, yaitu parameter yang nilainya akan dihasilkan oleh prosedur. Hasil nilai akan disimpan pada nama parameter Output ini.

Parameter Input/Output, yaitu parameter yang nilainya diperlukan prosedur sebagai masukan untuk melakukan aksi, dan pada akhir prosedur akan dihasilkan nilai yang baru.

9.9 Pemanggilan Prosedur

Memakai, atau "memanggil" prosedur adalah menuliskan nama prosedur yang pernah didefinisikan, dan memberikan harga-harga yang dibutuhkan oleh prosedur itu untuk dapat melaksanakan suatu aksi terdefinisi. Sebuah prosedur juga boleh "memakai" atau memanggil prosedur. Pada saat pemanggilan terjadi "passing parameter". Parameter aktual adalah nama-nama informasi yang dipakai ketika prosedur itu dipakai ("dipanggil"). Parameter aktual dapat berupa nama atau harga, tetapi harus berupa nama jika parameter tersebut adalah parameter Output (karena hasilnya akan disimpan dalam nama tersebut). Sesuai dengan jenis parameter formal, parameter aktual pada saat pemanggilan :

- ✓ parameter input harus terdefinisi nilainya (karena dibutuhkan oleh prosedur untuk menghasilkan nilai). Karena yang dibutuhkan untuk eksekusi hanya nilai, maka parameter input dapat digantikan dengan suatu nilai tanpa menggunakan nama.
- ✓ parameter output tidak perlu terdefinisi nilainya, tetapi justru setelah pemanggilan prosedur akan dimanfaatkan oleh deretan instruksi berikutnya, karena nilainya akan dihasilkan oleh prosedur. Karena parameter output menampung hasil, maka harus berupa nama, dan tidak boleh diberikan nilai saja.
- ✓ parameter input/output harus terdefinisi nilainya dan nilai baru yang diperoleh karena eksekusi prosedur akan dimanfaatkan oleh deretan instruksi berikutnya. Seperti halnya parameter output, maka parameter aktual harus berupa nama.

Pada saat eksekusi, terjadi asosiasi nama parameter formal dengan nama parameter aktual. Pada notasi algoritmik, asosiasi dilakukan dengan cara "by position". Urutan nama pada parameter aktual akan diasosiasikan sesuai dengan urutan parameter formal. Karena itu, type harus kompatibel. Beberapa bahasa pemrograman, dapat dilakukan asosiasi dengan nama dan memperbolehkan adanya nilai default. Beberapa bahasa pemrograman (Ada, C) juga memperbolehkan nama prosedur yang sama tetapi parameternya berbeda (overloading).

Contoh 1-Prosedur: VOLTAGE Tuliskanlah program yang membaca tahanan (Ohm) dan arus (Ampere), kemudian menghitung tegangan yang dihasilkan dan menuliskan hasilnya. Perhitungan tegangan harus dituliskan menjadi suatu prosedur bernama PROSES, supaya struktur program jelas: Input - Proses - Output.

Input : R : integer, tahanan (Ohm) dan A : integer, arus (Ampere)

Proses : menghitung $V = R * A$

Output : V : integer, tegangan (Volt)

Pelajarilah dua buah solusi yang diberikan berikut ini, dan berikan komentar anda.

Solusi 1 : Prosedur tanpa parameter

Program VOLTAGE1

{ Program yang membaca tahanan dan arus, menghitung Voltage dan mencetak hasil perhitungan }

Kamus :

R : integer { tahanan dalam ohm }

A : integer { arus dalam ohm }

V : integer { tegangan dalam ohm }

procedure PROSES1

{ Prosedur untuk "memproses" tahanan dan arus menjadi tegangan }

Algoritma :

input (R,A)

PROSES1

output (V)

Procedure PROSES1

{ I.S : diberikan harga R dan A yang telah terdefinisi }

{ F.S : memproses R dan A sehingga dihasilkan V yaitu tegangan dengan rumus : $V = R * A$ }

Kamus lokal :

Algoritma :

$V \leftarrow R * A$

Solusi 2 : Prosedur dengan parameter

Program VOLTAGE2

{ Program yang membaca tahanan dan arus, menghitung Voltage dan mencetak hasil perhitungan }

Kamus :

R : integer { tahanan dalam ohm }

A : integer { arus dalam ohm }

V : integer { tegangan dalam ohm }

Procedure PROSES2

(Input : R,A : integer; Output V:integer)

{ Prosedur untuk "memproses" tahanan R dan arus A menjadi tegangan V }

Algoritma :

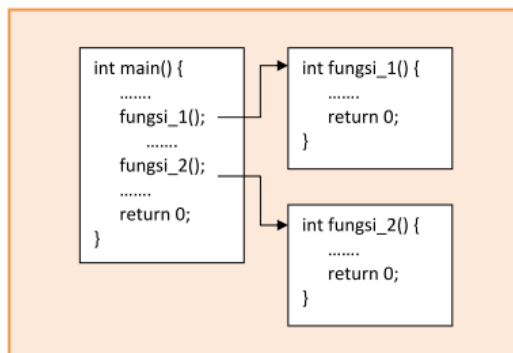
input (R,A)

9.10 Subrutin

Suatu program komputer biasanya merupakan suatu sistem besar yang terdiri dari sub sistem - sub sistem yang mempunyai tugas sendiri-sendiri, saling bekerja sama dan berinteraksi untuk menyelesaikan suatu permasalahan. Dengan adanya pembagian tugas oleh sub sistem – sub sistem ini maka suatu permasalahan dapat diselesaikan dengan lebih cepat dan kesalahan-kesalahan yang mungkin terjadi selama proses penyelesaian masalah dapat dideteksi dan diketahui sedini mungkin, termasuk di sub sistem mana kesalahannya terjadi.

Program komputer (terutama yang kompleks) juga sebaiknya “dipecah-pecah” menjadi program-program kecil. Program-program kecil tersebut disebut dengan sub rutin. Sub rutin dibagi menjadi dua macam, yaitu sub rutin yang mengembalikan nilai dan sub rutin yang tidak mengembalikan nilai. Dalam Pascal kedua sub rutin yang mengembalikan nilai disebut dengan function, sedangkan yang tidak mengembalikan nilai disebut dengan procedure. Tetapi untuk C++ dan Java, kedua sub rutin tersebut dijadikan satu tetapi dapat diatur untuk dapat mengembalikan nilai maupun tidak mengembalikan nilai.

Untuk C++, sub rutin tersebut ada dalam suatu function, sedangkan pada Java, sub rutin berbentuk suatu method yang disebut dengan function method.



Gambar 17.1 Subrutin

Baik C++ maupun Java terdapat dua macam fungsi, yaitu user-defined function dan built-in function. User-defined function merupakan fungsi yang didefinisikan sendiri atau dibuat sendiri oleh pemrogram. Sedangkan built-in function adalah fungsi yang sudah ada atau sudah disediakan oleh kompiler dan siap digunakan oleh pemrogram.

9.11 SubRutin dalam bahasa pemograman

Bentuk umum sub rutin (function) pada C++ dan Java sangat mirip. Untuk function yang mengembalikan nilai, setiap function harus didahului oleh tipe data yang sesuai dengan jenis data yang akan dikembalikan, kecuali tipe data array. Setiap function juga mempunyai daftar parameter dimana setiap parameter dipisahkan dengan tanda koma (.). Parameter-parameter ini digunakan untuk menerima data (nilai) dari program yang memanggilnya. Kita dapat mendeklarasikan banyak parameter atau tidak sama sekali. Untuk fungsi yang

tidak mempunyai parameter merupakan fungsi yang nilainya tetap (tidak berubah). Sedangkan fungsi yang mempunyai parameter nilai fungsinya dinamis (dapat berubah-ubah). Parameter-parameter ini merupakan variabel-variabel yang harus dideklarasikan sendiri-sendiri meskipun tipe datanya sama. Bentuk umum dari function yang mengembalikan nilai adalah sebagai berikut:

```
tipe_data nama_fungsi(daftar_parameter)
{
    isi dari fungsi
    return <ekspresi>
}
```

Perhatikan contoh program function yang benar

```
int contoh(int a, int b) {
    .....
    return(c);
}
```

Sedangkan program yang menggunakan function salah adalah sebagai berikut:

```
int contoh(int a, b) {
    .....
    return(c);
}
```

Setiap function mempunyai kode-kode program sendiri-sendiri karena tugas yang harus diselesaikan oleh setiap function juga berbeda-beda. Variabel-variabel yang digunakan dalam function disebut dengan variabel lokal. Oleh karena itu variabel tersebut hanya dapat digunakan didalam function itu sendiri tidak bisa digunakan oleh function lain atau program utama.

Variabel lokal ini berfungsi pada saat function tersebut aktif dan akan hilang (dihapus) jika function sudah tidak aktif lagi atau setelah function selesai melakukan tugasnya (kecuali variabel yang digunakan dalam function adalah variabel global yang dapat digunakan oleh semua function dan program utama).

Untuk dapat digunakan, function biasanya mempunyai parameter-parameter yang digunakan untuk menerima masukan dari program yang memanggilnya. Parameter-parameter ini disebut dengan parameter formal. Parameter formal ini termasuk dalam variabel lokal yang akan berfungsi pada saat function aktif dan akan dihapus pada saat function selesai melakukan tugasnya. Perhatikan contoh function dalam bahasa C++ berikut ini:

```
/* function akan menghasilkan nilai 1 jika c sama dengan s, sebaliknya bernilai 0 jika
c
tidak sama dengan s */
int cek(char s, char c) {
    if(s==c) return 1;
    return 0;
}
```

Function cek() mempunyai dua buah parameter formal, yaitu s dan c yang mempunyai tipe data yang sama, char. Function cek() merupakan function yang

mengembalikan nilai sehingga dia harus mempunyai tipe data dimana tipe data-nya dalam hal ini adalah integer.

Function cek() ini bertugas untuk memeriksa apakah variabel c sama dengan variabel s. Jika sama maka function cek() akan bernilai 1, sebaliknya jika tidak maka akan bernilai 0. Variabel s dan c ini merupakan variabel lokal yang hanya dapat digunakan dalam function cek() saja, tidak bisa digunakan oleh function atau program lain. Permasalahan yang sama untuk Java adalah sebagai berikut:

```
class ricek {  
    public int cek(char s, char c) {  
        if(s==c) return 1;  
        return 0;  
    }  
}
```

Telah disebutkan sebelumnya bahwa sub rutin dalam Java berbentuk class dimana didalam class tersebut dimungkinkan untuk mempunyai satu atau lebih function method. Class ricek() ini dapat digunakan oleh class lain, dalam hal ini untuk mengecek suatu karakter karena class ricek() mempunyai function method cek(). Berikut ini adalah kode lengkap dari kedua program di atas.

Perhatikan Program dalam bahasa C++ :

```
#include <iostream>  
using namespace std;  
  
int cek(char s, char c) {  
    if(s==c) return 1;  
    return 0;  
}  
  
int main(void) {  
    char a,b;  
    a = 'a';  
    b = 'a';  
    cout << cek(a,b) << endl;  
    return 0;  
}
```

Keluaran program adalah :

1

Dari kode program C++ di atas terlihat bahwa function cek() dipanggil melalui argumen pada program utama (main()) dimana argumen tersebut mempunyai variabel masukan untuk function cek() yaitu variabel a dan b. Variabel a dan b ini kemudian disalin oleh parameter formal function cek() yaitu s dan c. Parameter formal ini kemudian diproses lebih lanjut yaitu pengecekan apakah parameter formal s sama dengan parameter formal c. Jika sama maka function cek() akan bernilai 1, sebaliknya jika tidak sama maka function cek() akan bernilai 0. Nilai dari function cek() ini kemudian langsung dicetak ke layar oleh program utama. Kode program untuk bahasa Java dari permasalahan yang sama adalah:

```
class ricek {
    public int cek(char s, char c) {
        if(s==c) return 1;
        return 0;
    }
}
class ricekApp {
    public static void main
(String[ ] args) {
    ricek ck = new ricek();
    char a='a',b='b';
    System.out.println(ck.cek(a,b));
}
}
```

Keluaran program adalah :
0

Dalam Java, class ricek() yang mempunyai function method cek() tidak bisa langsung kita gunakan (kita panggil), tetapi harus dibuat / diciptakan dahulu obyek dari class ricek() seperti yang terlihat pada baris ke-9 dimana obyek baru dari class ricek() dalam class ricekApp() diberi nama ck. Dengan obyek ck inilah kita dapat mengakses method yang dipunyai oleh class ricek() karena secara otomatis obyek ck juga mempunyai method cek() yang dimiliki oleh class ricek().

Berikut ini adalah contoh function yang tidak mempunyai parameter sama sekali sehingga nilai function-nya tidak akan berubah-ubah seperti halnya pada contoh sebelumnya.

Contoh program dalam bahasa C++ :

```
#include <iostream>
using namespace std;
int fpb(){
    int a=24,b=18,hasil;
    int r = a % b;
    if (r==0) hasil = b;
    else {
        while(r!=0) {
            a = b;
            b = r;
            r = a % b;
            hasil = b;
        }
    }
    return(hasil);
}
void main() {
    cout << "FPB-nya = ";
    cout << fpb() << endl;
```

```
}
```

Keluaran programnya adalah :

FPB-nya = 6

Contoh program dalam bahasa Java :

```
1. class hitung {
2.     public int fpb(){
3.         int a=78,b=24,hasil=0;
4.         int r = a % b;
5.         if (r==0) hasil = b;
6.         else {
7.             while(r!=0) {
8.                 a = b;
9.                 b = r;
10.                r = a % b;
11.                hasil = b;
12.            }
13.        }
14.        return hasil;
15.    }
16. }
17. class fpbApp {
18.     public static void main(String[ ] args) {
19.         hitung sekutu = new hitung();
20.         System.out.println("Bilangan terbesarnya="+ sekutu.fpb());
21.     }
22. }
```

Keluaran programnya adalah :

Bilangan terbesarnya = 6

Nilai dari function fpb() di atas untuk bahasa C++ pasti 6 karena nilai dari a dan b telah ditetapkan besarnya, yaitu 24 dan 18. Sedangkan untuk bahasa Java nilai method fpb() dari class hitung() juga pasti tetap, yaitu 6 karena nilai a dan b juga telah ditentukan (a=78 dan b=24).

9.12 Tugas 6

1. Buatlah program untuk menghitung besarnya diskon yang diberikan atas besarnya sejumlah pembelian, dengan ketentuan sebagai berikut :
 - Jika belanja dibawah Rp. 1,000,000 , maka tidak mendapat diskon.
 - Jika belanja dimulai dari Rp. 1,000,000 , sampai dengan Rp. 5.000.000, maka mendapat diskon sebesar 20%.
 - Jika belanja diatas Rp. 5.000.000, maka mendapat diskon sebesar 35%.

Fungsi yang harus dibuat *potong()* untuk menghitung besar potongan yang akan diberikan. Dengan tampilan yang diinginkan sebagai berikut:

Program Hitung Potongan.

Besar pembelian barang Rp. <di input >

Besar diskon yang diberikan Rp.< hasil proses >

Besar harga yang harus dibayarkan Rp.< hasil proses >

2. Buatlah program untuk menghitung konversi dari derajat fahrenheit ke celcius

Petunjuk :

- Gunakan Function Overloading.
- Buatlah 3 (tiga) buah fungsi untuk dioverloading, dengan variabel untuk masing-masing fungsi berbeda-beda.
 - Untuk fungsi pertama variabel yang digunakan adalah double
 - Untuk fungsi pertama variabel yang digunakan adalah float
 - Untuk fungsi pertama variabel yang digunakan adalah integer
- Rumus konversi yang digunakan adalah

$$c = (f - 32.0) * 5 / 9;$$

Contoh :

Jika nilai Fahrenheit = 100

$$c = (100 - 32) * 5 / 9;$$

$$c = (68) * 5 / 9;$$

$$c = 37,7778$$

Hasil keluaran yang diinginkan :

Pemanggilan dengan tipe data double

Proses dengan tipe data double

100 sama dengan 37.7778

Pemanggilan dengan tipe data float

Proses dengan tipe data float

100 sama dengan 37.7778

Pemanggilan dengan tipe data integer

Proses dengan tipe data integer

100 sama dengan 37

3. Buatlah program untuk menghitung jumlah pembayaran pada perpustakaan "Kecil-Kecilan". Mempunyai ketentuan sebagai berikut:

Kode Jenis Buku	Jenis Buku	Tarif Buku
C	CerPen (Kumpulan Cerita Pendek)	500
K	Komik	700
N	Novel	1000

Petunjuk Proses :

- Buatlah Fungsi Tarif untuk menentukan tarif penyewaan
- Gunakan Pernyataan If – Else

Tampilan Masukan yang diinginkan:

Perpustakaan "Kecil-Kecilan".

Nama Penyewa Buku : ... <diinput>

Kode Buku [C/K/N] : ... <diinput>

Banyak Pinjam : ... <diinput>

Tampilan Keluaran yang diinginkan:

Tarif Sewa Rp. <hasil proses>

Jenis Buku : < hasil proses >

Penyewa dengan Nama <hasil proses>

Jumlah Bayar Penyewaan Sebesar Rp. <hasil proses>

Structure

Struktur digunakan untuk mengelompokkan sejumlah data yang mempunyai tipe data yang berbeda. Variabel-variabel yang membentuk sebuah struktur dinamakan elemen struktur. Struktur sama seperti Record di dalam Bahasa Pemrograman Pascal

10.1. Deklarasi Structure

Structure dapat deklarasikan seperti berikut:

```
struct nama_tipe_struktur
{
    elemen_struktur;
    ....
    ....
};
```

Atau

```
struct
{
    elemen_struktur;
    ....
    ....
} nama_tipe_struktur;
```

Contoh Deklarasi

```
struct mahasiswa
{
    char nim[5];
    char nama[15];
    float nilai;
};
```

atau

```
struct
{
    char nim[5];
    char nama[15];
    float nilai;
}mahasiswa;
```



Hal yang perlu di perhatikan :

- Penulisan nama Structure jika mengikuti bentuk umum pertama, penggunaan nama structure-nya tidak bisa langsung di gunakan, karena secara otomatis menjadi sebuah tipe data. Dan penggunaannya harus menggunakan objek/nama variabel yang menggunakan dari nama structure itu sendiri
- Penulisan nama Structure dengan mengikuti bentuk umum kedua, maka penggunaan nama structure-nya bisa langsung di aplikasikan.

Contoh-1a

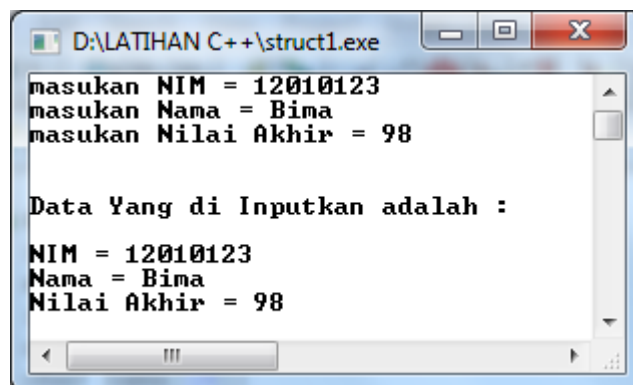
```
/* ----- */
/* Program Penggunaan structure */
/* Nama File : struct1.cpp */
/* ----- */
#include<stdio.h>

#include<conio.h>
#include<iostream.h>

main( )
{
    struct
    {
        char nim[5];
        char nama[15];
        int nilai;
    } mahasiswa;

    clrscr( );
    cout<<"masukan NIM = ";
    cin>>mahasiswa.nim;
    cout<<"masukan Nama = ";
    cin>>mahasiswa.nama;
    cout<<"masukan Nilai Akhir = ";
    cin>>mahasiswa.nilai;
    cout<<"\n\nData Yang di Inputkan adalah : \n\n";
    cout<<"NIM = "<<mahasiswa.nim<<endl;
    cout<<"Nama = "<<mahasiswa.nama<<endl;
    cout<<"Nilai Akhir = "<<mahasiswa.nilai<<endl;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-1 diatas adalah :



Gambar 10.1. Hasil Contoh 1b

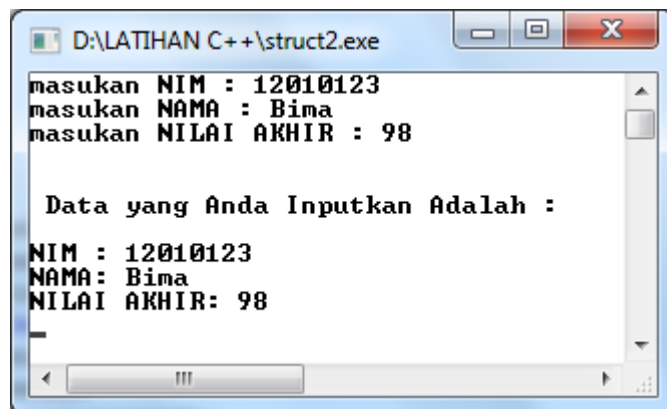
**Bandungkanlah Contoh-1a dengan Contoh-1b berikut ini.
Adakah perbedaannya ?**

Contoh-1b

```
/* ----- */
/* Program Penggunaan structure */
/* Nama File : struct2.cpp */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
main( )
{
    struct data
    {
        char nim[10],nama[15];
        int nilai;
    };

    clrscr( );
    data mahasiswa;
    cout<<"masukan NIM : ";cin>>mahasiswa.nim;
    cout<<"masukan NAMA : ";cin>>mahasiswa.nama;
    cout<<"masukan NILAI AKHIR : ";cin>>mahasiswa.nilai;
    cout<<"\n\n Data yang Anda Inputkan Adalah : "<<endl<<endl;
    cout<<"NIM : "<<mahasiswa.nim<<endl;
    cout<<"NAMA: "<<mahasiswa.nama<<endl;
    cout<<"NILAI AKHIR: "<<mahasiswa.nilai<<endl;
    getch( );
}
```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah :



Gambar 10.1b. Hasil Contoh 1b

10.2. Structure dengan Array dan Function

Penggunaan Array sering dikaitkan dengan Structure, sehingga membentuk Array dari Structure. Berikut bentuk deklarasi array struktur:

```

struct
{
    elemen_struktur ;
    ..... ;
} nama_tipe_struktur[jml_index];

```

Suatu elemen-elemen dari suatu Struktur dapat dikirimkan ke dalam suatu function dengan cara yang sama seperti mengirimkan suatu variabel sederhana ke dalam suatu function.

Berikut contoh sederhana yang anda dapat lihat pada contoh program berikut:

```

Contoh-2      /* ----- */
                /* Program Penggunaan structure pada function */
                /* Nama File : struct4.cpp */
                /* ----- */
                #include<stdio.h>
                #include<conio.h>
                #include<iostream.h>

                char ket(float n);

                main( )
                {
                    int i;
                    struct
                    {
                        char nim[5];
                        char nama[15];
                        float nilai;
                    } mhs[5];

                    clrscr( );
                    for(i=1; i<2; i++)
                    {
                        cout<<"Data Ke - "<<i++<<endl;
                        cout<<"masukan NIM = "; cin>>mhs[i].nim;
                        cout<<"masukan Nama = "; cin>>mhs[i].nama;
                        cout<<"masukan Nilai Akhir = "; cin>>mhs[i].nilai;
                        cout<<endl;
                    }
                    clrscr( );
                    for(i=1; i<2; i++)
                    {
                        cout<<"Data Ke - "<<i++<<endl;
                        cout<<"NIM = "<<mhs[i].nim<<endl;

```

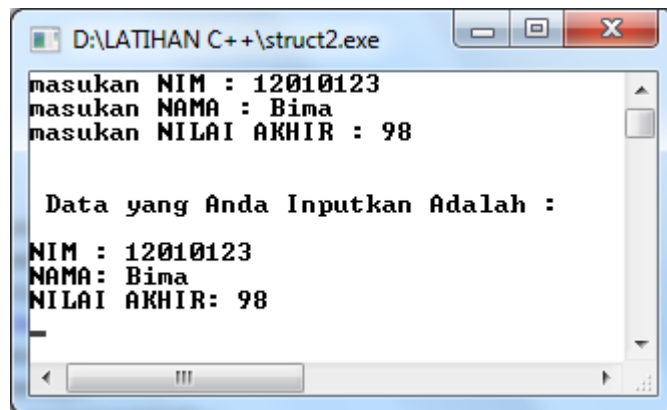
```

        cout<<"Nama = "<<mhs[i].nama<<endl;
        cout<<"Nilai Akhir = "<<mhs[i].nilai<<endl;
        cout<<"Keterangan yang didapat = ";
        cout<<ket(mhs[i].nilai)<<endl;
        cout<<endl;
    }
    getch( );
}

char ket(float n)
{
    if(n > 65)
        return 'L';
    else
        return 'G';
}

```

Output yang akan dihasilkan, dari program contoh-2 diatas adalah :



Gambar 10.2. Hasil Contoh 2

10.3. Latihan

Kerjakan tugas-tugas dibawah ini sesuai dengan petunjuk dan ketentuan pengerjaan yang telah diberikan :

1. Buatlah program untuk menghitung nilai Hasil dari nilai UTS, UAS dan menampilkan nilai huruf yang akan didapat.

Ketentuan :

- Masukan banyak data yang diinginkan untuk menentukan banyak data yang akan diproses.
- Buatlah sebuah function untuk menghitung nilai Hasil

$$\text{Nilai Hasil} = (\text{Nilai UAS} * 40\%) + (\text{Nilai UTS} * 60\%)$$
 - Jika Nilai Huruf = A, maka Nilai Hasil ≥ 80
 - Jika Nilai Huruf = B, maka Nilai Hasil ≥ 70
 - Jika Nilai Huruf = C, maka Nilai Hasil ≥ 56
 - Jika Nilai Huruf = D, maka Nilai Hasil ≥ 47
 - Jika Nilai Huruf = E, maka Nilai Hasil < 47
- Tampilan akhir adalah sebuah tabel, seperti dibawah ini:

Daftar Nilai Mata Kuliah C++

No	Nama Mahasiswa	Nilai UTS	Nilai UAS	Nilai Akhir	Nilai Huruf
...
...

2. Buatlah program untuk menghitung honor pegawai honorer dari suatu perusahaan dengan menghitung kelebihan jumlah jam kerja pegawai tersebut. Honor harian pegawai honorer sebesar Rp. 15000
- Ketentuan :
- Masukan banyak data yang diinginkan untuk menentukan banyak data yang akan diproses.
 - Buatlah sebuah function untuk menghitung honor lembur
- Ketentuan lembur dihitung dari kelebihan jam kerja pegawai tersebut. Jam kerja normal pegawai sebanyak 8 jam
- Jika jumlah jam kerja lebih dari 8 jam, maka kelebihan jam kerja dikalikan Rp. 5000 + Honor harian
 - Jika jumlah jam kerja hanya 8 jam tidak mendapat honor lembur, hanya mendapat honor harian saja.

OOP Dan Karakteristik OOP



11.1. Pengertian OOP

Object Oriented Programming atau yang lebih dikenal dengan OOP adalah pemrograman yang menitikberatkan kepada objek-objek untuk menyelesaikan tugas atau proses dari program tersebut. Sedangkan penitikberatkan ini dimaksudkan adanya interaksi pengiriman nilai, pesan atau pernyataan antar objek. Kemudian objek yang merespon hasil dari interaksi tersebut akan membentuk suatu tindakan atau aksi (*methode*).

Class

Class merupakan gambaran atau abstraksi karakter dan sifat dari suatu objek. *Class* juga dapat mendefinisikan ciri dan perilaku objek tersebut.

Object

Object (objek) adalah suatu data atau entitas yang berwujud maupun tidak berwujud, memiliki sifat (karakteristik) tertentu sesuai dengan kondisi atau status dari penggunaannya. Data atau entitas di dalam pemrograman dapat disebut dengan blok fungsi.

Contoh pensil adalah suatu objek yang memiliki *attribute* (karakter) jenis, warna, panjang dan lain-lain.

Methode

Metoda merupakan tata cara objek tersebut diperlakukan, atau penggunaan atau manfaat dari objek tersebut.

Pensil juga memiliki *methode* (perilaku) seperti diruncingkan, digunakan dan lain-lain.

11.1.1 Keuntungan Penggunaan OOP

Adapun keuntungan atau manfaat dari penggunaan OOP adalah:

1. Natural
OOP dapat melakukan pendekatan terhadap objek yang menggambarkan segala sesuatu yang nyata, seperti sifat suatu benda maupun kegunaan dari benda tersebut.
2. Modular
Objek yang sudah dibentuk dapat dikelompokkan kembali dengan objek-objek yang lain, seperti kelompok alat tulis yang dapat dikelompokkan kembali dengan kelompok pensil, kelompok buku dan lain-lain.
3. Mudah diperbaharui

Dikarenakan sifat jangkauan dari objek memiliki bagian *private* dan *public*, maka jika bagian *private* ingin digunakan pada objek-objek lain dapat diperbaharui dengan menempatkan objek lain tersebut di bagian *public*.

4. Dapat didaur ulang

Suatu objek yang telah didefinisikan baik jenis, bentuk, ciri maupun perilaku dapat didefinisikan kembali dengan objek yang lain. Misalkan objek rumah yang memiliki ciri umum ada pintunya, jendelanya, atapnya, temboknya dan lain-lain, dapat didefinisikan kembali ciri-ciri tersebut dengan menyebutkan cirinya masing-masing seperti temboknya yang memiliki ciri jenisnya, ketebalannya, warna catnya dan lain-lain.

11.1.2 Tata Cara Penggunaan Pemrograman Berbasis Obyek:

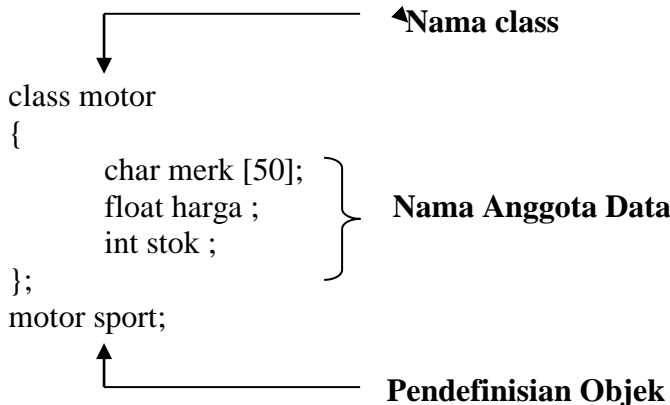
1. Fungsi dan data menjadi satu kesatuan yang disebut obyek
2. Obyek-obyek dalam OOP bersifat aktif
3. Cara pandang : program bukan urutan instruksi tapi diselesaikan oleh obyek-obyek yang bekerjasama untuk menyelesaikan masalah

Bentuk umum dari kelas:

```
class class_name
{
    private:
        data element_class;
        method;

    public:c
        data element_class;
        method;
        prototype function;
};
Object Declaration;
```

Contoh deklarasi :



Pernyataan diatas digunakan untuk mendefinisikan variabel bernama sport. Pada C++ variabel seperti **sport** berkedudukan sebagai variabel kelas yang biasa disebut dengan **objek**.

Pada sebuah kelas, item-item di dalamnya bisa bersifat `private` atau `public`. Secara default, semua item di dalam kelas bersifat **private**. Jadi tanpa menuliskan kata kunci **private**, semua item di dalam kelas sudah `private`.

A. Public pada kelas

Public (*public*) menyatakan bahwa deklarasi variabel atau item-item yang ada di dalam kelas dapat diakses dari luar kelas.

Contoh-1

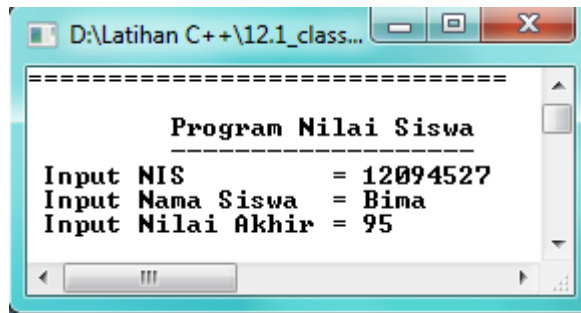
```
//Penggunaan public pada class
#include <iostream.h>
#include <conio.h>

garis( )
{
    cout<<"=====\\n";
}

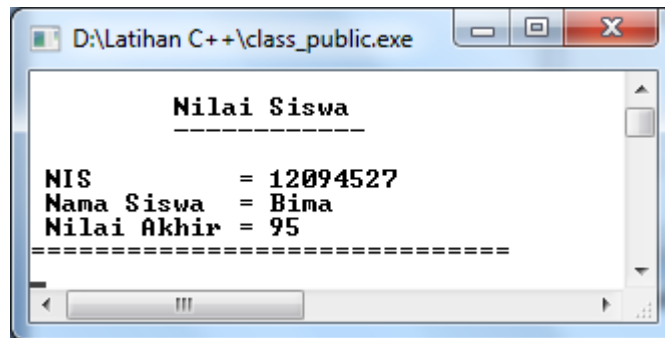
class siswa
{
    public :
    char nis[9],nama[20];
    float nilai;
};

main( )
{
    clrscr( );
    siswa sekolah;
    garis( );cout<<endl;
    cout<<"\\t Program Nilai Siswa"<<endl
    <<"\\t -----"<<endl;
    cout<<" Input NIS      = ";cin>>sekolah.nis;
    cout<<" Input Nama Siswa = ";cin>>sekolah.nama;
    cout<<" Input Nilai Akhir = ";cin>>sekolah.nilai;
    clrscr( );
    garis( );cout<<endl;
    cout<<"\\t Nilai Siswa"<<endl
    <<"\\t -----"<<endl<<endl
    <<" NIS      = "<<sekolah.nis<<endl
    <<" Nama Siswa = "<<sekolah.nama<<endl
    <<" Nilai Akhir = "<<sekolah.nilai<<endl;
    garis( );
    getch( );
}
```

Output yang dihasilkan dari program contoh-1 di atas adalah:



Gambar 11.1 Hasil Contoh-1 (Layar Input)



Gambar 11.2 Hasil Contoh-1 (Layar Output)

B. Private pada Kelas

Private digunakan pada kelas untuk memproteksi anggota-anggota tertentu agar tidak dapat diakses dari luar kelas secara langsung.

Contoh-2

```
//Penggunaan private pada class
#include <conio.h>
#include <iostream.h>
#define pi 3.14

class tabung
{
    private :
        int j,t;
        float v,k;
    public :
        tabung( );
        void keluaran( );
};

void main( )
{
    clrscr( );
    tabung s;
    s.keluaran( );
    getch( );
}

tabung :: tabung( )
```

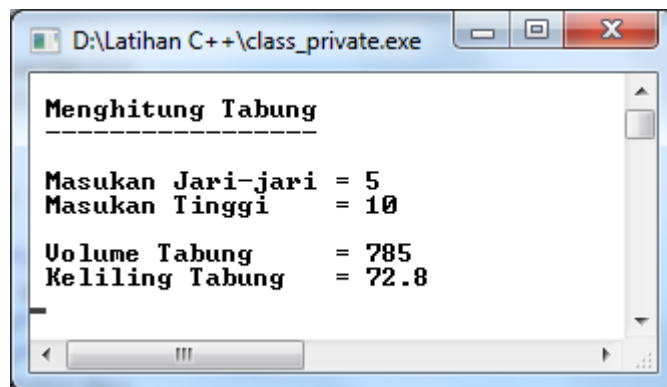
```

{
    cout<<"\n Menghitung Tabung"<<endl
        <<" -----"<<endl<<endl;
    cout<<" Masukan Jari-jari = ";cin>>j;
    cout<<" Masukan Tinggi = ";cin>>t;
    v=(pi*j*j)*t;
    k=(2*(pi*2*j))+t;
}

void tabung :: keluaran()
{
    cout<<endl
        <<" Volume Tabung = "<<v<<endl
        <<" Keliling Tabung = "<<k<<endl;
}

```

Output yang dihasilkan dari program contoh-2 di atas adalah:



Gambar 11.3 Hasil Contoh-2

11.2. Konstruktor

Konstruktor (*constructor*) merupakan suatu fungsi dari anggota suatu kelas yang memiliki nama yang sama dengan nama kelas fungsi itu berada. Konstruktor ini digunakan untuk mengalokasikan ruang untuk suatu objek dan untuk memberikan nilai awal.

Berikut contoh pendeklarasian suatu konstruktor di dalam sebuah kelas:

```

class hitung
{
    private:
        int a;
        int b;
    public:
        int inta( );
        int intb( );
        hitung(int mudah); //deklarasi constructor
};

```

Contoh-3 //Konstruktor
 #include <conio.h>

```
#include <iostream.h>

class bilangan
{
    private :
    int bulat;
    double nyata;
    public :
    bilangan( ); //konstruktor
    void info( );
};

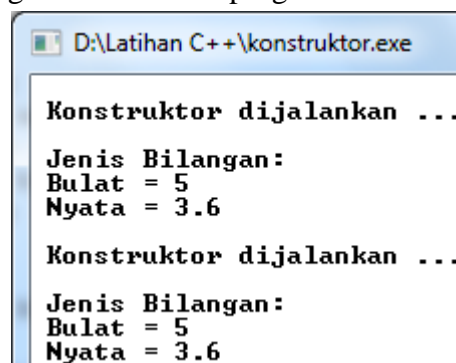
void main( )
{
    clrscr( );
    bilangan a;
    a.info( );
    bilangan b;
    b.info( );
    getch( );
}

bilangan :: bilangan( )
{
    cout<<"\n Konstruktor dijalankan ... "<<endl;
    bulat = 5.2;
    nyata = 3.6;
}

void bilangan :: info( )
{
    cout<<"\n Jenis Bilangan: "<<endl
    <<" Bulat = "<<bulat<<endl
    <<" Nyata = "<<nyata<<endl;
}

```

Output yang dihasilkan dari program contoh-3 di atas adalah:



```
D:\Latihan C++\konstruktor.exe
Konstruktor dijalankan ...
Jenis Bilangan:
Bulat = 5
Nyata = 3.6
Konstruktor dijalankan ...
Jenis Bilangan:
Bulat = 5
Nyata = 3.6

```

Gambar 11.4 Hasil Contoh-3

11.3. Destruktor

Destruktor merupakan suatu fungsi anggota yang dijalankan secara otomatis manakala suatu objek akan terbebas dari memori karena lingkup keberadaannya telah menyelesaikan tugasnya.

Destruktor harus mempunyai nama yang sama dengan kelas dan diawali karakter tilde(~) atau karakter tak terhingga.

Destruktor digunakan secara khusus manakala suatu objek menggunakan memori dinamis selama keberadaannya dan kemudian melepaskan memori itu setelah tidak menggunakannya lagi.

Contoh-4

```
//Destruktor
#include <conio.h>
#include <iostream.h>
#include <string.h>
class motor
{
private :
char *merk;
float cc_mesin; long harga;

public :
motor(char *nama, float cc, long hrg);//konstruktor
~motor();//destruktor
void keterangan( );
};

void main( )
{
clrscr( );
motor sport("Honda CBR",500,30500000);
motor matic("Honda Vario",125,14500000);
sport.keterangan( );
matic.keterangan( );
getch( );
}

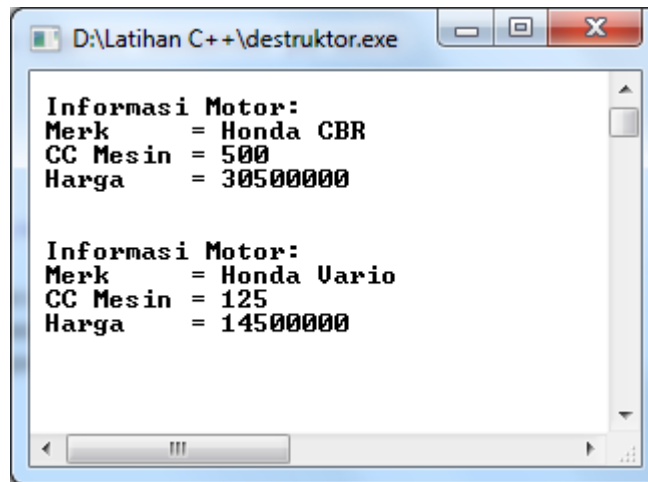
motor :: motor(char *nama, float cc, long hrg)
{
merk = new char[25];//merubah data karakter menjadi string
strcpy(merk,nama);
cc_mesin=cc;
harga=hrg;
}

motor :: ~motor( )
{
delete [ ] merk;//menghapus memori karakter pd merk
}
void motor :: keterangan( )
```

```
{
    cout<<"\n Informasi Motor:"<<endl

    <<" Merk    = "<<merk<<endl
    <<" CC Mesin = "<<cc_mesin<<endl
    <<" Harga   = "<<harga<<endl<<endl;
}
```

Output yang dihasilkan dari program contoh-4 di atas adalah:



Gambar 11.5 Hasil Contoh-4

11.4. Array pada Kelas

Anggota dari suatu kelas dapat pula berupa array. Berikut ini contoh anggota dari suatu kelas yang berupa array.

Contoh-5

```
//Class dg Array & Fungsi
#include <conio.h>
#include <iostream.h>

class kerja
{
    public :
    char nik[10],nm[25];
    int jam; double total;
    float lembur(int l);
};

main( )
{
    int x,y,a=1,b=1;
    double grand=0;
    kerja kary[15];
    cout<<endl<<"Jumlah Data :";cin>>y;

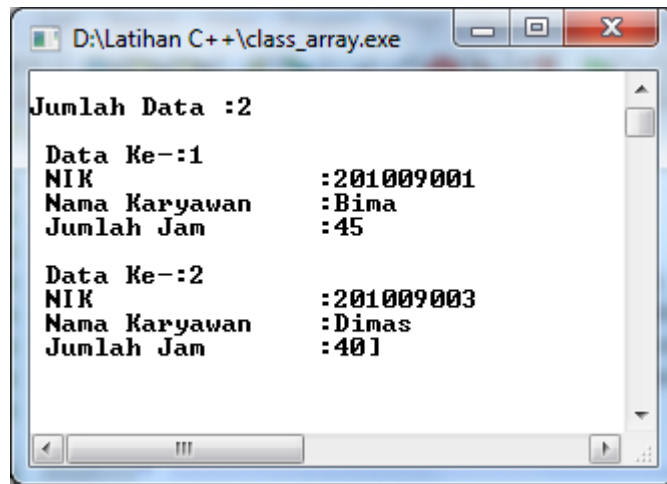
    for(x=0;x<y;x++)
    {
```

```

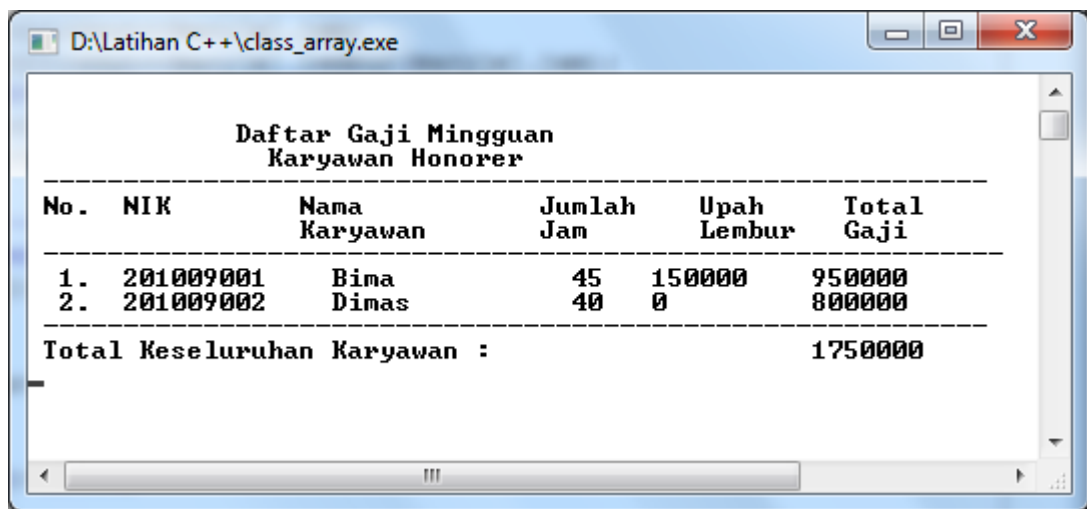
cout<<endl<<" Data Ke-:"<<a++<<endl;
cout<<" NIK      :";cin>>kary[x].nik;
cout<<" Nama Karyawan  :";cin>>kary[x].nm;
cout<<" Jumlah Jam    :";cin>>kary[x].jam;
}
clrscr();
gotoxy(14,3);cout<<"Daftar Gaji Mingguan\n";
gotoxy(16,4);cout<<"Karyawan Honorer\n";
cout<<" -----\n"
<<" No. NIK   Nama      Jumlah Upah  Total\n"
<<"          Karyawan  Jam    Lembur Gaji\n"
<<" -----\n";
for(x=0;x<y;x++)
{
gotoxy(3,wherey());cout<<b++<<".";
gotoxy(7,wherey());cout<<kary[x].nik;
gotoxy(15,wherey());cout<<kary[x].nm;
gotoxy(30,wherey());cout<<kary[x].jam;
gotoxy(36,wherey());cout<<kary[x].lembur(kary[x].jam);
kary[x].total=(40*20000)+kary[x].lembur(kary[x].jam);
gotoxy(44,wherey());cout<<kary[x].total<<endl;
grand=grand+kary[x].total;
}
cout<<" -----\n";
cout<<" Total Keseluruhan Karyawan :";
gotoxy(44,wherey());cout<<grand<<endl;
getch();
}
float kerja::lembur(int l)
{
if (l>40)
return(l-40)*30000;
else
return 0;
}

```

Output yang dihasilkan dari program contoh-5 di atas adalah:



Gambar 11.6 Hasil Contoh-5 (Layar Input)



Gambar 11.7 Hasil Contoh-5 (Layar Output)

11.5. Karakteristik OOP

Di dalam penggunaan konsep pemrograman yang berbasis objek atau yang disebut Object Oriented Pemrograman (OOP), haruslah memiliki karakteristik. Adapun karakteristik tersebut adalah memiliki sifat turunan atau pewarisan (*Inheritance*), satu nama memiliki banyak sifat atau perilaku (*Polymorphism*), pembungkusan sifat dari objek yang berbeda (*Encapsulation*). Berikut akan dijelaskan karakteristik OOP tersebut:

11.5.1. Inheritance

Inheritance memungkinkan programmer untuk "menurunkan" sebuah class menjadi class lain yang lebih spesifik dan memiliki fungsi yang lebih komplit. Inheritance merepresentasikan hubungan "seperti" atau "sejenis" (*a kind of*). Sebagai contoh, sebuah perahu motor adalah seperti perahu namun dengan kemampuan lebih, yakni memiliki motor. Contoh lain adalah kendaraan jenis mobil (sebagai super class) dan memiliki tipe sport (sebagai subclass), bila digabung menjadi mobil sport.

Ketika kita menurunkan sebuah class dari class yang lain, class yang baru akan mewarisi semua attribute dan method dari class yang sudah ada. Class yang sudah ada disebut dengan **base class** atau **super class** atau **parent class** dan class

yang baru disebut dengan **derived class** atau **subclass**, atau **child class**. Dengan inheritance, kita dapat men daur ulang program kita atau bahkan men daur ulang program orang lain agar sesuai dengan kebutuhan kita.

Tanpa inheritance, kelas merupakan sebuah unit yang berdiri sendiri. Inheritance akan membentuk suatu konsep dimana jika konsep yang diatas berubah maka perubahan akan juga berlaku pada yang ada dibawahnya. Inherate sangat mirip dengan hubungan orang tua dengan anak. Manakala suatu kelas menerima warisan, semua anggota data dan fungsi juga akan menerima warisan, walalupun tidak semuanya akan dapat di akses oleh anggota fungsi dari kelas.

Di dalam C++ penentuan akses pada inheritance ada tiga macam, yaitu :

1. Public
Penentuan akses berbasis public menyebabkan anggota dari public dari sebuah kelas utama akan menjadi anggota public kelas turunan dan menyebabkan juga anggota protect kelas utama menjadi anggota protect kelas turunan, namun untuk anggota kelas private tetap pada private kelas utama.
2. Private
Penentu akses berbasis private menyebabkan anggota dari anggota public dari kelas utama akan menjadi anggota protect kelas turunan, dan menyebabkan anggota dari kelas utama menjadi anggota protect kelas turunan, namun untuk anggota kelas private tetap pada private kelas utama.
3. Protected
Penentu akses berbasis protect menyebabkan anggota dari anggota protect dan public dari kelas utama akan menjadi anggota private dari kelas turunan. Anggota private dari kelas utama selalu menjadi anggota private kelas utama.

Pada waktu mendeklarasikan suatu kelas, anda dapat menandai bahwa suatu kelas berasal dari mana, yaitu dengan tanda titik dua (:) setelah nama kelas, tipe asalnya bias berupa public atau yang lainnya dan dari kelas mana berasal. Berikut contoh penulisan sintaksisnya:

```
Contoh-6      #include <iostream>
                #include <conio>

                class Values
                {
                protected: // dapat di gunakan di class turunannya nanti
                int P, L;

                public:

                void Nilai(int a, int b)
                { P=a;  L=b; }
                };

                // class ini turunan dari class values
                class luasPersegiPanjang: public Values
```

```

{
    public:
    int luas()
    { return (P*L); }
};

// class ini turunan dari class values
class kelilingPersegiPanjang: public Values
{
    public:
    int keliling()
    {
        return((P +L) * 2);
    }
};

int main ()
{
    luasPersegiPanjang a; //pembuatan objek luasPersegiPanjang

    //pembuatan objek kelilingPersegiPanjang
    kelilingPersegiPanjang b;
    a.Nilai(4,5); // input nilai ke dalam method Nilai
    b.Nilai(4,5);
    cout <<"Luas Persegi Panjang = "<<a.luas()<<endl;
    cout <<"Keliling Persegi Panjang = "<<b.keliling()<<endl;
    getch();
}

```

Output yang dihasilkan dari program contoh-6 di atas adalah:

```

Luas Persegi Panjang = 20
Keliling Persegi Panjang = 18

```

Gambar 11.7 Hasil Contoh-6

11.5.2. Polymorphism

Polymorphisms adalah kemampuan 2 buah object yang berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Contoh, saya melatih lumba-lumba saya dengan perintah untuk meloncat dan juga saya melatih burung untuk merespon perintah saya untuk berkicau. Saya lakukan latihan untuk merespon kepada mereka dengan perintah lisan. Melalui polymorphism saya tahu bahwa anjing dan burung akan merespon dengan gonggongan atau kicauan.

Dengan cara dan kemampuan seperti ini, jenis akses tertentu dapat diterapkan dengan berbagai cara oleh objek-objek yang memiliki tipe yang berbeda atau memiliki banyak bentuk. Fungsi virtual merupakan dasar dari *polymorphism*

yang berkerja hanya dengan pointer-pointer dan referensi dan hanya mendeklarasikan method sebagai virtual.

Adapun aturan dari virtual function sebagai berikut:

1. Virtual function harus anggota class.
2. Anggota class bukan anggota yang bersifat statis.
3. Anggota class dapat diakses dengan pointer objek.
4. Virtual function tidak dapat memiliki virtual constructor, akan tetapi dapat berupa virtual destructor.

Contoh-7

```
//Penggunaan Polymorphism
#include <iostream.h>
#include <conio.h>

class HewanPeliharaan
{
public:
    void lucu( )
    {
        cout<<" Lucunya hewan peliharaan"<<endl;
    }
    virtual void makan( ) // konsep polymorphisme
    {
        cout<<" Makan... dibutuhkan hewan peliharaan"<<endl;
    }
};

class Jinak : public HewanPeliharaan
{
public:
    void lucu( )
    {
        cout<<" Lucu dan Jinak"<<endl;
    }
    virtual void makan( ) // konsep polymorphisme
    {
        cout<<" Diberi makan agar jinak"<<endl;
    }
};

class Kucing : public Jinak
{
public:
    void lucu( )
    {
        cout<<" Lucunya kucing"<<endl;
    }
    virtual void makan( ) // konsep polymorphisme
    {
        cout<<" Makanan dibutuhkan Kucing"<<endl;
    }
};
```

```
};

void main( )
{
    //definisi Objek Jinak dan Kucing puma

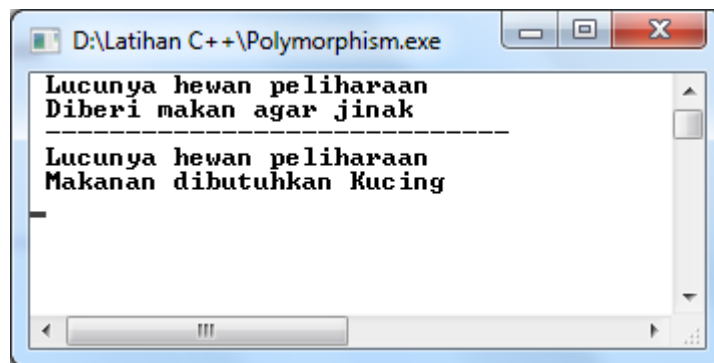
    Jinak jindak;
    Kucing puma;

    //definisi pointer ke objek
    HewanPeliharaan *hewan;

    hewan=&jindak;
    hewan->lucu( );
    hewan->makan( );
    cout<<" -----" <<endl;

    hewan = &puma;
    hewan->lucu( );
    hewan->makan( );
    getch( );
}
```

Output yang dihasilkan dari program contoh-7 di atas adalah:



Gambar 11.8 Hasil Contoh-7

11.5.3. Encapsulation

Ciri penting lainnya dari OOP adalah *encapsulation*. Encapsulation adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan *hidden*. Jika ingin mendapat data, maka harus berinteraksi dengan objek yang bertanggung jawab atas data tersebut. Berikut cirri dari encapsulation:

1. Variabel dan method dalam suatu obyek dibungkus agar terlindungi
2. Untuk mengakses, variabel dan method yang sudah dibungkus tadi perlu interface
3. Setelah variabel dan method dibungkus, hak akses terhadapnya dapat ditentukan.
4. Konsep pembungkusan ini pada dasarnya merupakan perluasan dari tipe data struktur

Dua hal dalam enkapsulasi :

1. Information hiding
2. Menyediakan perantara (method) untuk mengakses data

Pada intinya, encapsulation adalah pemisahan antara bagian private dan public pada sebuah objek. Atau, dapat dipandang sebagai pemisahan antara interface (bagian private) dan implementation (bagian public).

Objek-objek lain yang hendak berinteraksi dengan objek ini akan mengirimkan sebuah pesan (message) dan objek ini akan mengerjakan sesuatu dan mengirimkan pesan balik sebagai jawaban jika diperlukan.

Keuntungan utama dari encapsulation tentu saja adalah penyembunyian implementasi (implementation hiding). Dengan implementation hiding, kita dapat memperbaiki bagaimana objek kita bekerja tanpa harus khawatir bagaimana menginformasikan perubahan tersebut ke objek-objek yang lain. Selama kita tidak merubah interface dari objek kita, objek-objek yang lain akan tetap dapat menggunakan objek kita.

```
Contoh-8 //penggunaan encapsulation
#include <conio.h>
#include <iostream.h>

class satu_kandang
{
    int kandang;
public:
    void atur(int nilai);

    int hasil(void);
};

void satu_kandang :: atur(int nilai)
{
    kandang = nilai;
}

int satu_kandang :: hasil(void)
{
    return kandang;
}

void main( )
{
    satu_kandang ayam1, ayam2, ayam3;
    int bebek;
    ayam1.atur(5);
    ayam2.atur(7);
    ayam3.atur(2);
    bebek=20;

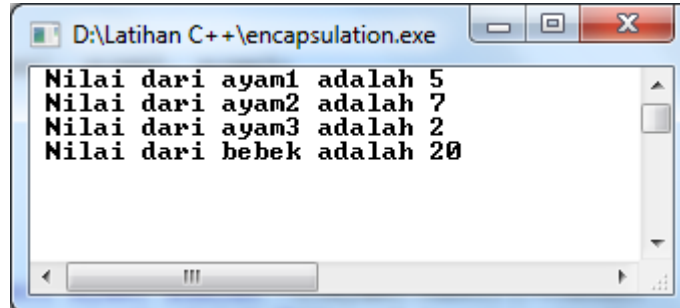
    cout<<" Nilai dari ayam1 adalah "<<ayam1.hasil( )<<endl;
    cout<<" Nilai dari ayam2 adalah "<<ayam2.hasil( )<<endl;
```

```

        cout<<" Nilai dari ayam3 adalah "<< ayam3.hasil()<<endl;
        cout<<" Nilai dari bebek adalah "<<bebek<<endl;
        getch();
    }

```

Output yang dihasilkan dari program contoh-8 di atas adalah:



Gambar 11.9 Hasil Contoh-8

Selain contoh diatas, penggunaan encapsulation dapat pula dengan membedakan fungsi method. Ada deklarasi method untuk input nilai dan ada pula yang di gunakan untuk output nilai. Contoh penulisan listingnya sebagai berikut :

```

Contoh 9 #include <iostream>
            #include <conio>

            class enkapsulasi
            {
            public :
                char *Nama;
                char *NIM;
                int nilaiku;

                void setNilai(int nilai){ // method yang digunakan untuk input
                    nilaiku=nilai;
                }

                int getNilai( ){ // method yang digunakan untuk output
                    return nilaiku;
                }
            };

            int main( )
            {
                enkapsulasi data; // pembuatan objek class

                //input datanya
                data.Nama = "Bima Chanindra";
                data.NIM = "12120001";
                data.setNilai(100);
                //output datanya
                cout << "Nama = " << data.Nama << endl;
                cout << "NIM = " << data.NIM << endl;
                cout << "Nilai = " << data.getNilai( );
            }

```

```
    getch();
}
```

Output yang dihasilkan dari program contoh-9 di atas adalah:

```
Nama = Bima Chanindra
NIM  = 12120001
Nilai = 100
```

Gambar 11.10 Hasil Contoh-9

11.6 Latihan

Seorang manager ingin dibuatkan sebuah program perhitungan gaji pegawai staff honor sederhana. Anda sebagai staff programmer harus membuat program tersebut.

Adapun ketentuan perhitungan gaji pegawai honorer adalah sebagai berikut : Honorer per hari yang diterima pegawai honorer sebesar Rp. 25.000. Jam kerja selama 8 Jam setiap harinya

Jam bekerja lebih dari 8 jam, maka kelebihan jam dikalikan dengan honor lembur perjam sebesar Rp. 1.500.

Petunjuk : Gunakan kelas **pegawai**.

Desain masukan yang diinginkan adalah sebagai berikut

PT. Meriang Gembira

Jumlah :.....
 Tanggal Input :.....

 Data Ke-1

Nama Pegawai :.....
 Jumlah Jam Kerja :.....Jam

Data Ke-2

Nama Pegawai :.....
 Jumlah Jam Kerja :.....Jam

<terus mengulang sesuai jumlah pegawai yang diinputkan>

Desain keluaran yang diinginkan adalah sebagai berikut :

PT. Meriang Gembira

Tanggal :

No.	Nama Pegawai	Honor	Jumlah Jam Kerja	Honor Lembur	Total Honor
...
...
Total Honor Pegawai Sebesar				

Struktur, Enum, Union, Bit-Field dan Typedef



12.1 Struktur

Struktur bermanfaat untuk mengelompokkan sejumlah data dengan tipe yang berlainan. Apabila suatu struktur telah dideklarasikan, struktur ini dapat digunakan untuk mendefinisikan suatu variabel.

Suatu struktur juga dapat mengandung struktur yang lain dan anggota struktur dapat diakses menggunakan bentuk :

variable_struktur.nama_anggota

Contoh program lengkap yang melibatkan pendeklarasian dan pendefinisian variabel struktur dan juga pengaksesan terhadap anggota variabel struktur dapat dilihat dibawah ini :

```

/*-----*
/* Contoh 1 : Pendeklarasian, pendefinisian dan *
/*           pengaksesan struktur           *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    struct data_tanggal           // Pendeklarasian
    {
        int tahun;
        int bulan;
        int tanggal;
    };
    data_tanggal tanggal_lahir // Pendefinisian struktur
    // Pengaksesan anggota struktur
    tanggal_lahir.tanggal = 28;
    tanggal_lahir.bulan = 11;
    tanggal_lahir.tahun = 1982;
    cout << tanggal_lahir.tanggal << ' / '
         << tanggal_lahir.bulan << ' / '
         << tanggal_lahir.tahun << endl;
}

```

Hasil eksekusi :

28/11/1982

Pada program diatas tanda titik diantara nama variabel dan nama anggota menyatakan penugasan untuk memberikan nilai 28 ke anggota *tanggal* pada variabel struktur *tanggal_lahir*.

Pemberian nilai terhadap suatu struktur dapat dilakukan dengan bentuk :

```
var1 = var2;
```

sepanjang kedua variabel adalah variabel struktur bertipe sama.

Contoh program :

```
/*-----*
/* Contoh 2 : Penugasan struktur      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    struct data_tanggal          // Pendeklarasian
    {
        int tahun;
        int bulan;
        int tanggal;
    };

    data_tanggal tgl1, tgl2 ;    // Pendefinisian struktur
    // Penugasan per anggota
    tgl1.tanggal = 28;
    tgl1.bulan = 11;
    tgl1.tahun = 1982;
    // Penugasan antaranggota struktur
    tgl1 = tgl2;
    cout << tgl2.tanggal << ' / '
         << tgl2.bulan << ' / '
         << tgl2.tahun << endl;
}
}
```

Hasil eksekusi :

28/11/1982

12.2 Union

Union menyerupai struktur (termasuk dalam hal pengaksesannya), namun mempunyai perbedaan nyata. Union biasa dipakai untuk menyatakan suatu memori dengan nama lebih dari satu.

Contoh program :

```

/*-----*
/* Contoh 1 : Pendefinisian union      *
/*-----*
#include <iostream.h>
#include <conio.h>
union bil_bulat
{
unsigned int di;
unsigned char dc[2];
};

void main()
{
clrscr();
bil_bulat bil_x;    // Pendefinisian union

bil_x.di = 0x2345;
cout << setiosflags(ios::showbase);
cout << hex << "di    : " << bil_x.di << endl;
cout << hex << "dc[0] : " << int(bil_x.dc[0]) << endl;
cout <<          "dc[1] : " << int(bil_x.dc[1]) << endl;
}

```

Hasil eksekusi :

<pre> Di : 0x2345 dc[0] : 0x45 dc[1] : 0x23 </pre>

Tampak bahwa dengan mengisikan nilai *bil_x.di*, data dapat diakses melalui *bil_x.dc*. dalam hal ini, *bil_x.dc[0]* menyimpan byte rendah dari *bil_x.di* dan *bil_x.dc[1]* berkaitan dengan byte tinggi dari *bil_x.di* (mengingat *bil_x.di* berukuran dua byte).

Seperti halnya struktur, variabel union juga dapat diinisialisasi saat didefinisikan.

Contoh program :

```

/*-----*
/* Contoh 2 : Inisialisasi Union      *
/*-----*
#include <iostream.h>
#include <conio.h>

```

```

union bil_bulat
{
unsigned int di;
unsigned char dc[2];
};

void main()
{
clrscr();
bil_bulat bil_x = 0x2345; // Inisialisasi

cout << setiosflags(ios::showbase);
cout << hex << "di : " << bil_x.di << endl;
cout << hex << "dc[0] : " << int(bil_x.dc[0]) << endl;
cout << "dc[1] : " << int(bil_x.dc[1]) << endl;
}

```

12.3 Struktur Bit-field

Satu bit atau beberapa bit dalam sebuah data berukuran suatu byte atau dua byte dapat diakses dengan mudah melalui *bit-field*. Dengan cara ini suatu bit atau beberapa bit dapat diakses tanpa melibatkan operator manipulasi bit (seperti & dan ||). Selain itu satu atau dua byte meri dapat dipakai untuk menyimpan sejumlah informasi.

Conto program :

```

/*-----*
/* Contoh 1 : bit-filed untuk mengakses bit-bit *
/*          dalam sebuah byte data          *
/*-----*
#include <iostream.h>
#include <conio.h>
// Pendeklarasian bit-field
struct info_bit
{
unsigned bit0 : 1;
unsigned bit1 : 1;
unsigned bit2 : 1;
unsigned bit3 : 1;
unsigned bit4 : 1;
unsigned bit5 : 1;
unsigned bit6 : 1;
unsigned bit7 : 1;
};
void main()
{
clrscr();
union ubyte // Pendeklarasian union
{
unsigned char byte;
info_bit bit;
}
}

```

```
};
ubyte ascii;          // Pendeklarian variabel union
int nilai;
cout << " Masukkan ascii antara 0 s/d 255 : ";
cout << ascii.bit.bit7 << ascii.bit.bit6
cout << ascii.bit.bit5 << ascii.bit.bit4
cout << ascii.bit.bit3 << ascii.bit.bit2
    << ascii.bit.bit1 << ascii.bit.bit0 << endl;
}
```

Hasil eksekusi :

```
Masukkan ascii antara 0 s/d 255 : 128 ↵
76543210 ← Posisi bit
10000000
```

```
Masukkan ascii antara 0 s/d 255 : 16 ↵
76543210 ← Posisi bit
00010000
```

Perlu diketahui, suatu variabel yang didefinisikan sebagai *bit-field* tidak bisa diisi secara langsung dengan suatu nilai. Oleh karena itu biasa dibentuk didalam union.

12.4 Enum

Tipe enum biasa dipakai kalau kemungkinan nilai dari suatu data telah diketahui, dan jumlah kemungkinannya tidak banyak.

Contoh program :

```
/*-----*
/* Contoh 9.6 : tipe enum          *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr();
enum nama_hari { Senin, Selasa, Rabu, Kamis, Jumat,
                Sabtu, Minggu };
nama_hari hari1, hari2; // Pendefinisian enum
// Pemberian nilai enum
hari1 = Senin;
hari2 = Jumat;
int selisih = hari2 - hari1;
cout << "Selisih Hari : " << selisih << endl;
}
```

Hasil eksekusi

```
Selisih Hari : 4
```

Pada contoh diatas terdapat pendeklarasian tipe **enum** berupa *nama_hari*. Anggota tipe **enum** *nama_hari* berupa Senin, Selasa, Rabu dan seterusnya.

nma_hari hari1, hari2 merupakan pernyataan untuk mendefinisikan variabel bernama *hari1* dan *hari2* yang bertipe **enum** *nama_hari*.

12.5 Typedef

Typedef biasa dipakai untuk memberikan nama alias terhadap suatu tipe data. Sebagai contoh anda dapat memberikan nama alias dari unsigned char berupa BYTE.

Contoh program :

```
/*-----*
/* Contoh 9.7 : typedef          *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr();
typedef unsigned char BYTE;
BYTE kode;          // Sebagai pemendekan dari : unsigned char code;
kode = 65;
cout << kode; // Karakter ASCII 65
}
```

Hasil eksekusi :

A

Latihan :

Buat program untuk menentukan selisih hari kelahiran anda dengan teman anda. (digenapkan dalam bentuk tahun)

Contoh :

Anda lahir : 8/12/1978

Teman Lahir : 28/11/1982

Selisih Lahir (digenapkan) = 4 tahun

(simpan dengan nama Latihan12.cpp)

Rekursi

Fungsi dalam C++ dapat dipakai secara rekursi, artinya suatu fungsi dapat memanggil fungsi yang merupakan dirinya sendiri. Penerapan rekursi diantaranya untuk menghitung nilai :

$$X^n$$

Dengan n merupakan bilangan bulat positif. Solusi dari persoalan ini berupa :

$$\text{Jika } n = 1 \text{ maka } X^n = X$$

$$\text{Selain itu : } X^n = X * X^{n-1}$$

Contoh program :

```

/*-----*
/* Contoh   : Opreasi pangkat secara *
/*           rekursi                 *
/*-----*
# include <iostream.h>
#include <conio.h>
long int pangkat ( int x, int n);
void main()
{
int x, y;
clrscr();
cout << " Menghitung x ^ y "<< endl;
cout << " x = " ;
cin >> x ;
cout << " y = " ;
cin >> y ;
cout << x << " ^ " << y << endl;
    << pangkat(x, y) << endl;
}

long int pangkat(int x, int n)
{
if (n == 1 )
    return(x);
else
    return(x * pangkat(x, n - 1));
}

```

Hasil elsekusi :

<p>Menghitung x ^ y x = 2 ↵ y = 3 ↵ 2 ^ 3 = 8</p>

Tugas:

Buat program untuk memasukkan data disertai keterangan data ke-.

Contoh :

Banyak data = 3

Masukkan Nilai ke-1 : 28 ↵

Masukkan Nilai ke-2 : 11 ↵

Masukkan Nilai ke-3 : 1982 ↵

Data ke-1 = 28

Data ke-2 = 11

Data ke-3 = 1982

(simpan dengan nama tugasrekursi.cpp)

13.1 Rekursi dengan satu pemanggilan Rekursif

Suatu fungsi C++ memiliki karakteristik yang sangat menarik, yaitu dia bisa memanggil dirinya sendiri. (Tidak seperti C, C++ tidak mengizinkan main() memanggil dirinya sendiri). Kemampuan ini diistilahkan dengan rekursi. Rekursi adalah fitur penting dalam beberapa area pemograman, seperti kecerdasan buatan, tetapi kita hanya akan membahas secara sederhana bagaimana rekursi bekerja.

Suatu fungsi rekursif memanggil dirinya sendiri, dan fungsi yang di panggil memanggil dirinya juga, dan begitu seterusnya, sampai waktu yang tak terhingga kecuali program menyertakan kode untuk memberhentikan rantai pemanggilan tersebut. Metode yang biasa digunakan adalah membuat bagian pemanggilan rekursif memuat statemen if. Sebagai contoh, suatu fungsi rekursif bertipe void, rekurs(), memiliki format seperti ini :

```
void rekurs(daftar_argumen)
{
    statemen1
    if (kondisi)
        rekurs (argumen)
    statemen2
}
```

Dengan perhitungan atau peruntungan, kondisi berakhir dengan nilai false, dan rantai pemanggilan rekursif terputus. Pemanggilan rekursif menghasilkan suatu rantai kejadian yang menarik. Sepanjang statemen if bernilai true, setiap pemanggilan rekurs() akan mengeksekusi statemen1 dan memanggil kembali rekurs() untuk mengeksekusi statemen1, dan begitu seterusnya, tanpa pernah meraih statemen2. Ketika statemen if bernilai false, maka pemanggilan rekurs() kali ini akan mengeksekusi statemen2.

Kemudian ketika pemanggilan fungsi tersebut berakhir, kendali program akan kembali ke fungsi rekurs() yang memanggilnya. Selanjutnya, ketika versi fungsi rekurs() tersebut selesai mengeksekusi statemen2 dan berhenti, kendali program akan kembali memanggil fungsi rekurs(), dan begitu seterusnya. Jadi, jika fungsi rekurs() mengalami lima kali pemanggilan rekursif, maka statemen1 dieksekusi lima kali terlebih dahulu dengan urutan yang sama dengan pemanggilan fungsi rekursif tersebut, dan statemen2 dieksekusi lima kali dengan urutan yang terbalik.

Contoh Program rekursi1:

```
#include <iostream>
void hitungmundur(int n);
int main()
{
    hitungmundur(4); //panggil fungsi rekursif
    return 0;
}
void hitungmundur(int n)
{
    using namespace std;
    cout<<"Hitung Mundur..."<<n<<"...."<<"Level"<< 5-n <<endl;
    if (n>0)
        hitungmundur(n-1); //fungsi memanggil dirinya sendiri
    cout<<n<<":BaaH"<<"....."<<"Level"<< 5-n <<endl;
}
```

13.2 Rekursi dengan pemanggilan Rekursif jamak

Rekursi secara khusus berguna pada situasi dimana terdapat tugas untuk membagi sesuatu secara berulang menjadi dua bagian yang sama. Sebagai contoh, perhatikan pendekatan ini untuk menggambar suatu penggaris. Tandai kedua ujung dan cari titik tengahnya. Kemudian, lakukan prosedur tersebut pada bagian sebelah kiri penggaris dan kemudian pada sebelah kanan. Jika anda menginginkan pembagian lagi, terapkan prosedur itu pada subbagian-subbagian yang ada .

Contoh program 1:

```
#include <iostream>
const int pjg=66;
const int bagian=6;
void subbagi(char ar[],int rendah, int tinggi, int level);
int main()
{
    char penggaris[pjg];
    int i;
    for (i=1; i<pjg-2;i++)
        penggaris[i]="";
    penggaris [pjg - 1]='\0';
    int maks=pjg-2;
    int min=0;
    penggaris[min]=penggaris[maks]="|";
    std::cout<<penggaris<<std::endl;
    for (i=1;i<=bagian;i++)
    {
        subbagi(penggaris,min,maks,i);
        std::cout<<penggaris<<"level ke-"<<i<<std::endl;
    }
    for (int j=1;j<pjg-2;j++)
```

```
    penggaris[j]=""; //reset penggaris kosong
    }
    return 0;
    }
    void subbagi(char ar[], int rendah,int tinggi,int level)
    {
    if(level==0)
    return;
    int tengah=(tinggi+rendah)/2;
    ar[tengah]='|';
    subbagi(ar,rendah,tengah,level-1);
    subbagi(ar,tengah,tinggi,level-1);
    }
```

Tugas :

Carilah beberapa contoh program rekursif melalui internet atau buku dan screeshoot hasilnya.



Review Materi

Dosen/Instruktur diharapkan Mereview kembali materi yang telah disampaikan pada pertemuan 1-13 dan mengecek hasil latihan pemograman C++ mahasiswa pada beberapa contoh program dipertemuan 1-13 dan memberkan studi kasus yang dapat dikerjakan oleh mahasiswa

Studi kasus

Buatlah sebuah program aplikasi bertemakan bisnis secara berkelompok. Adapun isi program yang dibuat harus memuat materi UAS : Array, kondisi, looping, label, fungsi, structure dan Class. Lalu buatlah bentuk rancangannya **seperti contoh berikut:**

Bentuk Menu Utama

```

PENYEWAAN PAKAIAN ADAT NASIONAL
CHONIO BOUTIQUE
*****
Selamat Datang Di Chonio Boutique

Pilihan Menu :
1. Input Data
2. Log Out
=====
Inputkan Pilihan Anda : 1
    
```

Bentuk Input

```

PENYEWAAN PAKAIAN ADAT NASIONAL
CHONIO BOUTIQUE
*****
NAMA PENYEWA      : Bisma
INPUT LAMA SEWA   : 2
INPUT JUMLAH DATA : 3

*****

DATA KE - 1
INPUT KODE PAKET BAJU [JB/JT/SB] : JB
INPUT KODE UKURAN BAJU [S/M/L]   : S
JUMLAH SEWA      : 1

DATA KE - 2
INPUT KODE PAKET BAJU [JB/JT/SB] : JT
INPUT KODE UKURAN BAJU [S/M/L]   : M
JUMLAH SEWA      : 2

DATA KE - 3
INPUT KODE PAKET BAJU [JB/JT/SB] : SB
INPUT KODE UKURAN BAJU [S/M/L]   : L
JUMLAH SEWA      : 1
    
```

Bentuk Output

```

                                PENYEWAAN PAKAIAN ADAT NASIONAL
                                CHONIO BOUTIQUE
                                * * * * *

NAMA PENYEWA   : Bisma

DATA BAJU YANG DISEWA
*****
No.   Nama Paket      Harga      Jumlah Sewa   Subtotal
*****
1     JAWA BARAT      200000     1             200000
2     JAWA TENGAH    255000     2             510000
3     SUMATERA BARAT 300000     1             300000
*****

                                TOTAL BAYAR   : Rp. 1010000
                                UANG BAYAR    : Rp. 1100000
                                UANG KEMBALI  : Rp. 90000

                                TERIMA KASIH
INPUT DATA LAGI [Y /T ] : Y

```

Presentasi UAS Project



Mahasiswa melakukan presentasi projek sesuai dengan project masing-masing sebagai nilai ujian akhir semester,

Presentasi UAS Project



Mahasiswa melakukan presentasi projek sesuai dengan project masing-masing sebagai nilai ujian akhir semester,