

Borland C++

Pemrograman C++ dengan Borland C++ 5.02 (Edisi Revisi)

Pembahasan

- Pengenalan Bahasa C++
- Pengenalan Model Data, Perintah Masukan dan Keluaran
- Operator Borland C++
- Operasi Penyeleksian Kondisi
- Proses Perulangan
- Operasi String
- Array
- Pointer
- Fungsi
- Macro
- Structure

Borland C++ 5.02

```
#include<iostream.h>
#include<conif.h>
main()
{
```

```
int a, b
flo
```



BORLAND C++

Kata Pengantar

Puji dan syukur kehadiran Allah S.W.T, yang telah memberikan rahmat dan hidayah yang sangat besar kepada penulis untuk menyelesaikan Edisi Revisi Buku Pemrogramman C++ dengan menggunakan Borland C++ 5.02.

Buku Edisi Revisi Pemrogramman C++ dengan Borland C++ 5.02 dibuat agar siswa/i SMK Negeri 1 Bondowoso, khususnya kompetensi keahlian Rekayasa Perangkat Lunak dapat mempelajari, mencoba dan melatih Pemrograman C++ dengan Borland C++ 5.02 secara cepat dan tepat sesuai dengan kebutuhan.

Di dalam Buku Edisi Revisi Pemrogramman C++ dengan Borland C++ 5.02 latihan-latihan yang tersedia dapat juga dicoba pada Borland Turbo C++. Disini tidak membahas beberapa hal yang mendasar sehingga dapat mempelajari hal-hal lainnya sendiri yang tidak dibahas dalam buku ini.

Semoga buku ini bermanfaat bagi siswa/i yang mau belajar.

Bondowoso, 05 Februari 2014

Tim Penulis

BORLAND C++**Daftar Isi**

Kata Pengantar	i
Daftar Isi	ii

Bab 1 Pengenalan Bahasa C++

1.1. Sekilas Perkembangan Bahasa C.....	1
1.2. Pengenalan IDE Borland C++	1
1.3. Membuat File Editor	3
1.4. Menyimpan File Editor.....	4
1.5. Menterjemahan Program	5
1.6. Menjalankan Program	5
1.7. Membuka File Editor	6
1.8. Mencetak File Editor.....	7
1.9. Keluar dari Borland C++ 5.02	7
1.10 Struktur Program C++	8

Bab 2 Pengenalan Model Data, Perintah Masukan dan Keluaran

2.1. Pengenalan Tipe Data	11
2.2. Konstanta	12
2.2.1. Konstanta Bilangan	12
2.2.2. Konstanta Teks	12
2.2.3. Deklarasi Konstanta	12
2.3. Variabel	13
2.3.1. Variabel Numerik	13
2.3.2. Variabel Teks	13
2.3.2. Deklarasi Variabel	13
2.4. Perintah Keluaran	14
2.4.1. printf()	14
2.4.2. puts()	18
2.4.3. putchar()	19
2.4.4. cout()	19
2.4.5. Fungsi Manipulator	20
2.5. Perintah Masukan	32
2.5.1. scanf()	32
2.5.2. gets()	33
2.5.3. cin()	34
2.5.4. getch()	35
2.5.5. getch()	36
2.6. Latihan	36

Bab 3 Operator Borland C++

3.1. Operator Aritmatika.....	39
3.1.1. Ekspresi Aritmatika.....	40
3.1.2. Hierarki Operator Aritmatika	41
3.2. Operator Pemberi Nilai	42
3.3. Operator Penambah dan Pengurang.....	43
3.4. Operator Relasi.....	45
3.5. Operator Logika	46
3.5.1. Operator Logika AND	46
3.5.2. Operator Logika OR	48
3.5.3. Operator Logika NOT	50
3.6. Operator Bitwise	51
3.6.1. Operator Bitwise << (Shift Left)	51
3.6.2. Operator Bitwise >> (Shift Right)	52
3.6.3. Operator Bitwise & (And).....	53
3.6.4. Operator Bitwise (Or).....	54
3.6.5. Operator Bitwise ^ (eXclusive Or).....	55
3.6.6. Operator Bitwise ~ (Not)	56
3.7. Latihan	57

Bab 4 Operasi Penyeleksian Kondisi

4.1. Pernyataan IF	59
4.1.1. Pernyataan IF - ELSE.....	61
4.1.2. Pernyataan Nested IF.....	62
4.1.2. Pernyataan IF – ELSE Majemuk	64
4.2. Pernyataan Switch - Case	66
4.3. Operator ?:.....	68
4.4. Latihan	70

Bab 5 Proses Perulangan

5.1. Pernyataan For	75
5.1.1. Pernyataan Nested For	79
5.1.2. Perulangan Tak Berhingga	80
5.2. Pernyataan goto	81
5.3. Pernyataan while	82
5.4. Pernyataan do - while	84
5.5. Pernyataan break	85
5.6. Pernyataan continue.....	86
5.7. Latihan	87

Bab 6 Operasi String

6.1. Struktur Fungsi	89
6.1.1. Fungsi strcat().....	89
6.1.2. Fungsi strcmp().....	90
6.1.3. Fungsi strcpy().....	91
6.1.4. Fungsi strlen().....	91
6.1.5. Fungsi strrev()	92
6.2. Prototipe Konversi String	93
6.2.1. Fungsi atof()	93
6.2.2. Fungsi atoi()	94

6.2.3. Fungsi atoi()	95
6.2.4. Fungsi strtolr()	95
6.2.5. Fungsi strupr()	96
6.3. Latihan	97

Bab 7 **Array**

7.1. Array Berdimensi Satu	99
7.1.1. Mengakses Array Berdimensi Satu	100
7.1.2. Inisialisasi Array Berdimensi Satu	102
7.2. Array Berdimensi Dua	103
7.2.1. Mengakses Array Berdimensi Dua	104
7.2.2. Inisialisasi Array Berdimensi Dua	105
7.3. Array Berdimensi Tiga.....	103
7.2.1. Mengakses Array Berdimensi Tiga	107
7.2.2. Inisialisasi Array Berdimensi Tiga	110
7.4. Latihan	112

Bab 8 **Pointer**

8.1. Operator Pointer	115
8.1.1. Operator Dereference (&)	115
8.1.2. Operator Dereference (*)	116
8.2. Deklarasi Pointer Pada Konstanta	117
8.3. Deklarasi Pointer Pada Variabel	118
8.4. Pointer pada Pointer	120
8.5. Pointer Pada Array	122
8.6. Pointer String	123

Bab 9 **Fungsi**

9.1. Struktur Fungsi	125
9.2. Prototipe Fungsi	126
9.3. Parameter Fungsi	127
9.3.1. Pemanggilan dengan Nilai (<i>Call by Value</i>)	128
9.3.2. Pemanggilan dengan Referensi (<i>Call by Reference</i>)	129
9.4. Pernyataan <i>return()</i>	130
9.5. Pengiriman Data Ke Fungsi	131
9.5.1. Pengiriman Data Konstanta Ke Fungsi	131
9.5.2. Pengirimam Data Variabel Ke Fungsi	132
9.6. Jenis Variabel	133
9.6.1. Variabel Lokal	133
9.6.2. Variabel Eksternal	134
9.6.2. Variabel Statis	136
9.7. Fungsi Inline	137
9.8. Fungsi Overloading	138
9.9. Latihan	140

Bab 10 **Macro**

10.1. Preprocessor Directive	143
10.1.1. #define	143
10.1.2. #include	146

10.1.3. #if - #endif	147
10.1.4. #if - #else - #endif	148
10.1.5. #i elif	148
10.1.6. #undef.....	149
10.1.7. #ifdef - #ifndef	150
10.2. Pembuatan File Header.....	152
10.3. Latihan.....	153

Bab 11 Structure

11.1. Deklarasi Structure	155
11.2. Nested Structure.....	157
11.3. Structure dengan Array.....	158
11.4. Structure dengan Function.	159
11.5. Tugas.....	161

Pemrogramman C++

Bab 1 : Pengenalan Bahasa C++

1.1. Sekilas Perkembangan Bahasa C

Penjelasan

Bahasa C merupakan pengembangan dari bahasa B yang ditulis oleh Ken Thompson pada tahun 1970. Bahasa C untuk pertama kali ditulis oleh Brian W. Kernighan dan Denies M. Ritchie pada tahun 1972. Bahasa C, pada awalnya dioperasikan diatas sistem operasi UNIX.

Bahasa C adalah merupakan bahasa pemrograman tingkat menengah yaitu diantara bahasa tingkat rendah dan tingkat tinggi yang biasa disebut dengan **Bahasa Tingkat Tinggi dengan Perintah Assambly**. Bahasa C mempunyai banyak kemampuan yang sering digunakan diantaranya kemampuan untuk membuat perangkat lunak, misalnya dBASE, Word Star dan lain-lain. Pada tahun 1980 seorang ahli yang bernama Bjarne Stroustrup mengembangkan beberapa hal dari bahasa C yang dinamakan "**C with Classes**" yang berganti nama pada tahun 1983 menjadi **C++**.

Penambahan yang terdapat pada C++ ini adalah Object Oriented Programming (OOP), yang mempunyai tujuan utamanya adalah membantu membuat dan mengelola program yang besar dan kompleks.

1.2. Pengenalan IDE Borland C++

Penjelasan

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

- Menulis Naskah Program.
- Mengkompilasi Program (**Compile**)
- Melakukan Pengujian Program (**Debugging**)
- Mengaitkan Object dan Library ke Program (**Linking**)
- Menjalankan Program (**Running**)

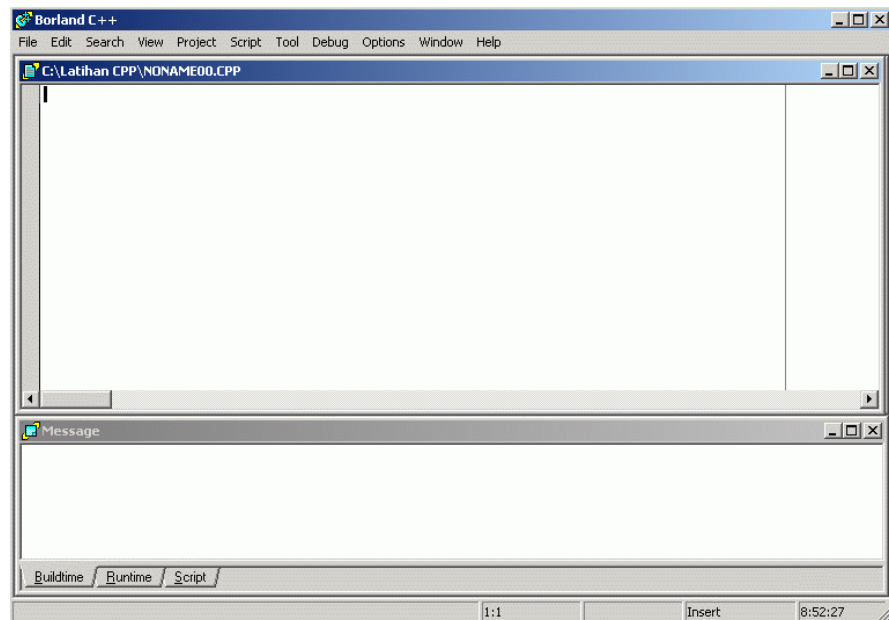
Untuk mengaktifkan aplikasi Borland C++ ver 5.02, anda bisa menggunakan banyak cara, diantaranya :

- Jika anda menggunakan sistem jaringan pada Windows XP, anda bisa membuatkan shortcut terlebih dahulu dari server, dengan cara :

- a. Klik kanan pada Desktop - Klik New - Klik Shortcut
 - b. Ketikkan pada command line : \\BC5\\BIN\\bcw.exe, klik tombol next, kemudian ketikkan nama shortcut, misalkan Borland C++, klik tombol Finish untuk selesai.
 - c. Atau anda dapat klik tombol Browse untuk mencari alamat dan folder yang menyediakan aplikasi Borland C++.
- Jika anda menggunakan Stand Alone Computer, anda bisa membuat shortcut seperti diatas. Biasanya jika sudah terinstall pada komputer anda, sudah dibuatkan shortcut yang anda bisa membukanya dengan cara :
 - Klik tombol Start - pilih All Programs - Borland C++ 5.02 - klik Borland C++
 - Berikut IDE dari Borland C++, seperti gambar dibawah ini;



Gambar 1.1. Layar Pembuka Borland C++



Gambar 1.2. IDE Borland C++ 5.02

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

a. Menu Utama (Menubar)

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

b. Jendela Text Edit

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

c. Jendela Message

Tempat untuk menampilkan pesan-pesan pada proses kompilasi dan link program.

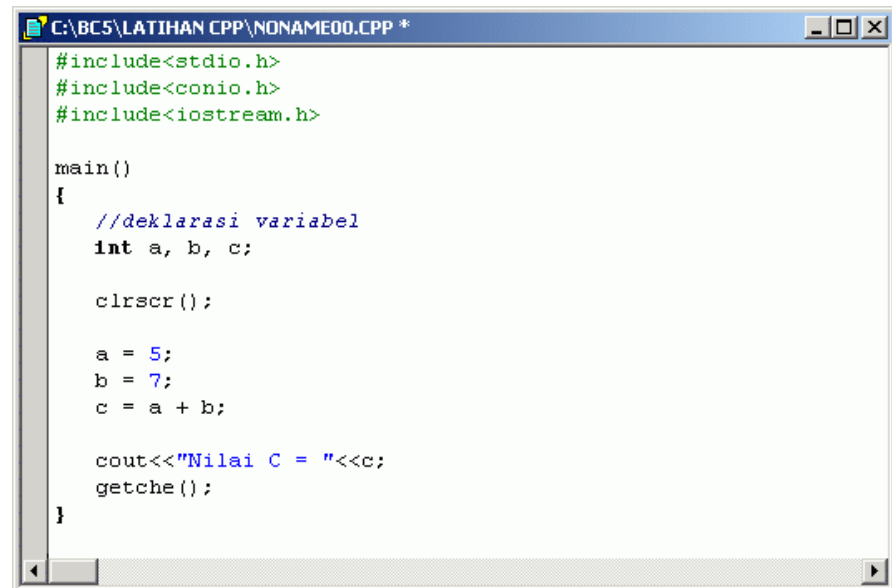
d. Baris Status

Baris dimana menampilkan keterangan-keterangan pada saat anda mengaktifkan menu bar dan sub menu.

1.3. Membuat File Editor

Penjelasan

File Editor merupakan File Program yang dapat dikompile, dan dijalankan untuk menampilkan hasilnya serta mempunyai ekstensi **.CPP**.
Cara mengaktifkannya: Klik Menu File - Klik New - Text Edit



```

C:\BC5\LATIHAN CPP\NONAME00.CPP *
#include<stdio.h>
#include<conio.h>
#include<iostream.h>

main()
{
    //deklarasi variabel
    int a, b, c;

    clrscr();

    a = 5;
    b = 7;
    c = a + b;

    cout<<"Nilai C = "<<c;
    getch();
}

```

Gambar 1.3. Jendela Text Edit

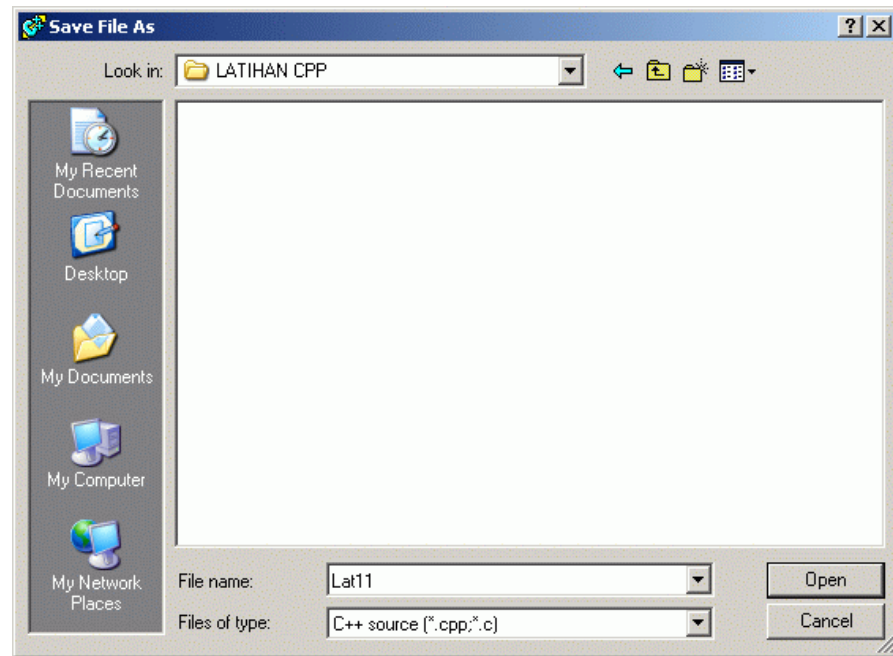
1.4. Menyimpan File Editor

Penjelasan

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

- Klik Menu **F**ile - **S**ave
- Menekan HotKey **Ctrl + KS**.

Selanjutnya tampil jendela Save File As, seperti dibawah ini :



Gambar 1.4. 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. |

1.5. Menterjemahkan Program

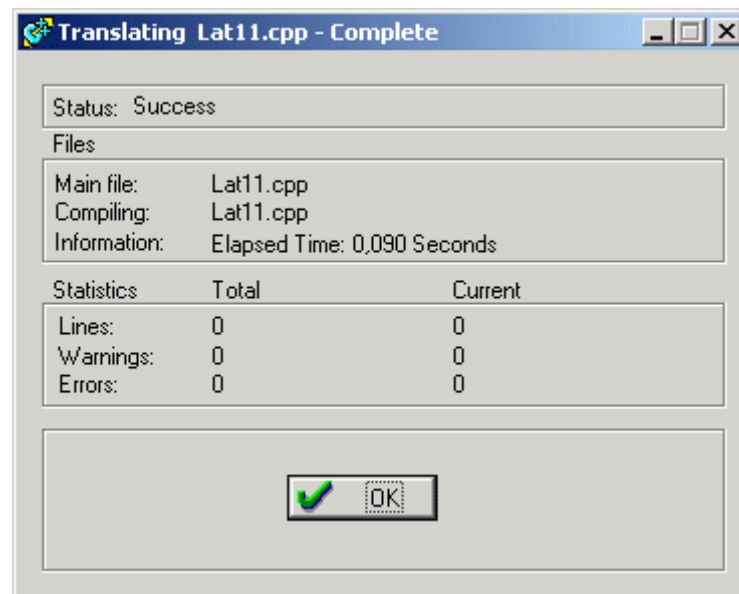
Penjelasan

Proses Compile merupakan suatu proses menterjemahkan program dari bahasa manusia kedalam bahasa yang dimengerti oleh komputer yaitu bahasa mesin.

Caranya adalah :

- Klik Menu Project - Compile
- Menekan HotKey **Alt + F9**

Selanjutnya tampil kotak dialog Compile, seperti dibawah ini :



Gambar 1.5. Kotak Dialog Compile

1.6. Menjalankan Program

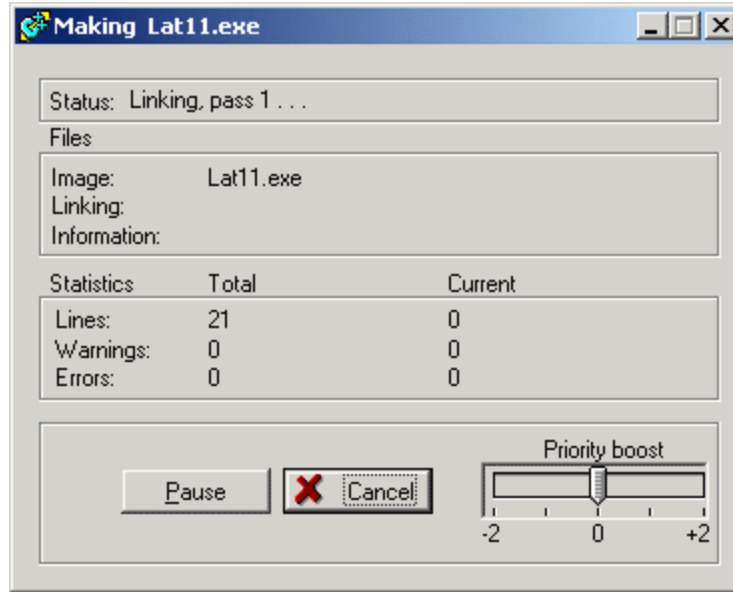
Penjelasan

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

Caranya adalah :

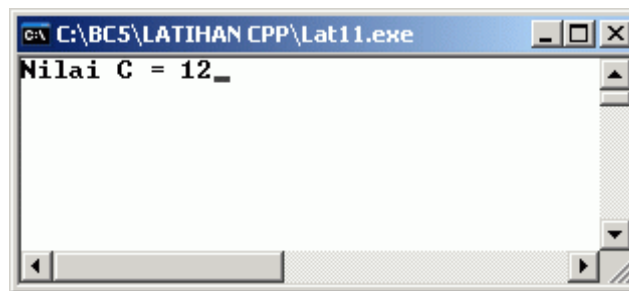
- Klik Menu Debug - Run
- Menekan HotKey **Ctrl + F9**

Selanjutnya tampil kotak dialog Run, seperti dibawah ini :



Gambar 1.6. Kotak Dialog Run

Setelah proses menterjemahkan program, proses linking, selanjutnya tampil hasil seperti gambar 1.7 dibawah ini :



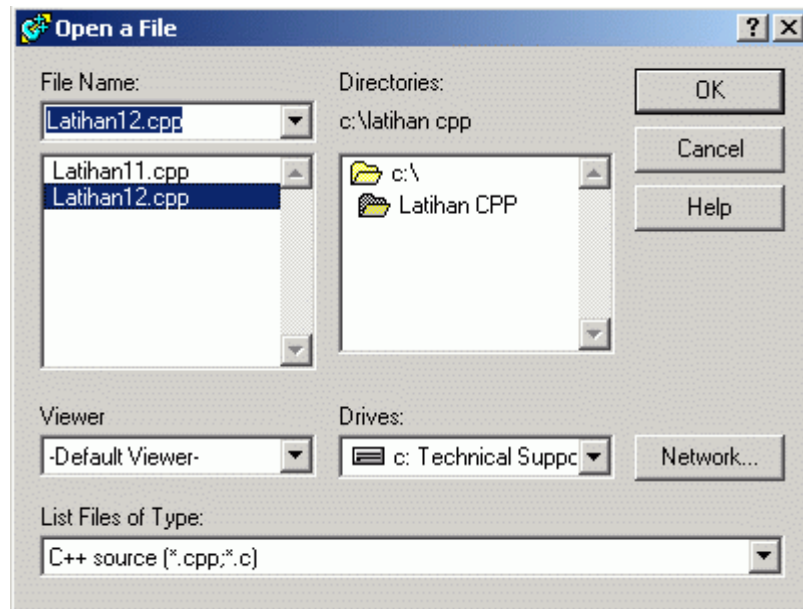
Gambar 1.7. Contoh Hasil Keluaran Program

1.7. 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 :

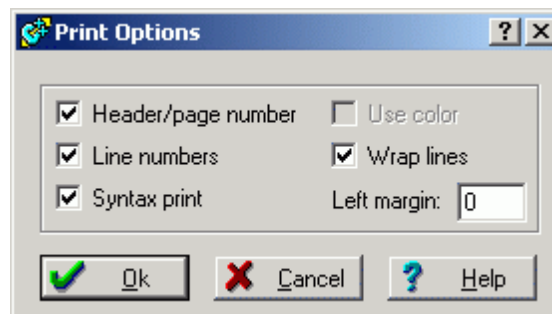


Gambar 1.8. Jendela Open

1.8. Mencetak File Editor

Penjelasan Mencetak file program pada jendela yang sedang aktif dengan cara Klik Menu File - Print

Selanjutnya tampil Jendela Print Option, seperti dibawah ini :



Gambar 1.9. Jendela Print Option

1.9. Keluar dari Borland C++ 5.02

Penjelasan Keluar dari Aplikasi Borland C++ 5.02, dengan cara File - Exit

1.10. Struktur Program C++

Penjelasan

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.

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

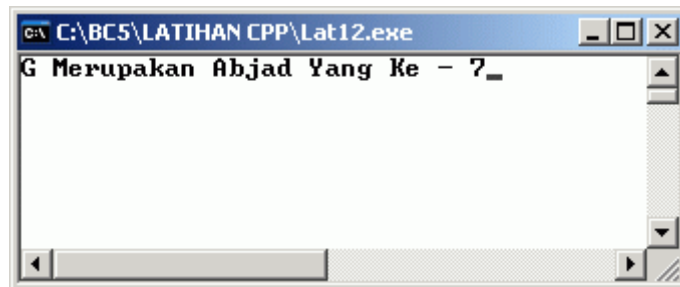
Contoh-1

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

main()
{
    int a = 7;
    char b = 'G';
    clrscr();

    printf("%c Merupakan Abjad Yang Ke - %d", b, a);
}
```

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



Gambar 1.10. Hasil Contoh-1

1.11. Model Memori

Penjelasan

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

- Model Tiny
- Model Small
- Model Medium
- Model Compact
- Model Large
- Model Huge

a. Model Tiny

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

b. Model Small

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

c. Model Medium

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

d. Model Compact

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

c. Model Large

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

d. Model Huge

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

Pemrograman C++

Bab 2 : Pengenalan Model Data, Perintah Masukan dan Keluaran

2.1. Pengenalan Tipe Data

Penjelasan Borland C++ memiliki 7 tipe data dasar, yaitu diantaranya adalah :

Tabel 2.1. Tipe Data

Tipe Data	Ukuran Memori	Jangkauan Nilai	Jumlah Digit
Char	1 Byte	-128 s.d 127	
Int	2 Byte	-32768 s.d 32767	
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

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

- Penjelasan** Konstanta adalah suatu nilai yang sifatnya tetap. Secara garis besar konstanta dapat dibagi menjadi dua bagian, yaitu
- **Konstanta Bilangan**
 - **Konstanta Teks**

2.2.1. Konstanta Bilangan

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

- Konstanta Bilangan Bulat.
Adalah bilangan yang tidak mengandung titik desimal.
Contoh : 1, 2, 3, 100
- Konstanta Desimal Berpresisi Tunggal (Floating Point)
Konstanta Floating Point, mempunyai bentuk penulisan, yaitu :
 - Bentuk Desimal (cth : 5.57)
 - Bentuk Eksponensial / Bilangan Berpangkat (cth : 4.22e3 - 4.22 x 10³)
- 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.

2.2.2. Konstanta Teks

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

- 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 atau notasi atau simbol.
Contoh : 'Y' 'y' '9' '&' dan lain-lain
- 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.

2.2.3. Deklarasi Konstanta

- Penjelasan** Bentuk deklarasi konstanta diawali dengan reserved word **const**.

Bentuk penulisannya :

```
const tipe_data nama-konstanta = nilai konstanta;
```

- Contoh** `const int x = 89;`

2.3. Variabel

Penjelasan

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 ;

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

2.3.1. Variabel Numerik

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

- Bilangan Bulat
- Bilangan Desimal Berpresisi Tunggal atau Floating Point.
- Bilangan Desimal Berpresisi Ganda atau Double Precision.

2.3.2. Variabel Text

- Character (Karakter Tunggal)
- String (Untuk Rangkaian Karakter)

2.3.3. Deklarasi Variabel

Penjelasan

Adalah proses memperkenalkan variabel kepada Borland C++ dan pendeklarasian tersebut bersifat mutlak karena jika tidak diperkenalkan terlebih dulu 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. Tipe 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;
char grade;
float rata_rata ;
int nilai;
```

2.4. Perintah Keluaran

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

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

2.4.1 printf()

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

Bentuk Penulisan

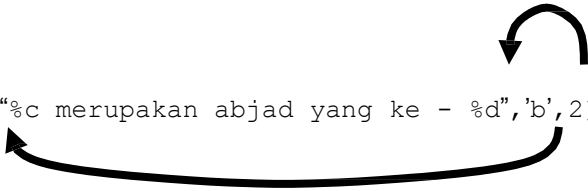
```
printf("string-kontrol", argumen-1, argumen-2, ...);
```

String-Kontrol dapat berupa keterangan yang akan ditampilkan pada layar beserta penentu format. Penentu format dipakai untuk memberi tahu kompiler 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);`



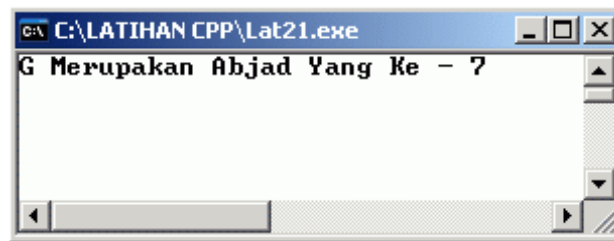
Contoh-1

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

main()
{
    int a = 7;
    char b = 'G';
    clrscr();

    printf("%c Merupakan Abjad Yang Ke - %d", b, a);
}
```

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

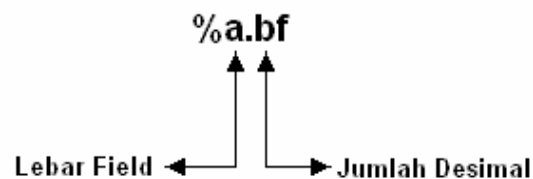


Gambar 2.1. Hasil Contoh-1

a. Penggunaan Penentu Lebar Field

Penjelasan

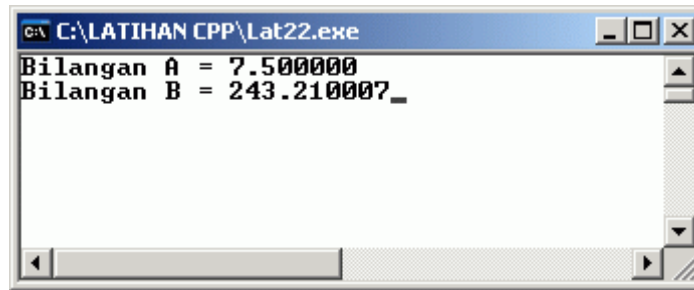
Bila ingin mencetak atau menampilkan data yang bertipe data **float** atau pecahan, tampilan yang tampak biasanya kurang bagus. Hal tersebut dapat diatur lebar field-nya dan jumlah desimal yang ingin dicetak. Berikut bentuk penulisannya :

**Contoh-2**

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

main()
{
    float a = 7.50, b = 243.21;
    clrscr();
    printf("Bilangan A = %f \n", a);
    printf("Bilangan B = %f", b);
}
```

Output yang akan dihasilkan, jika tidak menggunakan panentu lebar field adalah

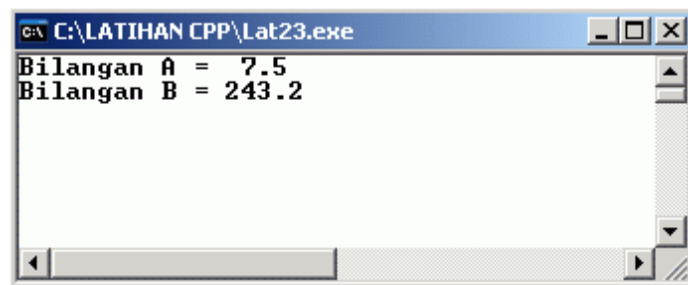


Gambar 2.2. Hasil Contoh-2

Contoh-3

```
#include <stdio.h>
#include <conio.h>
main()
{
    float a = 7.50, b = 243.21;
    clrscr();
    printf("Bilangan A = %4.1f \n", a);
    printf("Bilangan B = %4.1f", b);
}
```

Output yang akan dihasilkan, jika menggunakan panentu lebar field adalah



Gambar 2.3. Hasil Contoh-3

b. Penggunaan Escape Sequences.

Penjelasan

Escape Sequences merupakan suatu karakter khusus yang menggunakan notasi “\” (*back slash*) jika karakter terdapat notasi “\” ini sebagai karakter “escape” (menghindar).

Beberapa Escape Sequences lainnya antara lain :

Tabel 2.5. Escape Sequence

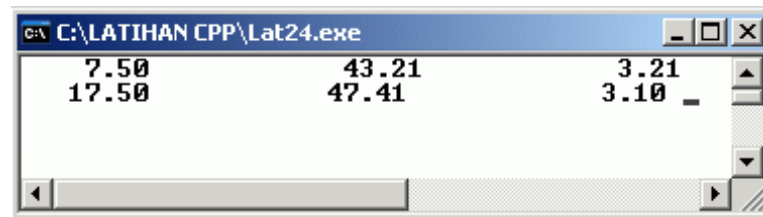
ESCAPE SEQUENCES	PENGERTIAN
\b	Backspace
\f	Formfeed (Pindah Halaman)
\n	NewLine (Baris Baru)
\r	Carriage Return
\t	Tab (default = 7 karakter)

\ '	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()
{
    float a = 7.50, b = 43.21, c = 3.21;
    float d = 17.50, e = 47.41, f = 3.1;
    clrscr();
    printf("%8.2f\t %8.2f\t %8.2f ", a, b, c);
    printf("\n%8.2f\t%8.2f\t%8.2f ", d, e, f);
}
```

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



Gambar 2.4. Hasil Contoh-3

2.4.2. puts()

Penjelasan

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. Perbedaaan puts() dengan printf()

printf()	puts()
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 a[4] = "RPL";
    clrscr();
    puts("Saya Sekolah di Smakensa. ");
    puts(a);
}
```

2.4.3. putchar()

Penjelasan

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

Contoh-6

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

main()
{
    clrscr();
    putchar('R');
    putchar('P');
    putchar('L');
}
```

2.4.4. cout()

Penjelasan

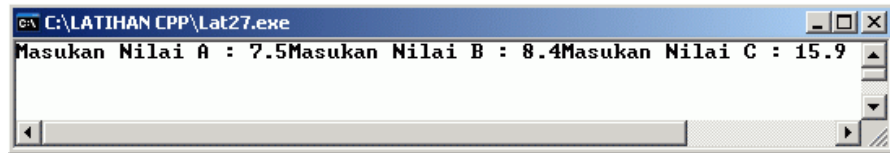
Fungsi **cout()** merupakan sebuah object didalam C++ digunakan untuk menampilkan suatu data kelayar. Untuk menggunakan fungsi cout() ini, harus menyertakan file header **iostream.h** .

Contoh-7

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

main()
{
    float a, b, c;
    a=7.5; b=8.4; c=0;
    clrscr();
    cout<<"Masukan Nilai A : "<<a;
    cout<<"Masukan Nilai B : "<<b;
    c = a + b;
    cout<<"Masukan Nilai C : "<<c;
    getch();
}
```

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



Gambar 2.7. Hasil Contoh-6

2.4.5. Fungsi Manipulator

Penjelasan

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 | • setbase() |
| • end | • setw() |
| • dec() | • setfill() |
| • hex() | • setprecision() |
| • oct() | • setiosflags() |

Berikut akan dibahas beberapa fungsi manipulator, diantaranya :

a. endl

Penjelasan

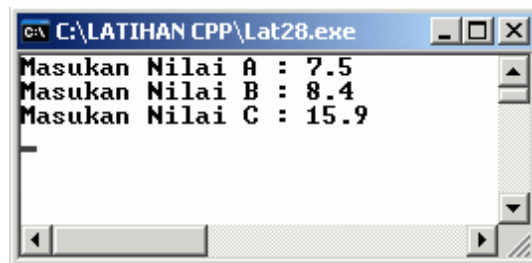
endl merupakan suatu fungsi manipulator yang digunakan untuk menyisipkan karakter NewLine atau mengatur pindah baris. Fungsi ini sangat berguna untuk piranti keluaran berupa file di disk. File header yang harus disertakan file header **iostream.h**.

Contoh-8

```
# include <stdio.h>
# include <conio.h>
# include <iostream.h>
main()
{
    float a, b, c;

    a=7.5; b=8.4; c=0
    clrscr();
    cout<<"Masukan Nilai A : "<<a<<endl;
    cout<<"Masukan Nilai B : "<<b<<endl;
    c = a + b;
    cout<<"Masukan Nilai C : "<<c<<endl;
    getch();
}
```

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



Gambar 2.8. Hasil Contoh-8

b. ends

Penjelasan

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 mengakhirinya dengan karakter NULL.. File header yang harus disertakan file header **iostream.h**.

Contoh-9

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

main()
{
    int a, b, c, d;
    clrscr();
```

```

a = 6;
b = 5;
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

Penjelasan

dec, oct dan hex merupakan suatu fungsi manipulator yang digunakan untuk konversi data dalam bentuk desimal, oktal dan hexadesimal. File header yang harus disertakan file header **iomanip.h**.

Contoh-10

```

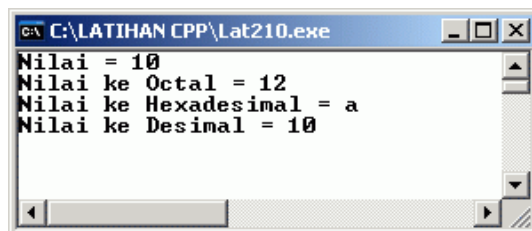
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>

void main()
{
    int nilai = 10;
    clrscr();
    cout<<"Nilai = "<<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 ()

Penjelasan

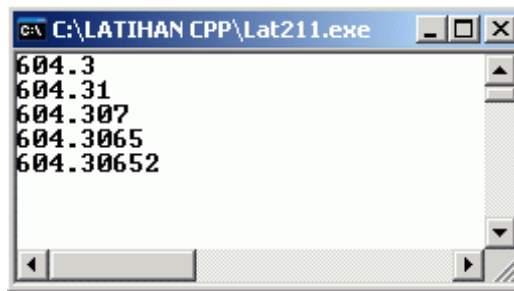
setprecision() merupakan suatu fungsi manipulator yang digunakan untuk mengatur jumlah digit desimal yang ingin ditampilkan. File header yang harus disertakan file header **iomanip.h**.

Contoh-11

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

void main()
{
    float a, b, c;
    a = 25.77;
    b = 23.45;
    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()

Penjelasan

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()

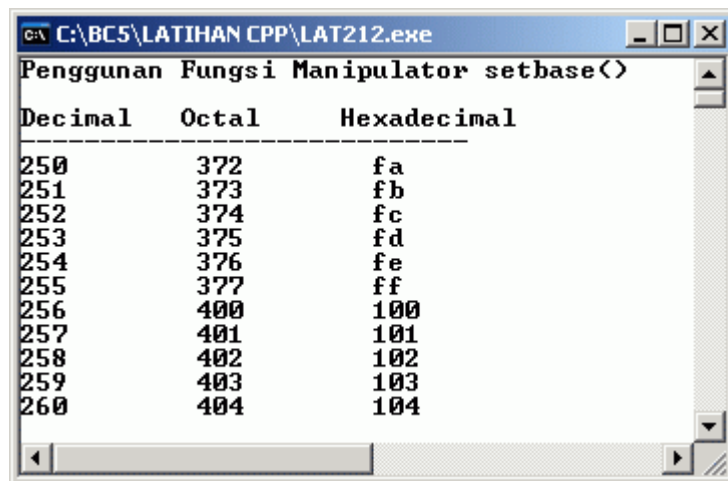
# include <stdio.h>
# include <conio.h>
# include <iostream.h>
# include <iomanip.h>

main()
{
    int a = 250;
    clrscr();

    cout<<"Penggunaan 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;
    }

    getch();
}
```

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



Decimal	Octal	Hexadecimal
250	372	fa
251	373	fb
252	374	fc
253	375	fd
254	376	fe
255	377	ff
256	400	100
257	401	101
258	402	102
259	403	103
260	404	104

Gambar 2.12. Hasil Contoh-12

f. setw()**Penjelasan**

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);
```

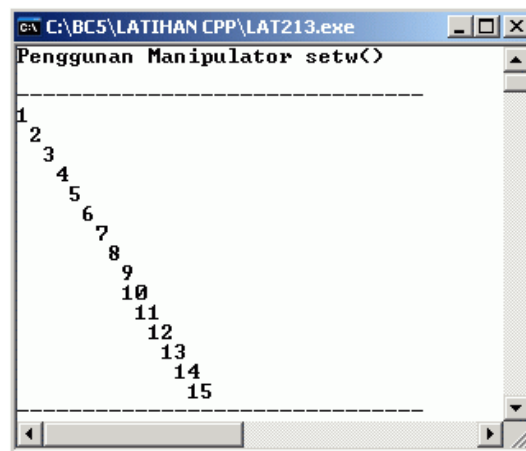
n = merupakan nilai lebar tampilan data, integer.

Contoh-13

```
// Penggunaan Manipulator setw()

# include <stdio.h>
# include <conio.h>
# include <iostream.h>
# include <iomanip.h>
main()
{
    int a;
    clrscr();
    cout<<"Penggunaan Manipulator setw() "<<"\n\n"; cout<<"-
    -----"<<"\n";
    for(a=1;a<=15;a++)
    {
        cout<<setw(a)<<a<<endl;
    }
    cout<<"-----"<<"\n";
    getch();
}
```

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



Gambar 2.13. Hasil Contoh-13

g. setfill()

Penjelasan

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(karakter);
```

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;
    }
    getch();
}
```

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



Gambar 2.14. Hasil Contoh-14

h. setiosflags ()

Penjelasan

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

Penjelasan

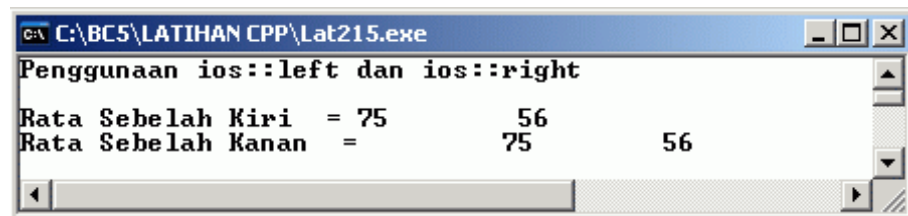
Terdapat dua buah tanda format yang digunakan untuk perataan kiri dan kanan, pengaturan terhadap lebar variabel untuk perataan kiri dan kanan ini melalui fungsi **setw()** .

- Ø **ios::left** digunakan untuk mengatur perataan sebelah kiri
- Ø **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;
    getch();
}
```

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



Gambar 2.15. Hasil Contoh-15

2. Tanda Format Keluaran Notasi Konversi

Penjelasan

Tanda format yang digunakan untuk keluaran Notasi, yaitu :

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

Contoh-16

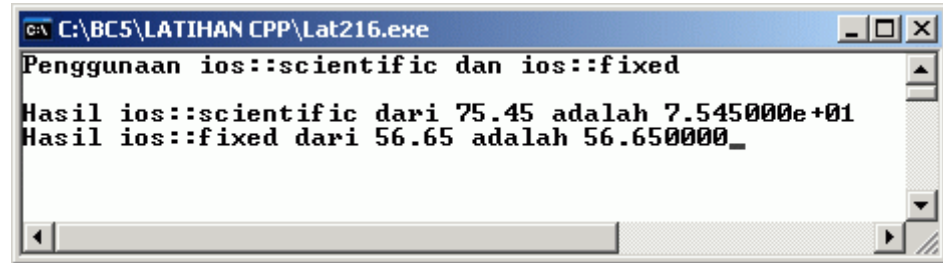
```
//tanda format ios::scientific dan ios::fixed
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
main()
{
    clrscr();
    cout<<"Penggunaan ios::scientific dan ios::fixed\n";
    cout<<"\nHasil ios::scientific dari 75.45 adalah ";
    cout<<setiosflags(ios::scientific)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

```

    cout<<setiosflags(ios::fixed)<<56.65;
    getch();
}

```

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



Gambar 2.16. Hasil Contoh-16

3. Tanda Format Konversi Dec, Oct dan Hex

Penjelasan

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

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

Contoh-17

```

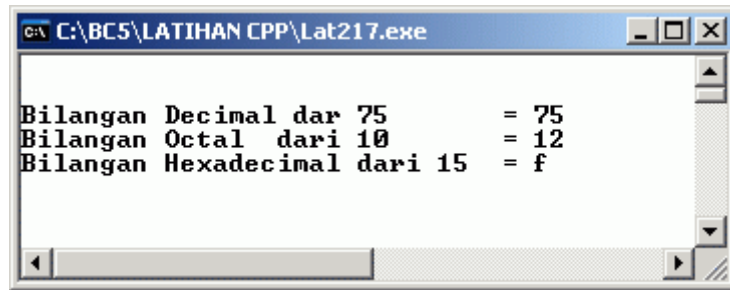
//tanda format ios::dec, ios::oct, ios::hex

# 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;
    getch();
}

```

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



Gambar 2.17. Hasil Contoh-17

4. Tanda Format Manipulasi Huruf Hexadecimal

Penjelasan

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

Ø `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;
    } getch();
}
```

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

Tanpa Konversi	Dengan Konversi
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
a	A
b	B
c	C
d	D
e	E
f	F

Gambar 2.18 Hasil Contoh-18

5. Tanda Format Keluaran Dasar Bilangan Hexadecimal dan Octal

Penjelasan

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

Ø `ios::showbase` digunakan untuk menampilkan tanda `0x` (not-x) diawal pada tampilan bilangan hexadecimal dan `0` (not) 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);

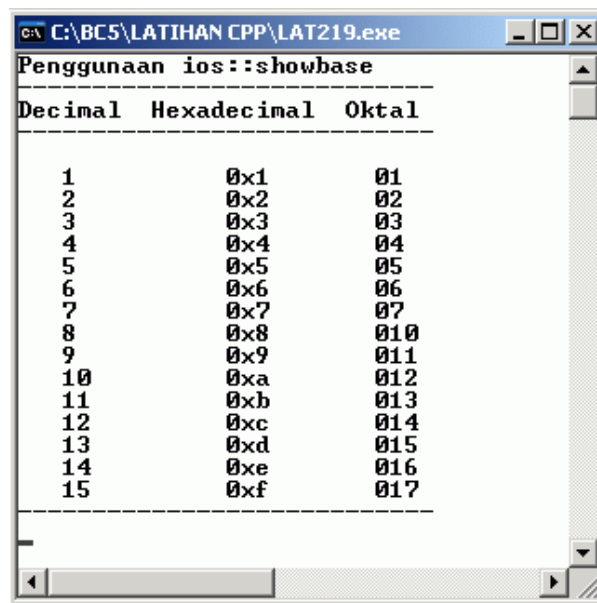
    for (a=1; a<=15; a++)
    {
        gotoxy(4,a+5);
        cout<<dec<<a<<endl;
    }
}
```

```

    }
    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	Oktal
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

Penjelasan

Untuk keperluan menampilkan titik desimal dengan menggunakan tanda format :

Ø **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;

```

```

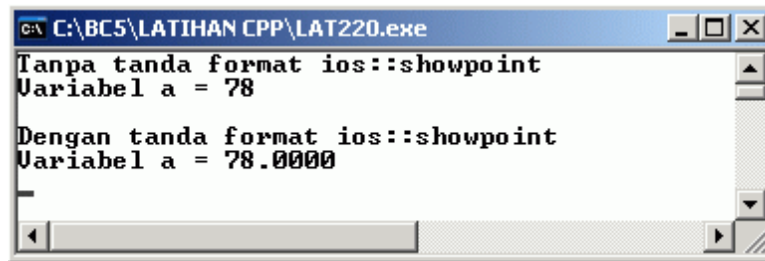
clrscr();

//-> 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-17 diatas adalah :



Gambar 2.17. Hasil Contoh-17

7. Tanda Format Menampilkan Simbol Plus (+)

Penjelasan

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

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

Contoh-21

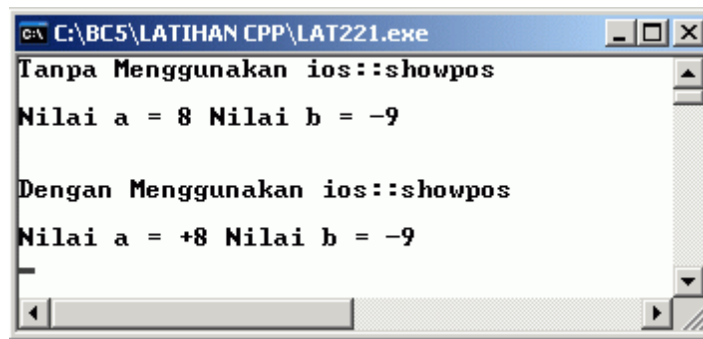
```

//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

Penjelasan

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

- `scanf()`
- `gets ()`
- `cin()`
- `getch()`
- `getche()`

2.5.1. scanf()

Penjelasan

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

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

Keterangan

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 Bentuk Berpangkat	%e atau %f %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

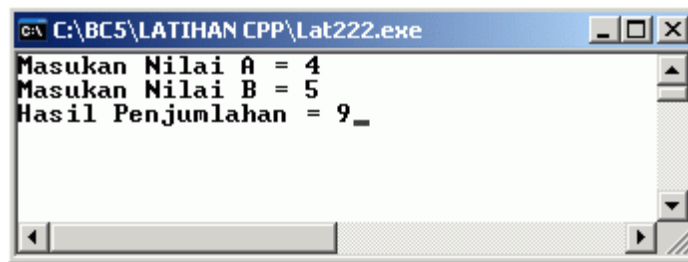
```
# include <stdio.h>
# include <conio.h>

main()
{
    int a, b, c = 0 ;
    clrscr();
    printf("Masukan Nilai A = "); scanf("%d",&a);
    printf("Masukan Nilai B = "); scanf("%d",&b);

    c = a + b;

    printf("Hasil Penjumlahan = %d",c);
}
```

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



Gambar 2.22. Hasil Contoh-22

2.5.2. gets()

Penjelasan

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()

scanf()	gets()
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);

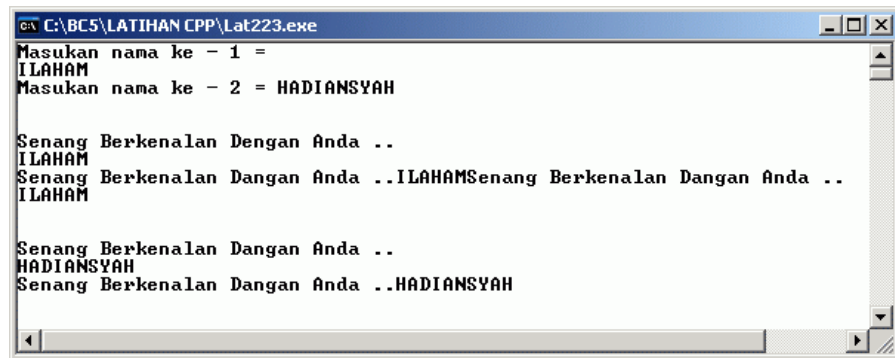
printf("\n\n");

puts("Senang Berkenalan Dengan Anda ..");
puts(nm1);
printf("Senang Berkenalan Dengan Anda ..%s", nm1);
puts("Senang Berkenalan Dengan Anda ..");
puts(nm1);
printf("\n\n");

puts("Senang Berkenalan Dengan Anda ..");
puts(nm2);
printf("Senang Berkenalan Dengan Anda ..%s", nm2);
}

```

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



Gambar 2.23. Hasil Contoh-23

2.5.3. cin ()

Penjelasan

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()
{
    float a, b, c;

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

```

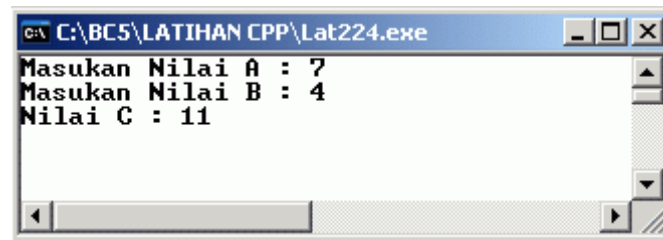


```

    cout<<"Masukan Nilai B : ";
    cin>>b;
    c = a + b;
    cout<<"Nilai C : "<<c<<endl;
}

```

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



Gambar 2.24. Hasil Contoh-24

2.5.4. getch ()

Penjelasan

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 dimasukkan tidak akan ditampilkan di layar. File header yang harus disertakan adalah **conio.h**.

Contoh-25

```

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

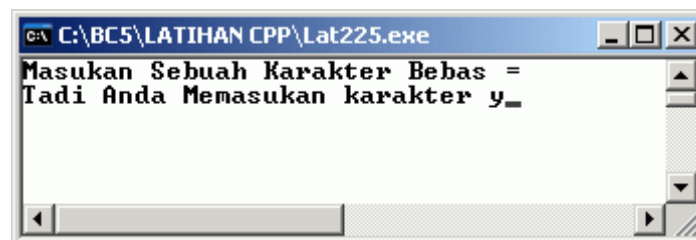
main()
{
    char kar;

    clrscr();

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

```

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



Gambar 2.25. Hasil Contoh-25

2.5.5. getch ()

Penjelasan

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***.

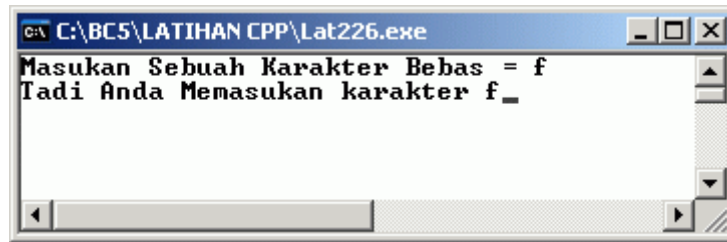
Contoh-26

```
# 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 2.16. Hasil Contoh-16

Selain itu kedua fungsi ini dapat digunakan untuk menahan agar tidak langsung balik kembali kedalam listing program dan hasil dari program yang di eksekusi langsung dapat dilihat. Karena fungsi ***getch()*** merupakan fungsi masukan, jadi sebelum program keluar harus memasukan satu buah karakter.

2.6. Latihan

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.
 - Tampilan yang diinginkan sebagai berikut :

Layar Masukkan

PROGRAM HITUNG NILAI RATA-RATA

```
Nama Siswa : ... < diinput >
Nilai Pertandingan I : ... < diinput >
Nilai Pertandingan II : ... < diinput >
Nilai Pertandingan III : ... < diinput >
```

Layar Keluaran

Siswa yang bernama ...<tampil data>
Memperoleh nilai rata-rata ...<hasil proses> 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
 - Tampilan yang diinginkan sebagai berikut :

Layar Masukkan

PROGRAM HITUNG NILAI AKHIR

```
Nama Siswa : ... < diinput >
Nilai Keaktifan : ... < diinput >
Nilai Tugas : ... < diinput >
Nilai Ujian : ... < diinput >
```

Layar Keluaran

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

Jadi Siswa yang bernama ... <hasil proses>
memperoleh nilai akhir sebesar ... <hasil proses>

Lembar ini Sengaja Dikosongkan
(Untuk Catatan Boleh Juga)

Pemrogramman C++

Bab 3 : Operator Borland C++

Penjelasan

Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program 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

Penjelasan

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	$5 \% 2$
+	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 a, b, c = 0, d = 0;
    clrscr();
    cout<<"Masukan Nilai A : "; cin>>a;
    cout<<"Masukan Nilai B : "; cin>>b;

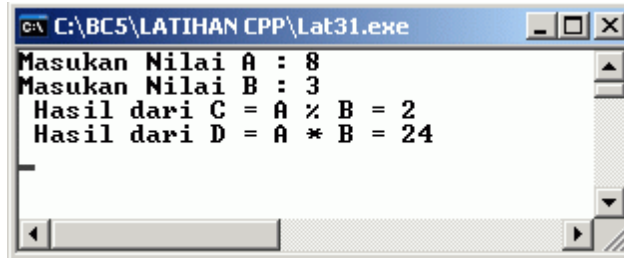
    c = a % b;
    d = a * b;

    cout<<" Hasil dari C = A % B = "<<c<<endl;
    cout<<" Hasil dari D = A * B = "<<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 :

$$\underbrace{\text{Varabel}}_{\text{LValue}} = \underbrace{\text{Ekspresi Aritmatika}}_{\text{RValue}}$$

- 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.

Penjelasan

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

Tabel. 3.3. Tabel Hierarki Operator Aritmatika

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

Contoh

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

Langkah perhitungannya :

$$A = 8 + 6 / 6 - (6 / 6 = 1)$$

$$A = 8 + 1$$

$$A = 9$$

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

```

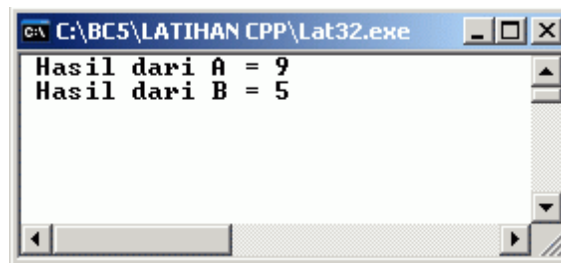
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

main()
{
    int a, b;
    clrscr();
    a = 8 + 2 * 3 / 6;
    b = (8 + 2) * 3 / 6;

    cout<<" Hasil dari A = "<<a<<endl;
    cout<<" Hasil dari B = "<<b<<endl;
    getch();
}

```

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



Gambar 3.2. Hasil Contoh-2

3.2. Operator Pemberi Nilai Aritmatika

Penjelasan

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 aritmatika diantaranya :

Tabel. 3.4. Tabel Operator Pemberi Nilai Aritmatika

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

3.3. Operator Penambah dan Pengurang

Penjelasan

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 :

$A += 1$ atau $A -= 1$; masih dapat disederhanakan menjadi $A ++$ atau $A --$

Notasi “++” atau “--” dapat diletakan didepan atau di belakang variabel.

Contoh

$A ++$ atau $++A$ / $A --$ atau $--A$

Kedua bentuk penulisan notasi 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-3

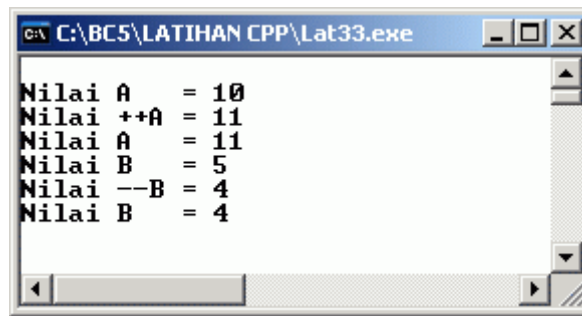
```
/* Penggunaan Notasi Didepan Variabel*/
#include <stdio.h>
#include <conio.h>

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

    printf("Nilai A    = %d", a);
    printf("\nNilai ++A = %d", ++a);
    printf("\nNilai 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-3 diatas adalah :



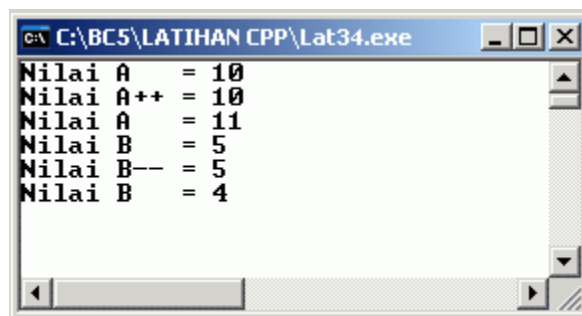
Gambar 3.3 Hasil Contoh-3

Contoh-4

```
/* Penggunaan Notasi Dibelakang 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 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

Penjelasan

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

Tabel. 3.6. 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>

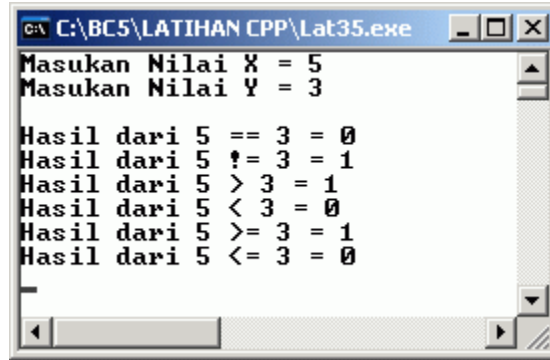
main()
{
    float a, b, c, d, e, f, x, y;
    clrscr();

    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

Penjelasan

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. 3.7. Tabel Operator Relasi

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

3.5.1. Operator Logika AND

Penjelasan

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

Contoh :

Ekspresi Relasi-1 - $A + 4 < 10$

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

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

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

Dari ekspresi relasi tersebut mempunyai nilai BENAR, maka

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

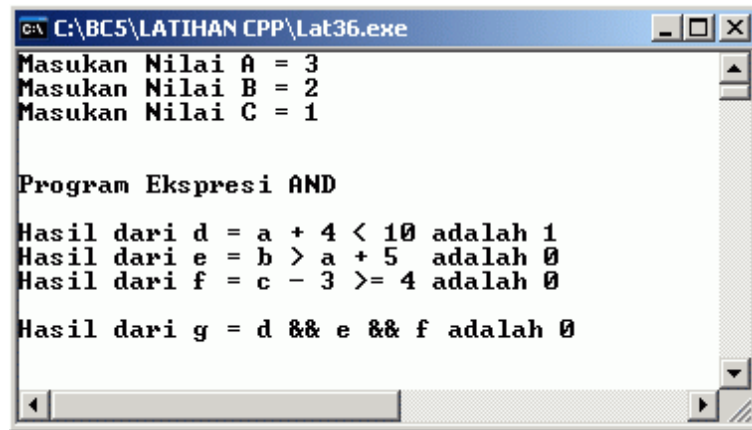
    clrscr();
    cout<<"Masukan Nilai A = "; cin>>a;
    cout<<"Masukan Nilai B = "; cin>>b;
    cout<<"Masukan Nilai C = "; cin>>c;

    // 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

Penjelasan

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.

Contoh :

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

Penggabungan ketiga ekspresi relasi diatas menjadi ;

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

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

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

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

$A + 4 < 10 \ || \ B > A + 5 \ || \ C - 3 > 4$ - BENAR = 1

Contoh-7

```

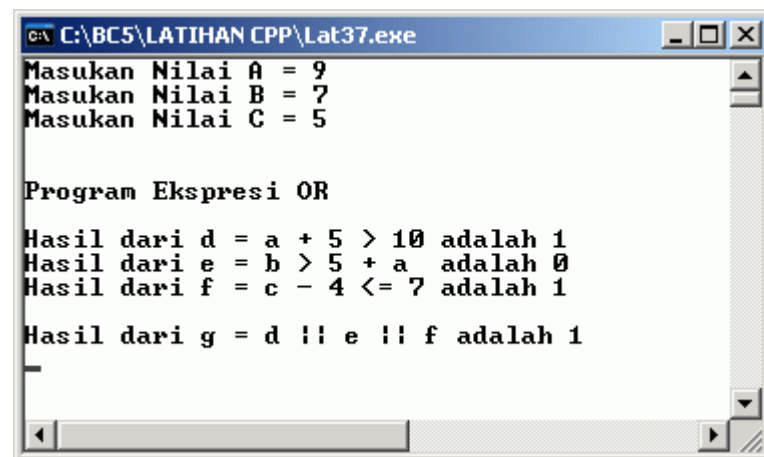
/* Penggunaan Operasi Logika OR */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
    float a, b, c, d, e, f, g, h;
    clrscr();
    cout<<"Masukan Nilai A = "; cin>>a;
    cout<<"Masukan Nilai B = "; cin>>b;
    cout<<"Masukan Nilai C = "; cin>>c;

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

    cout<<endl<<endl;
    cout<<"Program Ekspresi OR"<<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

3.5.3. Operator Logika NOT

Penjelasan

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 - $A + 4 < 10$ - $3 + 4 < 10$ - 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) - ! (BENAR) = SALAH = 0

Contoh-8

```
/* Penggunaan Operasi Logika NOT */

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

main()
{
    int a, b, c;
    clrscr();

    cout<<"Masukan Nila 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 :

```

C:\BC5\LATIHAN CPP\Lat38.exe
Masukan Nila A = 6
Program Ekspresi NOT
Nilai A = 6
Nilai b = (a + 4 < 10) = 0
Nilai c = !(b) = 1
  
```

Gambar 3.8 Hasil Contoh-8

3.6. Operator Bitwise

Penjelasan

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

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

Tabel. 3.6. Tabel Operator Bitwise

3.6.1. Operator Bitwise << (Shift Left)

Penjelasan

Operator Bitwise Shift Left digunakan untuk menggeser sejumlah bit kekiri.

Contoh :

```

0000000011001001 = 201
      //////////
00000000110010010 = 402
  
```

- digeser 1 bit ke kiri

Dibagian kanan disisipkan 0, sebanyak bit yang digeser

Contoh-9

```

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

void main()
{
    int x;
    clrscr();

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

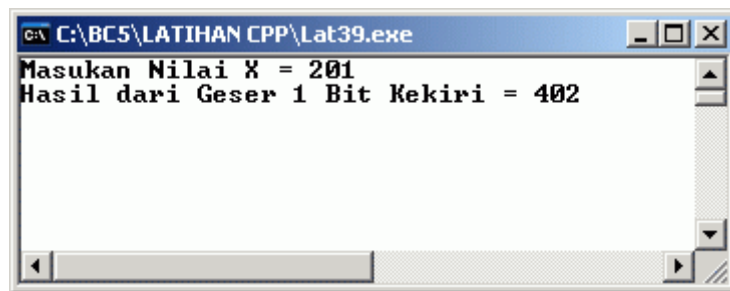
    x = x << 1;

    cout<<"Hasil dari Geser 1 Bit Kekiri = "<<x<<endl;

    getch();
}

```

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



Gambar 3.9 Hasil Contoh-9

3.6.2. Operator Bitwise >> (Shift Right)

Penjelasan

Operator Bitwise Shift Right digunakan untuk menggeser sejumlah bit kanan.

Contoh :

```

0000000011001001 = 201
      \\\\\\\\\\\ - digeser 1 bit ke kanan
0000000001100100 = 100

```

Dibagian kanan disisipkan 0, sebanyak bit yang digeser

Contoh-10

```

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

    x = x >> 1;

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

```

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



Gambar 3.10 Hasil Contoh-10

3.6.3. Operator Bitwise & (And)

Penjelasan

Operator Bitwise & (And) digunakan untuk membandingkan bit dari dua operand. Akan bernilai benar (1) jika semua operand yang digabungkan bernilai benar (1). Berikut anda dapat melihat 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
----- AND
01000000 = 64

```

Contoh-11

```

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

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

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

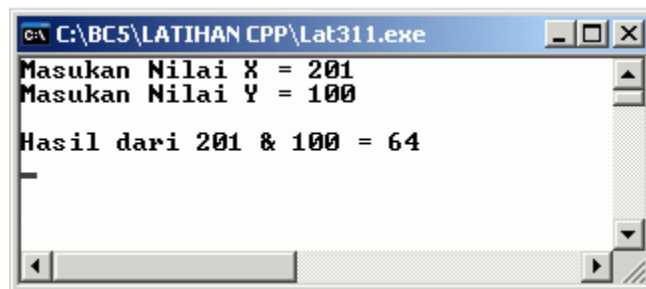
    a = x & y;

    cout<<'\\n';

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

```

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



Gambar 3.11 Hasil Contoh-11

3.6.4. Operator Bitwise | (Or)

Penjelasan

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 anda dapat melihat ilustrasi untuk membandingkan bit dari 2 operand.

Tabel. 3.8. Tabel Operator Bitwise Or

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

Contoh :

```
11001001 = 201
01100100 = 100
----- OR
11101101 = 237
```

Contoh-12

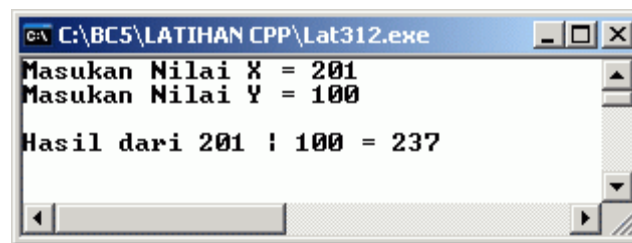
```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
main()
{
    int a, x, y;
    clrscr();

    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

3.6.5. Operator Bitwise ^ (eXclusive Or)

Penjelasan

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 anda dapat melihat 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 = 173

```

Contoh-13

```

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

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

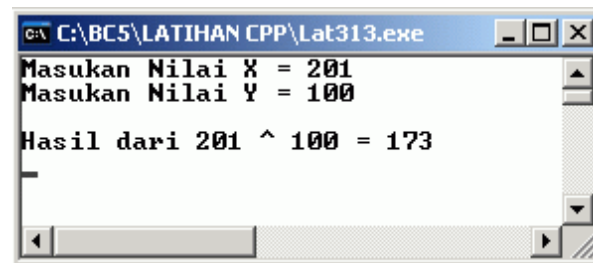
    cout<<"Masukan Nilai X = ";
    cin>>x;
    cout<<"Masukan Nilai Y = ";
    cin>>y;

    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 ~ (Not)

Penjelasan

Operator Bitwise ~ (Not) digunakan membalik nilai bit dari suatu operand. Berikut anda dapat melihat 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-14

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

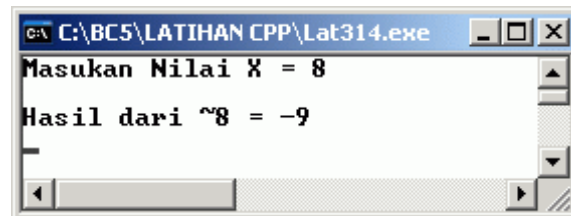
main()
{
    int a, x, y;
    clrscr();
    cout<<"Masukan Nilai X = ";
    cin>>x;
    a = ~x;

    cout<<'\\n';

    cout<<"Hasil dari ~"<<x<<" = "<<a<<endl;

    getch();
}
```

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



Gambar 3.14 Hasil Contoh-14

3.7. Latihan

Penjelasan No. 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

- $D = (4 + 2 > A \ \&\& \ B - 2 > 3 + 2 \ || \ B + 2 \leq 6 + 2)$
- $K + 5 < M \ || \ (C * M < L \ \&\& \ 2 * M - L > 0)$
- $L + 5 < M \ || \ C * K < L \ \&\& \ 2 * K - L > 0$
- $A * 4 \leq 3 * M + B$
- $K + 10 > A \ \&\& \ L - 2 > 4 * C$

Penjelasan No. 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();
}

```

Penjelasan No. 3 Dari program dibawah ini, bagaimanakah keluaran yang dihasilkan

```

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

main()
{
    int a = 25;
    cout<<endl<<"Nilai a = "<<a;
    cout<<endl<<"Nilai a++ = "<<a++;
    cout<<endl<<"Nilai ++a = "<<++a;
    cout<<endl<<"Nilai a-- = "<<a--;
    cout<<endl<<"Nilai a = "<<a;

    a*=2;

    cout<<endl<<"Nilai a = "<<a;
    cout<<endl<<"Nilai ++a = "<<++a;
    cout<<endl<<"Nilai a++ = "<<a++;
    cout<<endl<<"Nilai --a = "<<--a;
    cout<<endl<<"Nilai a-- = "<<a--;

    getch();
}

```

Pemrogramman C++

Bab 4 : Operasi Penyeleksian Kondisi

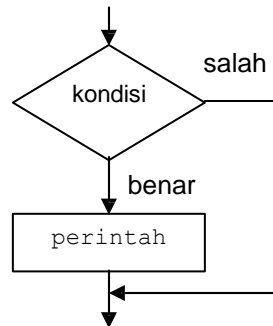
Penjelasan

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

Penjelasan

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

```

if (kondisi)
    pernyataan;
  
```

Penulisan **kondisi** harus didalam tanda kurung dan berupa ekspresi relasi dan penulisan pernyataan dapat berupa sebuah pernyataan tunggal, pernyataan majemuk atau pernyataan kosong. Jika pemakaian *if* diikuti dengan pernyataan majemuk, bentuk penulisannya sebagai berikut :

```

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

```

Contoh

Menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria :

- Tidak ada potongan jika total pembelian kurang dari Rp. 50.000,-
- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.

Contoh-1

```

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

main()
{
    double tot_beli, potongan=0, jum_bayar=0;
    clrscr();

    cout<<"Total Pembelian Rp. ";
    cin>>tot_beli;

    if (tot_beli >= 50000)
        potongan = 0.2 * tot_beli;

    cout<<"Besarnya Potongan Rp. "<<potongan<<endl;

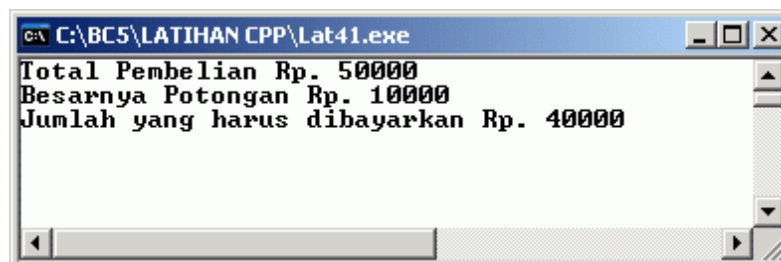
    jum_bayar = tot_beli - potongan;

    cout<<"Jumlah yang harus dibayarkan Rp. "<<jum_bayar;

    getch();
}

```

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

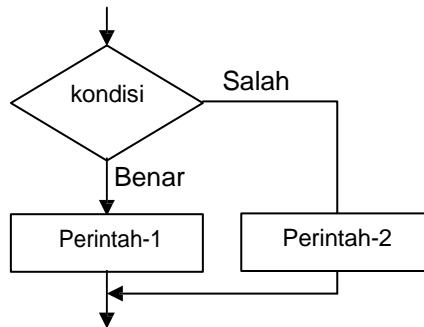


Gambar 4.1. Hasil Contoh-1

4.1.1. Pernyataan IF - ELSE

Penjelasan

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 7.2. Diagram Alir if-else

Bentuk umum dari pernyataan *if*

```

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-1)
{
  perintah-1;
  ....
}
else
{
  perintah-2;
  ....
}
  
```

Contoh

Menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria :

- jika total pembelian kurang dari Rp. 50.000,- potongan yang diterima sebesar 5% dari total pembelian.
- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.

Contoh-2

```
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
    double tot_beli, potongan=0, jum_bayar=0;
    clrscr();

    cout<<"Total Pembelian Rp. ";
    cin>>tot_beli;

    if (tot_beli >= 50000)
        potongan = 0.2 * tot_beli;
    else
        potongan = 0.05 * tot_beli;

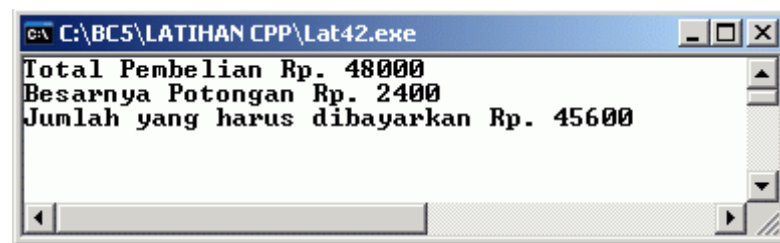
    cout<<"Besarnya Potongan Rp. "<<potongan<<endl;

    jum_bayar = tot_beli - potongan;

    cout<<"Jumlah yang harus dibayarkan Rp. "<<jum_bayar;

    getch();
}
```

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



Gambar 4.2. Hasil Contoh-2

4.1.2. Pernyataan NESTED IF

Penjelasan

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 memberikan komisi kepada para salesman dengan ketentuan sebagai berikut:

- Bila salesman dapat menjual barang hingga Rp. 20.000 ,- , 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. 20.000 ,- , 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. 50.000 ,- , akan diberikan uang jasa sebesar Rp. 30.000 ditambah dengan uang komisi Rp. 20% dari pendapatan yang diperoleh hari itu.

Contoh-3

```

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

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

    if (pendptan >= 0 && pendptan <= 200000)
    {
        jasa=10000;
        komisi=0.1*pendptan;
    }
    else
    {
        if(pendptan<=500000)
        {
            jasa=20000;
            komisi=0.15*pendptan;
        }
        else
        {
            jasa=30000;
            komisi=0.2*pendptan;
        }
    }
}

```

```

/* 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-3 diatas adalah :



Gambar 4.3 Hasil Contoh-3

4.1.3. Pernyataan IF – ELSE Majemuk

Penjelasan

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 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 pendptan, jasa=0, komisi=0, total=0;
    clrscr();

    cout>>"Pendapatan Hari ini Rp. ";
    cin<<pendptan;

    if (pendptan >= 0 && pendptan <= 200000)
    {
        jasa=10000;
        komisi=0.1*pendptan;
    }

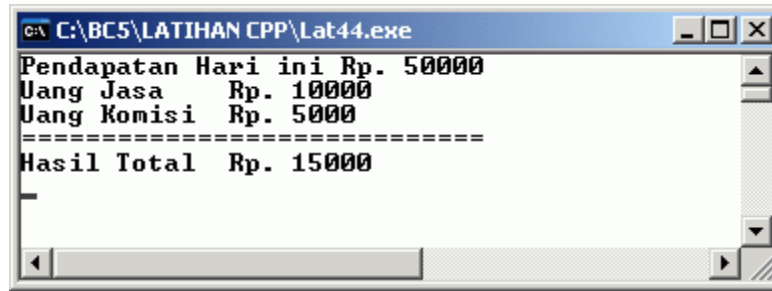
    else if(pendptan<=500000)
    {
        jasa=20000;
        komisi=0.15*pendptan;
    }
    else
    {
        jasa=30000;
        komisi=0.2*pendptan;
    }

    /* 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 :



```

C:\BC5\LATIHAN CPP\Lat44.exe
Pendapatan Hari ini Rp. 50000
Uang Jasa      Rp. 10000
Uang Komisi    Rp. 5000
=====
Hasil Total    Rp. 15000
  
```

Gambar 4.4 Hasil Contoh-4

4.2. Pernyataan switch - case

Penjelasan

Bentuk dari **switch - case** merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif. 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 cabang akan dijalankan jika syarat nilai konstanta tersebut dipenuhi dan default akan dijalankan jika semua cabang diatasnya tidak terpenuhi. Pernyataan **break** menunjukan bahwa perintah siap keluar dari **switch**. Jika pernyataan ini tidak ada, maka program akan diteruskan kecabang – cabang yang lainnya.

Contoh-5

```

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

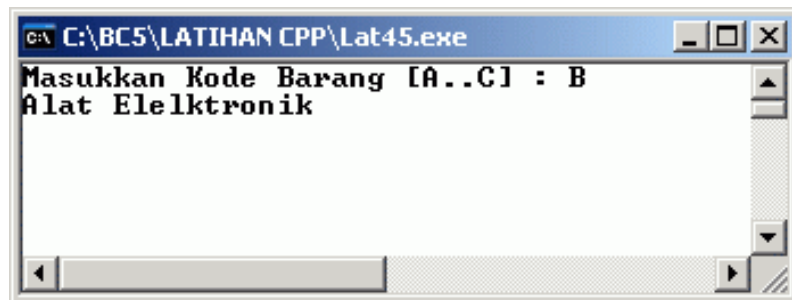
```

main()
{
    char kode;
    clrscr();
    cout<<"Masukkan Kode Barang [A..C] : ";
    cin>>kode;

    switch(kode)
    {
        case 'A' :
            cout<<"Alat Olah Raga";
            break;
        case 'B' :
            cout<<"Alat Elelctronik";
            break;
        case 'C' :
            cout<<"Alat Masak";
            break;
        default:
            cout<<"Anda Salah Memasukan kode";
            break;
    } getch();
}

```

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



Gambar 4.5 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 Elelkttronik";
        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 4.6 Hasil Contoh-6

4.3. Operator ?:

Penjelasan

Operator ?: disebut dengan **Conditional Operator** atau Operator Kondisi yang digunakan untuk menyeleksi nilai untuk mendapatkan hasil dari kondisi yang diseleksi. Operator ?: ini tergolong kedalam operator ternary.

Bentuk Penulisan :

Ekspresi Logika-OR ? Ekspresi : Ekspresi Kondisi

Contoh-7

```

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

```

```

main()
{
    int x, y , z ;

    clrscr();

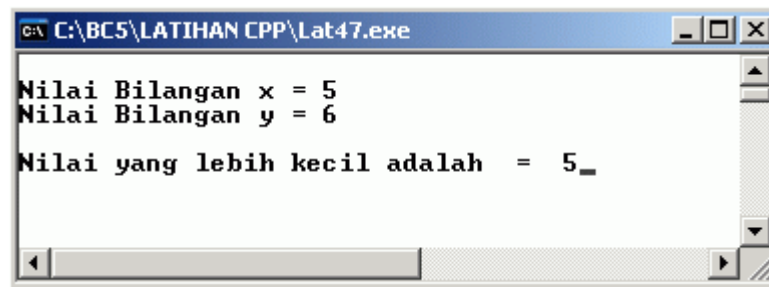
    x = 5;
    y = 6;

    z = (x < y) ? x : y;

    cout<<"\nNilai Bilangan x = "<<x;
    cout<<"\nNilai Bilangan y = "<<y<<endl;
    cout<<"\nNilai yang lebih kecil adalah  = "<< z;
    getche();
}

```

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



Gambar 4.7 Hasil Contoh-7

Pada contoh program – 7 diatas, merupakan pengaplikasian dari perintah if – else berikut :

Contoh-8

```

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

main()
{
    int x, y , z ;
    clrscr();

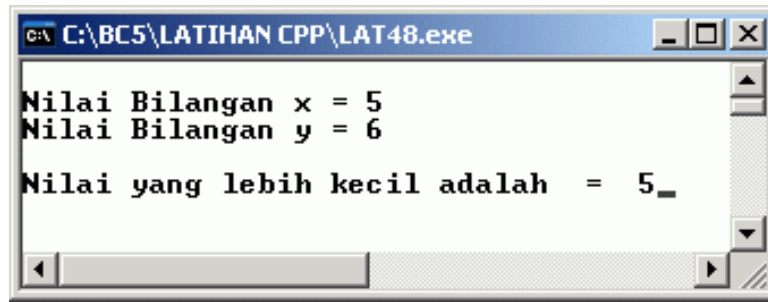
    x = 5;
    y = 6;

    if (x < y)
        z = x;
    else
        z = y;

    cout<<"\nNilai Bilangan x = "<<x;
    cout<<"\nNilai Bilangan y = "<<y<<endl;
    cout<<"\nNilai yang lebih kecil adalah  = "<< z;
    getche();
}

```

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



Gambar 4.8 Hasil Contoh-8

4.4. Latihan

Penjelasan No. 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 Mendapat Hadiah dari pertandingan :
 - Jika Nilai Rata-Rata ≥ 85 , maka mendapat hadiah Seperangkat Komputer P4
 - Jika Nilai Rata-Rata ≥ 70 , maka mendapat hadiah Seperangkat Uang sebesar Rp. 500,000
 - Jika Nilai Rata-Rata < 70 , maka mendapat hadiah Hiburan
- Tampilan yang diinginkan sebagai berikut :

Layar Masukkan

PROGRAM HITUNG NILAI RATA-RATA

```
Nama Siswa : ... <diinput>
Nilai Pertandingan I : ... <diinput>
Nilai Pertandingan II : ... <diinput>
Nilai Pertandingan III : ... <diinput>
```

Layar Keluaran

```
Siswa yang bernama ... <tampil data>
Memperoleh nilai rata-rata <hasil proses> dari hasil
perlombaan yang diikutinya.
Hadiah yang didapat adalah ... <hasil proses>
```

Penjelasan No. 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 Keaktifaan 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 untuk mendapatkan grade nilai :
 - Nilai Akhir ≥ 80 mendapat Grade A
 - Nilai Akhir ≥ 70 mendapat Grade B
 - Nilai Akhir ≥ 59 mendapat Grade C
 - Nilai Akhir ≥ 50 mendapat Grade D
 - Nilai Akhir < 50 mendapat Grade E
- Tampilan yang diinginkan sebagai berikut :

Layar Masukkan

PROGRAM HITUNG NILAI AKHIR

```
Nama Siswa : .....<diinput>
Nilai Keaktifan : ..... <diinput>
Nilai Tugas : ..... <diinput>
Nilai Ujian : ..... <diinput>
```

Layar Keluaran

```
Siswa yang bernama <tampil data>
Dengan Nilai Persentasi Yang dihasilkan.
  Nilai Keaktifan * 20% : ...<hasil proses>
  Nilai Tugas * 30% : ...<hasil proses>
  Nilai Ujian * 50% : ...<hasil proses>

Jadi Siswa yang bernama <tampil data>memperoleh
nilai akhir sebesar ... <hasil proses>
Grade nilai yang didapat adalah ... <hasil proses>
```

Penjelasan No. 3 Buatlah program untuk menghitung total pembayaran dari sebuah penjualan agen susu di kota besar ini.. Dengan ketentuan sebagai berikut :

- Jenis susu diinput diinput berdasarkan kode yang sudah ditentukan
 - Jika kode A adalah Dancow
 - Jika kode B adalah Bendera
 - Jika kode A adalah SGM
- Ukuran kaleng susu diinput berdasarkan kode yang sudah ditentukan.
 - Jika kode 1 adalah Kecil
 - Jika kode 2 adalah Sedang
 - Jika kode 3 adalah Besar
- Harga susu sesuai dengan jenis susu dan ukuran kaleng susu

JENIS SUSU	HARGA BERDASARKAN UKURAN KALENG SUSU		
	KECIL	SEDANG	BESAR
DANCOW	25000	20000	15000
BENDERA	20000	17500	13500
SGM	22000	18500	15000

- Proses yang dilakukan untuk mendapatkan Total Pembayaran

Total Bayar = Harga Susu per ukuran dan Jenis dikali dengan banyak beli

- Tampilan yang diinginkan sebagai berikut :

Layar Masukkan

TOKO KELONTONG KERONCONGAN

- ```

A. Susu Dancow
 1. Ukuran Kecil
 2. Ukuran Sedang
 3. Ukuran Besar
B. Susu Bendera
 1. Ukuran Kecil
 2. Ukuran Sedang
 3. Ukuran Besar
C. Susu SGM
 1. Ukuran Kecil
 2. Ukuran Sedang
 3. Ukuran Besar

```

#### Layar Keluaran

```

Nasukan Jenis Susu : < diinput >
Nasukan Ukuran Kaleng : < diinput >

Harga Satuan Barang Rp.< tampil harga satuan >

Jumlah Yang dibeli : ... < diinput >

Harga Yang Harus dibayar Sebesar Rp. <hasil proses>

```

**Penjelasan No. 4 PT. DINGIN DAMAI**, memberikan Honor tetap kepada karyawan kontraknya sebesar Rp. 300,000,- per bulan, dengan memperoleh tunjangan-tunjangan sebagai berikut :

- Tunjangan Jabatan

| Golongan | Persentase |
|----------|------------|
| 1        | 5%         |
| 2        | 10%        |
| 3        | 15%        |



Sebagai contoh : Jika seorang keryawan tersebut dengan golongan 3, maka mendapatkan tunjangan sebesar  $15\% \times \text{Rp. } 300,000,-$

- Tunjangan Pendidikan

| Kode | Pendidikan | Persentase |
|------|------------|------------|
| 1    | SMU        | 2,5%       |
| 2    | D3         | 5%         |
| 3    | S1         | 7,5%       |

- Honor Lembur

Jumlah jam kerja normal sebanyak 8 Jam Kerja. Honor lembur diberikan jika jumlah jam kerja lebih dari 8 jam, maka kelebihan jam kerja tersebut dikalikan dengan honor lembur perjam sebesar Rp. 2,500 untuk setiap kelebihan jam kerja perharinya.

- Tampilan yang diinginkan sebagai berikut :

#### Layar Masukkan

```
Program Hitung Honor Karyawan Kontrak
PT. DINGIN DAMAI
```

```
Nama Karyawan : ... <di input>
Golongan : ... <di input>
Pendidikan (SMU/D3/S1) : ... <di input>
Jumlah Jam Kerja : ... <di input>
```

#### Layar Keluaran

```
Karyawan yang bernama : ... <tampil data>
Honor yang diterima
```

```
Honor Tetap Rp. <hasil proses>
Tunjangan jabatan Rp. <hasil proses>
Tunjangan Pendidikan Rp. <hasil proses>
Honor Lembur Rp. <hasil proses>
 +
Honor Yang Diterima Rp. <hasil proses>
```

Lembar Ini Sengaja Dikosongkan  
( Untuk Catatan Boleh Juga )

## Pemrogramman C++

## Bab 5 : Proses Perulangan

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

### 5.1. Pernyataan for

**Penjelasan** 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**

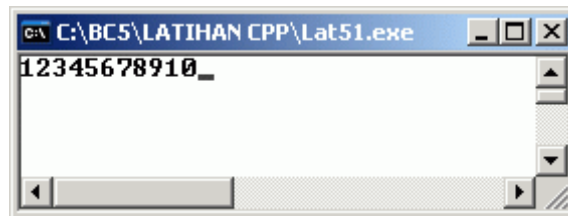
```

/* ----- */
/* 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**

```

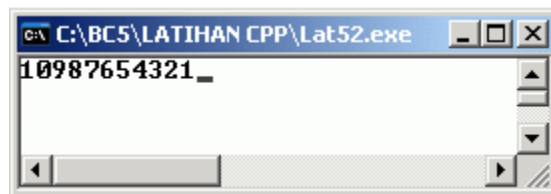
/* ----- */
/* Program for - bilangan turun */
/* ----- */
include <stdio.h>
include <conio.h>
include <iostream.h>

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

**Contoh-4**

```

/* ----- */
/* Program Menampilkan Warna - 1 */
/* ----- */

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

main()
{
 int a=2, b=1, c=2, d=1, e;

 clrscr();

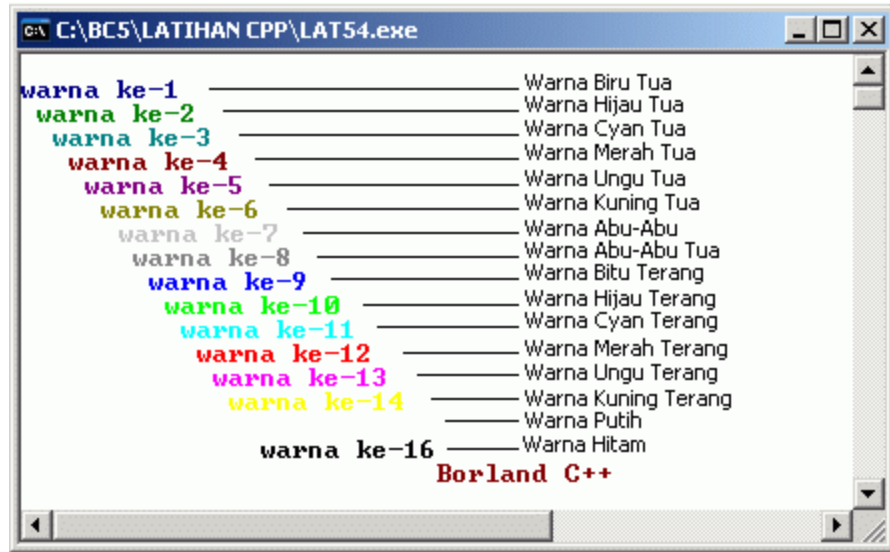
 for(e=1; 17>e; e++)
 {
 gotoxy(e, e); textcolor(e);
 cprintf("\nwarna ke-%d",e);
 }

 textcolor(4+BLINK); cprintf("Borland C++");

 getche();
}

```

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



Gambar 5.4. Hasil Contoh-4

**Contoh-5**

```

/* ----- */
/* Program Menampilkan Warna - 2 */
/* ----- */

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

int main(void)
{
 int i;

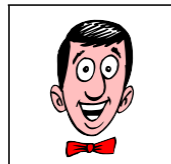
 clrscr();

 for (i=0; i<20; i++)
 {
 textattr(i + ((i+1) << 4));
 cprintf("Borland C++\r\n");
 }

 getch();

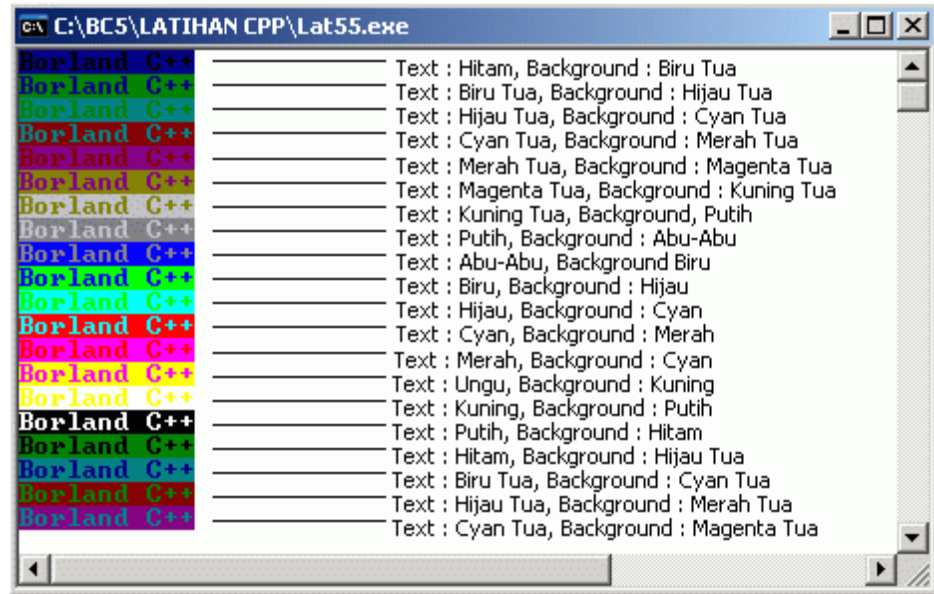
 return 0;
}

```



- **textattr()** = Menentukan warna tulisan dan warna latar belakang berdasarkan ketentuan angka 8 bit biner, penataan warna
- **cprintf()** = Menampilkan data dengan format pewarnaan teks dan pewarnaan latar belakang.

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



Gambar 5.5. Hasil Contoh-5

### 5.1.1. Pernyataan nested - for

#### Penjelasan

Pernyataaan 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 didalam terlebih dahulu dihitung hingga selesai, kemudian perulangan yang diluar diselesaikan.

#### Contoh-6

```
/* ----- */
/* Program for - Nested for */
/* ----- */

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

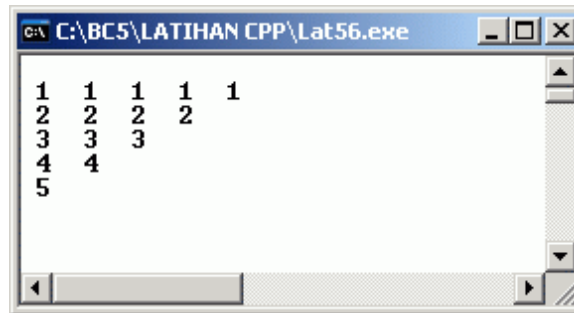
main()
{
 int a, b;
 clrscr();
```

```

for(a = 1; a <= 5; a++)
{
 printf("\n");
 for(b = a; b <= 5; b++)
 cout<<a<<" ";
}
getch();
}

```

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



Gambar 5.6. Hasil Contoh-6

### 5.1.2. Perulangan Tidak Berhingga

#### Penjelasan

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-7

```

/* ----- */
/* Program for Tdk Berhingga */
/* ----- */
#include<stdio.h>
#include<conio.h>
#include <iostream.h>

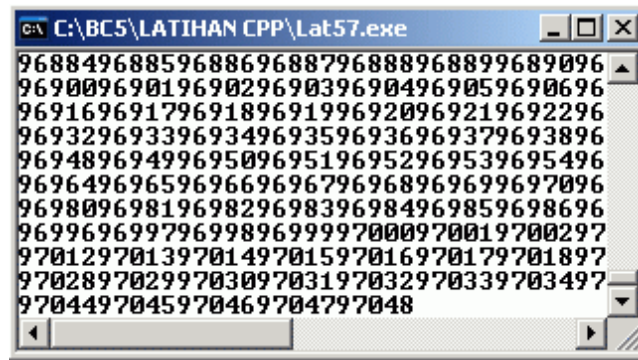
main()
{
 int bil;
 clrscr();
 for (bil = 60; bil >=10; bil++)
 cout<<bil<<" ";

 getch();
}

```

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





Gambar 5.7. Hasil Contoh-7

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 >= 1` selalu bernilai benar ( karena bil bernilai 6), maka pernyataan

```
cout<<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**, dapat juga langsung **klik Close Button**

## 5.2. Pernyataan goto

### Penjelasan

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

```
goto label;
```

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

### Contoh-8

```
/* ----- */
/* Program dengan pernyataan goto */
/* ----- */
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
main()
{
 int a, b;
 char lagi;
```

```

atas:
 clrscr();
 cout>>"Masukkan Bilangan = ";
 cin<<a;

 b = a % 2;

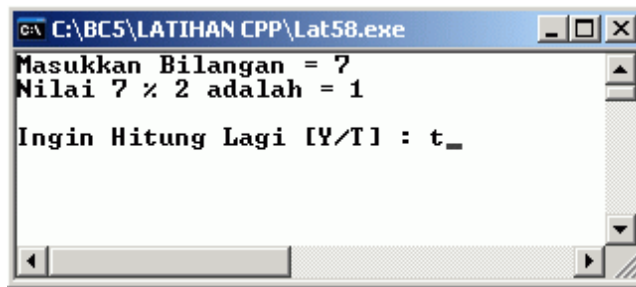
 printf("Nilai %d %% 2 adalah = %d",a, b);
 printf("\n\nIngin Hitung Lagi [Y/T] : ");
 lagi = getche() ;

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

 getch();
}

```

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



Gambar 5.8. Hasil Contoh-8

### 5.3. Pernyataan while

#### Penjelasan

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 ;
}

```

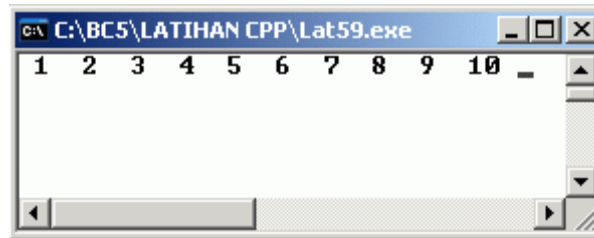
**Contoh-9**

```

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

```

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



Gambar 5.9. Hasil Contoh-9

**Contoh-10**

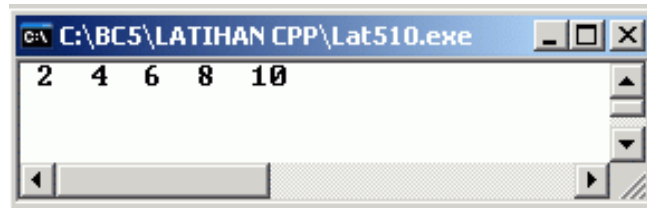
```

/* ----- */
/* Program while02.cpp */
/* ----- */
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

main()
{
 int bil=2;
 clrscr();
 while(bil<=10)
 {
 cout<<bil<<" ";
 bil+=2;
 }
 getch();
}

```

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



Gambar 5.10. Hasil Contoh-10

## 5.4. Pernyataan do - while

### Penjelasan

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);
```

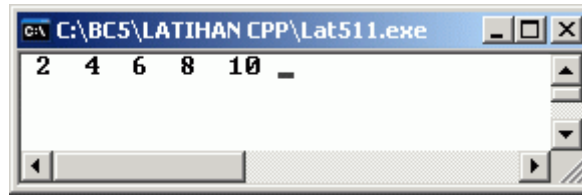
Bentuk umum perulangan **do - while**, dengan lebih dari perintah / pernyataan, sebagai berikut :

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

### Contoh-11

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

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



Gambar 5.11. Hasil Contoh-11

## 5.5. Pernyataan break

### Penjelasan

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-12

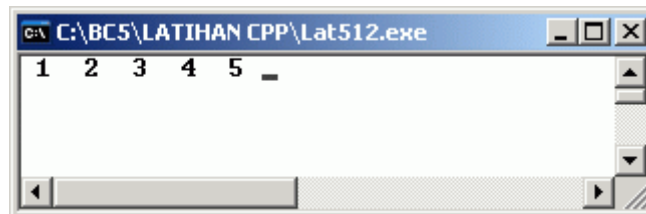
```
/* ----- */
/* Program do - while dengan break */
/* ----- */

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

 do
 {
 if (bil >= 6)
 break;
 cout<<bil<<" ";
 }
 while(bil++);

 getch();
}
```

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



Gambar 5.12. Hasil Contoh-12

**Contoh-13**

```

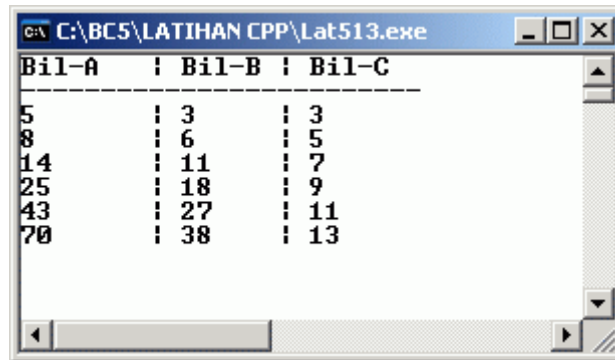
/* ----- */
/* Perulangan FOR dengan break; */
/* ----- */

#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;
 }
 getch();
}

```

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



Gambar 5.13. Hasil Contoh-13

## 5.6. Pernyataan continue

### Penjelasan

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-14**

```

/* ----- */
/* Perulangan FOR dengan continue */
/* ----- */

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

```

```

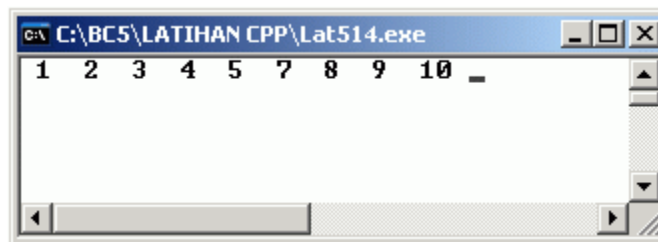
main()
{
 int bil;
 clrscr();

 for(bil=1; bil<=10; ++bil)
 {
 if(bil==6)
 continue;

 cout<<bil<<" ";
 }
 getch();
}

```

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



Gambar 5.14. Hasil Contoh-14

## 5.7. Latihan

### Penjelasan

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 membentuk segitiga siku dengan hasilnya :

$$\begin{array}{rcl}
 1 & & = 1 \\
 1 + 2 & & = 3 \\
 1 + 2 + 3 & & = 6 \\
 1 + 2 + 3 + 4 & & = 10 \\
 1 + 2 + 3 + 4 + 5 & & = 15
 \end{array}$$

4. Buatlah program untuk menghitung perkalian deret bilangan ganjil membentuk segitiga siku dengan hasilnya :

```

1 = 1
1 * 3 = 3
1 * 3 * 5 = 15
1 * 3 * 5 * 7 = 105
1 * 3 * 5 * 7 * 9 = 945

```

5. Buatlah program untuk menghitung perkalian deret bilangan genap membentuk segitiga siku terbalik dengan hasilnya :

```

10 + 8 + 6 + 4 + 2 = 30
10 + 8 + 6 + 4 = 28
10 + 8 + 6 = 24
10 + 8 = 18
10 = 10
 ----- +
 110

```

6. Buatlah program untuk menghitung perkalian nilai kolom dengan baris berikut tampilan keluaran yang diinginkan :

```

1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 10 16 20
5 10 15 20 25

```



## Pemrograman C++

# Bab 6 : Operasi String

**Penjelasan** Operasi string selalu dijumpai didalam bahasa pemrograman, disini akan dibahas beberapa perintah dan fungsi string.

## 6.1. Fungsi Manipulasi String

**Penjelasan** Borland C++ menyediakan beberapa fungsi yang digunakan untuk keperluan manipulasi string.

### 6.1.1. Fungsi strcat()

**Penjelasan** 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);
```

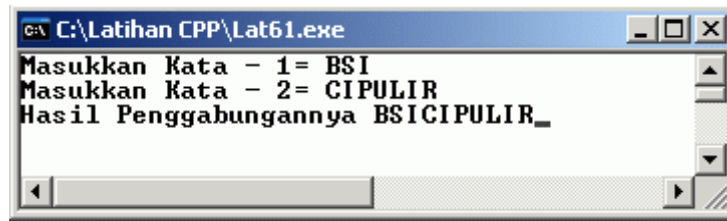
**Contoh-1**

```
#include <stdio.h>
#include <conio.h>
#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-1 diatas adalah :



Gambar 6.1. Hasil Contoh-1

## 6.1.2. Fungsi strcmp()

### Penjelasan

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 :

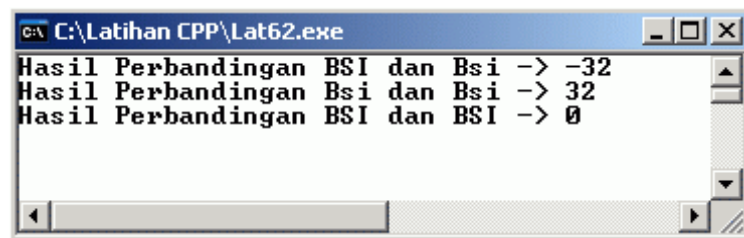
```
var_int = strcmp(str1, str2);
```

### Contoh -2

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <iostream.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-2 diatas adalah :



Gambar 6.2. Hasil Contoh-2

### 6.1.3. Fungsi strcpy()

#### Penjelasan

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

Bentuk Penulisan :

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

#### Contoh-3

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

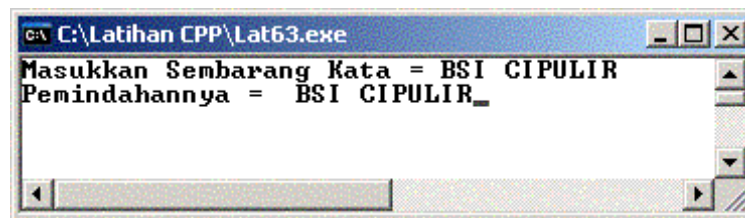
main()
{
 char huruf[20];
 char pindah[20];
 clrscr();

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

 /* Proses */
 strcpy(pindah, huruf);

 cout<<"Pemindahannya = "<<pindah;
 getch();
}
```

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



Gambar 6.3. Hasil Contoh-3

### 6.1.4. Fungsi strlen()

#### Penjelasan

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

Bentuk Penulisan :

```
strlen(str) ;
```

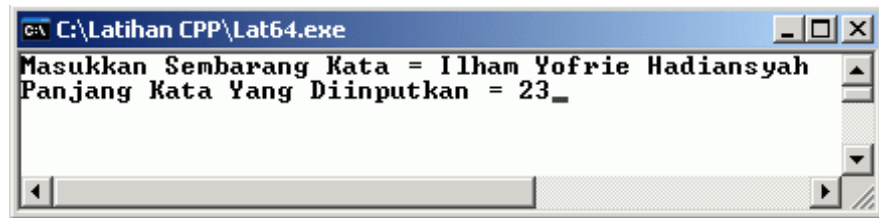
**Contoh-4**

```
#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-4 diatas adalah :



Gambar 6.4. Hasil Contoh-4

### 6.1.5. Fungsi strrev()

**Penjelasan**

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-5**

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

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

```

clrscr();

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

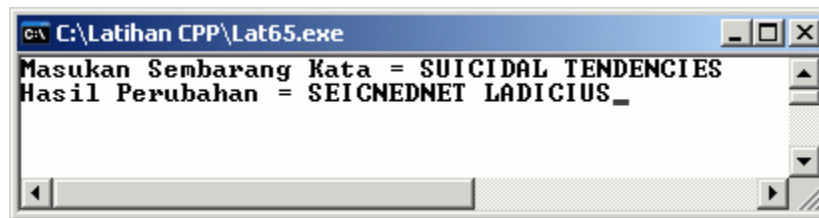
strrev(kata);

cout<<"Hasil Perubahan = "<<kata;

getch();
}

```

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



Gambar 6.5. Hasil Contoh-5

## 6.2. Fungsi Konfersi String

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

### 6.2.1. Fungsi atof()

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

#### Contoh-6

```

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

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

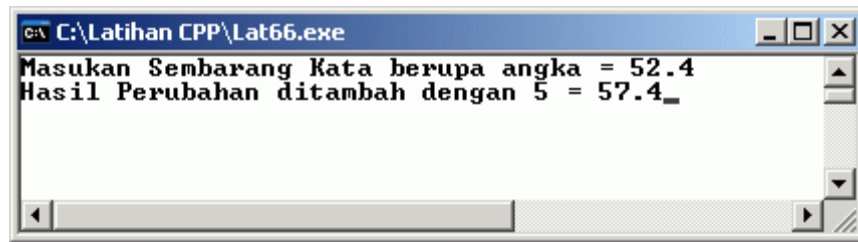
 cout<<"Masukan Sembarang Kata berupa angka = ";
 gets(kata);

 angka = atof(kata);
 a = angka + 5;

 cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;
 getch();
}

```

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



Gambar 6.6. Hasil Contoh-6

## 6.2.2. Fungsi atoi()

### Penjelasan

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

### Contoh-7

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

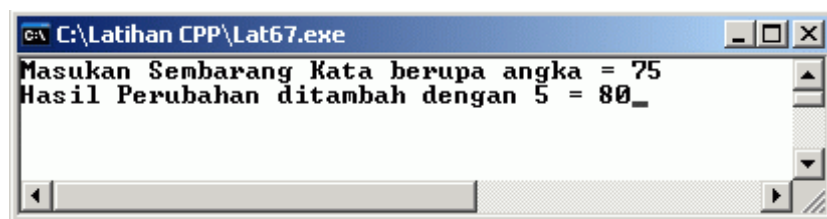
main()
{
 char kata[20];
 float angka, a, b;
 clrscr();

 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-7 diatas adalah :



Gambar 6.7. Hasil Contoh-7

### 6.2.3. Fungsi atol()

#### Penjelasan

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

#### Contoh-8

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

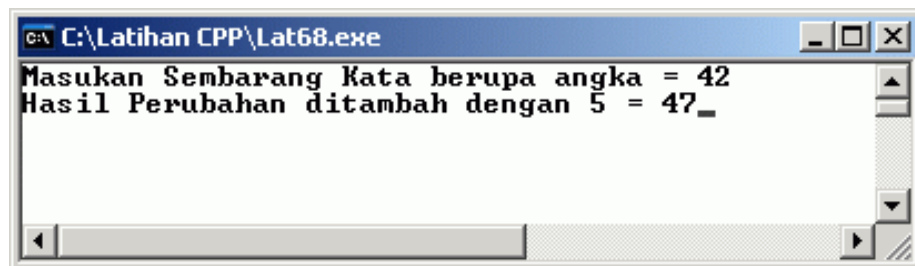
main()
{
 char kata[20];
 float angka, a, b;
 clrscr();

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

 angka = atol(kata);
 a = angka + 5;

 cout<<"Hasil Perubahan ditambah dengan 5 = "<<a;
 getch();
}
```

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



Gambar 6.8. Hasil Contoh-8

### 6.2.4. Fungsi strlwr()

#### Penjelasan

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-9**

```

#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-9 diatas adalah :



Gambar 6.9. Hasil Contoh-9

## 6.2.5. Fungsi strupr()

**Penjelasan**

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-10**

```

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

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

```



```

 cout<<"Masukan Sembarang Kata dengan Huruf Kecil=";
 gets(kata);
 strupr(kata);

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

```

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



Gambar 6.10. Hasil Contoh-10

## 6.3. Latihan

### Penjelasan

Buatlah beberapa program seperti petunjuk berikut :

1. Buatlah program untuk menghitung panjang kata berikut ini :

```

Akademi Manajemen Informatika dan Komputer Bina
Sarana Informatika

```

2. Buatlah program untuk membalik kata berikut ini :

```

Akademi Manajemen Informatika dan Komputer Bina
Sarana Informatika

```

Menjadi seperti berikut :

```

akitamrofni anaraS aniB retupmoK and akitamrofniI
nemejanaM imedakA

```

3. Buatlah program untuk menggabungkan dua buah string

```

Kalimat1 = Manajemen
Kalimat2 = Informatika

```

Menjadi seperti berikut :

```

ManajemenInformatika

```

4. 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

## Pemrograman C++

# Bab 7 : Array

### Penjelasan

Variabel Larik atau lebih dikenal dengan ARRAY adalah adalah Tipe terstruktur yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe yang 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 tiga buah dimensi :

- Array Berdimensi Satu.
- Array Berdimensi Dua
- Array Berdimensi Dua

## 7.1. Array Berdimensi Satu

### Penjelasan

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.

### Deklarasi Array

Bentuk Umum pendeklarasian array :

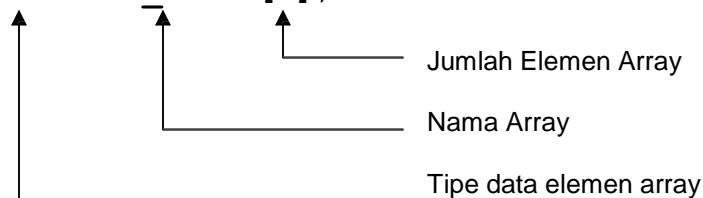
**Tipe-Data Nama\_Variabel[Ukuran]**

Keterangan :

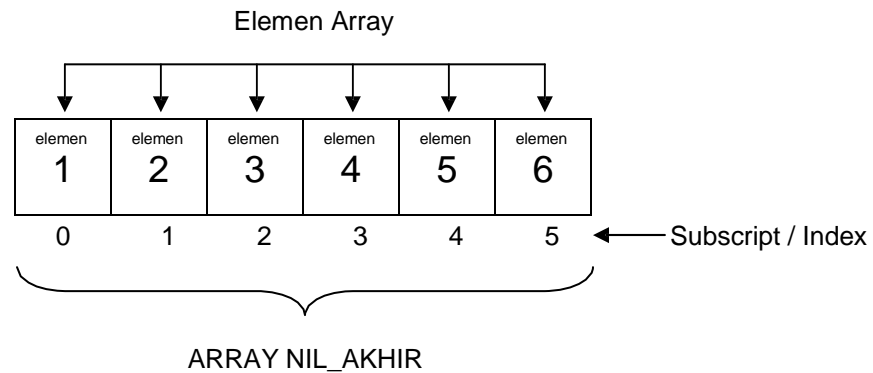
- Type Data : Untuk menyatakan type data yang digunakan.
- Ukuran : Untuk menyatakan jumlah maksimum elemen array.

### 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 )

### 7.1.1. Mengakses Array Berdimensi Satu

#### Penjelasan

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-1

```
/* ----- */
/* 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<<"-----";
cout<<"-----"<<endl;
cout<<"No. Nama Siswa Nilai Nilai ";
cout<<"Hasil"<<endl;
cout<<" MidTest Final ";
cout<<"Ujian"<<endl;
cout<<"-----";
cout<<"-----"<<endl;

for(i=1;i<=2;i++)
{
 cout<<setiosflags(ios::left)<<setw(4)<<i;
 cout<<setiosflags(ios::left)<<setw(20)<<nama[i];
 cout<<setprecision(2)<<" "<<nilai1[i];
 cout<<setprecision(2)<<" "<<nilai2[i];
 cout<<setprecision(2)<<" "<<hasil[i]<<endl;
}

cout<<"-----";
cout<<"-----"<<endl;
getch();
}

```

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

```

C:\BC5\LATIHAN CPP\Lat71.exe
Data Ke - 1
Nama Siswa : ILHAM
Nilai MidTest : 78
Nilai Final : 76

Data Ke - 2
Nama Siswa : HADIANSYAH
Nilai MidTest : 86
Nilai Final : 84

No. Nama Siswa Nilai Nilai Hasil
Hasil MidTest Final Ujian

1 ILHAM 78 76 77
2 HADIANSYAH 86 84 85

```

Gambar 7.1. Hasil Contoh-1

## 7.1.2. Inisialisasi Array Berdimensi Satu

**Penjelasan** 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** `float nilai[5] = {56.5, 66.7, 87.45, 98.5, 78.9 };`

**Contoh-2**

```
/* ----- */
/* Inisialisasi Array Dimensi 1 */
/* ----- */

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

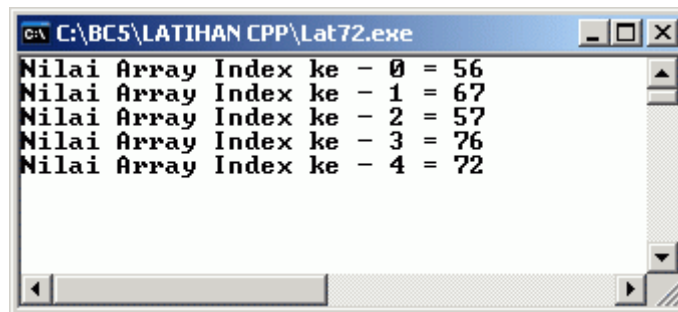
void main()
{
 int nilai[5] = {56, 67, 57, 76, 72};
 int i;

 clrscr();

 for(i=0; i<5; i++)
 {
 cout<<"Nilai Array Index ke - "<<i<<" = ";
 cout<<a<<endl;
 }

 getch();
}
```

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



Gambar 7.2. Hasil Contoh-2

## 7.2. Array Berdimensi Dua

### Penjelasan

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 pendatan penjualan, pendataan nilai dan lain sebagainya.

### Deklarasi Array

Bentuk Umum pendeklarasian array :

**Type-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

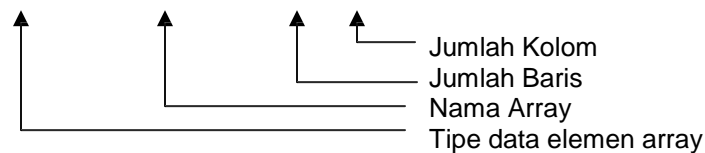
Sebagai contoh pendeklarasian yang akan kita gunakan adalah pengolahan data penjualan, berikut dapat anda lihat pada tabel berikut :

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

Tabel 7.1. Tabel Data Penjualan Tahun

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

```
int data_jual[3][3];
```



## 7.2.1. Mengakses Array Berdimensi Dua

### Penjelasan

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

```
Nama_Array[index-1][index-2]
```

### Contoh

```
data_jual[2][2];
data_jual[1][2];
```

### Contoh-3

```
/* ----- */
/* 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();

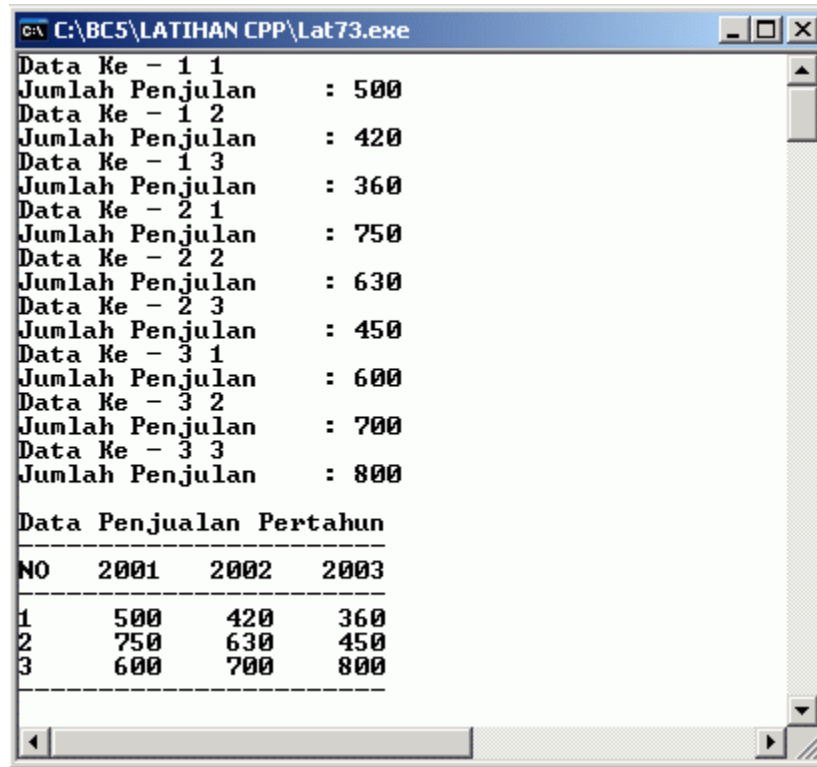
 for(i=1;i<=3;i++)
 {
 for(j=1;j<=3;j++)
 {
 cout<<"Data Ke - "<<i<<" "<<j<<endl;
 cout<<"Jumlah Penjualan : ";
 cin>>data_jual[i][j];
 }
 }

 cout<<"Data Penjualan Pertahun"<<endl;
 cout<<"-----"<<endl;
 cout<<"NO 2001 2002 2003"<<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-3 diatas adalah :



```

C:\BC5\LATIHAN CPP\Lat73.exe
Data Ke - 1 1
Jumlah Penjualan : 500
Data Ke - 1 2
Jumlah Penjualan : 420
Data Ke - 1 3
Jumlah Penjualan : 360
Data Ke - 2 1
Jumlah Penjualan : 750
Data Ke - 2 2
Jumlah Penjualan : 630
Data Ke - 2 3
Jumlah Penjualan : 450
Data Ke - 3 1
Jumlah Penjualan : 600
Data Ke - 3 2
Jumlah Penjualan : 700
Data Ke - 3 3
Jumlah Penjualan : 800

Data Penjualan Pertahun

NO 2001 2002 2003

1 500 420 360
2 750 630 450
3 600 700 800

```

Gambar 7.3. Hasil Contoh-3

## 7.2.2. Inisialisasi Array Berdimensi Dua

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

```
Tipe_data nama_array[jml_element] = { nilai array };
```

**Contoh**

```
int data[2][5] = { {2, 3, 4, 5, 2},
 {4, 2, 6, 2, 7},
 };
```

**Contoh-4**

```
/* ----- */
/* Inisialisasi Array Dimensi 2 */
/* ----- */

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

void main()
```

```

{
 int i, j;
 int data[2][5] = {{2, 3, 4, 5, 2},
 {4, 2, 6, 2, 7}};

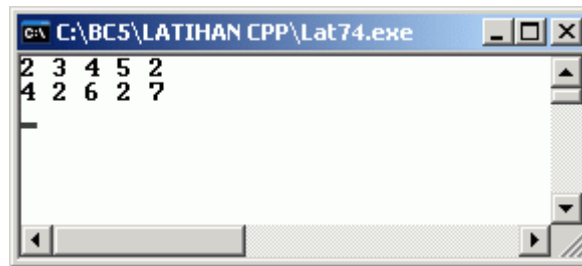
 clrscr();

 for(i=0;i<2;i++)
 {
 for(j=0;j<5;j++)
 {
 cout<<data[i][j];
 cout<<" ";
 }
 cout<<endl;
 }

 getch();
}

```

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



Gambar 7.4. Hasil Contoh-4

## 7.3. Array Berdimensi Tiga

### Penjelasan

Array dimensi dua tersusun dalam bentuk baris, kolom dan isi dari baris, dimana indeks pertama menunjukkan baris, indeks kedua menunjukkan kolom dan indeks ketiga menunjukkan isi dari baris.

### Deklarasi Array

Bentuk Umum pendeklarasian array :

**Type-Data Nama\_Variabel[index-1][index-2][index-3]**

Keterangan :

- Type Data : Untuk menyatakan type data yang digunakan.
- Index-1 : Untuk menyatakan jumlah baris
- Index-2 : Untuk menyatakan jumlah isi dari baris
- Index-3 : 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**

|               | Tahun Ke | Hasil Ke | Tahun Penjualan Ke. |     |
|---------------|----------|----------|---------------------|-----|
|               |          |          | 1                   | 2   |
| Indeks ke.1 { | 1        | 1        | 150                 | 159 |
|               |          | 2        | 200                 | 400 |
| {             | 2        | 1        | 100                 | 125 |
|               |          | 2        | 210                 | 125 |

Indeks ke.3

Tabel 7.2. Tabel Data Penjualan Pertahun

Jika anda lihat dari tabel 7.2 diatas maka dapat dituliskan kedalam array dimensi dua berikut :

```
int data_jualan[2][2][2];
```

Jumlah Kolom  
Jumlah Isi Baris  
Jumlah Baris  
Nama Array  
Tipe data elemen array

### 7.3.1. Mengakses Array Berdimensi Tiga

#### Penjelasan

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

**Nama\_Array[index-1][index-2][index-3]**

#### Contoh

```
data_jualan[1][1][1];
data_jualan[1][0][1];
```

#### Contoh-5

```
/* ----- */
/* Array Dimensi 3 */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>
#include<iomanip.h>

main()
{
 int i, j, k;
```

```

int data_jual[2][3][2];

clrscr();

for(i=0;i<2;i++)
{
 for(j=0;j<3;j++)
 {
 for(k=0;k<2;k++)
 {
 cout<<"Data Tahun Ke - "<<i+1<<endl;
 cout<<"Data Ke - "<<j+1<<" "<<k+1<<endl;
 cout<<"Jumlah Penjualan : ";
 cin>>data_jual[i][j][k];
 }
 cout<<endl;
 }
 cout<<endl;
}

cout<<endl;
cout<<"Data Penjualan Pertahun"<<endl;
cout<<"-----";
cout<<endl;
cout<<"Tahun Hasil Tahun Penjualan Ke. ";
cout<<endl;
cout<<"Ke. Ke. -----";
cout<<endl;
cout<<" 1 2 ";
cout<<endl;
cout<<"-----";
cout<<endl;

for(i=0;i<2;i++)
{
 for(j=0;j<3;j++)
 {
 cout<<setiosflags(ios::left)<<setw(11)<<i+1;
 cout<<setiosflags(ios::left)<<setw(9)<<j+1;
 for(k=0;k<2;k++)
 {
 cout<<setiosflags(ios::right)<<setw(5);
 cout<<data_jual[i][j][k];
 cout<<" ";
 }
 cout<<endl;
 }
 cout<<endl;
}

cout<<"-----";
cout<<endl;

getch();
}

```

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

```

C:\BC5\LATIHAN CPP\LAT75.exe
Data Tahun Ke - 1
Data Ke - 1 1
Jumlah Penjualan : 100
Data Tahun Ke - 1
Data Ke - 1 2
Jumlah Penjualan : 200

Data Tahun Ke - 1
Data Ke - 2 1
Jumlah Penjualan : 300
Data Tahun Ke - 1
Data Ke - 2 2
Jumlah Penjualan : 400

Data Tahun Ke - 1
Data Ke - 3 1
Jumlah Penjualan : 500
Data Tahun Ke - 1
Data Ke - 3 2
Jumlah Penjualan : 600

Data Tahun Ke - 2
Data Ke - 1 1
Jumlah Penjualan : 700
Data Tahun Ke - 2
Data Ke - 1 2
Jumlah Penjualan : 800

Data Tahun Ke - 2
Data Ke - 2 1
Jumlah Penjualan : 900
Data Tahun Ke - 2
Data Ke - 2 2
Jumlah Penjualan : 850

Data Tahun Ke - 2
Data Ke - 3 1
Jumlah Penjualan : 750
Data Tahun Ke - 2
Data Ke - 3 2
Jumlah Penjualan : 650

Data Penjualan Pertahun

Tahun Hasil Tahun Penjualan Ke.
Ke. Ke. 1 2

1 1 100 200
1 2 300 400
1 3 500 600

2 1 700 800
2 2 900 850
2 3 750 650

```

Gambar 7.5. Hasil Contoh-5

## 7.3.2. Inisialisasi Array Berdimensi Tiga

### Penjelasan

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

```
Tipe_data nama_array[jml_element] = { nilai array };
```

### Contoh

```
float data[2][4][3] = {
 { {100, 200, 300},
 {150, 240, 360},
 {250, 340, 460},
 {250, 340, 460}},
 { {160, 250, 365},
 {175, 275, 375},
 {275, 375, 575},
 {280, 380, 580}}
};
```

### Contoh-6

```
/* ----- */
/* Inisialisasi Array Dimensi 3 */
/* ----- */

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

main()
{
 int i, j, k;
 float data[2][4][3] = {
 { {100, 200, 300},
 {150, 240, 360},
 {250, 340, 460},
 {250, 340, 460}},
 { {160, 250, 365},
 {175, 275, 375},
 {275, 375, 575},
 {380, 480, 580}}
 };

 clrscr();

 cout<<"-----"
 cout<<endl;
 cout<<"Tahun Hasil Tahun Penjualan "
 cout<<endl;
 cout<<"Ke. Ke. -----"
 cout<<endl;
 cout<<" 2002 2003 2004 "
 cout<<endl;
 cout<<"-----"
 cout<<endl;
```

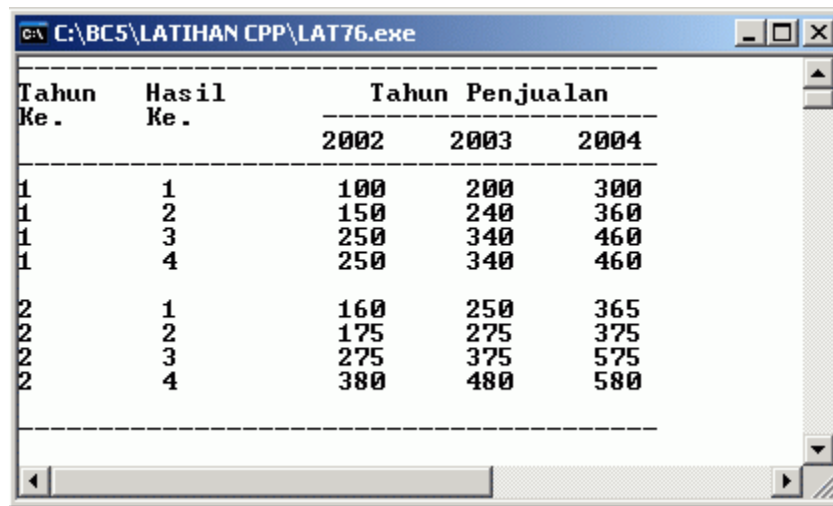
```

for(i=0;i<2;i++)
{
 for(j=0;j<4;j++)
 {
 cout<<setiosflags(ios::left)<<setw(9)<<i+1;
 cout<<setiosflags(ios::left)<<setw(9)<<j+1;
 for(k=0;k<3;k++)
 {
 cout<<setiosflags(ios::right)<<setw(5);
 cout<<data[i][j][k];
 cout<<" ";
 }
 cout<<endl;
 }
 cout<<endl;
}
cout<<"-----"
cout <<endl;

getch();
}

```

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



| Tahun<br>Ke . | Hasil<br>Ke . | Tahun Penjualan |      |      |
|---------------|---------------|-----------------|------|------|
|               |               | 2002            | 2003 | 2004 |
| 1             | 1             | 100             | 200  | 300  |
| 1             | 2             | 150             | 240  | 360  |
| 1             | 3             | 250             | 340  | 460  |
| 1             | 4             | 250             | 340  | 460  |
| 2             | 1             | 160             | 250  | 365  |
| 2             | 2             | 175             | 275  | 375  |
| 2             | 3             | 275             | 375  | 575  |
| 2             | 4             | 380             | 480  | 580  |

Gambar 7.6. Hasil Contoh-6

## 7.4. Latihan

### Penjelasan No. 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 | Jenis | 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 | Jenis | Harga    |
|------|-------|----------|
| D    | Dada  | Rp. 2500 |
| P    | Paha  | Rp. 2000 |
| S    | Sayap | Rp. 1500 |

Banyak Jenis : ... <diinput>

Jenis Ke - ... <proses counter>

Jenis Potong [D/P/S] : ... <diinput>

Banyak Potong : ... <diinput>

<<Terus berulang tergantung Banyak Jenis>>

#### Layar Keluaran

| GEROBAK FIRED CHICHEN |              |              |            |              |
|-----------------------|--------------|--------------|------------|--------------|
| No.                   | Jenis Potong | Harga Satuan | Bayak Beli | Jumlah Harga |
| ...                   | ....         | ....         | ....       | Rp ....      |
| ...                   | ....         | ....         | ....       | Rp ....      |
| Jumlah Bayar          |              |              |            | Rp ....      |
| Pajak 10%             |              |              |            | Rp ....      |
| Total Bayar           |              |              |            | Rp ....      |



**Penjelasan No. 2**

Buatlah program untuk menghitung nilai akhir seorang siswa dari kursus yang diikutinya. Dengan ketentuan sebagai berikut :

- Nama Mahasiswa, Nilai Tugas, Nilai UTS dan Nilai UAS diinput.
- Proses yang dilakukan untuk mendapatkan nilai murni dari masing-masing nilai, adalah
  - Nilai Murni Tugas = Nilai Tugas dikalikan dengan 30%
  - Nilai Murni UTS = Nilai UTS dikalikan dengan 30%
  - Nilai Murni UAS = Nilai UAS dikalikan dengan 40%
  - Nilai Akhir adalah Nilai Murni Tugas + Nilai Murni UTS + Nilai Murni UAS
- Ketentuan untuk mendapatkan grade nilai :
  - Nilai Akhir  $\geq 80$  mendapat Grade A
  - Nilai Akhir  $\geq 70$  mendapat Grade B
  - Nilai Akhir  $\geq 59$  mendapat Grade C
  - Nilai Akhir  $\geq 50$  mendapat Grade D
  - Nilai Akhir  $< 50$  mendapat Grade E
- Tampilan yang diinginkan sebagai berikut :

**Layar Masukkan**

```

PROGRAM HITUNG NILAI AKHIR
MATERI PEMROGRAMMAN C++

Masukkan Jumlah Mahasiswa : ... <diinput>

Mahasiswa Ke - ... <proses counter>
Nama Mahasiswa : <diinput>
Nilai Tugas : <diinput>
Nilai UTS : <diinput>
Nilai UAS : <diinput>

<<Terus berulang tergantung Jumlah Mahasiswa>>

```

**Layar Keluaran**

```

DAFTAR NILAI
MATERI : PEMROGRAMMAN C++

No. Nama Nilai Grade
 Mahasiswa Tugas UAS UAS Akhir

...
...

```

**Penjelasan No. 3** 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%        |

Sebagai contoh : Jika seorang karyawan tersebut dengan golongan 3, maka mendapatkan tunjangan sebesar  $15\% \times \text{Rp. } 700,000,-$

- Tunjangan Pendidikan

| Kode | Pendidikan | Persentase |
|------|------------|------------|
| 1    | SMU        | 2,5%       |
| 2    | D3         | 5%         |
| 3    | S1         | 7,5%       |

- Honor Lembur

Jumlah jam kerja normal dalam satu bulan sebanyak 240 Jam Kerja. Honor lembur diberikan jika jumlah jam kerja lebih 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.

- 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>
```

```
Pendidikan (1=SMU/2=D3/3=S1) : ... <di input>
```

```
Jumlah Jam Kerja : ... <di input>
```

```
<<Terus berulang tergantung Jumlah Karyawan>>
```

```
PT. STAY COOL
```

```

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

...
...

Total Gaji yang dikeluarkan Rp.
```

## Pemrogramman C++

# Bab 8 : Pointer

**Penjelasan** 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.

## 8.1. Operator Pointer

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

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

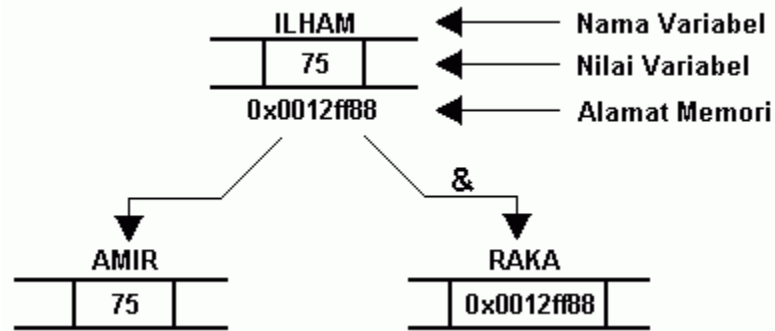
### 8.1.1. Operator Dereference ( `&` )

**Penjelasan** 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 identifier "**&**" (**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 8.1. Diagram Penggunaan Opeator Dereference

### 8.1.2. Operator Reference ( \* )

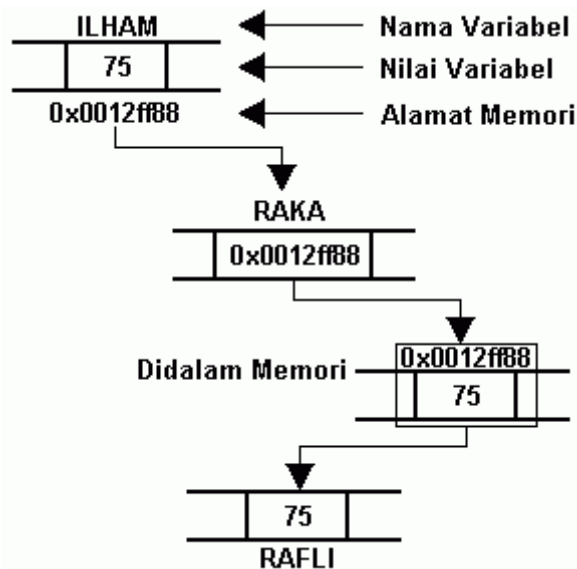
Penjelasan

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 8.2 Diagram Penggunaan Opeator Reference

## 8.2. Deklarasi Pointer Pada Konstanta

### Penjelasan

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;

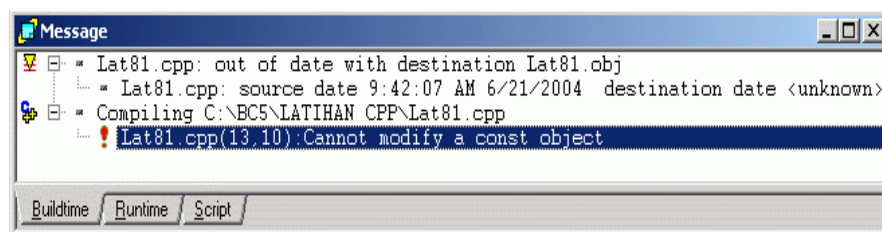
 nama = "Visual C++";

 cout<<"Nama Program = "<<nama<<endl;

 getch();
}
```

|                                                                                     |                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>Pada program diatas, terdapat kesalahan dan tidak dapat dijalankan, penyebabnya pada pernyataan <code>nama = "Visual C++";</code>. Karena variabel <code>nama</code>, merupakan merupakan pointer konstanta, yaitu tidak dapat diubah-ubah. Pesan Kesalahan Yang Tampil adalah : <b>Cannot modify a const object</b></p> |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

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



Gambar 8.3 Error Message Contoh-1

## 8.3. Deklarasi Pointer Pada Variabel

### Penjelasan

Karena keakhlian 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_konstanta;
```

### Contoh-2

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

main()
{
 int ilham, amir, *raka;

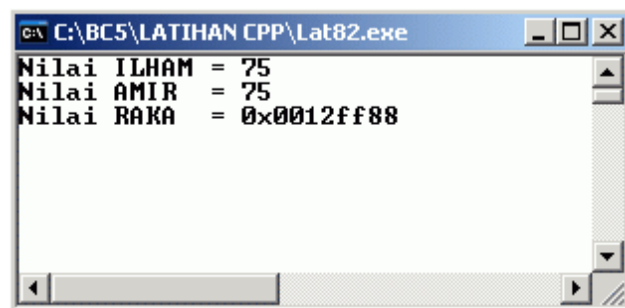
 clrscr();

 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 8.4 Hasil Contoh-2

**Contoh-3**

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

main()
{
 int ilham, *raka, rafli;

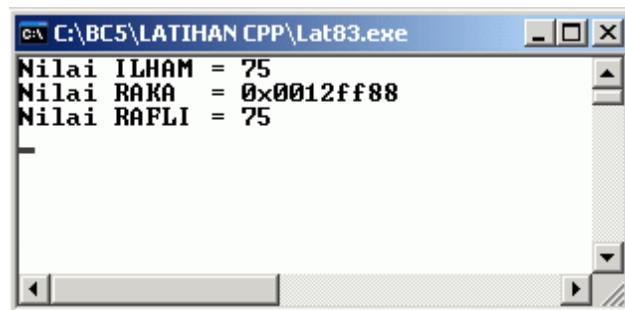
 clrscr();

 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 8.5 Hasil Contoh-3

**Contoh-4**

```
/* ----- */
/* Perubahan Nilai dengan Pointer */
/* ----- */

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

main()
{
 int yofrie = 93;
 int *hadiansyah ;

 clrscr();

 cout<<"Nilai awal yofrie = "<<yofrie<<endl;

 hadiansyah = &yofrie;
```

```

 cout<<"Nilai hadiansyah sekarang = ";
 cout<<*hadiansyah<<endl;

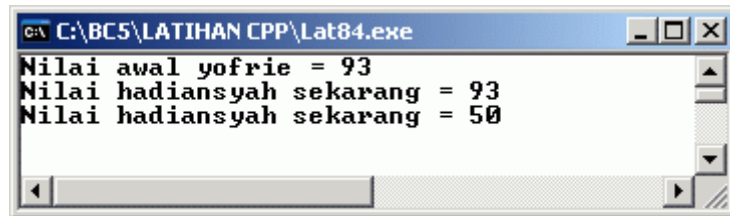
 *hadiansyah = 50;

 cout<<"Nilai hadiansyah sekarang = ";
 cout<<*hadiansyah<<endl;

 getch();
}

```

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



Gambar 8.6 Hasil Contoh-4

## 8.4. Pointer Pada Pointer

### Penjelasan

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 8.7 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

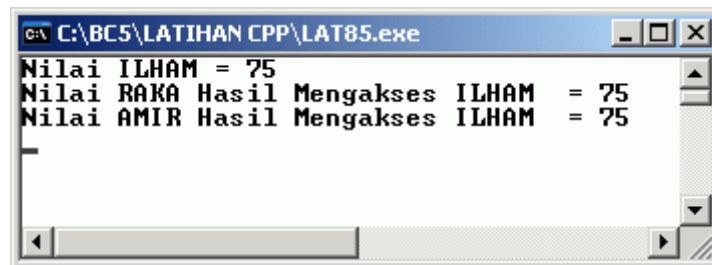
 raka = &ilham;
 amir = &raka;

 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 8.8 Hasil Contoh-5

## 8.5. Pointer pada Array

### Penjelasan

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;

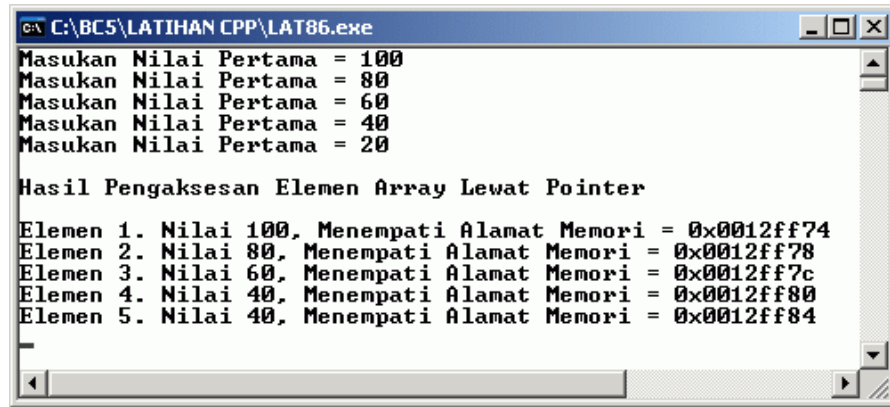
 ptrnilai = nilai;

 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 8.9 Hasil Contoh-6

## 8.6. Pointer pada String

### Penjelasan

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 :



```
C:\BC5\LATIHAN CPP\LAT87.exe
Nama Band Metal = SEPULTURA
Nama Band Punk = RANCID
Nama Band Punk = CID
```

Gambar 8.10 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.

|   |   |   |   |   |   |    |
|---|---|---|---|---|---|----|
| R | A | N | C | I | D | \0 |
|   |   |   | K |   |   |    |
| 1 | 2 | 3 | 4 | 5 | 6 |    |

## Pemrograman C++

# Bab 9 : Fungsi

### Penjelasan

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

### Penjelasan

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() */

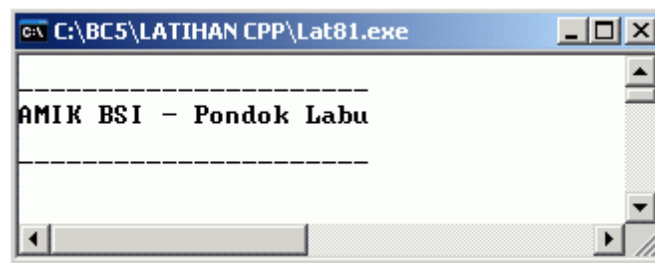
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

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 Fungsi

**Penjelasan**

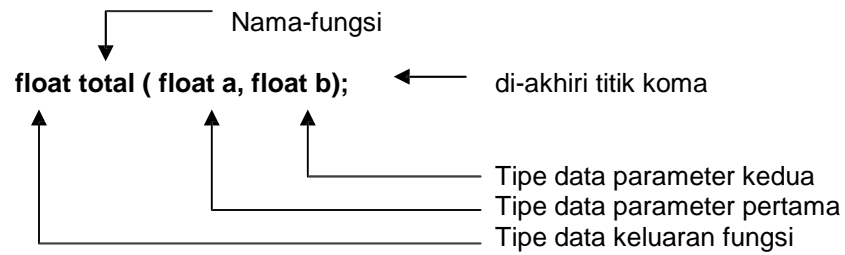
Prototipe fungsi digunakan untuk mendeklarasikan ke kompilasi mengenai:

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

Keuntungan didalam pemakai prototipe yaitu :

- Kompilasi 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,

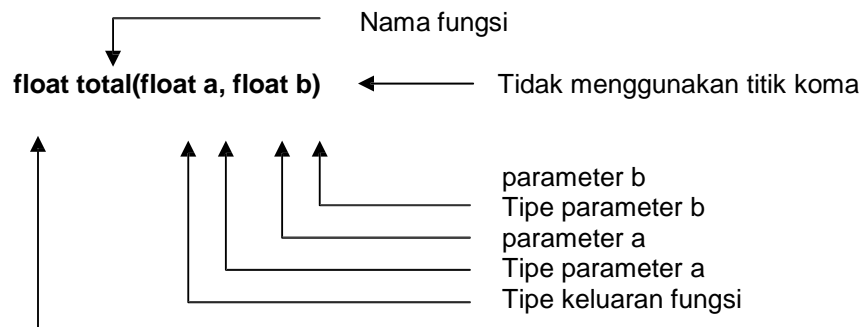
Contoh prototipe fungsi :



Bentuk definisi dalam penggunaan fungsi yang dideklarasikan dengan menggunakan prototipe, harus diubah. Sebagai contoh pada pendefinisian berikut :

```
float total(a, b)
float a, b;
```

Bentuk pendefinisian diatas harus diubah menjadi bentuk modern pendefinisian fungsi :



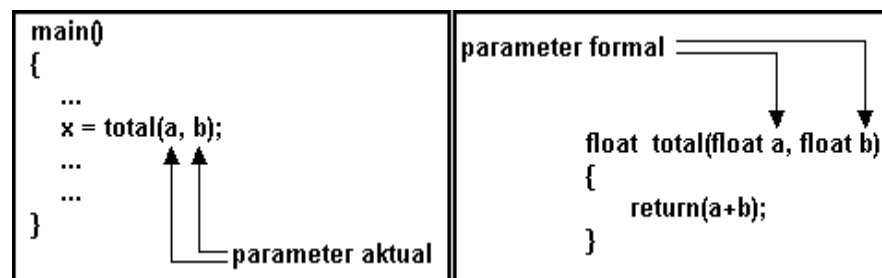
### 9.3. Parameter Fungsi

#### Penjelasan

Terdapat dua macam 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.



Ada dua cara untuk melewati parameter ke dalam fungsi, yaitu berupa :

### 9.3.1. Pemanggilan dengan nilai ( *Call by Value* )

#### Penjelasan

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

#### Contoh-2

```
/* ----- */
/* 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;

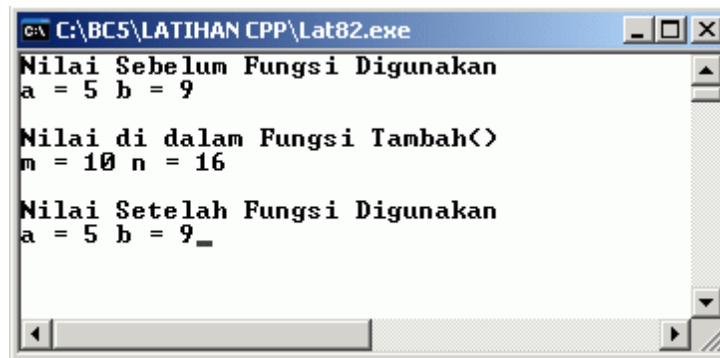
 getch();
}

tambah(int m, int n)
{
 m+=5;
 n+=7;

 cout<<"\n\nNilai di dalam Fungsi Tambah()";
 cout<<"\nm = "<<m<<" n = "<<n;
 cout<<endl;
}
```



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



Gambar 9.2. Hasil Contoh-2

### 9.3.2. Pemanggilan dengan Referensi (*Call by Reference*)

#### Penjelasan

Pemanggilan dengan reference merupakan pemanggilan alamat suatu variabel didalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel yang diluar dari fungsi dengan melaksanakan pengubahan nilai dari suatu variabel dilakukan didalam fungsi.

#### Contoh-3

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

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

tambah(int *c, int *d);

main()
{
 int a, b;
 a = 4;
 b = 6;

 clrscr();
 cout<<"Nilai Sebelum Pemanggilan Fungsi";
 cout<<"\na = "<<a<<" b = "<<b;

 tambah(&a, &b);

 cout<<endl;
 cout<<"\nNilai Setelah Pemanggilan Fungsi";
 cout<<"\na = "<<a<<" b = "<<b;
 getch();
}
```

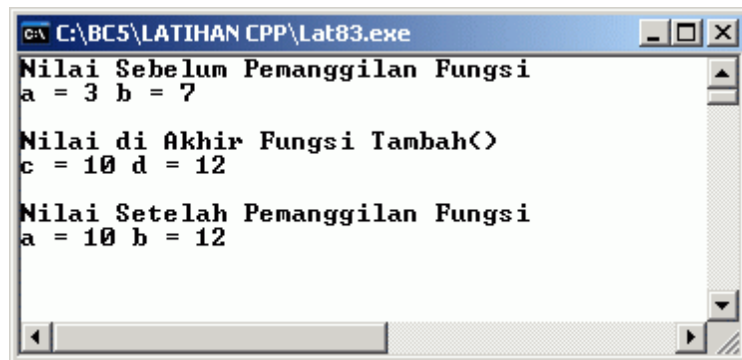
```

tambah(int *c, int *d)
{
 *c+=7;
 *d+=5;

 cout<<endl;
 cout<<"\nNilai di Akhir Fungsi Tambah() ";
 cout<<"\nc = "<<*c<<" d = "<<*d;
}

```

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



Gambar 9.3. Hasil Contoh-3

## 9.4. Pernyataan *return()*.

### Penjelasan

Digunakan untuk mengirimkan nilai atau nilai dari suatu fungsi kepada fungsi yang lain yang memanggilnya. Pernyataan ***return()*** diikuti oleh argumen yang berupa nilai yang akan dikirimkan. Contoh pemakaian pernyataan ***return()*** dapat dilihat pada contoh berikut ;

### Contoh-4

```

/* ----- */
/* Penggunaan Fungsi return() */
/* ----- */

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

tambah(int *c); //prototype fungsi tambah

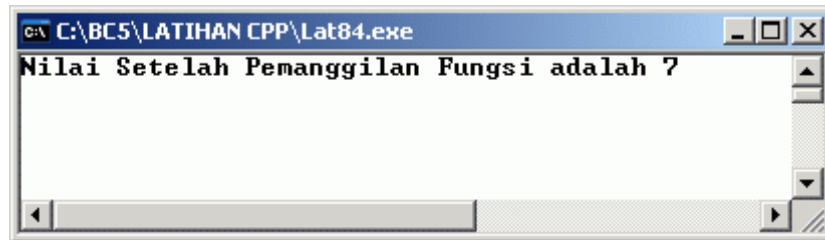
main()
{
 int a, b = 5;

 clrscr();
 a = tambah(&b);
 cout<<"Nilai Setelah Pemanggilan Fungsi adalah "<<a;
 getch();
}

```

```
tambah(int *c) //fungsi tambah
{
 return(*c+=2);
}
```

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



Gambar 9.4. Hasil Contoh-4

## 9.5. Pengiriman Data Ke Fungsi

### 9.5.1. Pengiriman Data Konstanta Ke Fungsi.

**Penjelasan** Mengirimkan suatu nilai data konstanta ke suatu fungsi yang lain dapat dilakukan dengan cara yang mudah, dapat dilihat dari program berikut :

**Contoh-5**

```
/* ----- */
/* Pengiriman data Konstanta */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

luas(float sisi);

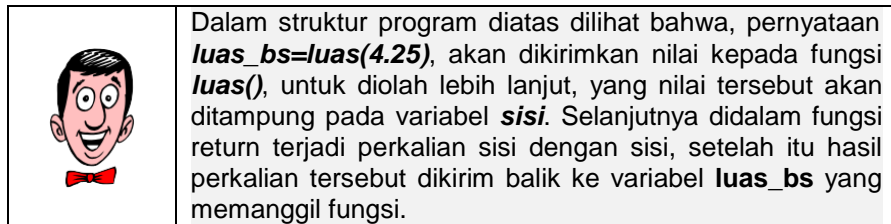
main()
{
 float luas_bs;

 clrscr();

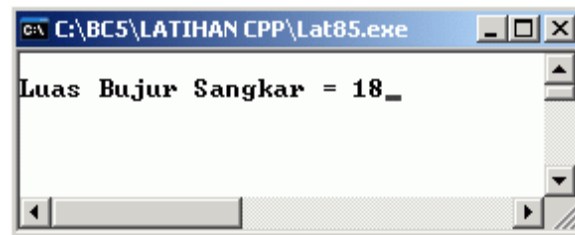
 luas_bs = luas(4.25);
 cout<<"\nLuas Bujur Sangkar = "<<luas_bs;
 getch();
}

luas(float sisi)
{
 return(sisi*sisi);
}
```

*Keterangan :*



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



Gambar 9.5. Hasil Contoh-5

## 9.5.2. Pengiriman Data Variabel Ke Fungsi

### Penjelasan

Bentuk pengiriman data Variabel, sama seperti halnya pengiriman suatu nilai data konstanta ke suatu fungsi, hanya saja nilai yang dikirimkan tersebut senantiasa dapat berubah-ubah. Bentuk pengiriman tersebut dapat dilihat dari program berikut :

### Contoh-6

```
/* ----- */
/* Pengiriman data Variabel */
/* ----- */

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

luas(float sisi);

main()
{
 float luas_bs, sisi_bs;

 clrscr();

 cout<<"\nMenghitung Luas Bujur Sangkar"<<endl;
 cout<<"\nMasukan Nilai Sisi Bujur Sangkar : ";
 cin>>sisi_bs;

 luas_bs = luas(sisi_bs);

 cout<<"\nLuas Bujur Sangkar = "<<luas_bs<<" Cm";

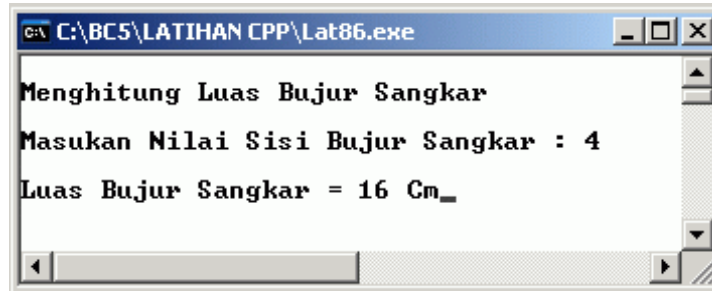
 getch();
}
```

```

luas(float sisi)
{
 return(sisi*sisi);
}

```

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



Gambar 9.6. Hasil Contoh-6

## 9.6. Jenis Variabel

**Penjelasan** 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.

### 9.6.1. Variabel Lokal

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

**Contoh-7**

```

/* ----- */
/* Variabel Lokal */
/* ----- */

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

lokal();

main()
{
 int a = 15;

 clrscr();

 cout<<"Pemanggilan Variabel Lokal"<<endl;

```

```

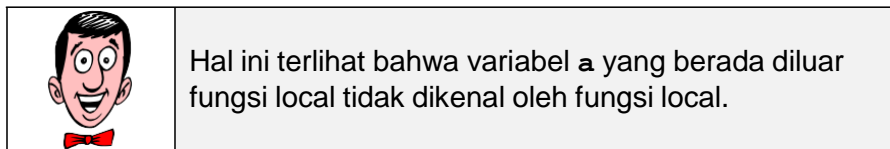
 cout<<"\nNilai didalam fungsi main() = : "<<a;

 lokal();

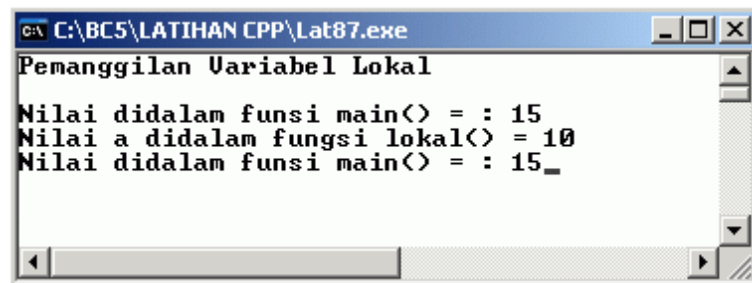
 cout<<"\nNilai didalam fungsi main() = : "<<a;
 getch();
 }

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

```



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



Gambar 9.7. Hasil Contoh-7

## 9.6.2. Variabel Eksternal

### Penjelasan

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-8

```

/* ----- */
/* 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;

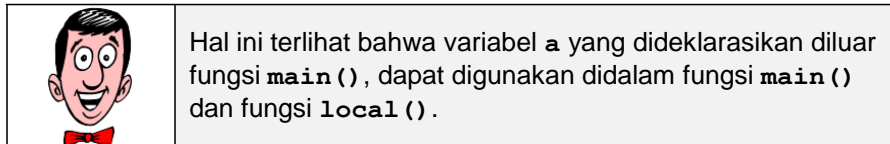
 cout<<"\nNilai didalam fungsi main() = : "<<a;

 lokal(); //--> pemanggilan fungsi lokal

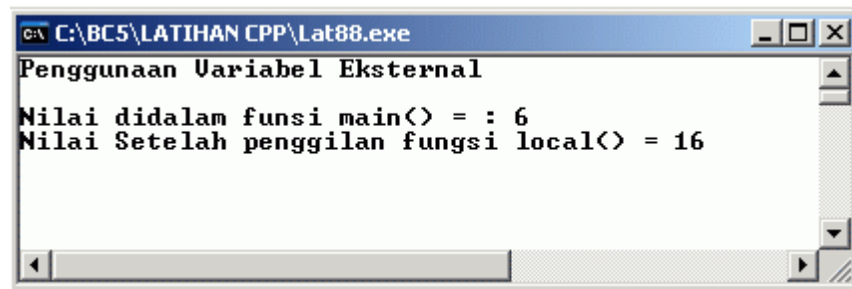
 cout<<"\nNilai Setelah panggilan fungsi lokal() = ";
 cout<<a;
 getch();
}

void lokal()
{
 a+=10;
}

```



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



Gambar 9.8. Hasil Contoh-8

### 9.6.3. Variabel Statis

#### Penjelasan

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-9**

```

/* ----- */
/* Penggunaan Variabel Statis */
/* ----- */
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

walah(); //--> prototipe fungsi walah


main()
{
 int k = 5;
 clrscr();

 walah();
 walah();

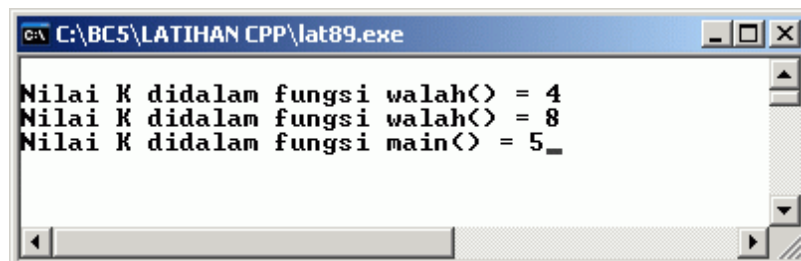
 cout<<"\nNilai K didalam fungsi main() = "<<k;
 getch();
}

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

```

|                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>Hal ini terlihat bahwa :</p> <ul style="list-style-type: none"> <li>• Pada pada prototipe fungsi <code>walah()</code> tidak tedapat nilai awal, maka secara otomatis variabel <code>k = 0</code>.</li> <li>• Pada pemanggilan fungsi <code>walah()</code> pertama, tercetak nilai variabel <code>k = 4</code>, didapat dari <code>k=0+4</code>.</li> <li>• Pada pemanggilan fungsi <code>walah()</code> kedua, tercetak nilai variabel <code>k = 8</code>, didapat dari <code>k=4+4</code>, karena nilai <code>k</code> yang terbaru adalah 4.</li> <li>• Pada pencetakan <code>k</code> didalam fungsi <code>main()</code>, adalah 5, karena variabel <code>k</code>, didalam fungsi <code>main()</code> bersifat lokal.</li> </ul> |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

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



Gambar 9.9. Hasil Contoh-9



## 9.7. Fungsi inline

### Penjelasan

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-10

```
/* ----- */
/* Penggunaan inline function */
/* ----- */

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

inline int kali(int i, int j)
{
 return(i * j);
}

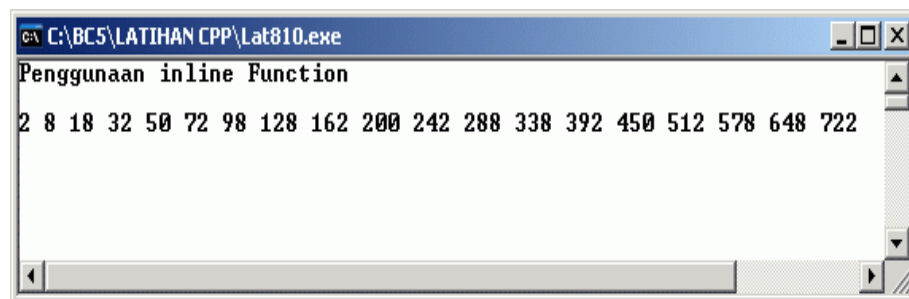
main()
{
 int k;

 clrscr();

 for(k = 1; k < 20; k++)
 cout<<kali(k, k*2)<<" ";

 getch();
}
```

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



Gambar 9.10. Hasil Contoh-10

### Contoh-11

```
/* ----- */
/* Penggunaan inline function */
/* ----- */
```

```

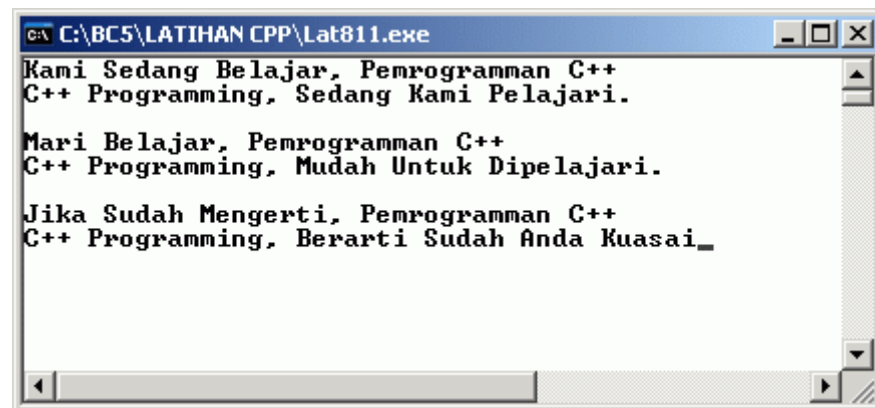
#include<conio.h>
#include<stdio.h>
#include<iostream.h>

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";
 }
 getch();
}

```

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



Gambar 9.11. Hasil Contoh-11

## 9.8. Function Overloading

### Penjelasan

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. Sebagai contoh membentuk fungsi yang sama dengan tipe yang berbeda-beda dan dibuatkan pula nama fungsi yang berbeda-beda pula.

**Contoh-12**

```

/* ----- */
/* 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();
}

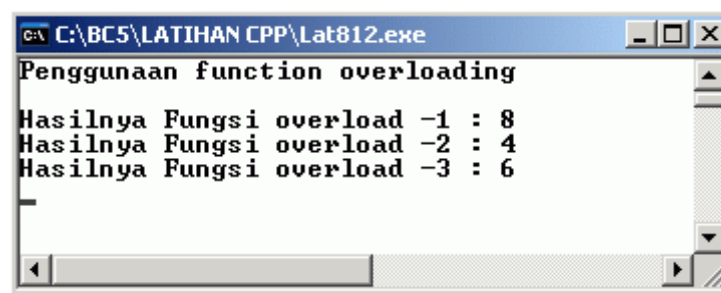
int hitung(int b)
{
 return(b*b);
}

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

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

```

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



Gambar 8.12. Hasil Contoh-12

## 9.9. Latihan

**Penjelasan No. 1** Buatlah program menghitung luas dan keliling lingkaran dengan menggunakan fungsi. Fungsi yang harus dibuat **luas()** untuk menghitung luas lingkaran dan **keliling()** untuk menghitung keliling lingkaran.

**Penjelasan No. 2** 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 >

**Penjelasan No. 3** Buatlah program untuk menghitung konversi dari derajat fahrenheit ke celcius

Petunjuk :

1. Gunakan Function Overloading.
2. 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
3. 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

**Penjelasan No. 4**     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>

Lembar ini Sengaja Dikosongkan  
( Untuk Catatan Boleh Juga )

## Pemrogramman C++

# Bab 10 : Macro

**Penjelasan** Didalam penyusunan suatu makro ada beberapa hal yang perlu dipelajari adalah :

## 10.1. Preprocessor Directives

**Penjelasan** Adalah 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 :

### 10.1.1. # define

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

```
#define nama_konstanta teks
```

Contoh :

```
#define A 6
```

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                      - #define PI 3.14
- Karakter                    - #define HURUF 'B'
- String                        - #define JABATAN "INSTRUCTOR"
- Pernyataan                - #define CETAK ("Borland C++")
- Fungsi Sederhana       - #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.

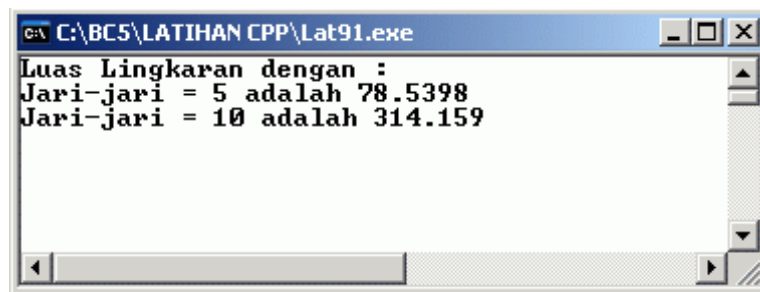
**Contoh-1**

```
/* ----- */
/* 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 :



Gambar 9.1 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();

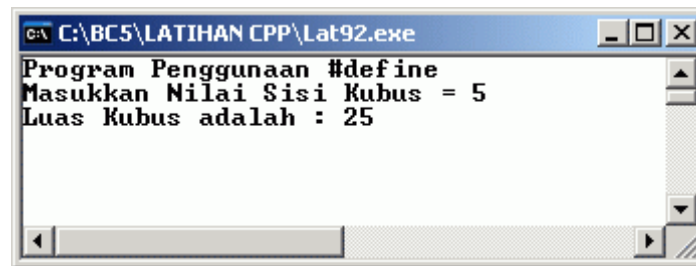
cetak<<"Program Penggunaan #define"<<endl;
cetak<<"Masukkan Nilai Sisi Kubus = ";
masuk>>sisi;

ls_kubus = LS_KUBUS;

cetak<<"Luas Kubus adalah : "<<ls_kubus;
tahan();
akhir

```

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



Gambar 9.2. Hasil Contoh-2

### Contoh-3

```

/*****
Contoh Bubble Sorting
*****/
#include <stdio.h>
#include <stdlib.h>
#define SWAP(a,b) { int t; t=a; a=b; b=t; }
#define INDEX 8

void bubble_srt(int a[], int n);
int main(void)
{
 int i;
 int array[INDEX] = {12, 9, 4, 99, 120, 1, 3, 10};

 printf("Sebelum di Urutkan :\n");
 for(i = 0; i < INDEX; i++)
 printf("%d ", array[i]);
 printf("\n");

 bubble_srt(array, INDEX); // Mengurutkan array

 printf("\nSetelah diurutkan :\n");
 for(i = 0; i < INDEX; i++)
 printf("%d ", array[i]);
 printf("\n\n");

 system("PAUSE"); // Menghentikan tampilan
 return 0;
}

```

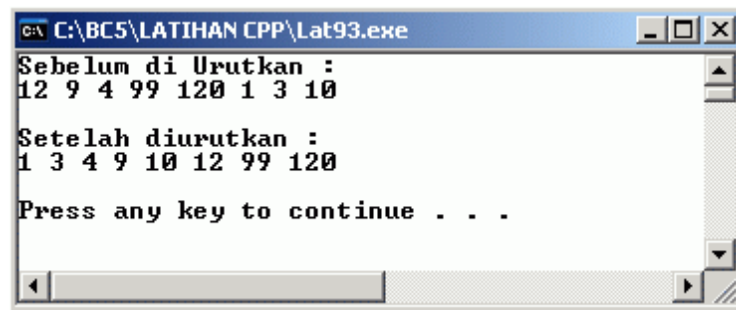
```

/*****
Fungsi Bubble Sort
*****/
void bubble_srt(int a[], int n)
{
 int i, j;

 for(i = 0; i < n; i++)
 {
 for(j = 1; j < (n-i); j++)
 {
 if(a[j-1] > a[j])
 SWAP(a[j-1],a[j]);
 }
 }
}

```

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



Gambar 9.3. Hasil Contoh-3

## 10.1.2. # include

### Penjelasan

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 direktori yang sedang aktif dan apa bila tidak ditemukan akan mencari pada direktori 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.

### 10.1.3. # if - #endif

**Penjelasan** 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-4**

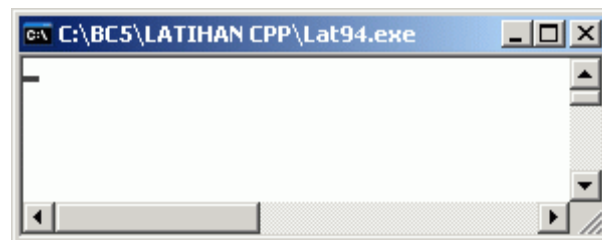
```

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

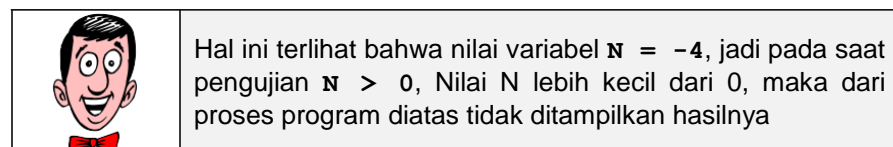
#define N -4
main()
{
 #if N > 0
 printf("Lebih Besar dari Nol");
 #endif
}

```

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



Gambar 9.4. Hasil Contoh-4



### 10.1.4. # if - #else - #endif

**Penjelasan** 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-5**

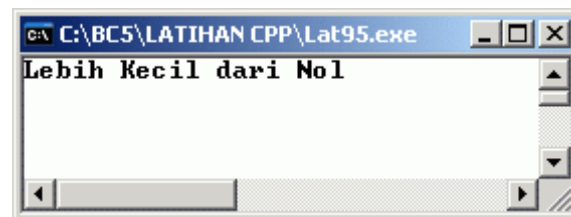
```

/*****
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-5 diatas adalah :



Gambar 9.5. Hasil Contoh-5

### 10.1.5. # elif

**Penjelasan** 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**

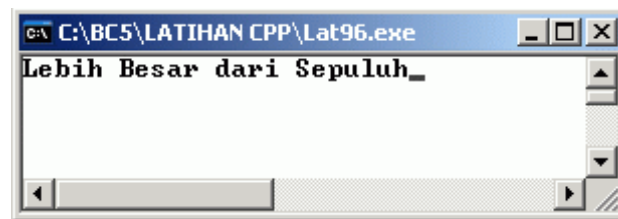
```
#if ekspresi-konstanta-1
 Pernyataan-1;
#elif ekspresi-konstanta-2 Pernyataan-
 2;
...
```

```
...
#elif ekspresi-konstanta-n
 Pernyataan-n;
#endif
```

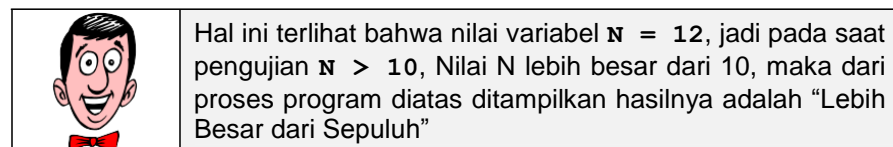
**Contoh-6**

```
/******
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-6 diatas adalah :



Gambar 9.6. Hasil Contoh-6



## 10.1.6. #undef

**Penjelasan**

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

**Contoh-7**

```
/******
Penggunaan #undef
*****/

#include<iostream.h>

#define LEBAR_MAKS 100

#if LEBAR_MAKS>200
#undef LEBAR_MAKS //--> menghilangkan LEBAR_MAKS
#define LEBAR_MAKS 200
```

```

#elseif LEBAR_MAKS <50
#undef LEBAR_MAKS //--> menghilangkan LEBAR_MAKS
#define LEBAR_MAKS 50

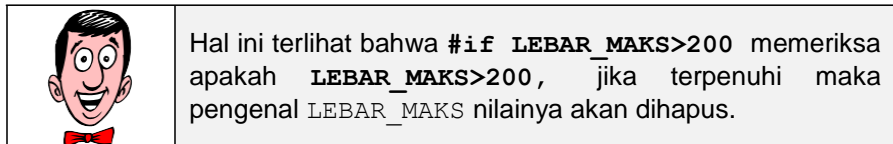
#else
#undef LEBAR_MAKS //--> menghilangkan LEBAR_MAKS
#define LEBAR_MAKS 50
#endif

main()
{
 char str[LEBAR_MAKS];

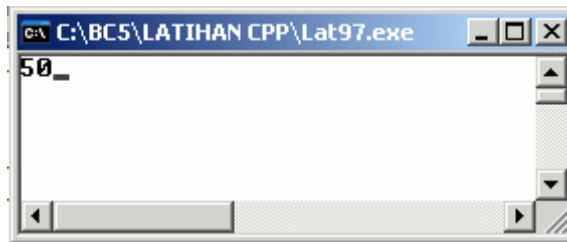
 cout<<LEBAR_MAKS;

}

```



Hasil dari program contoh-7 diatas adalah :



Gambar 9.7. Hasil Contoh-7

### 10.1.7. # ifdef - # ifndef

#### Penjelasan

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 Penulisan

```

#ifdef nama-konstanta
 pernyataan;
#endif

```

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

Contoh :

```
#ifdef LEBAR_MAKS
char str[LEBAR_MAKS];
#endif
```

```
#ifndef nama-konstanta
 pernyataan;
#endif
```

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

Contoh :

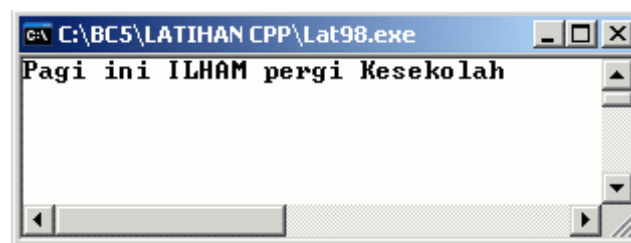
```
#ifndef LEBAR_MAKS
#define LEBAR_MAKS 100
#endif
char str[LEBAR_MAKS];
```

#### Contoh-8

//--> 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-8 diatas adalah :



Gambar 9.8. Hasil Contoh-8

## 10.2. Pembuatan File Header

### Penjelasan

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 longtime_t;
#endif
```

Sebagai latihan berikut kita akan membuat suatu file header sendiri yang akan digunakan pada file editor.

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

### Contoh-9

```
/* atur.h

 contoh pembuatan file header untuk
 pengaturan.

 Copyright (c) Frieyadie 2001
 All Rights Reserved.
*/

#define awal {
#define akhir }
#define mulai() main()
#define cetak cout
#define tampil cprintf
#define masuk scanf
#define hapus() clrscr()
#define jika if
#define warna textcolor
#define tahan getch()
```

Setelah disimpan, selanjutnya Compile file **atur.h**.



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

#### Contoh-10

```
/* ----- */
/* program dengan file header sendiri */
/* ----- */

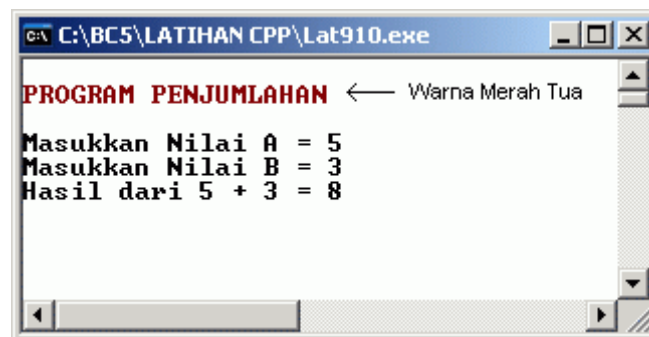
#include<stdio.h>
#include<conio.h>
#include<iostream.h>
#include"atur.h"

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-10 diatas adalah :



Gambar 9.9. Hasil Contoh-10

## 10.3. Latihan

**Penjelasan No. 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.

**Penjelasan No. 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 >

```

**Penjelasan No. 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>

## BORLAND C++

# Bab 10 : Structure

**Penjelasan** Structure digunakan untuk mengelompokan 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

## 11.1. Deklarasi Structure

**Penjelasan** Structure dapat mendeklarasikan seperti berikut

```
struct nama_tipe_struktur
{
 elemen_struktur;

};
```

atau

```
struct
{
 elemen_struktur;

} nama_tipe_struktur;
```

**Contoh Deklarasi**

```
struct
{
 char nim[5];
 char nama[15];
 float nilai;
} mahasiswa;
```

**Contoh-1**

```

/* ----- */
/* Program Penggunaan structure */
/* Nama File : struct1.cpp */
/* ----- */

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

main()
{
 struct
 {
 char nim[5];
 char nama[15];
 float nilai;
 } mahasiswa;

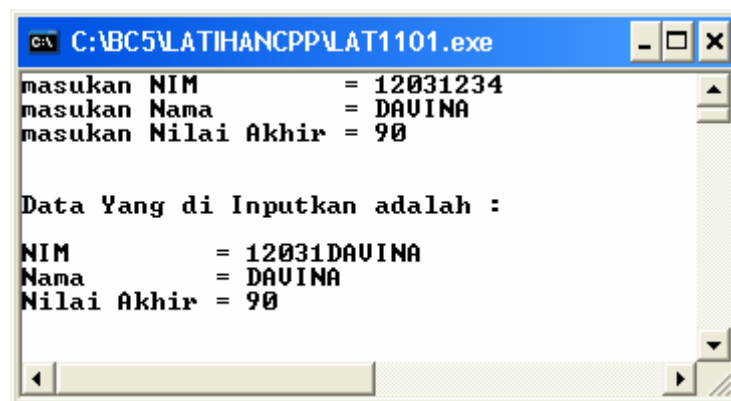
 clrscr();
 cout<<"masukan NIM = ";
 cin>>mahasiswa.nim;
 cout<<"masukan Nama = ";
 cin>>mahasiswa.nama;
 cout<<"masukan Nilai Akhir = ";
 cin>>mahasiswa.nilai;

 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 11.1. Hasil Contoh 1

## 11.2. Nested Structure

### Penjelasan

Nested Structure merupakan suatu Structure dapat digunakan didalam structure yang lainnya. Hal seperti ini anda dapat lihat pada program berikut ini :

### Contoh-2

```
/* ----- */
/* Program Penggunaan Nested structure */
/* Nama File : struct2.cpp */
/* ----- */
#include<stdio.h>
#include<conio.h>
#include<iostream.h>

main()
{
 struct dtmhs
 {
 char nim[9];
 char nama[15];
 };

 struct dtnil
 {
 float nil1;
 float nil2;
 };

 struct
 {
 struct dtmhs mhs;
 struct dtnil nil;
 } nilai;

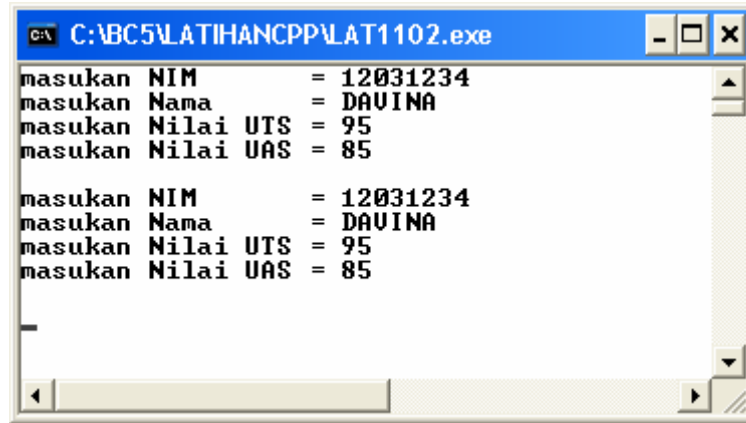
 clrscr();

 //-> masukan data
 cout<<"masukan NIM = "; cin>>nilai.mhs.nim;
 cout<<"masukan Nama = "; cin>>nilai.mhs.nama;
 cout<<"masukan Nilai UTS = "; cin>>nilai.nil.nil1;
 cout<<"masukan Nilai UAS = "; cin>>nilai.nil.nil2;
 cout<<endl;

 //-> menampilkan hasil masukan
 cout<<"masukan NIM = "<<nilai.mhs.nim<<endl;
 cout<<"masukan Nama = "<<nilai.mhs.nama<<endl;
 cout<<"masukan Nilai UTS = "<<nilai.nil.nil1<<endl;
 cout<<"masukan Nilai UAS = "<<nilai.nil.nil2<<endl;
 cout<<endl;

 getch();
 return(0);
}
```

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



Gambar 11.2. Hasil Contoh 2

### 11.3. Structure dengan Array

#### Penjelasan

Penggunaan Array sering dikaitkan dengan Structure, sehingga membentuk Array dari Structure. Berikut bentuk deklarasi array structure :

```
struct
{
 elemen_struktur;

} nama_tipe_struktur[jml_index];
```

#### Contoh-3

```
/* ----- */
/* Program Penggunaan array structure */
/* Nama File : struct3.cpp */
/* ----- */

#include<stdio.h>
#include<conio.h>
#include<iostream.h>
main()
{
 int i, j=1;
 struct
 {
 char nim[5];
 char nama[15];
 float nilai;
 } mhs[5];

 clrscr();
 for(i=0; i<2; i++)
 {
 cout<<"masukan NIM = "; cin>>mhs[i].nim;
 cout<<"masukan Nama = "; cin>>mhs[i].nama;
 cout<<"masukan Nilai Akhir = "; cin>>mhs[i].nilai;
 }
}
```

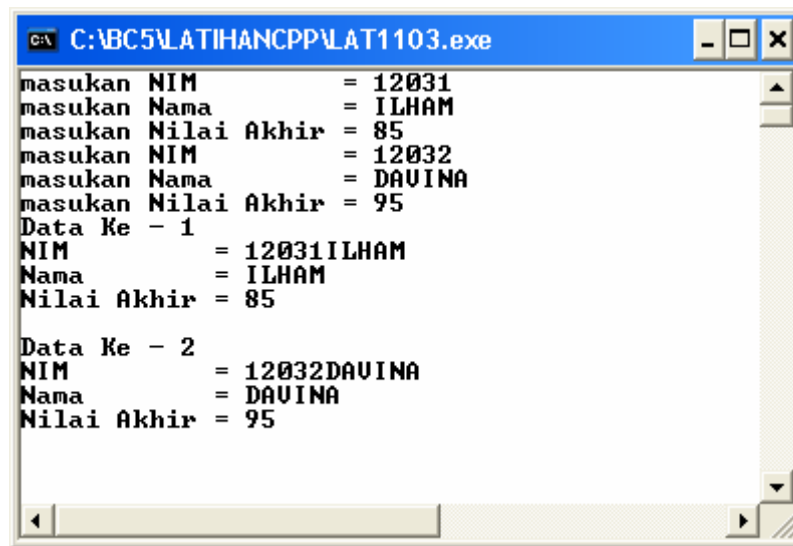
```

for(i=0; i<2; i++)
{
 cout<<"Data Ke - "<<j++<<endl;
 cout<<"NIM = "<<mhs[i].nim<<endl;
 cout<<"Nama = "<<mhs[i].nama<<endl;
 cout<<"Nilai Akhir = "<<mhs[i].nilai<<endl;
 cout<<endl;
}

getch();
}

```

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



Gambar 11.3. Hasil Contoh 3

## 11.4. Structure dengan Function

### Penjelasan

Suatu elemen-elemen dari suatu Structure dapat dikirimkan ke dalam suatu function dengan cara yang sama seperti mengirimkan suatu variabel sederhana kedalam suatu function.

Berikut contoh sederhana yang anda dapat lihat pada contoh program berikut :

### Contoh-4

```

/* ----- */
/* 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, j=1, k=1;

```

```

struct
{
 char nim[5];
 char nama[15];
 float nilai;
} mhs[5];

clrscr();
for(i=0; i<2; i++)
{
 cout<<"Data Ke - "<<j++<<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;
}

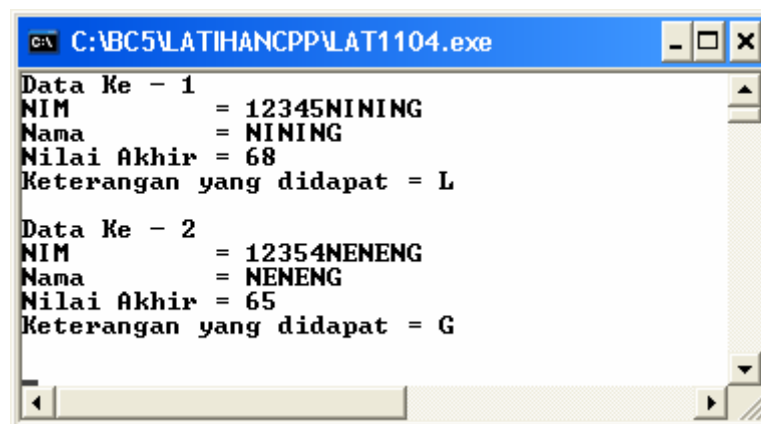
clrscr();
for(i=0; i<2; i++)
{
 cout<<"Data Ke - "<<k++<<endl;
 cout<<"NIM = "<<mhs[i].nim<<endl;
 cout<<"Nama = "<<mhs[i].nama<<endl;
 cout<<"Nilai Akhir = "<<mhs[i].nilai<<endl;
 cout<<"Keterangan yang didapat = ";
 cout<<ket(mhs[i].nilai)<<endl;
 cout<<endl;
}

getch();
}

char ket(float n)
{
 if(n > 65)
 return 'L';
 else
 return 'G';
}

```

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



Gambar 11.4. Hasil Contoh 4



## 11.5. Latihan

### Penjelasan

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  $\geq 80$
- Jika Nilai Huruf = B, maka Nilai Hasil  $\geq 70$
- Jika Nilai Huruf = C, maka Nilai Hasil  $\geq 56$
- Jika Nilai Huruf = D, maka Nilai Hasil  $\geq 47$
- 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.

% Tampilan akhir adalah sebuah tabel, seperti dibawah ini :

Daftar Honor Pegawai Honorer  
PT. ALAKADARNYA

| No  | Nama<br>Pegawai | Jumlah<br>Jam Kerja | Kelebihan<br>Jam Kerja | Jumlah<br>Honor |
|-----|-----------------|---------------------|------------------------|-----------------|
| ... | .....           | .....               | .....                  | .....           |
| ... | .....           | .....               | .....                  | .....           |

---

# Daftar Pustaka

Frieyadie. 2006. Panduan Pemrogramman C++. Penerbit Andi. Yogyakarta

Kadir, Abdul, 1995. Pemrogramman C++ Membahas Pemrogramman Berorientasi Objek. Penerbit Andi. Yogyakarta

Potts, Steve dan Clayton Walnum.1997.Pemrogramman Berorientasi Objek dengan Borland C++. Penerbit Andi. Yogyakarta

URL : <http://www.glenmccl.com/tutor.htm>

<http://cplus.about.com>

<http://www.cprogramming.com>

<http://codenewbie.com/tutorials>