MODUL

DASAR PEMROGRAMAN

**DAFTAR ISI**

Halaman Judul i Kata Pengantar ii Daftar Isi iii

BAB 1 Pengenalan Bahasa C++ 1

BAB 2 Pengenalan Tipe Data, Perintah Masukan dan Keluaran 10

BAB 3 Operasi Penyeleksian Kondisi 29

BAB 4 Proses Perulangan 39

BAB 5 Array 50

Daftar Pustaka 54

**BAB 1**

**Pengenalan Bahasa C++**

## Sejarah Singkat

### Sekilas Perkembangan Bahasa

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

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

### Sekilas Tentang C++

Pada tahun 1980 seorang ahli yang bernama Bjarne Stroustrup

mengembangkan beberapa hal dari bahasa C yang dinamakan “C with Classes” yang pada mulanya disebut “a better C” dan berganti nama pada tahun 1983 menjadi C++ oleh Rick Mascitti, dibuat di Laboratorium Bell, AT&T.

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

### Perbedaan Antara Bahasa C Dengan C++

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



## Pengenalan IDE Borland C++

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

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

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

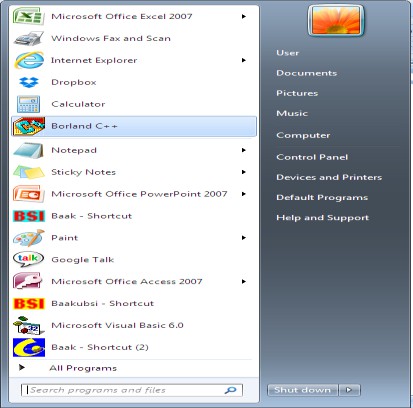
#### Langkah membuka program C++

KLIK TOMBOL START  ALL PROGRAM BORLAND C++ 5.02 

BORLAND C++

#### Langkah membuka program C++

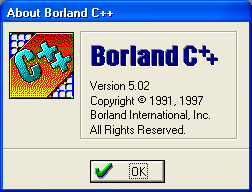
KLIK TOMBOL START  ALL PROGRAM BORLAND C++ 5.02 

BORLAND

#### Gambar 1.1.

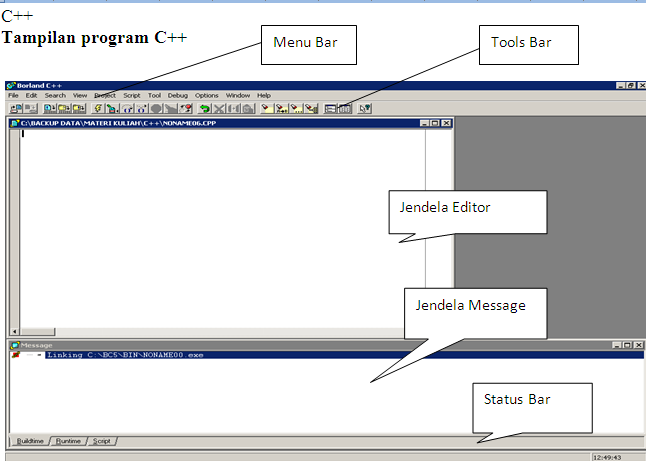
**Menu Untuk Mengaktifkan Program Borland C++**

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



**Gambar 1.2.**

**Layar Pembuka Borland C++**



**Gambar 1.3.**

**IDE Borland C++ 5.02**

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

#### Baris Menu ( Menu Bar )

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

#### Baris Peralatan (Tools Bar)

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

#### Jendela Editor

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

#### Jendela Message

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

#### Baris Status (Status Bar)

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

## Struktur Program C++

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

**Bentuk Umum:**

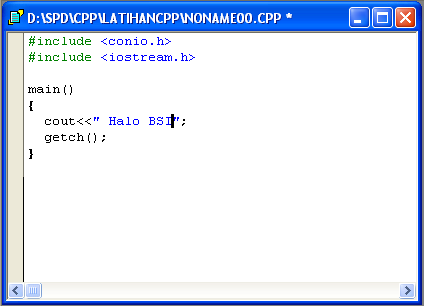
# prepocessor directive

main()

{

**//Batang tubuh program utama**

**Contoh-1**



**Penjelasan :**

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

## Model Memori

Borland C++, mempunyai enam model memori untuk program dan data.

Model-model memori tersebut adalah:

#### Model Tiny

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

#### Model Small

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

#### Model Medium

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

#### Model Compact

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

#### Model Large

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

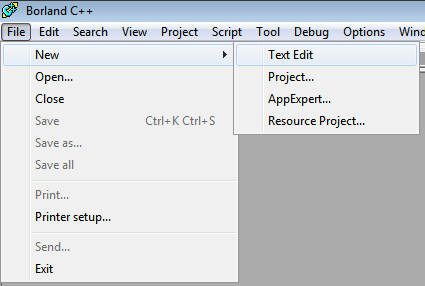
#### Model Huge

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

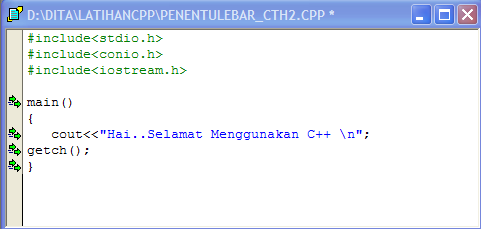
## Membuat File Editor

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

Cara mengaktifkannya : Klik Menu File Klik New Text Edit



**Gambar 1.5** Cara Menampilkan Text Edit



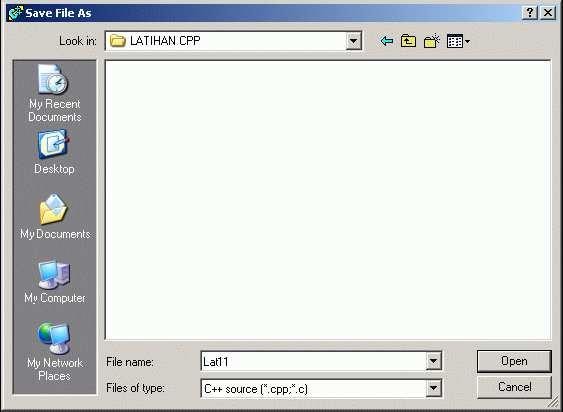
**Gambar 1.6** Jendela Text Edit

## Menyimpan File Editor

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

1. Kik Menu File Save
2. Menekan HotKey **Ctrl + KS**.

Selanjutnya tampil jendela Save File As, seperti dibawah ini :



**Gambar 1.7.** Jendela Save File As

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

**Save** Digunakan untuk menyimpan File Program pada jendela yang sedang aktif kedalam disk. Hotkey yang ada bisa gunakan untuk menyimpan dengan menekan tombol **Ctrl + KS**.

**Save As** Digunakan untuk menyimpan File Program pada jendela yang sedang aktif kedalam disk dengan nama file yang berbeda.

**Save All** Digunakan untuk menyimpan semua File Program pada jendela yang sedang aktif kedalam disk.

## Menterjemahkan Program

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

1. Kik Menu **Debug Compile**
2. Menekan HotKey Alt + F9

Selanjutnya tampil kotak dialog Compile, seperti dibawah ini :



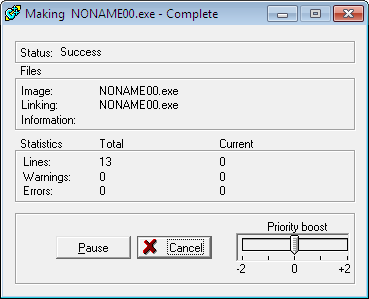
**Gambar 1.8** Kotak Dialog Compile

## Menjalankan Program

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

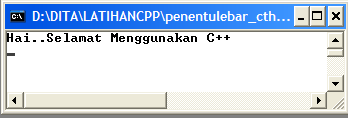
1. Kik Menu **Debug Run**
2. Menekan HotKey Ctrl + F9

Selanjutnya tampil kotak dialog Run, seperti dibawah ini :



**Gambar 1.9** Kotak Dialog Run

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

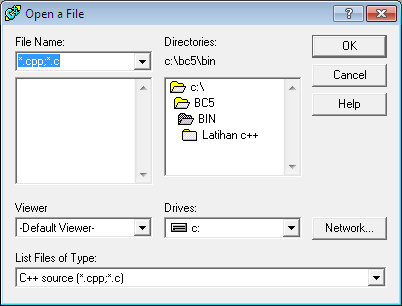


**Gambar 1.10** Contoh Hasil Keluaran Program

## Membuka File Editor

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

Selanjutnya tampil Jendela Open, seperti dibawah ini :



**Gambar 1.11** Jendela Open File pada C++

## Mencetak File Editor

Penjelasan Mencetak file program pada jendela yang sedang aktif dengan cara Klik

#### File Print

Selanjutnya tampil Jendela Print Option, seperti dibawah ini



**Gambar 1.12** Jendela Print Option

## Keluar dari Borland C++ 5.02

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



**BAB 2**

**Pengenalan Tipe Data,**

**Perintah Masukan dan Keluaran**

## Pengenalan Tipe Data

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

#### Tabel 2.1. Tipe Data

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

**Tipe Data Tambahan**, yang dimiliki oleh Borland C++, adalah :

**Unsigned** digunakan bila data yang digunakan hanya data yang positif saja.

**Tabel 2.2. Tipe Data Tambahan Tambahan**

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

## Konstanta

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

Konstanta Bilangan Konstanta Teks



### Konstanta Bilangan

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

* 1. Konstanta Bilangan Bulat (*Integer*).

Adalah bilangan yang tidak mengandung nilai desimal. Ini merupakan nilai default pada konstanta bilangan.

Contoh : 1, 2, 3, 100

* 1. Konstanta Desimal Berpresisi Tunggal ( *Floating Point* ) Konstanta Floating Point, mempunyai bentuk penulisan, yaitu :

Bentuk Desimal ( contoh : 5.57 )

Bentuk Eksponensial / Bilangan Berpangkat ( contoh : 4.22e3 4.22 x 103 )s

* 1. 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.

### Konstanta Teks

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

* 1. Data Karakter (*Character*).

Data karakter hanya terdiri dari sebuah karakter saja yang diapit oleh tanda kutip tunggal ( ‘ ). Data karakter dapat berbentuk abjad ( *huruf besar atau kecil* ), angka, notasi atau simbol.

Contoh : ‘Y’ ‘y’ ‘9’ ‘&’ dan lain-lain.

* 1. 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.

### Deklarasi Konstanta

Bentuk deklarasi konstanta diawali dengan reserved word ***const***. Bentuk penulisannya :

**const nama\_konstanta = nilai konstanta;**

atau

**const tipe\_data nama\_konstanta = nilai konstanta;**

**Contoh: const x = 89;**

**const float phi = 3.14;**

## Variabel

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

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

1. Tidak boleh ada spasi ( contoh : gaji bersih ) dan dapat menggunakan tanda garis bawah ( \_ ) sebagai penghubung (contoh : gaji\_bersih).
2. Tidak boleh diawali oleh angka dan menggunakan operator aritmatika. Variabel, dibagi menjadi dua jenis kelompok, yaitu :

Variabel Numerik Variabel Teks

### Variabel Numerik

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

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

### Variabel Text

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

### Deklarasi Variabel

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

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

#### Tabel 2.3. Deklarasi Variabel

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

Bentuk penulisannya :

**Tipe data nama\_variabel;**

**Contoh Deklarasi** char nama\_mahasiswa[20];

char grade; float rata\_rata ; int nilai;

## Perintah Keluaran

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

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

## printf( )

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

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

String-Kontrol dapat berupa keterangan yang akan ditampilkan pada layar beserta penentu format. Penentu format dipakai untuk memberi tahu 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**  ***printf( )*** |
| Integer | %d |
| Floating Point |  |
| Bentuk Desimal | %f |
| Bentuk Berpangkat | %e |
| Bentuk Desimal dan Pangkat | %g |
| Double Precision | %lf |
| Character | %c |
| String | %s |
| Unsigned Integer | %u |
| Long Integer | %ld |
| Long Unsigned Integer | %lu |
| Unsigned Hexadecimal Integer | %x |
| Unsigned Octal Integer | %o |



printf(“%c merupakan abjad yang ke - %d”,’b’,2);



**Contoh-1** #include <stdio.h> #include <conio.h> #include<iostream.h>

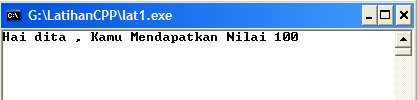
main( )

{

char nama =”dita”; int nilai = 100; clrscr( );

printf(“hai %s, Kamu mendapatkan Nilai %i” , nama, nilai);

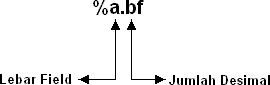
}



**Gambar 2. 1** Hasil Contoh 1

## Penggunaan Penentu Lebar Field

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



**Contoh-2** #include<stdio.h> #include<conio.h> #include<iostream.h> main( )

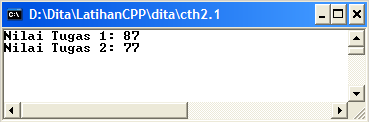
{

float a1=87.5, a2=77.50; clrscr( );

printf("Nilai Tugas 1: %2f \n",a1); printf("Nilai Tugas 2: %2f ",a2); getch( );

}

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



**Gambar 2. 2** Hasil Contoh 2

**Contoh-3** #include<stdio.h> #include<conio.h> #include<iostream.h> main( )

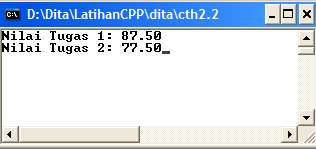
{

float a1=87.5, a2=77.50; clrscr();

printf("Nilai Tugas 1: %2.2f \n",a1); printf("Nilai Tugas 2: %2.2f",a2); getch( );

}

Output yang akan dihasilkan, jika menggunakan penentu lebar field adalah



**Gambar 2. 3** Hasil Contoh 3

## Penggunaan Escape Sequences.

Escape Sequences menggunakan notasi “ \ ” ( back slash ) jika karakter terdapat notasi “\” ini sebagai karakter “escape” ( menghindar).

Beberapa Escape Sequences lainnya antara lain :

#### Tabel 2.5. Escape Sequences

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

|  |  |
| --- | --- |
| \" | Tanda Kutip Ganda ( " ) |
| \\ | Backslash |
| \xaa | Kode ASCII dalam hexadecimal.  ( aa menunjukkan angka ASCII ybs ) |
| \aaa | Kode ASCII dalam octal. (aaa menunjukkan  angka ASCII ybs ) |

**Contoh-4** #include<stdio.h>

#include<conio.h> main( )

{

char nama1[10]="Dita",nama2[10]="Ani",nama3[10]="Fitri"; float a = 88.5, b = 90.8, c = 98.2;

clrscr( );

printf("%8s\t %7s\t %8s ", nama1,nama2,nama3); printf("\n%8.2f \t %8.2f \t %8.2f ", a,b,c);

getch( );

}



**Gambar 2. 4** Hasil Contoh 4

## puts( )

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

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

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

|  |  |
| --- | --- |
| ***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()

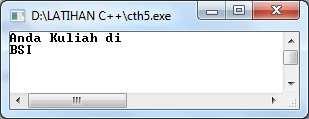
{

char nama[5] = "BSI"; clrscr();

puts("Anda Kuliah di "); puts(a);

}

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



**Gambar 2. 5** Hasil Contoh 5

## putchar( )

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

**Contoh-5** #include <stdio.h>

#include <conio.h>

main( )

{

clrscr( ); putchar('B');

putchar('S');

putchar('I');

putchar('-');

putchar('2');

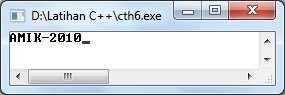
putchar('0');

putchar('0');

putchar('9'); getch( );

}

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



**Gambar 2. 6** Hasil Contoh 6

## cout

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

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

main( )

{

int jumbar1=150, jumbar2=23;

clrscr( );

cout<<"Jumlah Barang 1: "<<jumbar1; cout<<" Jumlah Barang 2: "<<jumbar2;

getch( );

}

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



**Gambar 2. 7** Hasil Contoh 7

## Fungsi Manipulator

Manipulator pada umumnya digunakan untuk mengatur tampilan layar, untuk mengguakan manipulator ini file header yang harus disertakan file header **iomanip.h**

. Ada beberapa fungsi manipulator yang disediakan oleh Borland C++, antara lain.

**endl end flush( ) dec( )**

**hex( )**

**oct( )**

**setbase( ) setw( ) setfill( )**

**setprecision( ) setosflags( )**

Berikut akan dibahas beberapa fungsi manipulator, diantaranya:

## endl

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

**Contoh-8** # include<stdio.h>

# include<conio.h>

# include<iostream.h> main( )

{

char nim[9]="12098890", nama[15]="Andi"; char alamat[20]="Jakarta Selatan";

clrscr( );

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

}

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



**Gambar 2. 8** Hasil Contoh 8

## ends

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

**Contoh-9** # include <stdio.h>

# include <conio.h>

# include <iostream.h> main( )

{

int a, b, c, d;

clrscr( );

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

cout<<"Masukan Nilai B : "; cin>>b;

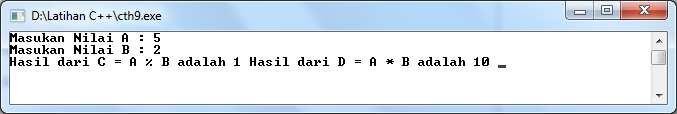
c = a % b;

d = a \* b;

cout<<"Hasil dari C = A % B adalah "<<c<<ends; cout<<"Hasil dari D = A \* B adalah "<<d<<ends; getch( );

}

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



**Gambar 2. 9** Hasil Contoh 9

## dec, oct dan hex

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

**Contoh-10** # include <stdio.h>

# include <conio.h>

# include <iostream.h> # include <iomanip.h> main( )

{

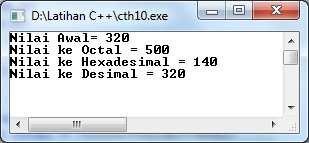
int nilai = 320; clrscr( );

cout<<"Nilai Awal= "<<nilai<<endl; cout<<"Nilai ke Octal = "<<oct<<nilai<<endl; cout<<"Nilai ke Hexadesimal = "<<hex<<nilai<<endl; cout<<"Nilai ke Desimal = "<<dec<<nilai<<endl;

getch( );

}

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



**Gambar 2. 10** Hasil Contoh 10

## setprecision( )

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

**Contoh-11** # include <conio.h>

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

main( )

{

float a,b,c;

a = 25.23;

b = 12.54;

clrscr( ); c = a \* b;

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

}

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

**Gambar 2. 11** Hasil Contoh 11

#### setbase( )

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

Bentuk penulisannya :

#### setbase(base bilangan);

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

* + Octal = basis 8
  + Decimal = basis 10
  + Hexadecimal = basis 16

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

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

# include <iostream.h> # include <iomanip.h> main( )

{

int a = 250; clrscr( );

cout<<"Penggunan Manipulator setbase()"<<"\n\n"; cout<<"Decimal Octal Hexadecimal"<<endl; cout<<" "<<"\n";

for(a=250;a<=260;a++)

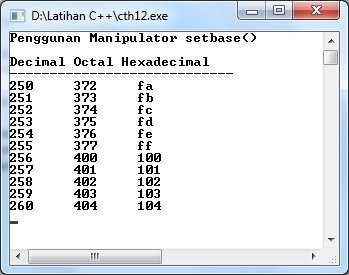
{

cout<<setbase(10)<<a<<" "; cout<<setbase(8)<<a<<" "; cout<<setbase(16)<<a<<endl;

}

getche( );

}

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

**Gambar 2.12.** Hasil Contoh-12

#### setw()

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

Bentuk penulisannya :

#### setw(int n);

n = merupakan nilai lebar tampilan data, integer.

**Contoh-13** // Penggunaan Manipulator setw()

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

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

main( )

{

int a; clrscr( );

cout<<"Penggunan Manipulator setw()"<<"\n\n"; cout<<" "<<"\n";

for(a=1;a<=15;a++)

{

cout<<setw(a)<<a<<endl;

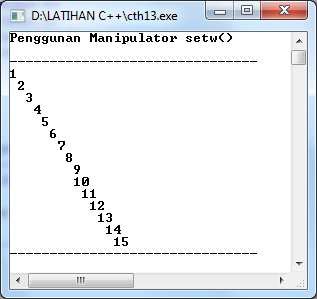
}

cout<<" "<<"\n";

getche( );

}

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



**Gambar 2.13.** Hasil Contoh-13

## Perintah Masukan

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

**scanf( ) gets( )**

**cout( ) getch( ) getche( )**

## scanf( )

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

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

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

Tabel 2.7. Penentu Format scanf( )

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

**Contoh-14** #include<stdio.h>

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



main( )

{

char nama[15],kelas[20],hobby[15]; clrscr( );

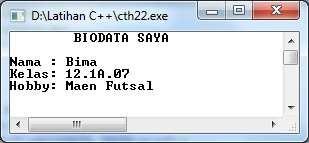
printf("\tBIODATA SAYA\n"); printf("Nama : ");scanf("%s",&nama);

printf("Kelas: ");scanf("%s",&kelas);

printf("Hobby: ");scanf("%s",&hobby); getch( );

}

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



**Gambar 2. 14** Hasil Contoh 22

## gets( )

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

**gets(nama-variabel-array);**

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

Tabel 2.8. Perbedaan scanf( ) dengan gets( )

|  |  |
| --- | --- |
| ***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-15** # include <stdio.h> # include <conio.h>

main( )

{

char nm1[20]; char nm2[20];

clrscr( );

puts("Masukan nama ke - 1 = "); gets(nm1);

printf("Masukan nama ke - 2 = "); scanf("%s",&nm2);

printf("\n\n");

puts("Senang Berkenalan Dengan Anda .."); puts(nm1);

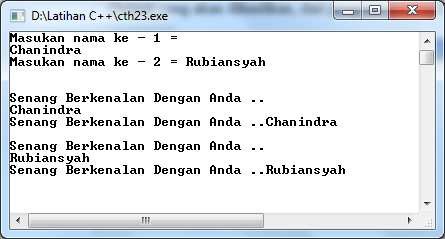
printf("Senang Berkenalan Dengan Anda ..%s", nm1); printf("\n\n");

puts("Senang Berkenalan Dengan Anda .."); puts(nm2);

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

}

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



**Gambar 2. 15** Hasil Contoh 23

## cin

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

**Contoh-16** # include <stdio.h> # include <conio.h>

# include <iostream.h> main( )

{

int nilai1,nilai2, total; clrscr( );

cout<<"Masukan Nilai 1 : "; cin>>nilai1; cout<<"Masukan Nilai 2 : ";

cin>>nilai2; total=nilai1+nilai2;

cout<<"Masukan Total Nilai : "<<total<<endl; getch( );

}

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



**Gambar 2. 16** Hasil Contoh 24

## getch ( )

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

**Contoh-17** # include <stdio.h> # include <conio.h>

main( )

{

char kar;

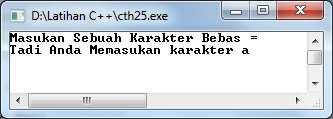
clrscr( );

printf("Masukan Sebuah Karakter Bebas = "); kar = getch( );

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

}

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



**Gambar 2. 17** Hasil Contoh 25

## getche()

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

**Contoh-18** # include <stdio.h> # include <conio.h>

main( )

{

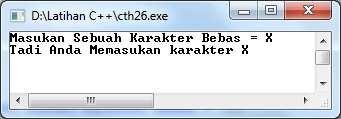
char kar; clrscr( );

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. 18** Hasil Contoh 26

**BAB 3**

**Operasi Penyeleksian Kondisi**

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

## Pernyataan IF

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

kondisi



salah

benar

perintah

**Gambar 3.1.** Diagram Alir IF Bentuk umum dari pernyataan *if*

**if (*kondisi*)**

**pernyataan;**

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

**if (*kondisi*)**

**{**

**pernyataan;**

**……**

**}**



**Contoh-1** #include<stdio.h> #include<conio.h> #include<iostream.h> main( )

{

float x,y; clrscr( );

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

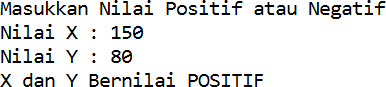
cout<<" Nilai Y : ";cin>>y;

if ((x >= 0 ) && (y>=0))

cout<<"X dan Y Bernilai POSITIF "<<endl; getch( );

}

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



**Gambar 3. 2** Hasil Contoh 1

## Pernyataan IF - ELSE

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



Perintah-1

Benar

tah-2

Perin

kondisii Salah

**Gambar 3.3** Diagram Alir if-else Bentuk umum dari pernyataan *if* -else

**if (*kondisi*)**

**perintah-1;**

**else**

**perintah-2;**

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

**...**

**if (*kondisi*)**

**{**

**perintah-1;**

**else**

**{**

**perintah-2;**

**...**

**}**

**}**

#### Contoh

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

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

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

main( )

{

float jumbel,hrg; char bonus[15]; float diskon;

clrscr( ); puts("");

cout<<" Harga Barang : " ; cin>> hrg;

cout << " Jumlah belinya : "; cin>> jumbel; if(jumbel>=15)

{ diskon = 0.15 \* hrg ; strcpy(bonus, "PAYUNG");

}

else

{ diskon =0 ;

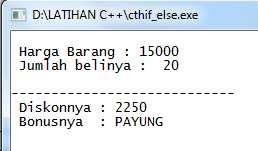
strcpy(bonus, "Tidak Dapat");

}

cout<<"\n "<<endl;

cout<<" Diskonnya : "<<diskon<<endl; cout<<" Bonusnya : "<<bonus<<endl; getch();

}

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

**Gambar 4. 4** Hasil Contoh 2

## Pernyataan NESTED IF

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

**if(syarat)**

**{**

**if(syarat)**

**… perintah; else**

**… perintah;**

**}**

**else**

**{**

**if(syarat)**

**… perintah; else**

**… perintah;**

**}**

#### Contoh

Suatu perusahaan menjual pakaian dengan ketentuan sebagai berikut:

Jika kode baju=1 maka Merk Baju = H&R, dengan ukuran baju=S,maka harganya 45000, Jika ukuran baju=M, maka harganya 60000, selain itu harganya = 0.

Jika kode baju=2 maka Merk Baju = Adidas, dengan ukuran baju=S, maka harganya 65000, Jika ukuran Baju=M, maka harganya 75000, selain itu harganya = 0.

Selain kode baju diatas, maka salah kode

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

main( )

{

char kode,ukuran,merk[15]; long harga=0;

clrscr( );

cout<<"Kode Baju : ";cin>>kode; cout<<"Ukuran : ";cin>>ukuran;

if (kode = ='1')

{

strcpy(merk,"H & R");

if (ukuran= ='S' || ukuran = ='s') harga=45000;

else if (ukuran= ='M' || ukuran = ='m') harga=60000;

else

harga = 0;

}

else if (kode=='2')

{

}

else

strcpy(merk," Adidas");

if (ukuran=='S' || ukuran == 's') harga=65000;

else if (ukuran= ='M' || ukuran = ='m') harga=75000;

else

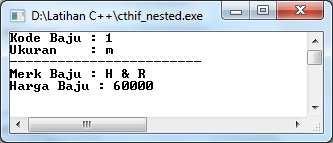
harga = 0;

cout<<”Salah Kode Baju”<<endl; cout<<" "<<endl;

cout<<"Merk Baju : "<<merk<<endl; cout<<"Harga Baju : "<<harga<<endl; getch();

}

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



**Gambar 3. 5** Hasil Contoh 3

## Pernyataan IF – ELSE Majemuk

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

Bentuk Umum Penulisannya:

**if (syarat)**

**{**

**… perintah;**

**… perintah;**

**}**

**else if (syarat)**

**{**

**… perintah;**

**… perintah;**

**}**

**else**

**{**

**… perintah;**

**… perintah;**

**}**

#### Contoh

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

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

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

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

**Contoh-4** #include<stdio.h> #include<conio.h> #include<iostream.h>

main( )

{

float pendapatan, jasa=0, komisi=0, total=0; clrscr( );

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

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

{

jasa=10000; komisi=0.1\*pendapatan;

}

else if(pendaptan<=500000)

{

jasa=20000; komisi=0.15\*pendapatan;

}

else

{

jasa=30000; komisi=0.2\*pendapatan;

}

/\* menghitung total \*/ total = komisi+jasa;

cout<<"Uang Jasa Rp. "<<jasa<<endl; cout<<"Uang Komisi Rp. "<<komisi<<endl; cout<<"============================="<<endl;

cout<<"Hasil Total Rp. "<<total<<endl; getch( );

}

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



**Gambar 3. 6** Hasil Contoh 4

## Pernyataan switch - case

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

**switch (ekspresi integer atau karakter )**

**{**

**case konstanta-1 :**

**… perintah;**

**… perintah; break;**

**case konstanta-2 :**

**… perintah;**

**… perintah; break;**

**default :**

**… perintah;**

**… perintah;**

**}**

Setiap pilihan akan dijalankan jika syarat nilai konstanta tersebut dipenuhi dan **default** akan dijalankan jika semua cabang diatasnya tidak terpenuhi.

Pernyataan *break* menunjukan bahwa perintah siap keluar dari *switch*. Jika pernyataan ini tidak ada, maka program akan diteruskan ke pilihan-pilihan yang lainnya.

**Contoh-5** #include<stdio.h> #include<conio.h> #include<iostream.h>

main( )

{

int kode; clrscr( );

cout<<"Masukkan Kode Jurusan [11/12/13] : "; cin>>kode;

switch(kode)

{

case 11 :

cout<<"Komputerisasi Akuntansi"; break;

case 12 :

cout<<"Manajemen Informatika"; break;

case 13 :

cout<<"Tehnik Komputer"; break;

default:

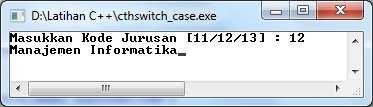
cout<<"Anda Salah Memasukan kode"; break;

}

getch( );

}

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



**Gambar 3. 7** Hasil Contoh 5

**Contoh-6** #include<stdio.h> #include<conio.h> #include<iostream.h>

main( )

{

char kode; clrscr( );

cout<<"Masukkan Kode Barang [A..C] : "; cin>>kode;

switch(kode)

{

case 'A' :

case 'a' :

cout<<"Alat Olah Raga"; break;

case 'B' :

case 'b' :

cout<<"Alat Elelktronik"; break;

case 'C' :

case 'c' :

cout<<"Alat Masak"; break;

default:

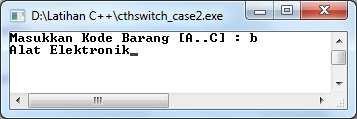
cout<<"Anda Salah Memasukan kode"; break;

}

getch( );

}

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



**Gambar 3.8** Hasil Contoh 6

**BAB 4**

**Proses Perulangan**

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

## Pernyataan for

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

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

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

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

**{**

**pernyataan / perintah; pernyataan / perintah; pernyataan / perintah;**

**}**

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

**Inisialisasi:** merupakan bagian untuk memberikan nilai awal untuk variabel- variabel tertentu.

**Syarat Pengulangan:** memegang kontrol terhadap pengulangan, karena bagian ini yang akan menentukan suatu perulangan diteruskan atau dihentikan.

**Pengubah Nilai Pencacah:** mengatur kenaikan atau penurunan nilai pencacah.

#### Contoh :

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

**Contoh-1** /\* \*/

/\* Program for - bilangan naik \*/

/\* \*/

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

{

int a; clrscr( );

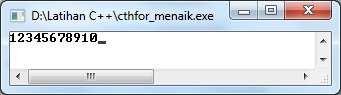


for(a = 1; a <= 10; ++a) cout>>a;

getch( );

}

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



**Gambar 4. 1** Hasil Contoh 1

**Contoh-2** /\* \*/

/\* Program for - bilangan turun \*/

/\* \*/

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

#include<iostream.h> main( )

{

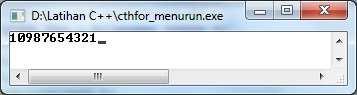
int a; clrscr( );

for(a = 10; a >= 1; --a) cout<<a;

getch( );

}

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



**Gambar 4. 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 4. 3** Hasil Contoh 3

## Pernyataan nested - for

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;

}

}

**Contoh-4** /\* \*/

/\* Program for - Nested for \*/

/\* \*/

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

main( )

{

int a, b; clrscr( );

for(a = 1; a <= 5; a++)

{

printf("\n");

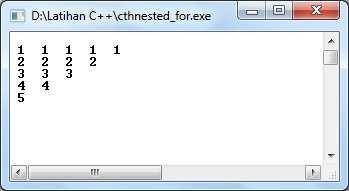
for(b = a; b <= 5; b++) printf(" %d ",a);

}

getch();

}

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



**Gambar 4. 4** Hasil Contoh 5

## Perulangan Tidak Berhingga

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

Sebagai contoh, jika penulisan perintah sebagai berikut:

**Contoh-5** /\* \*/

/\* Program for Tdk Berhingga \*/

/\* \*/

#include<stdio.h> #include<conio.h> main( )

{

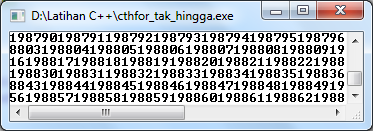
int bil; clrscr( );

for (bil = 60; bil >=10; bil++) printf("%d", bil);

getch( );

}

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



**Gambar 4. 4** Hasil Contoh 4

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

#### bil - -

Akan tetapi yang ditulis adalah :

#### bil ++

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

#### printf("%d", bil);

akan terus dijalankan.

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

## Pernyataan goto

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

**goto label;**

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

**Contoh-6** /\* \*/

/\* Program dengan pernyataan goto \*/

/\* \*/

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

main( )

{

int a, b; char lagi;

#### atas: // deklarasi label

clrscr( );

cout>>"Masukkan Bilangan = "; cin<<a; b = a % 2;

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

if (lagi == 'Y' || lagi == 'y')

goto atas; // penggunaan label

getch( );

}

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



**Gambar 4. 5** Hasil Contoh 6

## Pernyataan while

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

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

**while ( syarat )**

**Pernyataan / perintah ;**

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

**while ( syarat )**

**{**

**Pernyataan / perintah ; Pernyataan / perintah ;**

**}**

**Contoh-7** /\* \*/

/\* Program while01.cpp \*/

/\* \*/

#include <stdio.h> #include <conio.h> main( )

{

int bil=1; clrscr( ); while(bil<=10)

{

printf(" %d ",bil);

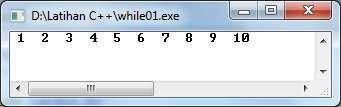
++bil;

}

getch( );

}

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



**Gambar 4. 6** Hasil Contoh 7

**Contoh-8** /\* \*/

/\* Program while02.cpp \*/

/\* \*/

#include <stdio.h> #include <conio.h> main( )

{

int bil=2; clrscr( ); while(bil<=10)

{

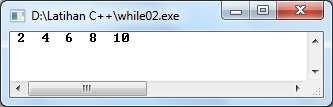
printf(" %d ",bil); bil+=2;

}

getch( );

}

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



**Gambar 4. 7** Hasil Contoh 8

## Pernyataan do - while

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

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

**do**

**pernyataan / perintah ; while ( syarat );**

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

**do**

**{**

**Pernyataan / perintah ; Pernyataan / perintah ;**

**}**

**while ( syarat );**

**Contoh-9** /\* \*/

/\* Program do - while \*/

/\* \*/

#include <stdio.h> #include <conio.h> main( )

{

int bil=2; clrscr( );

do

{

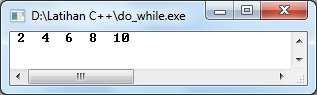
printf(" %d ",bil); bil+=2;

}

while(bil<=10); getch( );

}

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



**Gambar 4. 8** Hasil Contoh 9

## Pernyataan break

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

**Contoh-10** /\* \*/

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

/\* \*/

#include <stdio.h> #include <conio.h> main( )

{

int bil = 1; clrscr( );

do

{

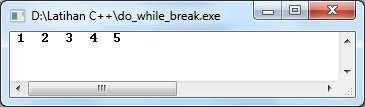
if (bil >= 6) break;

printf(" %d ",bil);

}

while(bil++); getch( );

}

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

**Gambar 4. 9** Hasil Contoh 10

**Contoh-11** /\* \*/

/\* Perulangan FOR dengan break; \*/

/\* \*/

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

main( )

{

int a=3, b=2, c=1, bil; clrscr( );

printf("Bil-A | Bil-B | Bil-C\n"); printf(" ");

for(bil=1; bil<=10; ++bil)

{

a+=b; b+=c; c+=2;

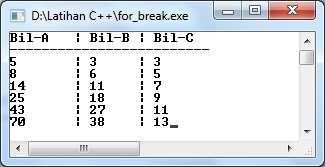
printf("\n%d \t| %d \t| %d",a, b, c); if(c==13)

break;

}

getche( );

}

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

**Gambar 4. 10** Hasil Contoh 11

## Pernyataan continue

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

**Contoh-12** /\* \*/

/\* Perulangan FOR dengan coninue \*/

/\* \*/

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

main( )

{

int bil; clrscr( );

for(bil=1; bil<=10; ++bil)

{

if(bil==6)

continue; printf(" %d ",bil);

}

getch( );

}

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



**Gambar 4. 11** Hasil Contoh 12

**BAB 5**

**Array**

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

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

* Array Berdimensi Satu.
* Array Berdimensi Dua

## Array Berdimensi Satu

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

Bentuk Umum pendeklarasian array:

**Tipe\_Data Nama\_Variabel[Ukuran]**

Keterangan :

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

**Contoh Pendeklarasian Array**

#### float Nil\_Akhir[6];

Jumlah Elemen Array Nama Variable Array Tipe data elemen array



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



Elemen Array

0 1 2 3 4 5 Subcript/Index ARRAY NIL\_AKHIR

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| elemen  1 | elemen  2 | elemen  3 | elemen  4 | elemen  5 | elemen  6 |

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

## Inisialisasi Array Berdimensi Satu

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

Tipe\_data nama\_array[jml\_elemen] = { nilai array };

**Contoh-1** /\* \*/

/\* Inisialisasi Array Dimensi 1 \*/

/\* \*/

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

main( )

{

char hari[7][10] =

{"Minggu","Senin","Selasa","Rabu","Kamis","jum'at","Sabtu"}; clrscr( );

cout<<"Nama-nama hari:"<<endl; cout<<hari[0]<<endl<<hari[1]<<endl<<hari[2]<<endl<<hari[3]

<<endl<<hari[4]<<endl; getch();

}

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



**Gambar 5.1.** Hasil Contoh-1



## Mengakses Array Berdimensi Satu

Suatu array, dapat diakses dengan menggunakan subscript atau indexnya.

Bentuk umum pengaksesan dengan bentuk :

**Nama\_Array[Subscript/Index]**

**Contoh** Nil\_Akhir[3]; Nil\_Akhir[1]; Nil\_Akhir[0];

**Contoh-2** /\* \*/

/\* Program Array Satu Dimensi \*/

/\* \*/

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

main( )

{

int i;

char nama[5][20]; float nilai1[5];

float nilai2[5];

float hasil[5]; clrscr( );

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

{

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

}

cout<<" "<<endl;

cout<<"No. Nama Siswa Nilai Nilai "; cout<<"Hasil"<<endl;

cout<<" MidTest Final "; cout<<"Ujian"<<endl;

cout<<" "<<endl;

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

{

cout<<setiosflags(ios::left)<<setw(4)<<i; cout<<setiosflags(ios::left)<<setw(10)<<nama[i]; cout<<setprecision(2)<<" "<<nilai1[i];

cout<<setprecision( 2)<<"

"<<nilai2[i]; cout<<setprecision( 2)<<"

"<<hasil[i]<<endl;

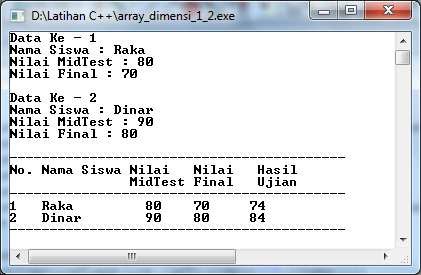
}

cout<<"

"<<endl; getch();

}

Output yang akandihasilkan, dari program contoh-2 diatas adalah:



#### Gambar 5.2.

Hasil Contoh-2

# DAFTAR PUSTAKA

Raharjo, Budi. 2015. Mudah dan Cepat Menjadi Master C++. Bandung. Informatika Kristanto, Andri, 2013. Algoritma dan Permograman dengan C++. Jakarta. Graha Ilmu Pranata, Antony.2002. Algoritma dan Pemrograman. Yogyakarta. J&J Learning

Kadir, Abdul. 2003. Pemrograman C++. Yogyakarta. ANDI

Munir, Rinaldi. 2002. Algoritma dan Pemrograman Buku 1. Bandung. Informatika