

MODUL I

PENGENALAN

A. Pengenalan C++

Berbicara tentang C++ tak lepas dari C, sebagai bahasa pendahulunya. C adalah bahasa pemrograman yang dapat dikatakan berada antara bahasa beraras rendah (bahasa yang berorientasi pada mesin) dan bahasa beraras tinggi (bahasa yang berorientasi pada manusia). Seperti diketahui bahasa tingkat tinggi mempunyai kompatibilitas yang tinggi antar *platform*.

Tujuan utama pembuatan C++ adalah untuk meningkatkan produktivitas perogram dalam membuat aplikasi. C++ dapat mengurangi kekompleksitan, terutam pada program yang besar yang terdiri dari 10.000 baris atau lebih.

B. Pengenalan Program C++

Program C++ dapat ditulis menggunakan sembarang editor teks, seperti **EDIT** (milik DOS), **WordStar**, **SideKick**, ataupun menggunakan editor bawaan dari kompiler. Program C++ biasa ditulis dengan nama ekstensi **.CPP** (dari kata C Plus Plus). Agar program bisa dijalankan (dieksekusi), program harus dikompilasi terlebih dahulu dengan menggunakan kompiler C++.

Untuk praktikum ini kita menggunakan Borland C++ versi 4.45 sebagai editor sekaligus sebagai kompiler.

1. Contoh Program C++

Sebuah contoh program C++ yang sangat sederhana dapat anda lihat dibawah ini :

Program 1.1

```
#include <iostream.h>
void main()
{
    cout << "Hai, Selamat belajar C++ \n";
}
```

Anda dapat menyimpan program diatas dengan nama **prak1.cpp**. Jika program dikompilasi dan dieksekusi, maka hasilnya berupa tulisan pada layar :

Hai, Selamat belajar C++

Penjelasan program diatas dapat dilihat pada beberapa subbab berikut.

1.1. Fungsi Main()

Fungsi adalah salah satu dasar penyusunan blok pada C++. Sebuah program C++ minimal mengandung sebuah fungsi, yaitu fungsi **main()**. Fungsi ini menjadi awal dan akhir eksekusi program C++. Sedangkan Tubuh fungsi dimulai dari tanda { hingga tanda }. Lebih spesifik lagi, semua yang terletak didalam tanda { } disebut blok.

Tanda () digunakan untuk mengapit argumen fungsi, yaitu nilai yang akan dilewatkan ke fungsi. Pada fungsi **main()** seperti pada contoh, tidak ada argumen yang diberikan. Oleh karena itu tidak ada entri di dalam ().

Kata **void** yang mendahului **main()** dipakai untuk menyatakan bahwa fungsi ini tidak memiliki nilai balik.

1.2. Pernyataan

Baris :

```
cout << "Hai, Selamat belajar C++ \n";
```

merupakan contoh sebuah pernyataan. Pada contoh diatas, pernyataan tersebut digunakan untuk menampilkan tulisan yang terletak pada sepasang tanda petik ganda ke layar. Dalam hal ini tulisan yang terletak pada sepasang tanda petik ganda disebut konstanta **string**.

Setiap pernyataan harus diakhiri tanda titik koma (;). Kealpaan dalam memberikan tanda ini kan menyebabkan kompiler memberikan pesan kesalahan selama waktu komplasi.

1.3. Mengenal cout

cout disediakan oleh C++ untuk mengarahkan data ke *standard output* (normalnya adaah layar).

Contoh :

```
cout << "Hai, Selamat Belajar C++ \n";
```

tanda << (dua buah tanda kurang dari berurutan) merupakan sebuah operator yang disebut operator “penyisipan/peletakan”. Operator ini akan mengarahkan operand (data) yang terletak disebelah kanannya ke obyek yang terletak disebelah kiri. Pada contoh di atas konstanta string diarahkan ke cout, yang memberikan hasil berupa tampilan string kelayar tersebut.

Didalam string terdapat tanda \n adalah merupakan karakter pindah baris (*newline*). Jadi bukan tanda \ dan n itulah sebabnya tulisan :

Hai, Selamat Belajar C++

Ditampilkan pada layar dan diikuti dengan pindah baris.

1.4. #include <iostream.h>

Baris :

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

bukan suatu pernyataan. Itulah sebabnya tidak ada tanda titik koma yang diperlukan. Baris tersebut menginstruksikan kepada kompiler untuk menyisipkan file lain (pada contoh diatas adalah **iostream.h**) saat program dikompilasi. Baris **#include <iostream.h>** perlu diikut sertakan pada program yang melibatkan **cout**.

1.5. clrscr() untuk menghapus layar

Perintah ini biasa dilibatkan dalam program untuk membersihkan layar. Apabila pernyataan **clrscr()** digunakan maka **#include <conio.h>** perlu disertakan pada program. Contoh :

Program 1.2 :

```
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    cout << "Hai, Selamat belajar C++ \n";
}
```

1.6. Gaya Penulisan Program

C++ memberikan keleluasaan kepada pemrogram dalam menuliskan bentuk atau gaya program. Misal pada program 1.2 dituliskan seperti berikut :

Program 1.3 :

```
#include <iostream.h>
#include <conio.h>
void main() { clrscr(); cout
    << "Hai, Selamat belajar C++ \n"
    ;
}
```

Penulisan diatas akan memberikan hasil yang sama dengan contoh sebelumnya. Namun penulisan ini tidak dianjurkan. Program hendaknya ditulis dalam bentuk yang mudah dibaca oleh siapa saja.

1.7. Komentar

Komentar merupakan bagian yang penting dalam program. Kehadirannya sangat membantu pemrogram ataupun orang lain dalam memahami program. Karena berupa penjelasan-penjelasan mengenai program atau bagian-bagian dari program.

Komentar pada C++ diawali dengan dua tanda garis miring (//). Semua tulisan yang terletak sesudah tanda (//) hingga akhir baris dengan sendirinya akan diperlakukan sebagai keterangan.

Sebuah contoh program yang menggunakan komentar :

Program 1.4 :

```
#include <iostream.h> // menyertakan file iostream.h
#include <conio.h>
void main()
{
    clrscr(); // Hapus Layar
    cout << "Hai, Selamat belajar C++ \n";
}
```

2. Elemen Dasar C++

2.1. Himpunan Karakter

Himpunan karakter pada C++ terdiri dari huruf, digit maupun simbol-simbol lainnya (termasuk spasi, karakter kontrol).

Huruf

A B C D E F G H I J K L M N O P Q R S T U V X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

Digit

0 1 2 3 4 5 6 7 8 9

Simbol dan Lain-lain

_ - + * dan sebagainya

2.2. Pengenal (Identifier)

Pengenal adalah suatu nama yang biasa dipakai dalam pemrograman untuk menyatakan :

- Variabel
- Fungsi
- Konstanta bernama
- Label
- Tipe data
- Objek

Serta hal-hal lain yang dideklarasikan atau didefinisikan oleh pemrogram.

Penamaan Pengenal

Suatu pengenal berupa satu atau beberapa karakter :

- Huruf
- Garis bawah (_)
- Digit

Dan berawalan dengan huruf atau garis bawah. Disarankan agar pemberian nama pengenal menggunakan kata yang berarti dan mudah dibaca.

Huruf Kecil dan kapital Berbeda

Pada C++, huruf kecil dan huruf kapital pada suatu pengenal dianggap berbeda. Sifat ini dikenal dengan istilah *case sensitive*. Sebab itu pengeanalan **NAMA**, **Nama** dan **nama** menyatakan tiga pengenal yang berbeda.

2.3. Kata Kunci

Kata kunci (*keyword*) adalah pengenal sistem yang mempunyai makna khusus bagi kompiler. Kata kunci tidak dapat dirubah. Karena itu kata kunci tidak dapat digunakan sebagai pengenal.

2.4. Tipe Data

Tipe data dasar pada C++ meliputi :

- char
- float
- long
- int
- double
- short
- long double

Tipe data yang berhubungan dengan bilangan bulat adalah **char**, **int**, **short** dan **long**. Sedangkan lainnya berhubungan dengan bilangan pecahan.

2.5. Variabel dan Konstanta

Data pada C++ tersusun dari :

- Variabel
- Konstanta

Variabel digunakan dalam program untuk menyimpan suatu nilai, nilai yang ada padanya dapat diubah selama eksekusi program berlangsung.

Mendeklarasikan dan Mendefinisikan Variabel

Variabel yang akan digunakan dalam program haruslah dideklarasikan terlebih dahulu. Pengertian deklarasi disini berarti mengenalkan sebuah pengenalan ke program dan menentukan jenis data yang disimpan didalamnya.

Bentuk pendefinisian variabel :

tipe daftar_variabel

Menentukan Tipe Variabel

Jika variabel hendak dipakai untuk menyimpan data bilangan bulat saja, maka pilihannya adalah tipe bilangan bulat (seperti **int** , **long**). Jika variabel hendak dipakai untuk data bilangan pecahan, maka variabel harus didefinisikan bertipe bilangan pecahan (seperti **float**).

Memberikan Nilai ke Variabel

Bentuk pernyataan yang digunakan untuk memberikan nilai ke variabel yang telah dideklarasikan atau didefinisikan :

variabel = nilai

Pernyataan diatas sering disebut sebagai pernyataan penugasan.

Inisialisai Variabel

Adakalanya dalam penulisan program, variabel langsung diberi nilai setelah didefinisikan. Sebagai contoh :

```
int jumlah;
jumlah = 10;
```

Dua pernyataan seperti diatas sebenarnya dapat disingkat melalui pendefinisian yang disertai penugasan nilai, sebagai berikut :

```
int jumlah = 10;
```

Contoh program :

```
/*-----*
/* contoh 1.1 : inisialisasi variabel dengan      *
/*          nilai konstan                        *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int jumlah = 10;           // inisialisasi
    float harga_per_unit = 17.5; // inisialisasi
    clrscr();
    cout << "Isi Jumlah          = "
         << jumlah << "\n";
    cout << "Isi harga per per unit = "
         << harga_per_unit << "\n";
}
```

Hasil eksekusi :

Isi Jumlah = 10
Isi harga per unit = 17.5

Pemberian inisialisasi terhadap suatu variabel dengan menggunakan ungkapan juga dapat diperkenalkan. Sebagai contoh :

```
float duaphi = 2 * 3.14;

Contoh Program :

/*-----*
/* Contoh 1.2 : Contoh Inisialisasi variabel dengan *
/*          suatu ungkapan                      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    float dupahi = 2 * 3.14; //inisialisasi dengan ungkapan

    clrscr();
    cout << "Isi duaphi = " << duaphi << '\n';
}
```

Hasil eksekusi :

Isi duaphi = 6.28

Pendefinisian dapat pula dilakukan dimana saja, seperti :

```
int i = 10;
cout << "Nilai i = " << i << '\n';
int j = 77;
cout << "Nilai j = " << j << '\n';

Contoh program :

/*-----*
/* Contoh 1.3 : Pendefiniasian variabel dimana saja *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int i = 10; // pendefinisian dan inisialisasi
    clrscr();
    cout << "Nilai i = " << i << '\n';
    int j = 77;
    cout << "Nilai j = " << j << '\n';
}
```

Hasil Eksekusi :

Nilai i = 10
Nilai j = 77

Tugas Minggu I :

Buatlah sebuah program untuk menampilkan sebuah kalimat sebagai berikut :
STMIK TOP MARKOTOP Yogyakarta
Jurusan Teknik Informatika
(Simpanlah program dengan nama Tugas1.cpp)

MODUL II
OPERATOR DAN UNGKAPAN

2.1. Pengantar Operator dan Ungkapan

Operator merupakan simbol yang biasa dilibatkan dalam program untuk melakukan suatu operasi atau manipulasi. Sebagaimana operator C++ tergolong sebagai operator *binary*, yaitu operator yang dikenakan terhadap dua buah nilai (*operand*).

Contoh :

a + b

simbol “ + ” merupakan *operand* untuk melakukan penjumlahan dari a dan b. Karena operator penjumlahan melibatkan dua *operand*, operator penjumlahan tergolong sebagai operator *binary*.

Contoh lain :

-c

simbol “ - ” (minus) merupakan *unary*, karena hanya memiliki sebuah *operand* (yaitu c pada contoh diatas).

Ungkapan (ekspresi) dalam C++ dapat berupa :

- Pengenal
- Konstanta
- Diantara kombinasi elemen diatas dengan operator

Contoh ungkapan :

3 + 2 - 1

Pada ungkapan diatas, 3,2 dan 1 merupakan operand dan simbol “ + ” serta “ - ” adalah operator. Nilai ungkapan sendiri adalah hasil penjumlahan 3 dan 2, dikurangi 1.

2.2. Operator Aritmatika

Operator untuk aritmatika yang tergolong sebagai operator *binary*. Contoh penggunaan operator aritmatikamisalnya untuk memperoleh nilai diskriminan dari suatu persamaan kuadrat.

$$d = b^2 - 4ac$$

untuk mengimplementasikan contoh diatas adalah seperti berikut :

$d = b * b - 4 * a * c ;$

Contoh program :

```
/*-----*
/* contoh 2.1 : Contoh pemakaian operator      *
/*          Aritmatika                          *
/*-----*
#include <iostream.h>
#include <conio.h>

void main()
{
    int a, b, c, d;
    clrscr();
    a = 5;
    b = 600;
    c = 5;

    d = b * b - 4 * a * c;
    cout << " d = " << d << '\n';
}
```

Hasil eksekusi :

d = 32220

Operator aritmatika mempunyai prioritas pengerjaan. Prioritas yang tinggi akan diutamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki prioritas yang lebih rendah. Urutan prioritas dapat dilihat dalam tabel berikut ini :

Operator	Prioritas
+ -- (Khusus yang berkedudukan sebagai awalan)	Tertinggi
- (Unary Minus)	
* / %	
+ -	Terendah

Jika operator memiliki prioritas yang sama, operator sebelah kiri akan diutamakan untuk dikerjakan terlebih dahulu.

Tanda kurung biasa digunakan untuk merubah urutan pengerjaan. Misalnya : $x = (2 + 3) * 2$; akan memeberikan nilai 10 ke x, sebab $2 + 3$ dikerjakan terlebih dahulu dan hasilnya baru dikalikan dengan 2.

Contoh program :

```
/*-----*
/* Contoh 2.2 : Penggunaan kurung untuk mengatur *
/*          prioritas pengerjaan terhadap suatu *
/*          operasi *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr()
    int x ;
    x = 2 + 3 * 2 ;
    cout << " x = " << x << '\n';
    x = (2 + 3) * 2 ;
    cout << " x = " << x << '\n';
}
```

Hasil eksekusi :

8

12

2.2. Operator Sisa Pembagian

Operator sisa pembagian (operator modulus) yang berupa %. Operator ini diterapkan pada operand bertipe integer. Untuk lebih jelasnya perhatikan contoh berikut :

7 % 2 → 1	Sisa pembagian bilangan 7 dengan 2 adalah 1
6 % 2 → 0	Sisa pembagian bilangan 6 dengan 2 adalah 0
8 % 3 → 2	Sisa pembagian bilangan 8 dengan 3 adalah 2

Contoh program :

```
/*-----*
/* Contoh 2.3 : Melihat sisi pembagian dengan *
/*          menggunakan operator %. *
/*-----*
#include <iostream.h>
#include <conio.h>

void main()
{
    clrscr();
    cout << 5 % 7 << '\n'; //sisa 5
    cout << 6 % 7 << '\n'; //sisa 6

    cout << 7 % 7 << '\n'; //sisa 0
    cout << 8 % 7 << '\n'; //sisa 1
    cout << 9 % 7 << '\n'; //sisa 2
}
```

Hasil eksekusi :

5

6

0

1

2

Kegunaan operator % diantaranya bisa dipakai untuk menentukan suatu bilangan bulat termasuk ganjil atau genap.

2.3. Operator Penurunan dan Peningkatan

Kedua operator ini digunakan pada operand bertipe bilangan bulat. Operator peningkatan digunakan untuk menaikkan nilai variabel sebesar satu, sedangkan operator penurunan dipakai untuk menurunkan nilai variabel sebesar satu. Sebagai contoh :

x = x + 1 ;
y = y - 1 ;
bisa ditulis menjadi :

++ x ;
-- y ;
atau :

x ++ ;
y -- ;

Peningkatan dibelakang

Efek peletakkan tanda ++ dibelakang variabel ditunjukkan pada program berikut :

Contoh program :

```
/*-----*  
/* Contoh 2.4 : Pemakaian operator peningkatan di *  
/*          belakang variabel *  
/*-----*  
#include <iostream.h>  
#include <conio.h>  
void main()  
{  
  int r = 10;  
  int s;  
  clrscr();  
  s = 10 + r++;  
  cout << " r = " << r << "\n" ;  
  
  cout << "s = " << s << "\n" ;  
}
```

Hasil eksekusi :

```
  r = 11  
  s = 20
```

Pada contoh diatas s diisi dengan penjumlahan nilai 10 dan r. Dengan demikian s akan bernilai 20. setelah s diisi dengan 20, nilai r baru dinaikan karena operator ++ ditulis dibelakang r. Disebut *post-increment* yang artinya dinaikkan dibelakang setelah penjumlahan antara r dan 10 dilaksanakan.

Peningkatan di Depan

Efek peletakkan tanda ++ di depan variabel ditunjukkan pada program berikut ini :

Contoh program :

```
/*-----*  
/* Contoh 2.4 : Pemakaian operator peningkatan di *  
/*          belakang variabel *  
/*-----*  
#include <iostream.h>  
#include <conio.h>  
void main()  
{  
  int r = 10;  
  int s;  
  clrscr();  
  s = 10 + ++r ;  
  cout << " r = " << r << "\n" ;  
  cout << "s = " << s << "\n" ;  
}
```

Hasil eksekusi :

```
  r = 11  
  s = 21
```


Pada contoh ini, nilai r mula-mula dinaikan terlebih dahulu karena operator ++ ditempatkan didepan r. Disebut *pre-increment* kemudian nilainya dijumlahkan dengan 10 dan diberikan ke s. Dengan demikian s bernilai 21 dan r sama dengan 11.

2.4. Operator Majemuk

Operator majemuk digunakan untuk memendekkan penulisan operasi penugasan semacam :

x = x + 2 ;
y = y * 4 ;

menjadi :

x += 2;
y *= 4;

Contoh program :

```
/*-----*  
/* Contoh 2.5 : penggunaan operator majemuk      *  
/*-----*  
#include <iostream.h>  
#include <conio.h>  
void main()  
{  
    int x = 2; // Mula-mula x bernilai 2  
    clrscr();  
    cout << "x = " << x << '\n' ;  
    x += 3 ;  
    cout << "Setelah x += 3, x = " << x << '\n' ;  
    x *= 2 ;  
    cout << "Setelah x *= 2, x = " << x << '\n' ;  
}
```

Hasil eksekusi :

x = 2
Setelah x += 3, x = 5
Setelah x += 3, x = 5

2.5. Operator Kondisi

Operator kondisi biasa dipakai untuk mendapatkan sebuah nilai dari dua buah kemungkinan, berdasarkan suatu kondisi. Format pemakaiannya :

ungkapan1 ? ungkapan 2 : ungkapan 3

Contoh program :

```
/*-----*  
/* Contoh 2.7 : Penggunaan operator kondisi untuk *  
/*           memperoleh bilangan terkecil         *  
/*           diantara dua buah bilangan           *  
/*-----*  
#include <iostream.h>  
#include <conio.h>  
void main()  
{  
    int bil1, bil2, minim;  
    clrscr();  
    bil1 = 53;  
    bil2 = 6;  
    minim = bil1 < bil2 ? bil1 : bil2;  
    cout << " Bilangan terkecil = " << minim << '\n';  
}
```

Bilangan terkecil = 6

Hasil eksekusi :

`minim = bil1 < bil2 ? bil1 : bil2;`
akan menyebabkan `minim` bernilai `bil1` kalau ungkapan :
`bil1 < bil2`
bernilai benar. Untuk keadaan sebaliknya, `minim` akan bernilai `bil2`.

2.6. Ungkapan Kondisi

Ungkapan adalah ungkapan yang menjadi dasar bagi pernyataan berkondisi (misalnya `if`). Hasil ungkapan berupa 1 kalau ungkapan bernilai benar dan ungkapan berupa 0 kalau ungkapan bernilai salah.

Oprator Relasi

Operator biasa digunakan untuk membandingkan dua buah nilai. Macam operator relasi dapat dilihat dalam tabel berikut :

Operator	Keterangan
<code>==</code>	Sama dengan (bukan penugasan)
<code>!=</code>	Tidak sama dengan
<code>></code>	Lebih dari
<code><</code>	Kurang dari
<code>>=</code>	Lebih dari atau sama dengan
<code><=</code>	Kurang dari atau sama dengan

```
Contoh program :
/*-----*
/* Contoh 2.6 : untuk menunjukkan nilai *
/*      hasil ungkapan kondisi      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int nilai;
    clrscr();
    nilai = 3 > 2 ; // hasil ungkapan : benar
    cout << "Nilai = " << nilai << endl;

    nilai = 2 > 3 ; // hasil ungkapan : salah
    cout << "Nilai = " << nilai << endl;
}
```

Hasil eksekusi :

Nilai = 1
Nilai = 0

Agar tidak salah dalam menuliskan suatu ungkapan, pengetahuan tentang prioritas operator perlu diketahui.

Contoh 1 :

`a = b = c`
pada pernyataan diatas, operator yang dilibatkan (`=`) mempunyai sifat pengerjaan dimulai dari kanan. Berarti :
`b = c`
akan dikerjakan terlebih dahulu, barulah kemudian mengerjakan :

`a = b`
Contoh 2 : `x = 2 * 3 * 4 ;`
pada pernyataan diatas, `2 * 3` akan dikerjakan terlebih dahulu, barulah kemudian mengerjakan perkalian hasil 6 dengan 4. Adapun prioritas `=` lebih rendah dari `*`, maka `2 * 3 * 4` dikerjakan lebih dahulu. Selanjutnya hasilnya baru diberikan ke `x`.

2.7. Fungsi Pustaka

Dalam melakukan operasi seperti memperoleh akar kuadrat ataupun memperoleh logaritma alamiah dari suatu nilai. Pada C++ memang tidak terdapat operator-operator yang khusus untuk melaksanakan operasi-operasi seperti itu. Tetapi tidak berarti C++ tidak dapat melakukan operasi itu. C++ menyediakan sejumlah fungsi pustaka (*library functions*) yang dirancang untuk memenuhi solusi dari berbagai persoalan.

Misalkan kita akan menghitung sebuah akar kuadrat, pemrogram bisa menggunakan fungsi **sqrt()**. Seperti contoh program berikut :

Contoh program :

```

/*-----*
/* Contoh 2.7 : Pemakaian pustaka fungsi sqrt()      *
/*-----*
#include <iostream.h>
#include <conio.h>
#include <math.h> // Perlu disertakan untuk fungsi sqrt()
void main ()
{
    clrscr();
    cout << "Akar dari 27 = " << sqrt(27) << "\n";
}

```

Hasil eksekusi :

Akar dari 27 = 5.196152

Jika program ingin menggunakan fungsi pustaka, perlulah untuk mencatumkan deklarasi dari fungsi bersangkutan. Untuk keperluan ini program mesti menyertakan baris :

#include <nama_file>

degan *nama_file* adalah nama *header*, yaitu file yang berakhiran **.h**. sebagai contoh program diatas menyertakan **#include <math.h>** disebabkan file *header* tersebut berisi deklarasi (prototipe) dari fungsi **sqrt()**.

Tugas Minggu II :

1. Buatlah program untuk menghitung luas dan keliling lingkaran
 2. Buatlah program untuk menentukan sisa pembagian menggunakan operator %
- (simpan program untuk no.1 -> tug21.cpp dan no.2 -> tug22.cpp)

MODUL III

OPERASI DASAR MASUKAN DAN KELUARAN

3.1. Cout

Dengan cout pemrogram dapat meletakkan suatu informasi ke *standart output* (normalnya berupa layar). Sebagai contoh, pernyataan berikut akan menampilkan tulisan “Pilihan Anda salah” diikuti dengan bunyi bel (speker) :

```
cout << "Pilihan Anda salah !\a\n";
```

Untuk lebih jelasnya dapat memerhatikan program dibawah ini :

Contoh program :

```

/*-----*
/* Contoh 3.1 : Memperkenalkan cout dan      *
/*          membunyikan bel dengan          *
/*          karakter \a                      *
/*-----*
#include <iostream.h>
void main()
{
    cout << "Pilihan Anda Salah !\a";
}

```

3.2. Manipulator

Manipulator umumnya digunakan untuk mengatur tampilan data. Misalnya untuk mengatur agar suatu nilai ditampilkan dengan lebar 10 karakter dan diatur rat kanan terhadap lebar tersebut.

Manipulator endl

Manipulator **endl** digunakan untuk menyisipkan karakter *newline*. Dengan kata lain manipulato ini identik dengan “\n”.

Contoh program berikut menunjukkan penggunaan endl.

```
Contoh program :
/*-----*
/* Contoh 3.2 : Menampilkan 3 jumlah barang dan      *
/*           menggunakan manipulator endl             *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int jumbar1 = 150;
    jumbar2 = 23;
    jumbar3 = 1401;
    clrscr(); // Hapus layar
    cout << "Barang 1 = " << jumbar1 << endl;
    cout << "Barang2 = " << jumbar2 << endl;
    cout << "Barang3 = " << jumbar3 << endl;
}
```

Hasil eksekusi :

Barang 1 = 150
Barang 2 = 23
Barang 3 = 1401

Manipulator setw()

Manipulator **setw()** bermanfaat untuk mangatur lebar dari suatu tampilan data. Untuk lebih jelasnya dapat dilihat dalam program berikut :

```
Contoh program :
/*-----*
/* Contoh 3.3 : Manampilkan 3 buah barang dengan      *
/*           menggunakan manipulator setw()           *
/*-----*
#include <iostream.h>
#include <iomaip.h> // Untuk manipulator setw()
#include <conio.h>
void main()
{
    int jumbar1 = 150;
    jumbar2 = 23;
    jumbar3 = 1401;

    clrscr(); // Hapus layar
    cout << "Barang 1 = " << setw(4) << jumbar1 << endl;
    cout << "Barang2 = " << setw(4) << jumbar2 << endl;
    cout << "Barang3 = " << setw(4) << jumbar3 << endl;
}
```

Hasil eksekusi :

Barang 1 = 150
Barang 2 = 23
Barang 3 = 1401

Dengan menggunakan `setw()`, terlihat hasil eksekusi adalah rata kanan. Ini dapat di bedakan dengan hasil eksekusi pada contoh 3.2 yang rata kiri.

Manipulator `setfill()`

Manipulator `setfill()` berguna untuk mengatur karakter yang dipakai memenuhi bagian *field* yang ditentukan `setw()`, yang tidak dipakai untuk menampilkan data.

Untuk lebih jelas dapat melihat program berikut :

Contoh program :

```
/*-----*
/* Contoh 3.4 : Pemakaian setfill()      *
/*-----*
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
void main()
{
    float harga = 123.45

    clrscr();
    cout << " setw(8) << harga << endl;
    cout << setfill('*') ; // Karakter pemenuh -> *
    cout << setw(8) << harga << endl;
}
```

Hasil eksekusi :

123.45

**123.45

Pada program diatas *123.45* ditampilkan dengan lebar 8 karakter dan didahului oleh 2 karakter spasi. Apabila terdapat pernyataan `setfill('*')` maka 2 karakter spasi diisi oleh karakter ‘ * ‘ .

Pemakaian `setiosflag()`

Manipulator `setiosflag()` merupakan manipulator yang dapat dipakai untuk mengontrol sejumlah tanda format yang tercantum dalam tabel berikut :

Tanda Format	Keterangan
<code>ios::left</code>	Menyetel rata kiri terhadap lebar <i>feeld</i> yang diatur melalui <code>setw()</code>
<code>ios::righ</code>	Menyetel rata kanan terhadap lebar <i>feeld</i> yang diatur melalui <code>setw()</code>
<code>ios::scientific</code>	Menformat keluaran dalam notasi eksponensial
<code>ios::fixed</code>	Menformat keluaran dalam bentuk notasi desimal
<code>ios::dec</code>	Menformat keluaran dalam basis 10 (desimal)
<code>ios::oct</code>	Memformat keluaran basisi 8 (oktal)
<code>ios::hex</code>	Memformat huruf dalam basis 16 (heksadesimal)
<code>ios::uppercase</code>	Memformat huruf pada notasi heksadesimal dalam bentuk huruf kapital
<code>ios::showbase</code>	Manmpilkan awalan 0x untuk bilangan heksadesimal atau 0 (nol) untuk bilangan oktal
<code>ios::showpoint</code>	Menampilkan titik desimal pada bilangan pecahan yang tidak memiliki bagian pecahan
<code>ios::showpos</code>	Untuk menampilkan tanda + pada bilangan positif

Manipulator setprecision()

Apabila anda bekerja dalam bilangan pecahan, Anda juga mengatur jumlah digit pecahan yang ingin ditampilkan. Hal ini dapat dilakukan dengan menggunakan manipulator **setprecision()**. Bentuknya :

setprecision(n)
dengan *n* menyatakan jumlah digit pecahan yang diinginkan.

Sebagai contoh :

cout << setprecision(2) << 123.56789 << endl;
akan menampilkan :

123.57
bagian pecahan terdiri atas dua digit sesuai dengan argumen pada **setprecision()**.

Contoh Program :

```
/*-----*
/* Contoh 3.5 : Menunjukkan efek manipulator *
/*          setprecisoin()                      *
/*-----*
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
    float nilai = 123.45;
    clrscr();
    cout << setiosflags(ios::fixed);
    cout << setprecision(0) << nilai << endl;
    cout << setprecision(1) << nilai << endl;
    cout << setprecision(2) << nilai << endl;
    cout << setprecision(3) << nilai << endl;
    cout << setprecision(4) << nilai << endl;
    cout << setprecision(5) << nilai << endl;
    cout << setprecision(6) << nilai << endl;
    cout << setprecision(7) << nilai << endl;
}
```

Hasil eksekusi : _____➔

123
123.4
123.45
123.450
123.4500
123.45000
123.449997
123.4499969

Contoh berikut menunjukkan hasil bila **setw()** dipakai tetapi **ios::fixed** tidak diikuti sertakan.

```
/*-----*
/* Contoh 3.6 : Menunjukkan efek setprecision() *
/*          tanpa tanda format ios::fixed        *
/*-----*
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main()
{
    float nilai = 123.45;
    clrscr(); // Hapus layar
    cout << setprecision(0) << nilai << endl;
    cout << setprecision(1) << nilai << endl;
    cout << setprecision(2) << nilai << endl;
    cout << setprecision(3) << nilai << endl;
    cout << setprecision(4) << nilai << endl;
    cout << setprecision(5) << nilai << endl;
    cout << setprecision(6) << nilai << endl;
    cout << setprecision(7) << nilai << endl;
}
```

Hasil eksekusi :

1e+02
1e+02
1.2e+02
123
123.4
123.45
123.45
123.45

3.3. cin

Obyek **cin** bermanfaat untuk untuk membaca data dari *standart input* (normalnya adalah keyboard).

cin dengan sebuah variabel

Bentuk pernyataan **cin** untuk membaca data dari keyboard dan meletakkan ke sebuah variabel variabel bernama *var* :

```
cin >> var
```

Contoh program yang menunjukkan penmakaian **cin** untuk membaca data bertipe **int** dan **float**.

```
/*-----*
/* Contoh 3.7 : Membaca data bertipe int dan *
/*          float dari keyboard          *
/*-----*
#include <iostream.h>
void main ()
{
    int bil_x ; // Definisi bilangan bulat
    float bil_y ; // Definisi bilangan pecahan
    cout << "Masukkan sebuah bilangan bulat = " ;
    cin >> bil_x ;
    cout << "Masukkan sebuah bilangan pecahan = " ;
    cin >> bil_y;

    cout << " Bilangan Bulat = " << bil_x << endl;
    cout << " Bilangan Pecahan = " << bil_y << endl;
}
```

Hasil eksekusi :

```
Masukkan sebuah bilangan bulat = 123
Masukkan sebuah bilangan pecahan =
23.1
Bilangan Bulat = 123
```

Tampak bahwa bentuk perintah untuk membaca data bertipe **int** ataupun **float** sama saja. Hal ini berlaku untuk semua jenis data dasar (**char**, **int**, **long**, **float** ataupun **double**).

Salah satu manfaat dari adanya fasilitas pemasukkan dari keyboard adalah memungkinkan untuk membuat program yang membaca data yang berubah-ubah.

cin dengan lebih dari satu variabel

Bentuk cin dapat juga sebagai berikut :

```
cin >> bil_x >> bil_y ;
```

Pada contoh ini kedua bilangan dapat bilangan dapat dimasukkan dari keyboard dengan pemisah berupa spasi, tab atau enter.

Contoh program :

```
/*-----*
/* Contoh 3.8 : Untuk membaca data dengan lebih *
/*          dari satu variabel          *
/*-----*
#include <iostream.h>
void main ()
{
    int bil_x ; // Definisi bilangan bulat
    float bil_y ; // Definisi bilangan pecahan
    cout << "Masukkan sebuah bilangan bulat dan " << endl;
    cout << "Masukkan sebuah bilangan pecahan " << endl ;
    cin >> bil_x >> bil_y;

    cout << " Bilangan Bulat = " << bil_x << endl;
    cout << " Bilangan Pecahan = " << bil_y << endl;
}
```

Hasil eksekusi :

```
Masukkan sebuah bilangan bulat dan
Masukkan sebuah bilangan pecahan 20 23.2 ↵
Bilangan Bulat = 20
Bilangan Pecahan 23.2
```

cin untuk membaca karakter

cin juga dapat dipakai untuk membaca sebuah karakter. Sebagai contoh :

```
/*-----*
/* Contoh 3.9 : Membaca karakter dengan cin      *
/*-----*
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
void main ()
{
    char karakter;
    clrscr();
    cout << "Masukkan sebuah karakter ";
    cin >> karakter;
    cout << "Anda mengetik " << karakter << endl;
    cout << "Masukkan sebuah karakter ";
    cin >> karakter;
    cout << "Anda mengetik " << karakter << endl;
}
```

Hasil eksekusi :

```
Masukkan sebuah karakter a ↵
Anda mengetik a
Masukkan sebuah karakter b ↵
Anda mengetik b
```

Tampak pada contoh program diatas, untuk setiap karakter yang dimasukkan harus diakhiri dengan enter (↵). Penekanan enter pada pemasukkan sebuah karakter terkadang tidak dikehendaki oleh pemakai. Jika demikian, anda dapat menggunakan fungsi **getch()** ataupun **getche()**.

3.4. Fungsi getch() dan getche()

Fungsi **getch()** dan **getche()** berguna untuk membaca sebuah karakter tanpa perlu menekan enter. Selain itu, fungsi ini juga dapat dipakai untuk membaca tombol seperti spasi, tab ataupun enter.

Bentuk pemakaiannya :

```
karakter = getch()
karakter = getche()
```

Perbedaan kedua fungsi ini adalah :

- getch() : tidak menampilkan karakter dari tombol yang ditekan
- getche() : menampilkan karakter dari tombol yang ditekan

Contoh program :

```
/*-----*
/* Contoh 3.10 : Membaca karakter dengan getch() *
/*-----*
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
void main()
{
    char karakter;
    clrscr();
    cout << "Masukkan sebuah karakter ";
    karakter = getch();
    cout << "Anda mengetik " << karakter << endl;
    cout << "Masukkan sebuah karakter ";
    karakter = getch();
    cout << "Anda mengetik " << karakter << endl;
}
```

```
Masukkan sebuah karakter a  «« Tidak ditampilkan
Anda mengetik a
Masukkan sebuah karakter b  «« Tidak ditampilkan
Anda mengetik b
```


Sebuah **getch()** bisa pula dimanfaatkan untuk menunggu sembarang tombol ditekan. Pada keadaan seperti ini tidak perlu diletakkan ke variabel.

Contoh program :

```
/*-----*
/* Contoh 3.11 : getch() untuk memebaca sembarang *
/*          tombol                                *
/*-----*
#include <iostream.h >
#include <conio.h>
void main()
{
    cout << "Tekanlah sembarang tombol" << endl;
    cout << "Untuk mengakhiri program ini " ;
    getch();
}
```

Tugas Minggu III:

Buatlah program yang semua datanya diinputkan dari keyboard untuk memasukkan tanggal, bulan dan tahun lahir anda dimana semua dibuat rata kiri.

Contoh :

```
Saya Lahir pada
Tanggal :  28
Bulan   :  11
Tahun   : 1982
(simpan dengan nama tugas3.cpp)
```

MODUL IV
PERNYATAAN DASAR

4.1. Pernyataan if

Pernyataan if dapat dipakai untuk mengambil keputusan berdasarkan suatu kondisi. Bentuk pernyataan ada dua macam :

- if
- if else

Pernyataan if sederhana

Pernyataan if paling sederhana berbentuk :

if (kondisi)
 pernyataan

- Kondisi digunakan untuk menentukan pengambilan keputusan
- Pernyataan dapat berupa sebuah pernyataan-pernyataan majemuk.

Penerapan if misalnya untuk menentukan seseorang boleh atau tidak menonton pertunjukkan bioskop. Kondisi yang digunakan seseorang boleh menonton kalau sudah berusia 17 tahun.

Contoh program :

```
/*-----*
/* Contoh 4.1 : Penggunaan if dalam pengambilan *
/*          keputusan                                *
/*-----*
#include <iostream.h>
#include <conio.h>
```

```

void main()
{
    int usia;
    clrscr(); // Hapus layar
    cout << "Berapa usia anda ? ";
    cin >> usia;
    if (usia < 17)
        cout << "Anda tidak diperkenankan menonton" << endl;
}

```

Hasil eksekusi :

Barapa usia anda ? 16 ↵

Anda Tidak diperkenankan menonton

Apabila program dieksekusi lagi untuk memasukkan usia diatas 17 maka :

Tampak diatas bila dimasukkan usia diatas 17 maka pesan tidak akan ditampilkan. Untuk mengatasi hal ini dapat dilakukan dengan menggunakan pernyataan kondisi **if else**.

Barapa usia anda ? 21 ↵

Pernyataan if else

Pernyataan **if else** mempunyai bentuk sebagai berikut :

if (kondisi)

Pernyataan 1;

else

Pernyataan 2;

Untuk **if else** kita dapat menggunakan contoh 4.1 untuk melihat perbedaan dengan if sederhana.

Contoh program :

```

/*-----*
/* Contoh 4.2 : Penggunaan if else dalam      *
/*           pengambilan keputusan           *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int usia;
    clrscr(); // Hapus layar
    cout << "Berapa usia anda ? ";
    cin >> usia;
    if (usia < 17)
        cout << "Anda tidak diperkenankan menonton" << endl;
    else
        cout << "Selamat menonton" << endl;
}

```

Hasil eksekusi :

Barapa usia anda ? 16 ↵

Anda Tidak diperkenankan menonton

Apabila kita memasukkan umur lebih dari 17 maka hasil eksekusi yang didapat adalah :

Barapa usia anda ? 21 ↵

Selamat menonton

Pernyataan if dalam if

Pernyataan **if** yang terletak dalam **if** sering disebut *nested if* atau **if** bersarang. Salah satu bentuknya adalah :

```
if (kondisi1)
    pernyataan1;
else if (kondisi2)
    pernyataan2;
else if (kondisi3)
    pernyataan3;
if (kondisiM)
    pernyataanM;
else
    pernyataanN; /*Opsional*/
```

Bentuk pernyataan if seperti ini bermanfaat untuk menyeleksi sejumlah kemungkinan tindakan. Penyeleksian dilakukan secara bertingkat.

Contoh program :

```
/*-----*
/* Contoh 4.3 : Pemakaian if bertingkat untuk *
/*          menentukan nama hari          *
/*-----*
#include <iostream.h>
#include <conio.h>
void main ()
{
    int kode_hari;
    clrscr(); // Hapus layar
    cout << "Menentukan hari " << endl;
    cout << "1 = Senin   3 = Rabu   5 = Jum'at   7 = Minggu " << endl;
    cout << "2 = Selasa  4 = Kamis  6 = Sabtu " << endl;
    cout << "Kode hari [1..7] : " ;
    cin >> kode_hari;
    // Proses seleksi;
    if (kode_hari == 1)
        cout << "Senin" << endl;
    if (kode_hari == 2)
        cout << "Selasa" << endl;
    if (kode_hari == 3)
        cout << "Rabu" << endl;
    if (kode_hari == 4)
        cout << "Kamis" << endl;
    if (kode_hari == 5)
        cout << "Jum'at" << endl;
    if (kode_hari == 6)
        cout << "Sabtu" << endl;
    if (kode_hari == 7)
        cout << "Minggu" << endl;
    else
        cout << "Kode hari salah" << endl;
}
```

Hasil eksekusi :

```
Menentukan hari
1 = Senin   3 = Rabu   5 = Jum'at   7 = Minggu "<< endl;
2 = Selasa  4 = Kamis  6 = Sabtu " << endl;
Kode hari [1..7] : 2 ↵
Selasa
```

Program diatas pertama-tama meminta kode hari dimasukkan dari keyboard. Kemudian **if** dan **else** secara bertingkat akan menyeleksi nilai tersebut dan memeberikan nama hari. Bila anda memasukkan kode hari yang salah maka :

```
Menentukan hari
1 = Senin   3 = Rabu   5 = Jum'at   7 = Minggu "<< endl;
2 = Selasa  4 = Kamis  6 = Sabtu  "<< endl;
Kode hari [1..7] : 9 ↵
Kode hari salah
```

4.2. Pernyataan switch

switch adalah pernyataan yang digunakan untuk menjalankan salah satu pernyataan dari beberapa kemungkinan pernyataan, berdasarkan nilai dari sebuah ungkapan dan nilai penyeleksi.

Kaidah umum pernyataan switch :

```
switch (ungkapan)
{
    case ungkapan1;
        pernyataan_1;
        break;
    case ungkapan2;
        pernyataan_2;
        break;
    .....
    default :                /*Opsinal*/
        pernyataan_x;       /*Opsinal*/
}
```

Pada pernyataan switch, *ungkapan* dapat berupa ungkapan, konstanta ataupun variabel. Adapun *ungkapan1*, *ungkapan2* dan seterusnya dapat berupa sembarang konstanta bertipe **int** atau **char**.

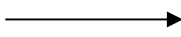
Contoh program :

```
/*-----*
/* Contoh 4.4 : Pemakaian switch untuk menentukan */
/*          nama hari                               */
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int kode_hari;
    clrscr(); // Hapus layar
    cout << "Menentukan Hari" << endl;
    cout << "1 = Senin   3 = Rabu   5 = Jum'at   7 = Minggu "<< endl;
    cout << "2 = Selasa  4 = Kamis  6 = Sabtu  "<< endl;
    cout << "Kode hari [1..7] : " ;
    cin >> kode_hari;
    // Proses seleksi dengan switch
    switch (kode_hari)
    {
        case 1:
            cout << "Senin" << endl;
            break;
        case 2 :
            cout << "Selasa" << endl;
            break;
        case 3 :
            cout << "Rabu" << endl;
            break;
        case 4 :
            cout << "Kamis" << endl;
            break;
```

```

case 5 :
    cout << "Jum'at" << endl;
    break;
case 6 :
    cout << "Sabtu" << endl;
    break;
case 7 :
    cout << "Minggu" << endl;
    break;
default :
    cout << "Kode hari salah" << endl;
    break;
} // akhir switch
}

```

Hasil eksekusi : 

Menentukan Hari
1 = Senin 3 = Rabu 5 = Jum'at 7 = Minggu
2 = Selasa 4 = Kamis 6 = Sabtu
Kode hari [1..7] : 2 ↴
Selasa

Kehadiran **break** pada setiap **case** sangat penting. Sebab **break** akan menyebabkan keluar dari **switch**.

4.3. Pernyataan while

Pernyataan **while** merupakan salah satu pernyataan yang berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali. Bentuk pernyataan **while**

while (*ungkapan*);
 pernyataan;

Pernyataan bisa berupa pernyataan majemuk, sehingga bentuknya bisa seperti :

while (*ungkapan*);
{
 pernyataan1;
 pernyataan2;

 pernyataanN;
}

Bagian pernyataan yang mengikuti **while** akan dieksekusi selama *ungkapan* pada **while** bernilai benar (tidak sama dengan nol). Pengujian terhadap ungkapan **while** dilakukan sebelum bagian pernyataan.

Contoh program :

```

/*-----*
/* Contoh 4.5 : pemakaian while untuk      *
/* menampilkan tulisan C++                *
/* sebanyak 10 kali                        *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int i; // Sebagai variabel pencacah yang menyatakan
           // jumlah tulisan C++ yang harus ditampilkan
    clrscr(); // Hapus layar
    i = 0;    // Mula-mula diisi sama dengan nol

```

```

while ( i < 10 )
{
    cout << " C++ " << endl;
    i ++ ;      // Menaikkan pencacah sebesar 1
}

```

Hasil eksekusi :

C++
C++
C++
C++
C++
C++
C++
C++
C++
C++

Pada program diatas, variabel i bertindak sebagai pencacah yang gunanya untuk mengingat jumlah tulisa C++ yang telah ditampilkan. Itulah sebabnya mula-mula didisi dengan nol. Kemudian untuk setiap putaran, isi variabel ini dinaikkan. Oleh karena variabel i dijadikan sebagai kondisi pada **while**, suatu ketika ketika kondisi i<10 akan bernilai salah, maka **while** berakhir.

4.4. Pernyataan do-while

Format :

do
{
 pernyataan1;
 pernyataan2;

 pernyataanN;
} while (ungkapan)

Bagian *pernyataan1* hingga *pernyataanN* dijalankan secara berulang sampai *ungkapan* bernilai salah (sama dengan nol). Namn berbeda dengan **while**, pengujian *ungkapan* dilakukan dibelakang (setelah bagian *pernyataan*).

Contoh program :

```

/*-----*
/* Contoh 4.6 : pemakaian do-while untuk      *
/*             menampilkan tulisan C++         *
/*             sebanyak 10 kali                 *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int i; // Sebagai variabel pencacah yang menyatakan
           // jumlah tulisan C++ yang harus ditampilkan
    clrscr(); // Hapus layar
    i = 0;    // Mula-mula diisi sama dengan nol
    do
    {
        cout << " C++ " << endl;
        i ++ ;      // Menaikkan pencacah sebesar 1
    } while ( i < 10 );
}

```

4.5. Pernyataan for

Pernyataan **for** berguna untuk menggulang pengeksekusian terhadap satu atau sejumlah pernyataan.

Bentuk format :

for (*ungkapan1*; *ungkapan2*; *ungkapan3*)
 pernyataan;

Contoh program :

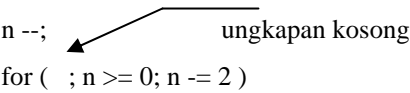
```
/*-----*
/* Contoh 4.7 : Menampilkan bilangan genap *
/*          yang nilainya kurang atau sama *
/*          dengan n dan ditampilkan dari *
/*          terbesar sampai nol           *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int n;
    clrscr();
    cout << "Menampilkan bilangan genap yang nilainya "<< endl;
    cout << "kurang atau sama dengan n "<< endl;

    cout << "Masukkan nilai n = " ;
    cin >> n;
    // Jika n ganjil, maka dikurangi 1
    if ( n % 2)
        n --;
    // tampilkan deret bilangan genap dari besar ke kecil
    for ( ; n >= 0; n -= 2 )
        cout << n << ' ';
}
```

Hasil eksekusi :

Menampilkan bilangan genap yang nilainya
kurang atau sama dengan n
Masukkan nilai n = 11 ↵
10 8 6 4 2 0

Pada program diatas terdapat :



sama artinya dengan :

```
for (n -- ; n >= 0 ; n - = 2 )
```

for bersarang (Nested for)

Pada aplikasi tertentu, terkadang kita menggunakan pernyataan **for** yang juga berada dalam pernyataan

for.

Contoh program :

```
/*-----*
/* Contoh 4.7 : Memebentuk segitiga yang berisi *
/*          karakter ' * ' dengan menggunakan *
/*          for didalam for                       *
/*-----*
#include <iostream.h>
#include <conio.h>
```

```

void main()
{
    int tinggi, // Menyatakan tinggi segi tiga
        baris, // Pencacah untuk baris
        kolom; // Pencacah untuk kolom
    clrscr();
    cout << "Tinggi segitiga = ";
    cin >> tinggi;
    cout << endl; //Membuat baris kosong
    for (baris = 1; kolom <= baris; kolom ++ )
    {
        for (klom = 1; kolom <= baris ; klom ++ )
            cout << ' * ' ;
        cout << endl ; // Pindah baris
    }
}

```

Hasil eksekusi :

Tinggi segitiga = 5 ↵

```

*
**
***
****
*****

```

4.6. Pernyataan Continue

Kegunaan dari **continue** dipakai untuk mengarahkan eksekusi ke putaran atau iterasi berikutnya pada pernyataan pengulangan. Efek dari dari perintah ini pada **for**, **while** dan **do-while** :

- Pada **for** :
 Ungkapan ke tiga pada **for** (ungkapan terkanan yang terletak didalam () pada **for**) akan dijalankan dan kemudian ungkapan ke dua diuji lagi.
- Pada **while** dan **do-while** :
 Pengujian terhadap ungkapan pada **while** dilakkan kembali.

Contoh menunjukan efek **continue** pada **for** :

```

/*-----*
/* Contoh 4.8 : Melihat efek continue pada for *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int bil;
    clrscr();
    for ( bil = 0; bil < 10; bil++)
    {
        cout << bil << ' ' ;
        continue;
        cout << "Perhatikan apakah saya muncul" << endl;
    }
}

```

Hasil eksekusi :

0 1 2 3 4 5 6 7 8 9

Pernyataan : cout << "Perhatikan apakah saya muncul " << endl;
 Tidak pernah tersentuh (hal ini akan dilaporkan sewaktu pengompilasian program), sebab **continue** menyebabkab bil++ diproses dan bil < 10 diuji kembali.

Contoh program penggunaan **continue** pada **while** :

```
/*-----*
/* Contoh 4.9 : Penggunaan continue pada      *
/*      while                                  *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int pencacah; // Menyatakan banyaknya nilai
    float nilai;  // Menyatakan nilai yang dimasukkan
    float total;  // Menyatakan total dari nilai

    clrscr();
    cout << "Menghitung nilai rata-rata " << endl;
    cout << "Akhir dengan memasukkan nilai negatif " << endl ;
    pencacah = 0; // Mula-mula diisi dengan nol
    total = 0;
    while ( 1 ) // Ungkapan selalu benar
    {
        cout << "Nilai ke- " << pencacah + 1 << " = ";
        cin >> nilai ; // Baca nilai
        if (nilai > 100)
        {
            cout << " Nilai diatas diabaikan" << endl;
            continue; // Ulangi keawal while
        }

        if ( nilai < 0 )
            break; // keluar dar while
        pencacah ++; // Naikkan pencacah sebesar 1
        total += nilai ; // Tambahkan nilai ke total
    }
    cout << "Jumlah nilai   = " << pencacah << endl;
    cout << " Nilai rata-rata = " << total / pencacah << endl;
}
```

Hasil eksekusi :

```
Menghitung nilai rata-rata
Akhir dengan memasukkan nilai negatif
Nilai ke-1 = 50 ↵
Nilai ke-2 = 60 ↵
Nilai ke-3 = 700 ↵
Nilai diatas diabaikan
Nilai ke-3 = 70 ↵
Nilai ke-4 = -1 ↵
Jumlah nilai   = 3
Nilai rata-rata = 60
```

4.7. Menghentikan Program dengan **exit()**

Suatu eksekusi program dapat dihentikan melalui pemanggilan fungsi **exit()**. Hal ini dapat dilakukan jika dalam sebuah program ada suatu eksekusi kondisi yang tidak dikehendaki.

Bentuk pemakaian **exit()** :

```
exit (nilai_keluar);
```

nilai_keluar dapat diisi dengan dengan 0 sampai dengan 255. Umumnya jika program dapat melakukan tugasnya dengan baik maka nilai keluarannya adalah 0. nilai keluar tidak sama dengan nol untuk menyatakan suatu kesalahan.

Contoh program :

```
/*-----*
/* Contoh 4.10 : Penggunaan exit()      *
/*-----*
#include <iostream.h>
#include <conio.h>
#include <ctype.h>    // Untuk toupper()
#include <stdlib.h>   // Untuk exit()
void main()
{
    char jawab;
    cout << " Y = ya, T = Tidak. Pilihan ( Y/T ) : " ;
    for ( ; ; )
    {
        // Baca tombol dan ubah ke huruf kapital
        jawab = toupper(getch());
        if ((jawab == 'Y') || (jawab == 'T'))
        {
            cout << jawab << endl;
            if ( jawab == 'Y')
                exit(1);
            else
                exit(0);
        }
    }
}
```

Tugas Minggu IV :

Buatlah program untuk suatu kondisi sebagai berikut :

Jika nilai = 81 - 90, nilai = A, keterangan = lulus

Jika nilai = 61 – 80, nilai = B, keterangan = lulus

Jika nilai = 41 – 60, nilai = C, keterangan = lulus

Jika nilai = 21 – 40, nilai = D, keterangan = ulang

Jika nilai = 5 – 20, nilai = E, keterangan = ulang

(simpan dengan nama Tugas4.cpp)

MODUL V

F U N G S I

5.1. Definisi Fungsi

Setiap fungsi yang dipanggil didalam program harus didefinisikan. Letaknya dapat dimana saja.

Contoh definisi kuadrat() :

```
// Prototipe fungsi
long kuadrat (long l);
-----
// Definisi fungsi
long kuadrat(long l)
{
    return(l * l);
}
```

Pernyataan **return** di dalam fungsi digunakan untuk memeberikan nilai balik fungsi. Pada contoh diatas, fungsi kuadrat() memberikan nilai balik berupa nilai kuadrat dari argumen.

Contoh program :

```
/*-----*
/* Contoh 5.1 : Pembuatan fungsi dengan argumen *
/*          bertipe long dan nilai balik berupa *
/*          long                                *
/*-----*
```

```
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
long kuadrat(long l); // prototipe fungsi
void main()
{
    clrscr();
    for ( long bil = 200; bil < 2000; bil+= 200 )
        cout << setw(8) << bil
            << setw(8) << kuadrat(bil) << endl;
}
```

// Definisi fungsi

```
long kuadrat(long l)
{
    return( l * l );
}
```

Hasil eksekusi :

200	40000
400	160000
600	360000
800	640000
1000	1000000
1200	1440000
1400	1960000
1600	2560000
1800	3240000

5.2. Fungsi tanpa Nilai Balik

Adakalanya suatu fungsi tidak perlu memiliki nilai balik. Misalnya fungsi yang hanya dimaksudkan untuk menampilkan suatu keterangan saja.pada funsi seperti ini, tipe nilai balik fungsi yang diperlukan adalah **void.**]

Contoh :

```
void tampilkan_judul ()
{
    cout << "PT. Aria perkasa Manunggal" << endl;
    cout << "Jl. Kertadipura Tenggara 27R" << endl;
    cout << "Kudus" << endl;
}
```

pada contoh funsi diatas, tidak ada pernyataan **return**, mengingat fungsi tidak memiliki nilai balik. Namaun penggunaan pernyataan **return** secara eksplisit juga diperkenenkan. Dalam hal ini digunakan berbentuk :

return
saja.

Jadi contoh diatas dapat ditulis :

```
void tampilkan_judul ()
{
    cout << "PT. Aria perkasa Manunggal" << endl;
    cout << "Jl. Kertadipura Tenggara 27R" << endl;
    cout << "Kudus" << endl;
    return;
}
```

5.3. Lingkup variabel

Lingkup variabel menentukan keberadaan suatu variabel tertentu dalam fungsi. Ada variabel yang hanya dikenal di suatu fungsi dan tidak dikenal pada fungsi lain. Namun ada juga variabel yang dapat diakses oleh semua fungsi.

Variabel Otomatis

Variabel yang didefinisikan di dalam suatu fungsi berlaku sebagai variabel lokal bagi fungsi. Artinya, variabel tersebut hanya dikenal didalam fungsi tempat variabel didefinisikan.

Contoh program :

```
/*-----*
/* Contoh 5.2 : Memperlihatkan efek variabel      *
/*          otomatis (lokal)                      *
/*-----*
#include <iostream.h>
#include <conio.h>
void alpha();    // Prototipe fungsi
void main()
{
    int x = 22;          // Variabel lokal pada main()
    double y = 2.22;
    clrscr();
    cout << " Pada main() : x = " << x
         << " y = " << y << endl;
    alpha();            // Panggil fungsi alpha
    cout << "Pada main() : x = " << x
         << " y = " << y << endl;
}

// Definisi fungsi alpha()

void alpha()
{
    int x = 20;          // Variabel lokal pada alpha()
    double y = 3.14;
    cout << "Pada alpha() : x = " << x
         << " y = " << y << endl;
}
```

Hasil eksekusi :

```
Pada main() : x = 22  y = 2.22
Pada alpha() : x = 22  y = 3.14
Pada main() : x = 22  y = 2.22
```

Tampak bahwa perubahan x dan y pada alpha() tidak mempengaruhi variabel bernama sama pada main(). Karena variabel-variabel tersebut bersifat lokal bagi masing-masing fungsi yang mendefinisikannya. Variabel x dan y pada fungsi alpha() yang disebut sebagai variabel otomatis.

Variabel Eksternal

Variabel eksternal adalah variabel yang didefinisikan diluar fungsi manapun. Variabel ini dikenal juga sebagai variabel global, sebab variabel ini dikenal disemua fungsi.

Contoh program :

```
/*-----*
/* Contoh 5.3 : Pemakaian Variabel eksternal    *
/*-----*
#include <iostream.h>
#include <conio.h>
int oblade = 550;      // Variabel eksternal
void tambah();         // Prototipe fungsi
void main()
{
    clrscr();
    cout << oblade << endl;
    tambah();
}
```

```

        cout << oblade << endl;
        tambah();
        cout << oblade << endl;
    }
    // Definisi fungsi
    void tambah ()
    {
        oblada ++ ;    // Variabel eksternal dinaikkan
    }

```

Hasil eksekusi :

```

551
552
553

```

Guna memperjelas suatu variabel didalam suatu fungsi yang menggunakannya dapat mendeklarasikan (bukan mendefinisikan, karena tidak ada pengalokasian memori) dengan menambahkan kata **extern** didepan tipa data.

Contoh program :

```

/*-----*
/* Contoh 5.4 : Pemakaian Variabel eksternal      *
/*          dengan tambahan kata extern            *
/*-----*
#include <iostream.h>
#include <conio.h>
int oblade = 550;    // Variabel eksternal
void tambah();       // Prototipe fungsi
void main()
{
    clrscr();
    cout << oblade << endl;
    tambah();
    cout << oblade << endl;
    tambah();
    cout << oblade << endl;
}

// Definisi fungsi
void tambah ()
{
    extern oblade;
    oblada ++ ;    // Variabel eksternal dinaikkan
}

```

Variabel Statis

Variabel eksternal maupun otomatis dapat berkedudukan sebagai variabel statis. Suatu variabel statis mempunyai sifat :

- Jika variabel lokal berdiri sebagai variabel statis, maka :
 - Variabel tetap hanya dapat diakses pada fungsi yang mendefinisikannya
 - Variabel tidak hilang saat eksekusi fungsi berakhir
 - Inisialisasi oleh pemrogram akan dilakukan sekali saja selama program dijalankan.
- Jika variabel eksternal dijadikan sebagai variabel statis, variabel ini dapat diakses oleh semua file yang didefinisikan pada file yang sama dengan variabel eksternal tersebut.

Contoh program :

```

/*-----*
/* Contoh 5.5 : Menunjukkan efek variabel statis  *
/*-----*
#include<iostream.h>
#include <conio.h>
void saya_ingat(); // Prototipe fungsi

```

```

void main()
{
    int mana = 50;
    clrscr();
    saya_ingat();
    saya_ingat();
    saya_ingat();
    cout << " main()      : mana = " << mana << endl;
}
// Pada fungsi berikut
// mana didefinisikan sebagai variabel statis
void saya_ingat()
{
    static int mana = 77; // variabel statis
    mana ++;             // Naikkan sebesar 1
    cout << " Saya_ingat () : mana = " << mana << endl;
}

```

Hasil eksekusi :

Saya_ingat () : mana = 78
 Saya_ingat () : mana = 79
 Saya_ingat () : mana = 80
 main() : mana = 50

Berdasarkan hasil diatas, terlihat bahwa variabel statis mana pada fungsi saya_ingat () hanya diinisialisasi (bernilai 77) sekali saja. Kemudian setiap kali saya_ingat () dipanggil nilai variabel tersebut dinaikkan sebesar 1.

5.4. Operator Resolusi Lingkup

Pada C++ terdapat operator dua buah tanda titik-dua (::). Operator ini disebut operator resolusi lingkup (*scope resolution*). Kegunaanya untuk mengakses variabel yang didefinisikan diluar suatu fungsi.

Contoh program :

```

/*-----*
/* Contoh 5.6 : Pemakaian operator resolusi      *
/*          lingkup ( :: )                        *
/*-----*
#include < iostream.h>
#include <conio.h>
int x = 50;          // Variabel eksternal

```

```

void main()
{
    double x;          // Definisi variabel lokal
    clrscr();
    x = 5.678901234; // Variabel lokal yang diberi nilai
    cout << x << " " << ::x << endl;
    ::x = 77;          // Variabel eksternal yang diberi nilai
    cout << x << " " << ::x << endl;
}

```

Hasil eksekusi :

5.6789 50
 5.6789 77

Program diatas mengungkapkan bahwa sekalipun didalam suatu fungsi terdapat variabel yang namanya sama dengan nama variabel eksternal, kedua variabel tersebut tetap bisa diakses.

5.5. Nilai Bawaan Untuk Nilai Fungsi

Salah satu keistimewaan C++ adalah adanya kemampuan untuk menyetel nilai bawaan (*default*) argumen fungsi. Argumen-argumen yang mempunyai nilai bawaan nantinya dapat tidak disertakan didalam pemanggilan fungsi dan dengan sendirinya C++ akan menggunakan nilai bawaan dari argumen yang tidak disertakan.

Contoh program :

```
/*-----*
/* Contoh 5.7 : Menggambarkan nilai bawaan dalam *
/*          argumen fungsi                        *
/*-----*
#include <iostream.h>
#include <conio.h>
void tulis_cplus(int jum); // Prototipe fungsi
void main()
{
    clrscr();
    tulis_cplus(1); // Untuk menuliskan sebuah tulisan C++
}

void tulis_cplus(int jum);
{
    for (int i = 0; i < jum; i ++ )
        cout << " C++ " << endl;
    cout << " Seleseai " << endl;
}
```

Hasil eksekusi :



Contoh program :

```
/*-----*
/* Contoh 5.8 : Menggambarkan nilai bawaan *
/*          Dalam argumen fungsi            *
/*-----*
#include <iostream.h>
#include <conio.h>
void tulis_cplus(int jum = 1); // Prototipe fungsi
                                // Dan menyetel nilai bawaan fungsi
void main()
{
    clrscr();
    tulis_cplus(); // Argumen tidak perlu disebutkan
}

void tulis_cplus(int jum);
{
    for (int i = 0; i < jum; i ++ )
        cout << " C++ " << endl;
    cout << " Seleseai " << endl;
}
```

Pada contoh program 5.7 dan 5.8 mempunyai kesamaan hanya saja pada contoh program 5.8 dalam prototipe fungsi nilai bawaannya dikut sertakan sehingga pada saat argumen pemanggilan fungsi tidak perlu di tuliskan lagi.

5.7. Referensi

Referensi digunakan untuk memberikan nama alias dari variabel. Bentuk pendeklarasiannya :

Int &ref = nama_variable ;

Tanda & mengawali nama referensi.

Setelah pendeklarasian seperti diatas, *ref* menjadi nama alias dari *nama_variabel*. Penggubahan nilai terhadap *nama_variabel* dapat dilakukan melalui *nama_variabel* itu sendiri atau melalui referensi *ref*, sebagaimana dapat dilihat pada contoh dibawah ini.

Contoh program :

```
/*-----*
/* Contoh 5.9 : contoh referensi      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int i ;
    int &r = 1;  // Deklarasi referensi
    clrscr();
    i =10;
    cout << "i = " << i << endl;
    cout << " r = " << r << endl;
    r = 55;
    cout << "i = " << i << endl;
    cout << " r = " << r << endl;
}
```

Hasil eksekusi :

```
i = 10
r = 10
i = 55
r = 55
```

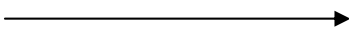
Tampak bahwa pengubahan nilai terhadap i maupun r memberikan efek sama.

Operator juga bekerja pada suatu variabel maupun referensinya dengan efek yang sama. Sebagai contoh :

```
/*-----*
/* Contoh 5.10 : Operasi penaikan isi variabel*
/*          melalui referensi                *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int i = 55;
    int &r = i;  // Referensi
    clrscr();
    cout << "i = " << i << " r = " << r << endl;
    i ++ ;

    cout << " i = " << i << " r = " << r << endl;
    r ++ ;
    cout << "i = " << i << " r = " << r << endl;
}
```

Hasil eksekusi :



```
i = 55  r = 55
i = 56  r = 56
i = 57  r = 57
```

Operator ++ pada i maupun r akan mengubah nilai keduanya, karena i dan r menyiratkan memori yang sama.

5.8. Inline Function

Inline function dengan cukup menyisipkan kata-kata **inline** didepan tipe nilai balik fungsi dalam pendefinisian fungsi. Contoh :

```
inline jumlah (int x, int y)
{
    return(x + y);
}
```

inline function disarankan dipakai pada fungsi yang sering dipanggil dan ukurannya kecil (terdiri satu atau dua pernyataan), terutama jika dilibatkan pada pernyataan pengulangan proses (**while**, **for** dan **do-while**). Misalnya pada bentuk seperti berikut :

```
for (int i = 1; i < 100; i++)
    cout << i << “.” << jumlah (i, 2 * i) << endl;
jika fungsi jumlah () tidak ditulis sebagai inline function, proses tersebut akan menjadi relatif lambat.
```

Contoh program :

```
/*-----*
/* Contoh 5.11 : Pembuatan fungsi inline      *
/*-----*
#include <iostream.h>
#include <conio.h>
// Definisi fungsi sebagai inline
inline int jumlah(int x, int y)
{
    return(x + y );
}
void main()
{
    clrscr();
    for (int i = 1; i < 100; i ++ )
        cout << i << “.” << jumlah(i, 2 * i) << endl;
}
```

Hasil eksekusi :

```
1. 3
2. 6
3. 9
4. 12
5. 15
6. 18
...
```

5.9. Function Overloading

Function Overloading atau *Overloading* terhadap fungsi memungkinkan sebuah fungsi dapat menerima bermacam-macam tipe dan memberikan nilai balik yang bervariasi pula.

Contoh program :

```
/*-----*
/* Contoh 5.12 : Contoh overloading          *
/*      terhadap fungsi                      *
/*-----*

#include <iostream.h>
#include <conio.h>
// Prototipe fungsi
int kuadrat (int i);
long kuadrat(long l);
double kuadrat(double d);
void main()
{
    cout << kuadrat(2) << endl;
    cout << kuadrat(66666) << endl;
    cout << kuadrat(1.2) << endl;
}
```

```
// Definisi fungsi
int kuadrat (int i)
{
    return(i * i);
}
long kuadrat (long l)
{
    return(l * l);
}
double kuadrat (double d)
{
    return(d * d);
}
```

Hasil rekursi :

4

149388260

1.44

Program 5.12 menunjukkan pemanggilan fungsi kuadrat() dengan berbagai bentuk tipe data.

5.10. Rekursi

Fungsi dalam C++ dapat dipakai secara rekursi, artinya suatu fungsi dapat memanggil fungsi yang merupakan dirinya sendiri. Penerapan rekursi diantaranya untuk menghitung nilai :

$$X^n$$

Dengan n merupakan bilangan bulat positif. Solusi dari persoalan ini berupa :

Jika $n = 1$ maka $X^n = X$

Selain itu : $X^n = X * X^{n-1}$

Contoh program :

```
/*-----*
/* Contoh 5.13 : Opreasi pangkat secara *
/*          rekursi                      *
/*-----*
# include <iostream.h>
#include <conio.h>
long int pangkat ( int x, int n);
void main()
{
    int x, y;
    clrscr();
    cout << " Menghitung x ^ y "<< endl;
    cout << " x = ";
    cin >> x ;
    cout << " y = ";
    cin >> y ;
    cout << x << " ^ " << y << endl;
        << pangkat(x, y) << endl;
}

long int pangkat(int x, int n)
{
    if (n == 1 )
        return(x);
    else
        return(x * pangkat(x, n - 1));
}
```

Hasil elsekusi :

Menghitung x ^ y

x = 2 ↵

y = 3 ↵

2 ^ 3 = 8

Tugas Minggu V :

Buat program untuk memasukkan data disertai keterangan data ke-.

Contoh :

Banyak data = 3

Masukkan Nilai ke-1 : 28 ↵

Masukkan Nilai ke-2 : 11 ↵

Masukkan Nilai ke-3 : 1982 ↵

Data ke-1 = 28

Data ke-2 = 11

Data ke-3 = 1982

(simpan dengan nama tugas5.cpp)

MODUL VI

MENGENAL ARRAY

6.1. Array Dimensi Satu

Gambaran sebuah array ditunjukkan pada Contoh Program 6.1. Program ini meminta pemakai untuk memasukkan 5 buah data temperatur dari keyboard. Kelima data tersebut disimpan pada array bernam suhu. Selanjutnya data yang ada pada array tersebut ditampilkan ke layar.

Contoh program :

```
/*-----*
/* Contoh 6.1 : program menggunakan   *
/*          array                      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    float suhu[5]; // Array dengan elemen 5 bertipe float
    clrscr();
    // Membaca data dari keyboard dan meletakkan array
    cout << "Masukkan 5 buah data suhu" << endl;
    for (int i = 0; i < 5; i++)
    {
        cout << i + 1 << " . ";
        cin >> suhu[i];
    }
    // Menampilkan isi array ke layar
    cout << "Data suhu yang anda masukkan : " << endl;
    for ( i = 0; i < 5; i++)
        cout << suhu[i] << endl;
}
```

Hasil eksekusi :

```
Masukkan 5 buah data suhu
1 : 27.5 ↵
2 : 28 ↵
3 : 27.5 ↵
4 : 30.5 ↵
5 : 27 ↵
Data suhu yang anda masukkan
27.5
28
27.5
30.5
27
```

Tampak diatas terdapat pernyataan :

- **float suhu[5];** menyatakan array **suhu** dapat menyimpan 5 (lima) buah data bertipe **float**.
- **suhu[i]** menyatakan elemen suhu dengan *subscript* sama dengan **i**.
- **cin >> suhu[i];** : membaca data dari keyboard dan meletakkan ke elemen nomor **i** pada array **suhu**.
- **cout << suhu[i];** : akan menampilkan elemen bernomor **i** pada array **suhu**.

6.2. Array Berdimensi Dua

Array ini dapat digunakan untuk berbagai keperluan. Sebagai gambaran, data kelulusan dari jurusan Teknik Tnformatika, Manajemen Informatika dan Teknik Komputer pada sekolah tinggi Komputer dari tahun 1992 hingga 1995 dapat dinyatakan dengan array berdimensi dua.

Sebelum membahas cara pendefinisian array berdimensi dua, perhatikan tabel berikut :

Jurusan	1992	1993	1994	1995
1. Teknik Informatika	35	45	80	120
2. Manajemen Informatika	100	110	70	101
3. Teknik Komputer	10	15	20	17

Bentuk seperti tabel diatas dapat dituangkan kedalam array berdimensi dua. Pendefinisiannya :

```
int data_lulus[3][4];
```

pada pendefiniasian diatas :

- 3 menyatakan jumlah baris (mewakili jurusan)
- 4 menyatakan jumlah kolom (mewakili tahun kelulusan)

array berdimensi dua dapat diakses dengan bentuk :

`nama_array[subscript_baris, subscript_kolom]`

Contoh program :

```
/*-----*
/* Contoh 6.2 : Pemakaian array berdimensi  *
/*          dua                               *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    int data_lulus[3][4]; // Array berdimensi dua
    int tahun, jurusan;
    clrscr();
    // Memberikan data ke elemen array data_lulus
    data_lulus[0][0] = 35; // data TI – 1992
    data_lulus[0][1] = 45; // data TI - 1993
    data_lulus[0][2] = 90; // data TI - 1994
    data_lulus[0][3] = 120; // data TI - 1995

    data_lulus[1][0] = 100; // data MI – 1992
    data_lulus[1][1] = 110; // data MI - 1993
    data_lulus[1][2] = 70; // data MI – 1994
    data_lulus[1][3] = 101; // data MI - 1995
    data_lulus[2][0] = 10; // data TK – 1992
    data_lulus[2][1] = 15; // data TK- 1993
    data_lulus[2][2] = 20; // data TK - 1994
    data_lulus[2][3] = 17; // data TK – 1995
    // Proses untuk memperoleh informasi kelulusan
    while (1)
    {
        cout << "Jurusan ( 0 = TI, 1 = MI, 2 = TK ) : “ ;
        cin >> jurusan;
        if ((jurusan == 0 ) || (jurusan == 1) || (jurusan == 3))
            break; // Keluar dari while
    }
}
```

```

while (1)
{
    cout << "Tahun ( 1992 – 1995 ) : “ ;
    cin >> tahun;
    if (( tahun >= 1992 ) && ( tahun <= 1995 )
    {
        tahun - = 1992; // Konversi ke 0, 1, 2 atau 3
        break; // Keluar dari while
    }
}
cout << "Jumlah yang lulus = “
<< data_lulus[jurusan][tahun] << endl;
}

```

Hasil eksekusi :

```

Jurusan ( 0 = TI, 1 = MI, 2 = TK ) : 1 ↵
Tahun (1992 – 1195) : 1992 ↵
Jumlah yang lulus = 100

```

Mula-mula program mengisi data ke array data_lulus. Kemudian program meminta data jurusan dan tahun dari keyboard. Kedua data masukan dipastikan tidak akan berada diluar jangkauan kedua subscript-nya.

6.3. Array Berdimensi Tiga

Bentuk umum pendefinisian array berdimensi tiga :

```

tipe nama_array[subscript1][subscript2][subscript3]

```

Sebagai contoh :

```

int huruf[2][8][8] ;

```

merupakan pendefinisian array data_huruf sebagai array berdimensi tiga.

Tugas Minggu VI :

Buatlah program untuk menghitung bilangan terbesar pada suatu array dan terletak pada data beberapa.

Contoh :

Banyak data = 6

Data 1 = 28

Data 2 = 11

Data 3 = 1982

Data 4 = 8

Data 5 = 12

Data = 1978

Data terbesar = 1982

Pada Data ke – 3 (simpan dengan nama tugas6.cpp)

MODUL VII
DASAR STRING

7.1. Konstatnta String

Suatu konstanta string ditulis dengan awalan dan akhiran tanda petik ganda (“). Misalnya :

“ C++ “

Konstanta string disimpan dalam memori secara berurutan, setiap karakter menempati memori sebesar 1 byte. Setelah karakter yang terakhir terdapat karakter NULL (karakter dengan nilai ASCII sama dengan nol atau disimbolkan dengan ‘\0’, yaitu tanda \ diikuti nol).

Bila suatu string hanya berisi karakter NULL, string disebut sebagai string kosong.

7.2. Variabel String

Variabel string adalah variabel yang dipakai untuk menyimpan string. Misalnya :

char teks[10];

merupakan pernyataan untuk mendefinisikan variabel string dengan panjang maksimal 15 karakter (sudah termasuk karakter NULL).

7.3. Memasukan Data String Dari Keyboard

Setelah suatu variabel string didefinisikan, bisa mengisi data ke variabel tersebut. Pemasukkan data dapat ditangani oleh **cin**.

Contoh program :

```
/*-----*
/* Contoh 7.1 : Pendefinisian variabel string dan *
/*           pengisian variabel melalui cin      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    char teks[13];           // String dengan panjang maksimal
                           // 13 karakter

    clrscr();
    cout << "Masukkan sebuah kata " << endl;
    cin >> teks;
    cout << "Yang Anda Masukkan : " << teks << endl;
}
```

Hasil eksekusi :

Masukkan sebuah kata
Halo ↵
Yang Anda Masukkan : Halo

Pada contoh diatas bila Anda memasukkan sebuah kata seperti :
“Halo, Sobat”. Maka kata setelah spasi tidak akan terbaca, untuk mengatasi hal ini anda dapat menggunakan fungsi anggota **get()** pada obyek **cin** (**cin.get()**). Seperti contoh program berikut :

```
/*-----*
/* Contoh 7.2 : Pendefinisian variabel string dan *
/*           pengisian variabel melalui cin.get() *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    char teks[13];           // String dengan panjang maksimal
                           // 13 karakter

    clrscr();
    cout << "Masukkan sebuah kata " << endl;
    cin.get >> (teks, 13);
    cout << "Yang Anda Masukkan : " << teks << endl;
}
```

Hasil eksekusi :

Masukkan sebuah kata
Halo, Sobat ↵
Yang Anda Masukkan : Halo Sobat

7.3. Fungsi dan Makro Berbasis Karakter

Fungsi Makro berguna untuk menganalisis karakter-karakter yang terdapat pada suatu string ataupun untuk melakukan konversi (misalnya huruf kecil menjadi huruf kapital).

Makro keluarga is...

- Sejumlah makro dengan awalan is, adalah sebagai berikut :
- isalnum() : karakter alphanumeric (digit, huruf kecil atau huruf kapital)
 - isalpha() : karakter alphabetis (huruf kecil atau huruf kapital)
 - isascii() : karakter ASCII standar
 - iscntrl() : karakter kontrol
 - isdigit() : karakter digit (0 sampai dengan 9)
 - isgraph() : karakter yang dapat ditampilkan tetapi spasi tidak
 - islower() : huruf kecil (a sampai z)
 - isprint() : karakter yang dapat ditampilkan termasuk spasi
 - ispunct() : tanda puntuasi (seperti ! ataupun ?)
 - isupper() : huruf kapital (A sampai Z)
 - isxdigit() : digit heksadesimal (0 samapi 9, Asamapai F, a sampai f)

Contoh program :

```
/*-----*
/* Contoh 7.3 : Contoh pemakaian islower(), isupper() *
/*          isdigit(), ispunct() *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    char st[128];
    int jum_kapital = 0,
        jum_kecil = 0,
        jum_puntuasi = 0,
        jum_digit = 0;
    clrscr();
    cout << "Masukkan string apa saja : " << endl;
    cin.getline(st, sizeof(st)) ;
    for (int i = 0; st[i]; i ++ )
    {
        if (islower(st[i]))
            jum_kecil++;

        if (isupper(st[i]))
            jum_kapital++;
        if (isdigit(st[i]))
            jum_digit++;
        if (ispunct(st[i]))
            jum_puntuasi++;
    }
    cout << endl ;           // Pindah baris
    cout << "Jumlah Huruf Kapital : " << jum_kapital << endl;
```

```
cout << Jumlah Huruf Kecil : “ << jum_kecil << endl;
cout << Jumlah Digit : “ << jum_digit << endl;
cout << Jumlah punctuation : “ << jum_punctuation << endl;
}
```

Hasil eksekusi :

```
Masukkan string apa saja :
123 halo 123. Test ↵

Jumlah Huruf Kapital : 1
Jumlah Huruf Kecil : 6
Jumlah Digit : 6
Jumlah punctuation : 2
```

7.4. Konversi String ke Angka atau Sebaliknya

Untuk melakukan konversi string ke bilangan, Anda dapat menggunakan sejumlah fungsi bawaan. Fungsi yang tersedia adalah sebagai berikut :

Fungsi	Prototipe	Keterangan
atoi()	stdlib.h	Mengkonversi string argumen menjadi nilai bertipe int
atof()	stdlib.h	Mengkonversi string argumen menjadi nilai bertipe float
atol()	stdlib.h	Mengkonversi string argumen menjadi nilai bertipe long int
_atold()	stdlib.h	Mengkonversi string argumen menjadi nilai bertipe long double (hanya terdapat dalam borland C++)

Adapun fungsi untuk bilangan ke string adalah sebagai berikut :

Fungsi	Prototipe	Keterangan
Itoa()	stdlib.h	Untuk mengkonversi suatu bilangan bertipe int menjadi string
Ltoa()	stdlib.h	Untuk mengkonversi suatu bilangan bertipe long int menjadi string
ultoa()	stdlib.h	Untuk mengkonversi suatu bilangan bertipe unsigned long int menjadi string

7.5. String sebagai Parameter Fungsi

String sebagai parameter fungsi pada dasarnya sama dengan array sebagai parameter fungsi. Contoh program :

```
/*-----*
/* Contoh 7.4 : pembuatan fungsi yang melibatkan *
/*          argument bertipe string *
/*-----*
#include <iostream.h>
#include <conio.h>
#include <string.h>
void RataKiri(char hasil[], char st[], int n);
void main()
{
    char teks[] = "Bahasa C++";
    char hasil[128];
    clrscr();
    int panjang = strlen(teks);
```



```

for (int i = 0; i < panjang; i++)
{
    RataKiri(hasil, teks, i+1);
    cout << hasil << endl;
}

```

Hasil eksekusi :

```

B
Ba
Bah
Baha
Bahas
Bahasa
Bahasa
Bahasa C
Bahasa C+
Bahasa C++

```

Dalam program diatas argumen fungsi ditulis dengan bentuk seperti *hasil[]* pada pendefinisian fungsi. Tetapi hal ini tidaklah umum sebagai pengantinya argumen string dinyatakan dengan pointer yang menunjuk ke tipe **char**.

Tugas Minggu VII :

Buatla program dengan menggunakan Makro keluarga is untuk mengubah huruf kecil menjadi huruf kapital dan sebaliknya.

(simpan dengan nama tugas7.cpp)

MODUL VIII
MENGENAL POINTER

8.1. Mendefinisikan Variabel Pointer

Suatu variabel pointer didefinisikan dengan bentuk sebagai berikut :

*tirp_data *nama_variabel*

- *tipe_data* dapat berupa sembarang tipe seperti halnya pada pendefinisian variabel bukan pointer.
- *nama_variabel* adalah nama variabel pointer

Supaya suatu variabel menunjuk ke variabel lain, mula-mula harus diisi dengan alamat dari variabel yang hendak ditunjuk.

Contoh program :

```
/*-----*
/* Contoh 8.1 : Pendefinisian variabel pointer dan *
/*           pengisian alamat ke variabel tersebut *
/*-----*
#include <iostreamh>
#include <conio.h>
void main()
{
    int vint = 55; // Variabel bukan pointer
    int *pint;     // Variabel pointer
    clrscr();
    pint = &vint;  // Pointer menunjuk ke vint
    cout << "Alamat vint = " << &vint << endl;
    cout << "pint = " << pint << endl;
}
```

Hasil eksekusi :

Alamat vint = 0xffff4
pint = 0xffff4

Pada program diatas :

cout << "pint = " << pint << endl;
ternyata menampilkan isi pointer itu sendiri, bukan isi dari variabel vint.

8.2. Mengakses Nilai yang ditunjuk Pointer

Berdasarkan contoh program 8.1. didepan, nilai dari vint dapat diakses melalui pint setelah pernyataan :

pint = &vint

dijalankan. Caranya dengan melibatkan operator "tak langsung". Operator ini berupa simbol "*" dan diletakkan di depan nama variabel pointer. Contoh

*pint

Contoh program :

```
/*-----*
/* Contoh 8.2 : Pengaksesan nilai melalui suatu *
/*           pointer *
/*-----*
#include <iostream.h>
#include <conio.h>

void main()
{
    int vint = 55; // Variabel bukan pointer
    int *pint;     // Variabel pointer
    clrscr();
    pint = &vint;  // Pointer menunjuk ke vint
    cout << "Nilai yang ditunjuk oleh pint = "
```

}

Nilai yang ditunjuk oleh pint = 55

```

    << tgl_lahir << endl;
}

```

Hasil eksekusi :

Nilai yang ditujuk oleh ptgl : 24
 Nilai dari tgl_lahir[0] : 24

Setelah penugasan `ptgl = tgl_lahir` `ptgl` akan menunjuk ke elemen pertama (`tgl_lahir[0]`) dari array `tgl_lahir`.

Tugas Minggu VIII :

Buatlah program menentukan bulan.

Contoh :

Masukkan Kelahiran Anda : 28 / 11 /1982

Anda Lahir pada : 28 November 1982

(simpan degan nama tugas8.cpp)

BAB IX

STRUKTUR, ENUM, UNION, BIT-FIELD DAN TYPEDEF

9.1. Struktur

Struktur bermanfaat untuk mengelompokkan sejumlah data dengan tipe yang berlainan. Apabila suatu struktur telah dideklarasikan, struktur ini dapat dgunakan untuk mendefinisikan suatu variabel.

Suatu struktur juga dapat mengandung struktur yang lain dan anggota struktur dapat diakses menggunakan bentuk :

`variable_struktur.nama_anggota`

Contoh program lengkap yang melibatkan pendeklarasian dan pendefinisian variabel struktur dan juga pengaksesan terhadap anggota variabel struktur dapat dilihat dibawah ini :

```

/*-----*
/* Contoh 9.1 : Pendeklarasian, pendefinisian dan *
/*           pengaksesan struktur                *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    struct data_tanggal          // Pendeklarasian
    {
        int tahun;
        int bulan;
        int tanggal;
    };
    data_tanggal tanggal_lahir  // Pendefinisian struktur
    // Pengaksesan anggota struktur
    tanggal_lahir.tangal = 28;
    tanggal_lahir.bulan = 11;
    tanggal_lahir.tahun = 1982;
    cout << tanggal_lahir.tanggal << ' / '
         << tanggal_lahir.bulan << ' / '
         << tanggal_lahir.tahun << endl;
}

```

Hasil eksekusi :

28/11/1982

Pada program diatas tanda titik diantara nama variabel dan nama anggota menyatakan penugasan untuk memberikan nilai 28 ke anggota *tanggal* pada variabel struktur *tanggal_lahir*.

Pemberian nilai terhadap suatu struktur dapat dilakukan dengan bentuk :

var1 = var2;

sepanjang kedua variabel adalah variabel struktur bertipe sama.

Contoh program :

```
/*-----*
/* Contoh 9.2 : Penugasan struktur      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    struct data_tanggal          // Pendeklarasian
    {
        int tahun;
        int bulan;
        int tanggal;
    };

    data_tanggal tgl1, tgl2 ;    // Pendefinisian struktur
    // Penugasan per anggota
    tgl1.tanggal = 28;
    tgl1.bulan = 11;
    tgl1.tahun = 1982;
    // Penugasan antaranggota struktur
    tgl1 = tgl2;
    cout << tgl2.tanggal << ' / '
         << tgl2.bulan << ' / '
         << tgl2.tahun << endl;
}
```

Hasil eksekusi :

28/11/1982

9.2. Union

Union menyerupai struktur (termasuk dalam hal pengaksesannya), namun mempunyai perbedaan nyata.

Union biasa dipakai untuk menyatakan suatu memori dengan nama lebih dari satu.

Contoh program :

```
/*-----*
/* Contoh 9.3 : Pendefinisian union      *
/*-----*
#include <iostream.h>
#include <conio.h>
union bil_bulat
{
    unsigned int di;
    unsigned char dc[2];
};

void main()
{
    clrscr();
    bil_bulat bil_x;    // Pendefinisian union
```

```

bil_x.di = 0x2345;
cout << setiosflags(ios::showbase);
cout << hex << "di : " << bil_x.di << endl;
cout << hex << "dc[0] : " << int(bil_x.dc[0]) << endl;
cout << "dc[1] : " << int(bil_x.dc[1]) << endl;
}

```

Hasil eksekusi :

Di : 0x2345 dc[0] : 0x45 dc[1] : 0x23

Tampak bahwa dengan mengisikan nilai *bil_x.di*, data dapat diakses melalui *bil_x.dc*. dalam hal ini, *bil_x.dc[0]* menyimpan byte rendah dari *bil_x.di* dan *bil_x.dc[1]* berkaitan dengan byte tinggi dari *bil_x.di* (mengingat *bil_x.di* berukuran dua byte).

Seperti halnya struktur, variabel union juga dapat diinisialisasi saat didefinisikan.

Contoh program :

```

/*-----*
/* Contoh 9.4 : Inisialisasi Union      *
/*-----*

#include <iostream.h>
#include <conio.h>
union bil_bulat
{
    unsigned int di;
    unsigned char dc[2];
};

void main()
{
    clrscr();
    bil_bulat bil_x = 0x2345;    // Inisialisasi

    cout << setiosflags(ios::showbase);
    cout << hex << "di : " << bil_x.di << endl;
    cout << hex << "dc[0] : " << int(bil_x.dc[0]) << endl;
    cout << "dc[1] : " << int(bil_x.dc[1]) << endl;
}

```

9.3. Struktur Bit-field

Satu bit atau beberapa bit dalam sebuah data berukuran suatu byte atau dua byte dapat diakses dengan mudah melalui *bit-field*. Dengan cara ini suatu bit atau beberapa bit dapat diakses tanpa melibatkan operator manipulasi bit (seperti & dan ||). Selain itu satu atau dua byte meri dapat dipakai untuk menyimpan sejumlah informasi.

Conto program :

```

/*-----*
/* Contoh 9.5 : bit-field untuk mengakses bit-bit *
/*          dalam sebuah byte data                *
/*-----*

#include <iostream.h>
#include <conio.h>
// Pendeklarasian bit-field
struct info_bit
{
    unsigned bit0 : 1;
    unsigned bit1 : 1;
    unsigned bit2 : 1;
    unsigned bit3 : 1;
    unsigned bit4 : 1;
}

```

```

        unsigned bit5 : 1;
        unsigned bit6 : 1;
        unsigned bit7 : 1;
    };
    void main()
    {
        clrscr();
        union ubyte    // Pendeklarasian union
        {
            unsigned char byte;
            info_bit bit;
        };
        ubyte ascii;    // Pendeklarian variabel union
        int nilai;
        cout << "Masukkan ascii antara 0 s/d 255 : ";
        cout << ascii.bit.bit7 << ascii.bit.bit6
        cout << ascii.bit.bit5 << ascii.bit.bit4
        cout << ascii.bit.bit3 << ascii.bit.bit2
            << ascii.bit.bit1 << ascii.bit.bit0 << endl;
    }

```

Hasil eksekusi :

Masukkan ascii antara 0 s/d 255 : 128 ↵ 76543210 ← Posisi bit 10000000
Masukkan ascii antara 0 s/d 255 : 16 ↵ 76543210 ← Posisi bit 00010000

Perlu diketahui, suatu variabel yang didefinisikan sebagai *bit-field* tidak bisa diisi secara langsung dengan suatu nilai. Oleh karena itu biasa dibentuk didalam union.

9.4. Enum

Tipe enum biasa dipakai kalau kemungkinan nilai dari suatu data telah diketahui, dan jumlah kemungkinannya tidak banyak.

Contoh program :

```

/*-----*
/* Contoh 9.6 : tipe enum      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    enum nama_hari { Senin, Selasa, Rabu, Kamis, Jumat,
                    Sabtu, Minggu };
    nama_hari hari1, hari2;    // Pendefinisian enum
    // Pemberian nilai enum
    hari1 = Senin;
    hari2 = Jumat;
    int selisih = hari2 – hari1;
    cout << "Selisih Hari : " << selisih << endl;
}

```

Hasil eksekusi :

Selisih Hari : 4

Pada contoh diatas terdapat pendeklarasian tipe **enum** berupa *nama_hari*. Anggota tipe **enum** *nama_hari* berupa Senin, Selasa, Rabu dan seterusnya.

nma_hari hari1, hari2 merupakan pernyataan untuk mendefinisikan variabel bernama *hari1* dan *hari2* yang bertipe **enum** *nama_hari*.

9.5. typedef

typedef biasa dipakai untuk memberikan nama alias terhadap suatu tipe data. Sebagai contoh anda dapat memberikan nama alias dari unsigned char berupa BYTE.

Contoh program :

```
/*-----*
/* Contoh 9.7 : typedef      *
/*-----*
#include <iostream.h>
#include <conio.h>
void main()
{
    clrscr();
    typedef unsigned char BYTE;
    BYTE kode;          // Sebagai pemendekan dari : unsigned char code;
    kode = 65;
    cout << kode; // Karakter ASCII 65
}
```

Hasil eksekusi :

A

Tugas Minggu IX :

Buat program untuk menentukan selisih hari kelahiran anda dengan teman anda. (digenapkan dalam bentuk tahun)

Contoh :

Anda lahir : 8/12/1978

Teman Lahir : 28/11/1982

Selisih Lahir (digenapkan) = 4 tahun

(simpan dengan nama tugas9.cpp)