Bab I Pendahuluan

1.1 Konsep Pemrograman

Sebuah komputer tidak dapat mengerjakan apapun tanpa adanya perintah dari manusia. Perintah – perintah yang terstruktur dan sistematis untuk membuat agar komputer dapat bekerja sesuai dengan apa yang diinginkan disebut **program**. Komputer dapat diprogram untuk berbagai hal misalnya untuk melakukan perhitungan suatu ekspresi matematika, memainkan lagu, mengurutkan sekumpulan data, melakukan permainan (games), menggambar dan sebagainya. Program-program semacam itu dibuat oleh manusia, syarat utama dalam membuat program komputer adalah perintah-perintah yang diberikan dalam program tersebut harus dimengerti oleh komputer.

Komputer hanya dapat mengerti sebuah bahasa yang disebut **bahasa mesin.** Bahasa yang sangat berbeda dengan bahasa manusia dan terlebih lagi akan amat menyulitkan untuk membuat sebuah program dalam bahasa mesin ini. Manusia menginginkan sebuah bahasa komputer yang sederhana yang dapat dimengerti dan mudah dipelajari oleh manusia sekaligus dapat dimengerti oleh komputer. Bahasa komputer tersebut disebut **bahasa pemrograman** (*programming language*). Yang perlu diingat, **konsep bahasa pemrograman** adalah merubah/menerjemahkan perintah-perintah (program) yang diberikan oleh manusia ke dalam bahasa mesin yang dapat dimengerti oleh komputer. Jadi bahasa pemrograman merupakan sarana interaksi antara manusia dan komputer. Penerjemah bahasa pemrograman dibedakan menjadi 3 macam yaitu :

1. *Asembler* adalah program yang digunakan utnuk menerjemahkan kode sumber dalam bahasa rakitan (assembly) ke dalam bahasa mesin
2. *Kompiler* adalah program penerjemah yang mengonversi **kode sumber** selain dalam bahasa rakitan menjadi **kode objek.** Hasil berupa kode objek inilah yang bisa dijalankan oleh komputer. Proses untuk melakukan penerjemahan ini disebut **kompilasi**. Bahasa yang menggunakan proses kompilasi adalah : bahasa COBOL, Pascal, bahasa C
3. Intepreter adalah program yang menerjemahkan satu persatu instruksi dalam kode sumber dan kemudian segera menjalankan instruksi yang telah diterjemahkan tersebut. Bahasa seperti BASIC pada awalnya menggunakan konsep intepreter ini

Bahasa pemrograman digunakan untuk mempermudah manusia dalam berinteraksi dengan komputer. Syarat utama untuk membuat program komputer adalah dengan menggunakannya sesuai dengan kaidah-kaidah yang berlaku dalam bahasa pemrograman

tersebut. Masing – masing bahasa pemrograman mempunyai ciri khas atau kaidah tersendiri. Oleh karena itu sebelum membuat sebuah program maka kita harus mengerti tentang aturan penulisan (sintax) dalam bahasa pemrograman tersebut.

1.2 Mengenal Bahasa Pemrograman

Saat ini banyak bahasa pemrograman yang beredar di pasaran. Masing – masing memberikan kemudahan dan fasilitas untuk membuat sebuah program komputer yang sesuai dengan keinginan.

a) FORTRAN

FORTRAN singkatan dari Formula Translation. Pertama kali dikembangkan pada tahun 1956 oleh John Backus di IBM. Ditujukan untuk mempermudah pembuatan aplikasi matematika, ilmu pengetahuan dan teknik. Merupakan bahasa pemrograman tingkat tinggi yang pertama. Keunggulan FORTRAN terletak pada dukungan untuk menangani perhitungan termasuk bilangan kompleks. Kelemahan bahasa ini terletak pada operasi masukan / keluaran yang sangat kaku. Selain itu kode sumbernya lebih sulit dipahami dibandingkan dengan bahasa pemrograman tingkat tinggi lainnya.

Contoh program dalam bahasa FORTRAN :

```
// JOB
// FOR
* ONE WORD INTEGERS
* IOCS(DISK,TYPEWRITER,KEYBOARD,PAPERTAPE)
      DIMENSION IEMG(10,15),IEMG1(13)
      DEFINE FILE 12(80,150,U,K)
      WRITE(1,10)
   10 FORMAT('PAPERTAPE'//'GIVE NUMBER EXPERIMENT (1-5 IN INT)')
      READ(6,30) M
   30 FORMAT(I1)
      PAUSE 1
      DO 25 N=1,16
      DO 15 I=1,15
      READ(4,20) IEMG1
   20 FORMAT(13I4)
      DO 15 J=4,13
      J3=J-3
   15 IEMG(J3,I)=IEMG1(J)
      NE=N+(M-1)*16
   25 WRITE(12'NE) IEMG
      CALL EXIT
      END
// DUP
*DELETE SJA1
*STORECI WS UA SJA1
*FILES(12,EMG)
```

a) COBOL

COBOL (Common Business Oriented Language) dikembangkan tahun 1959 dan tergolong sebagai bahasa tingkat tinggi. Sesuai dengan kepanjangan namanya, bahasa ini ditujukan untuk mempermudah pembuatan aplikasi di bidang bisnis. Sejauh ini bahasa ini

banyak digunakan di lingkungan komputer minikomputer dan mainframe. Keunggulan COBOL adalah :

-sintaksnya yang menggunakan kata-kata bahasa Inggris sehingga mempermudah programer

-kemudahan terhadap penanganan file

-kemudahan terhadap masukan/keluaran program

Contoh program dalam bahasa COBOL :

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400 DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500 DISPLAY "Hello world!" LINE 15 POSITION 10.
100600 STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800 EXIT.
```

c) BASIC

BASIC adalah singkatan dari *Beginner All-purpose Symbolic Instruction Code.* Dikembangkan tahun 1965 di Darmouth College. Penciptannya John Kemeny dan Thomas Kurtz. Awalnya BASIC digunakan sebagai pengajaran dasar untuk bahasa pemrograman sederhana. Keunggulan BASIC terletak pada kemudahannya untuk dipakai dan penggunaan bahasa Inggris yang mirip dengan kehidupan sehari – hari sebagai sintaksnya. BASIC merupakan bahasa pemrograman yang sangat populer sebelum Pascal dibuat.

Contoh program dalam bahasa BASIC :

```
REM Program mencari rata-rata 3 buah bilangan
INPUT "Masukkan tiga buah bilangan : ", a, b, c
rata=(a+b+c)/3
PRINT "Rata-rata ketiga bilangan adalah : "; rata
```

d) PASCAL

Sejarah perkembangan Pascal dimulai tahun 1960, yaitu ketika bahasa pemrograman ALGOL 60 digunakan sebagai Algorithmic Language yang digunakan untuk memecahkan masalah sehari – hari dengan menggunakan komputer. Nama Pascal

sendiri diambil dari nama seorang ahli matematika dan ilmu pengetahuan bangsa Perancis yaitu Blaise Pascal (1623 – 1662). Niklaus Wirth dari Sekolah Teknik Tinggi Zurich – Swiss, menjadi terkenal sebagai perancang bahasa Pascal. Keunggulan bahasa Pascal adalah keteraturan dalam pembuatan dan kelengkapan struktur data.

Contoh program dalam bahasa Pascal :

```
PROGRAM CariMin;
{Mencari Bilangan terkecil dari dua buah bilangan}
VAR
x,y,min:integer;
BEGIN
WRITE('Bilangan pertama : ');READLN(x);
WRITE('Bilangan kedua : ');READLN(y);
IF x>y THEN
Min:=y
ELSE
Min:=x;
WRITE('Bilangan terkecil : ',min);
END.
```

e) Bahasa C

Bahasa C diciptakan oleh Brian W. Kernighan dan Dennis M. Ritchie pada tahun 1972 di laboratorium Bell AT&T. Bahasa ini menggabungkan kemampuan pengendalian mesin dalam aras rendah dan struktur data serta struktur kontrol aras tinggi. Jadi dapat disebut bahasa C adalah bahasa pemrograman yang menggabungkan kemudahan pengontrolan hardware dalam bahasa pemrograman tingkat rendah serta struktur kontrol dalam bahasa tingkat tinggi. Bahasa C ini dipakai untuk menyusun sistem operasi UNIX dan Linux. Keunggulan bahasa C adalah :

- Sifat portabilitas, yaitu kode sumber pada sebuah platform dapat ditransfer ke platform lain tanpa ada perubahan
- kemudahan akses terhadap hardware
- cepat dan efisien

Pada tahun 1983, Bjarne Stroustrup mengembangkan bahasa C yang pada mulanya disebut sebagai ”a better C ”. Namun kemudian bahasa ini dikenal dengan nama C++ (C plus plus) yang mengunggulkan kelebihannya sebagai bahasa pemrograman berorientasi objek.

Contoh program dalam bahasa C :

```
/*Mencari Bilangan terkecil dari dua buah bilangan*/
#include <stdio.h>
main ()
{
int x,y, min;
printf ("Bilangan pertama : ");
scanf("%1f",&x);
printf ("Bilangan kedua : ");
scanf("%1f",&y);
if x>y
min=x
else
min=y;
printf("Bilangan terkecil : %1f\n",min);
}
```

e) Bahasa Java

Bahasa Java dikembangkan oleh Sun Microsystem pada tahun 1955. Merupakan bahasa yang berorientasi objek. Kode Java dikompilasi dalam format yang disebut bytecode. Bytecode ini dapat dijalankan di semua komputer yang telah dilengkapi dengan program Java intepreter dan Java Virtual Machine. Java sangat populer karena pada masa awal Internet menjadi populer, Java telah menyediakan sarana untuk membuat program (yang disebut sebagai applet) yang dapat berjalan pada web browser seperti Internet Explorer, Netscape Navigator.

Contoh program dalam bahasa Java :

```
Public class SayHello {
Public static void main(String[] args {
System.out.println("Hello world!");
}
}
```

Bab II Data dan Variabel

2.1 Tipe Data

Tipe data menentukan nilai yang dapat disimpan pada suatu variabel dan jenis operator yang dapat dikenakan pada variabel tersebut. Misalnya tipe data real hanya dapat menyimpan bilangan real dan operator yang dapat dikenakan padanya antara lain operator penjumlahan, pengurangan, perkalian, dll.

Tipe data dasar dalam bahasa C++ seperti tercantum dalam tabel 2.1.

Tipe	Jangkauan	Ukuran
char	-128…127	1 byte
int	-32768…32767	2 byte
long	-2.147.483.648…2.147.483.648	4 byte
float	$3.4x10^{-38}$ ... $3.4x10^{+38}$	4 byte
double	$1.7x10^{-308}$ ... $1.7x10^{+308}$	8 byte
long double	$3.4x10^{-4932}$ ... $3.4x10^{+4932}$	10 byte

Tabel 2.1 Tipe data dasar

Tipe data yang berhubungan dengan bilangan bulat adalah : char, int, dan long. Sedangkan tipe data yang lainnya berhubungan dengan bilangan pecahan atau real. Ukuran memori yang diperlukan untuk masing-masing tipe data adalah berbeda-beda seperti diperlihatkan dalam tabel 2.1.

2.2 Variabel

Variabel merupakan komponen penting dalam pemrograman. Variabel digunakan dalam pemrograman untuk menyimpan suatu nilai, dan nilai yang ada padanya dapat diubah selama eksekusi program berlangsung. Adapun ketentuan atau aturan dalam penulisan nama variabel adalah :

1. variabel bisa terdiri dari huruf, angka, atau under score (_)
2. variabel harus diawali oleh huruf

contoh yang benar :

```
nama_siswa
Latihan1
nilai1
```

contoh yang salah :

```
nama siswa   (mengandung spasi)
1Latihan     (diawali oleh angka)
```

3. Tidak boleh memakai kata kunci (lihat tabel 2.2)

asm	else	operator	template
auto	enum	private	this
break	extern	protected	typedef
case	float	public	union
char	for	register	unsigned
class	friend	return	virtual
const	goto	short	void
continue	if	signed	volatile
default	inline	sizeof	while
delete	int	static	
do	long	struct	
double	new	switch	

Tabel 2.2 Daftar kata - kata kunci (keyword)

2.2.1 Mendeklarasikan Variabel

Variabel yang akan digunakan dalam program harus dideklarasikan terlebih dahulu. Pengertian deklarasi di sini berarti mengenalkan sebuah variabel ke program dan menentukan tipe data yang bisa disimpan di dalamnya. Format atau sintax pendeklarasian variabel adalah:

```
tipe nama_variabel;
```

nama_variabel dapat berupa sebuah variabel atau beberapa variabel yang dipisahkan oleh koma.

contoh :

```
int jumlah;
float harga_per_unit, total_harga;
```

Pada deklarasi variabel jumlah menyatakan bahwa jumlah adalah variabel bertipe int (dipakai untuk menyimpan bilangan integer), sedangkan deklarasi variabel harga_per_unit dan harga_total berjenis float (untuk menyimpan data pecahan). Jika dikehendaki pendeklarasian variabel :

```
float harga_per_unit, total_harga;
```

bisa ditulis menjadi :

```
float harga_per_unit;
float  total_harga;
```

Setiap variabel harus ditentukan tipe datanya. Jika variabel akan dipakai untuk menyimpan data bilangan bulat saja, maka pilihannya adalah tipe bilangan bulat (seperti

int, long). Jika variabel hendak dipakai untuk menyimpan data bilangan pecahan, maka variabel harusi dideklarasikan bertipe bilangan pecahan (seperti float).

2.2.2 Memberikan Nilai ke Variabel (assignment)

Setelah variabel dideklarasikan maka kita dapat memberikan nilai pada variabel tersebut. Format atau sintax untuk memberikan nilai pada variabel adalah :

```
nama_variabel = nilai;
```

Pernyataan seperti diatas sering disebut sebagai pernyataan penugasan. Berikut ini adalah contoh pemberian nilai pada variabel.

```
int jumlah;
float harga_per_unit;
char huruf;
jumlah = 10;
harga_per_unit = 17.5;
huruf = 'B';
```

2.3 Konstanta

konstanta hampir sama fungsinya dengan variabel yaitu untuk menyimpan suatu nilai. Jika dalam variabel nilai yang disimpan dapat diubah selama eksekusi program berlangsung maka dalam konstanta nilai yang disimpan bersifat tetap (tidak berubah) selama eksekusi program. Ketentuan atau aturan dalam penulisan nama konstanta adalah sama dengan aturan dalam penulisan variabel.

Pendeklarasian konstanta didahului oleh kata kunci **const**.

```
const tipe nama_konstanta = nilai;
```

contoh :

```
const float phi = 3.141592;
const char huruf_awal = 'A' ;
const int MAKS = 10;
```

2.4 Masukan dan Keluaran

Pernyataan yang dipakai untuk memberikan masukan (input) dalam program bahasa C++ adalah : **cin**. Sedangkan pernyataan untuk menampilkan keluaran (output) pada layar adalah : **cout**. Untuk dapat memakai cin dan cout maka kita perlu menyertakan (include) file header yaitu : **iostream.h**.

2.4.1 Pernyataan Keluaran (Output) : cout

Pernyataan Keluaran berfungsi untuk menampilkan text ke layar (screen). Format pernyataan keluaran adalah :

```
cout<<"Text anda ";
        atau
cout<<"Text anda"<<endl;
```

Pernyataan cout yang pertama akan menampilkan text dalam tanda petik ke layar tanpa diikuti pindah baris baru sedangkan cout yang kedua akan menampilkan text dan diikuti pindah baris baru.

Contoh program :

```
//Program contoh output : cout
include<iostream.h>
void main()
{
        cout<<"Selamat Belajar Visual C++";
        cout<<"never say die!!!"<<endl;
        cout<<come and enjoy";
}
```

Jika program tersebut dieksekusi maka hasilnya adalah :

Selamat Belajar Visual C++ never say die
come and enjoy

2.4.2 Pernyataan Masukan (Input) : cin

Pernyataan masukan berfungsi untuk memasukkan data lewat keyboard. Format pernyataan masukan adalah :

```
cin>>nama_variabel;
```

Pernyataan cin akan meminta masukan dari keyboard dan diikuti pindah baris baru.

Contoh program :

1)
```
//Program hitung luas segi 4;
include <iostream.h>
void main()
{
        Int p,l,luas;
        cout<<"Masukkan Panjang : ";
        cin>>p;
        cout<<"Masukkan lebar : ";
        cin>>l;
        Luas=p*l;
        Cout<<"Luas segi empat = "<<luas;
}
```

Jika program tersebut dieksekusi maka hasilnya adalah :

Masukkan Panjang 10 (masukkan angka 10 dan tekan enter)

Masukkan lebar 5 (masukkan angka 5 dan tekan enter)

Luas segi empat = 50

2)
```
//Program luas lingkaran;
include<iostream.h>
void main()
{
    const float pi = 3.14;
    float luas,r ;
    cout<<"Masukkan jari-jari : ";
    cin>>r;
    luas = pi*r*r;
    cout<<"Luas lingkaran = "<<luas;
}
```

3)
```
//Program Energi Enstein
//program untuk menghitung energi yg terkandung dalam benda
include <iostream.h>
void main()
{
    const c=300000000;
    float  E,m ;
    cout<<"Masukkan massa benda : ";
    cin>>m;
    E=m*c*c;
    Cout<<"Energi yang terkandung dalam benda = "<<E);
}
```

Soal Latihan :

1. Buatlah program komputer untuk menghitung gaji total karyawan dimana rinciannya adalah :

   gaji total = gaji pokok + tunjangan + upah lembur

2. Buatlah program untuk menghitung panjang sisi miring sebuah segitiga siku-siku. Dimana rumusnya adalah :

   $c^2 = a^2 + b^2$

3. Buatlah program untuk menghitung akar dari persamaan kuadrat :

   $Ax^2 + Bx + C$

   dengan memakai rumus ABC :

$$x_1 = \frac{-b + \sqrt{b^2 - 4.A.C}}{2.A}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4.A.C}}{2.A}$$

Bab III Operator

Dalam bahasa pemrograman dikenal beberapa tipe operator yaitu : Operator Aritmatika, Operator Logika, dan Operator Relasi.

3.1 Operator Aritmatika

Operator aritmatika atau operator numerik biasanya dipakai untuk mengoperasikan 2 buah bilangan. Adapun yang tergolong dalam operator aritmatika adalah :

Operator	Fungsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	pembagian
%	modulo

Tabel 3.1. Operator Aritmatika

Contoh potongan program Visual C++ tentang pemakaian operator :

```
void main ()
{       int a = 3;
        int b = 5;
        int c,d,e,g;
        float f;
        c = a + b ;
        d = a – b ;
        e = c * d;
        f = c / d;   // F hasilnya berupa pecahan (float number)
        g = c % d;
        // dan seterusnya …
}
```

Jika program ini dieksekusi maka masing-masing variabel akan menyimpan nilai :

a = 5 d = 2 g = 2

b = 3 e = 15

c = 8 f = 1.67

Prioritas dalam operator Aritmatika

Adapun prioritas pemakaian operator aritmatika dalam bahasa pemrograman adalah :

- operator perkalian dan pembagian mempunyai prioritas yang sama
- operator penjumlahan dan pengurangan mempunyai prioritas yang sama
- operator perkalian atau pembagian mempunyai prioritas yang lebih tinggi daripada operator penjumlahan atau pengurangan

Latihan :

1. Potongan program berikut ini berisi tentang pemakaian operator aritmatika. Tentukan nilai yang disimpan oleh masing-masing variabel.

```
void main()
{
        int p,q,r,s,t;
        float u,v,w;
        p = 2;
        q = 3;
        r =  4;
        s = p * q + r ;
        t = p * (q + r);
        u = p+q / r;
        v = p*q/r;
        w=p+q-r;
        // dan seterusnya …
}
```

3.2 Operator Logika

Operator logika biasanya dipakai untuk membandingkan 2 pernyataan yang mengandung nilai logika benar (TRUE) atau salah (FALSE). Ada 3 operator logika yang umumnya dipakai dalam bahasa pemrograman komputer.

Operator	keterangan
&&	AND
||	OR
!	NOT

Tabel 3.2 Operator logika

A	B	A && B
0 (FALSE	0 (FALSE)	0 (FALSE)
1 (TRUE)	0 (FALSE)	0 (FALSE)
0 (FALSE	1 (TRUE)	0 (FALSE)
1 (TRUE)	1 (TRUE)	1 (TRUE)

Tabel 3.3 Tabel kebenaran operator AND (&&)

A	B	A || B
0 (FALSE	0 (FALSE)	0 (FALSE)
1 (TRUE)	0 (FALSE)	1 (TRUE)
0 (FALSE	1 (TRUE)	1 (TRUE)
1 (TRUE)	1 (TRUE)	1 (TRUE)

Tabel 3.4 Tabel kebenaran operator OR (||)

3.3 Operator Relasi

Operator relasi biasanya dipakai untuk membandingkan kuantitas dari 2 buah bilangan. Hasil dari perbandingan ini akan menghasilkan nilai logika : benar (TRUE) atau salah (FALSE).

Operator	keterangan
= =	sama dengan
!=	Tidak sama dengan
<	Lebih kecil
<=	Lebih kecil atau sama dengan
>	Lebih besar
>=	Lebih besar atau sama dengan

Tabel 3.5. Operator Relasi

Contoh :

A = 2, B = 3, C = 4

Operasi	Hasil operasi
A< B	TRUE
A != C/B	FALSE
C >= B	TRUE
B < C-A	FALSE
C >= B*A	FALSE
A == C	FALSE

Tabel 3.6 Contoh operasi dengan operator relasi

3.4 KOMENTAR (Comment)

Komentar merupakan pernyataan yang tidak akan ikut dieksekusi saat program dieksekusi. Komentar biasanya dipakai untuk memberikan keterangan atau penjelasan terhadap suatu kode dalam program. Dalam visual C++ komentar biasanya didahului oleh 2 buah karakter garis miring : //

```
#include <iostream.h>
void main()              // ini adalah fungsi utama
{
   int nilai_ujian;      // variabel untuk menyimpan nilai ujian
   cout << ”masukkan nilai ujian ”;
   cin >> nilai_ujian;   // masukkan nilai ujian
   if (nilai_ujian >=6)
   {
     cout << ”anda lulus ”;
     cout << ”selamat ”;
   }
   else
      cout << ”anda gagal”;
}
```

Bab IV Pernyataan Pengambilan Keputusan

4.1 STATEMEN if

Pernyataan yang biasanya dipakai untuk mengambil keputusan dari suatu kondisi adalah statemen *if*. Adapun sintax dari statemen *if* adalah sebagai berikut :

```
if  (kondisi)
{
     blok pernyataan
}
// dan seterusnya …
```

Dalam hal ini jika kondisi bernilai TRUE (atau 1) maka *blok pernyataan* akan dikerjakan sedangkan jika kondisi bernilai FALSE (atau 0) maka *blok pernyataan* tidak akan dikerjakan.

Contoh :

```
#include <iostream.h>
void main ()
{
    int nilai_ujian;
    cout << ”Masukkan nilai ujian = ”;
    cin >> nilai_ujian;
    if (nilai_ujian >= 6 )
    {
       cout << ” Anda Lulus “;
       cout <<endl;
       cout << ” Selamat ”;
     }
 cout << ”Tingkatkan terus semangat belajar ”;
}
```

Jika program diatas dieksekusi maka hasilnya adalah :

```
Masukkan nilai ujian = 7
Anda Lulus
Selamat
Tingkatkan terus semangat belajar
.....

Masukkan nilai ujian = 5
Tingkatkan terus semangat belajar
```

Dari hasil ini kita mendapatkan jika kita memasukkan nilai_ujian lebih besar atau sama dengan 6 maka akan ditampilkan pernyataan

```
Anda lulus
Selamat
Tingkatkan terus semangat belajar
```

Sedangkan jika kita memasukkan nilai_ujian lebih kecil dari 6 maka akan ditampilkan pernyataan :

```
Tingkatkan terus semangat belajar
```

4.2 STATEMEN if ... else

Format dari pernyataan pengambilan keputusan `if ...else` adalah sebagai berikut :

```
If (kondisi)
{
     blok pernyataan 1
}
else
{
    blok pernyataan 2
 }
```

Di sini diperlihatkan bahwa jika *kondisi* terpenuhi atau bernilai TRUE (1) maka *blok pernyataan 1* yang dikerjakan. Sedangkan jika kondisi bernilai FALSE (0) maka yang akan dikerjakan adalah *blok pernyataan 2*.

Contoh :

```
#include <iostream.h>
void main ()
{
    int nilai_ujian;
    cout << ”Masukkan nilai ujian = ”;
    cin >> nilai_ujian;
    if (nilai_ujian >= 6 )
    {
       cout << ” Anda Lulus “;
       cout <<endl;
       cout << ” Selamat ”;
     }
     else
     {
        cout << ” Maaf, Anda Tidak Lulus ”;
        cout << endl;
       cout << ”Belajar lagi”;
      }
}
```

Jika program diatas dieksekusi maka hasilnya adalah :

```
Masukkan nilai ujian = 7
```

```
Anda Lulus
Selamat
.....

Masukkan nilai ujian = 5
Maaf, Anda Tidak Lulus
Belajar lagi.
```

Latihan :

1. Buatlah program komputer yang dapat memilih nomer channel TV dan menampilkan stasiun TV yang bersangkutan. Andaikan nomer channel TV tersebut adalah bilangan integer dari 1 – 9 seperti berikut ini :

1. RCTI	5. ANTV	9. Bali TV
2. SCTV	6. TPI	
3. Indosiar	7. MetroTV	
4. Lativi	8. TVRI	

Petunjuk :

- Masukkan nomer chanel TV (angka 1 – 9)
- Tampilkan nama stasiun TV (pakai statemen if)

2. Andaikan kita diminta untuk mengubah nilai angka menjadi nilai huruf dari suatu data nilai ujian mahasiswa dengan ketentuan sebagai berikut :

   jika nilai_angka >=80 dan nilai_angka <=100 maka nilai_huruf = A
   jika nilai_angka >=60 dan nilai_angka <80 maka nilai_huruf = B
   jika nilai_angka >=40 dan nilai_angka <60 maka nilai_huruf = C
   jika nilai_angka >=20 dan nilai_angka <40 maka nilai_huruf = D
   jika nilai_angka >=0 dan nilai_angka <20 maka nilai_huruf = E

Petunjuk :

- masukkan nilai_angka
- gunakan statemen if dan untuk kondisi gunakan operator logika AND (&&)

  misal :

```
if (nilai_angka >=60 && nilai_angka <80)
{
    nilai_huruf = 'A';
    // dan seterusnya …
```

4.3 STATEMEN switch

Statemen switch biasanya dipakai untuk mengambil keputusan diantara banyak pilihan (kondisi). Adapun sintax dari statemen ini adalah :

```
Switch (ekspresi)
{
     case (ekspresi)       :
       {
              statemen1;
              break;
        }
     case (ekspresi)       :
        {
              statemen2;
              break;
        }
     case (ekspresi)       :
        {
              statemen3;
              break;
        }
     case (ekspresi)       :
       {
              statemen4;
              break;
        }
     default :
        {
          statemen4;
          break;
        }
}
```

Nilai ekpresi haruslah berupa tipe data *int* atau *char*. Potongan program berikut ini merupakan contoh pemakaian switch untuk nilai ekspresi bertipe data int.

```
#include <iostream.h>
void main()
{
        int no_channel;
        cout << ”masukkan nomer channel TV (1 – 4) = ”;
        cin >> no_channel;
        switch (no_channel)
        {
            case (1) :
                    {
                        cout << ”TVRI”;
                        break;
                    }
            case (2) :
                    {
                        cout << ”RCTI”;
                        break;
                    }
            case (3) :
                    {
                        cout << ”SCTV”;
                        break;
```

```
                    }
                case (4) :
                    {
                        cout << ”ANTV”;
                        break;
                    }
                default :
                    {
                        cout << ”Maaf, anda salah channel ! ”;
                        break;
                    }
            }
        }
```

Program berikut ini juga menampilkan contoh pemakaian switch dengan nilai ekspresi bertipe data *char*.

```
#include <iostream.h>
void main()
{
    char nilai_huruf;
    cout << ”masukkan nilai huruf (A,B,C,D)  = ”;
    cin >> nilai_huruf;
    switch (nilai_huruf)
    {
        case (’A’) :
            {
                cout << ”sangat memuaskan”;
                break;
            }
        case (’B’) :
            {
                cout << ”memuaskan”;
                break;
            }
        case (’C’) :
            {
                cout << ”cukup”;
                break;
            }
        case (’D’) :
            {
                cout << ”kurang”;
                break;
            }
        default :
            {
                cout << ”Maaf, anda salah memasukkan nilai ”;
                break;
            }
    }
}
```

Apakah hasil eksekusi dari program diatas?

Latihan :

1. Ulangi soal latihan no.1 diatas dengan menggunakan pernyataan *switch.*

2. Andaikan kita mempunyai 2 buah bilangan yaitu bil1 dan bil2. Kita akan mengoperasikan

kedua bilangan tersebut dengan pilihan sebagai berikut :

1. Penjumlahan
2. Pengurangan
3. Pekalian
4. Pembagian

misal jika kita pilih 1 maka program akan menjumlahkan bil1 dan bil2. Dan jika kita pilih 3 maka program komputer akan melakukan perkalian antara bil1 dan bil2. Dalam hal ini gunakanlah pernyataan *switch*.

Bab V PERNYATAAN PENGULANGAN (LOOPING)

Pernyataan pengulangan sering disebut *kalang*. Dalam pemrograman komputer ada beberapa kalang yang biasanya dipakai yaitu :

- **for**
- **while**
- **do-while**

Statemen pengulangan biasanya dipakai untuk menyatakan instruksi yang sama yang diulang berkali-kali, sehingga kode program menjadi lebih singkat. Seperti contoh sederhana misalnya kita diminta untuk menampilkan (cout) kalimat : "Selamat Tahun Baru Caka" secara berulang –ulang sebanyak 5 kali. Jika kita tidak memakai kalang maka potongan kode programnya adalah :

```
void main()
{
   cout <<"Selamat Tahun Baru Caka" <<endl;
   cout <<"Selamat Tahun Baru Caka" <<endl;
   cout <<"Selamat Tahun Baru Caka" <<endl;
   cout <<"Selamat Tahun Baru Caka" <<endl;
   cout <<"Selamat Tahun Baru Caka" <<endl;
   return;
}
```

Jika kita menggunakan kalang *for* maka kode program diatas dapat disingkat menjadi :

```
void main()
{
  for (i =1 ; i <=5 ; i++)
     {
        cout <<"Selamat Tahun Baru Caka" <<endl;
     }
}
```

5.1 Kalang for

Seperti diperlihatkan dalam contoh diatas maka format dari kalang *for* adalah :

```
for (ekspresiAwal; kondisi; ekspresiPencacah)
{
   //blok pernyataan
}
```

keterangan :

ekspresiAwal = nilai awal dari suatu nilai

kondisi = pernyataan logika yang mengetes apakah nilai memenuhi kondisi yang ditentukan

ekspresiPencacah = pernyataan untuk menaikkan/menurunkan cacahan suatu nilai.

Selama kondisi dalam kalang for dipenuhi maka blok pernyataan akan terus dikerjakan. Jika kondisi sudah tidak memenuhi maka blok pernyataan tidak akan dikerjakan dan keluar dari kalang. Misalkan kita ingin membuat kode program yang dapat mencetak (cout) 5 buah bilangan dari 1 sampai 5 maka kode programnya adalah sebagai berikut :

```
#include <iostream.h>
void main()
{
   int i;
   for  (i =1; i <= 5 ; i++)
   {
        cout << i;
        cout <<endl;
    }
}
```

Dari program diatas maka dalam kalang for mengandung pernyataan yaitu :

```
for  (i =1; i <= 5 ; i++)
```

yang berarti variabel *i* mempunyai nilai awal 1 (i =1) dan nilai *i* ini akan ditingkatkan sebesar satu (i++) setiap mengerjakan pernyataan dalam blok kalang *for*. Jika i sudah melebihi nilai 5 maka blok pernyataan kalang *for* tidak akan dikerjakan lagi.

Catatan :

Operator	contoh	ekivalen
++	i++	i = i +1
--	i--	i = i -1
+ =	bil + = 5	bil = bil +5
- =	bil - = 10	bil = bil - 10

Tabel 5.1 Operator overload

Latihan :

1. Buatlah kode program yang dapat mencetak bilangan dari 10 sampai 1.
2. Andaikan kita mempunyai kode program seperti dibawah ini. Maka tentukanlah hasil eksekusi dari kode tersebut.

```
#include <iostream.h>
void main()
{
    for (int i =1; i <=100; i +=2)
   {
```

```
            cout << i;
            cout <<endl;
        }
    }
```

3. Buatlah kode program yang dapat mencetak bilangan genap yang lebih kecil dari 50 dimulai dari 0.
4. Buatlah program yang dapat memasukkan 10 buah bilangan dan menampilkan hasil penjumlahan kesepuluh bilangan tersebut

   Petunjuk : - gunakan variabel array bertipe integer untuk bilangan

   contoh : int bil[10];

   - pakailah kalang for :

```
for (i = 0; i<10; i++)
{
    cout <<"masukkan bilangan ke-"<<i;
    cin >> bil[i];
    total += bil[i];
}
```

5. Andaikan kita diminta untuk membuat kode program yang dapat dipakai untuk memasukkan 10 nama orang beserta nomer teleponnya. Bagaimanakah bentuk kode programnya.

   Petunjuk : - gunakan variabel array bertipe char untuk nama dan no. telepon

   contoh : char nama[10][30];

   - pakailah kalang *for* :

```
for (i = 0; i<10; i++)
{
    cout <<"masukkan nama ke-"<<i;
    cin >> nama[i];
    //dan seterusnya…
}
```

5.2 Kalang while

Kalang *while* mirip dengan kalang *for* yaitu dipakai untuk mengulangi pernyataan yang sama berkali-kali selama kondisi terpenuhi. Adapun format atau sintax dari kalang *while* adalah :

```
while (ekspresi)
{
    //blok pernyataan
}
```

Ekspresi merupakan operasi relasi yang melibatkan operator-operator relasi . Hasil dari operasi relasi ini adalah TRUE (1) atau FALSE (0). *Blok pernyataan* akan diulangi

selama hasil operasi relasi (ekspresi) bernilai TRUE (1). Program berikut adalah contoh pemakaian kalang while :

```
// Program untuk mencetak “selamat belajar Visual C++” sebanyak 10 kali
// dengan kalang while
#include <iostream.h>
void main()
{
    int i;
    i =1;
    while (i < = 10)
    {
        cout << ”selamat belajar Visual C++ ”;
        cout <<endl;
    }
}
```

Berikut ini adalah contoh program untuk memasukkan nilai 5 buah bilangan dengan memakai kalang *while* :

```
//Program untuk memasukkan 5 buah bilangan dengan kalang while
#include <iostream.h>
void main()
{
     int i;
     float bil[5]; //variabel array dengan indeks maksimum 5
     i =0;
     while (i <  5 )
     {
        cout << ”masukkan bilangan ke –  ”<< i ;
        cin >> bil[i];
        i++;
  }
}
```

latihan :

1. Buatlah program komputer yang dapat memasukkan nilai 5 buah bilangan dan selanjutnya dapat menghitung nilai rata-rata dari bilangan tersebut. (gunakan kalang while).
2. Implementasikan kode program yang dapat menampilkan bilangan ganjil antara 0 – 50 dengan memakai kalang *while*

   Petunjuk :

   -gunakan kalang *while* yaitu :

```
bil = 1;
while (bil < 50)
{
    cout << bil <<endl;
    bil +=2;
```

```
    //dan seterusnya …
```

3. Buatlah kode program yang dapat menampilkan dan menghitung hasil penjumlahan bilangan genap dari 0 sampai 50.

   Petunjuk :

   -tampilkan bilangan genap dari 0 - 50

   -hitung hasil penjumlahan dari bilangan genap tersebut misal :

   total_jumlah = 0 + 2 + 4 +…+50

   -gunakan kalang *while*

5.3 Kalang do-while

Kalang *do-while* merupakan bentuk lain dari kalang *while*. Dalam kalang *while* ekspresi penguji berada di awal sedangkan dalam kalang *do-while* ekspresi penguji berada di akhir. Adapun format penulisan dari kalang *do-while* adalah :

```
do
{
    //blok pernyataan
}
while (ekspresi)
```

Blok pernyataan akan terus dieksekusi selama ekspresi bernilai benar (TRUE). Jadi dalam hal ini blok pernyataan paling tidak akan dieksekusi sekali sebelum mengalami pengujian ekspresi. Kode program berikut ini merupakan contoh sederhana pemakaian kalang *do - while* :

```
// Program pemakaian kalang do-while
// mencetak kalimat  ”Save our planet, keep violence away!”  // sebanyak 5 kali
#include <iostream.h>
void main()
{
    int i =1;
    do
    {
        cout <<”Save our planet, keep violence away!”<<endl;
        i++ ;
    }
    while (i <=5)
}
```

Latihan :

1. Ulangi soal-soal diatas dengan memakai kalang *do-while*.

Bab VI ARRAY

Array (larik) merupakan variabel berindeks yang menyimpan sekelompok data yang mempunyai tipe data yang sama. Array dapat terdiri : 1 dimensi atau multi dimensi (2 dimensi, 3 dimensi,...).

5.1 Array 1 Dimensi

Array ini hanya mempunyai 1 dimensi. Adapun sintaks atau format penulisan dari Array 1 dimensi dalam Visual C++ adalah :

```
TipeVar   NamaVar[index];
```

Keterangan :

TipeVar = tipe data dari variabel array

NamaVar = nama variabel dari array

Indeks = bilangan yang menyatakan nilai maximum indeks

Contoh :

```
Int nilai[5];
```

Ini berarti bahwa variabel array yang bernama *nilai* dengan index maksimum 5. Jadi variabel array ini dapat dijabarkan menjadi : *nilai[0], nilai[1], nilai[2],…,nilai[5]*. Indeks dari array dimulai dari 0. Tapi kita tidak harus mengawali indeks array dari 0, bisa saja dari 1,2,...

Untuk memberikan nilai (assignment) suatu variabel array dapat ditentukan dengan :

```
nilai[index] = nilai;
```

Contoh :

```
nilai[1] = 10;
nilai[2] = 20;
dst…
```

Untuk membuat array dengan tipe data karakter (char) maka cara mendeklarasikannya adalah sebagai berikut :

```
TipeVar   NamaVar[index][panjang_char];
```

Keterangan :

TipeVar = tipe data karakter

NamaVar = nama variabel dari array

Indeks = bilangan yang menyatakan nilai maximum indeks

Panjang_char = panjang karakter maksimum dari variabel array tersebut.

Contoh :

```
Char nama_pegawai[10][30];
```

Variabel nama_pegawai memiliki sepuluh index dan panjang masing-masing adalah maksimum 30 karakter (alfanumerik) seperti contoh berikut ini :

```
nama_pegawai[1] = 'Lestari'
nama_pegawai[2] = 'Hendrikus'
```

Berikut ini adalah potongan program Visual C++ yang memperlihatkan tentang pemakaian variabel array.

```
void main ()
{
        float bil[3] ;
        cout <<"masukkan nilai bilangan ke-0 = ";
        cin >>bil[0];
        cout <<"masukkan nilai bilangan ke-1 = ";
        cin >>bil[1];
        cout <<"masukkan nilai bilangan ke-2 = ";
        cin >>bil[2];
}
```

Dalam potongan program diatas kita diminta untuk memasukkan nilai 3 buah bilangan yaitu :

bil[0],bil[1], …,bil[2]. Pada program berikut ini kita diminta untuk memasukkan lima buah bilangan (cin) dan kemudian menghitung jumlah dari 5 buah bilangan tersebut. Serta akhirnya menampilkan(cout) hasil penjumlahannya.

```
#include <iostream.h>
void main()
{
        int bil[5];
        int jumlah;
        cout << "masukkan bilangan ke-0 =";
        cin >>bil[0];
        cout <<"masukkan bilangan ke-1=";
        cin>>bil[1];
        cout << "masukkan bilangan ke-2 =";
        cin >>bil[2];
        cout <<"masukkan bilangan ke-3=";
        cin>>bil[3];
        cout << "masukkan bilangan ke-4 =";
        cin >>bil[4];
        jumlah = bil[0]+bil[1]+bil[2]+bil[3]+bil[4];
        cout <<"Hasil penjumlahan adalah = " <<jumlah;
}
```

Latihan :

1.Buatlah program yang menghitung nilai rata-rata dari 5 buah bilangan.

Petunjuk :

-masukkan nilai dari 5 buah bilangan tersebut (pakai variabel array)

-hitung jumlahnya

-hitung rata-rata

-tampilkan nilai rata-rata (cout)

2.Buatlah program yang dapat memasukkan nama 5 orang mahasiswa beserta no. teleponnya. Selanjutnya tampilkan (cout) data-data ke lima orang mahasiswa tersebut.

Petunjuk :

-masukkan nama-nama dan nomer telepon dari mahasiswa (dengan variabel array bertipe char)

-tampilkan nama dan nomer telepon dari mahasiswa tersebut (cout)

-potongan programnya adalah :

```
void main()
{
        char nama_mhs[5][30];
        char no_phone[5][6];
        cout <<"Nama mahasiswa ke-0 =";
        cin >>nama_mhs[0];
        cout <<"No.Telephon mahasiswa ke-0 = ";
        cin >>no_phone[0];
        //dan seterusnya …
```

5.2 Array 2 Dimensi

Array yang sudah kita bahas sebelumnya adalah merupakan array 1 dimensi. Array 2 dimensi terdiri dari 2 dimensi yaitu baris dan kolom. Array 2 dimensi sering dipakai untuk menyatakan matrik.

Misal :

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Matrik A adalah matrik dua dimensi dengan 2 baris dan 2 kolom. Sintax untuk menyatakan variabel array 2 dimensi dalam Visual C++ adalah :

```
TipeVar namaVar[baris][kolom]
```

Dimana,

TipeVar = tipe variabel array

NamaVar = nama variabel array

Baris = jumlah baris maksimum

Kolom = jumlah kolom maksimum

Contoh :

int matrikA[2][2]

float matrikB[3][3]

Untuk memberikan nilai variabel array 2 dimensi dilakukan dengan :

```
NamaVar[i][j] = nilai
```

Contoh potongan program :

```
void main()
{
        int matrikA[2][2];

        matrikA[1][1] = 5;
        matrikA[1][2] = 3;
        matrikA[2][1] = 1;
        matrikA[2][2] = 2;

    dan seterusnya      …
```

Dari potongan program tersebut kita mendeklarasikan variabel array 2 dimensi dengan nama *matrikA*. Dimana banyaknya baris dan kolom matrikA adalah 2. Tipe data dari matrik ini adalah integer.

Latihan :

1.Buatlah sebuah program yang dapat dipakai untuk memasukkan array 2 dimensi dengan jumlah baris 3 dan jumlah kolom 3 (matrik 3 X 3). Kemudian tampilkan elemen-elemennya

Petunjuk :

- masukkan nilai masing-masing elemen matrik (cin)
- tampilkan masing-masing elemennya (cout)

2.Tulislah program komputer untuk menghitung hasil penjumlahan 2 buah matrik 2 dimensi. Misal hasil penjumlahan matrik A dan matrik B disimpan dalam matrik C seperti persamaan berikut ini :

$$C = A + B$$

Petunjuk :

-masukkan nilai elemen-elemen matrik A

-masukkan nilai elemen-elemen matrik B

-jumlahkan nilai elemen-elemen matrik A dengan B yang bersesuaian

-simpan hasil penjumlahan tiap-tiap elemen dalam matrik C

```
C[1][1] = A[1][1] + B[1][1];
C[1][2] = A[1][2] + B[1][2];  dan seterusnya …
```

-tampilkan matrik C.

3.Rancang program komputer untuk menampilkan transpose dari suatu matrik.

Petunjuk :

-masukkan nilai elemen-elemen matrik A

-tentukan traspose dari A.misal transpose dari A adalah matrik B. Sehingga nilai elemen elemen dari B :

B[1,1] = A[1,1]

B[1,2] = A[2,1]

B[2,1] = A[1,2]

Bab VII Fungsi (Function)

Selama ini kita sudah membuat kode program hanya dengan sebuah fungsi yaitu *main()*. Dimana seluruh kode program ditulis dalam blok pernyataan dari fungsi main().

```
.....
.....
void main()
{
    //blok pernyataan
}
```

Penulisan kode program yang sangat panjang sangat sulit untuk dipahami.Terlebih lagi ketika kita melacak kesalahan (error) saat proses kompilasi.

Untuk mengatasi masalah tersebut maka dalam pemrograman terstruktur diperkenalkan istilah fungsi. Dimana suatu masalah/program akan dipecah menjadi beberapa bagian yang disebut dengan fungsi. Masing-masing bagian ini akan mengerjakan tugas-tugas tertentu. Format atau sintaks penulisan fungsi adalah :

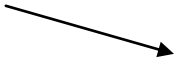
```
void namafungsi (daftar parameter)
{
    //blok pernyataan fungsi
}
```

contoh :

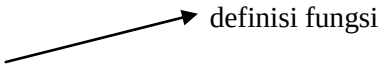
```
void hitung_luas_segi4( int panjang, int lebar)
{
   //blok pernyataan
}
```

contoh tersebut mendefinisikan fungsi yang bernama *hitung_luas_segi4*, dengan parameter *panjang* dan *lebar* yang bertipe integer. Sebelum fungsi didefinisikan maka dia perlu *dideklarasikan* dalam program utama (fungsi main()). Atau sering disebut dengan mengadakan *prototype fungsi*. Adapun deklarasi/prototype fungsi dapat dinyatakan seperti berikut ini :

```
#include <iostream.h>
void hitung_luas_segi4(int panjang, int lebar);
void main()
{
    // blok pernyataan fungsi utama main()
```

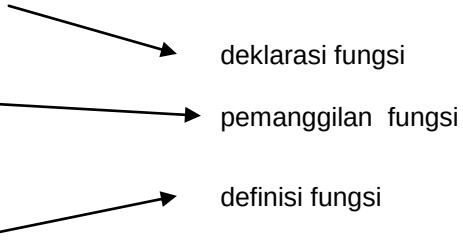


```
}

void hitung_luas_segi4(int panjang, int lebar);
{
    //blok pernyataan fungsi hitung_luas_segi4
}
```



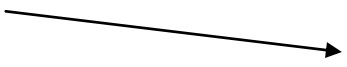
untuk mengaktifkan fungsi maka kita harus memanggil fungsi tersebut. Contoh berikut ini memperlihatkan pemanggilan fungsi hitung_luas_segi4 dari fungsi utama main().

```
#include <iostream.h>
void hitung_luas_segi4(int panjang, int lebar);
void main()
{
    hitung_luas_segi4(panjang.lebar);
    //dan seterusnya …
    return;
}

void hitung_luas_segi4(int panjang, int lebar);
{
    //blok pernyataan fungsi hitung_luas_segi4
}
```



Dari potongan kode diatas dapat dilihat pemanggilan fungsi dilakukan dengan menyebut nama fungsi (hitung_luas_segi4) dan diikuti oleh argumen (panjang dan lebar).
Salah satu contoh program sederhana tentang pemakaian fungsi diberikan dalam contoh berikut ini.

```
#include <iostream.h>
void cetak_tulisan(int bil);
void main()
{
    cetak_tulisan(10);    //pemanggilan fungsi dengan argumen
}
void cetak_tulisan(int bil)
{
  for (int i=1; i <=bil; i++)
  {
          cout <<"Manusia yang besar adalah manusia yang tidak pernah mengeluh";
          cout <<endl;
   }
}
```



Kode program diatas akan mencetak kalimat “Manusia yang besar adalah manusia yang tidak pernah mengeluh” sebanyak 10 kali. Dalam fungsi main dilakukan pemanggilan fungsi *cetak_tulisan* dengan argumen bernilai 10. Begitu fungsi cetak_tulisan ini dipanggil maka kontrol program akan berpindah ke fungsi cetak_tulisan. Nilai argumen 10 akan diterima oleh parameter bil dari fungsi cetak_tulisan. Sehingga nilai paramater bil

dalam fungsi cetak_tulisan menjadi 10. Seluruh kode program dalam fungsi cetak_tulisan akan dikerjakan sampai selesai. Setelah semua kode dalam fungsi cetak_tulisan selesai dikerjakan maka kontrol program akan kembali pada fungsi utama main().
Kalau kode program dalam fungsi main() diubah menjadi :

```
void main()
{
    cetak(10);
    cetak(20);
}
```

Maka hasil eksekusi program diatas adalah akan menampilkan kalimat ”Manusia yang besar adalah manusia yang tidak pernah mengeluh” sebanyak 30 kali.
Pengiriman data lewat argumen saat pemanggilan fungsi dapat dilakukan dengan 3 cara antara lain :

- pengiriman dengan nilai
- pengiriman dengan alamat
- pengiriman dengan referensi.

7.1 Pengiriman dengan nilai

Dalam pengiriman data dengan nilai maka nilai-nilai argumen dari fungsi pemanggil akan langsung dikirimkan kepada fungsi yg dipanggil. Selanjutnya nilai ini akan diterima oleh parameter- parameter dari fungsi yang dipanggil.
Contoh :

```
#include <iostream.h>
void cetak_variabel(int x,char y);  //prototype fungsi
void main()
{
     int x;
     char y;
     x = 5;
     y = 'A';
     cetak_variabel(x,y);          //panggil fungsi
     cetak_variabel(2,'C');
     return;
}

void cetak_variabel(int x, char y)          //definisi fungsi
{
     cout <<"nilai x = "<< x  <<endl;
     cout <<"nilai y = " << y <<endl;
     return;
}
```

hasil eksekusi dari program tersebut diatas adalah :

nilai x = 5

nilai y = A

nilai x = 2

nilai y = C

7.2 Pengiriman dengan Alamat

Pengiriman data dengan alamat dilakukan dengan mengirimkan variabel dari argumen tersebut secara langsung. Sehingga dalam hal ini fungsi yang dipanggil akan dapat mengubah nilai variabel yang bersangkutan. Format penulisan dari pengiriman data dengan alamat hampir sama dengan pengiriman dengan nilai hanya bedanya disini adalah kita harus menyertakan tanda & pada argumen fungsi pemanggil dan tanda asterisk (*) pada parameter fungsi yang dipanggil dan prototype fungsi.

```
#include <iostream.h>
void ubah(int *i);   //prototype fungsi
void main()
{
    ubah(&i)  //pemanggil fungsi
    //dan seterusnya …
}
void ubah(int *i)    //definisi fungsi
{
   //blok pernyataan
}
```

Contoh :

```
#include <iostream.h>
void ubah_variabel(int *x , y);
void main()
{
    int x,y;
    x =5;
    y =3;
    cout << “nilai x sebelum memanggil fungsi adalah “ << x  <<endl;
   cout << “nilai y sebelum memanggil fungsi adalah “ << y <<endl;
    ubah_variabel(&x , y);  // x dikirim dengan alamat, y dikirim dengan nilai
    cout <<”nilai x sesudah memanggil fungsi adalah “ << x <<endl;
    cout <<”nilai y sesudah memanggil fungsi adalah “ << y <<endl;
}
void ubah_variabel(int *x , y)
{
     *x = 10;
     y = 15;
}
```

hasil eksekusi program ini adalah :

nilai x sebelum memanggil fungsi adalah 5

nilai y sebelum memanggil fungsi adalah 3

nilai x sesudah memanggil fungsi adalah 10

nilai y sesudah memanggil fungsi adalah 3

7.3 Pengiriman dengan Referensi

Pengiriman data dengan referensi hampir sama dengan pengiriman dengan alamat, hanya saja dalam pengiriman dengan referensi kita perlu menambahkan tanda & pada parameter fungsi yang dipanggil. Dalam pengiriman data dengan referensi maka fungsi yang dipanggil dapat mengubah nilai dari variabel fungsi pemanggil.

```
#include <iostream.h>
void ubah(int  &i);   //prototype fungsi
void main()
{
    ubah(i)  //pemanggil fungsi
    //dan seterusnya …
}
void ubah(int &i)    //definisi fungsi
{
   //blok pernyataan
}
```

contoh :

```
#include <iostream.h>
void ubah_variabel(int &x , y);
void main()
{
    int x,y;
    x =5;
    y=3;
    cout << “nilai x sebelum memanggil fungsi adalah “ << x <<endl;
    cout << “nilai y sebelum memanggil fungsi adalah “ << y <<endl;
    ubah_variabel(x , y);  // x dikirim dengan referensi, y dikirim dengan nilai
    cout <<”nilai x sesudah memanggil fungsi adalah “ << x <<endl;
    cout <<”nilai y sesudah memanggil fungsi adalah “ << y <<endl;
    return;
}

void ubah_variabel(int &x , y)
{
     x = 10;
     y = 15;
     return;
}
```

hasil eksekusi program ini adalah :

nilai x sebelum memanggil fungsi adalah 5

nilai y sebelum memanggil fungsi adalah 3

nilai x sesudah memanggil fungsi adalah 10

nilai y sesudah memanggil fungsi adalah 3

7.4 Memecah dengan Fungsi

Untuk memecah program yang panjang maka dapat dilakukan dengan memecah program tersebut menjadi beberapa fungsi. Dimana masing-masing fungsi bertugas untuk mengerjakan tugas tertentu sesuai dengan keperluan. Misalnya kita ingin membuat program untuk menghitung volume dari sebuah kotak. Jika kita tidak memakai fungsi maka program tersebut adalah :

```
#include <iostream.h>
void main()
{
      int panjang,lebar, tinggi, vol;
      cout <<"masukkan panjang = ";
      cin >>panjang;
      cout <<"masukkan lebar = ";
      cin >>lebar;
      cout <<"masukkan tinggi = ";
      cin >>tinggi;
      vol = panjang*lebar*tinggi;
      cout << " Volume dari kotak adalah = " <<vol;
      cout <<endl;
}
```

Jika kita menggunakan fungsi maka program diatas dapat dibagi menjadi beberapa bagian yaitu :

- memasukkan dimensi kotak
- menghitung volume
- menampilkan hasil perhitungan volume kotak

Sehingga kalau dinyatakan dalam fungsi maka bentuk kode programnya :

- void dimensi (int &panjang, int &lebar, int &tinggi) ;
- void hitung_vol(int panjang, int lebar, int tinggi, int &vol);
- void cetak_vol(int vol);

Bentuk kode program selengkapnya sebagai berikut :

```
#include <iostream.h>
void dimensi(int &panjang, int &lebar, int &tinggi);
void hitung_vol(int panjang, int lebar, int tinggi, int &vol);
void cetak_vol(int vol);
void main()
{
        int panjang,lebar,tinggi,vol;
        dimensi(panjang,lebar,tinggi);
        hitung_vol(panjang,lebar,tinggi,vol);
        cetak_vol(vol);
        return;
```

```
}

void dimensi(int &panjang, int &lebar, int &tinggi)
{
        cout <<"masukkan panjang = ";
        cin >>panjang;
        cout <<"masukkan lebar = ";
        cin >>lebar;
        cout <<"masukkan tinggi = ";
        cin >>tinggi;
}

void hitung_vol(int panjang, int lebar, int tinggi, int &vol)
{
        vol = panjang*lebar*tinggi;
        return;
}

void cetak_vol(int vol)
{
        cout <<" volume kotak adalah = "<<vol;
        return;
}
```

7.5 VARIABEL LOKAL DAN GLOBAL

Pengertian tentang variabel global dan lokal sangatlah penting untuk dipahami. Terutama dalam hubungannya dengan pendefenisian fungsi. Sebuah variabel dapat berlaku global maupun lokal tergantung letak kita mendefinisikannya dalam program.

7.5.1 Variabel lokal

Variabel lokal adalah variabel yang hanya berlaku di dalam blok fungsi tempat variabel tersebut dideklarasikan.

Contoh :

```
#include <iostream.h>
void coba1();
void main()
{
    int x1;
    char huruf1;
    coba1();  //memanggil fungsi coba1

    //dan seterusnya …
}

void coba1()
{
    int x2;
    char huruf2;
    //dan seterusnya …
}
```

variabel *x1* dan *huruf1* adalah lokal dan hanya berlaku dalam fungsi *main()* dan tidak akan berlaku dalam fungsi *coba1*. Sedangkan dalam fungsi *coba1* juga dideklarasikan variabel lokal *x2* dan *huruf2*. Variabel *x2* dan *huruf2* ini hanya berlaku dalam fungsi *coba1* dan

tidak berlaku dalam fungsi *main().* Variabel x2 dan huruf2 akan lenyap begitu fungsi *coba1* selesai dikerjakan.

contoh :

```
#include <iostream.h>
void coba1();
void main()
{
     int x1;
     char huruf1;
     x1 = 100;
     huruf1= 'A';
     cout << x1;
     cout <<huruf1;
     coba1();  //memanggil fungsi coba1

     //dan seterusnya …
}

void coba1()
{
     int x2;
     char huruf2;
     x2 =200;
     huruf2 = 'B';
     cout << x2;
     cout << huruf2;
     cout << x1;
     cout << huruf1;
     //dan seterusnya …
}
```

Benar ! sebab var x1 dan huruf1 berlaku lokal dalam fungsi main

Benar ! sebab var x2 dan huruf2 berlaku lokal dalam fungsi coba1

Salah ! sebab var x1 dan huruf1 berlaku lokal dalam fungsi main dan tidak dikenal dalam fungsi coba1

7.5.2 Variabel Global

Variabel global adalah variabel yang berlaku bagi seluruh fungsi baik fungsi main() maupun fungsi-fungsi yang lain dalam program. Variabel global biasanya dideklarasikan sebelum pendefinisian fungsi main() dan setelah pernyataan preprosesor : #include.

Contoh :

```
#include <iostream.h>
int x1=100;
char huruf1= 'A';
void coba1();
void main()
{

     cout << x1;
     cout << huruf1;
     coba1()
     return;
}
void coba1();
{

}
```

deklarasi variabel global *x1* dan *huruf1*

variabel global *x1, huruf1* dapat dikenali dalam fungsi main

```
    cout << x1;          variabel global x1,huruf1 dapat dikenali dalam fungsi coba1
    cout <<huruf1;
    return;
}
```

Contoh program berikut akan lebih memperdalam pemahaman tentang variabel global :

```
#include <iostream.h>
int x1;
char huruf1;
void coba1();
void main()
{
    x1 = 10;
    huruf1 = 'A';
    cout << x1;
    cout << endl;
    cout << huruf1;
    coba1()
    cout << x1;
    cout << endl;
    cout << huruf1;
    return;
}
void coba1();
{
    x1 = 20;
    huruf1 = 'B';
    cout << x1;
    cout << endl;
    cout <<huruf1;
    return;
}
```

Hasil eksekusi dari program tersebut diatas adalah :

10
A
20
B
20
B

Contoh program berikut ini adalah sebuah program tentang pemakaian variabel global dan lokal. Tentukan apakah hasil eksekusi dari program tersebut.

```
#include <iostream.h>
int global1;
void cetak();
void main()
{
    int lokal1;
    lokal1=10;
    global1=20;
    cout <<lokal1;
    cout <<endl;
    cout <<global1;
    cetak();
    cout << lokal1;
```

```
    cout <<endl;
    cout << global1;
    return;
}
void cetak()
{
    int lokal2;
    lokal2=5;
    global1=30;
    cout << lokal2;
    cout <<endl;
    cout << global1;
    return;
}
```

Variabel global baru akan lenyap setelah semua pernyataan dalam program selesai dieksekusi. Sebelum selesai maka variabel global akan terus menempati memori komputer, sehingga sangat disarankan untuk memakai variabel global seperlunya.

7.6 Fungsi Inline

Fungsi *Inline* adalah sebuah fungsi yang biasanya dipakai untuk mengembalikan sebuah nilai lewat pernyataan *return.* Cara mendeklarasikan fungsi *inline* adalah dengan menyebutkan tipe data di depan nama fungsi. Tipe data ini disesuaikan dengan nilai yang akan dikembalikan oleh fungsi lewat pernyataan *return*. Sedangkan pemanggilan fungsi *inline* dilakukan dengan menugaskan fungsi ini ke sebuah variabel. Misalkan ada sebuah fungsi *inline* yang bernama *coba1,* maka bentuk pendeklarasian dan pemanggilan fungsi tersebut adalah :

```
#include <iostream.h>
float coba1(float a, float b);
void main()
{
    float nilai,a,b;
    a=5.5;
    b=10.5;
    nilai = coba1(a,b);  // memanggil fungsi inline coba1
    //dan seterusnya…
}
float coba1( float a, float b)
{
    float L;
    L = a*b;
    return L;  //mengembalikan nilai lewat return
}
```

contoh :

```
//Pemakaian fungsi inline untuk menghitung luas segi 4
#include <iostream.h>
int luas_segi4(int panjang, int lebar);
void main()
{
    int panjang,lebar,luas;
    cout << “masukkan panjang = “;
```

```
        cin >> panjang;
        cout << “masukkan lebar = “;
        cin >>lebar;
        luas= luas_segi4(panjang,lebar);
    }

    int luas_segi4(int panjang,int lebar)
    {
        int L;
        L=panjang*lebar;
        Return L;
    }
```

7.7 Rekursi

Rekursi adalah kemampuan suatu fungsi untuk memanggil dirinya sendiri. Jadi dengan rekursi memungkinkan anda untuk memanggil suatu fungsi dari dalam fungsi itu sendiri. Pemanggilan suatu fungsi dilakukan secara berulang-ulang sampai tercapai titik berhenti yang disebut *sentinel point.*

Contoh dari proses rekursi dapat dilihat dalam perhitungan faktorial. Dimana nilai faktorial dari suatu bilangan dapat ditentukan dengan :

$$n! = n*(n-1)! \quad ; jika\ n>1$$
$$=1 \quad ; jika\ n \le 1$$

contoh :

$$4! = 4*3!$$
$$3! = 3*2!$$
$$2! = 2*1!$$
$$1! = 1$$

$3! = 3*2!$, $2! = 2*1!$ $\longrightarrow n>1$

$1! = 1$ $\longrightarrow n \le 1$ (*titik berhenti / sentinel*)

$$= 4*3*2*1 = 24$$

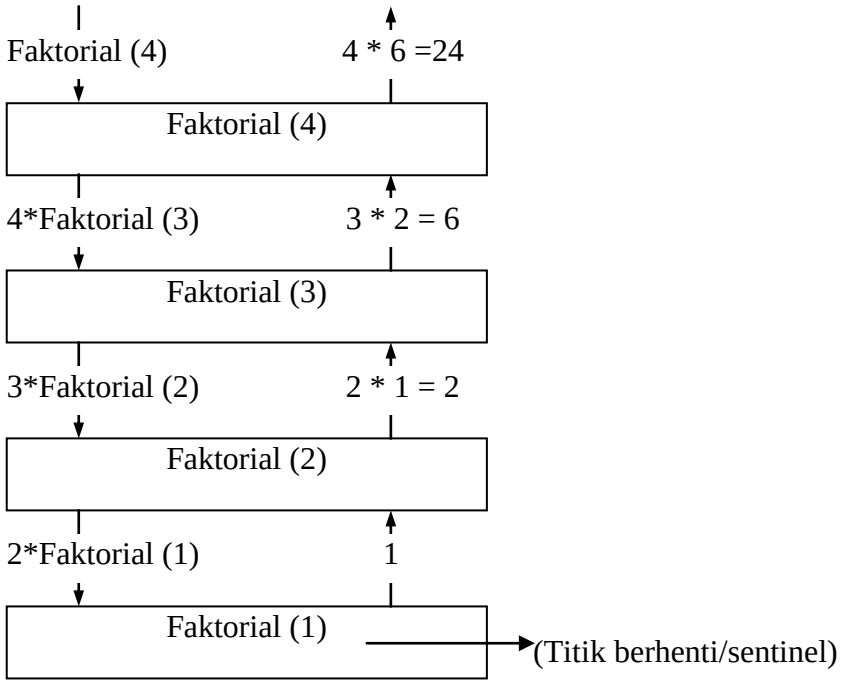
contoh program faktorial :

```
    #include <iostream.h>
    long fak(int n);
    void main()
    {
        int a;
        cout<<"Masukkan faktorial = ";
        cin>>a;
        cout<<fak(a);  //panggil fungsi
        cout<<endl;
    }

    long fak(int n)
    {
        int f;
        if (n<=1)
            f=1;
        else
            f=n*fak(n-1); //proses rekursi
        return f;
    }
```

Hasil eksekusi dari program tersebut adalah :

```
Masukkan faktorial = 5
125
Press any key to continue
```

Adapun proses kerja dari rekursi dari program diatas adalah seperti gambar (7.1) :



Gambar 7.1 Proses rekursi fungsi untuk menghitung faktorial (n = 4)

Dalam contoh berikut akan diberikan proses rekursi pada bilangan Fibonacci. Bilangan Fibonacci didefinisikan :

$f_n = f_{n-1} + f_{n-2}$, untuk n>2

$= 0$, untuk n = 0

$= 1$, untuk n = 1

berikut ini adalah barisan bilangan Fibonacci yang dimulai dari n =1 :

1 ,1, 2, 3, 5, 8, 13, 21, …

contoh untuk n = 4 maka proses perhitungan bilangan Fibonacci dapat ditentukan dengan cara :

$f_4 = f_3 + f_2$

$= (f_2 + f_1) + (f_1 + f_0)$

$= ((f_1+f_0)+f_1) + (f_1+f_0)$

$= ((1+0)+1)+(1+0)$

= 3

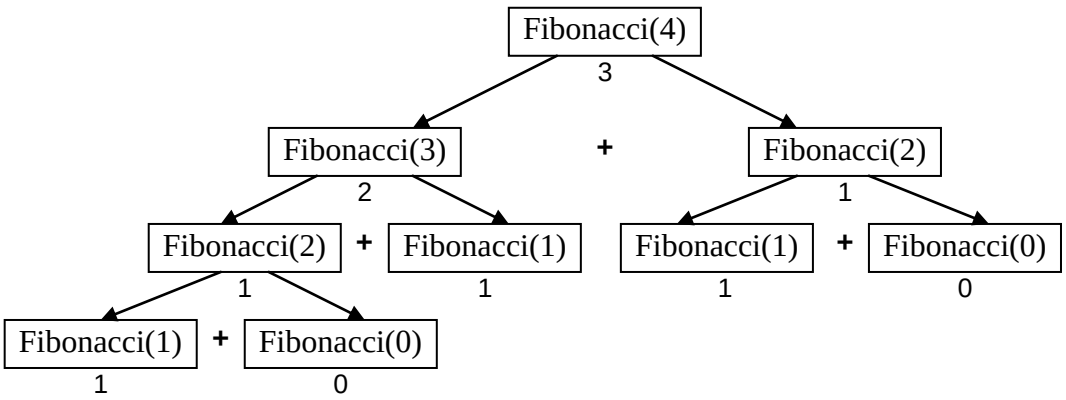
Program komputer dari perhitungan bilangan Fibonacci adalah :

```
#include <iostream.h>
long fibo(int n);
void main()
{
    int a;
    cout<<"Masukkan fibonacci = ";
    cin>>a;
    cout<<fibo(a);
    cout<<endl;
}

long fibo(int n)
{
    int f;
    if (n==0) //titik berhenti
        f=0;
    else
        if (n==1)
            f=1;  //titik berhenti
        else
            f=fibo(n-1)+fibo(n-2);  //rekursi
    return f;
}
```

Hasil eksekusi dari program tersebut adalah :

```
Masukkan faktorial = 4
3
Press any key to continue
```

Adapun proses kerja dari rekursi perhitungan bilangan Fibonacci adalah seperti ditunjukkan dalam gambar (7.2) :



Gambar 7.2 Proses rekursi perhitungan bilangan Fibonacci untuk n = 4

Latihan :

1.Apakah hasil eksekusi dari program berikut ini :

```
#include <iostream.h>
void bingo(int t);
void main()
{
    bingo(6);
}

void bingo(int t)
{
  if (t<2)
  {
       cout<<"bingo!";
       cout<<endl;
  }
  else
  {
       bingo(t-1);
       bingo(t-2);
  }
}
```

2.Apakah hasil eksekusi dari program berikut ini :

```
#include <iostream.h>
void ABC(char x[10],int n);
void main()
{
  ABC("LESTARI",6);
  cout<<endl;
}

void ABC(char x[10], int n)
{
  if (n>=0)
  {
          cout<<x[n];
          ABC(x,n-1);
  }
}
```

3.Andaikan kita mempunyai fungsi seperti berikut ini, apakah hasil eksekusinya jika fungsi tersebut dipanggil dengan : **abc(5,3)**

```
long abc(int a,int b)
{
          int x;
          if (b==0)
                    x=1;
          else
                    x=a*abc(a,b-1);
          return x;
}
```

4.Andaikan kita mempunyai fungsi seperti berikut ini, apakah hasil eksekusinya jika fungsi tersebut dipanggil dengan : **moo(100)**

```
void moo(int n)
```

```
{
        if (n>1)
        {
                moo(n / 5);
                cout<< n%5+1;
        }
}
```

5.Andaikan kita mempunyai fungsi seperti berikut ini, apakah hasil eksekusinya jika fungsi tersebut dipanggil dengan : **soo(100)**

```
void soo(int n)
{
        if (n>1)
        {
                n=n/3;
                cout<< n%3<<endl;
                soo(n);
        }
}
```

6.Andaikan kita mempunyai program seperti berikut ini, apakah hasil eksekusinya?

```
#include <iostream.h>
void call(int x);
void main()
{
    call(5);
}
void call(int x)
{
        if (x != 0)
        {
                cout<<"*";
                x--;
                call(x);
                x++
                cout<<x;
        }
}
```

7.Andaikan kita mempunyai fungsi seperti berikut ini, apakah hasil eksekusinya jika fungsi tersebut dipanggil dengan : **AFI(5)**

```
int AFI(int n)
{
        int j;
        if (n<3)
        {
                if (n==1)
                        j=1;
                else
                        if (n==2)
                                j=0;
        }
        else
                j=AFI(n-2)-AFI(n-1);
        return j;
}
```