PENGENALAN C++
Bab 1

1.1 SEJARAH C++

Bahasa C++ diciptakan oleh Bjarne Stroustrup di AT&T Bell Laboratories awal tahun 1980-an berdasarkan C ANSI (American National Standard Institute). Pertama kali, prototype C++ muncul sebagai C yang dipercanggih dengan fasilitas kelas. Bahasa tersebut disebut C dengan kelas (C wih class). Selama tahun 1983-1984, C dengan kelas disempurnakan dengan menambahkan fasilitas pembebanlebihan operator dan fungsi yang kemudian melahirkan apa yang disebut C++. Symbol ++ merupakan operator C untuk operasi penaikan, muncul untuk menunjukkan bahwa bahasa baru ini merupakan versi yang lebih canggih dari C.

Borland International merilis compiler Borland C++ dan Turbo C++. Kedua compiler ini sama-sama dapat digunakan untuk mengkompilasi kode C++. Bedanya, Borland C++ selain dapat digunakan dibawah lingkungan DOS, juga dapat digunakan untuk pemrograman Windows.

Selain Borland International, beberapa perusahaan lain juga merilis compiler C++, seperti Topspeed C++ dan Zortech C++.

Contoh Program C :

```
# include <stdio.h>
Main ( )
{
Char pesan [ ] = “Hai, C programmers !” ;
Printf (pesan) ;
Return 0 ;
}
```

Contoh Program C++ :

```
# include <iostream.h>
Main ( )
{
Char pesan [ ] = “Hai, C programmers !” ;
Cout << pesan ;
Return 0 ;
}
```

1.2 TENTANG C++

C++ diciptakan untuk mendukung pemrograman berorientasi pada objek (Object Oriented Pragramming/OOP) yang tidak dimiliki C. sementara C merupakan bahasa pemrograman terbaik dilingkungannya, bahasa ini tidak

memiliki kemampuan OOP. Reputasi C tidak diragukan lagi dalam menghasilkan program .EXE berukuran kecil, eksekusi yang cepat, antarmuka (interfacing) yang sederhana dengan bahasa lain dan fleksibilitas pemrograman. Apa yang membuat C tampak sukar dipelajari mungkin karena tiadanya pemeriksaan tipe. Sebagai contoh, dapat mencampur bilangan bulat dengan string untuk menghasilkan karakter. Namun, justru dsitu letak fleksibilitas C, dapat mengolah data C sebebas mengolah data dalam bahasa assembly.

1.3 BORLAND C++

Dibandingkan compiler C++ yang lain, Borland C++ memiliki keunggulan terutama dalam hal kecepatan dan efisiensi kompilasi. Disamping itu, Borland C++ mendukung beberapa system operasi yaitu DOS, Windows 16bit (Window 3.0) dan windows 32 bit (Windows NT). Meskipun demikian compiler Borland C++ juga memiliki kelemahan bila dibandingkan compiler C++ yang lain, misalnya : pemrograman dengan Borland C++ terutama yang menyangkut tampilan jauh lebih sulit daripada pemrograman dengan Microsoft Visual C++.

1.4 STRUKTUR BAHASA C++

Program C maupun C++ selalu tersusun dari 4 (empat) bagian utama, yaitu :

1. Bagian komentar yang ditandai dengan symbol // dan pasangan /* … */
2. Bagian pengarah compiler yang ditandai dengan symbol #
3. Bagian deklarasi
4. Bagian definisi

1.4.1 BAGIAN KOMENTAR

Program yang baik pada umumnya diberi komentar yang akan membantu orang lain maupun pembuat program itu untuk memahami program yang dibuat. Dalam C atau C++ setiap tulisan yang diapit oleh symbol /* … */ atau setiap baris yang dimulai dengan symbol // dianggap komentar. C++ tidak mengizinkan komentar bersarang (nested comment), namun Borland C++ lebih fleksibel dalam hal ini.

Contoh C++ menggunakan komentar:

```
/* FIRST.CPP */
/* Program C++ pertamaku */
// Praproses
# include <iostream.h>
Void main ( )
{
Char pesan [ ] = “Hello, C++ programmers!” ;
```

```
Cout << pesan ;
Return 0 ;
}
```

Pada Borland C++ dapat menggunakan komentar bersarang asalkan opsi cek **Nested comments** pada menu **Options/Compiler/Source** dipilih.

1.4.2 BAGIAN PENGARAH KOMPILER

Contoh program C++ :

```
# include <iostream.h>
Void main ( )
{
Char pesan [ ] = “Hello, C++ programmers!” ;
Cout << pesan ;
Return 0 ;
}
```

Merupakan statement praprosesor, disebut juga pengarah compiler karena berfungsi mengatur proses kompilasi.
IOSTREAM.H merupakan file program yang mengandung deklarasi kelas-kelas yang diperlukan oleh objek **cout**. File-file dengan ekstensi .H yang berisi deklarasi fungsi-fungsi standar C ini, disebut secara umum sebagai file header.
Beberapa pengarah compiler adalah :

1. # define
2. # include
3. # if, # else, # elif, # endif
4. # ifdef, # ifndef

1.4.2.1 PENGARAH KOMPILER # DEFINE

Untuk mendefinisikan suatu pengenal / konstanta yang nantinya akan digantikan oleh praprosesor saat program dikompilasi.

Contoh Program :

```
# define SIZE 30
Int array [SIZE] ;
For (register int i = 0 ; i < SIZE ; i++)
{
Cout << array [ i ] ;
}
```

1.4.2.2 PENGARAH KOMPILER # INCLUDE

Berfungsi membaca file program tertentu dan mengikutsertakan file tersebut dalam proses kompilasi. Nama file yang dimaksud harus diapit symbol ‘ < ‘ dan ‘ > ‘ atau tanda kutip dua (“ … “).

1.4.2.3 PENGARAH KOMPILER # IF, # ELSE, # ELIF, # ENDIF

Digunakan untuk memilih bagian program yang akan dikompilasi. Kompilasi cari ini disebut kompilasi bersyarat dan program yang baik biasanya memanfaatkan teknik ini.

1.4.2.4 PENGARAH KOMPILER # IFDEF, # IFNDEF

Digunakan juga dalam kompilasi bersyarat. # **Ifdef** dapat dibaca : ‘jika didefinisikan’ dan **# ifndef** dapat dibaca : ‘jika tidak didefinisikan’.
Pengarah compiler ini sering digunakan untuk menandai bahwa suatu file sudah diikutsertakan dalam kompilasi

1.4.3 BAGIAN DEKLARASI DAN DEFINISI

Semua program C pada dasarnya tersusun dari rangkaian pemanggilan fungsi yang bekerja atas sekelompok data. Selain pemanggilan fungsi, program C mengandung komponen lain yang disebut statement. Statement C ada dua, yaitu : statement yang tidak dapat dieksekusi / non executable (bila dikompilasi tidak menghasilkan kode objek dan biasanya digunakan untuk mengatur alur program), dan statement yang dapat dieksekusi / executable (bila dikompilasi akan menghasilkan kode objek). Setiap pemanggilan fungsi maupun statement executable dalam C harus diakhiri dengan tanda titik koma (;).

Contoh program C++:

```
# include <iostream.h>
Void main ( )
{
Char pesan [ ] = “Hello, C++ programmers!” ;
Cout << pesan ;
Return 0 ;
}
```

Dalam contoh program C++ diatas, **Return** merupakan contoh statement executable yang menginstruksikan agar suatu fungsi mengembalikan nilai balik tertentu. Contoh statement non executable adalah : **If, else**, dan **while.**

Main ( ) merupakan contoh fungsi, sedangkan **pesan** adalah contoh data. Baik data maupun fungsi harus dideklarasikan. Data perlu dideklarasikan agar compiler tahu berapa byte memori yang harus disediakan untuk data yang bersangkutan, sedangkan fungsi perlu dideklarasikan agar compiler dapat memeriksa ketepatan pemanggilan fungsi yang bersangkutan. Deklarasi fungsi sering disebut pula prototype fungsi.

1.5 INPUT DAN OUTPUT

Di ANSI C, operasi input dan output dilakukan dengan menggunakan fungsi-fungsi yang ada di header file **stdio.h**. contohnya untuk input dan output ke layer monitor digunakan perintah seperti **printf, scanf, putch**, dsb. Untuk input dan output ke file digunakan perintah seperti **fread, fwrite, fputc**, dsb.

C++ mempunyai teknik input dan output yang baru, yaitu : menggunakan **stream**. Header file untuk input dan output stream adalah **iostream.h** dan beberapa file lain, seperti **strstrea.h, fstream.h**, dan **constrea.h**.

Stream adalah suatu logika device (peralatan logika) yang menghasilkan dan menerima informasi atau suatu wadah yang digunakan untuk menampung keluaran dan menampung aliran data. Stream adalah nama umum untuk menampung aliran data (contoh : file, keyboard, mouse), maupun untuk keluaran (contoh : layer, printer).

Dalam C++ input berarti membaca dari stream dan output berarti menulis ke stream.

Bentuk Umum Output operator :

Cout << ekspresi ;

Bentuk umum Input operator :

Cin >> variable ;

Dalam C++, menggunakan escape sequences untuk merepresentasikan suatu karakter yang tidak terdapat dalam tradisional symbol. Beberapa diantaranya :

\ n : linefeed / baris baru
\ b : back space
\ “ : petik ganda

Contoh program versi ANSI C :

```
# include <stdio.h>
Void main ( )
{
Int x ;
Printf ( “Masukkan sebuah bilangan : \n” ) ;
Scanf ( “ %d ”, &x ) ;
Printf ( “Bilangan yang dimasukkan adalah %d\n “, x ) ;
}
```

Contoh program versi C++ :

```
# include <iostream.h>
Void main ( )
{
Int x ;
Cout << “Masukkan sebuah bilangan : ” << endl ;
Cin >> x ;
cout << “Bilangan yang dimasukkan adalah “ << x << endl ;
}
```

Contoh Program untuk input dan output :

```
# include <iostream.h>
Main ( )
{
Int a ;
Cout << “masukkan suatu bilangan :” ;
Cin >> a ;
Cout << “nilai tersebut ditambah 1 = ‘ << a+1 ;
Return 0 ;
}
```

VARIABEL, TIPE DATA DAN EKSPRESI
Bab 2

2.1 IDENTIFIER

Indentifier adalah nama yang diberikan untuk nama objek, nama fungsi, nama variable, dll (sifatnya 'case sensitive'). Identifier pada C++ terdiri dari :

1. huruf 'A' sampai 'Z'
2. huruf 'a' sampai 'z'
3. underscore (_)
4. bilangan antara '0' sampai '9'

Ketentuan dalam memberi nama identifier dalah C++ adalah :

1. karakter pertama harus huruf atau underscore
2. untuk compiler Borland, panjang maksimum 32 karakter
3. identifier harus tidak sama dengan keyword yang ada di C++

contoh identifier :

- Yang benar : nilai, Nilai_nama, No8
- Yang salah : 1Buah, nomor-data, if

2.2 TIPE DATA DI C++

Tipe data diklasifikasikan berdasarkan bagaimana keadaan data disimpan dalam memori, dan jenis operasi yang dapat dilakukan.

2.2.1 CHAR

Adalah sembarang huruf, angka, tanda baca tunggal. Ada 2 (dua) macam char, yaitu :

1. signed
   mendeklarasikan char bertanda, digunakan untuk nilai negative. Rentang nilai mulai -128 sampai 127
2. unsigned
   mendeklarasikan char tidak bertanda, untuk nilai positif. Rentang nilai mulai 0 sampai 255

contoh deklarasi char :

char letter = ‘A’ ;
unsigned char number = 245 ;
signed char value = -71 ;

2.2.2 SHORT, INT, LONG

Digunakan untuk menyatakan bilangan bulat. Seperti pada char, perubah tipe signed dan unsigned dapat ditambahkan.
Rentang nilai short int mulai -32.768 sampai 32.767
Rentang nilai long / int mulai -2.147.483.648 sampai 2.147.483.647

Contoh deklarasi int :

Int nilai, total ; *atau*
Int nilai = 90 ;

2.2.3 FLOAT, DOUBLE

Menyatakan bilangan pecahan/real, maupun eksponensial. Dalam keadaan default, bilang floting point dianggap bertipe double.

Rentang nilai float mulai 3,4 E $^{-38}$ sampai 3,4 E $^{+38}$
Rentang nilai double mulai 1,7 E $^{-308}$ sampai 1,7 E $^{+308}$

2.2.4 ENUMERATION / ENUM

Adalah serangkaian symbol berurutan yang menspesifikasikan konstanta bertipe integer. Dalam C++ tidak terdapat tipe Bolean, sehingga untuk merepresentasikan TRUE dengan nilai integer bukan nol (1, 2, dst), sedangkan FALSE dengan nilai nol (0)

Contoh deklarasi enum :

Enum BOOLEAN { False, True } ; *atau*
Enum BOOLEAN { Benar = 3, Salah = 0 } ;

2.2.5 VOID

Menyatakan tipe kosong untuk :

1. mendeklarasikan fungsi yang tidak mengembalikan nilai apapun.
2. mendeklarasikan fungsi yang tidak menerima parameter apapun.
3. bila diawali dengan operator *, menyatakan penunjuk terhadap sembarang tipe data.

Contoh deklarasi void :

Void cctrputs (char*, int) ; *atau*
Main (**void**) ; *atau*
Void* action ;
Int ivalue = 100 ;
Action = &ivalue ;

2.2.6 PENUNJUK / POINTER

Adalah variable yang berisi nilai alamat suatu lokasi memori tertentu. Deklarasi penunjuk dilakukan dengan menspesifikasikan *, sebelum nama varabel / konstanta.

2.2.7 PENUNJUK / POINTER

Adalah sekelompok data bertipe sama yang menduduki lokasi memori yang berurutan. Jumlah elemen array dinyatakan dengan cara mengapit jumlah yang di maksud dengan tanda ‘ […] ‘

Bentuk umum : tipe namaArray [jumlahelemen] ;

Untuk menyatakan array berdimensi lebih dari 1 (satu), tambahkan tanda ‘[…]’ sebanyak dimensi yang diinginkan.

Contoh deklarasi array 2 dimensi :

Int matrix [2][3] ;

2.2.8 STRING

Deretan karakter yang diakhiri dengan sebuah karakter kosong. String ditulis dengan mengapit string dengan tanda petik dua (“ …….”)

Contoh deklarasi string :

Char text [ ] = “ C++ “ ;
Puts (text) ;

2.2.9 STRUCT, UNION

Digunakan untuk mendeklarasikan sekelompok data yang memiliki tipe yang berlainan. **Struct** : elemennya ada dilokasi memori yang berbeda, dan **union** : elemennya ada dilokasi memori yang sama.

Bentuk umum :

```
Struct tipestruktur
{
        Tipeanggota1 namaAnggota1 ;
        Tipeanggota2 namaAnggota2 ;
        ……..
}
namaStruktur ;
```

2.3 DATA OBJEK

Data objek adalah bagian dari memori yang digunakan untuk menampung nilai dari variable. **Variable** umumnya digunakan untuk data objek yang nilainya dapat diubah selama pemrosesan berlangsung.

Contoh deklarasi variable :

Int **nilai** ; atau int **nilai** = 80 ;

Dalam C++ pendeklarasian termasuk statemen, sehingga pendeklarasian dapat diletakkan pada sembarang tempat dalam program.

Konstanta data objek adalah : variable yang nilainya tidak dapat diubah selama pemrosesan berlangsung.

Contoh deklarasi konstanta :

Const double pi = 3.14 ;

2.4 SCOPE IDENTIFIER

Ruang lingkup / scope adalah bagian mana dari program, identifier tersebut dapat diakses. Scope dari suatu identifier dimulai dari pendeklarasian sampai dengan akhir dari suatu blok. Scope identifier ada 2, yaitu :

1. local identifier
   dideklarasikan di dalam blok ‘ { … } ‘
2. global identifier
   dideklarasikan di luar dari blok

Scope relolution operator (::) dapat digunakan untuk mengakses variable global secara langsung.

Contoh variable global :

```
Int x ;
F ( )
{
        Int x ;
        :: x = 4 ;      // pemberian nilai untuk variable global x
}
```

2.5 OPERATOR DAN EKSPRESI

Ekspresi adalah rangkaian dari operator, operand, dan punctuator (;)
contoh :

```
3 + 4 * 1 ;
```

2.5.1 OPERATOR ARITMATIKA

Terdiri dari :

- penjumlahan (+)
- pengurangan (-)
- sisa bagi / hanya untuk tipe data integer (%)
- perkalian (*)
- pembagian (/)

Jika operator bagi (/) diterapkan pada tipe integer, akan menghasilkan bilangan integer dengan decimal yang dihilangkan.

2.5.2 ASSIGNMENT OPERATOR (=)

Berfungsi untuk memberi nilai pada variable.

Table kombinasi assignment :

NO.	PENYINGKATAN	ARTI
1	X += Y	X = X + Y
2	X -= Y	X = X – Y
3	X *= Y	X = X * Y
4	X /= Y	X = X / Y
5	X %= Y	X = X % Y

2.5.3 INCREMENT DAN DECREMENT OPERATOR

Increment adalah penambahan suatu variable dengan nilai 1, dan **decrement** adalah pengurangan suatu variabel dengan nilai 1.

Tabelnya :

INCREMENT	DECREMENT
++X ;	- -X ;
X += 1 ;	X -= 1 ;
X = X + 1	X = X – 1 ;

Operator increment dan decrement dapat diletakkan pada awal atau akhir variable, seperti dibawah ini :

++X , nilai variable X dinaikkan dahulu sebelum diproses
X++ , nilai variable X diproses dahulu sebelum dinaikkan

Contoh program increment :

```
# include <iostream.h>
Main ( )
{
Int X = 5 ;
Cout << " Nilai X = " << X << ' \n ' ;
Cout << " Nilai X++ = " << X++ << ' \n ' ;
Cout << " X = " << X << ' \n ' :
X = 5 ;
Cout << " Nilai X = " << X << ' \n ' ;
Cout << " Nilai ++X = " << ++X << ' \n ' ;
Cout << " X = " << X << ' \n ' ;
Return 0 ;
}
```

Outputnya :

```
Nilai X = 5
Nilai X++ = 5
X = 6
Nilai X = 5
Nilai ++X = 6
X = 6
```

2.5.4 EQUALITY, RELATIONAL, LOGIKA OPERATOR

Equality digunakan untuk menentukan apakah 2 buah variable memiliki nilai yang sama atau tidak.

= = , sama dengan, contoh : 5 **= =** 5
! = , tidak sama dengan , contoh : 5 **! =** 4

Relational operator digunakan untuk menentukan apakah suatu variable memiliki nilai lebih besar atau sama dengan lebih besar, lebih kecil atau lebih kecil sama dengan.

Operatornya : **<** , **< =** , **>** , **> =**

Logika operator adalah : (penulisan dibawah ini berdasarkan prioritas operator yang akan diproses terlebih dahulu)

1. **!** (not)
2. **&&** (and)
3. **||** (or)

2.6 EKSPRESI CONDITIONAL

Bentuk umumnya :

Ekspresi C **?** ekspresi T **:** ekspresi S ;

Keterangan :
- ekspresi C = kondisi yang akan diproses lebih dahulu
- ekspresi T = jika kondisi ekspresi C nilainya TRUE, akan dijalankan
- ekspresi S = jika kondisi ekspresi C nilainya FALSE, akan dijalankan

contoh program :

```
# include <iostream.h>
Main ( )
{
Double nilai ;
Cout << " Masukkan suatu nilai = ' ;
Cin >> nilai ;
Nilai = (nilai < 0) ? –nilai : nilai ;
Cout << "nilai absolutnya =" << nilai ;
Return 0 ;
}
```

2.7 FORMAT OUTPUT PADA BILANGAN REAL

Beberapa format yang dapat dilakukan :

1. derajat ketelitian, dengan fungsi : **precision**.
2. lebar output dapat diubah dengan fungsi : **width**.
3. format bilangan real diubah dengan fungsi : **setf** diikuti argument : **ios :: scientific** atau **ios :: fixed**
4. aligment (rata kiri/kanan) dengan fungsi : **setf** atau **unsetf** diikuti argument : **ios :: left** atau **ios :: right**
5. karakter pengisi dengan fungsi : **fill** diikuti argument karakter
6. tampilan tanda '+' diubah dengan fungsi : **setf** atau **unsetf** diikuti argument : **ios :: showpos**
7. tampilan tanda '.' (titik) bila ada angka dibelakang koma diubah dengan fungsi : **setf** atau **unsetf** diikuti argument **ios :: showpoint**

contoh program :

```
# include <iostream.h>
# pragma hdrstop
Void main ( )
{
Double y = 1234.56789 ;
Cout << "menuliskan bilangan real dengan presisi berbeda' ;
Cout << " \n Presisi 3 = " ;
Cout.precision (3) ;
Cout.width (15) ;
Cout << y ;
Cout << " \n Presisi 7 = " ;
Cout.precision (7) ;
Cout.width (15) ;
Cout << y ;
Cout << " \n \n Menggunakan notasi scientific / fixed " ;
Cout.setf (ios :: scientific | ios :: showpos) ;
Cout << " \n Scientific = " << y ;
Cout.setf (ios :: fixed) ;
Cout << " \n Fixed = ' << y ;
Cout.unsetf (ios :: scientific | ios :: fixed | ios :: showpos) ;
Cout << " \n \n Menggunakan shompoint " ;
Double z = 123 ;
Cout.setf (ios :: showpoint) ;
Cout << " \n Showpoint aktif = " << z ;
Cout.unsetf (ios :: showpoint) ;
Cout << "\n Showpoint non aktif : " << z ;
}
```

FUNCTION
Bab 3

3.1 PENGERTIAN FUNGSI DI C++

Function/fungsi adalah satu blok kode yang melakukan tugas tertentu atau satu blok instruksi yang di eksekusi ketika dipanggil dari bagian lain dalam suatu program.
Bentuk umum deklarasi fungsi :

Tipenilaibalik fungsi (tipepara, …) ;

Keterangan :

- **tipenilaibalik =**
  tipe nilai yang dikembalikan dengan statemen 'return'. Tipe default nya : 'int'. Untuk menyatakan fungsi yang tidak mengembalikan nilai balik, dideklarasikan sebagai : 'void'
- **fungsi =**
  nama fungsi tersebut
- **tipepara =**
  tipe parameter, bila parameter lebih dari satu (1), masing-masing dipisahkan dengan tanda koma (,)
  untuk menyatakan fungsi tanpa parameter dispesifikasikan : 'void'. Bila tipe parameter tidak dispesifikasikan, defaultnya : 'void'

Fungsi harus dideklarasikan terlebih dahulu sebelum didefinisikan. Maksudnya adalah memberitahu compiler jumlah dan tipe parameter yang diterima dan nilai balik fungsi (bila ada) agar compiler dapat memeriksa ketepatannya. **Definisi fungsi** itu sendiri adalah menspesifikasikan tugas fungsi tersebut.

Contoh 1 - program fungsi :

```
# include <iostream.h>
Int tambah (int a, int b )
{
Int r ;
r = a + b ;
return 0 ;
}
Int main ( )
{
Int z ;
z = tambah ( 5,3 ) ;
cout << "Hasilnya = ' << z ;
return 0 ;
}
```

Contoh 2 – program fungsi tanpa tipe :

```
# include <iostream.h>
Void contoh (void)
{
Cout << " ini adalah FUNGSI " ;
}
Int main ( )
{
Contoh ( ) ;
Return 0 ;
}
```

3.2 PARAMETER FUNGSI

Parameter fungsi ada 2, yaitu : parameter formal dan parameter actual.

1. Parameter Formal
   Parameter yang dideklarasikan dibagian blok fungsi.
2. Parameter Aktual
   Parameter yang sebenarnya / parameter yang dilewatkan saat fungsi dipanggil.

Contoh program yg menunjukkan parameter formal dan actual :

```
# include <iostream.h>
Int tambah (int a, int b)          // parameter formal
{
Int r ;
r = a + b ;
return 0 ;
}
Int main ( )
{
Int x, y, z ;
Cin >> x >> y ;
z = tambah ( x , y ) ;             //parameter aktual
cout << "Hasilnya = ' << z ;
return 0 ;
}
```

3.3 PARAMETER FUNGSI PASS BY VALUE

Parameter yang diberikan ke fungsi adalah 'Nilainya', tidak pernah menspesifikasikan variabelnya. Bila ada perubahan pada parameter formal, tidak akan mempengaruhi nilai pada parameter actual.

3.4 PARAMETER FUNGSI PASS BY REFERENCE

Memanipulasi nilai dari dalam fungsi. Setiap perubahan terhadap parameter formal akan memepengaruhi nilai pada parameter actual. Parameter formal diberi symbol '**&**'. Cara ini adalah cara efektif yang memungkinkan sebuah fungsi mengembalikan lebih dari satu nilai.

Contoh program :

```
# include <iostream.h>
Void perkalian (int& a, int& b, int& c)
{
a *= 2 ; b *= 2 ; c *= 2 ;
}
Int main ( )
{
Int x = 1, y = 3, z = 7 ;
Perkalian ( x, y, z ) ;
Cout << " X = " << x << " Y = " << y << " Z = " << z ;
Return 0 ;
}
```

Outputnya :

X = 2 Y = 6 Z = 14

Contoh program :

```
# include <iostream.h>
Void prevnext (int x, int& prev, int& next)
{
prev = x – 1 ;
Next = x + 1 ;
}
Int main ( )
{
Int x = 100 , y, z ;
Prevnext ( x, y, z ) ;
Cout << " Previous = " << y << " , Next = " << z ;
Return 0 ;
}
```

Outputnya :

Previuos = 99, Next = 101

Saat pendeklarasian fungsi dapat langsung diberikan nilai default untuk setiap parameter.

Contoh program :

```
# include <iostream.h>
Int pembagian (int a, int b = 2)
{
Int r ;
r = a / b ;
Return  ( r ) ;
}
Int main ( )
{
Cout << pembagian ( 12 ) ;
Cout << endl ;
Cour << pembagian ( 20, 4) ;
Return 0 ;
}
```

3.4 POLIMORPHISM

Polimorphisme adalah objek-objek yang berbeda tetapi berasal dari satu orang tua, dapat mempunyai metode yang sama tetapi cara pelaksanaannya berbeda-beda. Contohnya adalah objek kendaraan yang terdiri dari objek kendaraan bermotor dan kendaraan tidak bermotor.

OVERLOADING FUNCTION
Bab 4

4.1 PENGERTIAN

Overloading function dalah beberapa fungsi dapat memiliki argument berbeda tetapi namanya sama.

Contoh program :

```
# include <iostream.h>
Void tulis (char c)
{
Cout << “karakter : “ << c << endl ;
}
Void tulis (long x)
{
Cout << “bilangan bulat =” << x << endl ;
}
Void main ( )
{
Tulis ( ‘A’ ) ;
Tulis (1805) ;
}
```

Keterangan :

- walaupun 2 fungsi diatas mempunyai nama yang sama, tetapi compiler tahu fungsi manakah yang harus dipanggil.

C++ hanya dapat mengkompilasi overloading function jika argument fungsinya berbeda. Bila argument fungsinya sama tetapi tipe nilai kembaliannya berbeda, compiler akan melaporkan adanya kesalahan.

Contoh :

Fungsi : **double coba (int) ;** *dengan* Fungsi : **int coba (int) ;**

4.2 DEFAULT ARGUMEN

Contoh :

Void func (int x = 0, int y = 0) ;

Maksudnya adalah :

Contoh fungsi diatas dapat dipanggil dengan 3 (tiga) cara berbeda :
1. func ( ) *// nilai x dan y otomatis 0*
2. func (7) *// nilai x = 7 dan y = 0*
3. func (7, 33) *// nilai x = 7 dan y = 33*

Contoh deklarasi fungsi diatas akan terjadi kesalahan, jika variable x diberi nilai, sedangkan variable y tidak beri nilai.

Contoh :

Void func (**int x = 2, int y**) ;

4.3 AMBIGUITY PADA OVERLOADING FUNCTION

Beberapa hal yang menimbulkan ketidakjelasan pada program :
1. type conversion
Contoh :

```
Void f ( double nilai )
Main ( )
{
Int I = 12 ;
F ( I ) ;
}
```

2. reference parameter
Parameter yang memiliki perbedaan hanya pada argument, dimana fungsi yang satu menggunakan reference parameter dan yang lainnya menggunakan value parameter.
Contoh :

Void func **(int x , int y);** *dengan* void func func **(int x , int& y) ;**

3. default argumen

4.4 TATA CARA PENULISAN PROGRAM

1. Jangan menggunakan nama identifier yang terlalu singkat.
2. Identifier yang tidak melambangkan sesuatu dapat ditulis dengan satu huruf saja.
3. hindari pendeklarasian variable yang bertipe sama dalam satu baris, contoh : **int a, b, c** ;
4. Tambahkan satu spasi setelah tanda koma atau titik dua atau titik koma, jangan sebelumnya.

5. Jangan tambahkan spasi sesudah operator unary : ++ , - - , dsb.
   Contoh : **a ++**
6. Jangan tambahkan spasi setelah tanda kurung buka dan kurung tutup.
7. Jangan gunakan spasi dipenulisan index, contoh : **x[10]** *atau* **x [10]**
8. Hindari penulisan secara horizontal.
   Contoh :

```
Void main ( )
{
Tulis ( 'A' ) ; Tulis (1805) ;
}
```

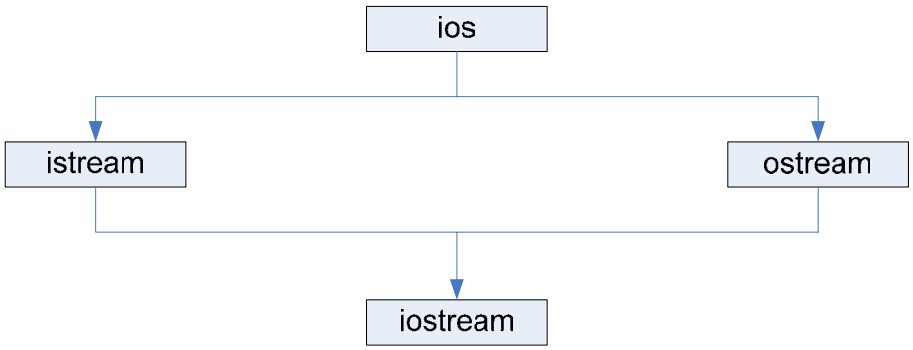
FILE & STREAM
Bab 5

5.1 FILE

File adalah sekumpulan data yang disimpan dalam media penyimpanan luar seperti disket/harddisk. Dalam C++ file adalah sebuah stream yang disimpan dalam media penyimpanan luar. Karena merupakan sebuah stream, operasi yang berlaku pada stream berlaku juga untuk file.

Stream adalah suatu logika device yang menghasilkan dan menerima informasi atau wadah yang digunakan untuk menampung keluaran dan menampung aliran data.

Hirarki I/O class :



Penjelasan :

- ios adalah virtual base class untuk class istream dan ostream. Berisi fasilitas dasar untuk melakukan proses input/output. Dideklarasikan untuk pointer ke buffer untuk tempat penyimpanan data sementara.
- Istream (input stream) mendefinisikan fasilitas untuk melakukan input suatu informasi. Di dalamnya didefinisikan fungsi get ( ), getline ( ), extractor operator >>
- Ostream (output stream) mendefinisikan fasilitas untuk melakukan setting terhadap output.
- Iostream : berisi semua fasilitas dari ios, istream, ostream ditambah beberapa fungsi untuk menyempurnakan kerja dari fungsi yang dideklarasikan pada base class.

Untuk melakukan proses file I/O, diperlukan file header fstream.h didalam program. Dalam file fstream.h didefinisikan beberapa class/object yang berhubungan dengan pemrosesan file, yaitu : ifstream, ofstream, fstream yang diturunkan dari istream dan ostream.

5.2 KETERHUBUNGAN STREAM DENGAN FILE

Jika mendeklarasikan suatu stream, dapat menghubungkan stream tersebut dengan file, dimana proses ini berhubungan dengan operasi terhadap file.

Tiga proses utama dalam mengelola file adalah :

1. membuka file
2. melakukan proses terhadap file
3. menutup file

Sebelum membuka file, harus mengetahui keadaan mode filenya. Keadaan yang perlu diketahui adalah :

1. untuk membuka file dengan tujuan output, digunakan ofstream.
2. untuk membuka file dengan tujuan input, digunakan ifstream.
3. untuk membuka file dalam keadaan input maupun output, gunakan fstream.

Fungsi open ( ) digunakan untuk membuka file.
Bentuk umumnya adalah :

Void open (char* file_name, int mode, int access) ;

Keterangan :

- dengan fungsi open ( ), menhubungkan stream dengan file yang bernama file_name.
- nilai dari var mode akan menentukan bagaimana keadaan file jika dibuka.
- Variable access akan menentukan bagaimana metode pengaksesan terhadap file tersebut. Nilai ini berhubungan dengan atribut file yang ada pada DOS.

Nilai pada variable mode adalah :

1. ios : : app
   semua informasi yang ditulis ke dalam file (output) akan ditambahkan dibagian akhir file tersebut.
2. ios : : ate
   file akan dibuka dengan pointer file menunjuk pada akhir file.
3. ios : : in
   file akan dibuka sebagai input.
4. ios : : out
   file akan dibuka sebagai output
5. ios : : nocreate
   akan mengakibatkan kesalahan jika file tidak ada.
6. ios : : noreplace
   file yang dibuka tidak dapat diganti, atau mengakibatkan kesalahan jika file yang akan dibuka sudah ada.
7. ios : : trunc
   menyebabkan isi dari file yang sudah ada akan hilang

Nilai pada variable access adalah :
0 : normal file 4 : system file
1 : read only file 8 : archive bit-set file
2 : hidden file

Contoh penggunaan open ( ) :

Akan dibuka sebuah file sebagai output, dimana atribut dari file tersbut adalah read only dan nama file tersebut adalah tes.
Jawab :

```
Ofstream fout ;
Fout.open ("tes", ios : : out, 1) ;
```

Jika dalam pembukaan file terdapat kesalahan, maka fungsi open ( ) akan mengembalikan nilai NULL.

Fungsi close ( ) untuk menutup file yang telah dibuka.
Bentuk umumnya :

```
Void close ( ) ;
```

Contoh penggunaan close ( ) :

```
Fout.close ( ) ;
Fin.close ( ) ;
Fio. Close ( ) ;
```

Didalam class ios terdapat pendefinisian fungsi : int eof ( ) digunakan untuk menentukan apakah pointer pada file telah mencapai akhir darti file tersebut.

5.3 DETEKSI ERROR

Dalam class ios ada 4 (empat) buah fungsi untuk melakukan tes terhadap error yang terjadi didalam stream. Fungsi tersebut adalah :

1. int good ( )
2. int eof ( )
3. int bad ( )
4. int fail ( )

Keempat fungsi itu akan menghasilkan nilai 0 (nol) jika kondisi tersebut salah dan nilai bukan nol jika kondisi tersebut benar.

Jika error terjadi, maka stream harus dibersihkan dahulu dari kesalahan sebelum melanjutkan ke proses berikutnya. Fungsi yang digunakan untuk membersihkan kesalahan pada stream jika terjadi kesalahan : clear ( )

Bentuk umumnya :

Void clear (int flags = 0) ;

Fungsi clear ( ) dalam keadaan default akan membersihkan seluruh flag. Dapat ditentukan flag mana yang akan dibersihkan, dengan memberi argument yang sesuai.

Fungsi int rdstate ( ) digunakan untuk menentukan jenis kesalahan yang terjadi, dengan mengembalikan nilai integer kesalahan tersebut.
Nilai enum yang dikembalikan oleh fungsi rdstate ( ) adalah :

1. goodbit
   0 jika tidak ada error dan 1 jika terjadi error
2. eofbit
   1 jika eof ditemukan dan 0 jika eof tidak ditemukan
3. failbit
   1 jika non fatal error ditemukan dan 0 jika tidak
4. badbit
   1 jika fatal error ditemukan dan 0 jika tidak.

5.4 INPUT/OUTPUT KARAKTER

Input/output pada file binary digunakan fungsi get ( ), put ( ).

Bentuk Umumnya :

Istream &get (char& ch) ;
Ostream &put (char ch) ;

Fungsi get ( ) akan membaca sebuah karakter dari stream dan karakter tersebut akan diletakkan pada variable ch.
Fungsi ini akan mengembalikan reference ke stream jika EOF ditemukan, maka nilai yang dikembalikan NULL.

Fungsi put ( ) akan meletakkan sebuah karakter ke sutau stream dan akan mengembalikan reference ke stream.

Untuk membaca dan menulis file binary dalam bentuk blok data dapat menggunakan fungsi read ( ) dan write 9)

Bentuk Umum :

Istream& read (unsigned char* buf, int num) ;
Ostream& write (const unsigned char* buf, int num);

Fungsi read ( ) akan membaca sejumlah num byte dari stream dan akan meletakkan data tersebut ke dalam variable yang ditunjukkan oleh pointer buf.

Fungsi write ( ) akan menulis sejumlah num byte ke stream dari data yang ditunjuk oleh pointer buf.

Fungsi yang akan mengembalikan karakter yang terakhir dibaca ke dalam memori lagi adalah : putback ( )

Bentuk umum :

Istream& putback (char ch) ;

Keterangan :

- variable ch adalah karakter yang dibaca yang akan dikembalikan lagi ke stream (didalam memori)

STRUKTUR KONDISI DAN STRUKTUR PERULANGAN
BAB 6

6.1 PERCABANGAN / KONDISI

Percabangan adalah suatu proses pemilihan aksi diantara beberapa alternative yang diberikan.

Bentuk umum statemen if :

```
If ( cond-exp) statement ;
```

Bentuk umum statement if … else :

```
If ( cond-exp) statement true
Else statement false ;
```

Jika ada lebih dari 1 (satu) instruksi yang akan dijalankan maka harus dibuat dalam bentuk blok instruksi dengan menggunakan tanda kurung kurawal { … }

Contoh program :

```
If ( nilai > 60 )
        Cout << "Anda lulus \n" ;
Else
        Cout << "Anda tidak lulus \n" ;
```

Untuk membentuk multiway selection, dapat digunakan nested if-else, hal ini dapat dilakukan sebanyak yang diinginkan.

Bentuk umum nested if … else :

```
If ( cond-exp1 )
        Statement1 ;
Else if ( cond-exp2 )
        Statement2 ;
…
…
Else
        Statement n ;
```

Contoh program :

```
If ( x > 0 )
        Cout << " x adalah positif " ;
Else if ( x < 0 )
        Cout << " x adalah negative " ;
Else
        Cout << " x adalah nol " ;
```

6.2 PERULANGAN / LOOP

Ada 3 (tiga) cara untuk melakukan perulangan di C++ yaitu :

1. for statement
2. while statement
3. do while statement

Loop adalah : perulangan statement dengan jumlah tertentu jika kondisi terpenuhi.

6.2.1 FOR STATEMENT

Bentuk umum :

For (<init-exp> ; <test-exp> ; <inc/dec-exp>)
    Statement ;

Keterangan :

- init-exp : ekspresi yang digunakan untuk melakukan inisialisasi terhadap variable-variabel tertentu, terutama variable yang digunakan untuk melakukan iterasi. Init-exp dapat berupa ekspresi maupun pendefinisian variable.
- Test-exp : ekspresi yang memegang control terhadap proses perulangan tersebut, pada bagian ini akan ditentukan apakah proses perulangan akan tetap berlangsung atau tidak.
- Inc/dec-exp : digunakan untuk mengatur perubahan nilai variable. Umumnya nilai variable tersebut bertambah / berkurang 1 (satu)

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int batas ;
Cout << " Nilai tertinggi = " ;
Cin >> batas ;
For ( int i = 1 ; i < = batas ; i++ )
Cout << i << ' ' ;
Return 0 ;
}
```

6.2.2 WHILE STATEMENT

Bentuk umum :

While (cond-exp) statement ;

Statement akan dilaksanakan terus selama cond-exp bernilai true.

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int I = 0 ;
While ( I < 10 )
{
        Cout << " Belajar C++ \n " ;
        I++ ;
}
Return 0 ;
}
```

6.2.3 DO WHILE STATEMENT

Bentuk umum :

Do statement while (cond-exp) ;

Perbedaan dengan while statement adalah : pada do while kondisi akan dievaluasi setelah dilakukan statement/proses. Selain itu pada do while minimal akan dilaksanakan 1 (satu) kali statement/instruksinya.

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int I = 0 ;
Do
{
        Cout << " Belajar C++ \n " ;
        I++ ;
}
While ( I < 10 ) ;
Return 0 ;
}
```

C/C++ Keywords

asm	insert an assembly instruction
auto	declare a local variable
bool	declare a boolean variable
break	break out of a loop
case	a block of code in a switch statement
catch	handles exceptions from throw
char	declare a character variable
class	declare a class
const	declare immutable data or functions that do not change data
const_cast	cast from const variables
continue	bypass iterations of a loop
default	default handler in a case statement
delete	make memory available
do	looping construct
double	declare a double precision floating-point variable
dynamic_cast	perform runtime casts
else	alternate case for an if statement
enum	create enumeration types
explicit	only use constructors when they exactly match
export	allows template definitions to be separated from their declarations
extern	tell the compiler about variables defined elsewhere
false	the boolean value of false
float	declare a floating-point variable
for	looping construct
friend	grant non-member function access to private data
goto	jump to a different part of the program
if	execute code based off of the result of a test
inline	optimize calls to short functions
int	declare a integer variable
long	declare a long integer variable
mutable	override a const variable
namespace	partition the global namespace by defining a scope
new	allocate dynamic memory for a new variable
operator	create overloaded operator functions
private	declare private members of a class
protected	declare protected members of a class
public	declare public members of a class

register	request that a variable be optimized for speed
reinterpret_cast	change the type of a variable
return	return from a function
short	declare a short integer variable
signed	modify variable type declarations
sizeof	return the size of a variable or type
static	create permanent storage for a variable
static_cast	perform a nonpolymorphic cast
struct	define a new structure
switch	execute code based off of different possible values for a variable
template	create generic functions
this	a pointer to the current object
throw	throws an exception
true	the boolean value of true
try	execute code that can throw an exception
typedef	create a new type name from an existing type
typeid	describes an object
typename	declare a class or undefined type
union	a structure that assigns multiple variables to the same memory location
unsigned	declare an unsigned integer variable
using	import complete or partial namespaces into the current scope
virtual	create a function that can be overridden by a derived class
void	declare functions or data with no associated data type
volatile	warn the compiler about variables that can be modified unexpectedly
wchar_t	declare a wide-character variable
while	looping construct

ARRAY DAN STRING
BAB 8

8.1 ARRAY

Adalah sekelompok data yang sejenis yang disimpan didalam memori secara berurutan dengan sebuah nama variable, dan untuk membedakan antara 1 data dengan data yang lainnya digunakan index.

Contoh deklarasi array :

Int arr [5] ; atau double d [10] ;

8.1.1 INISIALISASI ARRAY

Untuk menginisialisasi array, elemen-elemen array diletakkan diantara tanda kurung.

Contoh :

Int arr [5] = { 1, 3, -3, 5, 2 } ;

Jika jumlah elemen yang diinisialisasikan kurang dari jumlah elemen yang tersedia, maka sisa elemen tersebut akan diberikan nilai 0 (nol) secara otomatis oleh compiler.

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int arr [ 5 ] = { 1, 3 } ;
For ( int i = 0 ; i < 5 ; i++ )
Cout << " arr [ i ] << ' ' ;
Return 0 ;
}
```

Output : 1 3 0 0 0

8.2 STRING

String adalah kumpulan beberapa karakter atau array dari karakter. String dan karakter dibedakan cara penulisannya. String ditulis dengan diapit oleh tanda petik ganda " … " , dan karakter ditulis dengan diapit oleh tanda petik tunggal ' … '

Akhir dari string ditunjukkan oleh NULL karakter. Semua fungsi yang digunakan untuk melakukan manipulasi terhadap string menganggap string diakhiri dengan null. Karakter null direpresentasikan dengan tanda ' \0 '.

Contoh :

Cout << " abcdefg\0hijklmn" ;

Pada contoh diatas, akan tercetak di monitor adalah : abcdefg, karena ada tanda '\0' setelah huruf g dan itu menandakan akhir dari string.

Untuk mendeklarasikan string, maka array yang ada harus dilebihkan 1(satu) untuk tempat menampung karakter null.

Contoh deklarasi string :

Akan dideklarasikan array str untuk menampung string sepanjang 6 (enam) karakter, maka :

Char str [7] = " string " ; atau
Char str [7] = {'s', 't', 'r', 'i', 'n', 'g', '\0' } ;

8.2.1 MEMBACA STRING DARI KEYBOARD

Contoh program :

```
# include <iostream.h>
# include <conio.h>
Void main ( )
{
        Char nama [ 20 ] ;
        Char alamat [ 30 ] ;
        Cout << "Masukkan nama Anda :" ;
        Cin.getline (nama, sizeof (nama)) ;
        Cout << "Masukkan alamat Anda :" ;
        Cin.getline (alamat, sizeof(alamat)) ;
        Cout << "Nama Anda :" << nama << endl ;
        Cout << "Alamat Anda :" << alamat << endl ;
}
```

8.2.2 MENGCOPY STRING

Untuk memberikan nilai string dari suatu variable ke variable yang lain tidak dapat dilakukan hanya dengan perintah : kata2 = kata1. Untuk memberikan nilai ke variable lain dalam C++ digunakan perintah :

Strcpy (kata2, kata1) ;

Maksudnya adalah akan dicopi isi dari kata1 ke kata2

Contoh program copi string :

```
# include <iostream.h>
# include < conio.h>
# include < string.h>
Void main ( )
{
        Char kata1 [ 6 ] = "Hallo" ;
```

```
        Char kata2 [ 6 ] ;
        Strcpy (kata2, kata1) ;
        Cout << “Kata 1 adalah” << kata1 << endl ;
        Cout << “Kata 2 adalah” << kata2 << endl ;
}
```

8.2.3 FUNGSI UNTUK OPERASI STRING

Sebelum fungsi digunakan, tambahkan file header : ‘string.h’ pada # include.

8.2.3.1 PANJANG STRING

Sintaks : strlen (string) ;

Contoh program :

```
Int panjangteks ;
Char kalimat [ 27 ] = “Belajar C++ tidaklah sulit” ;
panjangteks = strlen (kalimat) ;
Cout << “Panjang string adalah :’ << panjangteks ;
```

8.2.3.2 MENGGABUNGKAN STRING

Sintaks : strcat (string1, string2)

Maksudnya : akan menambahkan string2 ke dalam string1.

Contoh program :

```
Char kata1 [ 5 ] = “Satu” ;
Char kata2 [ 5 ] = “Dua” ;
Strcat (kata1, kata2) ;
```

Hasil dari potongan program diatasa adalah : Satu Dua

8.2.3.3 KONVERSI KE HURUF KAPITAL

Sintaks : strupr (string) ;

Maksudnya adalah : akan mengubah huruf kecil ke huruf besar/capital

Contoh program :

```
Char string1 [ 30 ] = “aBcDefgHIJKLmn” ;
Strupr (string1) ;
```

Hasil dari potongan program diatas, nilai string1 akan menjadi : ABCDEFGHIJKLMN

8.2.3.4 KONVERSI KE HURUF KECIL

Sintaks : strlwr (string);

Fungsi ini adalah kebalikan dari fungsi strupr yaitu akan merubah huruf capital menjadi huruf kecil.

8.2.3.5 MENCARI SUBSTRING

Sintaks : strstr (string1, string2) ;

Fungsi ini akan mereturn nilai 1 jika nilai string2 merupakan substring dari string1 dan akan mereturn nilai 0(nol) jika string2 bukan substring dari string1.

Contoh program :

Diberikan suatu string “Jakarta Kota Metropolitan”. Apakah string “Metro” terdapat dalam kalimat tersebut ?

Potongan program untuk menjawabnya adalah :

```
If (strstr(“Jakarta Kota Metropolitan”, “Metro”) = = 1 )
        Cout << “Merupakan substring” ;
Else
        Cout << “Bukan substring” ;
```

8.2.3.6 MEMBALIK STRING

Sintaks : strrev (string);

Contoh program :

```
Char kata [ 10 ] = “C++” ;
Strrev (kata) ;
Cout << kata ;
```

Hasil dari potongan program diatas adalah : ++C

ARRAY DAN STRING
BAB 8

8.1 ARRAY

Adalah sekelompok data yang sejenis yang disimpan didalam memori secara berurutan dengan sebuah nama variable, dan untuk membedakan antara 1 data dengan data yang lainnya digunakan index.

Contoh deklarasi array :

Int arr [5] ; atau double d [10] ;

8.1.1 INISIALISASI ARRAY

Untuk menginisialisasi array, elemen-elemen array diletakkan diantara tanda kurung.

Contoh :

Int arr [5] = { 1, 3, -3, 5, 2 } ;

Jika jumlah elemen yang diinisialisasikan kurang dari jumlah elemen yang tersedia, maka sisa elemen tersebut akan diberikan nilai 0 (nol) secara otomatis oleh compiler.

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int arr [ 5 ] = { 1, 3 } ;
For ( int i = 0 ; i < 5 ; i++ )
Cout << " arr [ i ] << ' ' ;
Return 0 ;
}
```

Output : 1 3 0 0 0

8.2 STRING

String adalah kumpulan beberapa karakter atau array dari karakter. String dan karakter dibedakan cara penulisannya. String ditulis dengan diapit oleh tanda petik ganda " … " , dan karakter ditulis dengan diapit oleh tanda petik tunggal ' … '

Akhir dari string ditunjukkan oleh NULL karakter. Semua fungsi yang digunakan untuk melakukan manipulasi terhadap string menganggap string diakhiri dengan null. Karakter null direpresentasikan dengan tanda ' \0 '.

Contoh :

Cout << " abcdefg\0hijklmn" ;

Pada contoh diatas, akan tercetak di monitor adalah : abcdefg, karena ada tanda '\0' setelah huruf g dan itu menandakan akhir dari string.

Untuk mendeklarasikan string, maka array yang ada harus dilebihkan 1(satu) untuk tempat menampung karakter null.

Contoh deklarasi string :

Akan dideklarasikan array str untuk menampung string sepanjang 6 (enam) karakter, maka :

Char str [7] = " string " ; atau
Char str [7] = {'s', 't', 'r', 'i', 'n', 'g', '\0' } ;

8.2.1 MEMBACA STRING DARI KEYBOARD

Contoh program :

```
# include <iostream.h>
# include <conio.h>
Void main ( )
{
        Char nama [ 20 ] ;
        Char alamat [ 30 ] ;
        Cout << "Masukkan nama Anda :" ;
        Cin.getline (nama, sizeof (nama)) ;
        Cout << "Masukkan alamat Anda :" ;
        Cin.getline (alamat, sizeof(alamat)) ;
        Cout << "Nama Anda :" << nama << endl ;
        Cout << "Alamat Anda :" << alamat << endl ;
}
```

8.2.2 MENGCOPY STRING

Untuk memberikan nilai string dari suatu variable ke variable yang lain tidak dapat dilakukan hanya dengan perintah : kata2 = kata1. Untuk memberikan nilai ke variable lain dalam C++ digunakan perintah :

Strcpy (kata2, kata1) ;

Maksudnya adalah akan dicopi isi dari kata1 ke kata2

Contoh program copi string :

```
# include <iostream.h>
# include < conio.h>
# include < string.h>
Void main ( )
{
        Char kata1 [ 6 ] = "Hallo" ;
```

```
        Char kata2 [ 6 ] ;
        Strcpy (kata2, kata1) ;
        Cout << “Kata 1 adalah” << kata1 << endl ;
        Cout << “Kata 2 adalah” << kata2 << endl ;
}
```

8.2.3 FUNGSI UNTUK OPERASI STRING

Sebelum fungsi digunakan, tambahkan file header : ‘string.h’ pada # include.

8.2.3.1 PANJANG STRING

Sintaks : strlen (string) ;

Contoh program :

```
Int panjangteks ;
Char kalimat [ 27 ] = “Belajar C++ tidaklah sulit” ;
panjangteks = strlen (kalimat) ;
Cout << “Panjang string adalah :’ << panjangteks ;
```

8.2.3.2 MENGGABUNGKAN STRING

Sintaks : strcat (string1, string2)

Maksudnya : akan menambahkan string2 ke dalam string1.

Contoh program :

```
Char kata1 [ 5 ] = “Satu” ;
Char kata2 [ 5 ] = “Dua” ;
Strcat (kata1, kata2) ;
```

Hasil dari potongan program diatasa adalah : Satu Dua

8.2.3.3 KONVERSI KE HURUF KAPITAL

Sintaks : strupr (string) ;

Maksudnya adalah : akan mengubah huruf kecil ke huruf besar/capital

Contoh program :

```
Char string1 [ 30 ] = “aBcDefgHIJKLmn” ;
Strupr (string1) ;
```

Hasil dari potongan program diatas, nilai string1 akan menjadi : ABCDEFGHIJKLMN

8.2.3.4 KONVERSI KE HURUF KECIL

Sintaks : strlwr (string);

Fungsi ini adalah kebalikan dari fungsi strupr yaitu akan merubah huruf capital menjadi huruf kecil.

8.2.3.5 MENCARI SUBSTRING

Sintaks : strstr (string1, string2) ;

Fungsi ini akan mereturn nilai 1 jika nilai string2 merupakan substring dari string1 dan akan mereturn nilai 0(nol) jika string2 bukan substring dari string1.

Contoh program :

Diberikan suatu string “Jakarta Kota Metropolitan”. Apakah string “Metro” terdapat dalam kalimat tersebut ?

Potongan program untuk menjawabnya adalah :

```
If (strstr(“Jakarta Kota Metropolitan”, “Metro”) = = 1 )
        Cout << “Merupakan substring” ;
Else
        Cout << “Bukan substring” ;
```

8.2.3.6 MEMBALIK STRING

Sintaks : strrev (string);

Contoh program :

```
Char kata [ 10 ] = “C++” ;
Strrev (kata) ;
Cout << kata ;
```

Hasil dari potongan program diatas adalah : ++C

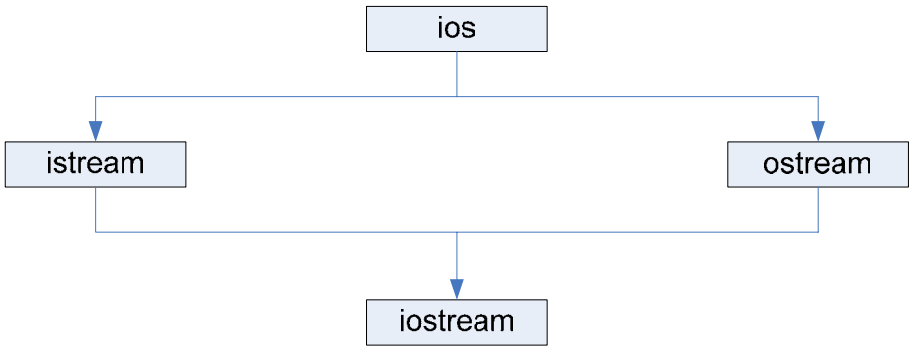
FILE & STREAM
Bab 5

5.1 FILE

File adalah sekumpulan data yang disimpan dalam media penyimpanan luar seperti disket/harddisk. Dalam C++ file adalah sebuah stream yang disimpan dalam media penyimpanan luar. Karena merupakan sebuah stream, operasi yang berlaku pada stream berlaku juga untuk file.

Stream adalah suatu logika device yang menghasilkan dan menerima informasi atau wadah yang digunakan untuk menampung keluaran dan menampung aliran data.

Hirarki I/O class :



Penjelasan :

- ios adalah virtual base class untuk class istream dan ostream. Berisi fasilitas dasar untuk melakukan proses input/output. Dideklarasikan untuk pointer ke buffer untuk tempat penyimpanan data sementara.
- Istream (input stream) mendefinisikan fasilitas untuk melakukan input suatu informasi. Di dalamnya didefinisikan fungsi get ( ), getline ( ), extractor operator >>
- Ostream (output stream) mendefinisikan fasilitas untuk melakukan setting terhadap output.
- Iostream : berisi semua fasilitas dari ios, istream, ostream ditambah beberapa fungsi untuk menyempurnakan kerja dari fungsi yang dideklarasikan pada base class.

Untuk melakukan proses file I/O, diperlukan file header fstream.h didalam program. Dalam file fstream.h didefinisikan beberapa class/object yang berhubungan dengan pemrosesan file, yaitu : ifstream, ofstream, fstream yang diturunkan dari istream dan ostream.

5.2 KETERHUBUNGAN STREAM DENGAN FILE

Jika mendeklarasikan suatu stream, dapat menghubungkan stream tersebut dengan file, dimana proses ini berhubungan dengan operasi terhadap file.

Tiga proses utama dalam mengelola file adalah :

1. membuka file
2. melakukan proses terhadap file
3. menutup file

Sebelum membuka file, harus mengetahui keadaan mode filenya. Keadaan yang perlu diketahui adalah :

1. untuk membuka file dengan tujuan output, digunakan ofstream.
2. untuk membuka file dengan tujuan input, digunakan ifstream.
3. untuk membuka file dalam keadaan input maupun output, gunakan fstream.

Fungsi open ( ) digunakan untuk membuka file.
Bentuk umumnya adalah :

Void open (char* file_name, int mode, int access) ;

Keterangan :

- dengan fungsi open ( ), menhubungkan stream dengan file yang bernama file_name.
- nilai dari var mode akan menentukan bagaimana keadaan file jika dibuka.
- Variable access akan menentukan bagaimana metode pengaksesan terhadap file tersebut. Nilai ini berhubungan dengan atribut file yang ada pada DOS.

Nilai pada variable mode adalah :

1. ios : : app
   semua informasi yang ditulis ke dalam file (output) akan ditambahkan dibagian akhir file tersebut.
2. ios : : ate
   file akan dibuka dengan pointer file menunjuk pada akhir file.
3. ios : : in
   file akan dibuka sebagai input.
4. ios : : out
   file akan dibuka sebagai output
5. ios : : nocreate
   akan mengakibatkan kesalahan jika file tidak ada.
6. ios : : noreplace
   file yang dibuka tidak dapat diganti, atau mengakibatkan kesalahan jika file yang akan dibuka sudah ada.
7. ios : : trunc
   menyebabkan isi dari file yang sudah ada akan hilang

Nilai pada variable access adalah :
0 : normal file 4 : system file
1 : read only file 8 : archive bit-set file
2 : hidden file

Contoh penggunaan open ( ) :

Akan dibuka sebuah file sebagai output, dimana atribut dari file tersbut adalah read only dan nama file tersebut adalah tes.
Jawab :

```
Ofstream fout ;
Fout.open ("tes", ios : : out, 1) ;
```

Jika dalam pembukaan file terdapat kesalahan, maka fungsi open ( ) akan mengembalikan nilai NULL.

Fungsi close ( ) untuk menutup file yang telah dibuka.
Bentuk umumnya :

```
Void close ( ) ;
```

Contoh penggunaan close ( ) :

```
Fout.close ( ) ;
Fin.close ( ) ;
Fio. Close ( ) ;
```

Didalam class ios terdapat pendefinisian fungsi : int eof ( ) digunakan untuk menentukan apakah pointer pada file telah mencapai akhir darti file tersebut.

5.3 DETEKSI ERROR

Dalam class ios ada 4 (empat) buah fungsi untuk melakukan tes terhadap error yang terjadi didalam stream. Fungsi tersebut adalah :

1. int good ( )
2. int eof ( )
3. int bad ( )
4. int fail ( )

Keempat fungsi itu akan menghasilkan nilai 0 (nol) jika kondisi tersebut salah dan nilai bukan nol jika kondisi tersebut benar.

Jika error terjadi, maka stream harus dibersihkan dahulu dari kesalahan sebelum melanjutkan ke proses berikutnya. Fungsi yang digunakan untuk membersihkan kesalahan pada stream jika terjadi kesalahan : clear ( )

Bentuk umumnya :

Void clear (int flags = 0) ;

Fungsi clear ( ) dalam keadaan default akan membersihkan seluruh flag. Dapat ditentukan flag mana yang akan dibersihkan, dengan memberi argument yang sesuai.

Fungsi int rdstate ( ) digunakan untuk menentukan jenis kesalahan yang terjadi, dengan mengembalikan nilai integer kesalahan tersebut.
Nilai enum yang dikembalikan oleh fungsi rdstate ( ) adalah :

1. goodbit
   0 jika tidak ada error dan 1 jika terjadi error
2. eofbit
   1 jika eof ditemukan dan 0 jika eof tidak ditemukan
3. failbit
   1 jika non fatal error ditemukan dan 0 jika tidak
4. badbit
   1 jika fatal error ditemukan dan 0 jika tidak.

5.4 INPUT/OUTPUT KARAKTER

Input/output pada file binary digunakan fungsi get ( ), put ( ).

Bentuk Umumnya :

Istream &get (char& ch) ;
Ostream &put (char ch) ;

Fungsi get ( ) akan membaca sebuah karakter dari stream dan karakter tersebut akan diletakkan pada variable ch.
Fungsi ini akan mengembalikan reference ke stream jika EOF ditemukan, maka nilai yang dikembalikan NULL.

Fungsi put ( ) akan meletakkan sebuah karakter ke sutau stream dan akan mengembalikan reference ke stream.

Untuk membaca dan menulis file binary dalam bentuk blok data dapat menggunakan fungsi read ( ) dan write 9)

Bentuk Umum :

Istream& read (unsigned char* buf, int num) ;
Ostream& write (const unsigned char* buf, int num);

Fungsi read ( ) akan membaca sejumlah num byte dari stream dan akan meletakkan data tersebut ke dalam variable yang ditunjukkan oleh pointer buf.

Fungsi write ( ) akan menulis sejumlah num byte ke stream dari data yang ditunjuk oleh pointer buf.

Fungsi yang akan mengembalikan karakter yang terakhir dibaca ke dalam memori lagi adalah : putback ( )

Bentuk umum :

Istream& putback (char ch) ;

Keterangan :

- variable ch adalah karakter yang dibaca yang akan dikembalikan lagi ke stream (didalam memori)

FUNCTION
Bab 3

3.1 PENGERTIAN FUNGSI DI C++

Function/fungsi adalah satu blok kode yang melakukan tugas tertentu atau satu blok instruksi yang di eksekusi ketika dipanggil dari bagian lain dalam suatu program.
Bentuk umum deklarasi fungsi :

Tipenilaibalik fungsi (tipepara, …) ;

Keterangan :

- **tipenilaibalik =**
  tipe nilai yang dikembalikan dengan statemen 'return'. Tipe default nya : 'int'. Untuk menyatakan fungsi yang tidak mengembalikan nilai balik, dideklarasikan sebagai : 'void'
- **fungsi =**
  nama fungsi tersebut
- **tipepara =**
  tipe parameter, bila parameter lebih dari satu (1), masing-masing dipisahkan dengan tanda koma (,)
  untuk menyatakan fungsi tanpa parameter dispesifikasikan : 'void'. Bila tipe parameter tidak dispesifikasikan, defaultnya : 'void'

Fungsi harus dideklarasikan terlebih dahulu sebelum didefinisikan. Maksudnya adalah memberitahu compiler jumlah dan tipe parameter yang diterima dan nilai balik fungsi (bila ada) agar compiler dapat memeriksa ketepatannya. **Definisi fungsi** itu sendiri adalah menspesifikasikan tugas fungsi tersebut.

Contoh 1 - program fungsi :

```
# include <iostream.h>
Int tambah (int a, int b )
{
Int r ;
r = a + b ;
return 0 ;
}
Int main ( )
{
Int z ;
z = tambah ( 5,3 ) ;
cout << "Hasilnya = ' << z ;
return 0 ;
}
```

Contoh 2 – program fungsi tanpa tipe :

```
# include <iostream.h>
Void contoh (void)
{
Cout << " ini adalah FUNGSI " ;
}
Int main ( )
{
Contoh ( ) ;
Return 0 ;
}
```

3.2 PARAMETER FUNGSI

Parameter fungsi ada 2, yaitu : parameter formal dan parameter actual.

1. Parameter Formal
   Parameter yang dideklarasikan dibagian blok fungsi.
2. Parameter Aktual
   Parameter yang sebenarnya / parameter yang dilewatkan saat fungsi dipanggil.

Contoh program yg menunjukkan parameter formal dan actual :

```
# include <iostream.h>
Int tambah (int a, int b)          // parameter formal
{
Int r ;
r = a + b ;
return 0 ;
}
Int main ( )
{
Int x, y, z ;
Cin >> x >> y ;
z = tambah ( x , y ) ;             //parameter aktual
cout << "Hasilnya = ' << z ;
return 0 ;
}
```

3.3 PARAMETER FUNGSI PASS BY VALUE

Parameter yang diberikan ke fungsi adalah 'Nilainya', tidak pernah menspesifikasikan variabelnya. Bila ada perubahan pada parameter formal, tidak akan mempengaruhi nilai pada parameter actual.

3.4 PARAMETER FUNGSI PASS BY REFERENCE

Memanipulasi nilai dari dalam fungsi. Setiap perubahan terhadap parameter formal akan memepengaruhi nilai pada parameter actual. Parameter formal diberi symbol '**&**'. Cara ini adalah cara efektif yang memungkinkan sebuah fungsi mengembalikan lebih dari satu nilai.

Contoh program :

```
# include <iostream.h>
Void perkalian (int& a, int& b, int& c)
{
a *= 2 ; b *= 2 ; c *= 2 ;
}
Int main ( )
{
Int x = 1, y = 3, z = 7 ;
Perkalian ( x, y, z ) ;
Cout << " X = " << x << " Y = " << y << " Z = " << z ;
Return 0 ;
}
```

Outputnya :

X = 2 Y = 6 Z = 14

Contoh program :

```
# include <iostream.h>
Void prevnext (int x, int& prev, int& next)
{
prev = x – 1 ;
Next = x + 1 ;
}
Int main ( )
{
Int x = 100 , y, z ;
Prevnext ( x, y, z ) ;
Cout << " Previous = " << y << " , Next = " << z ;
Return 0 ;
}
```

Outputnya :

Previuos = 99, Next = 101

Saat pendeklarasian fungsi dapat langsung diberikan nilai default untuk setiap parameter.

Contoh program :

```
# include <iostream.h>
Int pembagian (int a, int b = 2)
{
Int r ;
r = a / b ;
Return  ( r ) ;
}
Int main ( )
{
Cout << pembagian ( 12 ) ;
Cout << endl ;
Cour << pembagian ( 20, 4) ;
Return 0 ;
}
```

3.4 POLIMORPHISM

Polimorphisme adalah objek-objek yang berbeda tetapi berasal dari satu orang tua, dapat mempunyai metode yang sama tetapi cara pelaksanaannya berbeda-beda. Contohnya adalah objek kendaraan yang terdiri dari objek kendaraan bermotor dan kendaraan tidak bermotor.

C/C++ Keywords

asm	insert an assembly instruction
auto	declare a local variable
bool	declare a boolean variable
break	break out of a loop
case	a block of code in a switch statement
catch	handles exceptions from throw
char	declare a character variable
class	declare a class
const	declare immutable data or functions that do not change data
const_cast	cast from const variables
continue	bypass iterations of a loop
default	default handler in a case statement
delete	make memory available
do	looping construct
double	declare a double precision floating-point variable
dynamic_cast	perform runtime casts
else	alternate case for an if statement
enum	create enumeration types
explicit	only use constructors when they exactly match
export	allows template definitions to be separated from their declarations
extern	tell the compiler about variables defined elsewhere
false	the boolean value of false
float	declare a floating-point variable
for	looping construct
friend	grant non-member function access to private data
goto	jump to a different part of the program
if	execute code based off of the result of a test
inline	optimize calls to short functions
int	declare a integer variable
long	declare a long integer variable
mutable	override a const variable
namespace	partition the global namespace by defining a scope
new	allocate dynamic memory for a new variable
operator	create overloaded operator functions
private	declare private members of a class
protected	declare protected members of a class
public	declare public members of a class

register	request that a variable be optimized for speed
reinterpret_cast	change the type of a variable
return	return from a function
short	declare a short integer variable
signed	modify variable type declarations
sizeof	return the size of a variable or type
static	create permanent storage for a variable
static_cast	perform a nonpolymorphic cast
struct	define a new structure
switch	execute code based off of different possible values for a variable
template	create generic functions
this	a pointer to the current object
throw	throws an exception
true	the boolean value of true
try	execute code that can throw an exception
typedef	create a new type name from an existing type
typeid	describes an object
typename	declare a class or undefined type
union	a structure that assigns multiple variables to the same memory location
unsigned	declare an unsigned integer variable
using	import complete or partial namespaces into the current scope
virtual	create a function that can be overridden by a derived class
void	declare functions or data with no associated data type
volatile	warn the compiler about variables that can be modified unexpectedly
wchar_t	declare a wide-character variable
while	looping construct

STRUKTUR KONDISI DAN STRUKTUR PERULANGAN
BAB 6

6.1 PERCABANGAN / KONDISI

Percabangan adalah suatu proses pemilihan aksi diantara beberapa alternative yang diberikan.

Bentuk umum statemen if :

```
If ( cond-exp) statement ;
```

Bentuk umum statement if … else :

```
If ( cond-exp) statement true
Else statement false ;
```

Jika ada lebih dari 1 (satu) instruksi yang akan dijalankan maka harus dibuat dalam bentuk blok instruksi dengan menggunakan tanda kurung kurawal { … }

Contoh program :

```
If ( nilai > 60 )
        Cout << "Anda lulus \n" ;
Else
        Cout << "Anda tidak lulus \n" ;
```

Untuk membentuk multiway selection, dapat digunakan nested if-else, hal ini dapat dilakukan sebanyak yang diinginkan.

Bentuk umum nested if … else :

```
If ( cond-exp1 )
        Statement1 ;
Else if ( cond-exp2 )
        Statement2 ;
…
…
Else
        Statement n ;
```

Contoh program :

```
If ( x > 0 )
        Cout << " x adalah positif " ;
Else if ( x < 0 )
        Cout << " x adalah negative " ;
Else
        Cout << " x adalah nol " ;
```

6.2 PERULANGAN / LOOP

Ada 3 (tiga) cara untuk melakukan perulangan di C++ yaitu :

1. for statement
2. while statement
3. do while statement

Loop adalah : perulangan statement dengan jumlah tertentu jika kondisi terpenuhi.

6.2.1 FOR STATEMENT

Bentuk umum :

For (<init-exp> ; <test-exp> ; <inc/dec-exp>)
    Statement ;

Keterangan :

- init-exp : ekspresi yang digunakan untuk melakukan inisialisasi terhadap variable-variabel tertentu, terutama variable yang digunakan untuk melakukan iterasi. Init-exp dapat berupa ekspresi maupun pendefinisian variable.
- Test-exp : ekspresi yang memegang control terhadap proses perulangan tersebut, pada bagian ini akan ditentukan apakah proses perulangan akan tetap berlangsung atau tidak.
- Inc/dec-exp : digunakan untuk mengatur perubahan nilai variable. Umumnya nilai variable tersebut bertambah / berkurang 1 (satu)

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int batas ;
Cout << " Nilai tertinggi = " ;
Cin >> batas ;
For ( int i = 1 ; i < = batas ; i++ )
Cout << i << ' ' ;
Return 0 ;
}
```

6.2.2 WHILE STATEMENT

Bentuk umum :

While (cond-exp) statement ;

Statement akan dilaksanakan terus selama cond-exp bernilai true.

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int I = 0 ;
While ( I < 10 )
{
        Cout << " Belajar C++ \n " ;
        I++ ;
}
Return 0 ;
}
```

6.2.3 DO WHILE STATEMENT

Bentuk umum :

Do statement while (cond-exp) ;

Perbedaan dengan while statement adalah : pada do while kondisi akan dievaluasi setelah dilakukan statement/proses. Selain itu pada do while minimal akan dilaksanakan 1 (satu) kali statement/instruksinya.

Contoh program :

```
# include <iostream.h>
Main ( )
{
Int I = 0 ;
Do
{
        Cout << " Belajar C++ \n " ;
        I++ ;
}
While ( I < 10 ) ;
Return 0 ;
}
```

MODUL PEMROGRAMAN BAHASA COBOL/C DENGAN LOGIKA DAN ALGORITMA

Gratcy Palma P Hutapea
gratcypalma@gmail.com
http://mr-amateur.co.cc

Definisi Algoritma
"Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis".

Struktur Bahasa C
/* Definisikan semua tetapan & type */
/* Prosedur & Fungsi */
/* Deklarasi */
/* Deskripsi */

Type Data
Dalam bahasa C terdapat lima type dasar, yaitu :

No.	Type data	Ukuran	Format	Keterangan
1.	char	1 byte	%c	Karakter/string
2.	int	2 byte	%i , %d	Integer/bil bulat
3.	float	4 byte	%f	Float/bil pecahan
4.	double	8 byte	%lf	Pecahan presisi ganda
5.	void	0 byte	-	Tidak bertype

Operator

Operator	Arti
=	Operator penugasan
scanf	Pembacaan
printf	Penulisan

1. Operator Aritmatika

Operator	Arti
*	Untuk perkalian
/	Untuk pembagian
%	Untuk sisa pembagian (modulus)
+	Untuk pertambahan
-	Untuk pengurangan

2. Operator Perbandingan

Operator	Arti
<	Kurang dari
<=	Kurang dari sama dengan
>	Lebih dari
>=	Lebih dari sama dengan
==	Sama dengan
!=	Tidak sama dengan

3. Operator Logika

Operator	Arti
&&	Logika AND (DAN)
||	Logika OR (ATAU)
!	Logika NOT (INGKARAN)

Percabangan (Selection)
* IF
* IF… ELSE…
* IF… ELSE majemuk
* CASE

Perulangan (Iterasi/Repetisi)
* FOR… DO
* REPEAT… UNTIL
* WHILE… DO

BAB I STRUKTUR DASAR & INPUT OUTPUT

Contoh kasus

1. **Program penjumlahan dua buah bilangan**

   Algoritma : [Masukkan bilangan pertama]
   [Masukkan bilangan kedua]
   [tampilkan hasil]

   Program :

```
#include <stdio.h>
#include <conio.h>
main()
{
int a,b,c;
printf ("Program Penjumlahan dua bilangan\n\n");
printf("Masukkan bilangan pertama : ");
scanf("%d",&a);
printf("Masukkan bilangan kedua : ");
scanf ("%d",&b);
c=a+b;
printf ("Jadi jumlah %d+%d=%d",a,b,c);
getche();
}
```

2. **Membuat program menghitung Luas segitiga**

   Algoritma : [masukkan nilai alas]
   [masukkan nilai tinggi]
   [hitung luas]

[tampilkan hasil]

Program :

```
#include <stdio.h>
#include <conio.h>
main ()
{       /* penanda awal blok program */
float alas, tinggi, luas;      /* Perjanjian awal  (tetapan dan type) */
clrscr ();     /* bersihkan layar */
printf ("Masukkan nilai alas dan tinggi. \n");
printf ("\n");       /* ganti baris */
printf ("Alas = ");
scanf ("%f",&alas);        /* input nilai alas */
printf ("Tinggi = ");
scanf ("%f",&tinggi);      /* input nilai tinggi */
luas = 0.5 * alas * tinggi ;      /* rumus segitiga */
printf ("Luas segitiga = %f", luas);      /* cetak hasil */
getche ();     /* tahan Layar */
}       /* penanda akhir blok program */
```

BAB II PENYELEKSIAN KONDISI (“IF, IF ELSE, CASE)

Contoh kasus

1. **Membuat program mencari nilai ganjil dan genap**

Algoritma :
[masukkan sebuah bilangan]
[tentukan pilihan]

Program :

```
#include <stdio.h>
#include <conio.h>
main()
{
int x, sisa;
printf ("Program Menentukkan nilai ganjil/genap\n\n");
printf("Masukkan Nilai X : ");
scanf("%d",&x);
sisa=x%2;
if (sisa==0)
printf ("X adalah bilangan genap");
else
printf ("X adalah bilangan ganjil");

getche();
}
```

2. **Membuat program status kelulusan**

Algoritma :
[masukkan nilai pertama]
[masukkan nilai kedua]
[hitung nilai rata-rata]
[tentukan pilihan]
[jika nilai rata-rata dibawah 60, maka status ‘TIDAK LULUS’]
[jika nilai rata-rata diatas 60, maka status ‘LULUS’]

Program :

```
#include <stdio.h>
#include <conio.h>
main ()
{       /* penanda awal blok program */
```

```
float nilai, nilai1, nilai2; char status;      /* Perjanjian awal  (tetapan dan type) */
clrscr ();     /* bersihkan layar */
printf ("Masukkan nilai. \n");
printf ("\n");       /* ganti baris */
printf ("Nilai pertama = ");
scanf ("%f",&nilai1);        /* input nilai pertama */
printf ("Nilai kedua = ");
scanf ("%f",&nilai2);        /* input nilai kedua */
nilai = (nilai1 + nilai2)/2 ;       /* hitung nilai rata-rata */
printf ("Nilai rata-rata = %f", nilai);
printf ("\n");
if          (nilai >=60 && nilai <=100)       /* percabangan */
            printf ("Status kelulusan = LULUS");
else        printf ("Status kelulusan = TIDAK LULUS");     /* cetak hasil */
getche ();     /* tahan Layar */
}           /* penanda akhir blok program */
```

3. Program menentukan karakter

   Algoritma :
   ```
   [masukkan huruf dari keyboard]
   [tampilkan hasil]
   ```

   Program :
   ```
   #include<stdio.h>
   #include<conio.h>
   main()
   {
   char huruf;
   printf("Program menentukan karakter\n");
   printf("Masukkan sebuah huruf : ");
   scanf("%c",&huruf);
   switch (huruf)
   {
   case 'A' : printf("Huruf yang dimasukkan adalah A"); break;
   case 'B' : printf("Huruf yang dimasukkan adalah B"); break;
   case 'C' : printf("Huruf yang dimasukkan adalah C"); break;
   default  : printf("Bukan A,B,dan C"); break;
   }
   getche();
   }
   ```

BAB III PERULANGAN “WHILE, DO WHILE, FOR”

Contoh kasus

1. Program menuliskan bilangan 1,-2,3,-4,5,-6

   Algoritma :
   ```
   tanda := 1;
    bil:=1;
   while(bil<=6) do
   write(tanda*bil)
   tanda:=-tanda;
   bil:=bil+1
   endwhile
   ```

   Program :
   ```
   #include <stdio.h>
   ```

```
#include <conio.h>
main()
{
int bil, tanda;
clrscr();
printf ("Program menuliskan 1,-2,3,-4,5,-6\n\n");
printf("\n");

tanda=1;
bil=1;
while (bil<=6)
{
printf ("%d\n", tanda*bil);
tanda = -tanda;
bil=bil+1;
}
getche();
}
```

2. Program menghitung jumlah 1+2+3+4+5=..

   Algoritma :
   ```
   Jumlah:=0
   for bil 1 to 5 do
   Jumlah := jumlah+bil;
   efor
   write ('jumlah:=',jumlah)
   ```

   Program :
   ```
   #include <stdio.h>
   #include <conio.h>
   main()
   {
   int bil, jumlah;
   clrscr();
   printf ("Program menghitung jumlah 1+2+3+4+5=..\n\n");
   printf("\n");
   jumlah=0;
   for (bil=1;bil<=5;bil++)
   jumlah=jumlah+bil;
   printf("jumlah 1+2+3+4+5=%d", jumlah);
   getche();
   }
   ```

STUDI KASUS

1. Membuat program menghitung nilai mahasiswa dan rata-rata

   Algoritma :
   ```
   write('tentukan jumlah mahasiswa yang diinput');
   for i=0 <= n= -1 do
   write('input data dan nilai mahasiswa');
   efor
   hasil:=nilai[1]+nilai[2]/2;
   write('tentukan nilai');
   for i=0 <= n= -1 do
   write('total:=total+nilai/2');
   efor
   for i=0 <= n=-1 do
   ```

```
                write('tampilkan hasil');
                efor
Program :
        #include <stdio.h>
        #include <conio.h>
        main()
        {
                struct record_Mhs      {
                                                                char Nim[20];
                                                                char Nama [20];
                                                                float Nilai1;
                                                                float Nilai2;
                                                                char Nilai_huruf;
                                        };


                struct record_Mhs A[10];
                int n,i;
                float nilai_rata2, total;
                float nilai_mid, nilai_final;

                clrscr(); /*bersihkan layar*/
                printf("----------------------- \n");
                printf("            program record          \n");
                printf("            dibuat oleh  : nama anda\n");
                printf("                        Nama : nama anda                        \n");
                printf("                        Nim  :    nim anda                      \n");
                printf("----------------------- \n");
                printf(" Berapa jumlah mahasiswa yang akan diinput?");
                scanf("%d",&n);

                printf("\n");
                for (i=0;i<=n-1;i++)
                {
                        printf("Mahasiswa ke- %d\n",i);
                        printf("Nim :");
                        scanf("%s",A[i].Nim);
                        printf("Nama :");
                        scanf("%s",A[i].Nama);
                        printf("Nilai ujian ke-1:");
                        scanf("%f",&nilai_mid);
                        A[i].Nilai1=nilai_mid;
                        printf("Nilai ujian ke-2:");
                        scanf("%f",&nilai_final);
                        A[i].Nilai2=nilai_final;

                        nilai_rata2=(int)(A[i].Nilai1+A[i].Nilai2)/2;

                        if (nilai_rata2 >=0 && nilai_rata2 <=30)
                                        A[i].Nilai_huruf='E';
                        if (nilai_rata2 >=31 && nilai_rata2 <=50)
                                        A[i].Nilai_huruf='D';
                        if (nilai_rata2 >=51 && nilai_rata2 <=60)
                                        A[i].Nilai_huruf='C';
                        if (nilai_rata2 >=61 && nilai_rata2 <=80)
                                        A[i].Nilai_huruf='B';
```

```
		if (nilai_rata2 >=81 && nilai_rata2 <=100)
				A[i].Nilai_huruf='A';
		printf("\n");
	}


	total=0;

	for (i=0;i<=n-1;i++)
			total=total+(A[i].Nilai1+A[i].Nilai2)/2;


	clrscr();/*bersihkan layar*/
	printf("Isi array mahasiswa adalah:\n");

	for (i=0;i<=n-1;i++)
	{
		printf("Mahasiswa ke-%d\n",i);
		printf("Nim:%s\n",A[i].Nim);
		printf("Nama:%s\n",A[i].Nama);
		printf("Nilai ujian ke-1=%f\n",A[i].Nilai1);
		printf("Nilai ujian ke-2=%f\n",A[i].Nilai2);
		printf("Rata_rata nilai= %f\n",(A[i].Nilai1+A[i].Nilai2)/2);
		printf("Nilai huruf=%c\n",A[i].Nilai_huruf);
		printf("\n");
	}
	printf("\n");/*baris kosong*/
	printf("Nilai total kelas=%f\n",total);
	printf("Rata-rata kelas=%f\n",total);
	printf("\nTekan<enter>untuk melanjutkan!");
	getche();/*tahan layarnya*/
}
```

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab II
Struktur Dasar Bahasa C

1 Tipe Data

Tipe data merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh computer. Misalnya saja 5 dibagi 2 bisa saja menghasilkan hasil yang berbeda tergantung tipe datanya. Jika 5 dan 2 bertipe integer maka akan menghasilkan nilai 2, namun jika keduanya bertipe float maka akan menghasilkan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif.

Dalam bahasa C terdapat lima tipe data dasar, yaitu :

No	Tipe Data	Ukuran	Range (Jangkauan)	Format	Keterangan
1	char	1 byte	- 128 s/d 127	%c	Karakter/string
2	int	2 byte	- 32768 s/d 32767	%i , %d	Integer/bilangan bulat
3	float	4 byte	- 3.4E-38 s/d 3.4E+38	%f	Float/bilangan pecahan
4	double	8 byte	- 1.7E-308 s/d 1.7+308	%lf	Pecahan presisi ganda
5	void	0 byte	-	-	Tidak bertipe

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
void main()
```

```
{   int x;
    float y;
    char z;
    double w;
    clrscr();                   /* untuk membersihkan layar */
    x = 10;                     /* variable x diisi dengan 10 */
    y = 9.45;                   /* variable y diisi dengan 9.45 */
    z = 'C';                    /* variable z diisi dengan karakter "C" */
    w = 3.45E+20;               /* variable w diisi dengan 3.45E+20 */
    printf("Nilai dari x adalah : %i\n", x); /* Menampilkan isi variable x */
    printf("Nilai dari y adalah : %f\n", y); /* Menampilkan isi variable y */
    printf("Nilai dari z adalah : %c\n", z); /* Menampilkan isi variable z */
    printf("Nilai dari w adalah : %lf\n", w);       /* Menampilkan isi variable w */
    getch(); }
```

2 Konstanta

Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta nilainya selalu tetap. Konstanta harus didefinisikan terlebih dahulu di awal program. Konstanta dapat bernilai integer, pecahan, karakter dan string. Contoh konstanta : 50; 13; 3.14; 4.50005; 'A'; 'Bahasa C'. Selain itu, bahasa C juga menyediakan beberapa karakter khusus yang disebut karakter escape, antara lain :

- \a : untuk bunyi bell (alert)
- \b : mundur satu spasi (backspace)
- \f : ganti halaman (form feed)
- \n : ganti baris baru (new line)
- \r : ke kolom pertama, baris yang sama (carriage return)
- \v : tabulasi vertical
- \0 : nilai kosong (null)
- \' : karakter petik tunggal
- \" : karakter petik ganda
- \\ : karakter garis miring

3 Variable

Variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variable bisa diubah-ubah sesuai kebutuhan. Nama dari suatu variable dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut :

- Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa C bersifat case-sensitive artinya huruf besar dan kecil dianggap berbeda. Jadi antara **nim, NIM** dan **Nim** dianggap berbeda.
- Tidak boleh mengandung spasi.
- Tidak boleh mengandung symbol-simbol khusus, kecuali garis bawah (underscore). Yang termasuk symbol khusus yang tidak diperbolehkan antara lain : $, ?, %, #, !, &, *, (,), -, +, = dsb
- Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.
  Contoh penamaan variabel yang benar :
  NIM, a, x, nama_mhs, f3098, f4, nilai, budi, dsb.
  Contoh penamaan variable yang salah :
  %nilai_mahasiswa, 80mahasiswa, rata-rata, ada spasi, penting!, dsb

4 Deklarasi

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variable, konstanta dan fungsi.

Deklarasi Variabel

Bentuk umum pendeklarasian suatu variable adalah :

```
Nama_tipe nama_variabel;
```

Contoh :

```
int x;                          // Deklarasi x bertipe integer
char y, huruf, nim[10];         // Deklarasi variable bertipe char
float nilai;                    // Deklarasi variable bertipe float
double beta;            // Deklarasi variable bertipe double
int array[5][4];                // Deklarasi array bertipe integer
char *p;                        // Deklarasi pointer p bertipe char
```

Deklarasi Konstanta

Dalam bahasa C konstanta dideklarasikan menggunakan preprocessor #define. Contohnya :

```
#define PHI 3.14
#define nim "0111500382"
#define nama "Sri Widhiyanti"
```

Deklarasi Fungsi

Fungsi merupakan bagian yang terpisah dari program dan dapat diaktifkan atau dipanggil di manapun di dalam program. Fungsi dalam bahasa C ada yang sudah disediakan sebagai fungsi pustaka seperti printf(), scanf(), getch() dan untuk menggunakannya tidak perlu dideklarasikan. Fungsi yang perlu dideklarasikan terlebih dahulu adalah fungsi yang dibuat oleh programmer. Bentuk umum deklarasi sebuah fungsi adalah :

```
Tipe_fungsi nama_fungsi(parameter_fungsi);
```

Contohnya :

```
float luas_lingkaran(int jari);
void tampil();
int tambah(int x, int y);
```

5 Operator

Operator Penugasan

Operator Penugasan (*Assignment operator*) dalam bahasa C berupa tanda sama dengan ("="). Contoh :

```
nilai = 80;
A = x * y;
```

Artinya : variable "nilai" diisi dengan 80 dan variable "A" diisi dengan hasil perkalian antara x dan y.

Operator Aritmatika

Bahasa C menyediakan lima operator aritmatika, yaitu :

- * : untuk perkalian
- / : untuk pembagian
- % : untuk sisa pembagian (modulus)
- + : untuk pertambahan
- - : untuk pengurangan

Catatan : operator % digunakan untuk mencari sisa pembagian antara dua bilangan.
Misalnya :
9 % 2 = 1
9 % 3 = 0

9 % 5 = 4
9 % 6 = 3

Contoh Program 1 :

```
#include "stdio.h"
#include "conio.h"
void main()
{   clrscr();              // untuk membersihkan layar
    printf("Nilai dari 9 + 4 = %i", 9 + 4);          /* mencetak hasil 9 + 4 */
    printf("Nilai dari 9 - 4 = %i", 9 - 4);          /* mencetak hasil 9 - 4 */
    printf("Nilai dari 9 * 4 = %i", 9 * 4);          /* mencetak hasil 9 * 4 */
    printf("Nilai dari 9 / 4 = %i", 9 / 4);          /* mencetak hasil 9 / 4 */
    printf("Nilai dari 9 \% 4 = %i", 9 % 4);         /* mencetak hasil 9 % 4 */
    getch();
}
```

Contoh Program 2 :

```
/* Penggunaan operator % untuk mencetak deret bilangan genap antara 1 – 100 */
#include "stdio.h"
#include "conio.h"
void main()
{   int bil;
    clrscr();                                   // untuk membersihkan layar
    for (bil=1; bil<100; bil++)
    {   if(bil % 2 == 0)                  //periksa apakah 'bil' genap
            printf("%5.0i", bil);
    }
    getch();
}
```

- **Operator Hubungan (Perbandingan)**
  Operator Hubungan digunakan untuk membandingkan hubungan antara dua buah operand (sebuah nilai atau variable. Operator hubungan dalam bahasa C :

Operator	Arti	Contoh	
<	Kurang dari	x < y	Apakah x kurang dari y
<=	Kurang dari sama dengan	x <= y	Apakah x kurang dari sama dengan y
>	Lebih dari	x > y	Apakah x lebih dari y
>=	Lebih dari sama dengan	x >= y	Apakah x lebih dari sama dengan y
==	Sama dengan	x == y	Apakah x sama dengan y
!=	Tidah sama dengan	x != y	Apakah x tidak sama dengan y

- **Operator Logika**
  Jika operator hubungan membandingkan hubungan antara dua buah operand, maka operator logika digunakan untuk membandingkan logika hasil dari operator-operator hubungan. Operator logika ada tiga macam, yaitu :
  - && : Logika AND (DAN)
  - || : Logika OR (ATAU)
  - ! : Logika NOT (INGKARAN)

- **Operator Bitwise**
  Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori. Operator bitwise dalam bahasa C :
  - << : Pergeseran bit ke kiri

- >> : Pergeseran bit ke kanan
- & : Bitwise AND
- ^ : Bitwise XOR (exclusive OR)
- | : Bitwise OR
- ~ : Bitwise NOT

Operator Unary

Operator Unary merupakan operator yang hanya membutuhkan satu operand saja. Dalam bahasa C terdapat beberapa operator unary, yaitu :

Operator	Arti/Maksud	Letak	Contoh	Equivalen
-	Unary minus	Sebelum operator	A + -B * C	A + (-B) * C
++	Peningkatan dengan penambahan nilai 1	Sebelum dan sesudah	A++	A = A + 1
--	Penurunan dengan pengurangan nilai 1	Sebelum dan sesudah	A--	A = A – 1
sizeof	Ukuran dari operand dalam byte	Sebelum	sizeof(I)	-
!	Unary NOT	Sebelum	!A	-
~	Bitwise NOT	Sebelum	~A	-
&	Menghasilkan alamat memori operand	Sebelum	&A	-
*	Menghasilkan nilai dari pointer	Sebelum	*A	-

Catatan Penting ! :

Operator peningkatan ++ dan penurunan -- jika diletakkan sebelum atau sesudah operand terdapat perbedaan. Perhatikan contoh berikut :

Contoh Program 1 :

```
/* Perbedaan operator peningkatan ++ yang diletakkan di depan dan dibelakang operand */
        #include <stdio.h>
        #include <conio.h>
        void main()
        {
          int x, nilai;
            clrscr();
          x = 5;
              nilai = ++x;   /* berarti x = x + 1; nilai = x;  */
          printf("nilai = %d, x = %d\n", nilai, x);
          nilai = x++;       /* berarti nilai = x;  nilai = x + 1;  */
          printf("nilai = %d, x = %d\n", nilai, x);
            getch();
        }
```

Contoh Program 2 :

```
#include “stdio.h”
#include “conio.h”
void main()
{   int b, nilai;
     clrscr();                  // untuk membersihkan layar
     b = 15;
     nilai = --b;    /* berarti b = b - 1; nilai = b;  */
```

```
    printf("nilai = %d, b = %d\n", nilai, b);
    nilai = b--;    /* berarti nilai = b;  b = b + 1;  */
    printf("nilai = %d, b = %d\n", nilai, b);
    getch();
}
```

6 Kata Tercadang (Reserved Word)

Bahasa C standar ANSI memiliki 32 kata tercadang (reserved word) dan Turbo C menambahkannya dengan 7 kata tercadang. Semua *reserved word* tidak boleh digunakan dalam penamaan identifier (variable, nama fungsi dll). Kata Tercadang yang tersedia dalam bahasa C adalah sbb:

*asm	default	for	*pascal	switch
auto	do	goto	register	typedef
break	double	*huge	return	union
case	else	if	short	unsigned
*cdecl	enum	int	signed	void
char	extern	*interrupt	sizeof	volatile
const	*far	long	static	while
continue	float	*near	struct	

Keterangan : tanda * menunjukkan tambahan dari Turbo C

7 Komentar Program

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan komentar atau penjelasan dalam bahasa C digunakan pembatas /* dan */ atau menggunakan tanda // untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan).

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
void main()
{
    clrscr();           /* Ini untuk membersihkan layar tampilan */
    printf(“Contoh Penggunaan Komentar”);   //komentar tidak ikut diproses
    getch();            /* Dan ini untuk menahan tampilan di layer */
}
```

Latihan 2

Buatlah Program dalam Bahasa C untuk :

1. Mencari jawaban dari sebuah persamaan kuadrat dengan menggunakan rumus abc.
2. Mencetak deret bilangan 1 2 4 8 16 32 64
3. Menghitung jumlah dan rata-rata dari 5 buah bilangan bulat positip.

BAHASA PEMROGRAMAN C

Bahasa Pemrograman C

A. Pengenalan

Bahasa C diciptakan oleh Dennis Ritchie tahun 1972 di Bell Laboratories.

Kelebihan Bahasa C:

- Bahasa C tersedia hampir di semua jenis computer.
- Kode bahasa C sifatnya adalah portable dan fleksibel untuk semua jenis computer.
- Bahasa C hanya menyediakan sedikit kata-kata kunci. hanya terdapat 32 kata kunci.
- Proses executable program bahasa C lebih cepat
- Dukungan pustaka yang banyak.
- C adalah bahasa yang terstruktur
- Bahasa C termasuk bahasa tingkat menengah

  penempatan ini hanya menegaskan bahwa c bukan bahasa pemrograman yang berorientasi pada mesin. yang merupakan ciri bahasa tingkat rendah. melainkan berorientasi pada obyek tetapi dapat dinterprestasikan oleh mesin dengan cepat. secepat bahasa mesin. inilah salah satu kelebihan c yaitu memiliki kemudahan dalam menyusun programnya semudah bahasa tingkat tinggi namun dalam mengesekusi program secepat bahasa tingkat rendah.

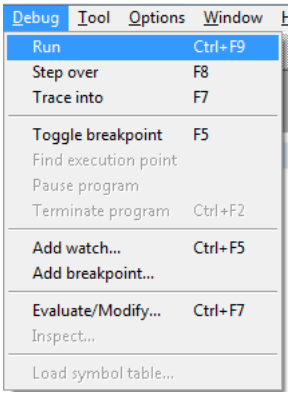
Kekurangan Bahasa C:

- Banyaknya operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
- Bagi pemula pada umumnya akan kesulitan menggunakan pointer.

B. MENGKOMPILASI PROGRAM

Suatu source program C baru dapat dijalankan setelah melalui tahap kompilasi dan penggabungan. Tahap kompilasi dimaksudkan untuk memeriksa source-program sesuai dengan kaidah-kaidah yang berlaku di dalam bahasa pemrograman C. Tahap kompilasi akan menghasilkan *relocatable object file*. File-file objek tersebut kemudian digabung dengan perpustakaan-fungsi yang sesuai. untuk menghasilkan suatu *executable-program*. Shortcut yang digunakan untuk mengkompile:

- ALT + F9 → dipakai untuk melakukan pengecekan jika ada error pada program yang telah kita buat.
- CTRL + F9 → dipakai untuk menjalankan program yang telah kita buat atau bisa juga dengan mengklik tombol debug pada tool bar.



C. Struktur Bahasa Pemrograman C

<preprosesor directive>

{

<statement>;

<statement>;

}

Ketikkanlah program sederhana berikut ini:

```
#include <stdio.h>

void main ()
{
   printf ("hello ^^");
}
```

Kemudian kompile. Apa hasilnya?

Penjelasan:

1. **Header File**

   adalah berkas yang berisi prototype fungsi. definisi konstanta. dan definisi variable. Fungsi adalah kumpulan code C yang diberi nama dan ketika nama tersebut dipanggil maka kumpulan kode tersebut dijalankan.

   Contoh :

   stdio.h

math.h

conio.h

2. **Preprosesor Directive (#include)**

   Preprosesor directive adalah bagian yang berisi pengikutsertaan file atau berkas-berkas fungsi maupun pendefinisian konstanta.

   Contoh:

   #include <stdio.h>

   #include phi 3.14

3. **Void**

   artinya fungsi yang mengikutinya tidak memiliki nilai kembalian (return).

4. **Main ( )**

   Fungsi main ( ) adalah fungsi yang pertama kali dijalankan ketika program dieksekusi. tanpa fungsi main suatu program tidak dapat dieksekusi namun dapat dikompilasi.

5. **Statement**

   Statement adalah instruksi atau perintah kepada suatu program ketika program itu dieksekusi untuk menjalankan suatu aksi. Setiap statement diakhiri dengan titik-koma (;).

D. Kata Kunci (Keyword)

Kata kunci-kata kunci yang terdapat di C, sebagai berikut:

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

E. IDENTIFIER

Identifier atau nama pengenal adalah nama yang ditentukan sendiri oleh pemrogram yang digunakan untuk menyimpan nilai, misalnya nama variable, nama konstanta, nama suatu elemen (misalnya: nama fungsi, nama tipe data, dll). Identifier punya ketentuan sebagai berikut :

1. Maksimum 32 karakter (bila lebih dari 32 karakter maka yang diperhatikan hanya 32 karakter pertama saja).
2. Case sensitive: membedakan huruf besar dan huruf kecilnya.
3. Karakter pertama harus karakter atau underscore (_) . selebihnya boleh angka.
4. Tidak boleh mengandung spasi atau blank.
5. Tidak boleh menggunakan kata yang sama dengan kata kunci dan fungsi.

VARIABEL

Variabel adalah identifier yang nilainya dapat berubah atau diubah selama program berjalan (dieksekusi). Pengubahnya adalah user atau proses.

- Deklarasi variabel (tipe_data nama_variabel;)

  Variabel yang akan digunakan dalam program haruslah dideklarasikan terlebih dahulu. Pengertian deklarasi di sini berarti memesan memori dan menentukan jenis data yang bisa disimpan di dalamnya.

  Contoh :

```
int a, b, c;
```

- Inisialisasi variabel (tipe_data nama_variabel = nilai;)

```
int a=15, b=7, c=0;
```

KONSTANTA

Konstanta adalah identifier yang nilainya tetap selama program berjalan/ dieksekusi. Cara untuk mengubahnya hanya melalui source codenya saja seperti halnya variabel, konstanta juga memiliki tipe. Penulisan konstanta mempunyai aturan tersendiri, sesuai dengan tipe masing-masing.

1. Konstanta karakter misalnya ditulis dengan diawali dan diakhiri dengan tanda petik tunggal, contohnya : ‘A’ dan ‘@’.
2. Konstanta integer ditulis dengan tanda mengandung pemisah ribuan dan tidak mengandung bagian pecahan, contohnya : –1 dan 32767.
3. Konstanta real (*float* dan *double*) bisa mengandung pecahan (dengan tanda berupa titik) dan nilainya bisa ditulis dalam bentuk eksponensial (menggunakan tanda e), contohnya : 27.5f (untuk tipe *float*) atau 27.5 (untuk tipe *double*) dan 2.1e+5 (maksudnya 2,1 x 105).
4. Konstanta string merupakan deretan karakter yang diawali dan diakhiri dengan tanda petik-ganda (“), contohnya :“Pemrograman Dasar C”.

Contoh :

```
#define phi 3.14
#define max_data 50
```

F. Tipe Data Dasar

Data merupakan suatu nilai yang bisa dinyatakan dalam bentuk konstanta atau variabel. Konstanta menyatakan nilai yang tetap, sedangkan variabel menyatakan nilai yang dapat diubah-ubah selama eksekusi berlangsung.

Ukuran Memori untuk Tipe Data

Tipe Data	Ukuran Memori	Kawasan
unsigned char	8 bits	0 s/d 255
char	8 bits	-128 s/d 127
short int	16 bits	-32.768 s/d 32.767
unsigned int	32 bits	0 s/d 4.294.967.295
int	32 bits	-2.147.483.648 s/d 2.147.483.647
unsigned long	32 bits	0 s/d 4.294.967.295
enum	16 bits	-2147483.648 to 2.147.483.648
long	32 bits	-2.147.483.648 s/d 2.147.483.647
float	32 bits	3,4 x 10-38 s/d 3,4 x 10+38
double	64 bits	1.7 x 10-308 to 1.7 x 10+308
long double	80 bits	3.4 x 10-4932 to 3.4 x 10+4932
near (pointer)	32 bits	not applicable
far (pointer)	32 bits	not applicable

Untuk mengetahui ukuran memory bisa dipakai fungsi sizeof (<tipe_data>).

Catatan:

Ukuran dan kawasan dari masing-masing tipe data adalah bergantung pada jenis mesin yang digunakan (misalnya mesin 16 bit bisa jadi memberikan hasil berbeda dengan mesin 32 bit).

Untuk menampilkan hasil output dibutuhkan kode format, berikut adalah daftar kode format:

Kode format	**Kegunaan**
%c	Menampilkan sebuah karakter
%s	Menampilkan nilai string
%d	Menampilkan nilai decimal integer
%i	Menampilkan nilai decimal integer
%u	Menampilkan nilai decimal integer tidak bertanda (unsigned integer)
%ld	Menampilkan nilai decimal long integer
%lu	Menampilkan nilai decimal long integer tak bertanda
%li	Menampilkan nilai decimal long integer
%hu	Menampilkan nilai decimal short integer tak bertanda
%hi	Menampilkan nilai decimal short integer
%x	Menampilkan nilai heksa decimal integer
%o	Menampilkan nilai okta integer
%f	Menampilkan nilai pecahan / float
%e	Menampilkan nilai float scientific
%g	Sebagai pengganti %f atau %e tergantung yang terpendek
%lf	Menampilkan nilai pecahan double
%le	Menampilkan nilai pecahan double
%lg	Menampilkan nilai pecahan double
%p	Menampilkan suatu alamat memory untuk pointer

Contoh:

```
#include <stdio.h>

void main()
{
   int a;
   a=sizeof (long);
   printf ("ukuran tipe data long adalah %i bytes",a);
}
```

1. Coba hilangkan %i nya, apa hasilnya?
2. Coba ganti a menjadi ukutan tipe data yang lain, misalnya float dan char!

G. Karakter Escape

Karakter escape adalah karakter yang diawali dengan tanda backslash (/), yang masing-masing memiliki makna tertentu. Berikut adalah daftar karakter:

KARAKTER ESCAPE	ARTI
\a	Bunyi bel (speaker komputer)
\b	Mundur satu spasi (backspace)
\f	Ganti halaman (form feed)
\n	Ganti baris baru (new line)
\r	Ke kolom pertama baris yang sama (carriage return)
\t	Tab horizontal
\v	Tab vertical
\0	Nilai kosong (NULL)
\’	Karakter petik tunggal
\”	Karakter petik ganda
\\	Garis miring terbalik (backslash)
\?	Karakter tanda tanya
\DDD	Menyatakan sebuah karakter yang nilai ASCII nya sama dengan nilai octal DDD
\xHH	Menyatakan sebuah karakter yang nilai ASCII nya sama dengan nilai heksadesimal HH

H. STATEMENT

Apa itu statement?

Macam-macam statement:

1. **Satement Kosong/ Empty Statement/ Null Statement**

   Statement kosong adalah statement yang hanya terdiri dari pengakhiran titik koma (;) saja, sehingga tidak ada tindakan yang akan dilakukan. Statement kosong digunakan untuk memberikan jarak waktu atau delay.

   Contoh :

```
for (int i=0; i<1000; i++);
```

2. **Statement Ungkapan/ Expression Statement**

   Statement Ungkapan adalah statement yang dibentuk dari suatu ungkapan yang diakhiri dengan titik koma (;).

   Contoh :

```
luas = panjang * lebar;
scanf ("%i",&luas);
printf ("nilai luas persegi adalah  %i",luas);
luas--;
```

3. **Statement Kendali/ Control Statement**

   Statement kendali adalah statement yang dipakai untuk mengendalikan proses dari program, yaitu penyeleksian kondisi/ percabangan (if, case, switch) dan lompatan /perulangan (for, while, do-while, break, continue).

   Contoh :

```
if (N<0)
printf("nilai N negatif");
else
printf("nilai N positif");
```

4. **Statement Jamak/ Compound Statement/ Block Statement**

   Statement jamak adalah statement yang terdiri dari gabungan beberapa statement tunggal yang ditulis diantara tanda kurung kurawal ("{" dan "}").

   Contoh :

```
{
    scanf("%f",&Panjang);
    scanf("%f",&Lebar);
    Luas=Panjang*Lebar;
    Printf("Luas = %f",Luas);
```

LATIHAN 1:

```
#include <stdio.h>

void main ()
{
   printf ("1.karakter escape\t2.karakter escape");
}
```

1. Coba ganti \t dengan \n atau \b!
2. Tambahankan //printf (“memberikan comment”); di bawah print (“1.karakter escape\t2.karakter escape");

LATIHAN 2:

```
#include <stdio.h>

void main ()
{
   int a=36000;
   printf ("%i",a);
}
```

1. Jika program di atas di jalankan, bagaimanakah tampilannya? Mengapa demikian?
2. Coba ganti **int** menjadi **long int** dan **%i** menjadi **%li**. Bagaimana tampilannya?

LATIHAN 3:

```
#include <stdio.h>
void main()
{
   int angka=12345;
   printf("%d\n",angka);
   printf("%8d\n",angka);
   printf("%11d\n",angka);
   printf("%08d\n",angka);
   printf("%-8d***\n",angka);
}
```

Perhatikan setiap perbedaan yang dihasilkan!

LATIHAN 4:

```
#include <stdio.h>
void main()
{
   float pecahan=12.3456789;
   printf("%f\n%.2f\n%.3f",pecahan,pecahan,pecahan);
}
```

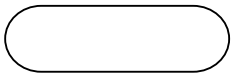
Adakah perbedaan hasil format yang dihasilkan???

LATIHAN 5:

```
#include <stdio.h>
void main()
{
   char huruf='s';
   printf("Huruf anda : %c...",huruf);
   printf("\nHuruf anda : %5c...",huruf);
   printf("\nHuruf anda : %-5c...",huruf);
}
```

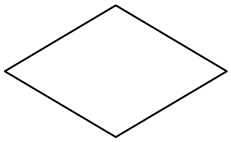
Perhatikan setiap perbedaan yang dihasilkan! Coba bandingkan dengan latihan 3!

I. FLOWCHART

1.  Dipakai untuk menunjukkan awal dimulai/diakhirinya suatu proses

2.  Dipakai untuk memberikan inputan/ output

3.  Dipakai untuk menuliskan proses dalam suatu program

4.  Dipakai untuk menuliskan jika adanya percabangan, seperti if, case

5.  Dipakai untuk menuliskan jika adanya perulangan atau lompatan (for, while, do-while, break, continue)

6. Dipakai untuk menunjukkan alur jalannya program

DAFTAR PUSTAKA

http://opensource.telkomspeedy.com/forum/viewtopic.php?id=4088

http://buletin.melsa.net.id/okt/1020/bahasa-c.html

http://lecturer.ukdw.ac.id/anton/download/algoTI2.pdf

modul-praktikum-pki-ukdw(2008)

Program-program percobaan mengenal algoritma dan bahasa C

File : **Hitung1.C**
(Program dengan *Input langsung-Proses-dan output*)

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

int A;
int B;
int C;

void main()
{
A=25;
B=35;
C=A+B;
printf("Hasil Jumlahnya adalah C=%d",C);
getch();
};
```

File : **Hitung2.C**
(*Input tidak langsung menggunakan scanf - Proses – dan Output*)

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

int A;
int B;
int C;

void main()
{
printf("Masukkan angka pertama :");scanf("%d",&A);
printf("Masukkan angka kedua    :");scanf("%d",&B);
C=A+B;
printf("Hasil Jumlahnya adalah C=%d",&C);
getch();
};
```

File : **Cetak1.C** (*Program tanpa input dan tanpa proses, hanya mencetak*)

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

void main()
{
printf("NIM         : 0523100\n");
printf("Nama        : Hasan\n");
printf("Umur        : 35\n");
printf("Alamat      : Jakarta\n");
getch();
};
```

File : **Cetak2.C** (*Program dengan input dan output tetapi tanpa proses*)

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

void main()
{
char *Nim;
char *Nm;
int Umr;
char *Almt;

Nim="0523100";
Nm="Hasan";
Umr=35;
Almt="Jakarta";

printf("NIM       : %s\n",Nim);
printf("Nama      : %s\n",Nm);
printf("Umur      : %d\n",Umr);
printf("Alamat    : %s\n",Almt);
getch();
};
```

File : **InputString1.C**
(Program dengan *Input data String tidak langsung)*

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

void main()
{
char Nim[12];
char Nm[25];
int Umr;
char Almt[30];

printf("Masukkan NIM  anda :");gets(Nim);
printf("Masukkan Nama anda :");gets(Nm);
printf("Masukkan Alamat anda :");gets(Almt);
printf("Masukkan Umur anda :");scanf("%d",&Umr);

printf("\n\nAnda telah memasukkan data pribadi
sebagai berikut :");
printf("\nNIM       : %s",Nim);
printf("\nNama      : %s",Nm);
printf("\nUmur      : %d tahun",Umr);
printf("\nAlamat    : %s",Almt);
printf("\n\nTerima kasih...Tekan sembarang
tombol...!");
getch();
};
```

File : **FormatCetak.C**
(*Mencetak beberapa tipe data dengan berbagai formatnya*)

```
//************************************
//Program : Mengatur tampilan cetakan
//NIM  : NIMAnda
//Nama : NamaAnda
//************************************

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

void main()
{
int a;
float c;
char *S;

a=250;
c=2.5;
S="Hallo Unsada";
printf("\nNilai pada a : %d",a);
printf("\nNilai pada a : %5d",a);
printf("\nNilai pada c : %f ",c);
printf("\nNilai pada c : %3f ",c);
printf("\nNilai pada c : %3.1f ",c);
printf("\nNilai pada c : %3.2f ",c);
printf("\nNilai pada c : %3.4f ",c);
printf("\nData pada S : %s ",S);
printf("\nData pada S : %15s ",S);
printf("\nData pada S : %-15s ",S);
printf("\nData pada S : %25s ",S);
printf("\nData pada S : %-25s ",S);
getch();
};
```

File : **Pembagian1.C**
(*Program Pembagian tipe data integer*)

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

void main()
{
int A,B,C;

A=5;
B=2;
C=A/B;
printf("Hasil baginya : %d ",C);
getch();
};
```

Ganti tipe data **int** dengan **float** dan beri nama file yang baru : **Pembagian2.C** dan jalankan !

Latihan :

Buatlah program dengan tampilan seperti berikut :

1.

```
Masukkan nilai A : 15
Masukkan Nilai B : 13

Hasil jumlahnya = 28
Hasil Pengurangannya = 2
```

Simpan ke server
Nama file : Latihan01.c

2.

```
Ketikkan nama anda : Budi
Masukkan umur anda : 25

Hai Budi
Anda berumur : 25 tahun
Selamat Belajar Pemrograman dengan bahasa C
```

Simpan ke server
Nama file : Latihan02.c

3.

```
Aplikasi Perhitungan Nilai
==========

Ketikkan Nama Anda : Herman

Masukkan nilai Teori (0 - 100) : 65
Masukkan nilai Praktek (0 – 100) : 70

Hai Herman
Nilai Rata-rata Anda : 67.50
Anda dinyatakan : LULUS
```

Simpan ke server
Nama file : Latihan03.c

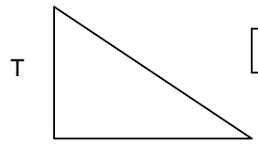
Ket. Jika nilai lebih besar atau sama dengan 60 maka LULUS, selain itu GAGAL

Praktikum 3

Membuat Program sendiri :

Contoh soal :

Buatlah program untuk menghitung Luas (L) segitiga siku-siku jika diketahui panjang Alas (A) dan panjang Tingginya (T)



L=1/2 AT =0.5*A*T

Penyelesaian :

a. Menentukan Variabel input dan Variabel output yang dibutuhkan
   Ada 2 Variabel Input (Data yang diketahui), yaitu :
   - Panjang Alas, diberi nama variabelnya : A
   - Panjang Tinggi, diberi nama variabel : T

   Ada 1 variabel Output (Data yang ditanya), yaitu :
   - Luas segititiga tersebut, diberi nama variabel : Luas

b. Menentukan tampilan program :

```
Program Menghitung Luas Segitiga
========================

Masukkan nilai panjang Alas : _
Masukkan nilai tinggi : _

Luas segitiga tersebut adalah : ….
```

c. Menuliskan algoritma dari program :
   Secara sederhana algoritma program dapat ditulis demikian :
   - Masukkan nilai Alas ke variabel A (scanf(“%d”,&A))
   - Masukkan nilai Tinggi ke variabel T(scanf(“%d”,&T))
   - Hitung Luas = 0.5*A*T
   - Tampilkan hasil Luas (printf (“Hasil : %d”,Luas))

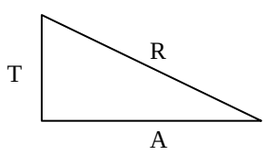
d. Menuliskan program C seperti berikut :

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

void main()
{
int A,T, Luas;

printf(“Program Menghitung Luas Segitiga\n”);
printf(“=========================\n\n”);
printf(“Masukkan nilai panjang Alas ”);scanf(“%d”,&A);
printf(“Masukkan nilai Tinggi ”);scanf(“%d”,&T);
Luas=0.5*A*T;
printf(“\nLuas segitiga tersebut adalah : %d”,Luas);
getch()

}
```

Latihan :

1. Buatlah program untuk menghitung Sisi Miring (R) segitiga siku-siku !

$$R = \sqrt{A^2 + T^2}$$



2. Buatlah program untuk menghitung Luas lingkaran dengan rumus **Luas = 0.5 * PI * radius * radius**
3. Buatlah program untuk menghitung pembayaran Pulsa telepon jika ditentukan biaya pulsa telpon per 100 detik adalah 500 rupiah, berapakan pembayarannya untuk input waktu yang dimasukkan 3 jam.

- Selamat Belajar -

Praktikum 4

Menggunakan perintah if untuk kontrol Alur : Decision

A. Program Tanpa kontrol if

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

void main()
{
int Nilai;

printf("Masukkan Nilai Anda :");scanf("%d",&Nilai);
printf("Selamat, Anda Lulus");
printf("Maaf, Anda Belum Berhasil");
getch();
}
```

Perhatikan semua yang ada diperintah printf ditampilkan di layar

Jalankan program cobalah dengan beberapa macam nilai !

B. Menggunakan kontrol logika if

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

void main()
{
int Nilai;

printf("Masukkan Nilai Anda :");scanf("%d",&Nilai);

 if (Nilai>60)
 {
  printf("Selamat, Anda Lulus");
 }
printf("Maaf, Anda Belum Berhasil");
getch();
}
```

Perhatikan perintah :
printf("Selamat Anda Lulus")
kadang ditampilkan di layar kadang tidak, tergantung Nilai yang dimasukkan, sedangkan perintah ***printf("Maaf anda belum berhasil")*** selalu ditampilkan sebab tidak berada dalam kurung kurawal dari perintah if

Jalankan program cobalah dengan beberapa macam nilai !

C. Menggunakan kontrol logika if - else

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

void main()
{
int Nilai;

printf("Masukkan Nilai Anda :");scanf("%d",&Nilai);

 if (Nilai>60)
 {
  printf("Selamat, Anda Lulus");
 }
 else
 {
  printf("Maaf, Anda Belum Berhasil");
 }
getch();
}
```

Perhatikan perintah :
printf("Selamat Anda Lulus") atau
printf("Maaf anda belum berhasil"), hanya ditampilkan salah satu, tergantung Nilai yang dimasukkan

Jalankan program cobalah dengan beberapa macam nilai !

Latihan :

1. Buatlah program untuk menentukan apakah umur yang dimasukkan termasukkan telah tua atau masih muda, dengan aturan jika umur lebih kecil dari 45 masih muda dan jika umur lebih besar dari 45 sudah tua !
2. Buatlah program yang pertama sekali meminta memasukkan nilai tertentu, kemudian setelah nilai dimasukkan akan muncul pesan apakah nilai tersebut GENAP atau GANJIL.
3. Buatlah program yang pertama sekali meminta memasukkan nilai tertentu, kemudian setelah nilai dimasukkan akan muncul pesan apakah nilai tersebut NEGATIP , NOL, atau POSITIP.

- ***Selamat Belajar –***

Praktikum 5

If bersarang dan **Looping while**

Program if bersarang dengan **if else** **1**

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

void main()
{
int kode;

printf("Masukkan kode hari [1..7] :");scanf("%d",&kode);
if(kode==1)
 printf("\nIni kode hari SENIN ");
else
 if(kode==2)
   printf("\nIni kode hari SELASA ");
 else
  if(kode==3)
    printf("\nIni kode hari RABU ");
  else
   if(kode==4)
    printf("\nIni kode hari KAMIS ");
   else
    if(kode==5)
     printf("\nIni kode hari JUMAT ");
    else
     if(kode==6)
      printf("\nIni kode hari SABTU ");
     else
      if(kode==7)
        printf("\nIni kode hari MINGGU ");
      else
        printf("\nKode yang anda masukkan tidak dikenal");
getch();
}
```

Program if bersarang dengan **switch** **2**

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

void main()
{
int kode;

printf("Masukkan kode hari [1..7] :");scanf("%d",&kode);
switch (kode)
{
case 1:
printf("\nIni kode hari SENIN ");
break;
case 2:
printf("\nIni kode hari SELASA ");
break;
case 3:
printf("\nIni kode hari RABU ");
break;
case 4:
printf("\nIni kode hari KAMIS ");
break;
case 5:
printf("\nIni kode hari JUMAT ");
break;
case 6:
printf("\nIni kode hari SABTU ");
break;
case 7:
printf("\nIni kode hari MINGGU ");
break;
default:
printf("\nKode yang anda masukkan tidak dikenal");
}
getch();
}
```

Program if bersarang dengan **if else** **3**

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

void main()
{
char nama[25];
int umur;

printf("Masukkan nama anda :");scanf("%s",nama);
printf("Masukkan umur anda :");scanf("%d",&umur);
clrscr();
printf("Hai %s",nama);
if(umur<=5)
 printf("\nAnda ternyata masih BALITA");
else if(umur<17)
   printf("\nAnda masih kanak-kanak");
else if(umur<45)
    printf("\nAnda sudah dewasa");
else if(umur<=55)
     printf("\nAnda sudah cukup tua");
else if(umur<=75)
    printf("\nAnda sudah tua bangka");
else
    printf("\numur anda keterlaluan");
getch();
}
```

Program if bersarang jenis ini **tidak bisa** diimplementasikan dengan switch, sebab pernyatan setelah switch didepan case harus konstanta

Tambahkan pada program di atas dengan pengulangan **while** seperti berikut :
(Yang ditebalkan adalah bagian yang ditambahkan)

4

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

void main()
{
char nama[25];
int umur;
char lg;

lg='Y';
while(lg=='Y' || lg=='y')
{
clrscr();
printf("Masukkan nama anda :");scanf("%s",nama);
printf("Masukkan umur anda :");scanf("%d",&umur);
clrscr();
printf("Hai %s",nama);
if(umur<=5)
 printf("\nAnda ternyata masih BALITA");
else if(umur<17)
   printf("\nAnda masih kanak-kanak");
else if(umur<=45)
    printf("\nAnda sudah dewasa");
else if(umur<=55)
      printf("\nAnda sudah cukup tua");
else if(umur<=75)
    printf("\nAnda sudah tua bangka");
else
    printf("\numur anda keterlaluan");
printf("\n\nIngin mengulang lagi
[yt]?");scanf("%c",&lg);
}
getch();
}
```

Tambahkan perintah ***fflush(stdin)***; Jika perintah input *scanf()* pada bagian ini tidak bekerja

(Amati hasilnya)

Latihan :

1. Tambahan perintah pengulangan **while** seperti pada program 3 untuk program 1 dan 2 di atas !
2. Buatlah program dengan masukan berupa nilai dari 0 s/d 100 dan program menghasilkan tampilan ouput berupa nilai : A, B, C, D dan E dengan ketentuan :
   - A jika Nilai $\geq$ 80
   - B jika 65 $\leq$ Nilai < 80
   - C jika 55 $\leq$ Nilai < 65
   - D jika 40 $\leq$ Nilai < 55
   - E jika Nilai < 40

   Gunakan perintah if bersarang untuk mengimpelementasikan ketentuan di atas !

- ***Selamat Belajar –***

Praktikum 6

Menggunakan **switch** untuk membuat program Menu

Cobakan program berikut :

1

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

int main()
{
int pil;

clrscr();
printf("MENU UTAMA\n");
printf("==========\n");
printf("1. Entri Data\n");
printf("2. Edit Data\n");
printf("3. Hapus Data\n");
printf("4. Selesai\n\n");
printf("Tentukan pilihan anda :");scanf("%d",&pil);

switch(pil)
{
case 1 :
   clrscr();
   printf("Menjalankan prosedur entri\n");
   getch();
   break;
case 2 :
   clrscr();
   printf("Menjalankan prosedur edit\n");
   getch();
   break;
case 3 :
   clrscr();
   printf("Menjalankan prosedur hapus\n");
   getch();
   break;
case 4 :
  return(0);
}
return(0)
}
```

Tambahkan perintah **while** seperti pada program 2 dan amati bagaimana penggunaan perintah while pada program tersebut untuk membuat pengulangan terus-menerus.

2

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

int main()
{
int ulang,pil;


ulang=4;
do
{
   clrscr();
   printf("MENU UTAMA\n");
   printf("==========\n");
   printf("1. Entri Data\n");
   printf("2. Edit Data\n");
   printf("3. Hapus Data\n");
   printf("4. Selesai\n\n");
   printf("Tentukan pilihan anda :");scanf("%d",&pil);

   switch(pil)
   {
      case 1 :
         clrscr();
         printf("Menjalankan prosedur entri\n");
         getch();
         break;
      case 2 :
         clrscr();
         printf("Menjalankan prosedur edit\n");
         getch();
         break;
      case 3 :
         clrscr();
         printf("Menjalankan prosedur hapus\n");
         getch();
         break;
      case 4 :
        return(0);
   }
}while (ulang!=0);

return(0);
}
```

Perhatikan hal-hal berikut :

- Kenapa untuk program kali ini fungsi utama **main()** menggunakan **int main()** bukan **void main()** seperti sebelum-sebelumnya
- Amati perbedaan antara perintah pengulangan while () dan do – while() seperti yang digunakan pada program2

Selanjutnya cobakan dan amati program berikut :

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

//prosedur untuk menggerakkan kalimat dengan memanfaatkan perintah for
//prosedur gerak1 dari kiri ke kanan
void gerak1()
{
   int i,x,y,d;
   for(i=1;i<=60;i++)
   {
   clrscr();
   for(d=1;d<=500;d++){gotoxy(i,3);printf("Darma Persada");}
}
}//akhir prosedur gerak1

//prosedur untuk menggerakkan kalimat dengan memanfaatkan perintah for
//prosedur gerak2 dari kanan ke kiri
void gerak2()
{
      printf("Belum ada programnya, silahkan dibuat sendiri !");
}//akhir prosedur gerak2

//prosedur untuk menggerakkan kalimat dengan memanfaatkan perintah for
//prosedur gerak3 dari atas ke bawah
void gerak3()
{
   int i,x,y,d;
   for(i=1;i<=20;i++)
   {
   clrscr();
   for(d=1;d<=500;d++){gotoxy(3,i);printf("Darma Persada");}
}
}//akhir prosedur gerak3

//prosedur untuk menggerakkan kalimat dengan memanfaatkan perintah for
//prosedur gerak4 dari bawah ke atas
void gerak4()
{
      printf("Belum ada programnya, silahkan dibuat sendiri !");
}//akhir prosedur gerak4

//PROGRAM UTAMA
int main()
{
int ulang,pil;

ulang=4;
do
{
clrscr();
printf("MENU UTAMA\n");
printf("==========\n");
printf("1. Prosedur gerak1 kiri ke kanan\n");
printf("2. Prosedur gerak2 kanan ke kiri\n");
printf("3. Prosedur gerak3 atas ke bawah\n");
printf("4. Prosedur gerak4 bawah ke atas \n");
printf("5. Selesai\n\n");
printf("Tentukan pilihan anda :");scanf("%d",&pil);
switch(pil)
{
case 1 :
   clrscr(); gerak1();  getch();
   break;
case 2 :
   clrscr(); gerak2();  getch();
   break;
case 3 :
   clrscr(); gerak3();  getch();
   break;
case 4 :
   clrscr(); gerak4();  getch();
   break;
case 5 :
  return(0);
}
}while (ulang!=0);
return(0);
}
```

Praktikum 7

Pengolahan data **string** untuk membuat Password

Cobakan program berikut :

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


void main()
{
char username[25],password[12];


printf("User Name :");gets(username);
printf("Password  :");gets(password);

if ((strcmp(username,"daku")==0) &&
(strcmp(password,"tif05")==0))
{
printf("Username dan password OK !!!\n");
}
else
{
printf("Username Or password invalid !!!\n");
}
getch();
}
```

Note :

- Perhatikan adanya penambahan header ***<string.h>***
- Perhatikan bahwa untuk membandingkan dua string digunakan perintah ***strcmp(string1,string2),*** bukan operator "==" , jika strcmp() bernilai 0 (strcmp()==0) berarti string tersebut sama.

Jalankan program dan coba isi ***username*** dengan : **daku** dan ***password*** dengan : **tif05**. Coba juga isi dengan username dan password dengan data yang lain.

Latihan :

1. Modifikasi program di atas sehingga dapat membedakan antara kesalahan pengisian username (***username invalid***) dan kesalahan pengisian password (***password invalid***) !
2. Tambahkan kode pada program di atas sehingga jika username atau password salah akan meminta mengulangi pengisi username dan password.

Selanjutnya cobakan program berikut ini :

Program berikut ini akan memberikan tanda * saat password dimasukkan

2

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


void main()
{
char username[25],password[12],pwd[12];
char ch;
int i;

while(1) //ulangi selama belum exit atau break
 {
  clrscr();
  printf("User Name :");gets(username); //ambil data username
  printf("Password  :");
  i=-1;
  do //loooping ini berfungsi untuk mencetak tanda * saat memasukkan password
  {
    i=i+1;
    ch=getch();
    if (ch!=13)
     {
       pwd[i]=ch;
     }
    printf("*");
  }while(int(ch)!=13); //looping dihentikan jika ditekan tombol Enter(=13)

strcpy(password,pwd); //ambil password yang dimasukkan melalui keyboard (pwd)
if ((strcmp(username,"daku")==0) && (strcmp(password,"tif05")==0))
//Apakah username dan password sama ?
   {
    printf("\nUsername dan password OK !!!");
    getch();
    break;
   }
   else
   {
    printf("\n\nUsername Or password invalid !!!");
    textcolor(18);printf("\nSilahkan ULANGI !!!");
   getch();
   }
  }
}
```

Latihan :

- Modifikasi program sehingga jika password salah sebanyak tiga kali maka program akan langsung keluar !

Praktikum 8

Menggunakan variabel **Array**

Variabel array adalah variabel yang dapat menyimpan beberapa item data sekaligus. Setiap item data lokasi penyimpanannya dibedakan oleh nomor indeks dalam kurung siku []. Setiap item data pada array type dasarnya harus sama, misalnya sama-sama integer, sama-sama char, float dan seterusnya.

Dengan menggunakan array maka kebutuhan pengolahan data bertipe sama secara serentak dapat dilakukan lebih mudah dengan bantuan looping dan umumnya menggunakan looping for.

Pada praktikum kali ini akan dilakukan percobaan pengolahan data menggunakan array meliputi :

- Cara Memasukkan/meng-inputkan data ke array
- Cara mencetak data pada array
- Men-total-kan data integer pada array
- Me-rata-rata-kan data integer pada array

Lakukan percobaan dengan program-program berikut :

1

```
//****************************************
//Program Percobaan menggunakan tipe Array
//****************************************
#include <stdio.h>
#include <conio.h>


void main()
{
int a[5]; //mendeklarasikan variavel tipe array
int i;

a[0]=4;  //memasukkan data ke variabel array indek ke-0
a[1]=7;  //memasukkan data ke variabel array indek ke-1
a[2]=12; //memasukkan data ke variabel array indek ke-2
a[3]=9;  //memasukkan data ke variabel array indek ke-3
a[4]=8;  //memasukkan data ke variabel array indek ke-4

//mencetak data array dengan looping for
for(i=0;i<=4;i++)
printf("\nData pada var a indeks yang ke-%d adalah : %d",i,a[i]);

getch();

}
```

2

```
//****************************************
//Program Percobaan menggunakan tipe Array
//Memasukkan data dengan scanf
//****************************************
#include <stdio.h>
#include <conio.h>


void main()
{
int a[5]; //mendeklarasikan variavel tipe array
int i;

//Masukkan data ke var array dengan memanfaatkan looping for
for(i=0;i<=4;i++)
{
printf("Masukkan data integer untuk a[%d] :",i);scanf("%d",&a[i]);
}

//mencetak data array dengan looping for
for(i=0;i<=4;i++)
printf("\nData pada var a indeks yang ke-%d adalah : %d",i,a[i]);

getch();
```

```
//****************************************
//Program Percobaan menggunakan tipe Array
//Mentotalkan data pada array
//****************************************
#include <stdio.h>
#include <conio.h>


void main()
{
int a[5]; //mendeklarasikan variavel tipe array
int i,total;

//Masukkan data ke var array dengan memanfaatkan looping for
for(i=0;i<=4;i++)
{
printf("Masukkan data integer untuk a[%d] :",i);scanf("%d",&a[i]);
}


//Menghitung total pada array dengan looping for
total=0;
for(i=0;i<=4;i++)
{
total=total+a[i];
}

//mencetak data array dengan looping for
for(i=0;i<=4;i++)
printf("\nData pada var a indeks yang ke-%d adalah : %d",i,a[i]);

printf("\n\nTotal data tersebut adalah : %d",total);
getch();

}
```

Latihan :

1. Berdasarkan program di atas buatlah program untuk me-rata-ratakan 5 bilangan integer dalam array !
2. Berdasarkan program no. 3 di atas, buatlah program untuk me-rata-ratakan 10 buah bilangan integer !
3. Berdasarkan program di atas, buatlah program me-rata-rata-kan sebanyak N buah bilangan, dimana nilai N dimasukkan terlebih dahulu sebelum data diinputkan dengan scanf !
4. Modifikasi program 3 di atas sehingga dengan 5 buah bilangan integer pada var a menggunakan penomoran indeks 1, 2, 3, 4, 5 bukan 0, 1, 2, 3, 4.

- Selamat Bekerja -

Praktikum 9

Lanjutan : Menggunakan variabel **Array**

(Algoritma **Sorting** dengan Array**)**

Sorting berarti proses meng-urutkan. Ada dua jenis sorting yaitu ASCending (menaik) : urutan dari kecil ke besar, dan DESCending(menurun) : urutan dari besar ke kecil.

Ada beberapa algoritma meng-urutkan, pada percobaan kali akan diberikan program sorting paling sederhana yang sering disebut dengan *bubble sort*. Perbedaan algoritma dalam hal ini bukan menentukan hasil sorting-nya tapi lebih menentukan efektivitas proses sorting apakah butuh waktu lama atau tidak. Algoritima sorting yang lain akan diberikan pada mata kuliah : struktur data.

Pada percobaan kali ini yang paling utama untuk diperhatikan peserta praktikan adalah bagaimana variabel array digunakan untuk proses sorting tersebut.

Selanjutnya lakukan percobaan dengan program-program berikut :

```
//****************************************
//Program Percobaan menggunakan tipe Array
//Algoritma Sorting
//****************************************
#include <stdio.h>
#include <conio.h>

void main()
{
const N=10; //membuat konstanta, maksudnya N tidak boleh berubah(tetap=constant)
int a[N]; //mendeklarasikan variavel tipe array
int i,j,T;

//Masukkan data ke var array  looping for
for(i=0;i<=N-1;i++)
{
printf("Masukkan data integer untuk a[%d] :",i);scanf("%d",&a[i]);
}

//Data pada array Sebelum Sorting
clrscr();
printf("\nData pada var a sebelum disorting :");
printf("\n==================================");
for(i=0;i<=N-1;i++)
{
printf("\nData pada var a[%d] : %d",i,a[i]);
}
printf("\n\nEnter dulu dong...!");getch();

//proses sorting
//Perhatikan begaimana digunakan for dalam for (for bersarang = nested for)
for(i=0;i<=N-2;i++)
{
 for(j=i+1;j<=N-1;j++)
  {
   if (a[i]<=a[j])
    {
     T=a[i];
     a[i]=a[j];
     a[j]=T;
    }
  }
}

//Data pada array Setelah Sorting
printf("\n\nData pada var a setelah disorting :");
printf("\n==================================");
for(i=0;i<=N-1;i++)
{
printf("\nData pada var a[%d] : %d",i,a[i]);
}
printf("\n\nBandingkan dengan sebelumnya...!");getch();

}//akhir dari main()
```

Latihan :

- Modifikasi program di atas untuk meng-ururtkan (sorting) secara Ascending
- Buatlah program untuk meng-urutkan data tipe char !
- Buatlah program untuk meng-urutkan data tipe string !

- Selamat Berlatih -

OVERLOADING FUNCTION
Bab 4

4.1 PENGERTIAN

Overloading function dalah beberapa fungsi dapat memiliki argument berbeda tetapi namanya sama.

Contoh program :

```
# include <iostream.h>
Void tulis (char c)
{
Cout << “karakter : “ << c << endl ;
}
Void tulis (long x)
{
Cout << “bilangan bulat =” << x << endl ;
}
Void main ( )
{
Tulis ( ‘A’ ) ;
Tulis (1805) ;
}
```

Keterangan :

- walaupun 2 fungsi diatas mempunyai nama yang sama, tetapi compiler tahu fungsi manakah yang harus dipanggil.

C++ hanya dapat mengkompilasi overloading function jika argument fungsinya berbeda. Bila argument fungsinya sama tetapi tipe nilai kembaliannya berbeda, compiler akan melaporkan adanya kesalahan.

Contoh :

Fungsi : **double coba (int) ;** *dengan* Fungsi : **int coba (int) ;**

4.2 DEFAULT ARGUMEN

Contoh :

Void func (int x = 0, int y = 0) ;

Maksudnya adalah :

Contoh fungsi diatas dapat dipanggil dengan 3 (tiga) cara berbeda :
1. func ( ) *// nilai x dan y otomatis 0*
2. func (7) *// nilai x = 7 dan y = 0*
3. func (7, 33) *// nilai x = 7 dan y = 33*

Contoh deklarasi fungsi diatas akan terjadi kesalahan, jika variable x diberi nilai, sedangkan variable y tidak beri nilai.

Contoh :

Void func (**int x = 2, int y**) ;

4.3 AMBIGUITY PADA OVERLOADING FUNCTION

Beberapa hal yang menimbulkan ketidakjelasan pada program :
1. type conversion
Contoh :

```
Void f ( double nilai )
Main ( )
{
Int I = 12 ;
F ( I ) ;
}
```

2. reference parameter
Parameter yang memiliki perbedaan hanya pada argument, dimana fungsi yang satu menggunakan reference parameter dan yang lainnya menggunakan value parameter.
Contoh :

Void func **(int x , int y);** *dengan* void func func **(int x , int& y) ;**

3. default argumen

4.4 TATA CARA PENULISAN PROGRAM

1. Jangan menggunakan nama identifier yang terlalu singkat.
2. Identifier yang tidak melambangkan sesuatu dapat ditulis dengan satu huruf saja.
3. hindari pendeklarasian variable yang bertipe sama dalam satu baris, contoh : **int a, b, c** ;
4. Tambahkan satu spasi setelah tanda koma atau titik dua atau titik koma, jangan sebelumnya.

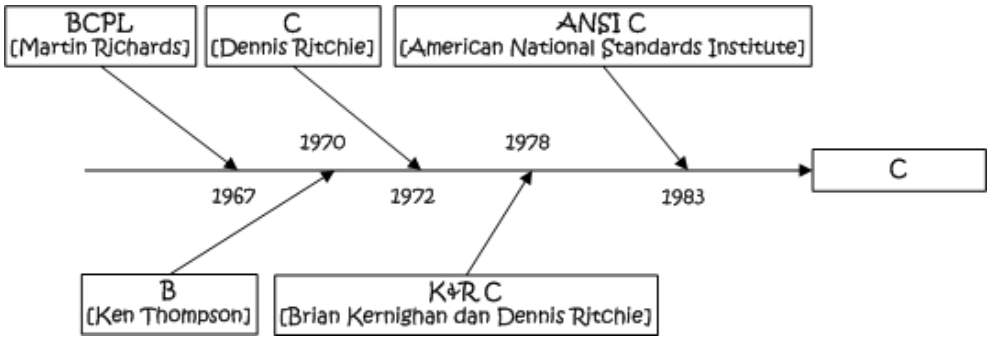
5. Jangan tambahkan spasi sesudah operator unary : ++ , - - , dsb.
   Contoh : **a ++**
6. Jangan tambahkan spasi setelah tanda kurung buka dan kurung tutup.
7. Jangan gunakan spasi dipenulisan index, contoh : **x[10]** *atau* **x [10]**
8. Hindari penulisan secara horizontal.
   Contoh :

```
Void main ( )
{
Tulis ( 'A' ) ; Tulis (1805) ;
}
```

Bab 1
Pendahuluan

1.1. Sejarah Bahasa C

Sejarah perkembangan dan latar belakang munculnya bahasa C adalah seperti dalam Gambar 1.



Gambar 1: Sejarah Bahasa C

Boleh dikatakan bahwa akar dari bahasa C adalah bahasa BCPL yang dikembangkan oleh *Martin Richards* pada tahun 1967. Kemudian berdasar pada bahasa BCPL ini *Ken Thompson* yang bekerja di Bell Telephone Laboratories (Bell Labs) mengembangkan bahasa B pada tahun 1970. Saat itu bahasa B telah berhasil diimplementasikan di komputer DEC PDP-7 dengan operating system (OS) UNIX. Pada tahun 1972, peneliti lain di Bell Labs bernama *Dennis Ritchie* menyempurnakannya menjadi bahasa C.

Gambar 2: Tokoh Yang Berjasa Dalam Pengembangan Bahasa C

Pada tahun 1978, *Dennis Ritchie* bersama dengan *Brian Kernighan* mempublikasikan buku yang kemudian menjadi legenda dalam sejarah perkembangan bahasa C, yang berjudul *The C Programming Language*. Buku ini diterbitkan oleh Prentice Hall, dan pada saat ini telah diterjemahkan dalam berbagai bahasa di dunia. Boleh dikatakan bahwa buku ini adalah buku yang paling banyak direfer orang dan dijadikan buku panduan tentang pemrograman bahasa C sampai saat ini. Teknik dan gaya penulisan bahasa C yang merefer kepada buku ini kemudian terkenal dengan sebutan *K&R C* atau *Classic C* atau *Common C*.

Seiring dengan berkembang pesatnya bahasa C, banyak vendor mengembangkan kompiler C menurut versi masing-masing. Hal ini menggerakkan ANSI (*American National Standards Institute*) pada tahun 1983 untuk membuat suatu komite yang kemudian diberi nama *X3J11*, yang betujuan untuk membuat definisi standar bahasa C yang lebih modern dan komprehensif, dengan memperbaiki *syntax* dan *grammar* bahasa C. Usaha ini berhasil diselesaikan 5 tahun kemudian, yaitu ditandai dengan lahirnya standard ANSI untuk bahasa C yang kemudian terkenal dengan sebutan *ANSI C* pada tahun 1988.

1.2. Mengapa Memakai Bahasa C

Sampai saat ini, bahasa C telah berhasil digunakan untuk mengembangkan berbagai jenis permasalahan pemrograman, dari level operating system (unix, linux, ms dos, dsb), aplikasi perkantoran (text editor, word processor, spreadsheet, dsb), bahkan sampai pengembangan sistem pakar (*expert system*). Kompiler C juga telah tersedia di semua jenis platform komputer, mulai dari Macintosh, UNIX, PC, Micro PC, sampai super komputer.

C bisa disebut bahasa pemrograman tingkat menengah (*middle level programming language*). Arti tingkat (level) disini adalah kemampuan mengakses fungsi-fungsi dan perintah-perintah dasar bahasa mesin/hardware (*machine basic instruction set*). Semakin tinggi tingkat bahasa pemrograman (misalnya: java), semakin mudahlah bahasa pemrograman dipahami manusia, namun membawa pengaruh semakin berkurang kemampuan untuk mengakses langsung instruksi dasar bahasa mesin. Demikian juga sebaliknya dengan bahasa pemrograman tingkat rendah (misalnya: assembler), yang

semakin sulit dipahami manusia dan hanya berisi perintah untuk mengakses bahasa mesin. Dalam perspektif mudahnya dipahami manusia, C bisa digolongkan dalam bahasa tingkat tinggi, namun C juga menyediakan kemampuan yang ada pada bahasa tingkat rendah, misalnya operasi bit, operasi byte, pengaksesan memori, dsb.

Beberapa alasan mengapa memakai bahasa C adalah terangkum dibawah.

- *C adalah bahasa pemrograman yang paling populer saat ini*
  Dengan banyaknya programmer bahasa C, membawa pengaruh semakin mudahnya kita menemukan pemecahan masalah yang kita dapatkan ketika menulis program dalam bahasa C. Pengaruh positif lain adalah semakin banyaknya kompiler yang dikembangkan untuk berbagai platform (berpengaruh ke portabilitas).

- *C adalah bahasa pemrograman yang memiliki portabilitas tinggi*
  Program C yang kita tulis untuk satu jenis platform, bisa kita kompile dan jalankan di platform lain dengan tanpa ataupun hanya sedikit perubahan. Ini bisa diwujudkan dengan adanya standarisasi ANSI untuk C.

- *C adalah bahasa pemrograman dengan kata kunci (keyword) sedikit*
  Kata kunci disini adalah merupakan fungsi ataupun kata dasar yang disediakan oleh kompiler suatu bahasa pemrograman. Hal ini membawa pengaruh semakin mudahnya kita menulis program dengan C. Pengaruh lain dari sedikitnya kata kunci ini adalah proses eksekusi program C yang sangat cepat. C hanya menyediakan 32 kata kunci seperti terangkum dibawah:

auto	**break**	**case**	**char**	**const**	**continue**	**default**
do	**double**	**else**	**enum**	**extern**	**float**	**for**
goto	**if**	**int**	**long**	**register**	**return**	**short**
signed	**sizeof**	**static**	**struct**	**switch**	**typedef**	**union**
unsigned	**void**	**volatile**	**while**			

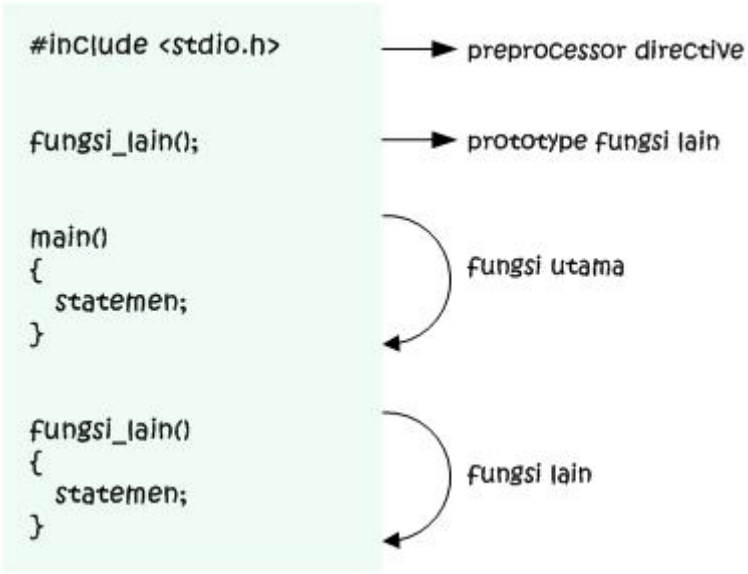
- *C adalah bahasa pemrograman yang fleksibel*
  Dengan menguasai bahasa C, kita bisa menulis dan mengembangkan berbagai jenis program mulai dari operating system, word processor, graphic processor, spreadsheets, ataupun kompiler untuk suatu bahasa pemrograman.

- *C adalah bahasa pemrograman yang bersifat moduler*
  Program C ditulis dalam *routine* yang biasa dipanggil dengan fungsi. Fungsi-fungsi yang telah kita buat, bisa kita gunakan kembali (*reuse*) dalam program ataupun aplikasi lain.

1.3. Struktur Program Bahasa C

Program bahasa C adalah suatu program terdiri dari satu atau lebih fungsi-fungsi. Fungsi utama dan harus ada pada program C yang kita buat adalah fungsi **main()**. Fungsi main() ini adalah fungsi pertama yang akan diproses pada saat program di kompile dan dijalankan, sehingga bisa disebut sebagai fungsi yang mengontrol fungsi-fungsi lain. Karena struktur program C terdiri dari fungsi-fungsi lain sebagai program bagian (*subroutine*), maka bahasa C biasa disebut sebagai *bahasa pemrograman terstruktur*. Cara penulisan fungsi pada program bahasa C adalah dengan memberi nama fungsi dan kemudian dibuka dengan kurang kurawal buka (**{**) dan ditutup dengan kurung kurawal tutup (**}**).

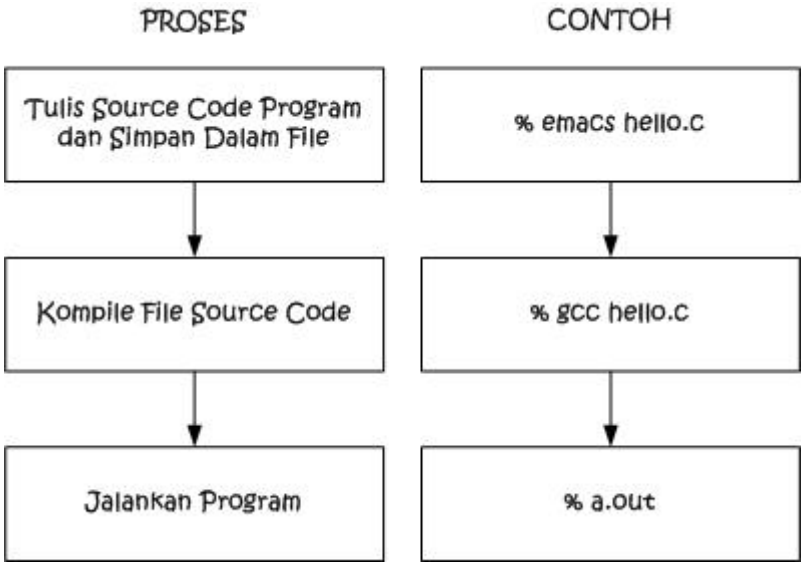
Fungsi-fungsi lain selain fungsi utama bisa dituliskan setelah atau sebelum fungsi utama dengan deskripsi prototype fungsi pada bagian awal program. Bisa juga dituliskan pada file lain yang apabila kita ingin memakai atau memanggil fungsi dalam file lain tersebut, kita harus menuliskan header filenya, dengan *preprocessor directive* **#include**. File ini disebut file pustaka (*library file*). Untuk lebih jelas tentang struktur program bahasa C ini, silakan melihat pada Gambar 3 dibawah.



Gambar 3: Struktur Dasar Program C

1.4. Proses Pembuatan Program C

Proses pembuatan program dengan menggunakan bahasa C adalah seperti dalam gambar 4.



Gambar 4: Proses Pembuatan Program C

1. **Tulis source code program bahasa C dengan menggunakan text editor, kemudian simpan dalam sebuah file.**
   Text editor disini bisa merupakan aplikasi notepad atau editplus pada windows, untuk operating system unix/linux kita bisa menggunakan aplikasi emacs yang cukup terkenal.

2. **Kompile file yang berisi source code program bahasa C.**
   Kompilasi atau kompile (compile) adalah suatu proses merubah source code ke bahasa mesin sehingga bisa dieksekusi (executable) atau dijalankan. Banyak sekali kompiler bahasa C yang ada saat ini, baik yang gratis maupun yang kita harus membeli untuk menggunakannya. Untuk memudahkan proses belajar bahasa C, penulis memberikan rekomendasi untuk menginstall Cygwin (*www.cygwin.com*) bagi yang menggunakan operating system Windows. Cygwin adalah satu set free software yang dikembangkan oleh Redhat, yang berisi koleksi aplikasi dan tools UNIX yang didesain khusus untuk bisa dijalankan di Windows. Kebutuhan akan kompiler (GCC, GNU C Compiler) dan shell (Bash Shell) untuk membuat program C bisa kita dapatkan dengan menginstall Cygwin ini.

3. **Jalankan program yang telah dikompile.**
   Setelah kita kompile file yang berisi source code, maka sebagai hasil kompilasi tersebut kita akan mendapatkan suatu file yang bisa dijalankan (*executable file*). Menjalankan program yang kita buat berarti menjalankan file hasil proses kompilasi tersebut.

SEJARAH PENGEMBANGAN COBOL

Fatur
artur@gmail.com

Materi Pembahasan :

1. Sejarah perkembangan bhs C
2. Pengenalan Program C
3. Type data, Konstanta, Variabel, & Operator pada C
4. Latihan membuat program sederhana menggunakan C

Tujuan Praktikum

Praktikan mengetahui sejarah perkembangan bahasa C

Praktikan mengerti struktur program bahasa C

Praktikan mengerti konsep tipe data dalam bahasa C

Praktikan dapat membuat program sederhana dengan menggunakan bahasa C

PEMBAHASAN

1 Sejarah Perkembangan Bahasa C

Berasal dari bahasa BCPL (Basic Combined Programming Language) oleh MARTIN RICHARD, Cambridge tahun 1967, KEN THOMPSON membuat bahasa B untuk dipakai pada komputer DEC PDP-7 dibawah sistem operasi UNIX pada Bell laboratory, Murray Hill, New Jersey tahun 1970.

Bahasa B merupakan suatu bahasa pemrograman yang tidak memiliki jenis suatu data seperti halnya PL/M. Berdasarkan gambaran bahasa B, DENNIS RITCHIE menulis bahasa C. Nama C diambil berdasarkan urutan sesudah B dari bahasa BCPL. Tujuan bahasa C pada mulanya untuk

membentuk suatu sistem operasi yang akan digunakan pada mesin komputer DEC PDP-11 yang baru.

Pada tahun 1975, sistem operasi UNIX versi 6 dan bahasa C mulai diberikan kepada Universitas maupun Akademi. Dan pada tahun 1979, sistem operasi UNIX versi 7 dikeluarkan dengan bahasa C. Sistem operasi ini(versi 7) seluruhnya ditulis dalam bahasa C.

Pada 1978 Dennis Ritchie dan Brian Kernighan kemudian mempublikasikan buku *The C Programming Language* yang semakin memperluas pemakaiannya dan dijadikan standar oleh *ANSI (American National Standard Institute)* pada tahun 1989. C kemudian dikembangkan lagi oleh Bjarne Stroustrup menjadi C++ (1986). C dan/atau C++ banyak digunakan (sehingga menjadi 'standar') sebagai bahasa pemrograman untuk membuat sistem operasi.

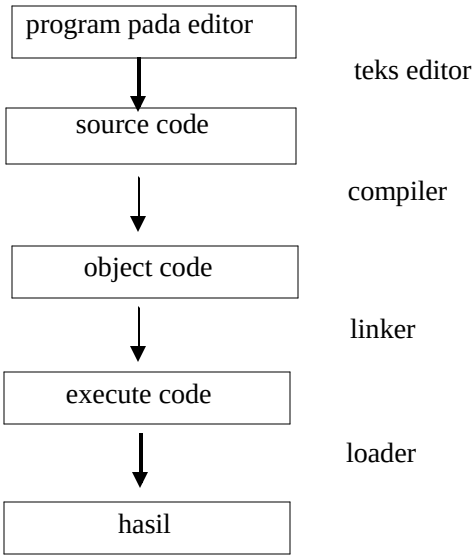
2 Pengenalan Program C

2.1 Proses Penerjemahan bahasa C

Untuk dapat dimengerti oleh komputer, bahasa C yang ditulis harus diterjemahkan terlebih dahulu ke dalam bentuk yang dikenal oleh bahasa mesin. Ada dua jenis translator atau penerjemah yang digunakan dalam bahasa C, yaitu interpreter dan compiler.

Interpreter merupakan suatu translator yang menerjemahkan bahasa C ke dalam bahasa yang dikenal mesin satu persatu, dan hasil terjemahan langsung dikerjakan. Sedangkan pada compiler, diterjemahkan secara keseluruhan dan hasil terjemahan tersebut disimpan dalam bentuk kode mesin (object code), dan kode eksekusi(execute code).

Object code dari compiler baru dapat dilaksanakan setelah object code tersebut diubah menjadi execute code oleh linker. Secara lengkap prosesnya adalah sebagai berikut :



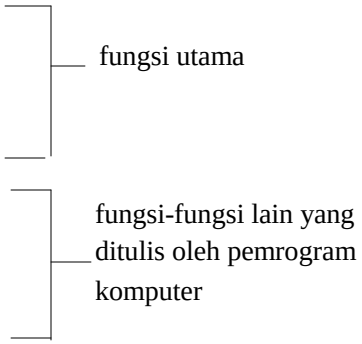
Gambar 2.1 Proses penerjemahan bahasa C

2.2 Struktur Program C

Untuk dapat memahami bagaimana suatu program ditulis, maka struktur dari program harus dimengerti terlebih dahulu. Tiap bahasa komputer mempunyai struktur program yang berbeda. Jika struktur dari program tidak diketahui, maka akan sulit bagi pemula untuk memulai menulis suatu program dengan bahasa yang bersangkutan.

Struktur dari program C terdiri dari koleksi satu / lebih fungsi-fungsi. Fungsi pertama yang harus ada di program C sudah ditentukan namanya, yaitu bernama ***main()***. Suatu fungsi didalam program C dibuka dengan kurung kurawal buka ({) dan ditutup dengan kurung kurawal tutup (}). Diantara kurung kurawal dapat dituliskan statemen-statemen program C dan pada setiap statemen diakhiri dengan tanda titik koma (;). Berikut adalah struktur dari program C :

```
main()
{
        statement-statement;
}

Fungsi_Fungsi_Lain()
 {
        statemen-statemen ;
}
```



Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian (subroutine). Fungsi-fungsi selain fungsi utama merupakan program-program bagian. Fungsi-fungsi ini dapat ditulis setelah fungsi utama atau diletakkan pada file pustaka dan akan dipakai di suatu program, maka nama judulnya (header file) harus dilibatkan di dalam program yang menggunakan *preprocessor directive* **#include**.

Header file merupakan file yang berisi dengan prototype (judul, nama, sintak) dari sekumpulan fungsi-fungsi pustaka tertentu. Jadi file ini hanya berisi dengan prototype dari fungsi-fungsi pustaka, sedangkan fungsi-fungsi pustakanya sendiri disimpan dalam file pustaka (library file dengan nama extension file-nya adalah .LIB) Misalnya prototype dari fungsi-fungsi pustaka printf() dan scanf() terdapat di file stdio.h, sehingga jika fungsi-fungsi ini digunakan di program, maka nama file judulnya harus dilibatkan dengan menggunakan preposcessor #include. File judul stdio.h berisi prototype fungsi-fungsi pustaka untuk operasi input dan output standar. Ada dua cara melibatkan file judul disuatu program C, yaitu :

```
        #include<stdio.h>
atau
        #include "stdio.h"
```

File judul selain berisi dengan prototype dari fungsi-fungsi pustaka, juga umumnya berisi dengan konstanta-konstanta terdefinisi dan makro-makro. Misalnya nama konstanta terdefinisi M_PI telah didefinisikan di file judul math.h oleh Turbo-C. Selanjutnya untuk menggunakan nilai phi, nama konstanta M_PI dapat digunakan yang telah berisi dengan nilai konstanta 3.14159265358979323846.

2.3 Program C Sederhana

Berikut merupakan contoh program sederhana menggunakan bahasa C :

```
/* Program C yang sederhana */
#include <stdio.h>
main()
{
        float Celcius, Fahrenheit;
        printf("masukkan nilai celcius?";
        scanf("%f",&Celcius);
        Fahrenheit = Celcius * 1.8 + 3.2;
        printf("%f celcius adalah %f fahrenheit \n", Celcius, Fahrenheit);
}
```

Output dari listing program diatas adalah :
masukkan nilai celcius?10
10.000000 celcius adalah 50.000000 fahrenheit

Pembahasan program secara singkat :

1. Untuk memberi keterangan program, suatu komentar bebas dapat diletakkan dimanapun di program C. Komentar atau keterangan program diawali dengan bentuk /* dan diakhiri dengan bentuk */ yang terlihat sebagai berikut :
   /* Program C yang sederhana */

2. Karena program ini menggunakan fungsi-fungsi pustaka printf() dan scanf() yang disediakan oleh C dan diletakkan di file pustaka, maka nama file yang berisi prototype dari fungsi-fungsi ini (file ini disebut header file) harus disebutkan dengan prepocessor directive #include. File judul (header file) untuk fungsi-fungsi printf() dan scanf() adalah dengan nama stdio.h. Nama extention file .h menunjukkan duatu header file.

3. Di dalam fungsi utama, digunakan lima buah statemen sebagai berikut :

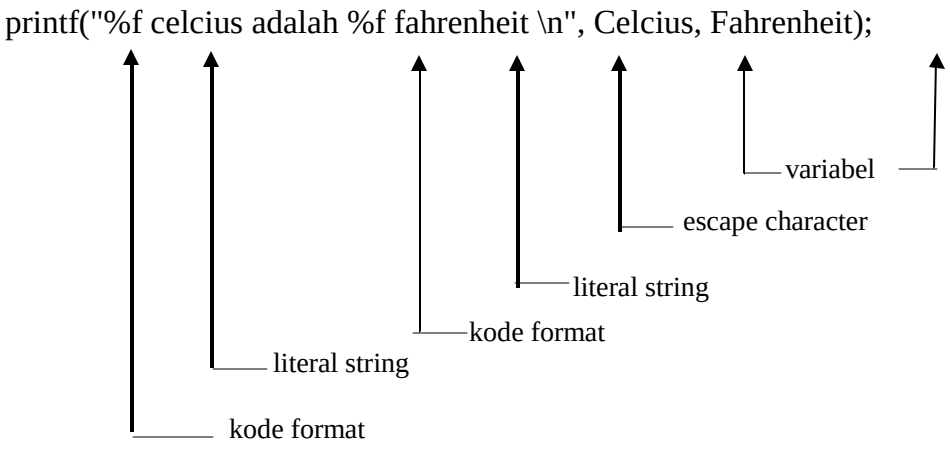
```
float Celcius, Fahrenheit;
printf("masukkan nilai celcius?";
scanf("%f",&Celcius);
Fahrenheit = Celcius * 1.8 + 3.2;
printf("%f celcius adalah %f fahrenheit \n", Celcius, Fahrenheit);
```

   Masing-masing statemen ditulis dengan diakhiri oleh titik koma. Statemen statemen (pernyataan-pernyataan) di program C dapat dibentuk dari kata-kata kunci (seperti misalnya float, int, for atau if), dibentuk dari fungsi-fungsi (misalnya printf(), scanf()) atau dibentuk dari suatu pengerjaan (seperti perhitungan, memindahkan suatu nilai ke suatu variabel)

4. Semua variabel yang digunakan di program C harus dideklarasikan terlebih dahulu. Deklarasi ini mempunyai maksud memberitahukan kepada C tipe dari variabel-variabel yang akan digunakan. Di contoh program ini digunakan dua buah variabel, yaitu Celcius dan Fahrenheit. Variabel-variabel ini dimaksudkan untuk dapat menyimpan nilai numerik pecahan (floating-point), sehingga harus dideklarasikan sebagai tipe float, sebagai berikut :

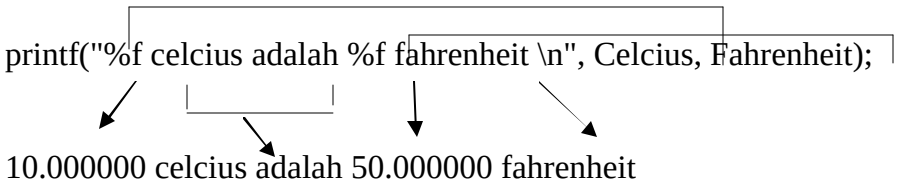
```
float Celcius, Fahrenheit;
```

5. Salah satu cara untuk menampilkan hasil di layar adalah dengan menggunakan statemen yang dibentuk dari fungsi standar printf(). Statemen terakhir di contoh program adalah statemen untuk menampilkan hasil di layar sebagai berikut :



- Kode format
  Menunjukkan format dari variabel yang akan ditampilkan. Dalam contoh terdapat dua variabel yang akan ditampilkan nilainya, sehingga digunakan dua buah kode format, satu untuk variabel Celcius dan satu lagi untuk variabel Fahrenheit. Kode format "%f" menunjukkan tipe dari variabelnya adalah numerik pecahan (floating point). Jika tipe variabelnya adalah numerik integer (nilai bulat yang dihasilkan dengan kata kunci int).
- Literal string
  Adalah suatu konstanta string yang mempunyai bentuk yang tetap dan akan ditampilkan sesuai dengan apa yang ditulis.
- Escpae character
  Merupakan suatu konstanta karakter yang ditulis dengan diawali oleh garis miring.

  Hasil dari statement yang dibentuk dari fungsi printf() ini adalah sebagai berikut:



6. Salah satu cara untuk memasukkan data dari keyboard adalah dengan menggunakan fungsi pustaka scanf() sebagai berikut :

   scanf("%f",&Celcius);

   Di fungsi ini, yang ditulis iantara tanda petik dua adalah kode format dan yang ditulis diluar tanda petik dua adalah variabel yang akan digunakan untuk menerima nilai yang diketikkan dari keyboard. Untuk fungsi scanf(), nama variabelnya harus ditulis dengan diawali variabel pointer &, sehingga pada contoh untuk variabel Celcius ditulis menjadi &Celcius.

3 Tipe Data , Konstanta, Variabel, & Operator

3.1 Tipe Data

Bahasa C menyediakan lima macam tipe data dasar, yaitu tipe data integer (nilai numerik bulat yang dideklarasikan dengan int), floating-point (nilai numerik pecahan ketepatan tunggal yang

dideklarasikan dengan float), double-precision (nilai numerik pecahan ketepatan ganda yang dideklarasikan dengan double), karakter (dideklarasikan dengan char), dan kosong (dideklarasikan dengan void). Int, float, double dan char dapat dikombinasikan dengan pengubah (modifier) signed, unsigned, long, dan short. Hasil dari kombinasi tersebut diantaranya dapat dilihat pada tabel berikut :

Tipe data	length	range
unsigned char	8 bit	0 s/d 255
char	8 bit	-128 s/d 127
enum	16 bit	-32,768 s/d 32,767
unsigned int	16 bit	0 s/d 65,535
short int	16 bit	-32,768 s/d 32,767
int	16 bit	-32,768 s/d 32,767
unsigned long	16 bit	0 s/d 4,294,967,295
long	32 bit	-2,147,483,648 s/d 2,147,483,647
float	32 bit	3.4 * (10**-38) s/d 3.4 * (10**+38)
double	32 bit	1.7 * (10**-308) s/d 1.7 * (10**+308)
long double	32 bit	3.4 * (10**-4932) to 1.1 * (10**+4932)

3.1.1 Tipe Data Numerik Integer

Karakteristik dari nilai numerik integer adalah sebagai berikut :

a. Nilai numerik pecahan yang disimpan di tempat nilai integer akan dibulatkan ke bawah. Misalkan nilai 34.56 disimpan dan ditampilkan sebagai nilai 34 untuk variabel numerik integer.

b. Nilai numerik integer negatif disimpan di memori dengan cara komplemen dua. Misal nilai numerik 7 akan disimpan di memori untuk tipe numerik integer 16 bit dalam bentuk nilai binary 111 dengan perhitungan :

$1X2_2$ = 1 x 4 = 4
$1X2_1$ = 1 x 2 = 2
$1X2_0$ = 1 x 1 = 1
---------- +
7

c. Nilai variabel yang melebihi jangkauannya akan dipotong sepanjang jumlah bit yang tersedia.

3.1.2 Tipe Data Karakter

Variabel tipe karakter dideklarasikan dengan tipe char atau signed char atau unsigned char dan dimaksudkan untuk menampung nilai sebuah huruf (karakter). Kode format "%c" digunakan untuk menampilkan nilai tipe char. Karakter yang disimpan dimemori dengan deklarasi tipe char menempati posisi 1 byte (8 bit) yang diwakili kode ASCII. Misal karakter 'A' dalam memori akan diwakilkan dengan nilai biner 01000001. Nilai binari itu dalam desimal bernilai 65.

3.1.3 Tipe Data String

Bahasa C tidak menyediakan deklarasi variabel untuk tipe string. Nilai string adalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk larik dimensi satu. Larik dimensi satu dideklarasikan dengan menyebutkan jumlah dari elemnnya yang ditulis diantara operator '[]' (bahasa C menganggap '[]' sebagai operator. Kode format untuk menampilkan nilai strng adalah "%s".

3.2 Konstanta

Konstanta adalah suatu nilai yang tidak berubah selama proses dari program. Misalnya suatu statemen ungkapan sebagai berikut :

Fahrenheit=Celcius*1.8+32;

Celcius dan Fahrenheit adalah variabel yang nilainya dapat berubah selama proses program. Nilai Celcius dapat berubah tergantung dari nilai yang dimasukkan sebagai input data dan nilai Fahrenheit akan berubah tergantung nilai dari Celcius. Nilai 1.8 dan 32 sebaliknya tidak akan pernah berubah di dalam proses program, karena nilai-nilai ini adalah nilai-nilai konstanta.

3.2.1 Konstanta Numerik Integer

Konstanta numerik integer merupakan nilai numerik bilangan bulat. Konstanta-konstanta inetegr dapat ditulis dalam bentuk desimal, heksadesimal, maupun oktal. Berikut contoh dalam bentuk nilai desimal :

123	(integer bertanda)
-123	(integer bertanda)
37000U	(integer tidak bertanda)
-75000L	(integer panjang bertanda)
4185988210UL	(integer panjang tidak bertanda)

Bila user ingin suatu konstan diinterpretasikan sebagai oktal,tambahkan digit 0 didepan.

Contoh :

015 ekivalen dengan 13 dalam decimal

Jika yang diinginkan bilangan hexadesimal, tambahkan 0x di depan.

Contoh :

0x20 ekivalen dengan 32

3.2.2 Konstanta Numerik Pecahan

Merupakan nilai numerik yang dapat mempunyai nilai pecahan dibelakang titik desimal. Konstanta numerik pecahan juga dapat ditulis dengan notasi saintifik (dengan notasi e atau E). Berikut contoh-contoh konstanta numerik pecahan :

123.
-123.00
12.34e29

3.2.3 Konstanta Karakter dan Konstanta String

Konstanta karakter merupakan nilai sebuah karakter yang ditulis diantara tanda petik tunggal. Konstanta string merupakan nilai sebuah atau lebih karakter yang ditulis dalam tanda petik ganda. Pengertian karakter adalah semua karakter yang sah seperti yang didefinisikan oleh kode ASCII, yaitu dapat meliputi huruf(a samapi z), digit (0 samapi 9), karakter-karakter khusus(misal +=$). Berikut contoh dari konstanta-konstanta karakter dan string :

'a'	konstanta karakter huruf a bernilai ASCII 97
'7'	konstanta karakter huruf a bernilai ASCII 55
"a"	konstanta string huruf a

3.2.4 Konstanta Karakter Escape

Konstanta ini banyak digunakan distatemen-statemen untuk menampilkan hasil, misal membuat kursor kemablike kolom pertama(carriage return).

Karakter	Keterangan
\n	Pindah kursor ke baris baru
\t	Horisontal tab, pindah kursor ke posisi tab
\a	Membunyikan bel sistem atau beep
\\	Mencetak atau menampilkan tanda \ (backslash)
\"	Mencetak atau menampilkan "

3.3 Operator

Operator atau tanda operasi adalah suatu tanda atau simbol yang digunakan untuk suatu operasi tertentu. Bahasa C menyediakan banyak sekali tanda-tanda operasi, yaitu operator pengerjaan, operator aritmatika, operator tipe, operator hubungan, operator logika, dan lain-lain.

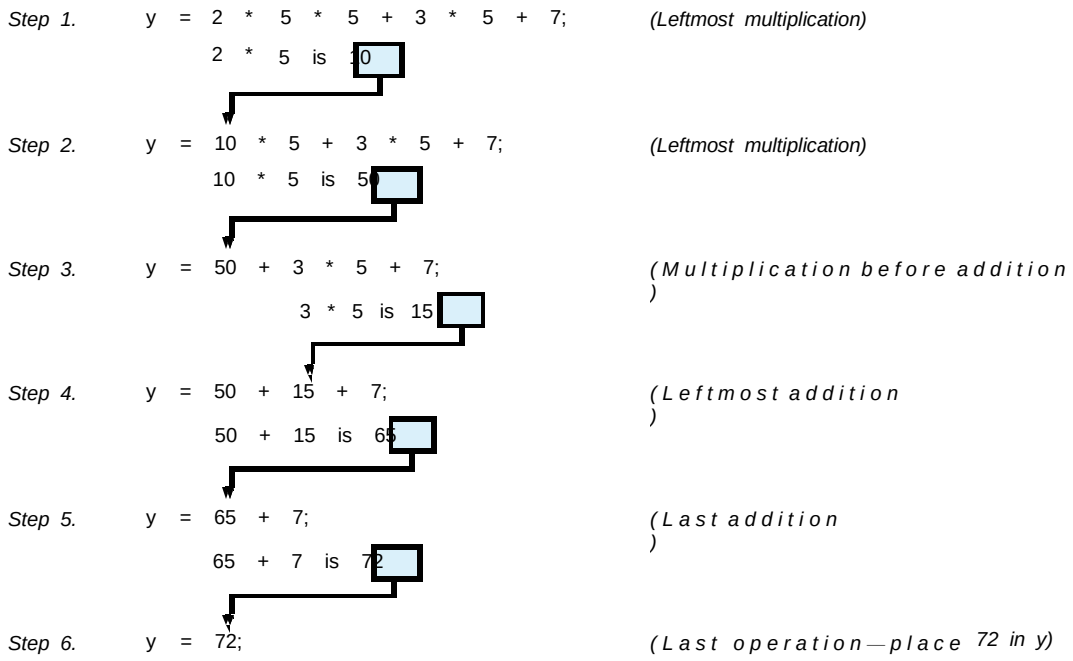
3.3.1 Operator Aritmatika

Operator	Deskripsi	Contoh	Hasil
*	Perkalian	2*3	6
/	Pembagian	9/3	3
%	Pembagian Modulus (pembagian sisa)	2+3	5
-	Pengurangan	3-2	1
+	Pertambahan	10%3	1

Prioritas operator aritmatika

Operator	Operasi	Keterangan
( )	Parenthesis	Operasi didalam tanda ( ) akan dikerjakan terlebih dahulu. Apabila bersarang, yang dikerjakan pertama kali adalah tanda ( ) yang paling dalam. Apabila terdapat tanda ( )

		satu level (tidak bersarang) dikerjakan dari kiri sampai kanan.
*, /, %	Kali, bagi, modulo	Dievaluasi kedua setelah parenthesis. Apabila terdapat beberapa, dieksekusi dari kiri dan kanan.
-, +	Tambah, kurang	Dievaluasi terakhir. Apabila terdapat beberapa, dieksekusi dari kiri dan kanan.



3.3.2 Operator Pengerjaan

digunakan untuk memindahkan nilai dari suatu ungkapan ke suatu pengenal.

Operator	Contoh	Keterangan
=	A=B+C	Mengerjakan B+C ke A
+=	A+=1	A=A+1
-+	A-+B	A=A+B
*=	A*=B	A=A*B
/=	A/=B	A=A/B
%=	A%=B	A=A%B

Contoh :

Misalnya variabel I dan J adalah variabel-variabel type integer dengan nilai awal keduanya adalah 10. Maka

I+=3; artinya : I=I+3;

I=10+3;

I=13;

I/=J-8; artinya : I=I/(J-8);

I=10/(10-8);

I=5;

3.3.3 Operator Hubungan

Digunakan untuk menunjukkan hubungan antara dua buah operand

Operator	Keterangan
<	Lebih kecil dari
<=	Lebih kecil atau sama dengan
>	Lebih besar dari
>=	Lebih besar atau sama dengan
==	Sama dengan
!=	Tidak sama dengan

Operator hubungan banyak digunakan untuk penyeleksian kondisi dengan statemen if, do-while atau while.

3.3.4 Operator Logika

Digunakan untuk membandingkan logika hasil dari operator-operator hubungan.

Hubungan Logika

X	Y	X&&Y	X||Y	!X
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

3.3.5 Operator Unary

Operator	Keterangan
-	Unary minus
++	Peningkatan dengan penambahan nilai 1
--	Penurunan dengan pengurangan nilai 1
(type)	Cast
sizeof	Ukuran dari operand dalam byte
!	Unary NOT
&	Menghasilkan alamat memory operand
*	Menghasilkan nilai pengenal dialamatnya
~	Komplemen satu

Catatan :

- Operator Unary minus

  Digunakan untuk memberi nilai minus suatu nilai numerik (bukan untuk pengurangan). Misalnya ungkapan A+-B*C akan dartikan sebagai A+(-B)*C. Penulisan A--B*C berarti A-(-B)*C

- Operator ++ dan --

  Operator ini banyak dijumpai di statemen perulangan yang berbentuk sebagai berikut :

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

  dapat ditulis dengan operator ++ sebagai berikut :

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

  Operator ++ dan -- yang digunakan di suatu ungkapan akan berarti lain jika ditulis sebelum atau sesudah operandnya.

  Contoh :

  X=5;
  A=X++;
  Hasil
  X=6 dan A=5

  X=5;
  A=++X;
  Hasil
  X=6 dan A=6

4 Latihan sederhana menggunakan C

a. Contoh 1

```
1  /* Fig. 2.4: fig02_04.c
2     Printing multiple lines with a single printf */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main()
7  {
8     printf( "Welcome\nto\nC!\n" );
9
10    return 0; /* indicate that program ended successfully */
11
12 } /* end function main */
```

Output :

```
Welcome
To
C!
```

b. Contoh 2

Tentukan outputnya

```
#include <stdio.h>
int A;
int B;
int C;
void main()
{
        A=25;
        B=35;
        C=A+B;
        printf("Hasil Jumlahnya adalah C=%d",C);
        getch();
};
```

1. Latihan 1

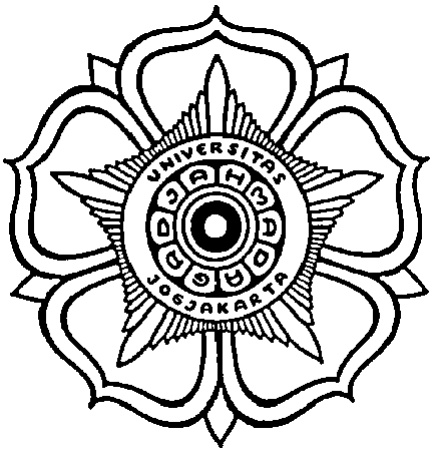
   Buatlah program dengan tampilan ouput seperti berikut :

   Masukkan nilai A : 15
   Masukkan Nilai B : 13
   Hasil jumlahnya = 28
   Hasil Pengurangannya = 2

2. Buatlah program dengan tampilan output sebagai berikut :

Ketikkan nama anda : Budi
Masukkan umur anda : 25
Hai Budi
Anda berumur : 25 tahun
Selamat Belajar Pemrograman dengan bahasa C

MODUL LATIHAN
PEMROGRAMAN KOMPUTER



Disusun oleh:

Ir. BALZA ACHMAD, M.Sc.E.

JURUSAN TEKNIK FISIKA
FAKULTAS TEKNIIK
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2005

LATIHAN PEMROGRAMAN KOMPUTER
MODUL I - INPUT/OUTPUT & JENIS DATA

Program perdana: Hello World!

```
main()
{
  puts("Hello world!")
}
```

Ada beberapa buah error, apa sajakah?(1) Tambahkan baris berikut pada bagian paling atas dan jalankan. Perbaiki pula error yang lain jika ada.

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

Buatlah program menggunakan fungsi **puts** untuk menampilkan tulisan berikut di layar:(2)

```
Hai, nama saya Balzach
Saya sedang belajar memprogram memakai Bahasa C
Lumayan deh kalau sudah lancar nanti
OK, bye-bye ya.
```

Program tampilan menggunakan kode escape

Gantilah fungsi **puts** pada program di atas menjadi **printf** (tanpa mengubah parameternya). Apa yang terjadi? (3)Apakah perbedaan antara **puts** dan **printf**? (4)

Buatlah program berikut:

```
#include <stdio.h>

main()
{
  printf("1 \t2 \t3 \t4 \t5 \t6 \t7 \t8 \n");
  printf("Program\tKomputer\tBahasa\tC++
  printf("\tdi Lab\tini\n");
  printf("Saya sedang mempelajari\r");
  printf("kode escape\n");
  puts("Suara apakah ini?\a");
}
```

Perhatikan tampilan yang ada di layar. Bagaimana efek kode escape \n, \t, \r dan \a? (5)

Buatlah program menggunakan fungsi **puts** atau **printf** untuk menampilkan tulisan berikut di layar: (6)

```
Motto saya:
"Anything you can perceive, you can achieve"
Kode \n adalah untuk ganti baris
```

Program mengisi input ke variabel string

```
#include <stdio.h>

main()
{
   char nama[80];

   printf("Masukkan nama: ");
   gets(nama);
   printf("Halo ");
   puts(nama);
   printf("Betul kan, kamu si %s?\n", nama);
}
```

Apa arti angka 80 pada **nama[80]** di atas? *(7)*
Apa kegunaan fungsi **gets**? *(8)*

Gantilah baris
```
   gets(nama);
```
menjadi:
```
   scanf("%s",nama);
```

Apa pula kegunaan fungsi **scanf**? *(9)*

Program matematika bilangan bulat & riil: pembagian

```
#include <stdio.h>

main()
{
   int a = 10, c;
   float b = 3.5, d;

   c = a/b; d = a/b;
   printf("a = %d\n",a);
   printf("b = %f\n",b);
   printf("c = %d\n",c);
   printf("d = %f\n",d);
}
```

Perhatikan cara menginisialisasi nilai variabel a dan b.
Berapakah nilai c dan d?*(10)* Mengapa nilai keduanya berbeda meskipun operasinya sama?*(11)*

Program input ke variabel bilangan: menghitung akar

```
#include <stdio.h>

main()
{
   int a;
   float b;

   printf("Masukkan nilai a = ");
   scanf("%d",a);
   b = sqrt(a);
   printf("akar dari a = %f",b);
}
```

Terjadi error waktu program di atas dicompile, mengapa?*(12)*

Tambahkan `#include <math.h>` pada bagian atas, lalu compile lagi. Jika berhasil, jalankan. Runtime-error apa yang muncul? *(13)* Perbaiki dan jalankan lagi. Isikan nilai a dengan 9, 16, 25, dll.

Variabel a bertipe bilangan bulat. Coba masukan nilai a = 100.5
Bagaimana hasilnya? *(14)* Mengapa bisa begitu? *(15)*

Gantilah baris
```
   scanf("%d",&a);
```
menjadi:
```
   scanf("%f",&a);
```

Apa yang terjadi? *(16)* Mengapa harus **%d**, dan bukannya **%f**.?*(17)*

Gantilah baris
```
   printf("akar dari a = %f",b);
```
menjadi:
```
   printf("akar dari a = %d",b);
```

Apa yang terjadi? *(18)* Mengapa harus **%f**, dan bukannya **%d**.?*(19)*

Program dengan beberapa input: menghitung rerata

```
#include <stdio.h>

main()
{
   float a, b, c, rerata;

   printf("Masukkan nilai a, b dan c = ");
   scanf("%f %f %f",&a,&b,&c);
   rerata = (a+b+c)/3;
   printf("Rerata = %f",rerata);
}
```

Jalankan program di atas dengan mengisikan tiga buah bilangan dengan diselingi spasi (contoh: 10 20 45). Coba pula dengan diselingi Enter.

Ubahlah tipe variabel a, b, dan c menjadi bilangan bulat. Apalagi yang harus diubah agar tidak terjadi error? *(20)*

Program menghitung invers

```
#include <stdio.h>

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

   printf("Masukkan a = ");
   scanf("%d",&a);
   b = 1/a;
   printf("b = 1/%d = %f\n",a,b);
   c = 1.0/a;
   printf("c = 1/%d = %f\n",a,c);
```

```
}
```

Kompil dan jalankan program di atas. Mengapa nilai b dan c berbeda meskipun operasinya sama? *(21)*

Kepresisian bilangan riil

```
#include <stdio.h>

main()
{
   float a = 3, b;
   double c;

   b = 1000/a;
   c = 1000/a;
   printf("a = %f\n",a);
   printf("b = %f\n",b);
   printf("c = %f\n",c);
}
```

Kompil dan jalankan program di atas. Mengapa nilai b dan c berbeda meskipun operasinya sama? *(22)*

Pengaturan tampilan

```
#include <stdio.h>

main()
{
   double a = 3.0, b;
   b = 1000.0/a;
   printf("a = %f\t\t b = %f\n",a,b);
   printf("a = %7.2f\t\t b = %7.2f\n",a,b);
   printf("a = %+7.4f\t\t b = %+7.4f\n",a,b);
   printf("a = %g\t\t\t b = %g\n",a,b);
   printf("a = %e\t b = %e\n",a,b);
   printf("a = %5.2e\t b = %5.2e\n",a,b);
}
```

Kompil dan jalankan program di atas. Perhatikan tampilan dari nilai a dan b untuk setiap format specifier yang berbeda. Tuliskan secara singkat maksud dari masing-masing format tadi (%f, % 7.2f, %+7.4f, %g, %e, %5.2e) *(23).*

LATIHAN PEMROGRAMAN KOMPUTER
MODUL II - INPUT/OUTPUT & JENIS DATA

Program kombinasi input-output: data diri

Dalam membuat program, usahakan agar mudah dimengerti, yaitu dengan memberi nama variabel yang mempunyai arti, memberi keterangan, dan membuat program terstruktur dengan baik.

Buatlah program untuk mengisikan data diri melalui keyboard dan menampilkannya di layar berikut ini:

```
#include <stdio.h>

main()
{
   char Nama[50], progStudi[15];
   int angkatan, NIM;

   /* Mengisikan data */
   printf("Nama : "); gets(nama);
   printf("Program studi : "); gets(progStudi);
   printf("Angkatan : "); scanf("%d",&angkatan);
   printf("NIM : "); scanf("%d",&NIM);

   /* Menampilkan data */
   printf("\nData anda adalah :\n");
   printf("%s (%s %d/%d)\n", nama,progStudi,angkatan,NIM);
}
```

Pesan error apakah yang muncul saat program tersebut dikompil, serta bagaimana cara membetulkannya? *(1).*

Perhatikan baris ke-9 di atas:

```
   printf("Nama : "); gets(nama);
```

dari sini dapat disimpulkan bahwa beberapa instruksi dapat dituliskan dalam 1 baris.

Perhatikan pula baris ke-2 dan 3 dari bawah:

```
   printf("%s (%s %d/%d)\n",
      nama,progStudi,angkatan,NIM);
```

dari sini dapat disimpulkan bahwa sebuah instruksi tunggal dapat dituliskan dalam lebih dari 1 baris.

Gantilah pemotongan kedua baris tersebut
(a) menjadi:

```
   printf("%s (%s %d/%d)\n",nama,progStudi,angkatan,NIM);
```

(b) menjadi:

```
   printf("%s (%s %d/%d)
      \n",nama,progStudi,angkatan,NIM);
```

Manakah di antara (a) dan (b) yang memberikan pesan error, dan mengapa bisa terjadi error? *(2).*

Output menggunakan stream: Hello World! dalam C++

```
#include <iostream.h>
main()
{
  cout << "Hello world!\nWe're in C++ now";
}
```

Jalankan, dan selamat! Anda telah berhasil membuat program pertama dalam bahasa C++. Gantilah isi programnya menjadi:

```
  cout << "Hello world!" << endl
     << "We're in C++ now";
```

Hasilnya sama dengan sebelumnya. Apakah fungsi dari **endl**? *(3)*

Buatlah program menggunakan **cout** untuk menampilkan tulisan yang sama dengan soal no (2) pada modul I *(4)*.

Input string menggunakan stream: program nama

```
#include <iostream.h>
main()
{
   char nama[80];

   cout << "Masukkan nama: ";
   cin >> nama;
   cout << "Halo " << nama << endl
      << "Betul kan, kamu si " << nama;
}
```

Bandingkan program di atas dengan program yang serupa menggunakan bahasa C dalam modul 1. Menggantikan fungsi apakah **cin** dan **cout** *(5)*.

Program input ke variabel bilangan: menghitung akar

```
#include <iostream.h>
#include <math.h>
main()
{
   int a;
   float b;

   cout << "Masukkan nilai a = ";
   cin >> a;
   b = sqrt(a);
   cout << "akar dari a = " << b;
}
```

Bandingkan dengan program serupa pada modul 1 yang mengisikan variabel dengan fungsi **scanf**. Apakah perbedaan perlakuan terhadap variabel **a** untuk input menggunakan **cin** dalam program di atas dengan yang menggunakan **scanf** *(6)*?

Gantilah tipe variabel a menjadi bilangan riil. Periksalah apakah ada bagian lain yang perlu dimodifikasi akibat pergantian tersebut *(7)*?

Program dengan beberapa input: menghitung rerata

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

   cout << "Masukkan nilai a, b dan c = ";
   cin << a << b << c;
   rerata = (a+b+c)/3;
   cout << "Rerata = " << rerata;
}
```

Apakah yang salah dalam program tersebut dan bagaimanakah yang benar [8]?

Program perhitungan gaji

Buatlah program untuk menghitung total pendapatan bulanan seorang karyawan dengan ketentuan sebagai berikut:

- Tunjangan istri/suami = 10% dari gaji pokok
- Tunjangan anak = 5% dari gaji pokok untuk setiap anak
- THR = Rp 5000 kali masa kerja (tahun)
- (-) Pajak = 15% dari gaji pokok, tunjangan istri & anak
- Bantuan transport = Rp 3000 kali masuk kerja (hari)
- (-) Polis asuransi = Rp 20000

tanda (-) artinya mengurangi pendapatan.

Tentukan dahulu apa-apa saja yang akan menjadi input (dari kibod) dan output (ke layar) dari program [9]. Tentukan juga variabel-variabel yang akan digunakan beserta tipenya [10] (ingatlah untuk menggunakan nama yang mudah dimengerti). Buat *pseudo-code*nya[11] serta programnya dalam bahasa C++ [12].

LATIHAN PEMROGRAMAN KOMPUTER
MODUL III - STRUKTUR PEMROGRAMAN: PENCABANGAN

Pencabangan tunggal: IF

Buatlah program di bawah ini. Jika umur diisi lebih dari 60 tahun maka akan ditampilkan komentar.

```
#include <iostream.h>
main()
{  int umur;

   cout << "Masukkan umurmu = ";
   cin >> umur;
   if umur >= 60
      cout << "Halo mbah" << endl;
   cout << "Jadi umurmu " << umur << " tahun";
}
```

Kompilelah program tersebut, error apakah yang muncul dan bagaimana yang benar? *(1)* Perbaiki program tersebut dan jalankan beberapa kali dengan mengisikan nilai umur yang bervariasi.

Tambahkan baris berikut setelah baris cout << "Halo mbah..., untuk memberi komentar tambahan jika umur lebih dari 60 tahun

```
      cout << "Salam buat cucumu ya" << endl;
```

Jalankan dengan mengisikan umur 80 tahun, lalu jalankan lagi untuk umur 20 tahun. Mengapa komentar tambahan selalu muncul untuk umur berapapun (tidak sesuai dengan yang diinginkan) serta bagaimana yang benar? *(2)*

Buatlah program sesuai contoh berikut

```
#include <iostream.h>
main()
{  int tahun, umur;

   cout << "Masukkan tahun kelahiranmu = ";
   cin >> tahun;
   umur = 2003-tahun;
   cout << "Umurmu " << umur << " tahun\n";
   if (umur < 17);
   {  cout << "Kamu belum sweet seventeen\n";
      cout << "Belum cukup umur\n"; }
}
```

Kompile program tersebut, tidak ada *compile* error (kecuali beberapa warning). Jalankan dan isikan tahun kelahiran 1988 dan 1975 serta nilai lain. Kesalahan apakah yang terjadi dan bagaimana yang benar? *(3)*

Pencabangan ganda: IF-ELSE

Buatlah program untuk menentukan apakah seseorang lulus atau tidak menggunakan nilainya

```
#include <iostream.h>
main()
{  int N;

   cout << "Masukkan nilai (0 s/d 100) = ";
   cin >> N;
   if (N >= 50)
      cout << "Lulus"
   else (N < 50)
      cout << "Tidak lulus";
}
```

Error apa yang muncul waktu dikompile dan bagaimana yang benar? [(4)] Perbaiki dan jalankan dengan mengisikan beberapa nilai N.

Pencabangan bertingkat: IF-ELSE-IF

Buatlah program untuk memberi nilai huruf berikut.

```
#include <iostream.h>
main()
{  int N;

   cout << "Masukkan nilai = ";
   cin >> N;
   cout << "Nilai huruf = ";
   if (N >= 80)
      cout << "A";
   if (N >= 60)
      cout << "B";
   if (N >= 40)
      cout << "C";
   if (N < 40)
      cout << "D";
}
```

Jalankan dan isikan nilai 15, 25, 50, 75, dan 100. Hasilnya tidak sesuai dengan yang seharusnya, jelaskan mengapa bisa terjadi seperti itu[(5)]. Perbaiki program di atas dengan struktur pencabangan bertingkat: *if ... else if ... else if ....* dst.[(6)]. Buat flowchart dari program tersebut [(7)].

Kondisi kombinatorial

Perbaiki program di atas dengan kondisi yang merupakan kombinasi logika, misalnya: *jika N lebih dari atau sama dengan 60 tetapi kurang dari 80 maka ...*, tanpa menggunakan pencabangan bertingkat [(8)]. Buat flowchart dari program tersebut [(9)].

Buat flowchart dari program di bawah ini[(10)].

```
#include <iostream.h>
main()
{  int N;

   cout << "Masukkan nilai = ";
   cin >> N;
   cout << "Nilai huruf = ";
   if (N >= 40)
      if (N >= 60)
         if (N >= 80)
```

```
            cout << "A";
         else
            cout << "B";
      else
         cout << "C";
   else
      cout << "D";
}
```

Menu: Program konversi suhu

Buatlah program untuk menghitung konversi suhu dari C ke F atau R dan sebaliknya. Program dimulai dengan mengisikan suhu yang akan dikonversi, kemudian menampilkan pilihan (menu) sebagai berikut:

```
  Pilih konversi berikut:
    A. Celcius ke Fahrenheit
    B. Fahrenheit ke Celcius
    C. Celcius ke Reamur
    D. Reamur ke Celsius
    E. Fahrenheit ke Reamur
    F. Reamur ke Fahrenheit
```

Setelah dipilih, maka program melakukan perhitungan konversi sesuai dengan yang dipilih dan menampilkan hasilnya. Gunakan struktur pencabangan IF untuk program tersebut. Buat dahulu flowchartnya baru programnya *(11)*.

LATIHAN PEMROGRAMAN KOMPUTER
MODUL IV - STRUKTUR PEMROGRAMAN: LOMPATAN & KALANG

Pencabangan

Jalankan program berikut dan isikan umur = 20, 19, 21, 23, dst. Bagaimana keluarannya? *(1)*. Bagian mana yang salah dan bagaimana modifikasinya? *(2)*.

```
#include <iostream.h>
main()
{  int umur;

   cout << "Masukkan umur anda = ";
   cin >> umur;
   if (umur = 20)
      cout << "Umur anda tepat duapuluh tahun";
   else
      cout << "Umur anda bukan duapuluh tahun";
}
```

Pencabangan berganda: SWITCH

Jalankan program berikut dan isikan kategori = A,B,C,D, atau E. Bagaimana keluarannya? *(3)*. Modifikasilah agar menjadi benar*(4)*.

```
#include <iostream.h>
main() {
   char kategori;
   float diskon;

   cout << "Kategori pelanggan (A/B/C/D/E) = ";
   cin >> kategori;
   switch (kategori) {
      case 'A':
        diskon = 40;
      case 'B':
        diskon = 25;
      case 'C':
      case 'D':
        diskon = 10;
      default:
        diskon = 0;
   }
   cout << "Diskon = " << diskon << "%";
}
```

Lompatan: label dan goto

Buatlah program di bawah ini. Jalankan dan tulislah tampilan yang muncul *(5)*. Terangkan fungsi dari label dan goto *(6)*.

```
#include <iostream.h>
main()
{
   cout << "Ini langkah pertama" << endl;
   goto LABEL2;
LABEL1:
   cout << "Ini langkah kedua" << endl;
   goto LABEL3;
```

```
LABEL2:
   cout << "Ini langkah ketiga" << endl;
   goto LABEL1;
LABEL3:
   cout << "Ini langkah keempat" << endl;
}
```

Pencabangan & lompatan untuk perulangan

```
#include <iostream.h>
#include <conio.h>
main()
{  float C, F;
   int tombol;

Ulangi
   cout << "Masukkan suhu dalam Celcius = ";
   cin >> C;
   F = 1.8*C+32;
   cout << "Suhu dalam Fahrenheit = " << F
        << endl;
   cout << "Apakah mau mengulangi (Y/T)? ";
   tombol = getch();
   cout << endl;
   if (tombol == 'Y')
     goto Ulangi;
   cout << "Selesai";
}
```

Kompile program di atas, ada error yaitu kesalahan dalam menulis label, bagaimana yang benar? *(7)*. Perbaiki dan jalankan. Modifikasi program tersebut untuk dapat mengulangi jika diberi jawaban karakter Y maupun y*(8)*. Apakah fungsi dari **getch()***(9)*

Kalang bersyarat "periksa-jalankan" : WHILE

```
#include <iostream.h>
#include <conio.h>
main()
{  char nama[80];
   int tombol;

   while ((tombol == 'Y')||(tombol == 'y')) {
      cout << "Masukkan nama anda = ";
      cin >> nama;
      cout << "Halo " << nama << endl << endl;
      cout << "Apakah mau mengulangi (Y/T)? ";
      tombol = getch();
      cout << endl << endl;
   }
   cout << "Selesai";
}
```

Jalankan program di atas, apa yang terjadi dan mengapa bisa begitu? *(10)*. Bagaimana yang benar? *(11)*. Perbaiki dan jalankan.

Kalang bersyarat "jalankan- periksa": DO-WHILE

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

```
main()
{  char nama[80];
   int tombol, cacah = 0;
   float nilai, jumlah = 0, rerata;

   cout << "Menghitung rerata nilai\n";
   cout << "Masukkan nilai, "
        << "isikan negatif jika selesai\n\n";
   do {
      cacah++;
      cout << "Data ke-" << cacah << " = ";
      cin >> nilai;
      jumlah = jumlah+nilai;
   } while (nilai >= 0);
   rerata = jumlah/cacah;
   cout << "\nBanyaknya data = " << cacah;
   cout << "\nJumlah = " << jumlah;
   cout << "\nRerata = " << rerata;
}
```

Apa fungsi instruksi **cacah++** *(12)*.
Jalankan program di atas dan masukkan beberapa nilai untuk dihitung reratanya, akhiri masukan dengan memberikan nilai negatif (jangan cuma tanda minus thok). Periksa jawabannya, apakah jawaban yang diperoleh sudah benar? Perbaiki program tersebut agar menjadi benar *(13)*.

Buatlah program untuk menghitung jumlah & rerata nilai, tetapi dengan memasukkan dahulu berapa banyaknya nilai yang akan dimasukkan. Tampilannya adalah sbb (huruf tebal adalah nilai yang diisikan melalui keyboard). Buatlah dalam 2 versi, pertama menggunakan kalang WHILE *(14)*, dan kedua menggunakan kalang DO-WHILE *(15)*.

```
Banyaknya nilai = 3
Data ke-1 = 10
Data ke-2 = 9
Data ke-3 = 8

Jumlah = 27
Rerata = 9
```

Menu: Program konversi suhu dengan SWITCH

Buat pula program yang sama dengan pada bagian akhir Modul III menggunakan struktur SWITCH, Buat dahulu flowchartnya baru programnya *(16)*.

Tabel kuadrat

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

main()
{
   int x = 0;

   cout << "  x    x kuadrat\n";
   while (x <= 20)
     cout <<setw(3)<<x<<setw(8)<<(x*x)<<endl;
     x++;
}
```

Jalankan program untuk menampilkan tabel kuadrat di atas. Jangan kaget karena program akan terus mengalir ke bawah. (untuk menghentikannya tekan tombol CTRL C. Jika masih belum bisa, tekan CTRL ALT DEL dan tunggu beberapa saat lalu tekan tombol End Task pada bagian yang Not Responding). Apa yang menyebabkan hal itu terjadi[(1)]? Bagaimana yang benar[(2)]? Gantilah angka-angka pada fungsi **setw()** dan perhatikan hasilnya. Apa guna fungsi tersebut [(3)] ?

Memutus kalang: BREAK & CONTINUE

```
#include <iostream.h>

main()
{
   int i = 0;

   cout << i;
   do {
      i++;
      cout << " - ";
//      if (i == 4)
//        break;
      cout << i;
   } while (i < 10);
   cout << "\nSelesai\n";
}
```

Jalankan program di atas dan catat tampilannya di layar. Apa fungsi simbul / / [(4)] ? Hilangkan kedua tanda / / tersebut dan jalankan. Apa fungsi dari keyword **break** [(5)] ? Gantilah **break** dengan **continue** dan jalankan. Apa fungsi dari **continue**[(6)] ?

Tabel sinus

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

main()
{
   const float PI = 3.141593;
   int x = 0, x2;
```

```
   float y, y2;

   puts("Tabel sinus\n-----------");
   puts("  x    sin(x)       x    sin(x)");
   do {
      y = sin(x/180.0*PI);
      x2 = x+180;
      y2 = sin(x2/180*PI);
      printf("%4d %8.4f     %4d %8.4f\n",
         x,y,x2,y2);
      x += 10;
   } while (x <= 180);
}
```

Program di atas menampilkan tabel sinus. Apa yang menyebab-kan nilai dalam tabel tersebut salah [(7)] Bagaimana yang benar [(8)] Apakah fungsi dari operator += pada instruksi **x += 10** [(9)]. Tambahkan instruksi berikut pada baris setelah while

```
   PI = 3.14;
```

Pesan error apa yang muncul dan mengapa terjadi error [(10)]?

Tabel konversi suhu

Dengan struktur yang serupa (tapi tak sama) memakai DO-WHILE, buatlah tabel konversi suhu dari Celcius ke Fahrenheit, Reamur, dan Kelvin, mulai dari suhu 0°C sampai 100°C dengan langkah 5°C. Tampilannya kurang lebih adalah sbb [(11)]

```
Tabel konversi suhu
--------------------
  C       F        R       K
  0     32.0      0.0   273.25
  5     41.0      4.0   278.25
 10     50.0      8.0   283.25
...
 95    203.0     76.0   368.25
100    212.0     80.0   373.25
```

Metode numeris: Bisection

Program untuk mencari akar persamaan (zeros) menggunakan metode bisection berikut ini adalah contoh aplikasi pemrograman komputer pada metode numeris. Jawaban secara analitik adalah x = 3 atau -1. Jalankan dan masukkan tebakan awal bawah = 2 dan atas = 5 (cobalah juga kombinasi yang lain, misalnya xL = -5 dan xU = 0).

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

main()
{
   int iterasi = 0;
   double xL, xC, xU, yL, yC, yU;

   cout << "Mencari akar persamaan x*x-2*x-3=0"
        << " dengan metode bisection\n"
        << "Masukkan tebakan awal bawah = ";
   cin >> xL;
   cout << "Masukkan tebakan awal atas = ";
```

```
   cin >> xU;
   cout << "iterasi     xL          xC"
        << "          xU          yL"
        << "          yC          yU\n";
   do {
      xC = (xL+xU)/2;
      yL = xL*xL-2*xL-3;
      yU = xU*xU-2*xU-3;
      yC = xC*xC-2*xC-3;
      cout << setw(5) << iterasi
           << setiosflags(ios::fixed)
           << setprecision(5)
           << setw(12) << xL
           << setw(12) << xC
           << setw(12) << xU
           << setw(12) << yL
           << setw(12) << yC
           << setw(12) << yU << endl;
      if (yL*yC <= 0)
         xU = xC;
      else
         xL = xC;
      iterasi++;
   } while ((iterasi <= 50)&&(fabs(yC)>1e-5));
   cout << "Jadi akarnya adalah " << xC;
}
```

Gantilah parameter pada fungsi **setprecision()** dengan angka 7. Apakah guna dari fungsi **setprecision()** *(12)*? Dengan menggunakan nilai presisi 4, gantilah **ios::fixed** menjadi **ios::scientific**. Apa beda antara **ios::fixed** dan **ios::scientific** *(13)*? Apakah fungsi dari **fabs()***(14)*?

Apabila hendak menggunakan program tersebut untuk persamaan yang lain yaitu **exp(x) - 2*x - 2 = 0**, apakah yang harus diubah dalam program tersebut*(15)*? (akar persamaan tersebut kira-kira adalah = 1.67834.

LATIHAN PEMROGRAMAN KOMPUTER
MODUL VI - STRUKTUR PEMROGRAMAN: KALANG FOR & VARIABEL LARIK (ARRAY)

Kalang FOR untuk perulangan

```
#include <iostream.h>
main()
{
   int i;
   for (i=1; i<=20; i++) {
     cout << "kalang ke-" << i << endl;
   }
}
```

Jalankan program di atas dan lihat hasilnya. Perhatian fungsi dari ketiga bagian pada struktur for (yang dibatasi dengan tanda titik-koma) dengan mengubah baris for di atas menjadi:

```
(a)   for (i=10; i<=20; i++) {
(b)   for (i=1; i<=10; i++) {
(c)   for (i=1; i<=20; i=i+2) {
```

Jelaskan fungsi masing-masing bagian pada baris for *(1)*

Kalang FOR bertingkat: Tabel Perkalian

```
#include <iostream.h>
#include <iomanip.h>
main()
{
   for (int i=1; i<=10; i++) {
     for (int j=1; j<=10; j++) {
       cout << setw(5) << i*j;
     }
     cout << endl;
   }
}
```

Jalankan program di atas dan perhatikan tampilannya. Ubahlah satu baris saja program tersebut untuk mendapatkan tabel yang berbentuk segitiga. dengan contoh tampilan sbb: *(2)*

```
  1
  2    4
  3    6    9
  ...
  9   18   27   36   45   54   63   72   81
 10   20   30   40   50   60   70   80   90  100
```

Kalang FOR untuk Tabel Kode ASCII

Jalankan program untuk menampilkan tabel ASCII di bawah ini.
Tabel ini berisi kode-kode ASCII dan karakter yang diwakilinya. Kelompok karakter apa yang masing-masing mempunyai kode dari 48 s/d 57, dari 65 s/d 90, dari 97 s/d 122 *(3)*?

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

```
   cout << "\t";
   for (int i=32; i<=255; i++) {
     cout << setw(3) << i << " "
          << char(i) << "   ";
   }
}
```

Kalang FOR bertingkat untuk simulasi bunyi dering telepon

Kompile program di bawah ini. 2 buah error apa yang terjadi dan bagaimana yang benar [(4)]? Ubahlah angka 1000 menjadi 5000, dan periksalah apa fungsi **delay()** [(5)]? Ubahlah nilai pada sound(), dan periksalah apa fungsi **sound()** [(6)]?

```
#include <iostream.h>
#include <dos.h>
main()
{
  int kring;
  for (kring = 1, kring <=2, kring++) {
    delay(1000);
    cout << "kr";
    for (i=1; i<=20; i++) {
      cout << "i";
      sound(600);
      delay(30);
      nosound();
      sound(1500);
      delay(30);
      nosound();
    }
  cout << "ng\n";
  }
}
```

Variabel larik pada perhitungan nilai rerata

Ingat kembali program menghitung rerata 3 nilai pada Modul II. Alih-alih menggunakan 3 buah variabel (a, b, dan c), program tersebut dapat dimodifikasi menggunakan sebuah variabel larik a. Error apa yang terjadi dan bagaimana yang benar [(7)]?

```
#include <iostream.h>
main()
{
   float rerata, a[3];

   cout << "Masukkan 3 buah nilai = ";
   cin >> a(0) >> a(1) >> a(2);
   rerata = (a[0]+a[1]+a[2])/3;
   cout << "Rerata = " << rerata;
}
```

Program menghitung rerata secara umum dapat berupa program di bawah ini. Error apa yang ada dan bagaimana yang benar [(8)]? Isikan banyaknya nilai = 4 dan isikan nilai-nilainya. Program berjalan dengan lancar. Lalu coba lagi dengan banyaknya nilai = 15 dan isikan nilai-nilainya. Apa yang terjadi dan mengapa begitu, serta bagaimana memperbaikinya [(9)]?

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

```
{
   int N;
   float jumlah, rerata, a[5];
   cout << "Banyaknya nilai = ";
   cin >> N;
   jumlah = 0;
   for (int i=0; i<N; i++) {
     cout << "Nilai ke-" << (i+1) << " = ";
     cin >> a[i];
     jumlah = jumlah+a[i];
   }
   rerata = jumlah/N;
   cout << "Rerata = " << rerata;
}
```

Perhitungan nilai rerata dan standar deviasi

Tambahkan beberapa baris (atau modifikasi) program di atas sehingga selain memberikan nilai rerata juga menghitung standar deviasi [10]?

Program menghitung tinggi pantulan bola jatuh bebas

Buatlah program untuk menghitung tinggi pantulan bola yang jatuh, dengan asumsi tinggi pantulan bola adalah 0.8 kali tinggi sebelumnya. Tinggi awal bola jatuh diisikan melalui keyboard. Contoh tampilan program terlihat di bawah ini. Maksimal perhitungan adalah sampai pantulan ke 20 atau apabila tinggi pantulan sudah di bawah 1. Gunakan kalang **for** dan **break**. [11]?

```
Tinggi awal = 10
Pantulan ke      tinggi
     1           8.000
     2           6.400
     3           5.120
    ...
    10           1.074
    11           0.859
```

LATIHAN PEMROGRAMAN KOMPUTER
MODUL VII - VARIABEL LARIK: MATRIKS & STRING

Sekup variabel dalam kalang FOR

```
#include <iostream.h>
main()
{
  int i = 5;

  cout << "di luar kalang i = " << i << endl;
  for (i=1; i<=10; i++) {
    cout << "di dalam kalang i = " << i << endl;
  }
  cout << "di luar kalang i = " << i << endl;
}
```

Jalankan program di atas dan lihat hasilnya. Gantilah pernyataan for di atas menjadi:

```
  for (int i=1; i<=10; i++) {
```

jalankan program tersebut dan perhatikan nilai i yang ditampil-kan. Apa pengaruh pendefinisian variabel i di dalam kalang for[(1)]. Gantilah pernyataan for di atas dengan menghilangkan bagian inisialisasi kalang for menjadi:

```
  for (; i<=10; i++) {
```

jalankan program tersebut dan lihat nilai i yang ditampilkan. Apakah yang terjadi apabila tidak dilakukan inisialisasi pada kalang for[(2)].

Kalang FOR untuk mengisi dan menampilkan matriks

```
#include <iostream.h>
main()
{
  int N = 3;
  float A[5][5];

  for (int i=0; i<N; i++) {
    for (int j=0; j<N; j++) {
      cout << "A(" << i << "," << j << ") = ";
      cin >> A[i][j];
    }
  }
  cout << "Matriks A = \n";
  for (int i=0; i<N; i++) {
    for (int j=0; j<N; j++) {
      cout << "\t" << A[i][j];
    }
    cout << endl;
  }
}
```

Program di atas adalah sebuah contoh program yang digunakan untuk mengisi nilai-nilai elemen pada matriks ukuran 3x3. Modifikasi (dengan menambah atau mengubah sekitar 3 baris saja) program tersebut agar dapat digunakan untuk mengisi dan menampilkan matriks yang berukuran 4 baris x 5 kolom [(3)].

Tambahkan beberapa baris perintah-perintah lain pada program di atas untuk mengisi dan menampilkan juga matriks B dengan ukuran yang sama (3x3), kemudian juga menampilkan

matriks C yang merupakan matriks hasil penjumlahan A dan B. Ingatlah bahwa penjumlahan matriks dilakukan elemen demi elemen [4].

Kemudian dari program tersebut gantilah kalang yang digunakan untuk menghitung matriks C sebagai penjumlahan antara A dan B menjadi perkalian antara matriks A dan B, seperti di bawah ini

```
  for (int i=0; i<N; i++) {
    for (int j=0; j<N; j++) {
      C[i][j] = 0;
      for (int k=0; k<N; k++) {
        C[i][j] = C[i][j]+A[i][k]*B[k][j];
      }
    }
  }
```

Operasi pada string: Kutak-katik nama

```
#include <stdio.h>
#include <iostream.h>
#include <string.h>
main()
{
   char nama[80], nama2[80], tulisan[80];
   int panjang, posisi;

   cout << "Masukkan nama = ";
   gets(nama);
   strcpy(tulisan,"Halo ");
   cout << strcat(tulisan,nama) << endl;
   if (strcmp(nama,"Fella") == 0) {
     cout << "Namamu Fella kan" << endl; }
   else {
     cout << "Namamu bukan Fella" << endl; }
   panjang = strlen(nama);
   cout << "panjang namamu " << panjang << endl;
   posisi = strcspn(nama,"z");
   if (posisi < panjang) {
     cout << "Huruf z pada posisi " << posisi
          << endl; }
   else {
     cout << "Tidak ada huruf z nya" << endl; }
   strcpy(nama2,nama);
   cout << strupr(nama2) << endl;
   cout << strlwr(nama2) << endl;
   cout << strrev(nama2) << endl;
   cout << strset(nama2,'x') << endl;
}
```

Jalankan program di atas dan jelaskan kegunaan dari fungsi berikut ini: strlen(), strcpy(), strcat(), strcmp(),strlen(), strcspn(), strupr(), strlwr(), strrev(), dan strset() [5]. Gantilah baris:

```
   gets(nama);
```

menjadi:

```
   cin >> nama;
```

Apakah efeknya pada waktu program dijalankan? [6]

String sebagai larik karakter: Caesar Chiper

Program berikut ini digunakan untuk mengenkripsi (mengkode-kan) sebuah tulisan dengan Caesar Chiper. Caranya adalah dengan menggeser kode ASCII dari masing-masing karakter pada tulisan tersebut dengan sebuah kunci yang berupa bilangan bulat. Kode yang valid (*visible*) adalah antara 32 s/d 126.

```
#include <iostream.h>
#include <stdio.h>
#include <string.h>
main()
{
   char tulisan[100], terenkripsi[100];
   int kunci, panjang, kode;

   cout << "Masukkan tulisan = ";
   gets(tulisan);
   cout << "Masukkan nilai kunci = ";
   cin >> kunci;
   panjang = strlen(tulisan);
   strcpy(terenkripsi,tulisan);
   for (int i=0;i<panjang;i++) {
     kode = int(tulisan[i])+kunci;
     if (kode < 32)
       kode = kode + 95;
     if (kode > 126)
       kode = kode - 95;
     terenkripsi[i] = char(kode);
   }
   cout << "Terenkripsi = " << terenkripsi << endl;
}
```

Jalankan program tersebut, masukkan sebuah tulisanb dan juga kunci untuk mengenkripsi (bil bulat positif atau negatif). Sifat operasinya adalah dapat-balik (*reversible*). Untuk mendekripsi gunakan nilai lawan dari kunci tersebut. Bagaimana bentuk terenkripsi dari “Teknik Fisika” dengan kunci 10 [7]? Apa tulisan asli dari pesan berikut ini serta berapa nilai kunci yang dipakai untuk mengenkripsi tulisan aslinya [8]?

```
Ru|qzq#0!u}# w#q}q~0rqxq$q0S
```

Hilangkan baris `strcpy(...` di atas, apa yang terjadi [9]?

LATIHAN PEMROGRAMAN KOMPUTER
MODUL VIII - FUNGSI

Fungsi untuk memotong-motong program

Ingat kembali program untuk menghitung rerata pada modul VI. Kita dapat memotong-motong program tersebut menggunakan fungsi sehingga mudah dipelajari. Jalankan program tersebut, lalu ubah dan tambahkan perhitungan standar deviasi [1].

```
#include <iostream.h>

int N;
float jumlah, rerata, data[100];

void Masukkan_Data() {
  cout << "Banyaknya nilai = ";
  cin >> N;
  for (int i=0; i<N; i++) {
    cout << "Nilai ke-" << (i+1) << " = ";
    cin >> data[i];
  }
}

void Hitung_Rerata() {
  jumlah = 0;
  for (int i=0; i<N; i++) {
    jumlah = jumlah+data[i];
  }
  rerata = jumlah/N;
}

void Tampilkan_Hasil() {
  cout << "Jumlah = " << jumlah << endl;
  cout << "Rerata = " << rerata << endl;
}

main()
{
  Masukkan_Data();
  Hitung_Rerata();
  Tampilkan_Hasil();
}
```

Fungsi untuk mengisi dan menampilkan matriks

Program berikut sama dengan program pada modul VII, namun instruksi untuk mengisi dan menampilkan matriks dijadikan fungsi yang dipanggil pada main program.

```
#include <iostream.h>
int N = 2;

void MengisiMatriks(float X[5][5])
{
  for (int i=0; i<N; i++) {
    for (int j=0; j<N; j++) {
      cout <<"elemen "<< i << "," << j << " = ";
      cin >> X[i][j];
    }
  }
}
```

```
void MenampilkanMatriks(float X[5][5])
{
  for (int i=0; i<N; i++) {
    for (int j=0; j<N; j++) {
      cout << "\t" << X[i][j];
    }
    cout << endl;
  }
}

main()
{
  float A[5][5];

  cout << "Masukkan elemen matriks A\n";
  MengisiMatriks(A);
  cout << "Matriks A = \n";
  MenampilkanMatriks(A);
}
```

Selanjutnya, modifikasilah program di atas untuk dapat mengisi dan menampilkan matriks B dengan ukuran yang sama, serta matriks C yang merupakan penjumlahan antara A dan B [(2)].

Fungsi dengan nilai kembalian (*return value*) : Bisection

Review kembali program bisection pada Modul V. Modifikasi program tersebut dengan menggunakan fungsi, yaitu dengan menambahkan fungsi berikut sebelum main().

```
double y(double x)
{
  return x*x-2*x-3;
}
```

Kemudian gantilah instruksi yang digunakan untuk mengitung nilai-nilai yL, yC, dan, yU menjadi berikut ini:

```
  yL = y(xL);
  yC = y(xC);
  yU = y(xU);
```

Jalankan dan periksa hasilnya. Apabila program tersebut digunakan untuk persamaan yang lain yaitu **exp(x) - 2*x - 2 = 0**, apakah yang harus diubah dalam program tersebut [(3)]? (akar persamaan tersebut kira-kira adalah = 1.67835).

Program konversi suhu

Berikut ini adalah program untuk mengkonversi suhu dalam Celcius ke Fahrenheit menggunakan fungsi.

```
#include <iostream.h>

float Suhu_C, Suhu_F;                        // definisikan variabel

float C_ke_F(float C) {
  float F;
  F = 1.8*C+32.0;
  return F;
}

void Mengisi_Input() {
```

```
  cout << "Isikan nilai Suhu C = ";    // tampilkan tulisan
  cin >> Suhu_C;                                  // isikan nilai Suhu C
}

void Mengkonversi() {
  Suhu_F = C_ke_F(Suhu_C);             // hitung nilai Suhu F
}

void Menampilkan_Hasil() {
  cout << "Temperatur " << Suhu_C << " C = "
       << Suhu_F << " F" << endl;
}

main() {
  Mengisi_Input();
  Mengkonversi();
  Menampilkan_Hasil();
}
```

Jalankan program di atas dan lihat hasilnya. Pindahkan keempat fungsi yang ada ke bagian bawah setelah akhir dari program utama main() lalu kompilelah. Error apa yang muncul?[4] Lalu tambahkan pendefinisian keempat fungsi tadi di atas main():

```
  float C_ke_F(float C);
  void Mengisi_Input();
  void Mengkonversi();
  void Menampilkan_Hasil();
```

Jalankan dan jelaskan apa guna definisi fungsi tersebut[5].

Buatlah program untuk menghitung konversi suhu dari C ke F atau R dan sebaliknya. Program dimulai dengan menampilkan pilihan (menu) sebagai berikut:

```
  Pilih konversi berikut:
    A. Celcius ke Fahrenheit
    B. Fahrenheit ke Celcius
    C. Celcius ke Reamur
    D. Reamur ke Celsius
    E. Fahrenheit ke Reamur
    F. Reamur ke Fahrenheit
    X. Selesai
```

Setelah dipilih, lalu mengisikan suhu yang akan dikonversi, kemudian program melakukan perhitungan konversi sesuai dengan yang dipilih dan menampilkan hasilnya. Gunakan struktur pemrograman fungsi untuk masing-masing jenis konversi [6].

LATIHAN PEMROGRAMAN KOMPUTER
MODUL IX - FUNGSI

Sekup variabel di dalam fungsi

```
#include <iostream.h>
int X = 10;

void SebuahFungsi()
{
  cout << "Di dalam fungsi, X = "
       << X << endl;
}

main()
{
  cout << "Di dalam program utama, X = "
       << X << endl;
  SebuahFungsi();
  cout << "Keluar ke program utama lagi, X = "
       << X << endl;
}
```

(a) Jalankan program di atas dan perhatikan hasilnya. Tambahkan sebaris di atas `cout` dalam `SebuahFungsi` dengan:
(b) `X = 20;`
(c) `int X = 20;`
Apakah yang terjadi pada X untuk ketiga kasus tersebut [1].

Pelewatan parameter pada pemanggilan fungsi

```
#include <iostream.h>

void Gandakan(int A, int *B, int &C)
{
  A = A*2;
  *B = *B*2;
  C = C*2;
}

main()
{
  int X = 1, Y = 10, Z = 6;
  cout << "Sebelum fungsi Gandakan dipanggil\n"
       << "X = " << X << endl
       << "Y = " << Y << endl
       << "Z = " << Z << endl;
  Gandakan(X, &Y, Z);
  cout << "Setelah fungsi Gandakan dipanggil\n"
       << "X = " << X << endl
       << "Y = " << Y << endl
       << "Z = " << Z << endl;
}
```

Variabel manakah yang akan diubah nilainya ketika dijadikan parameter yang dilewatkan pada fungsi Gandakan [2].

Overloading, pendefinisian fungsi dengan nama sama

```
#include <iostream.h>

float HitungLuas(float R);
{
  return 3.14159*R*R;
}

float HitungLuas(float P, float L)
{
  return float P*L;
}

main()
{
  float Radius, Luas, Panjang, Lebar;
  cout << "Radius lingkaran = ";
  cin >> Radius;
  cout << "Panjang segiempat = ";
  cin >> Panjang;
  cout << "Lebar segiempat = ";
  cin >> Lebar;
  Luas = HitungLuas(Radius);
  cout << "Luas lingkaran = " << Luas << endl;
  Luas = HitungLuas(Panjang, Lebar);
  cout << "Luas segiempat = " << Luas << endl;
}
```

Apakah yang menyebabkan terjadinya error pada program di atas? *(3)*. Perbaikilah. Mengapa tidak terjadi error ketika 2 buah fungsi menggunakan nama yang sama? *(4)*.

Nilai default pada parameter fungsi

```
#include <iostream.h>

void Keterangan(int N : 10)
{
  if (N >= 8)
    cout << "Nilai " << N << " itu bagus\n";
  else if (N < 5)
    cout << "Nilai " << N << " itu parah\n";
  else
    cout << "Nilai " << N << " itu cukupan\n";
}

main()
{
  int Nilai;
  Keterangan();
  cout << "Masukkan nilai = ";
  cin >> Nilai;
  Keterangan(Nilai);
}
```

Apakah yang menyebabkan terjadinya error pada program di atas? *(5)*. Perbaikilah. Mengapa tidak terjadi error ketika fungsi di atas dipanggil tanpa mengisikan parameternya? *(6)*

Fungsi penukaran dua buah variabel

```
#include <iostream.h>

void Tukarkan(int Angka1, int Angka2)
```

```
{
  int temp = Angka1;
  Angka1 = Angka2;
  Angka2 = temp;
}

main()
{
  int X = 1, Y = 99;
  cout << "Sebelum ditukarkan";
  cout << "\nX = " << X << ", Y = " << Y;
  Tukarkan(X, Y);
  cout << "\nSetelah ditukarkan";
  cout << "\nX = " << X << ", Y = " << Y;
}
```

Mengapa nilai X dan Y tidak bertukaran pada program di atas, dan bagaimana cara memperbaikinya [7]. Cobalah program sorting di bawah ini, dengan masih menggunakan fungsi Tukarkan di atas.

```
void Cetak(int D[])
{
  for (int i=0; i<6; i++)
    cout << D[i] << " ";
  cout << endl;
}

main()
{
  int Data[6] = {10, 3, 5, 20, 15, 7};

  cout << "Sebelum disortir:\n";
  Cetak(Data);
  cout << "Proses sortir:\n";
  for (int i=0; i<5; i++) {
    for (int j=4; j>=i; j--) {
      if (Data[j+1] < Data[j])
        Tukarkan(Data[j], Data[j+1]);
      Cetak(Data);
    }
    cout << "-----\n";
  }
}
```

LATIHAN PEMROGRAMAN KOMPUTER
MODUL X - FILE & GRAFIK

Menulis ke file teks

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

main()
{
   char namafileoutput[] = "c:\\filesaya.txt";
   char teks[80];
   FILE *fout;

   fout = fopen(namafileoutput, "w");
   cout << "Ketikkan teks yang akan ditulis "
        << "ke file, akhiri dengan END\n";
   while (1) {
     gets(teks);
     if (strcmp(teks,"END") == 0)
       break;
     strcat(teks,"\n");
     fputs(teks, fout);
   }
   fclose(fout);
   cout << "Selesai" << endl;
}
```

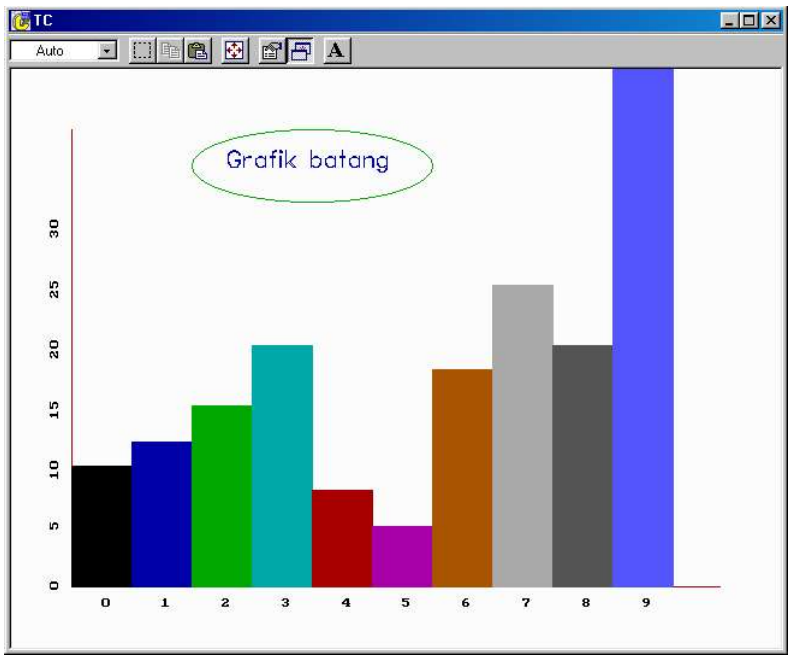
Membaca dari file teks

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

main()
{
   char namafileinput[] = "c:\\filesaya.txt";
   char teks[80];
   FILE *fin;

   fin = fopen(namafileinput, "rt");
   cout << "Isi file " << namafileinput << endl;
   while (1) {
     if (fgets(teks, 80, fin)==NULL) {
       break;
     }
     cout << teks;
   }
   fclose(fin);
   cout << "Selesai" << endl;
}
```

Menampilkan grafik batang

Program berikut ini menampilkan grafik batang dari data yang ada. Untuk pilihan-pilihan huruf, warna, fill style, dan lain-lain lihat dalam file c:\bc5\include\graphics.h
Perhatian: untuk dapat menggunakan perintah-perintah yang berhubungan dengan grafik dari Borland Graphics Interface, pada saat membuat project, pilihlah Target Type: Application [*.exe], Platform: DOS (Standard), Libraries: BGI.

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

main()
{
  char s[5];
  int data[] = {10,12,15,20,8,5,18,25,20};
  int grDriver = DETECT, grMode, grErr;

  initgraph(&grDriver, &grMode, "C:\\BC5\\BGI");
  grErr = graphresult();
  if (grErr != grOk) {
    printf("BGI error: %s\n",
      grapherrormsg(grErr));
    return grErr;
  }
  unsigned xMax = getmaxx();
  unsigned yMax = getmaxy();
  setfillstyle(SOLID_FILL, WHITE);
  bar(0, 0, xMax, yMax);
  setcolor(GREEN);
  ellipse(250, 80, 0, 360, 100, 30);
  setcolor(BLUE);
  settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 2);
  outtextxy(180, 60, "Grafik batang");
  setcolor(RED);
  line(50, yMax-50, xMax-50, yMax-50);
  line(50, 50, 50, yMax-50);
  setcolor(BLACK);
  settextstyle(DEFAULT_FONT, VERT_DIR, 1);
  for (int i=0; i<=30; i=i+5) {
    sprintf(s, "%d", i);
    outtextxy(40, yMax-55-10*i, s);
  }
  settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
  for (i=0; i<10; i++) {
    sprintf(s, "%d", i);
    outtextxy(50*(i+1)+25, yMax-40, s);
    setfillstyle(SOLID_FILL, i);
    bar(50*(i+1), yMax-50, 50*(i+2),
      yMax-50-10*data[i]);
  }
  getch();
  closegraph();
  return 0;
  }
```


TC
Auto
Grafik batang
0
5
10
15
20
25
30
0
1
2
3
4
5
6
7
8
9

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab I
Berkenalan dengan Bahasa C

1 Sejarah

Bahasa C merupakan perkembangan dari bahasa BCPL yang dikembangkan oleh Martin Richards pada tahun 1967. Selanjutnya bahasa ini memberikan ide kepada Ken Thompson yang kemudian mengembangkan bahasa yang disebut bahasa B pada tahun 1970. Perkembangan selanjutnya dari bahasa B adalah bahasa C oleh Dennis Ricthie sekitar tahun 1970-an di Bell Telephone Laboratories Inc. (sekarang adalah AT&T Bell Laboratories). Bahasa C pertama kali digunakan di computer Digital Equipment Corporation PDP-11 yang menggunakan system operasi UNIX. Hingga saat ini penggunaan bahasa C telah merata di seluruh dunia. Hampir semua perguruan tinggi di dunia menjadikan bahasa C sebagai salah satu mata kuliah wajib. Selain itu, banyak bahasa pemrograman populer seperti PHP dan Java menggunakan sintaks dasar yang mirip bahasa C. Oleh karena itu, kita juga sangat perlu mempelajarinya.

2 Kelebihan dan Kekurangan Bahasa C

- **Kelebihan Bahasa C**
  - Bahasa C tersedia hampir di semua jenis computer.
  - Kode bahasa C sifatnya adalah portable dan fleksibel untuk semua jenis computer.
  - Bahasa C hanya menyediakan sedikit kata-kata kunci, hanya terdapat 32 kata kunci.
  - Proses executable program bahasa C lebih cepat

- Dukungan pustaka yang banyak.
- C adalah bahasa yang terstruktur
- Bahasa C termasuk bahasa tingkat menengah

Kekurangan Bahasa C

- Banyaknya Operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
- Bagi pemula pada umumnya akan kesulitan menggunakan pointer

3 Mengenal Editor Bahasa C

Memulai Bahasa C

Buka Editor Bahasa C yang ada, seperti Bordland C, Turbo C, dan sebagainya. Semua program yang ada di tutorial ini bisa dicoba Turbo C.

Sekilas Mengenai Editor Turbo C

Untuk mengkompilasi Program, langkah-langkahnya sbb :

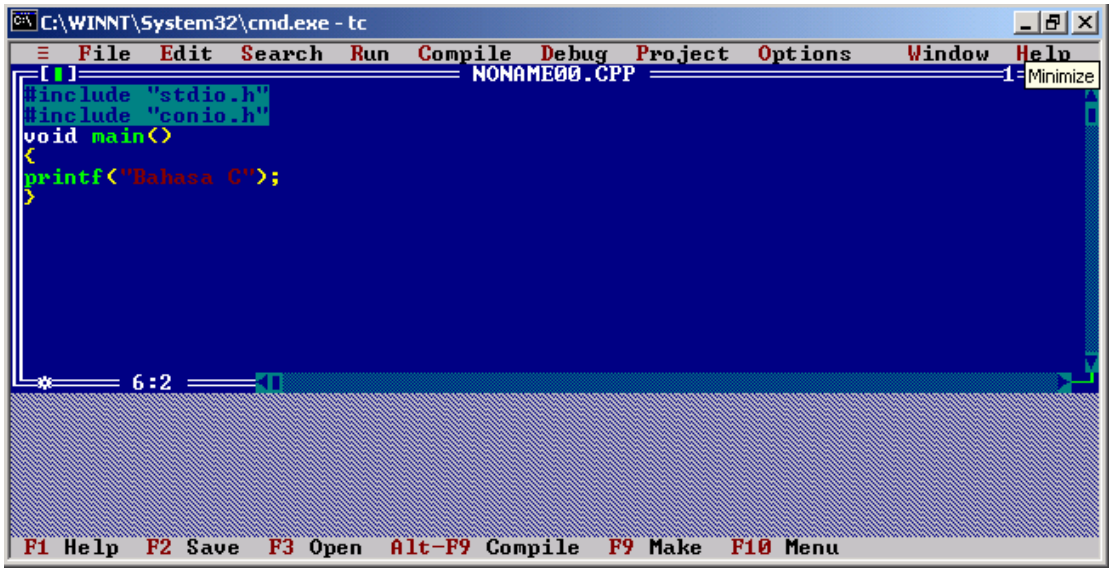
- Pilih menu Compile dengan menekan **Alt + C**
- Pilih Submenu **Compile**
- Enter

Akan ditampilkan hasil kompilasi Program, tekan sembarang tombol

Untuk menjalankan program :

- Pilih menu Run dengan menekan **Alt + R**
- Pilih submenu **Run** dan tekan Enter

Menu-menu dalam Turbo C :



Tampilan Menu Editor Turbo C

- **File** , terdiri dari :
  (1) **New**, untuk memulai program baru
  (2) **Open**, untuk mengambil atau membuka program
  (3) **Save**, untuk menyimpan file/program
  (4) **Save as**, untuk menyimpan file/program
  (5) **Save all,** untuk menyimpan seluruh file/program
  (6) **Change dir**, untuk mengubah directory

(7) **Print,** untuk mencetak program
(8) **DOS Shell**, untuk menuju ke DOS Shell
(9) **Quit**, untuk keluar dari Turbo C

- **Edit**, terdiri dari :
  (1) **Undo**, untuk membatalkan pengeditan terakhir
  (2) **Redo,** untuk kembali ke pengeditan terakhir yang telah di undo.
  (3) **Cut,** untuk memotong bagian tertentu dari program.
  (4) **Copy**, untuk menduplikasi bagian program
  (5) **Paste**
  (6) **Clear,** untuk menghapus bagian tertentu dari program
  (7) **Copy example**
  (8) **Show Clipboard**
- **Search,** terdiri dari :
  (1) **Find…**
  (2) **Replace…**
  (3) **Search again**
  (4) **Previous error**
  (5) **Next error**
  (6) **Locate function…**
- **Run**, terdiri dari :
  (1) **Run…**, untuk menjalankan program
  (2) **Program reset**
  (3) **Go to cursor**
  (4) dst
- **Compile**, terdiri dari :
  (1) **Compile**, untuk mengkompilasi program
  (2) **Make**
  (3) **Link**
  (4) **Build all, dst**
- **Debug**, terdiri dari
  (1) **Inspect**
  (2) **Evaluate/modify**
  (3) Dst
- **Project**, terdiri dari :
  (1) **Open project**
  (2) **Close project**
  (3) dst
- **Options**, terdiri dari :
  (1) **Application**
  (2) **Compiler**
  (3) **Transfer**
  (4) Dst
- **Window**, terdiri dari :
  (1) **Size/Move**
  (2) **Zoom**
  (3) **Tile**
  (4) **Cascade**
  (5) **Next**
  (6) dst
- **Help**, terdiri dari
  (1) **Contens**
  (2) **Index**
  (3) **Topic search**
  (4) **Previous topic**
  (5) dst

4 Penulisan Program Bahasa C

Program Bahasa C tidak mengenal aturan penulisan di kolom tertentu, jadi bisa dimulai dari kolom manapun. Namun demikian, untuk mempermudah pembacaan program dan untuk keperluan dokumentasi, sebaiknya penulisan bahasa C diatur sedemikian rupa sehingga mudah dan enak dibaca.

Berikut contoh penulisan Program Bahasa C yang baik dan yang kurang baik :

```
#include “stdio.h”
void main()
{
    printf(“Bahasa C\n”);
}
```

```
#include “stdio.h”
void main() { printf(“Bahasa C”); }
```

Kedua Program di atas bila dijalankan akan menghasilkan hasil yang sama berupa tulisan “Bahasa C” di layar, namun dari segi penulisannya program yang pertama tampaknya lebih mudah dibaca dan lebih rapih dibanding dengan program yang kedua.

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab II
Struktur Dasar Bahasa C

1 Tipe Data

Tipe data merupakan bagian program yang paling penting karena tipe data mempengaruhi setiap instruksi yang akan dilaksanakan oleh computer. Misalnya saja 5 dibagi 2 bisa saja menghasilkan hasil yang berbeda tergantung tipe datanya. Jika 5 dan 2 bertipe integer maka akan menghasilkan nilai 2, namun jika keduanya bertipe float maka akan menghasilkan nilai 2.5000000. Pemilihan tipe data yang tepat akan membuat proses operasi data menjadi lebih efisien dan efektif.

Dalam bahasa C terdapat lima tipe data dasar, yaitu :

No	Tipe Data	Ukuran	Range (Jangkauan)	Format	Keterangan
1	char	1 byte	- 128 s/d 127	%c	Karakter/string
2	int	2 byte	- 32768 s/d 32767	%i , %d	Integer/bilangan bulat
3	float	4 byte	- 3.4E-38 s/d 3.4E+38	%f	Float/bilangan pecahan
4	double	8 byte	- 1.7E-308 s/d 1.7+308	%lf	Pecahan presisi ganda
5	void	0 byte	-	-	Tidak bertipe

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
void main()
```

```
{   int x;
    float y;
    char z;
    double w;
    clrscr();                   /* untuk membersihkan layar */
    x = 10;                     /* variable x diisi dengan 10 */
    y = 9.45;                   /* variable y diisi dengan 9.45 */
    z = 'C';                    /* variable z diisi dengan karakter "C" */
    w = 3.45E+20;               /* variable w diisi dengan 3.45E+20 */
    printf("Nilai dari x adalah : %i\n", x); /* Menampilkan isi variable x */
    printf("Nilai dari y adalah : %f\n", y); /* Menampilkan isi variable y */
    printf("Nilai dari z adalah : %c\n", z); /* Menampilkan isi variable z */
    printf("Nilai dari w adalah : %lf\n", w);       /* Menampilkan isi variable w */
    getch(); }
```

2 Konstanta

Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta nilainya selalu tetap. Konstanta harus didefinisikan terlebih dahulu di awal program. Konstanta dapat bernilai integer, pecahan, karakter dan string. Contoh konstanta : 50; 13; 3.14; 4.50005; 'A'; 'Bahasa C'. Selain itu, bahasa C juga menyediakan beberapa karakter khusus yang disebut karakter escape, antara lain :

- \a : untuk bunyi bell (alert)
- \b : mundur satu spasi (backspace)
- \f : ganti halaman (form feed)
- \n : ganti baris baru (new line)
- \r : ke kolom pertama, baris yang sama (carriage return)
- \v : tabulasi vertical
- \0 : nilai kosong (null)
- \' : karakter petik tunggal
- \" : karakter petik ganda
- \\ : karakter garis miring

3 Variable

Variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu variable bisa diubah-ubah sesuai kebutuhan. Nama dari suatu variable dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut :

- Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf. Bahasa C bersifat case-sensitive artinya huruf besar dan kecil dianggap berbeda. Jadi antara **nim, NIM** dan **Nim** dianggap berbeda.
- Tidak boleh mengandung spasi.
- Tidak boleh mengandung symbol-simbol khusus, kecuali garis bawah (underscore). Yang termasuk symbol khusus yang tidak diperbolehkan antara lain : $, ?, %, #, !, &, *, (,), -, +, = dsb
- Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.
  Contoh penamaan variabel yang benar :
  NIM, a, x, nama_mhs, f3098, f4, nilai, budi, dsb.
  Contoh penamaan variable yang salah :
  %nilai_mahasiswa, 80mahasiswa, rata-rata, ada spasi, penting!, dsb

4 Deklarasi

Deklarasi diperlukan bila kita akan menggunakan pengenal (identifier) dalam program. Identifier dapat berupa variable, konstanta dan fungsi.

Deklarasi Variabel

Bentuk umum pendeklarasian suatu variable adalah :

```
Nama_tipe nama_variabel;
```

Contoh :

```
int x;                          // Deklarasi x bertipe integer
char y, huruf, nim[10];         // Deklarasi variable bertipe char
float nilai;                    // Deklarasi variable bertipe float
double beta;            // Deklarasi variable bertipe double
int array[5][4];                // Deklarasi array bertipe integer
char *p;                        // Deklarasi pointer p bertipe char
```

Deklarasi Konstanta

Dalam bahasa C konstanta dideklarasikan menggunakan preprocessor #define. Contohnya :

```
#define PHI 3.14
#define nim "0111500382"
#define nama "Sri Widhiyanti"
```

Deklarasi Fungsi

Fungsi merupakan bagian yang terpisah dari program dan dapat diaktifkan atau dipanggil di manapun di dalam program. Fungsi dalam bahasa C ada yang sudah disediakan sebagai fungsi pustaka seperti printf(), scanf(), getch() dan untuk menggunakannya tidak perlu dideklarasikan. Fungsi yang perlu dideklarasikan terlebih dahulu adalah fungsi yang dibuat oleh programmer. Bentuk umum deklarasi sebuah fungsi adalah :

```
Tipe_fungsi nama_fungsi(parameter_fungsi);
```

Contohnya :

```
float luas_lingkaran(int jari);
void tampil();
int tambah(int x, int y);
```

5 Operator

Operator Penugasan

Operator Penugasan (*Assignment operator*) dalam bahasa C berupa tanda sama dengan ("="). Contoh :

```
nilai = 80;
A = x * y;
```

Artinya : variable "nilai" diisi dengan 80 dan variable "A" diisi dengan hasil perkalian antara x dan y.

Operator Aritmatika

Bahasa C menyediakan lima operator aritmatika, yaitu :

- * : untuk perkalian
- / : untuk pembagian
- % : untuk sisa pembagian (modulus)
- + : untuk pertambahan
- - : untuk pengurangan

Catatan : operator % digunakan untuk mencari sisa pembagian antara dua bilangan.
Misalnya :
9 % 2 = 1
9 % 3 = 0

9 % 5 = 4
9 % 6 = 3

Contoh Program 1 :

```
#include "stdio.h"
#include "conio.h"
void main()
{   clrscr();              // untuk membersihkan layar
    printf("Nilai dari 9 + 4 = %i", 9 + 4);          /* mencetak hasil 9 + 4 */
    printf("Nilai dari 9 - 4 = %i", 9 - 4);          /* mencetak hasil 9 - 4 */
    printf("Nilai dari 9 * 4 = %i", 9 * 4);          /* mencetak hasil 9 * 4 */
    printf("Nilai dari 9 / 4 = %i", 9 / 4);          /* mencetak hasil 9 / 4 */
    printf("Nilai dari 9 \% 4 = %i", 9 % 4);         /* mencetak hasil 9 % 4 */
    getch();
}
```

Contoh Program 2 :

```
/* Penggunaan operator % untuk mencetak deret bilangan genap antara 1 – 100 */
#include "stdio.h"
#include "conio.h"
void main()
{   int bil;
    clrscr();                                   // untuk membersihkan layar
    for (bil=1; bil<100; bil++)
    {   if(bil % 2 == 0)                  //periksa apakah 'bil' genap
            printf("%5.0i", bil);
    }
    getch();
}
```

- **Operator Hubungan (Perbandingan)**
  Operator Hubungan digunakan untuk membandingkan hubungan antara dua buah operand (sebuah nilai atau variable. Operator hubungan dalam bahasa C :

Operator	Arti	Contoh	
<	Kurang dari	x < y	Apakah x kurang dari y
<=	Kurang dari sama dengan	x <= y	Apakah x kurang dari sama dengan y
>	Lebih dari	x > y	Apakah x lebih dari y
>=	Lebih dari sama dengan	x >= y	Apakah x lebih dari sama dengan y
==	Sama dengan	x == y	Apakah x sama dengan y
!=	Tidah sama dengan	x != y	Apakah x tidak sama dengan y

- **Operator Logika**
  Jika operator hubungan membandingkan hubungan antara dua buah operand, maka operator logika digunakan untuk membandingkan logika hasil dari operator-operator hubungan. Operator logika ada tiga macam, yaitu :
  - && : Logika AND (DAN)
  - || : Logika OR (ATAU)
  - ! : Logika NOT (INGKARAN)

- **Operator Bitwise**
  Operator bitwise digunakan untuk memanipulasi bit-bit dari nilai data yang ada di memori. Operator bitwise dalam bahasa C :
  - << : Pergeseran bit ke kiri

- >> : Pergeseran bit ke kanan
- & : Bitwise AND
- ^ : Bitwise XOR (exclusive OR)
- | : Bitwise OR
- ~ : Bitwise NOT

Operator Unary

Operator Unary merupakan operator yang hanya membutuhkan satu operand saja. Dalam bahasa C terdapat beberapa operator unary, yaitu :

Operator	Arti/Maksud	Letak	Contoh	Equivalen
-	Unary minus	Sebelum operator	A + -B * C	A + (-B) * C
++	Peningkatan dengan penambahan nilai 1	Sebelum dan sesudah	A++	A = A + 1
--	Penurunan dengan pengurangan nilai 1	Sebelum dan sesudah	A--	A = A – 1
sizeof	Ukuran dari operand dalam byte	Sebelum	sizeof(I)	-
!	Unary NOT	Sebelum	!A	-
~	Bitwise NOT	Sebelum	~A	-
&	Menghasilkan alamat memori operand	Sebelum	&A	-
*	Menghasilkan nilai dari pointer	Sebelum	*A	-

Catatan Penting ! :

Operator peningkatan ++ dan penurunan -- jika diletakkan sebelum atau sesudah operand terdapat perbedaan. Perhatikan contoh berikut :

Contoh Program 1 :

```
/* Perbedaan operator peningkatan ++ yang diletakkan di depan dan dibelakang operand */
        #include <stdio.h>
        #include <conio.h>
        void main()
        {
          int x, nilai;
            clrscr();
          x = 5;
              nilai = ++x;   /* berarti x = x + 1; nilai = x;  */
          printf("nilai = %d, x = %d\n", nilai, x);
          nilai = x++;       /* berarti nilai = x;  nilai = x + 1;  */
          printf("nilai = %d, x = %d\n", nilai, x);
            getch();
        }
```

Contoh Program 2 :

```
#include “stdio.h”
#include “conio.h”
void main()
{   int b, nilai;
     clrscr();                  // untuk membersihkan layar
     b = 15;
     nilai = --b;    /* berarti b = b - 1; nilai = b;  */
```

```
    printf("nilai = %d, b = %d\n", nilai, b);
    nilai = b--;    /* berarti nilai = b;  b = b + 1;  */
    printf("nilai = %d, b = %d\n", nilai, b);
    getch();
}
```

6 Kata Tercadang (Reserved Word)

Bahasa C standar ANSI memiliki 32 kata tercadang (reserved word) dan Turbo C menambahkannya dengan 7 kata tercadang. Semua *reserved word* tidak boleh digunakan dalam penamaan identifier (variable, nama fungsi dll). Kata Tercadang yang tersedia dalam bahasa C adalah sbb:

*asm	default	for	*pascal	switch
auto	do	goto	register	typedef
break	double	*huge	return	union
case	else	if	short	unsigned
*cdecl	enum	int	signed	void
char	extern	*interrupt	sizeof	volatile
const	*far	long	static	while
continue	float	*near	struct	

Keterangan : tanda * menunjukkan tambahan dari Turbo C

7 Komentar Program

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan komentar atau penjelasan dalam bahasa C digunakan pembatas /* dan */ atau menggunakan tanda // untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan).

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
void main()
{
    clrscr();           /* Ini untuk membersihkan layar tampilan */
    printf(“Contoh Penggunaan Komentar”);   //komentar tidak ikut diproses
    getch();            /* Dan ini untuk menahan tampilan di layer */
}
```

Latihan 2

Buatlah Program dalam Bahasa C untuk :

1. Mencari jawaban dari sebuah persamaan kuadrat dengan menggunakan rumus abc.
2. Mencetak deret bilangan 1 2 4 8 16 32 64
3. Menghitung jumlah dan rata-rata dari 5 buah bilangan bulat positip.

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab III
Input dan Output

1 MEMASUKKAN DATA

Dalam bahasa C proses memasukkan suatu data bisa menggunakan beberapa fungsi pustaka yang telah tersedia. Beberapa fungsi pustaka yang bisa digunakan adalah :

- **scanf()**
  - Fungsi pustaka scanf() digunakan untuk menginput data berupa data numerik, karakter dan string secara terformat.
  - Hal-hal yang perlu diperhatikan dalam pemakaian fungsi scanf() :
    - ☑ Fungsi scanf() memakai penentu format
    - ☑ Fungsi scanf() memberi pergantian baris secara otomatis
    - ☑ Fungsi scanf() tidak memerlukan penentu lebar field
    - ☑ Variabelnya harus menggunakan operator alamat &

Kode penentu format :

- %c : Membaca sebuah karakter
- %s : Membaca sebuah string
- %i, %d : Membaca sebuah bilangan bulat (integer)
- %f, %e : Membaca sebuah bilangan pecahan (real)
- %o : membaca sebuah bilangan octal
- %x : Membaca sebuah bilangan heksadesimal
- %u : Membaca sebuah bilangan tak bertanda

Contoh Program :

```
/* Program memasukan inputan dengan beberapa tipe data */
#include <stdio.h>
#include <conio.h>
void main()
{    int jumlah;
     char huruf, nim[10];
     float nilai;
     clrscr();

     printf("Masukkan sebuah bilangan bulat :  ");
     scanf("%d", &jumlah );                    /* membaca sebuah bilangan bulat */
     printf("Masukkan sebuah karakter :  ");
     scanf("%c", &huruf );                     /* membaca sebuah karakter */
     printf("Masukkan nim Anda : ");
     scanf("%s", &nim );                  /* membaca sebuah string */
     printf("Masukkan sebuah bilangan pecahan :  ");
     scanf("%f", &nilai );                /* membaca sebuah bilangan float */


     printf("\nNilai variable yang Anda masukkan adalah :\n");
     printf("jumlah = %d\n", jumlah );
     printf("huruf   = %c\n", huruf );
     printf("nim     = %s\n", nim );
     printf("nilai    = %f\n", nilai );
     getch();
}
```

- **gets()**
  - Fungsi gets() digunakan untuk memasukkan data bertipe karakter dan tidak dapat digunakan untuk memasukkan data numerik.
  - Harus diakhiri dengan penekanan tombol enter
  - Cursor secara otomatis akan pindah baris
  - Tidak memerlukan penentu format

Contoh Program :

```
/* Program inputan tipe data karakter/string */
#include “stdio.h”
#include “conio.h”
void main()
{
     char nama[20];
     clrscr();
     printf(“Masukkan nama Anda : “);
     gets(nama);
     printf(“Hello, Nama Anda adalah %s”, nama);
     getch();
}
```

- **getchar()**
  - Fungsi getchar() digunakan untuk membaca data yang bertipe karakter
  - Harus diakhiri dengan penekanan tombol enter
  - Karakter yang dimasukkan terlihat pada layar
  - Pergantian baris secara otomatis

- **getch() dan getche()**
  - Fungsi getch() dan getche() digunakan untuk membaca data karakter.
  - Karakter yang dimasukkan tidak perlu diakhiri dengan penekanan tombol enter.
  - Tidak memberikan efek pergantian baris secara otomatis
  - Jika menggunakan fungsi getch() karakter yang dimasukkan tidak akan ditampilkan pada layer sehingga sering digunakan untuk meminta inputan berupa password.
  - Sedangkan pada getche() karakter yang dimasukkan akan ditampilkan pada layar.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{
    char huruf1, huruf2;
    printf("Masukkan sebuah karakter : ");
    huruf1 = getche();  // karakter yang dimasukkan akan terlihat di layar
    printf("\nKarakter yang Anda masukkan adalah %c\n", huruf1);
    printf("\nMasukkan sebuah karakter lagi : ");
    huruf2 = getch();   // karakter yang dimasukkan tidak terlihat di layar
    printf("\nKarakter yang Anda masukkan adalah : %c, huruf2);
    getch();
}
```

CATATAN :

Jika terdapat beberapa proses input (memasukkan data) sekaligus, maka sebaiknya ditambahkan fungsi **fflush(stdin);** setelah fungsi scanf(). Fungsi fflush(stdin) berfungsi menghapus buffer di dalam alat I/O.

2 MENAMPILKAN DATA

Menampilkan data ke layer monitor

- Menggunakan fungsi printf(), puts(), dan putchar().
- Fungsi printf() digunakan untuk menampilkan semua jenis data (numeric dan karakter)
- Fungsi puts() digunakan untuk menampilkan data string dan secara otomatis akan diakhiri dengan perpindahan baris.
- Fungsi putchar() digunakan untuk menampilkan sebuah karakter.

Mengatur tampilan bilangan pecahan (float).

Bentuk umum :

```
printf("%m.nf", argument);
```

- m : menyatakan panjang range
- n : menyatakan jumlah digit di belakang koma.
- argument : nilai atau variable yang akan ditampilkan.

Contoh :

```
printf("%5.2f", nilai);
```

artinya variable **nilai** akan ditampilkan sebanyak 5 digit dengan 2 digit di belakang koma.

Contoh Program 1;

```
/* Program untuk menampilkan data berupa bilangan pecahan */
#include "stdio.h"
#include "conio.h"
void main()
```

```
{    float nilai;
     clrscr();
     puts("Masukkan nilai Anda : ); scanf("%f", &nilai);
     printf("Anda memperoleh nilai %5.2f", nilai);
     printf("Apakah Anda telah puas mendapat nilai %6.4f ?", nilai);
     getch();
}
```

Contoh Program 2;

```
/* Program untuk menampilkan data berupa bilangan integer dan string */
#include "stdio.h"
#include "conio.h"
void main()
{    int umur;
     char nama[20];
     clrscr();
     puts("Masukkan nama Anda : ); gets(nama);
     puts("Masukkan umur Anda : "); scanf("%d", &umur);
     printf("Nama Anda : %s \n", nama);          //tipe data string
     printf("Umur Anda  : %d \n", umur);                 //tipe data integer
     getch();
}
```

Menampilkan data ke printer

- Untuk menampilkan data ke printer dapat menggunakan fungsi fprintf(), fputs() dan fputc().
- Fungsi fprintf() digunakan untuk mencetak semua jenis tipe data ke printer dan secara otomatis memberikan efek perpindahan baris.
- Fungsi fputs() digunakan untuk mencetak tipe data string ke printer
- Fungsi fputc() digunakan untuk mencetak tipe data karakter ke printer

Contoh program :

```
#include "stdio.h"
#include "conio.h"
void main()
{
     fprintf(stdprn, "Hallo, Saya akan tercetak di printer");
     fputs(stdprn, "Saya juga akan tercetak di printer");
}
```

LATIHAN 3

Buatlah Program dalam Bahasa C untuk :

1. Menginput dan menampilkan biodata pribadi seseorang yang terdiri dari nama, tempat, tanggal lahir, alamat, nomor telepon, agama, dan jenis kelamin.
2. Mencetak sejumlah deret bilangan ganjil antara 1 sampai N, dimana N dimasukkan oleh user.
3. Menentukan bilangan terbesar dan terkecil dari sejumlah bilangan yang dimasukkan oleh user (misalnya N buah bilangan).

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab IV
Penyeleksian Kondisi

Penyeleksian kondisi digunakan untuk mengarahkan perjalanan suatu proses. Penyeleksian kondisi dapat diibaratkan sebagai katup atau kran yang mengatur jalannya air. Bila katup terbuka maka air akan mengalir dan sebaliknya bila katup tertutup air tidak akan mengalir atau akan mengalir melalui tempat lain. Fungsi penyeleksian kondisi penting artinya dalam penyusunan bahasa C, terutama untuk program yang kompleks.

1 STRUKTUR KONDISI "IF…."

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisi tunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang ada di dalam blok if akan diproses dan dikerjakan. Bentuk umum struktur kondisi if adalah :

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

Contoh Program 1 :

```
/* Program struktur kondisi if untuk memeriksa suatu kondisi */
#include "stdio.h"
#include "conio.h"
void main()
{   float nilai;
```

```
    printf("Masukan nilai yang didapat : ");
    scanf("%f", &nilai);
    if(nilai > 65)
    printf("\n ANDA LULUS !!!!\n");
    getch();
}
```

Bila program tersebut dijalankan dan kita memasukan nilai 80, maka perintah mencetak perkataan LULUS !!!! akan dilaksanakan, namun sebaliknya bila kita memasukan sebuah nilai yang kurang dari 65 maka program akan berhenti dan tidak dihasilkan apa-apa.

Contoh Program 2 :

```
/* Program contoh penerapan struktur kondisi if */
#include"stdio.h"
#include"conio.h"

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

    printf("Entry bil 1 : ");fflush(stdin);scanf("%i",&a);
    printf("Entry bil 2 : ");fflush(stdin);scanf("%i",&b);
    printf("Entry bil 3 : ");fflush(stdin);scanf("%i",&c);

    if((a>b)&&(a>c))
      max=a;

    if((b>a)&&(b>c))
      max=b;

    if((c>a)&&(c>b))
      max=c;

    printf("Bil terbesar : %i\n",max);

    if(max>0)
      printf("Bil tsb adalah bil positif\n");

    if(max<0)
      printf("Bil tsb adalah bil negatif");
    getch();
}
```

2 STRUKTUR KONDISI "IF.....ELSE…."

Dalam struktur kondisi if.....else minimal terdapat dua pernyataan. Jika kondisi yang diperiksa bernilai benar atau terpenuhi maka pernyataan pertama yang dilaksanakan dan jika kondisi yang diperiksa bernilai salah maka pernyataan yang kedua yang dilaksanakan. Bentuk umumnya adalah sebagai berikut :

```
if(kondisi)
    pernyataan-1
else
    pernyataan-2
```

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   float nilai;
    clrscr();

    printf("Masukan nilai yang didapat : ");
    scanf("%f", &nilai);        /* Masukan akan disimpan dalam variable nilai */

    if (nilai > 65)
        printf("\n LULUS !!!\n");
    else
        printf("\n TIDAK LULUS !!!\n");

    getch();
}
```

Bila program tersebut dijalankan dan kita memasukan nilai 80 maka akan dicetak perkataan "LULUS !!!" namun bila kita memasukan nilai yang kurang dari 65 maka akan tercetak perkataan "TIDAK LULUS !!!". Hal ini berbeda dengan struktur if dimana program akan berhenti bila kita memasukan nilai kurang dari 65.

3 STRUKTUR KONDISI "SWITCH....CASE....DEFAULT…"

Struktur kondisi switch....case....default digunakan untuk penyeleksian kondisi dengan kemungkinan yang terjadi cukup banyak. Struktur ini akan melaksanakan salah satu dari beberapa pernyataan 'case' tergantung nilai kondisi yang ada di dalam switch. Selanjutnya proses diteruskan hingga ditemukan pernyataan 'break'. Jika tidak ada nilai pada case yang sesuai dengan nilai kondisi, maka proses akan diteruskan kepada pernyataan yang ada di bawah 'default'.

Bentuk umum dari struktur kondisi ini adalah :

```
switch(kondisi)
{
    case 1 : pernyataan-1;
             break;
    case 2 : pernyataan-2;
             break;
    .....
    .....
    case n : pernyataan-n;
             break;
    default : pernyataan-m
}
```

Contoh Program :

```
/* Program menentukan nama hari berdasarkan inputan */
#include "stdio.h"
#include "conio.h"
void main()
{   clrscr();
    int hari;

    puts("Menentukan nama hari\n");
    puts("1 = Senin      2 = Selasa      3 = Rabu      4 = Kamis");
```

```
    puts("5 = Jum'at      6  = Sabtu          7 = Minggu");
    printf("\nMasukan kode hari( 1-7) : ");
    scanf("%d", &hari);
    switch(hari)
    {   case 1 : puts("Hari Senin");                    /* kemungkinan pertama */
                 break;
         case 2 : puts("Hari Selasa");        /* kemungkinan kedua */
                 break;
         case 3 : puts("Hari Rabu");                    /* kemungkinan ketiga */
                 break;
         case 4 : puts("Hari Kamis");         /* kemungkinan keempat */
                 break;
         case 5 : puts("Hari Jum'at");        /* kemungkinan kelima */
                 break;
         case 6 : puts("Hari Sabtu");         /* kemungkinan keenam */
                 break;
         case 7 : puts("Hari Minggu");        /* kemungkinan ketujuh */
                 break;
         default : puts("Kode hari yang Anda masukan SALAH");
    }
    getch();
}
```

Bila program tersebut dijalankan, dan kita memasukan kode hari dengan 1 maka akan tercetak "Hari Senin", bila 2 akan tercetak "Hari Selasa" dan seterusnya.

LATIHAN 4

Buatlah sebuah Program untuk mencetak bilangan terbesar dari 5 buah bilangan yang dimasukkan oleh user, dengan cara membandingkan bilangan sebelumnya dengan bilangan berikutnya. Misalnya bilangan tersebut A, B, C, D, dan E maka A dan B diperbandingkan. Jika A lebih besar dari B maka A dibandingkan dengan C, jika A lebih besar dari C maka A dibandingkan dengan D, demikian seterusnya sampai didapat nilai yang terbesar.

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab V
Perulangan

Dalam bahasa C tersedia suatu fasilitas yang digunakan untuk melakukan proses yang berulang-ulang sebanyak keinginan kita. Misalnya saja, bila kita ingin menginput dan mencetak bilangan dari 1 sampai 100 bahkan 1000, tentunya kita akan merasa kesulitan. Namun dengan struktur perulangan proses, kita tidak perlu menuliskan perintah sampai 100 atau 1000 kali, cukup dengan beberapa perintah saja. Struktur perulangan dalam bahasa C mempunyai bentuk yang bermacam-macam.

1 STRUKTUR PERULANGAN " WHILE"

Perulangan WHILE banyak digunakan pada program yang terstruktur. Perulangan ini banyak digunakan bila jumlah perulangannya belum diketahui. Proses perulangan akan terus berlanjut selama kondisinya bernilai benar (true) dan akan berhenti bila kondisinya bernilai salah.

Contoh Program 1 :

```
/* Program Perulangan menggunakan while */
#include "stdio.h"
#include "conio.h"
void main()
{    int x;
     x = 1;                                  /* awal variabel */
     while (x <= 10)              /* Batas akhir perulangan */
     {   printf("%d BAHASA C\n", x);
```

```
        x ++;                          /* variabel x ditambah dengan 1 */
    }
    getch();
}
```

Jika program tersebut dijalankan maka akan menghasilkan hasil sebagai berikut

```
1BAHASA C
2BAHASA C
3BAHASA C
4BAHASA C
5BAHASA C
6BAHASA C
7BAHASA C
8BAHASA C
9BAHASA C
10BAHASA C
```

Pada perulangan while di atas, proses atau perintah mencetak kata-kata "BAHASA C" akan terus dilakukan selama variabel x masih kurang atau sama dengan 10. Setiap kali melakukan perulangan, nilai dari variabel x akan bertambah 1.

Contoh Program 2 :

```
/* Program mencetak deret bilangan dengan menggunakan while */
#include"stdio.h"
#include"conio.h"

void main()
{   clrscr();
    int i=1,x;

    while(i<=3)
    {   x=1;
        while(x<=i)
        {   printf("%3i",x);
            x=x+1;
        }
        printf("\n");
        i=i+1;
    }
    getch();
}
```

2 STRUKTUR PERULANGAN "DO.....WHILE…"

Pada dasarnya struktur perulangan do....while sama saja dengan struktur while, hanya saja pada proses perulangan dengan while, seleksi berada di while yang letaknya di atas sementara pada perulangan do....while, seleksi while berada di bawah batas perulangan. Jadi dengan menggunakan struktur do…while sekurang-kurangnya akan terjadi satu kali perulangan.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   int x;
    x = 1;
```

```
do
{   printf("%d BAHASA C\n", x);
    x ++;
}
while(x <= 10);
getch(); }
```

3 STRUKTUR PERULANGAN "FOR"

Struktur perulangan for biasa digunakan untuk mengulang suatu proses yang telah diketahui jumlah perulangannya. Dari segi penulisannya, struktur perulangan for tampaknya lebih efisien karena susunannya lebih simpel dan sederhana. Bentuk umum perulangan for adalah sebagai berikut :

```
for(inisialisasi; syarat; penambahan)
  pernyataan;
```

Keterangan :

- ***Inisialisasi*** : pernyataan untuk menyatakan keadaan awal dari variabel kontrol.
- ***syarat*** : ekspresi relasi yang menyatakan kondisi untuk keluar dari perulangan.
- ***penambahan*** : pengatur perubahan nilai variabel kontrol.

Contoh Program 1 :

```
/* Program perulangan menggunakan for */
#include "stdio.h"
#include "conio.h"
void main()
{   int x;
    for(x = 1; x<= 10; x++)
    {   printf("%d BAHASA C\n", x);  }
    getch();
}
```

Contoh Program 2 :

```
/* Mencari total dan rata-rata sejumlah bilangan menggunakan for */
#include"stdio.h"
#include"conio.h"

void main()
{   clrscr();
    float r,i,x,t=0;int y;


    for(y=1;y<=3;y++)
        for(i=0;i<=2;i++)
        {   printf("Entry bilangan %i : ",y);scanf("%f",&x);
            t=t+x;
            y=y+1;
        }

    printf("\n Total : %.2f",t);
```

```
    r=t/i;
    printf("\n Rata rata : %.2f",r);

    getch();
}
```

LATIHAN 5

1. Buatlah Program untuk mencetak tampilan sebagai berikut :

```
*********
********
*******
******
*****
****
***
**
*
```

Gunakan perulangan while atau for..!

2. Buatlah Program untuk mencetak 10 bilangan prima pertama.
2 3 5 7 13 17….

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab VI
Array (Larik)

Array merupakan kumpulan dari nilai-nilai data yang bertipe sama dalam urutan tertentu yang menggunakan nama yang sama. Letak atau posisi dari elemen array ditunjukkan oleh suatu index. Dilihat dari dimensinya array dapat dibagi menjadi Array dimensi satu, array dimensi dua dan array multi-dimensi.

1 ARRAY DIMENSI SATU

- Setiap elemen array dapat diakses melalui indeks.
- Indeks array secara default dimulai dari 0.
- Deklarasi Array

  Bentuk umum :

  Tipe_array nama_array[ukuran];

  Contoh :

int Nilai[5];

Nilai[0]	Nilai[1]	Nilai[2]	Nilai[3]	Nilai[4]
70	80	82	60	75

Contoh Program :

```
/* Program untuk menginput nilai mahasiswa ke dalam array satu dimensi */
#include “stdio.h”
#include “conio.h”
```

```
void main();
{    int index, nilai[10];
     clrscr();

     /* input nilai mahasiswa */
     printf("Input nilai 10 mahasiswa : ");
     for(index=0; index < 10; index++)
     {   printf("Mahasiswa %i : ", index+1);
         scanf("%i", &nilai[index]);
     }
     /* tampilkan nilai mahasiswa */
     printf("Nilai mahasiswa yang telah diinput");
     for(index=0; index < 10; index++)
     {  printf("%5.0i", nilai[index]);
     }
     getch();
}
```

CATATAN :

String juga sebenarnya merupakan array yang bertipe karakter. Jumlah elemen array menyatakan jumlah string.

Contoh aplikasi array satu dimensi :

```
/* Program untuk menentukan jurusan & jenjang mahasiswa berdasarkan NIM*/
#include "stdio.h"
#include "conio.h"
#include "string.h"
void main()
{    char jurusan[25], jenjang[10], nim[10], nama[20];
     crlscr();

     printf("Masukkan  nama Anda  : "); gets(nama);
     printf("Masukkan  NIM Anda    : "); gets(nim);

     /***** cari jurusan *****/
     switch(nim[2])
     {  case '1' : strcpy(jurusan, "Teknik Informatika");
                   break;
        case '2' :  strcpy(jurusan, "Sistem Informasi");
                   break;
        case '3' :  strcpy(jurusan, "Teknik Komputer");
                   break;
        case '4' :  strcpy(jurusan, "Komputerisasi Akuntansi");
                   break;
        default :   printf("Anda salah memasukkan NIM. Coba periksa lagi !");
                   break;
     }

     /***** cari jenjang *****/
     if(nim[4] == '5')
     {   strcpy(jenjang, "Strata-1");
     }
     else
     {  if(nim[4] == '3')
        {  strcpy(jenjang,"Diploma-3");
```

```
        }
        else
            printf("ANda salah memasukkan NIM. Coba periksa lagi !");
    }

    /***** tampilkan data mahasiswa *****/
    printf(" << Data Mahasiswa Universitas Budi Luhur  >>");
    printf("Nama            : %s", nama);
    printf("NIM             : %s", nim);
    printf("Jurusan         : %s", jurusan);
    printf("Jenjang         : %s", jenjang);
    getch();
}
```

2 ARRAY DIMENSI DUA

- Array dua dimensi merupakan array yang terdiri dari m buah baris dan n buah kolom. Bentuknya dapat berupa matriks atau tabel.
- Deklarasi array :

  Tipe_array nama_array[baris][kolom];

  Contoh :

  Int X[3][4];

X[0][0]	X[0][1]	X[0][2]	X[0][3]
X[1][0]	X[1][1]	X[1][2]	X[1][3]
X[2][0]	X[2][1]	X[2][2]	X[2][3]

- Cara mengakses array :

  Untuk mengakses array, misalnya kita ingin mengisi elemen array baris 2 kolom 3 dengan 10 maka perintahnya adalah sbb :

  X[1][2] = 10;
- Untuk mengisi dan menampilkan isi elemen array ada dua cara yaitu :
  - Row Major Order (secara baris per baris)
  - Column Major Order (secara kolom per kolom)

Contoh Program 1 :

```
/* Program menginput nilai(bilangan) ke dalam array dimensi dua dan menampilkannya */

#include "stdio.h"
#include "conio.h"
void main()
{   int baris, kolom, matriks[3][4];
    crlscr();

    // Input elemen array secara Row Major Order
    printf("Input elemen Array : \n");
    for(baris=0; baris<3; baris++)
    {  for(kolom=0; kolom<4; kolom++)
       {  printf("matriks[%i][%i]", baris+1, kolom+1);
          scanf("%i", &matriks[baris][kolom]);
       }
```

```
        printf("\n");
    }

    // Tampilkan elemen Array secara Row Major Order
    printf("Isi array : \n");
    for(baris=0; baris<3; baris++)
    {  for(kolom=0; kolom<4; kolom++)
       {   printf("%i", &matriks[baris][kolom]);
       }
       printf("\n");
    }

    getch();
}
```

Contoh Program 2 :

```
/* Program penjumlahan matriks dua dimensi */
#include "stdio.h"
#include "conio.h"

void main()
{   int A[3][4], B[3][4], X[3][4], Y[3][4], C[3][4], i, j;
    clrscr();

/******* Masukkan matriks A *******/
    for(i=0;i<3;i++)
    {   for(j=0;j<4;j++)
        {   printf("input data matrik A[%i][%i] : ",i+1,j+1);
            fflush(stdin);scanf("%i",&A[i][j]);
        }
    }

/******** Masukkan matriks B ********/
    for(i=0;i<3;i++)
    {   for(j=0;j<4;j++)
        {   printf("input data matrik B[%i][%i] : ",i+1,j+1);
            fflush(stdin);scanf("%i",&B[i][j]);
        }
    }

/******** Proses penjumlahan matriks A dan B ********/
    for(i=0;i<3;i++)
    {   for(j=0;j<4;j++)
        {   X[i][j]=A[i][j]+B[i][j];
        }
    }

/******** Cetak isi matriks A ********/
    printf("\n matrik A\n");
    for(i=0;i<3;i++)
    {   for(j=0;j<4;j++)
        printf("%6i",A[i][j]);
        printf("\n");
    }
```

```
    printf("\n");

/******** Cetak isi matriks B *******/
    printf("\n matrik B\n");
    for(i=0;i<3;i++)
    {   for(j=0;j<4;j++)
        printf("%6i",B[i][j]);printf("\n");
    }
    printf("\n");

/******** Cetak hasil penjumlahan matriks A dan B *******/
    printf("\n matrik penjumlahan A+B\n");
    for(i=0;i<3;i++)
    {   for(j=0;j<4;j++)
        printf("%6i",X[i][j]);printf("\n");
    }
    printf("\n\n");

    getch();
}
```

Contoh aplikasi Array untuk menghitung invers suatu matriks dengan ukuran m x n dengan metode Gauss-Jordan :

```
/* MENGHITUNG INVERS MATRIKS DENGAN METODE GAUSS-JORDAN */
#include "stdio.h"
#include "conio.h"
void main()
{  float p[20], a[20][20], t;
  int m, i, j, k, x;
  clrscr();

  printf("\nMasukkan ukuran matriks : \n");
  scanf("%d", &m);
  printf("\nMasukkan nilai elemen matriks yang akan diinvers");
  printf("\nsecara baris per baris\n");

  /* Membaca matriks asli */
  for(i=1; i<=m; i++)
  {  printf("\n");
    for(j=1; j<=m; j++)
    {  printf("A(%d,%d)= ",i, j);
        scanf("%f", &a[i][j]);
    }
  }

  /* Mencetak Matriks asli */
  printf("\nMatriks asli : ");
  for(i=1; i<=m; i++)
  {  printf("\n");
    for(j=1; j<=m; j++)
        printf("  %.2f", a[i][j]);
  }

  /* Proses inversi */
```

```
  for(i=1; i<=m; i++)
  {  p[i] = a[i][j];
     a[i][j] = 1;
     for(j=1; j<=m; j++)
     {  a[i][j] = a[i][j]/p[i];
     }
     for(k=1; k<=m; k++)
     {  if(k != i)
           {  t = a[k][i];
              a[k][i] = 0;
              for(x=1; x<=m; x++)
                 a[k][x] = a[k][x] - a[i][x] * t;
           }
     }
  }

     /* Mencetak matriks hasil inversi*/
  printf("\n\nMatriks inversi : \n");
  for(i =1; i <=m; i++)
  {  for(j=1; j<=m; j++)
           printf("  %.1f", a[i][j]);
     printf(" \n");
  }
  getch();
}
```

3 ARRAY MULTI-DIMENSI

- Array multi-dimensi merupakan array yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian array sama saja dengan array dimensi satu maupun array dimensi dua.
  Bentuk umumnya yaitu :
  tipe_array nama_array[ukuran1][ukuran2]…[ukuranN];
  Contoh :
  float X[2][4][3];

X[0][0][0]	X[0][0][1]	X[0][0][2]
X[0][1][0]	X[0][1][1]	X[0][1][2]
X[0][2][0]	X[0][2][1]	X[0][2][2]
X[0][3][0]	X[0][3][1]	X[0][3][2]

X[1][0][0]	X[1][0][1]	X[1][0][2]
X[1][1][0]	X[1][1][1]	X[1][1][2]
X[1][2][0]	X[1][2][1]	X[1][2][2]
X[1][3][0]	X[1][3][1]	X[1][3][2]

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
void main()
{  int i, j, k;
     static int data_huruf[2][8][8] =
     {  {  { 1, 1, 1, 1, 1, 1, 1, 0 },
           { 1, 1, 0, 0, 0, 0, 1, 0 },
           { 1, 1, 0, 0, 0, 0, 0, 0 },
           { 1, 1, 1, 1, 1, 1, 1, 0 },
           { 0, 0, 0, 0, 1, 1, 1, 0 },
           { 1, 0, 0, 0, 1, 1, 1, 0 },
           { 1, 1, 1, 1, 1, 1, 1, 0 },
           { 0, 0, 0, 0, 0, 0, 0, 0 }
```

```
        },
        {   { 1, 1, 0, 0, 0, 1, 1, 0 },
            { 1, 1, 0, 0, 0, 1, 1, 0 },
            { 1, 1, 0, 0, 0, 1, 1, 0 },
            { 1, 1, 1, 1, 1, 1, 1, 0 },
            { 1, 1, 1, 0, 0, 1, 1, 0 },
            { 1, 1, 1, 0, 0, 1, 1, 0 },
            { 1, 1, 1, 0, 0, 1, 1, 0 },
            { 0, 0, 0, 0, 0, 0, 0, 0 }
        }
    };

    /* Tampilkan Huruf */
    for(i=0; i<2; i++)
    {   for(j=0; j<8; j++)
        {   for(k=0;k<8; k++)
            if(data_huruf[i][j][k])
                putchar('\xDB');
            else
                putchar(" ");   /* spasi */
        puts("");
        }
        puts("");
    }
    getch(); }
```

LATIHAN 6

1. Buatlah sebuah program untuk menginput, menghitung dan mencetak perkalian matriks 3 x 3
2. Apa yang tercetak dari program berikut ini :

```
#include <stdio.h>
#define SIZE 10

int whatIsThis(int [], int);

void main() {
   int total, a[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   total = whatIsThis(a, SIZE);
   printf("\nNilai variabel total adalah %d", total);
 }

int whatIsThis(int b[], int size) {
   if (size == 1)
      return b[0];
      else
      return b[size-1] + whatIsThis(b, size-1);
}
```

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab VII
Fungsi

1 PENGERTIAN FUNGSI

Fungsi merupakan suatu bagian dari program yang dimaksudkan untuk mengerjakan suatu tugas tertentu dan letaknya terpisah dari program yang memanggilnya. Fungsi merupakan elemen utama dalam bahasa C karena bahasa C sendiri terbentuk dari kumpulan fungsi-fungsi. Dalam setiap program bahasa C, minimal terdapat satu fungsi yaitu fungsi main(). Fungsi banyak diterapkan dalam program-program C yang terstruktur. Keuntungan penggunaan fungsi dalam program yaitu program akan memiliki struktur yang jelas (mempunyai readability yang tinggi) dan juga akan menghindari penulisan bagian program yang sama.

Dalam bahasa C fungsi dapat dibagi menjadi dua, yaitu fungsi pustaka atau fungsi yang telah tersedia dalam Turbo C dan fungsi yang didefinisikan atau dibuat oleh programmer.

2 BEBERAPA FUNGSI PUSTAKA DALAM BAHASA C

- **Fungsi Operasi String (tersimpan dalam header file "string.h")**
  - **strcpy()**
    - ☑ Berfungsi untuk menyalin suatu string asal ke variable string tujuan.
    - ☑ Bentuk umum : strcpy(var_tujuan, string_asal);
  - **strlen()**
    - ☑ berfungsi untuk memperoleh jumlah karakter dari suatu string.
    - ☑ Bentuk umum : strlen(string);

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "string.h"
void main()
{   char nama[25];
    strcpy(nama, "Achmad Solichin");
    printf("Nama : %s", nama);
    printf("Banyaknya karakter nama Anda adalah : %i", strlen(nama));
    getch();
}
```

- **strcat()**
  - ☑ Digunakan untuk menambahkan string sumber ke bagian akhir dari string tujuan.
  - ☑ Bentuk umum : strcat(tujuan, sumber);

- **strupr()**
  - ☑ Digunakan untuk mengubah setiap huruf dari suatu string menjadi huruf capital.
  - ☑ Bentuk umum : strupr(string);
- **strlwr()**
  - ☑ Digunakan untuk mengubah setiap huruf dari suatu string menjadi huruf kecil semua.
  - ☑ Bentuk umum : strlwr(string);

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "string.h"
void main()
{   char satu[30] = "Fakultas Teknologi Informasi";
    char dua[30] = "Universitas Budi Luhur";
    clrscr();
    strcat(satu, dua);
    printf("Hasil penggabungannya : %s\n", satu);
    printf("Jika diubah menjadi huruf kapital semua :\n");
    printf("%s", strupr(satu));
    printf("Jika diubah menjadi huruf kecil semua :\n");
    printf("%s", strlwr(satu));
    getch();
}
```

- **strcmp()**
  - ☑ Digunakan untuk membandingkan dua buah string.
  - ☑ Hasil dari fungsi ini bertipe integer dengan nilai :
    - (a) Negative, jika string pertama kurang dari string kedua.
    - (b) Nol, jika string pertama sama dengan string kedua
    - (c) Positif, jika string pertama lebih besar dari string kedua.
  - ☑ Bentuk umum : strcmp(string1, string2);

Contoh Program :

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

```
void main()
{   char string1[5], string2[5];
    int hasil;
    clrscr();

    printf("Masukkan string 1 : "); scanf("%s", &string1);
    printf("Masukkan string 2 : "); scanf("%s", &string2);
    hasil = strcmp(string1, string2);
    if(hasil > 0)
        printf("%s > %s", string1,string2);
    else
        if(hasil == 0)
            printf("%s = %s", string1, string2);
        else
            printf("%s < %s", string1, string2);
    getch();
}
```

- **Fungsi Operasi Karakter (tersimpan dalam header "ctype.h")**
  - **islower()**
    - ☑ Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kecil.
    - ☑ Bentuk umum : islower(char);
  - **isupper()**
    - ☑ Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan huruf kapital.
    - ☑ Bentuk umum : isupper(char);
  - **isdigit()**
    - ☑ Fungsi akan menghasilkan nilai benar (bukan nol) jika karakter merupakan sebuah digit.
    - ☑ Bentuk umum : isdigit(char);
  - **tolower()**
    - ☑ Fungsi akan mengubah huruf capital menjadi huruf kecil.
    - ☑ Bentuk umum : tolower(char);
  - **toupper()**
    - ☑ Fungsi akan mengubah huruf kecil menjadi huruf kapital.
    - ☑ Bentuk umum : toupper(char);

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "ctype.h"
void main()
{   char karakter;
    clrscr();
    printf("Masukkan sebuah karakter : ");  karakter = getche();
        if(isupper(karakter))     //periksa apakah "karakter" adalah huruf kapital
    {           puts(" adalah huruf besar");
                printf("Huruf kecilnya adalah : %c", tolower(karakter));
    }
    else
        if(islower(karakter))     //periksa apakah "karakter" adalah huruf kecil
        {   puts(" adalah huruf kecil");
            printf("Huruf besarnya adalah : %c", toupper(karakter));
        }
```

```
        else
            if(isdigit(karakter))  //periksa apakah "karakter" adalah digit
              puts(" adalah karakter digit");
            else
              puts(" bukan huruf besar, huruf kecil atau digit");

    getch();
}
```

- **Fungsi Operasi Matematik (tersimpan dalam header "math.h" dan "stdlib.h")**
  - **sqrt()**
    - ☑ Digunakan untuk menghitung akar dari sebuah bilangan.
    - ☑ Bentuk umum : sqrt(bilangan);
  - **pow()**
    - ☑ Digunakan untuk menghitung pemangkatan suatu bilangan.
    - ☑ Bentuk umum : pow(bilangan, pangkat);

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
void main()
{   int x, y;
    float z;
    clrscr();
    printf("Menghitung x pangkat y\n");
    printf("x = "); scanf("%i", &x);
    printf("y = "); scanf("%i", &y);
    printf(" %i dipangkatkan dengan %i adalah %7.2lf", x, y, pow(x, y));
    getch();

    clrscr();
    printf("Menghitung akar suatu bilangan z\n");
    printf("z = "); scanf("%f", &z);
    printf("Akar dari %f adalah %7.2lf", z, sqrt(z));
    getch();
}
```

- **sin(), cos(), tan()**
  - ☑ Masing-masing digunakan untuk menghitung nilai sinus, cosinus dan tangens dari suatu sudut.
  - ☑ Bentuk umum :

```
sin(sudut);
cos(sudut);
tan(sudut);
```

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
void main()
{   float sudut;
    clrscr();
    printf("Menghitung nilai sinus, cosinus dan tangens\n");
    printf("Masukkan sudut : "); scanf("%f", &sudut);
    printf("Nilai sinus %.2f derajat adalah %.3f", sudut, sin(sudut));
```

```
    printf("Nilai cosinus %.2f derajat adalah %.3f", sudut, cos(sudut));
    printf("Nilai tangens %.2f derajat adalah %.3f", sudut, tan(sudut));
    getch();
}
```

- **atof()**
  - ☑ Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe double.
  - ☑ Bentuk umum : atof(char x);
- **atoi()**
  - ☑ Digunakan untuk mengkonversi nilai string menjadi bilangan bertipe integer.
  - ☑ Bentuk umum : atoi(char x);

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "math.h"
void main()
{   char x[4] = "100", y[5] ="10.3";
    int a;
    float b;
    clrscr();
    a = atoi(x);  b = atof(y);
    printf("Semula A = %s B = %s\n", x,y);
    printf("Setelah dikonversi A = %i B = %.2f", a,b);
    getch();
}
```

- **div()**
  - ☑ Digunakan untuk menghitung hasil pembagian dan sisa pembagian.
  - ☑ Bentuk umum : **div_t div(int x, int y)**
  - ☑ Strukturnya :

```
typedef struct
{  int qout;          // hasil pembagian
   int rem            // sisa pembagian
} div_t;
```

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
void main()
{   int x, y;
    div_t hasil;
    clrscr();
    printf("Menghitung sisa dan hasil pembagian x dengan y\n");
    printf("x = "); scanf("%i", &x);
    printf("y = "); scanf("%i", &y);
    hasil = div(x,y);
    printf("\n\n %3i div %3i = %3i sisa %3i", x, y, hasil.quot, hasil.rem);
    getch();
}
```

- **max()**
  - ☑ Digunakan untuk menentukan nilai maksimal dari dua buah bilangan.
  - ☑ Bentuk umum : max(bilangan1, bilangan2);

- **min()**
  - ☑ Digunakan untuk menentukan bilangan terkecil dari dua buah bilangan.
  - ☑ Bentuk umum : min(bilangan1, bilangan2);

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
void main()
{   int x, y, z;
    clrscr();

    printf("Menentukan bilangan terbesar dan terkecil\n");
    printf("X = "); scanf("%i", &x);
    printf("Y = "); scanf("%i", &y);
    printf("Z = "); scanf("%i", &z);
    printf("\nBilangan terbesar : %i", max(max(x, y), z));
    printf("\nBilangan terkecil : %i", min(min(x, y), z));
    getch();
}
```

3 MEMBUAT FUNGSI SENDIRI

Deklarasi Fungsi

Sebelum digunakan (dipanggil), suatu fungsi harus dideklarasikan dan didefinisikan terlebih dahulu. Bentuk umum pendeklarasian fungsi adalah :

```
tipe_fungsi nama_fungsi(parameter_fungsi);
```

Sedangkan bentuk umum pendefinisian fungsi adalah :

```
Tipe_fungsi nama_fungsi(parameter_fungsi)
{   statement
    statement
    ……...
    ……...
}
```

Hal-hal yang perlu diperhatikan dalam penggunaan fungsi :

- Kalau tipe fungsi tidak disebutkan, maka akan dianggap sebagai fungis dengan nilai keluaran bertipe integer.
- Untuk fungsi yang memiliki keluaran bertipe bukan integer, maka diperlukan pendefinisian penentu tipe fungsi.
- Untuk fungsi yang tidak mempunyai nilai keluaran maka dimasukkan ke dalam tipe void
- Pernyataan yang diberikan untuk memberikan nilai akhir fungsi berupa pernyataan return.
- Suatu fungsi dapat menghasilkan nilai balik bagi fungsi pemanggilnya.

Contoh Program 1 :

```
#include "stdio.h"
#include "conio.h"
float tambah(float x, float y);              /* prototype fungsi tambah(), ada titik koma */
void main()
{   float a, b, c;
    clrscr();
    printf("A = ");  scanf("%f", &a);
```

```
    printf("B = ");  scanf("%f", &b);
    c = tambah(a, b);            /* pemanggilan fungsi tambah() */
    printf("A + B = %.2f", c);
    getch();
}

float tambah(float x, float y)    /* Definisi fungsi , tanpa titik koma */
{   return (a+b);                 /* Nilai balik fungsi */
}
```

Contoh Program 2 :

```
/* Program menghitung nilai factorial */
#include "stdio.h"
#include "conio.h"
long int faktorial(int N);              /* prototype fungsi factorial() */

void main()
{   int N;
    long int fak;

    printf("Berapa factorial ? "); scanf("%i", &N);
    fak = faktorial(N);          /* pemanggilan fungsi factorial() */
    printf("%i factorial = %ld\n", N, fak);
    getch();
}

long int faktorial(int N)        /* definisi fungsi factorial */
{   int I;
    long int F = 1;
    if(N<=0)
    return(0);
    for(I=2; I<=N; I++)
    F = F * I;
    return(F);
}
```

- **Parameter Formal dan Parameter Aktual**
  - **Parameter Formal** adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
  - **Parameter Aktual** adalah variabel (parameter) yang dipakai dalam pemanggilan fungsi.

  Dalam contoh program pertambahan di atas parameter formal terdapat pada pendefinisisan fungsi :

```
float tambah(float x, float y)  //parameter formal
{  return (a+b);
}
```

Sedangkan parameter aktual terdapat pada pemanggilan fungsi :

```
void main()
{  ……………
   ……..
   c = tambah(a, b);            //parameter aktual
   ……..
}
```

- **Cara Melewatkan Parameter**

  Cara melewatkan suatu parameter dalam Bahasa C ada dua cara yaitu :

  - Pemanggilan Secara Nilai (*Call by Value*)
    - ☑ Call by value akan menyalin nilai dari parameter aktual ke parameter formal.
    - ☑ Yang dikirimkan ke fungsi adalah nilai dari datanya, bukan alamat memori letak dari datanya.
    - ☑ Fungsi yang menerima kiriman nilai akan menyimpannya di alamat terpisah dari nilai aslinya yang digunakan oleh bagian program yang memanggil fungsi.
    - ☑ Perubahan nilai di fungsi (parameter formal) tidak akan merubah nilai asli di bagian program yang memanggilnya.
    - ☑ Pengiriman parameter secara nilai adalah pengiriman searah, yaitu dari bagian program yang memanggil fungsi ke fungsi yang dipanggil.
    - ☑ Pengiriman suatu nilai dapat dilakukan untuk suatu ungkapan, tidak hanya untuk sebuah variabel, elemen array atau konstanta saja.

    Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void tukar(int x, int y);              /* pendeklarasian fungsi */

void main()
{   int a,b;
    clrscr();
    a = 15;
    b = 10;
    printf("Nilai sebelum pemanggilan fungsi\n");
    printf("a = %i b = %i\n\n", a, b);     // a dan b sebelum pemanggilan fungsi

    tukar(a,b);            /* pemanggilan fungsi tukar() */

    printf("Nilai setelah pemanggilan fungsi\n");
    printf("a = %i b = %i\n\n", a, b);     // a dan b setelah pemanggilan fungsi
    getch();
}
void tukar(int x, int y)  /* Pendefinisian fungsi tukar() */
{   int z;                             /* variabel sementara */
    z = x;
    x = y;
    y = z;
    printf("Nilai di akhir fungsi tukar()\n");
    printf("x = %i y = %i\n\n", x, y);
}
```

  - Pemanggilan Secara Referensi (*Call by Reference*)
    - ☑ Pemanggilan secara Referensi merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi.
    - ☑ Yang dikirimkan ke fungsi adalah alamat letak dari nilai datanya, bukan nilai datanya.
    - ☑ Fungsi yang menerima kiriman alamat ini makan menggunakan alamat yang sama untuk mendapatkan nilai datanya.
    - ☑ Perubahan nilai di fungsi akan merubah nilai asli di bagian program yang memanggil fungsi.
    - ☑ Pengiriman parameter secara referensi adalah pengiriman dua arah, yaitu dari fungsi pemanggil ke fungsi yang dipanggil dan juga sebaliknya.
    - ☑ Pengiriman secara acuan tidak dapat bdilakukan untuk suatu ungkapan.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void tukar(int *px, int *py);

void main()
{   int a,b;
    clrscr();
    a = 15;
    b = 10;
    printf("Nilai sebelum pemanggilan fungsi\n");
    printf("a = %i b = %i\n\n", a, b);

    tukar(&a,&b);                    /* parameter alamat a dan alamat b */

    printf("Nilai setelah pemanggilan fungsi\n");
    printf("a = %i b = %i\n\n", a, b);
    getch();
}

void tukar(int *px, int *py)
{   int z;              /* variabel sementara */
    z = *px;
    *px = *py;
    *py = z;
    printf("Nilai di akhir fungsi tukar()\n");
    printf("*px = %i *py = %i\n\n", *px, *py);
}
```

- **Penggolongan Variabel berdasarkan Kelas Penyimpanan (Storage Class)**
  - Variabel lokal
    Variabel lokal adalah variabel yang dideklarasikan di dalam fungsi.
    Sifat-sifat variabel lokal :
    - ☑ Secara otomatis akan diciptakan ketika fungsi dipanggil dan akan lenyap ketika proses eksekusi terhadap fungsi berakhir.
    - ☑ Hanya dikenal oleh fungsi tempat variabel dideklarasikan
    - ☑ Tidak ada inisialisasi secara otomatis (saat variabel diciptakan nilainya random).
    - ☑ Dideklarasikan dengan menambahkan kata "**auto**" (opsional).

  - Variabel global (eksternal)
    Variabel global (eksternal) adalah variabel yang dideklarasikan di luar fungsi.
    Sifat-sifat variabel global :
    - ☑ Dikenal (dapat diakses) oleh semua fungsi.
    - ☑ Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
    - ☑ Dideklarasikan dengan menambahkan kata "**extern**" (opsional).

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void tampil(void);
int i = 25;                      /* variabel global */
void main()
{   clrscr();
    printf("Nilai variabel i dalam fungsi main() adalah %i\n\n", i);
```

```
    tampil();
    i = i * 4;                    /* nilai i yang dikali 4 adalah 25 (global) bukan 10  */
    printf("\nNilai variabel i dalam fungsi main() sekarang adalah %i\n\n", i);

    getch();
}

void tampil(void)
{   int i = 10;             /* variabel lokal */
   printf("Nilai variabel i dalam fungsi tampil() adalah %i\n\n", i);
}
```

- Variabel Statis
  Variabel statis adalah variabel yang nilainya tetap dan bisa berupa variabel lokal (internal) dan variabel global (eksternal).
  Sifat-sifat variabel statis :
  - ☑ Jika bersifat internal (lokal), maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan.
  - ☑ Jika bersifat eksternal (global), maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada program yang sama.
  - ☑ Nilai variabel statis tidak akan hilang walau eksekusi terhadap fungsi telah berakhir.
  - ☑ Inisialisasi hanya perlu dilakukan sekali saja, yaitu pada saat fungsi dipanggil pertama kali.
  - ☑ Jika tidak diberi nilai awal secara otomatis berisi nilai nol.
  - ☑ Dideklarasikan dengan menambahkan kata "**static**".

- Variabel Register
  Variabel Register adalah variabel yang nilainya disimpan dalam resister dan bukan dalam memori RAM.
  Sifat-sifat variabel register :
  - ☑ Hanya dapat diterapkan pada variabel lokal yang bertipe int dan char.
  - ☑ Digunakan untuk mengendalikan proses perulangan (looping).
  - ☑ Proses perulangan akan lebih cepat karena variabel register memiliki kecepatan yang lebih tinggi dibandingkan variabel biasa.
  - ☑ Dideklarasikan dengan menambahkan kata "**register**".

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   register int x;                /* variabel register */
    int jumlah;
    clrscr();
    for(i=1; i<=100; i++)
    jumlah = jumlah + I;
    printf("1+2+3+…+100 = %i\n", jumlah);
    getch();
}
```

- **Fungsi Rekursif**
  Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri.

Contoh Program :

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

```
#include "conio.h"
long int faktorial(int N);                    /* prototype fungsi faktorial */
void main()
{    int N;

     printf("Berapa factorial ? "); scanf("%i", &N);

     printf("Faktorial dari %i = %ld\n", N, faktorial(N));
     getch();
}

long int faktorial(int N)          /* definisi fungsi factorial */
{
     if(N==0)
        return(1);
     else
        return(N * faktorial(N - 1)); // fungsi faktorial() memanggil fungsi factorial()
}
```

LATIHAN 7

1. Buat fungsi untuk menentukan apakah suatu bilangan bulat bersifat ganjil atau genap. Jika genap maka fungsi menghasilkan nilai 1, dan 0 untuk selainnya.
2. Buatlah fungsi menjumlahkan bilangan 1,2,3, ….., n secara rekursif.
3. Buatlah Program untuk menghitung jarak maksimum (xmax) dan ketinggian maksimum (hmax) dari sebuah peluru yang ditembakkan dengan sudut elevasi A. Anggap g = 10 m/s2 (Gunakan fungsi sin() dan cos())

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab VIII
Pointer

1 PENGERTIAN POINTER

- **Pointer** (variabel penunjuk) adalah suatu variabel yang berisi alamat memori dari suatu variabel lain. Alamat ini merupakan lokasi dari obyek lain (biasanya variabel lain) di dalam memori. Contoh, jika sebuah variabel berisi alamat dari variabel lain, variabel pertama dikatakan menunjuk ke variabel kedua
- Operator Pointer ada dua, yaitu :
  - Operator &
    - ☑ Operator & bersifat unary (hanya memerlukan satu operand saja).
    - ☑ Operator & menghasilkan alamat dari operandnya.
  - Operator *
    - ☑ Operator * bersifat unary (hanya memerlukan satu operand saja).
    - ☑ Operator * menghasilkan nilai yang berada pada sebuah alamat.

2 DEKLARASI POINTER

Seperti halnya variabel yang lain, variabel pointer juga harus dideklarasikan terlebih dahulu sebelum digunakan. Bentuk Umum :

Tipe_data *nama_pointer;

Tipe data pointer mendefinisikan tipe dari obyek yang ditunjuk oleh pointer. Secara teknis, tipe apapun dari pointer dapat menunjukkan lokasi (dimanapun) dalam memori. Bahkan operasi pointer dapat dilaksanakan relatif terhadap tipe dasar apapun yang ditunjuk. Contoh, ketika kita mendeklarasikan pointer dengan tipe int*, kompiler akan menganggap alamat yang ditunjuk menyimpan nilai integer - walaupun sebenarnya bukan (sebuah pointer int* selalu menganggap bahwa ia menunjuk ke sebuah obyek bertipe integer, tidak peduli isi sebenarnya). Karenanya, sebelum mendeklarasikan sebuah pointer, pastikan tipenya sesuai dengan tipe obyek yang akan ditunjuk.

Contoh :

```
int *px;
char *sh;
```

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   int x, y;              /* x dan y bertipe int */
    int *px;               /* px pointer yang menunjuk objek */
    clrscr();

    x = 87;
    px = &x;               /* px berisi alamat dari x */
    y = *px;               /* y berisi nilai yang ditunjuk px */

    printf("Alamat x = %p\n", &x);
    printf("Isi px     = %p\n", px);
    printf("Isi x      = %i\n", x);
    printf("Nilai yang ditunjuk oleh px = %i\n", *px);
    printf("Nilai  y   = %i\n", y);
    getch();
}
```

3 OPERASI POINTER

- **Operasi Penugasan**
  - Suatu variable pointer seperti halnya variable yang lain, juga bisa mengalami operasi penugasan. Nilai dari suatu variable pointer dapat disalin ke variable pointer yang lain.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   float *x1, *x2, y;
    clrscr();
    y = 13.45;
    x1 = &y;               /* Alamat dari y disalin ke variabel x1 */
    x2 = x1;               /* Isi variabel x1 disalin ke variabel x2 */
    printf("Nilai variabel y = %.2f ada di alamat %p\n", y, x1);
    printf("Nilai variabel y = %.2f ada di alamat %p\n", y, x2);
    getch();
}
```

- **Operasi Aritmatika**
  - Suatu variabel pointer hanya dapat dilakukan operasi aritmatika dengan nilai integer saja. Operasi yang biasa dilakukan adalah operasi penambahan dan pengurangan. Operasi penambahan dengan suatu nilai menunjukkan lokasi data berikutnya (index selanjutnya) dalam memori. Begitu juga operasi pengurangan.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   int nilai[3], *penunjuk;
    clrscr();
    nilai[0] = 125;
    nilai[1] = 345;
    nilai[2] = 750;
    penunjuk = &nilai[0];
    printf("Nilai %i ada di alamat memori %p\n", *penunjuk, penunjuk);
    printf("Nilai %i ada di alamat memori %p\n", *(penunjuk+1), penunjuk+1);
    printf("Nilai %i ada di alamat memori %p\n", *(penunjuk+2), penunjuk+2);
    getch();
}
```

- **Operasi Logika**
  - Suatu pointer juga dapat dikenai operasi logika.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   int a = 100, b = 200, *pa, *pb;
    clrscr();
    pa = &a;
    pb = &b;
    if(pa < pb)
    printf("pa menunjuk ke memori lebih rendah dari pb\n");
    if(pa == pb)
    printf("pa menunjuk ke memori yang sama dengan pb\n");
    if(pa > pb)
    printf("pa menunjuk ke memori lebih tinggi dari pb\n");
    getch();
}
```

4 POINTER DAN STRING

Contoh Program 1 :

```
#include "stdio.h"
#include "conio.h"
char *nama1 = "SPIDERMAN";
char *nama2 = "GATOTKACA";

void main()
{   char namax;
    clrscr();
```

```
    puts("SEMULA :");
    printf("Saya suka >> %s\n", nama1);
    printf("Tapi saya juga suka >> %s\n", nama2);

    /* Penukaran string yang ditunjuk oleh pointer nama1 dan nama2 */
    printf("SEKARANG :");
    printf("Saya suka >> %s\n", nama1);
    printf("Dan saya juga masih suka >> %s\n", nama2);
    getch();
}
```

Contoh Program 2 :

```
#include <stdio.h>
void misteri1(char *);

void main() {
   char string[] = "characters";
   printf("String sebelum proses adalah %s", string);
   misteri1(string);
   printf("String setelah proses adalah %s", string);
}

void misteri1(char *s) {
   while ( *s != '\0' ) {
      if ( *s >= 'a' && *s <= 'z' )
         *s -= 32;
      ++s;
   }
}
```

5 POINTER MENUNJUK SUATU ARRAY

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   static int tgl_lahir[] = { 13,9,1982 };
    int *ptgl;

    ptgl = tgl_lahir;              /* ptgl berisi alamat array */
    printf("Diakses dengan pointer");
    printf("Tanggal  = %i\n", *ptgl);
    printf("Bulan    = %i\n", *(ptgl + 1));
    printf("Tahun    = %i\n", *(ptgl + 2));

    printf("\nDiakses dengan array biasa\n");
    printf("Tanggal = %i\n", tgl_lahir[0]);
    printf("Bulan   = %i\n", tgl_lahir[1]);
    printf("Tahun   = %i\n", tgl_lahir[2]);
    getch();
}
```

6 MEMBERI NILAI ARRAY DENGAN POINTER

Contoh Program :

```
#include “stdio.h”
#include “conio.h”
void main()
{   int x[5], *p, k;
     clrscr();
     p = x;
     x[0] = 5;              /* x[0] diisi dengan 5 sehingga x[0] = 5 */
     x[1] = x[0];                  /* x[1] diisi dengan x[0] sehingga x[1] = 5 */
     x[2] = *p + 2;                /* x[2] diisi dengan x[0] + 2 sehingga x[2] = 7 */
     x[3] = *(p+1) – 3;  /* x[3] diisi dengan x[1]  - 3 sehingga x[3] = 2 */
     x[4] = *(x + 2);      /* x[4] diisi dengan x[2] sehingga x[4] = 7 */
     for(k=0; k<5; k++)
     printf(“x[%i] = %i\n”, k, x[k]);

     getch();
}
```

LATIHAN 8

Apa yang tercetak dari program-program berikut ini ?

1.
```
#include <stdio.h>
void misteri2(const char *);

main() {
   char *string = "ilmu komputer";
   misteri2(string);
   putchar('\n');
   return 0;
}
void misteri2(const char *s) {
   for ( ; *s != '\0' ; s++ )
      putchar(*s);
}
```

2.
```
#include <stdio.h>
int misteri3(const char *);
void main() {
   char string[80];
   printf("Ketik sebuah string : ");   scanf("%s", string);
   printf("%d\n", misteri3(string));
}

int misteri3(const char *s) {
   int x = 0;

   for ( ; *s != '\0' ; s++)
```

```
        ++x;
    return x;
}
```

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Bab IX
Operasi File

File adalah sebuah organisasi dari sejumlah record. Masing-masing record bisa terdiri dari satu atau beberapa field. Setiap field terdiri dari satu atau beberapa byte.

1 MEMBUKA FILE

- Untuk membuka atau mengaktifkan file, fungsi yang digunakan adalah fungsi **fopen().**
- File dapat berupa file biner atau file teks.
- File biner adalah file yang pola penyimpanan di dalam disk dalam bentuk biner, yaitu seperti bentuk pada memori (RAM) computer.
- File teks adalah file yang pola penyimpanan datanya dalam bentuk karakter.
- Penambahan yang perlu dilakukan untuk menentukan mode teks atau biner adalah "t" untuk file teks dan "b" untuk file biner.
- Prototype fungsi fopen() ada di header fungsi "**stdio.h**"
- **Bentuk umum :**

```
file *fopen(char *namafile, char *mode);
```

Keterangan :

- **namafile** adalah nama dari file yang akan dibuka/diaktifkan.
- **mode** adalah jenis operasi file yang akan dilakukan terhadap file.

- Jenis-jenis operasi file :
  - r : menyarakan file hanya dapat dibaca (file harus sudah ada)
  - w : menyatakan file baru akan dibuat/diciptakan (file yang sudah ada akan dihapus)
  - a : untuk membuka file yang sudah ada dan akan dilakukan proses penambahan data (jika file belum ada, otomatis akan dibuat)
  - r+ : untuk membuka file yang sudah ada dan akan dilakukan proses pembacaan dan penulisan.
  - w+ : untuk membuka file dengan tujuan untuk pembacaan atau penulisan. Jika file sudah ada, isinya akan dihapus.
  - a+ : untuk membuka file, dengan operasi yang akan dilakukan berupa perekaman maupun pembacaan. Jika file sudah ada, isinya akan dihapus.
- Contoh :

```
pf = fopen("COBA.TXT", "w");
```

2 MENUTUP FILE

- Untuk menutup file, fungsi yang digunakan adalah **fclose().**
- Prototype fungsi fclose() ada di header file **"stdio.h"**
- Bentuk Umum :

```
int fclose(FILE *pf);
```

atau

```
int fcloseall(void);
```

3 MELAKSANAKAN PROSES FILE

- **Menulis Karakter**
  - Untuk menulis sebuah karakter, bentuk yang digunakan adalah :

```
putc(int ch, file *fp)
```

    - ☑ fp adalah pointer file yang dihasilkan oleh fopen()
    - ☑ ch adalah karakter yang akan ditulis.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
#define CTRL_Z 26
void main()
{   file *pf;                                          /* pointer ke file */
    char kar;
    if((pf = fopen("COBA.TXT", "w")) == NULL)      /* ciptakan file */
    {   cputs("File tak dapat diciptakan !\r\n");
        exit(1);                                       /* selesai */
    }
    while((kar=getche()) != CTRL_Z)
    putc(kar, pf);                                     /* tulis ke file */
    fclose(pf);                                        /* tutup file */
}
```

- **Membaca Karakter**
  - Untuk membaca karakter dari file, fungsi yang digunakan adalah :

```
getc(file *fp);
```

    - ☑ fp adalah pointer file yang dihasilkan oleh fopen()
    - ☑ Fungsi feof(), digunakan untuk mendeteksi akhir file.

- ☑ Pada saat membaca data foef(file *fp)

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   file *pf;                                   /* pointer ke file */
    char kar;
    clrscr();

    if((pf = fopen("COBA.TXT", "r")) == NULL)   /* buka file */
    {  cputs("File tak dapat dibuka !\r\n");
       exit(1);                                 /* selesai */
    }

    while((kar=getc(pf)) != EOF)
        putch(kar);                             /* tampilkan ke layar */
    fclose(pf);                                 /* tutup file */
}
```

- **Membaca dan Menulis String**
  - Fungsi untuk membaca dan menulis string adalah : fgets() dan fputs()
  - Bentuk Umum :
    fgets(char *str, int p, file *fp)
    fputs(char *str, file *fp)

- **Membaca dan Menulis Blok Data**
  - Fungsi untuk membaca dan menulis blok data adalah : fread() dan fwrite()
  - Bentuk umum :
    fread(void *buffer, int b_byte, int c, file *fp);
    fwrite(void *buffer, int b_byte, int c, file *fp);

Keterangan :

- ☑ **buffer** adalah pointer ke sebuah area di memori yang menampung data yang akan dibaca dari file.
- ☑ **b_byte** adalah banyaknya byte yang akan dibaca atau ditulis ke file
- ☑ **c** adalah banyaknya item dibaca/ditulis.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"
void main()
{   file *f_struktur;
    char jawaban;
    struct data_pustaka
    {   char judul[26];
        char pengarang[20];
        int jumlah;
    } buku;             /* variabel buku bertipe struktur */

    /* buka file */
    if((f_struktur = fopen("DAFBUKU.DAT", "wb")) == NULL)/* buka file */
    {  cputs("File tak dapat diciptakan !\r\n");
       exit(1);                                         /* selesai */
    }
```

```
    do
    {   clrscr();
        cputs("Judul Buku              : ");
        gets(buku.judul);
        cputs("Nama Pengarang          : ");
        gets(buku.pengarang);
        cputs("Jumlah buku             : ");
        scanf("%i", $buku.jumlah);
        fflush(stdin);                 /* Hapus isi penampung keyboard */

        /*Rekam sebuah data bertipe struktur */
        fwrite(&buku, sizeof(buku), 1, f_struktur);

        cputs("\r\nMau merekam data lagi (Y/T) ?");
        jawaban = getche();
    }
    while(jawaban == 'Y' || jawaban == 'y');

    fclose(f_struktur);                        /* tutup file */);
}
```

- **Membaca dan Menulis File yang Terformat**
  - Jika diinginkan, data bilangan dapat disimpan ke dalam file dalam keadaan terformat.
  - Fungsi yang digunakan adalah :
    fprintf(ptr_file, "string control", daftar argument);
    fscanf(pts_file, "string control", daftar argument);

Contoh Program :

```
#include "stdio.h:
#include "conio.h"
void main()
{   FILE *pformat;
    char jawaban;
    struct
    {   int x;
        int y;
    } koordinat;

    /* Buka dan ciptakan file. Periksa kalau gagal dibuka */
    if((pformat = fopen("KOORDINAT.TXT", "w")) == NULL) /* buka file */
    {   cputs("File tak dapat dibuka !\r\n");
       exit(1);                                        /* selesai */
    }
    do
    {   clrscr();
        cputs("Masukkan data koordinat (bilangan integer)\r\n");
        cputs("Format : posisi x posisi y\r\n");
        cputs("Contoh :   20        30   [ENTER]\r\n");
        scanf("%i %i, &koordinat.x, &koordinat.y);
        fflush(stdin);

        /* Rekam ke file */
        fprintf(pformat, %5i %5i\n", koordinat.x, koordinat.y);
        cputs("\r\nMenyimpan data lagi (Y/T) ??");
```

```
            jawaban = getche();
        }
        while(jawaban == 'y' || jawaban == 'Y');
        fclose(pformat);
        getch();
}
```

Contoh Program 2 :

```
#include <stdio.h>

FILE *in;
void BACA( int[ ] );
void CETAK( int[ ] );

void main() {
   int tabel[26] = {0};
   BACA(tabel);
   CETAK(tabel);
}
void BACA ( int huruf[] ) {
   char c;
   if (( in = fopen("data.txt" , "r")) == NULL)
      printf ("File tidak bisa dibaca\n");
      else
      while ( (ch = fgetc(in)) != EOF ) {
         c = ( (( c >= 97) || ( c <= 122)) ? c - 32 : c );
         if ( (c >= 65) || (c <= 90) )
            ++huruf [ c - 65 ];
      }
fclose(in);
}

void CETAK ( int huruf[] ) {
   int counter;
   for ( counter = 0 ; counter <= 25 ; counter++ )
      printf ("\n%c%5d", counter + 65, huruf[counter] );
}
```

4 FILE SEQUENSIAL

File sekuensial berisi rekord-rekord data yang tidak mengenal posisi baris atau nomor rekord pada saat aksesnya, dan setiap record dapat mempunyai lebar yang berbeda-beda. Akses terhadapnya selalu dimulai dari awal file dan berjalan satu persatu menuju akhir dari file. Dengan demikian, penambahan file hanya dapat dilakukan terhadap akhir file, dan akses terhadap baris tertentu harus dimulai dari awal file.

Fungsi baku yang terkait dengan file sekuensial ini antara lain adalah **fprintf**,**fscanf**, dan **rewind** . Program berikut menyajikan penanganan file sekuensial tentang data nasabah yang berisi tiga field, yaitu nomor identitas (*account*), nama (*name*), dan posisi tabungannya (*balance*) untuk (1) menyajikan yang tabungannya bernilai nol, (2) berstatus kredit, dan (3) berstatus debet. File data tersimpan dengan nama klien.dat.

```
#include <stdio.h>

void main() {
   int request, account;
   float balance;
   char name[25];
   FILE *cfPtr;

   if ( (cfPtr = fopen("klien.dat", "r+") ) == NULL )
      printf("File could not be opened\n");
      else {
         printf ( "Enter request\n"
                  "1 - List accounts with zero balances\n"
                  "2 - List accounts with credit balances\n"
                  "3 - List accounts with debit balances\n"
                  "4 - End of run\n? " ) ;
         scanf( "%d", &request );

        while (request != 4) {
            fscanf (cfPtr, "%d%s%f", &account, name, &balance);

            switch (request) {
               case 1:
                  printf ("\nAccounts with zero balances:\n");
                   while ( !feof(cfPtr) ) {
                     if (balance == 0)
                        printf ("%-10d%-13s7.2f\n", account, name, balance);
                     fscanf (cfPtr, "%d%s%f", &account, name, &balance);
                   }
                  break;

               case 2:
                  printf ("\nAccounts with credit balances:\n");
                  while ( !feof(cfPtr) ) {
                     if (balance < 0)
                        printf ("%-10d%-13s7.2f\n", account, name, balance);
                        fscanf (cfPtr, "%d%s%f", &account, name, &balance);
                  }
                  break;

               case 3:
                  printf ("\nAccounts with debit balances:\n");
                  while ( !feof(cfPtr) ) {
                     if (balance > 0)
                        printf ("%-10d%-13s7.2f\n", account, name, balance);
                        fscanf (cfPtr, "%d%s%f", &account, name, &balance);
                  }
                  break;
            }

            rewind(cfPtr);
            printf( "\n? ");
            scanf ("%d", &request);
         }
```

```
        printf ("End of run.\n");
        fclose(cfPtr);
    }
}
```

VARIABEL, TIPE DATA DAN EKSPRESI
Bab 2

2.1 IDENTIFIER

Indentifier adalah nama yang diberikan untuk nama objek, nama fungsi, nama variable, dll (sifatnya 'case sensitive'). Identifier pada C++ terdiri dari :

1. huruf 'A' sampai 'Z'
2. huruf 'a' sampai 'z'
3. underscore (_)
4. bilangan antara '0' sampai '9'

Ketentuan dalam memberi nama identifier dalah C++ adalah :

1. karakter pertama harus huruf atau underscore
2. untuk compiler Borland, panjang maksimum 32 karakter
3. identifier harus tidak sama dengan keyword yang ada di C++

contoh identifier :

- Yang benar : nilai, Nilai_nama, No8
- Yang salah : 1Buah, nomor-data, if

2.2 TIPE DATA DI C++

Tipe data diklasifikasikan berdasarkan bagaimana keadaan data disimpan dalam memori, dan jenis operasi yang dapat dilakukan.

2.2.1 CHAR

Adalah sembarang huruf, angka, tanda baca tunggal. Ada 2 (dua) macam char, yaitu :

1. signed
   mendeklarasikan char bertanda, digunakan untuk nilai negative. Rentang nilai mulai -128 sampai 127
2. unsigned
   mendeklarasikan char tidak bertanda, untuk nilai positif. Rentang nilai mulai 0 sampai 255

contoh deklarasi char :

char letter = ‘A’ ;
unsigned char number = 245 ;
signed char value = -71 ;

2.2.2 SHORT, INT, LONG

Digunakan untuk menyatakan bilangan bulat. Seperti pada char, perubah tipe signed dan unsigned dapat ditambahkan.
Rentang nilai short int mulai -32.768 sampai 32.767
Rentang nilai long / int mulai -2.147.483.648 sampai 2.147.483.647

Contoh deklarasi int :

Int nilai, total ; *atau*
Int nilai = 90 ;

2.2.3 FLOAT, DOUBLE

Menyatakan bilangan pecahan/real, maupun eksponensial. Dalam keadaan default, bilang floting point dianggap bertipe double.

Rentang nilai float mulai 3,4 E $^{-38}$ sampai 3,4 E $^{+38}$
Rentang nilai double mulai 1,7 E $^{-308}$ sampai 1,7 E $^{+308}$

2.2.4 ENUMERATION / ENUM

Adalah serangkaian symbol berurutan yang menspesifikasikan konstanta bertipe integer. Dalam C++ tidak terdapat tipe Bolean, sehingga untuk merepresentasikan TRUE dengan nilai integer bukan nol (1, 2, dst), sedangkan FALSE dengan nilai nol (0)

Contoh deklarasi enum :

Enum BOOLEAN { False, True } ; *atau*
Enum BOOLEAN { Benar = 3, Salah = 0 } ;

2.2.5 VOID

Menyatakan tipe kosong untuk :

1. mendeklarasikan fungsi yang tidak mengembalikan nilai apapun.
2. mendeklarasikan fungsi yang tidak menerima parameter apapun.
3. bila diawali dengan operator *, menyatakan penunjuk terhadap sembarang tipe data.

Contoh deklarasi void :

Void cctrputs (char*, int) ; *atau*
Main (**void**) ; *atau*
Void* action ;
Int ivalue = 100 ;
Action = &ivalue ;

2.2.6 PENUNJUK / POINTER

Adalah variable yang berisi nilai alamat suatu lokasi memori tertentu. Deklarasi penunjuk dilakukan dengan menspesifikasikan *, sebelum nama varabel / konstanta.

2.2.7 PENUNJUK / POINTER

Adalah sekelompok data bertipe sama yang menduduki lokasi memori yang berurutan. Jumlah elemen array dinyatakan dengan cara mengapit jumlah yang di maksud dengan tanda ‘ […] ‘

Bentuk umum : tipe namaArray [jumlahelemen] ;

Untuk menyatakan array berdimensi lebih dari 1 (satu), tambahkan tanda ‘[…]’ sebanyak dimensi yang diinginkan.

Contoh deklarasi array 2 dimensi :

Int matrix [2][3] ;

2.2.8 STRING

Deretan karakter yang diakhiri dengan sebuah karakter kosong. String ditulis dengan mengapit string dengan tanda petik dua (“ …….”)

Contoh deklarasi string :

Char text [ ] = “ C++ “ ;
Puts (text) ;

2.2.9 STRUCT, UNION

Digunakan untuk mendeklarasikan sekelompok data yang memiliki tipe yang berlainan. **Struct** : elemennya ada dilokasi memori yang berbeda, dan **union** : elemennya ada dilokasi memori yang sama.

Bentuk umum :

```
Struct tipestruktur
{
        Tipeanggota1 namaAnggota1 ;
        Tipeanggota2 namaAnggota2 ;
        ……..
}
namaStruktur ;
```

2.3 DATA OBJEK

Data objek adalah bagian dari memori yang digunakan untuk menampung nilai dari variable. **Variable** umumnya digunakan untuk data objek yang nilainya dapat diubah selama pemrosesan berlangsung.

Contoh deklarasi variable :

Int **nilai** ; atau int **nilai** = 80 ;

Dalam C++ pendeklarasian termasuk statemen, sehingga pendeklarasian dapat diletakkan pada sembarang tempat dalam program.

Konstanta data objek adalah : variable yang nilainya tidak dapat diubah selama pemrosesan berlangsung.

Contoh deklarasi konstanta :

Const double pi = 3.14 ;

2.4 SCOPE IDENTIFIER

Ruang lingkup / scope adalah bagian mana dari program, identifier tersebut dapat diakses. Scope dari suatu identifier dimulai dari pendeklarasian sampai dengan akhir dari suatu blok. Scope identifier ada 2, yaitu :

1. local identifier
   dideklarasikan di dalam blok ‘ { … } ‘
2. global identifier
   dideklarasikan di luar dari blok

Scope relolution operator (::) dapat digunakan untuk mengakses variable global secara langsung.

Contoh variable global :

```
Int x ;
F ( )
{
        Int x ;
        :: x = 4 ;      // pemberian nilai untuk variable global x
}
```

2.5 OPERATOR DAN EKSPRESI

Ekspresi adalah rangkaian dari operator, operand, dan punctuator (;)
contoh :

```
3 + 4 * 1 ;
```

2.5.1 OPERATOR ARITMATIKA

Terdiri dari :

- penjumlahan (+)
- pengurangan (-)
- sisa bagi / hanya untuk tipe data integer (%)
- perkalian (*)
- pembagian (/)

Jika operator bagi (/) diterapkan pada tipe integer, akan menghasilkan bilangan integer dengan decimal yang dihilangkan.

2.5.2 ASSIGNMENT OPERATOR (=)

Berfungsi untuk memberi nilai pada variable.

Table kombinasi assignment :

NO.	PENYINGKATAN	ARTI
1	X += Y	X = X + Y
2	X -= Y	X = X – Y
3	X *= Y	X = X * Y
4	X /= Y	X = X / Y
5	X %= Y	X = X % Y

2.5.3 INCREMENT DAN DECREMENT OPERATOR

Increment adalah penambahan suatu variable dengan nilai 1, dan **decrement** adalah pengurangan suatu variabel dengan nilai 1.

Tabelnya :

INCREMENT	DECREMENT
++X ;	- -X ;
X += 1 ;	X -= 1 ;
X = X + 1	X = X – 1 ;

Operator increment dan decrement dapat diletakkan pada awal atau akhir variable, seperti dibawah ini :

++X , nilai variable X dinaikkan dahulu sebelum diproses
X++ , nilai variable X diproses dahulu sebelum dinaikkan

Contoh program increment :

```
# include <iostream.h>
Main ( )
{
Int X = 5 ;
Cout << " Nilai X = " << X << ' \n ' ;
Cout << " Nilai X++ = " << X++ << ' \n ' ;
Cout << " X = " << X << ' \n ' :
X = 5 ;
Cout << " Nilai X = " << X << ' \n ' ;
Cout << " Nilai ++X = " << ++X << ' \n ' ;
Cout << " X = " << X << ' \n ' ;
Return 0 ;
}
```

Outputnya :

```
Nilai X = 5
Nilai X++ = 5
X = 6
Nilai X = 5
Nilai ++X = 6
X = 6
```

2.5.4 EQUALITY, RELATIONAL, LOGIKA OPERATOR

Equality digunakan untuk menentukan apakah 2 buah variable memiliki nilai yang sama atau tidak.

= = , sama dengan, contoh : 5 **= =** 5
! = , tidak sama dengan , contoh : 5 **! =** 4

Relational operator digunakan untuk menentukan apakah suatu variable memiliki nilai lebih besar atau sama dengan lebih besar, lebih kecil atau lebih kecil sama dengan.

Operatornya : **<** , **< =** , **>** , **> =**

Logika operator adalah : (penulisan dibawah ini berdasarkan prioritas operator yang akan diproses terlebih dahulu)

1. **!** (not)
2. **&&** (and)
3. **||** (or)

2.6 EKSPRESI CONDITIONAL

Bentuk umumnya :

Ekspresi C **?** ekspresi T **:** ekspresi S ;

Keterangan :
- ekspresi C = kondisi yang akan diproses lebih dahulu
- ekspresi T = jika kondisi ekspresi C nilainya TRUE, akan dijalankan
- ekspresi S = jika kondisi ekspresi C nilainya FALSE, akan dijalankan

contoh program :

```
# include <iostream.h>
Main ( )
{
Double nilai ;
Cout << " Masukkan suatu nilai = ' ;
Cin >> nilai ;
Nilai = (nilai < 0) ? –nilai : nilai ;
Cout << "nilai absolutnya =" << nilai ;
Return 0 ;
}
```

2.7 FORMAT OUTPUT PADA BILANGAN REAL

Beberapa format yang dapat dilakukan :

1. derajat ketelitian, dengan fungsi : **precision**.
2. lebar output dapat diubah dengan fungsi : **width**.
3. format bilangan real diubah dengan fungsi : **setf** diikuti argument : **ios :: scientific** atau **ios :: fixed**
4. aligment (rata kiri/kanan) dengan fungsi : **setf** atau **unsetf** diikuti argument : **ios :: left** atau **ios :: right**
5. karakter pengisi dengan fungsi : **fill** diikuti argument karakter
6. tampilan tanda '+' diubah dengan fungsi : **setf** atau **unsetf** diikuti argument : **ios :: showpos**
7. tampilan tanda '.' (titik) bila ada angka dibelakang koma diubah dengan fungsi : **setf** atau **unsetf** diikuti argument **ios :: showpoint**

contoh program :

```
# include <iostream.h>
# pragma hdrstop
Void main ( )
{
Double y = 1234.56789 ;
Cout << "menuliskan bilangan real dengan presisi berbeda' ;
Cout << " \n Presisi 3 = " ;
Cout.precision (3) ;
Cout.width (15) ;
Cout << y ;
Cout << " \n Presisi 7 = " ;
Cout.precision (7) ;
Cout.width (15) ;
Cout << y ;
Cout << " \n \n Menggunakan notasi scientific / fixed " ;
Cout.setf (ios :: scientific | ios :: showpos) ;
Cout << " \n Scientific = " << y ;
Cout.setf (ios :: fixed) ;
Cout << " \n Fixed = ' << y ;
Cout.unsetf (ios :: scientific | ios :: fixed | ios :: showpos) ;
Cout << " \n \n Menggunakan shompoint " ;
Double z = 123 ;
Cout.setf (ios :: showpoint) ;
Cout << " \n Showpoint aktif = " << z ;
Cout.unsetf (ios :: showpoint) ;
Cout << "\n Showpoint non aktif : " << z ;
}
```

Tulisan Berjalan Dan Image Ke Atas

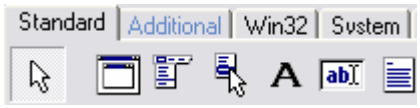
Pada Kesempatan ini team penulis akan menyajikan artikel yang bertemakan tentang " TULISAN BERJALAN DAN IMAGE KE ATAS ". Pembaca tentunya ingin tampilan programnya menarik. Dengan adanya tampilan yang menarik, maka dengan sendirinya akan meningkatkan nilai jual program itu sendiri. Agar tampilan program yang dibuat menjadi menarik, maka pembaca harus mempercantik tampilan programnya. Salah satu caranya yaitu dengan membuat tampilan tulisan berjalan dan image ke atas. Untuk melakukannya tidaklah terlalu rumit. Team penulis akan membantu pembaca dengan

memberikan beberapa langkah sederhana untuk membuat tampilan berjalan dan image ke atas. Program yang dipakai oleh team penulis adalah C++ Builder 6, ini dikarenakan C++ Builder memiliki fiture – fiture yang lengkap. Berikut ini adalah beberapa langkah sederhana untuk membuat tampilan berjalan dan image ke atas.

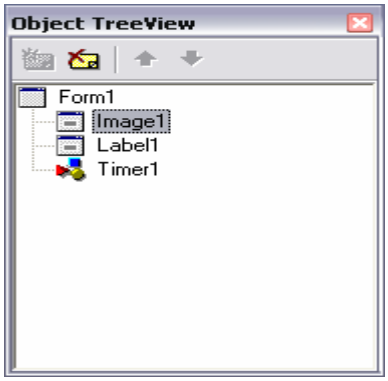
- Langkah yang pertama buka Program C++ Builder 6, kemudian pilih fiture**Timer** pada menu **System** dan masukkan ke dalam Form

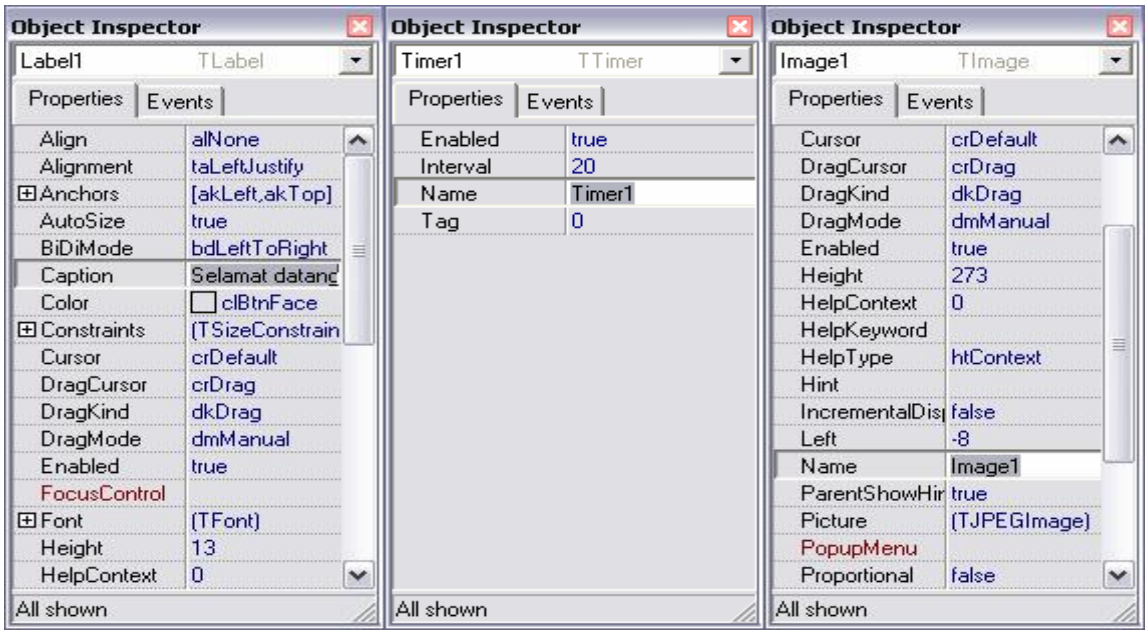
- Untuk langkah yang kedua pilih Fiture **Image** pada menu **additional**, kemudian masukkan ke dalam Form

- Selanjutnya pilih Future **Label** pada menu **Standard** dan masukkan ke dalam Form



- Setelah memasukkan **label**, pilih **Image1** pada **Object TreeView**, kemudian atur Properties pada **Object Inspector** seperti contoh dibawah ini





Kemudian akan muncul tampilan setelah pengaturan Properties pada Object Inspector,sebagai keterangan Image dan Label dapat diganti sesuai kebutuhan

Langkah selanjutnya adalah Klik 2 Kali Pada Icon **Timer1** dan Isikan rumus logika dibawah ini :

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
Image1->Top=Image1->Top>1;
if (Image1->Top=200)
      Image1->Top=77;
Label1->Left=Label1->Left<1;
if (Label1->Left=20)
      Label1->Left=200;
}
//---------------------------------------------------------------------------
```

- Dengan memasukkan rumus logika tersebut maka Image dan Tulisan akan bergerak sesuai dengan perintah di dalam rumus logika tersebut. Hal ini dapat dilihat pada gambar dibawah ini

Selesai sudah pembuatan tulisan berjalan dan Image ke Atas. Selanjutnya Pembaca bisa mengembangkannya sesuai dengan kreasinya sendiri.

- Sekian dulu tips dari team penulis, apabila ada suatu kesalahan haraplah dimaklumi karena "Tidak Ada Manusia Yang Sempurna "dan kami hanyalah ***manusia biasa*** yang masih dalam tahap belajar.
- Tidak lupa Team Penulis mengucapkan Terima Kasih kepada semua pihak yang telah mendukung baik secara moril dan materil sehingga tersajilah artikel ini.
- Akhir kata Team Penulis mengucap syukur Kepada Allah SWT atas karunianya, dan sampai jumpa pada artikel selanjutnya, Wss

Email : creative_kab03_unsri@yahoo.co.id