10/5/2007

UNIKOM

PEMROGRAMAN C

Teknik Informatika | Eko Kurniawan Khannedy

Diktat Mata Kuliah Pemrograman I

BAB I
PENDAHULUAN

IF

Apakah Bahasa C Itu ?

Bahasa C atau C++ adalah suatu bahasa pemrograman. Bahasa C termasuk sebagai bahasa pemrograman tingkat menengah, maksudnya bahasa C bisa dipelajari dengan lebih mudah karena mudah dimengerti tetapi mempunyai kemampuan yang tinggi.

Bahasa C bisa digunakan untuk merekayasa program untuk segala kebutuhan, baik untuk aplikasi bisnis, matematis atau bahkan game.

Semua bahasa mempunyai kelemahan atau kelebihan sendiri-sendiri. Begitu juga dengan bahasa C. Adapun sebagian kelebihan dari bahasa C adalah sebagai berikut :

- Banyak memiliki operator untuk mengolah / memanipulasi data.
- Bahasa C termasuk sebagai bahasa yang terstruktur sehingga program dapat lebih mudah dipahami atau dikembangkan.
- Bahasa C lebih mudah dimengerti karena lebih mirip kepada bahasa manusia.
- Kecepatan eksekusi tinggi.
- Mengenal data pointer.

Sedangkan kelemahan dari bahasa C adalah :

- Banyaknya operator atau cara penulisan program kadang menimbulkan kebingungan para pemakainya.
- Perlunya ketelitian dalam penulisan program karena perintah (statement) dalam bahasa C bersifat case sensitiv (huruf kapital dan huruf kecil dibedakan.

Kekurangan-kekurangan tersebut akan banyak terjadi pada awal-awal mempelajari bahasa C. Tetapi setelah sering membuat program atau mempelajarinya maka kesulitan tersebut sedikit demi sedikit akan berkurang.

Memanggil Program Turbo C

Untuk masuk ke dalam lingkungan Turbo C++, ada beberapa cara yang bisa dilakukan, cara-cara berikut disusun dengan asumsi bahwa pemakai sedang memakai sistem operasi window.

1. Fasilitas **Run** dari start menu
   - Klik tombol **Start** kemudian pilih Menu **Run**
   - Klik tombol **Browse** kemudian cari folder yang didalamnya terdapat file TC.EXE. Biasanya berada pada C:\TC\BIN atau C:\TC atau C:\TURBOC3, kemudian klik TC.EXE.
   - Tekan tombol OK, tunggu sampai muncul program C++.
2. MS-DOS Prompt
   - Klik tombol **Start** kemudian menu **Program** lalu pilih **MS-DOS Prompt**.
   - Ketik perintah CD\TC\BIN jika program C++ disimpan pada subdirektori C:\TC\BIN kemudian tekan tombol Enter.
   - Ketik TC.EXE atau TC, kemudian tekan Enter, tunggu sampai muncul program C++.
3. Short Cut
   - Klik tombol kanan mouse di desktop kemudian pilih sub menu **New** pada menu popup kemudian Pilih **Short Cut**.
   - Klik tombol **Browse** kemudian cari file TC.EXE di subdierktori C:\TC\BIN atau C:\TC. Kemudian klik tombol **Next**, tulis nama short cutnya, kemudian klik tombol **Next**. Pilih icon short cut kemudian tekan tombol **Finish**.
   - Untuk menjalankannya, double Klik di short cut yang telah dibuat.

Diktat Mata Kuliah Pemrograman I

BAB II
PROGRAM PERTAMA

IF

Contoh Program Berbahasa C

Untuk membuat program dalam C++ maka langkah pertama adalah memanggil program C seperti yang telah dijelaskan pada Bab I.

Tulis program berikut pada layar editornya.

```
/*
  Program Ke –1
  Nama File : Lat-1.CPP
*/
// Program Hello
#include "stdio.h"
#include <conio.h>
main()
{
   printf("Ini Program Saya Yang Pertama\n");
   printf("\nSaya Belajar Bahasa C++\n Di \"UNIKOM\"\nBandung.");
   getch();
   return 0;
}
```

Program Ke-1. Lat-1.CPP

Setelah selesai menulis perintah tersebut simpan program tersebut dengan menekan tombol **F2** atau **Alt+F – Save**. Tulis nama file dengan ketentuan hanya terdiri dari 8 huruf tanpa spasi. Jika telah ditulis tekan tombol Enter atau klik tombol OK. File-file tersebut berekstensi C atau CPP.

Untuk memeriksa apakah program yang telah ditulis itu bisa dimengerti oleh kompiler bahasa C, maka perlu dilakukan langkah **Compile** dengan cara tekan **Menu Compile** atau **Alt+C** kemudian pilih **Compile** atau dengan menekan tombol **Alt+F9**. Jika masih ada kesalahan, maka akan diperlihatkan dibaris mana kesalahan penulisan program terjadi, perbaiki program kemudian *compile* ulang sehingga muncul keterangan bahwa *compile* berhasil/sukses.

Proses *compile* hanya memeriksa program secara bahasa saja dan belum mengeksekusi / menjalankan programnya. Untuk menjalankan program maka tekan menu **Run** kemudian pilih sub menu **Run** atau dengah hotkey Ctrl+F9. Proses Run ini sebenarnya melakukan dua langkah yaitu proses *compile* dan menjalankan programnya.

Setelah proses run terjadi, maka akan tercipta suatu file berekstensi EXE yang mempunyai nama seperti nama file C++ nya. Suatu file EXE dapat langsung

dijalankan di semua komputer walaupun di komputer tersebut tidak terdapat program C++.

Program tersebut ketika dijalankan akan menghasilkan tulisan dilayar sebagai berikut :

```
Ini Program Saya Yang Pertama

Saya Belajar Bahasa C++
Di "UNIKOM"
Bandung.
```

Keterangan Program Lat-1.CPP

- **Komentar**

  Pada baris ke-1, ditemukan tanda /* dan pada baris 4 ditemukan tanda */. Kedua tanda tersebut berpasangan yang berguna untuk menuliskan suatu komentar tentang program atau perintah-perintah. Komentar tidak mempengaruhi program karena komentar tidak dijalankan seperti perintah (statement). Komentar dengan menggunakan tanda /* berlaku sampai ditemukan tanda */. Cara lain untuk memberikan komentar adalah dengan memberikan tanda garis miring 2 kali. Komentar dengan tanda ini hanya berlaku pada 1 baris saja. Komentar bersifat opsional untuk mempermudah orang mengetahui fungsi dari suatu program atau suatu algoritma.

- **#include**

  Pada baris 6 ditemukan perintah #include "stdio.h" dan pada baris 7 terdapat perintah #include <conio.h>. Kedua perintah tersebut digunakan untuk memanggil file header (include file) yang didalamnya terdapat perintah, fungsi atau prototype yang bisa digunakan dalam program yan dibuat. Jika perintah #include ini tidak ditulis, maka komputer tidak mengerti perintah-perintah yang ditulis.

- **Header file**

  Nama file yang digunakan dalam #include seperti conio.h dan stdio.h, disebut sebagai header file karena ditempatkan di paling atas program. Extention H berarti header. Dalam file header ini, terdapat fungsi atau prototipe yang bisa digunakan dalam program. Sebuah file header memiliki lebih dari 1 fungsi atau variabel global.

  File header stdio.h digunakan untuk penanganan input / output standar seperti penulisan ke layar, ke file atau pembacaan data dari keyboard atau file.

File header stdio.h digunakan untuk penanganan ke layar seperti pengaturan warna, waktu jeda (delay), suara internal.

Masih banyak file header standar selain stdio.h dan conio.h.

- **Fungsi main()**

  Pada baris 8 terdapat pendeklarasian fungsi main(). Fungsi ini adalah suatu fungsi khusus yang akan dieksekusi pertama. Setiap program harus mempunyai fungsi main(). Fungsi main() diawali dengan tanda { yang menyatakan awal dari program dan diakhiri dengan tanda } yang menyatakan akhir dari program.

- **printf()**

  printf() adalah suatu fungsi yang berguna untuk menulis pesan ke layar. Pesan yang akan ditulis dalam diapit oleh tanda kutip. Pesan yang tertulis dapat diatur dengan mengatur format dari penulisannya. Fungsi printf() tidak hanya menulis pesan dalam 1 baris saja tetapi bisa lebih.

  Untuk berpindah baris maka gunakan perintah \n yang berarti new line (baris baru). Penulisan \n boleh ditempatkan di depan, ditengah atau diakhir.

  Untuk menuliskan tanda " (kutip) maka harus digunakan tanda \".

  Keterangan lebih lanjut akan diterangkan dalam bab-bab berikutnya.

- **Tanda ; (semikolon)**

  Setiap perintah harus diakhiri dengan tanda ;. Hilangnya tanda ; akan menyebabkan kesalahan kompile.

- **getch()**

  getch() adalah suatu fungsi yang berfungsi untuk pembacaan data sebuah karakter, sehingga program akan terdiam sampai pengguna menekan suatu tombol. Fungsi ini berada dalam file header conio.h sehingga perintah #include "conio.h" harus dituliskan. Kalau perintah getch() tidak ditulis, maka program akan dikerjakan dengan cepat dan eksekusi tidak dapat terlihat.

- **return**

  return adalah perintah yang memberikan nilai kepada fungsinya. Setiap fungsi harus mempunyai nilai kembaliannya (return value).

Catatan 	Kalau eksekusi yang dilakukan cepat, untuk melihat hasil akhir dari eksekusi program dapat dilakukan dengan cara menekan Alt-F5 atau dengan memilih menu **Window** kemudian pilih sub menu **User Screen**.

Diktat Mata Kuliah Pemrograman I

BAB III
TIPE DATA, VARIABEL & OPERASI PERHITUNGAN

IF

Tipe Data

Komputer bisa diartikan sebagai alat untuk menghitung. Untuk melakukan proses perhitungan tersebut, maka dibutuhkan data yang akan diproses. Tipe data ada beberapa jenis yaitu :

- Tipe data karakter
- Tipe data bilangan bulat.
- Tipe data bilangan pecahan.

Jika kita membutuhkan suatu tipe data yang baru yang tidak terdapat pada tipe data standar, maka kita dapat membuat tipe data baru dengan menggunakan perintah **struct**. Perintah struct akan dijelaskan pada bab selanjutnya.

Setiap tipe data mempunyai jangkauan nilai yang berbeda.

1. Tipe data karakter

Untuk tipe data karakter kita gunakan perintah char.

Contoh

```
char karakter;
char kar1,kar2,kar3;
char kar4='A';
char kar5=65;
```

Tipe data ini mempunyai jangkauan dari 0 sampai 255 atau karakter ASCII ke 0 sampai karakter ASCII 255. Tipe data karakter bisa ditampilkan sebagai suatu karakter atau sebagai suatu bilangan. Hal ini tergantung dari bagaimana penulisannya apakah dianggap sebagai karakter atau sebagai bilangan.

Untuk menuliskan isi dari data bertipe char adalah dengan menggunakan printf dengan format penulisannya menggunakan tanda %c kalau ingin ditampilkan sebagai suatu karakter atau dengan %i jika ingin ditampilkan sebagai suatu angka.

Pemberian nilai kepada suatu karakter digunakan perintah sebagai berikut :

```
karakter='A';
```

Atau

```
karakter=65;
```

Kedua cara tersebut menghasilkan suatu efek yang sama yaitu memberikan nilai 65 atau karakter A ke variabel karakter. Kalau pengisian variable ingin menggunakan karakter maka karakter yang akan dimasukan harus diapit dengan tanda apostrof.

Untuk melihat nilai yang ada dalam suatu variable yang bertipe char gunakan perintah berikut :

```
printf("Karakter dilihat dalam bentuk karakter = %c.\n",karakter);
printf("Karakter dilihat dalam bentuk angka = %d.\n",karakter);
```

Contoh program

```
//Program Ke-2 Nama File : Lat2.CPP
#include <stdio.h>
#include <conio.h>
main()
{
  char k1,k2;
  k1='A';
  k2=k1;
  printf("Nilai variable K1 adalah %c\n",k1);
  printf("Nilai variable K2 dalam bentuk angka = %d\n",k2);
  getch();
  return 0;
}
```

Hasil dari eksekusi program adalah :

```
Nilai variable K1 adalah A
Nilai variable K2 dalam bentuk angka = 65
```

Keterangan program Lat2.CPP

Perintah "`char k1,k2;`" pada baris 6 berarti program memesan 2 buah tempat di memori untuk menyimpan data bertipe karakter dengan nama k1 dan k2.

Perintah "`k1='A';`" pada baris 7 adalah perintah untuk memasukan nilai karakter A kapital ke dalam variable k1 sehingga untuk baris berikutnya k1 berisi karakter A kapital atau angka 65.

Perintah "`k2=k1;`" pada baris 8 berarti bahwa nilai k2 diisi dari nilai k1 sehingga isi k2 sama dengan isi variable di k1.

Perintah printf pada baris 9 berarti perintah penulisan ke layar sesuai dengan format "Nilai variable k1 adalah %c\n". Karakter %c tidak dicetak sebagai %c tetapi akan diganti dari variable yang sesuai dengan urutannya yaitu k1 dalam bentuk karakter. Perintah printf pada baris 10 cara kerjanya sama dengan perintah printf pada baris 9 bedanya hanya tanda %d berasal ditulis berdasarkan

isi variable k2 yang dicetak dalam bentuk angka bukan karakter. Tanda %d digunakan untuk format pencetakan data dalam bentuk bilangan bulat.

Perintah **getch()** digunakan untuk menunggu sampai pengguna menekan sembarang karaker.

Perintah **return** digunakan untuk memberikan nilai kembalian dari fungsi main().

2. Tipe data bilangan bulat

Ada beberapa tipe data standar yang digunakan untuk data bilangan bulat.

Tipe Data	Memori	Format	Jangkauan Nilai
int	2 byte	%d/%i	-32.768 s/d 32.767
unsigned int	2 byte	%u	0 s/d 65.535
char	1 byte	%d/%i	-128 s/d 127
unsigned char	1 byte	%u	0 s/d 255
unsigned long	4 byte	%lu	0 s/d 4.294.967.295
long	4 byte	%ld/%li	-2.147.483.648 s/d 2.147.483.647

Tipe-tipe data yang ada dalam table tersebut khusus untuk data yang nilai bilangannya bulat. Cara pendeklarasian tipe data ini sama seperti pendeklarasian lainnya, yaitu :

```
int a;
unsigned int b;
char c;
long d;
```

Contoh Program :

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a=1000,b=64000;
    unsigned int c=64000;
    printf("Nilai yang telah dimasukan\na: %i dan b: %i\n",a,b);
    printf("Nilai yang telah dimasukan : %u\n",c);
    getch();
    return 0;
}
```

Perintah di atas akan menampilkan hasil seperti di bawah ini :

```
a: 1000 dan b: -1536
Nilai yang telah dimasukan : 64000
```

3. Tipe data bilangan pecahan

Tipe data untuk bilangan pecahan terdiri dari beberapa jenis yaitu :

Tipe Data	Memori	Format	Jangkauan Nilai
float	4 byte	%f	3.4*(10^-38) - 3.4*(10^+38
double	8 byte	%f	1.7*(10^-308) - 1.7*(10^+308)
long double	10 byte	%lf	3.4*(10^-4932) - 1.1*(10^+4932)

Contoh Program

```
#include <stdio.h>
#include <conio.h>
main()
{
    float  a=1234567890123456789;
    double d=1234567890123456789;
    printf("Nilai a adalah : %30.20f\n",a);
    printf("Nilai d adalah : %30.20f\n",d);
    getch();
    return 0;
}
```

Hasil eksekusi program dapat dilihat di bawah ini :

```
Nilai a adalah :         1234567939550609410.00
Nilai d adalah :         1234567890123456770.00
```

4. Tipe data string

Dalam pemrograman C, untuk variabel yang menampung data string tidak ada perintah khusus, karena dalam bahasa C, string adalah sebuah array karakter atau sebuah pointer ke sebuah variabel char.

Cara pendeklarasian adalah :

```
char nama[50];
char *alamat;
```

Contoh program :

```
#include <stdio.h>
#include <conio.h>
main()
{
   char nama[50]; //deklarasi dengan cara array
   char *alamat; //deklarasi dengan cara pointer
   printf("Nama : ");scanf("%s",nama);
   printf("Alamat : ");gets(alamat);
   printf("Data yang telah dimasukan adalah : \n");
   printf("Nama : %s\nAlamat : %s\n",nama,alamat);
   getch();
   return 0;
}
```

Catatan	Pemilihan tipe data harus hati-hati. Pertimbangkan jangkauan yang dimiliki oleh tipe data yang dipilih. Kesalahan dalam memilih tipe

data akan menimbulkan suatu hasil yang tidak diperkirakan. Contoh :

```
int a=32000;
int b=769;
int c;
c=a+b;
printf("%i + %i = %i\n",a,b,c);
```

Jika program tersebut dijalankan, maka akan menghasilkan output seperti berikut :

32000 + 769 = -32767

Hal tersebut terjadi karena jangkuan nilai c sudah melebihi jangkauan nilai untuk sebuah tipe data int. Bila suatu variable telah melebihi jangkauan nilainya maka nilai variable tersebut akan berputar menjadi nilai minimalnya dan jika nilainya kurang dari minimal jangkauan nilainya maka variable tersebut akan terisi oleh bilangan maksimal tipe tersebut.

Nilai yang diharapkan 32767 32768 32769 Nilai pada variable C 32767 -32768 -32767			

Operator-Operator Perhitungan

Untuk melakukan perhitungan-perhitungan data, maka diperlukan operator-operator perhitungannya. Operator-operator yang paling umum dipakai dalam pemrograman dengan bahasa C adalah :

Operator	Contoh	Arti
+	c=a+b	Variable c diisi dari isi variable a ditambah isi variable b
-	c=a-b	Variable c diisi dari isi variable a dikurangi isi variable b
*	c=a*b	Variable c diisi dari isi variable a dikali dengan isi variable b
/	c=a/b	Variable c diisi dari isi variable a dibagi oleh isi variable b
++	a++	Isi variable a ditambah 1. Perintah ini sama dengan a=a+1 atau a+=1
--	b--	Isi variable a dikurang. Perintah ini sama dengan a=a-1 atau a-=1
%	c=a % b	Variable c diisi dari sisa pembagian variable a dibagi variable b
+=	c+=a	Variable c ditambah dengan isi variable a. Sama dengan c=c+a
/=	c/=a	Variable c dibagi dengan isi variable a. Sama dengan c=c/a
-=	c-=a	Variable c dikurangi dengan isi variable a. Sama dengan c=c-a
*=	c*=a	Variable c dikali dengan isi variable a. Sama dengan c=c*a
%=	c%=a	Variable c diisi dari sisa pembagian c dibagi isi variable a. Sama dengan c=c%a

Contoh program :

```
#include <stdio.h>
#include <conio.h>
main()
{
   int x=20, y=8, z;
   clrscr();
   printf("X = %d dan Y = %d\n",x,y);
   printf("X / Y = %d\n",x/y);
   printf("X %% Y = %d\n", x % y);
   x+=2;
   printf("Nilai X sekarang : %i\n",x);
   x++;
   printf("Nilai X setelah X++ : %i\n",x);
   printf("Nilai Y : %d\n",y++);
   printf("Nilai Y setelahnya : %d\n",y);
   z=++x;
   printf("Nilai Z : %d\n",z);
   getch();
   return 0;
}
```

Program di atas akan menampilkan hasil seperti berikut :

```
X = 20 dan Y = 8
X / Y = 2
X % Y = 4
Nilai X sekarang : 22
Nilai X setelah X++ : 23
Nilai Y : 8
Nilai Y setelahnya : 9
Nilai Z : 24
```

Opr.	Istilah	Keterangan
I++	Post increment	Nilai I dikeluarkan dulu, kemudian I ditambah 1
++I	Pre increment	Nilai I ditambah 1 dulu, kemudian nilainya dikeluarkan
I--	Post decrement	Nilai I dikeluarkan dulu, kemudian I dikurangi 1
--I	Pre decrement	Nilai I dikurangi 1 dulu, kemudian nilainya dikeluarkan

Operator-Operator Manipulasi Bit

Untuk keperluan memanipulasi data dalam bentuk bit, Turbo C menyediakan operator-operator berikut :

Operator	Operasi
<<	Geser bit ke kiri
>>	Geser bit ke kanan
&	Dan (AND)
|	Atau (OR)
^	XOR
~	NOT (Komplemen)

Seluruh operator manipulasi bit hanya dikenakan pada operand / variabel / angka yang bertipe bilangan bulat (int, char, long).

Prioritas eksekusinya adalah

Tertinggi	~
	<< >>
	&
	^
Terendah	|

Tabel Kebenaran dari tiap operator manipulasi bit adalah :

A	B	~A	A & B	A | B	A ^ B
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Contoh :

```
#include <stdio.h>
#include <conio.h>
main()
{
    unsigned int x,y,and,or,not,xor;
    x=78;// 41h
```

```
    y=520;// 208h

    and=x & y;
    or=x | y;
    xor=x ^ y;
    not = ~y;
    clrscr();
    printf("x     : %6u : %4Xh\n",x,x);
    printf("y     : %6u : %4Xh\n",y,y);
    printf("-------------------------\n");
    printf("x & y : %6u : %4Xh\n",and,and);
    printf("x | y : %6u : %4Xh\n",or,or);
    printf("x ^ y : %6u : %4Xh\n",xor,xor);
    printf("  ~y  : %6u : %4Xh\n",not,not);
    getch();
    return 0;
}
```

Jika di-Run akan menghasilkan :

```
x     :     78 :   4Eh
y     :    520 :  208h
-------------------------
x & y :      8 :    8h
x | y :    590 :  24Eh
x ^ y :    582 :  246h
  ~y  :  65015 : FDF7h
```

Pembuktian :

```
x     :     78 :   4eh : 00000000 01001110
y     :    520 :  208h : 00000010 00001000
x & y :             x : 00000000 01001110
                    y : 00000010 00001000
                        ----------------- & (and)
                x & y : 00000000 00001000 = 8h = 8 (terbukti)
x | y :             x : 00000000 01001110
                    y : 00000010 00001000
                        ----------------- | (or)
                x | y : 00000010 01001110 = 24eh = 590 (terbukti)
x ^ y :             x : 00000000 01001110
                    y : 00000010 00001000
                        ----------------- ^ (xor)
                x ^ y : 00000010 01000110 = 246h = 582 (terbukti)
  ~y  :             y : 00000010 00001000
                        ----------------- ~ (komplemen)
                  ~y : 11111101 11110111 = FDF7h = 65015 (terbukti)
```

Operator-operator pergeseran bit, berguna untuk menggeserkan bit yang ada dalam suatu variabel.

Contoh :

```
#include <stdio.h>
#include <conio.h>
main()
{
    unsigned int angka,x,y;
    angka=50;
    x=angka << 2;
    y=angka >> 2;
    clrscr();
    printf("Angka        : %5u : %xh\n",angka,angka);
    printf("x=angka << 2 : %5u : %xh\n",x,x);
    printf("y=angka >> 2 : %5u : %xh\n",y,y);
    getch();
    return 0;
}
```

Jika di-Run akan menghasilkan :

```
Angka        :    50 : 32h
x=angka << 2 :   200 : c8h
x=angka >> 2 :    12 : ch
```

Pembuktian :

```
Angka    :  50 : 00000000 00110010
x=angka << 2   : 00000000 11001000  = 128 + 64 + 8 = 200 : c8h
x=angka >> 2   : 00000000 00001100  = 8 + 4        =  12 : ch
```

Aturan-Aturan Perhitungan

Perhatikan perintah berikut :

```
float a;
a= 9/5;
```

Jika anda mengharapkan bahwa nilai yang didapat adalah 1.8, maka anda akan kecewa, karena angka yang didapat adalah 1. Kenapa ini terjadi?.

Ada aturan-aturan pengkonversian data yang berlaku dalam suatu operasi perhitungan, diantaranya :

1. Jika suatu bilangan bulat dioperasikan dengan bilangan bulat, maka nilai yang didapat adalah bilangan bulat pula.
2. Operasi perhitungan dilakukan berdasarkan tipe bilangan yang terbesarnya. Jadi jika ada suatu perhitungan antara int dengan long, maka komputer akan memperlakukan int sebagai long.

Untuk mengkonversi suatu variabel menjadi suatu variabel yang berbeda tipe, maka bisa dilakukan dengan type cast. Caranya adalah dengan menulis tipe data yang diinginkan diapit dengan tanda kurung. Contoh :

```
float a,b;
a=(float)9/5;
b=(float)(9/5);
```

Pada perintah a=(float) 9/5, maka angka 9 akan dikonversikan menjadi float sehingga perintah tersebut akan menghasilkan nilai 1.8, tetapi jika perintah b=(float)(9/5) dikerjakan maka akan menghasilkan nilai 1.0 karena yang dikerjakan duluan adalah 9/5 yang menghasilkan nilai 1 yang kemudian dikonversikan ke dalam bentuk float.

Konversi tipe data juga terjadi dalam operasi penugasan / pengisian data terhadap variabel. Perhatikan perintah berikut :

```
#include <stdio.h>
#include <conio.h>
main()
{
    char c;
    int i;
    float f;
    f=65.9;
    i=f;
    c=i;
    printf("F  : %f\n",f);
    printf("I  : %d\n",i);
    printf("C  : %c\n",c);
    getch();
    return 0;
}
```

Jika dieksekusi, akan menghasilkan :

```
F  : 65.900002
I  : 65
C  : A
```

Keterangan :

- **`f=65.9;`** pengisian nilai 65.9 ke variabel f
- **`i=f;`** pengisian nilai f ke variabel i. Dalam baris ini terjadi konversi dari float ke int. Pengkonversian float ke int selalu menghilangan angka pecahannya, dan tidak terjadi pembulatan.
- **`c=i;`** pengisian nilai i ke variabel c. Dalam baris ini terjadi konversi dari int ke char.

Mendefinisikan Konstanta Simbolis

Untuk mendefinisikan suatu konstanta, perintah yang bisa dipakai adalah perintah $define diikuti dengan nama konstanta dan isinya.

Contoh :

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

#define PI 3.14
#define pembuat "Andri Heryandi"
main()
{
    float radius=10,keliling,luas;
    keliling=2*PI*radius;
    luas=PI*radius*radius;
    printf("Perhitungan Lingkaran\nDibuat Oleh : %s\n",pembuat);
    printf("======================\n");
    printf("Radius   : %6.2f\n",radius);
    printf("Keliling : %6.2f\n",keliling);
    printf("Luas     : %6.2f\n",luas);
    getch();
    return 0;
}
```

Jika dieksekusi, akan menghasilkan :

```
Perhitungan Lingkaran
Dibuat Oleh : Andri Heryandi
======================
Radius   :  10.00
Keliling :  62.80
Luas     : 314.00
```

Ketika program dieksekusi, maka compiler akan mengganti semua PI dengan nilai 3.14 dan pembuat dengan "Andri Heryandi". Jadi ketika kita ingin menggunakan nilai PI yang lebih exact, maka pada define PI, nilai 3.14 diganti dengan 22.0/7.

	Diktat Mata Kuliah Pemrograman I	
	BAB IV INPUT DAN OUTPUT DATA	IF

Pemasukan (Input) Data

Umumnya suatu program mempunyai proses pemasukan data. Dalam program berbahasa C, pemasukan data dapat dilakukan dengan perintah scanf. Fungsi scanf merupakan fungsi yang dapat digunakan untuk memasukan berbagai jenis data, tergantung dengan format penentunya.

Format-format penentu tipe data yang umum dipakai adalah :

Format	Kegunaan
%c	Digunakan untuk pemasukan data bertipe char
%i atau %d	Digunakan untuk pemasukan data bertipe int, char.
%u	Digunakan untuk pemasukan data berupa unsigned int atau unsigned char.
%f	Digunakan untuk pemasukan data berupa bilangan pecahan (float)
%o	Digunakan untuk pemasukan data angka berbasis oktal
%x	Digunakan untuk pemasukan data angka berbasis hexadesimal
%s	Digunakan untuk pemasukan data berupa string.

Bentuk umum penggunaan fungsi scanf adalah

```
scanf("format",&namavariabel);
```

Contoh :

```
int i,jam,menit,detik;
unsigned int j;
float f;
char nama[60];
scanf("%i",&i);
scanf("%u",&j);
scanf("%f",&f);
scanf("%i %i %i",&jam,&menit,&detik);
scanf("%s",nama);
```

Fungsi scanf() kurang begitu bagus jika dipakai untuk pembacaan string. Karena data yang tersimpan adalah hanya sampai menemukan spasi, maksudnya jika kita mengisikan 2 buah kata dengan pemisah spasi, maka data yang masuk ke variabel tersebut hanyalah kata yang pertama.

Oleh karena itu, pembacaan data bertipe string biasanya menggunakan perintah gets() yang bentuk umumnya adalah :

```
gets(namavariabel);
```

Contoh :

```
gets(nama);
gets(alamat);
```

Untuk pembacaan data bertipe char, selain dengan menggunakan scanf() dengan format %c, bisa juga dengan menggunakan fungsi getch() atau getche(). Perbedaan dari getch() dan getche() adalah getch() digunakan untuk membaca data bertipe char tanpa menampilkannya di layar, dan getche() digunakan untuk membaca data bertipe char dengan menampilkan data karakternya di layar.

Contoh :

```
#include <stdio.h>
#include <conio.h>
main()
{
    char a,b;
    printf("Masukan Huruf pertama : ");
    a=getch();
    printf("Masukan Huruf kedua : ");
    b=getche();
    printf("Data yang dimasukan adalah %c dan %d\n",a,b);
    getch();
    return 0;
}
```

Pengeluaran (Output) Data

Untuk output data, perintah yang bisa dipakai adalah printf(). Untuk menampilkan data dengan fungsi printf(), kita harus mengatur format tampilannya, dengan format-format penentu. Untuk lebih jelas perhatikan program di bawah ini.

```
#include <stdio.h>
#include <conio.h>
main()
{
     int a=25000;
     unsigned int b=50000;
     float c=12345.678;
     char nama[50]="Universitas Komputer Indonesia";
     char alamat[10]="Bandung";
     clrscr();
     printf("Penampilan data tanpa di format\n");
     printf("Nilai a : %d\n",a);
     printf("Nilai b : %u\n",b);
     printf("Nilai c : %f\n",c);
     printf("String  : %s %s\n",nama,alamat);//rata kanan
     printf("Penampilan data setelah di format\n");
     printf("Nilai a : %8d\n",a);
     printf("Nilai b : %8u\n",b);
     printf("Nilai c : %11.2f\n",c);
     printf("String 1: %40s %10s\n",nama,alamat);//rata kanan
     printf("String 2: %-40s %-10s\n",nama,alamat);//rata kanan
     getch();
     return 0;
}
```

Program di atas akan menampilkan hasil eksekusi seperti di bawah ini :

```
Penampilan data tanpa di format
Nilai a : 25000
Nilai b : 50000
Nilai c : 12345.677734
String  : Universitas Komputer Indonesia Bandung
Penampilan data setelah di format
Nilai a :    25000
Nilai b :    50000
Nilai c :    12345.68
String 1:          Universitas Komputer Indonesia    Bandung
String 2: Universitas Komputer Indonesia           Bandung
```

Contoh Program

Contoh Kasus :

Di suatu perusahaan, data penggajian dihitung dengan ketentuan sebagai berikut :

Gaji Pokok : Rp. 5000000

Gaji Lembur/jam : Rp. 5000

Total Gaji Lembur : Lama Lembur * Gaji Lembur/jam

Gaji Kotor : Gaji Pokok + Total Gaji Lembur

Pajak : 10%

Gaji Bersih : Gaji Kotor - Pajak

Data yang diinputkan adalah : Nama Pegawai, Lama Lembur.

Program ke-1 (tanpa memformat tampilan data).

```
#include <stdio.h>
#include <conio.h>
main()
{
   int jamlembur;
   long int gajipokok=500000,gajikotor,totalgajilembur;
   float pajak,gajibersih;
   char nama[50];
   clrscr();
   printf("Nama Pegawai : ");gets(nama);
   printf("Lama Lembur  : ");scanf("%i",&jamlembur);
   totalgajilembur=(long int)5000*jamlembur;
   gajikotor=gajipokok+totalgajilembur;
   pajak=0.1*gajikotor;
   gajibersih=gajikotor-pajak;
   clrscr();
   printf("Hasil Perhitungan\n");
   printf("Nama Pegawai      : %s\n",nama);
   printf("Gaji Pokok        : Rp. %li\n",gajipokok);
   printf("Lama Lembur       : %i jam\n",jamlembur);
   printf("Total Gaji Lembur : Rp. %li\n",totalgajilembur);
   printf("Gaji Kotor        : Rp. %li\n",gajikotor);
   printf("Pajak (10%%)       : Rp. %f\n",pajak);
   printf("Gaji Bersih       : Rp. %f\n",gajibersih);
   getch();
   return 0;
}
```

Program di atas akan menghasilkan tampilan program seperti di bawah ini :

```
Hasil Perhitungan
Nama Pegawai      : Shelly Septiani
Gaji Pokok        : Rp. 500000
Lama Lembur       : 50 jam
Total Gaji Lembur : Rp. 250000
Gaji Kotor        : Rp. 750000
Pajak (10%)       : Rp. 75000.000000
Gaji Bersih       : Rp. 675000.000000
```

Program Ke-2 (dengan memformat tampilan datanya).

```
#include <stdio.h>
#include <conio.h>
main()
{
   int jamlembur;
   long int gajipokok=500000,gajikotor,totalgajilembur;
   float pajak,gajibersih;
   char nama[50];
   clrscr();
   printf("Nama Pegawai : ");gets(nama);
   printf("Lama Lembur  : ");scanf("%i",&jamlembur);
   totalgajilembur=(long int)5000*jamlembur;
   gajikotor=gajipokok+totalgajilembur;
   pajak=0.1*gajikotor;
   gajibersih=gajikotor-pajak;
   clrscr();
   printf("Hasil Perhitungan\n");
   printf("Nama Pegawai      : %s\n",nama);
   printf("Gaji Pokok        : Rp. %10li\n",gajipokok);
   printf("Lama Lembur       : %i jam\n",jamlembur);
   printf("Total Gaji Lembur : Rp. %10li\n",totalgajilembur);
   printf("Gaji Kotor        : Rp. %10li\n",gajikotor);
   printf("Pajak (10%%)       : Rp. %10.0f\n",pajak);
   printf("Gaji Bersih       : Rp. %10.0f\n",gajibersih);
   getch();
   return 0;
}
```

Tampilan data ketika program di atas di eksekusi.

```
Hasil Perhitungan
Nama Pegawai      : Shelly Septiani
Gaji Pokok        : Rp.     500000
Lama Lembur       : 50 jam
Total Gaji Lembur : Rp.     250000
Gaji Kotor        : Rp.     750000
Pajak (10%)       : Rp.      75000
Gaji Bersih       : Rp.     675000
```

Diktat Mata Kuliah Pemrograman I

BAB X
STRING DAN MANIPULASINYA

IF

String ?

String adalah sebuah array yang bertipe char yang diakhiri dengan karakter null (\0).

Sebagai contoh, deklarasi dibawah ini merupakan deklasai sebuah array yang bertipe char, dan bisa disamakan dengan deklarasi sebuah string dengan nama st.

```
char array_ch[7] = {`H', `e', `l', `l', `o', `!', `\0'};
```

Dalam C, karakter null dapat digunakan untuk menandai akhir sebuah string. Setiap karakter akan membutuhkan 1 byte dalam memori.

Sekumpulan karakter-karakter yang diapit dengan kutip ganda ("") disebut sebagai konstanta string. C akan secara otomatis menambahkan karakter null pada setiap akhir konstanta string untuk menandakan akhir dari sebuah string.

Mendeklarasikan String

Cara pendeklarasian variabel-variabel yang berjenis string dapat dilakukan dengan cara sebagai berikut :

```
char nama[21];
char *nama2;
```

Kedua cara tersebut dapat dipergunakan. Cara ke-1 adalah dengan membuat suatu array char sebanyak 21 karakter. Pada langkah ini, variabel nama hanya diperbolehkan diisi sampai panjangnya 20 karakter karena untuk menutup suatu string membutuhkan satu byte untuk karater null.

Cara ke-2 adalah dengan membuat suatu variabel ke suatu pointer char, yang menunjuk ke suatu alamat di memori yang berisi data stringnya (isinya).

Cara pendeklarasian yang ke-1 (array) lebih baik dari pada yang pointer, karena kalau membuat string sebagai pointer maka ketika akan mengisikan data maka harus meminta tempat dulu ke memori untuk menampung datanya contohnya dengan malloc, karena ketika kita tidak meminta alokasi memori dulu, maka ada kemungkinan data string yang diisikan ke pointer akan mengisi ke suatu tempat yang dimiliki oleh variabel lain.

Inisialisasi String

Cara untuk menginisialisasi string, dapat dilakukan dengan salah satu cara di bawah ini :

```
char nama[]="Ini adalah string";
char nama2[]={'i','n','i', ' ','s','t','r','i','n','g','\0'};
char nama3[5]="BUDI";
char nama4[5]={'B','u','d','i','\0'}
char *nama5="Ini juga string";
```

Untuk mengisi suatu string caranya adalah :

```
strcpy(nama,"Ini string");
nama5="Ini juga string";
```

Coba diperhatikan, untuk string yang dideklarasikan sebagai sebuah array karakter, pengisian nilainya adalah dengan menggunakan suatu perintah **strcpy** yang berguna untuk mengisikan suatu string ke string lain. Pengisiannya tidak boleh langsung. Tetapi jika string dideklarasikan sebagai sebuah pointer karakter, maka pengisiannya boleh diisikan secara langsung.

Fungsi-Fungsi Manipulasi String

- **gets dan puts**

  Fungsi **gets** digunakan untuk membaca data berupa string dari keyboard.

  Fungsi **puts** digunakan untuk menampilkan suatu string ke layar (monitor).

  Contoh program :

```
#include <stdio.h>
int main(void)
{
   char string[80];
   printf("Masukan Sebuah string:");gets(string);
   puts(string);
   return 0;
}
```

  Hasil run program :

```
Masukan Sebuah string:String adalah sekumpulan karakter
String adalah sekumpulan karakter
```

- **strlen**

Fungsi **strlen** digunakan untuk mengetahui panjang suatu string.

Contoh progam :

```
#include <stdio.h>
#include <string.h>
int main(void)
{
   char string[80];
   int panjang;
   printf("Masukan String: ");gets(string);
   panjang=strlen(string);
   printf("Panjang String adalah %i karakter\n",panjang);
   return 0;
}
```

Hasil run program :

```
Masukan String: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Panjang String adalah 26 karakter
```

- **strcpy dan strncpy**

Fungsi **strcpy** berfungsi untuk menyalin isi suatu string ke string lain.

Fungsi **strncpy** berfungsi untuk menyalin isi suatu string ke string lain sebanyak n karakter.

Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char string[31];
   char *str1 = "Ini adalah sebuah string";
   char str2[31];
   strcpy(string, str1);
   printf("Isi String : %s\n", string);
   strncpy(str2,str1,10);
   str2[10]='\0';// menutup string
   printf("Isi Str2 : %s",str2);
   getch();
   return 0;
}
```

Hasil run program :

```
Isi String : Ini adalah sebuah string
Isi Str2 : Ini adalah
```

- **strcmp, strncmp, strcmpi dan strncmpi**

Fungsi **strcmp** digunakan untuk membandingkan 2 buah string secara case sensitive.

Fungsi **strncmp** digunakan untuk membandingkan 2 buah string sebanyak n buah karakter secara case sensitive

Fungsi **strcmpi** digunakan untuk membandingkan 2 buah string secara case insensitive.

Fungsi **strncmpi** digunakan untuk membandingkan 2 buah string sebanyak n buah karakter secara case insensitive.

Semua fungsi tersebut akan menghasilkan sebuah nilai integer yang mempunyai ketentuan :

- Nilai return akan lebih dari 0 (>0) ketika string1 lebih besar dari string2
- Nilai return akan sama dengan 0 (==0) ketika string1 sama dengan string 2
- Nilai return akan kurang dari 0 (<0) ketika string1 lebih kecil dari string 2

Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
 char str1[5]="ABCD",str2[5]="abcd",str3[5]="BCDE",str4[5]="BCda";
   int hasil;
   hasil=strcmp(str1,str2);
   printf("Hasil STRCMP : \n");
   if(hasil==0) printf("String1 sama dengan String2\n"); else
   if(hasil>0) printf("String1 lebih besar dari String2\n"); else
   if(hasil<0) printf("String1 lebih kecil dari String2\n");
   hasil=strcmpi(str1,str2);
   printf("Hasil STRCMPI : \n");
   if(hasil==0) printf("String1 sama dengan String2\n"); else
   if(hasil>0) printf("String1 lebih besar dari String2\n"); else
   if(hasil<0) printf("String1 lebih kecil dari String2\n");
   hasil=strncmp(str3,str4,3);
   printf("Hasil STRNCMP : \n");
   if(hasil==0) printf("String3 sama dengan String4\n"); else
   if(hasil>0) printf("String3 lebih besar dari String4\n"); else
   if(hasil<0) printf("String3 lebih kecil dari String4\n");
   hasil=strncmpi(str3,str4,3);
   printf("Hasil STRNCMPI : \n");
   if(hasil==0) printf("String3 sama dengan String4\n"); else
   if(hasil>0) printf("String3 lebih besar dari String4\n"); else
   if(hasil<0) printf("String3 lebih kecil dari String4\n");
   getch();
   return 0;
}
```

Hasil run program :

```
Hasil STRCMP :
String1 lebih kecil dari String2
Hasil STRCMPI :
String1 sama dengan String2
Hasil STRNCMP :
String3 lebih kecil dari String4
Hasil STRNCMPI :
String3 sama dengan String4
```

- **strcat dan strncat**

  Fungsi **strcat** berfungsi untuk menggabungkan 2 buah string.

  Fungsi **strncat** berfungsu untuk menggabungkan 2 buah string sebanyak n karakter.

  Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char str1[80];
   char str2[15]="Saya Belajar ";
   char str3[10]="Turbo C";
   clrscr();
   strcpy(str1,str2);
   strcat(str1,str3);
   printf("Hasil penggabungan dengan STRCAT  : %s\n",str1);
   strcpy(str1,str2);
   strncat(str1,str3,5);
   printf("Hasil penggabungan dengan STRNCAT : %s\n",str1);
   getch();
   return 0;
}
```

  Hasil Run program :

```
Hasil penggabungan dengan STRCAT  : Saya Belajar Turbo C
Hasil penggabungan dengan STRNCAT : Saya Belajar Turbo
```

- **strlwr dan strupr**

Fungsi **strlwr** berguna untuk mengubah isi string menjadi huruf kecil.

Fungsi **strupr** berguna untuk mengubah isi string menjadi capital.

Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char str1[80]="Saya Belajar Turbo C";
   clrscr();
   printf("Normal    : %s\n",str1);
   strupr(str1);
   printf("UpperCase : %s\n",str1);
   strlwr(str1);
   printf("LowerCase : %s\n",str1);
   getch();
   return 0;
}
```

Hasil run program :

```
Normal    : Saya Belajar Turbo C
UpperCase : SAYA BELAJAR TURBO C
LowerCase : saya belajar turbo c
```

- **strrev**

Fungsi **strrev** berguna untuk membalikan urutan string (reverse).

Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char str1[80]="Saya Belajar Turbo C";
   clrscr();
   printf("Normal  : %s\n",str1);
   strrev(str1);
   printf("Reverse : %s\n",str1);
   getch();
   return 0;
}
```

Hasil run program :

```
Normal  : Saya Belajar Turbo C
Reverse : C obruT rajaleB ayaS
```

- **strset dan strnset**

Fungsi strset berguna untuk mengganti isi suatu string dengan suatu karakter tertentu.

Fungsi strnset berguna untuk mengganti isi suatu string dengan suatu karakter tertentu sebanyak n buah data.

Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char str1[21]="Saya Belajar Turbo C";
   clrscr();
   strnset(str1,'A',12);
   printf("Setelah strnset 12 : %s\n",str1);
   strset(str1,'x');
   printf("Setelah strset     : %s\n",str1);
   getch();
   return 0;
}
```

Hasil run program :

```
Setelah strnset 12 : AAAAAAAAAAAA Turbo C
Setelah strset     : xxxxxxxxxxxxxxxxxxxx
```

- **strstr**

Fungsi strstr berguna untuk mencari urutan pertama suatu string di string lain.

Contoh program :

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
main()
{
   char str1[21]="Saya Belajar Turbo C";
   char str2[6]="Turbo";
   char *str3;
   clrscr();
   str3 = strstr(str1, str2);
   printf("String Asli: %s\n",str1);
   printf("Sub string : %s\n", str3);
   printf("Posisi     : %d\n",str3-str1);
   getch();
   return 0;
}
```

Hasil run program :

```
String Asli: Saya Belajar Turbo C
Sub string : Turbo C
Posisi     : 13
```

Diktat Mata Kuliah Pemrograman I

BAB V
PERCABANGAN

IF

Statement *if - else*

Bentuk dasar perintah if – else adalah sebagai berikut :

```
if (expression)
    Statement1;
else
    Statement2;
StatementBerikutnya;
```

Jika ketika dieksekusi ekspresi menghasilkan nilai true, maka **statement1** akan dieksekusi dan **statement2** tidak akan dikerjakan dan kemudian program akan mengeksekusi **statementberikutnya**, dan jika ekspresi tersebut bernilai false maka **statement1** tidak akan dieksekusi dan **statement2** akan dieksekusi, dan dilanjutkan dengan mengeksekusi **statementberikutnya**.

Operator-operator yang biasa digunakan dalam operasi logika, dapat dilihat di tabel di bawah ini.

Operator	Contoh	Arti
==	A==B	Apakah Isi Variabel A sama dengan Isi Variabel B
!=	A!=B	Apakah Isi Variabel A Tidak Sama Dengan Isi Variabel B
>	A>B	Apakah Isi Variabel A lebih besar dari Isi Variabel B
<	A<B	Apakah Isi Variabel A lebih kecil dari Isi Variabel B
>=	A>=B	Apakah Isi Variabel A lebih besar atau sama dengan Isi Variabel B
<=	A<=B	Apakah Isi Variabel A lebih kecil atau sama dengan Isi Variabel B
&&	(A<=100) &&(A>=80)	Apakah A lebih kecil atau sama dengan dari 100 dan A lebih besar atau sama dengan 80
||	(A>100)|| (A<0)	Apakah A lebih besar dari 100 atau A lebih kecil dari 0
!	!(A==B)	Apakah A Tidak Sama dengan B

Untuk lebih jelas, perhatikan perintah berikut :

```
#include <stdio.h>
#include <conio.h>
main()
{
   int Nilai;
   printf("Nilai : ");scanf("%i",&Nilai);
   if(Nilai>=50)
     printf("Selamat Anda Lulus.");
   else
     printf("Maaf. Anda Tidak Lulus.");
```

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

Perintah di atas hanya mempunyai 2 kemungkinan yaitu keterangan "Selamat Anda Lulus" jika nilai lebih besar dari atau sama dengan 50 dan keterangan "Maaf. Anda Tidak Lulus", ketika nilai lebih kecil dari 50.

Jika perintah yang akan dieksekusi ketika kondisi tercapai lebih dari 1 perintah, maka perintah-perintah tersebut harus diblok dengan tanda kurawal {}.

Perhatikan program di bawah ini yang merupakan perubahan dari program di atas.

```
#include <stdio.h>
#include <conio.h>
main()
{
   int Nilai;
   printf("Nilai : ");scanf("%i",&Nilai);
   if(Nilai>=50)
   {
       printf("Anda Hebat!\n");
       printf("Selamat Anda Lulus.");
   }
   else
   {
       printf("Maaf. Anda Tidak Lulus.");
       printf("Perbaiki semester depan yah!.");
   }
   getch();
   return 0;
}
```

Perhatikan juga program di bawah ini :

```
#include <stdio.h>
#include <conio.h>
main()
{
   int a,b;
   printf("Masukan A : ");scanf("%i",&a);
   printf("Masukan B : ");scanf("%i",&b);
   if(a==b)
      printf("Isi Variabel A Sama Dengan B");
   else
   if(a>b)
      printf("Isi Variabel A lebih besar dari B");
   else
   if(a<b)
      printf("Isi Variabel A lebih kecil dari B");
   getch();
   return 0;
}
```

Program di atas akan meminta anda untuk memasukan nilai variabel A dan B, kemudian program akan memeriksa apakah variabel A samadengan B, atau A lebih besar dari B, dan A lebih kecil dari B.

Contoh Kasus :

Di sebuah universitas penilaian yang dipakai adalah :

NilaiAkhir=20%*tugas+30%*uts+50%*uas

Nilai Akhir	Index
>=80	A
>=68	B
>=56	C
>=45	D
>=0	E

Diluar nilai di atas, maka index adalah X (index tidak diketahui).

Data yang diinputkan adalah : tugas, uts, uas.

```
#include <stdio.h>
#include <conio.h>
main()
{
   int tugas,uts,uas;
   float nilaiakhir;
   char index;
   printf("Nilai Tugas : ");scanf("%i",&tugas);
   printf("Nilai UTS   : ");scanf("%i",&uts);
   printf("Nilai UAS   : ");scanf("%i",&uas);
   nilaiakhir=0.2*tugas+0.3*uts+0.5*uas;
   printf("Nilai Akhir : %f\n",nilaiakhir);
   if(nilaiakhir>=80)
       index='A';
   else
   if(nilaiakhir>=68)
       index='B';
   else
   if(nilaiakhir>=56)
       index='C';
   else
   if(nilaiakhir>=45)
       index='D';
   else
   if(nilaiakhir>=0)
       index='E';
   else
       index='X';
   printf("Index       : %c\n",index);
   getch();
   return 0;
}
```

Perintah *switch – case - default*

Selain if-else, perintah yang digunakan untuk percabangan adalah switch – case. Bentuk dasar dari perintah tersebut adalah :

```
switch(ekspresi)
{
    case kondisi1:perintah1;break;
    case kondisi2:perintah2:break;
    default : perintah3;
}
```

Cara kerja perintah di atas adalah : "Jika ekspresi sama dengan kondisi1, maka perintah1 akan dieksekusi dan kemudian keluar dari switch, dan jika ekspresi sama dengan kondisi2 maka perintah 2 yang akan dieksekusi dan kemudian keluar dari switch, dan jika tidak 1 kondisi pun yang sama dengan ekspresi maka perintah3 (perintah default) yang akan dieksekusi. Perintah default kalau tidak diperlukan bisa dihilangkan.

Untuk lebih jelas, perhatikan perintah di bawah ini.

```
   switch (index)
   {
       case 'A':printf("Keterangan  : Bagus Sekali\n");break;
       case 'B':printf("Keterangan  : Bagus\n");break;
       case 'C':printf("Keterangan  : Cukup\n");break;
       case 'D':printf("Keterangan  : Kurang\n");break;
       case 'E':printf("Keterangan  : Kurang Sekali\n");break;
       default :printf("Keterangan  : Index Tak Diketahui\n");
   }
```

Keterangan program di atas adalah jika index='A' maka keterangan Bagus Sekali, jika index='B' maka keterangan Bagus, jika index='C' maka keterangan Cukup, jika index='D' maka keterangan Kurang, jika index='E' maka keterangan Kurang Sekali, dan jika index bukan A – E, maka keterangan adalah Index Tidak Diketahui.

Perintah *if else* dengan banyak kondisi

Jika kondisi yang harus diperiksa lebih dari 1 kondisi, maka hanya if-else lah yang bisa dipakai. Operator-operator logika yang dipakai adalah operator && (and), dan operator || (or).

Untuk lebih jelas, perhatikan perintah di bawah ini:

```
if((index=='A')||(index=='B')||(index=='C'))
   printf("Selamat Anda Lulus");
else
if((index=='D')||(index=='E'))
   printf("Anda Tidak Lulus. Lebih giat lagi belajar!");
```

Perintah di atas akan menampilkan string "Selamat Anda Lulus" ketika index berisi A, B atau C, dan akan menampilkan keterangan "Anda Tidak Lulus. Lebih giat lagi belajar!" ketika index berisi D atau E.

Untuk lebih jelas, buatlah program untuk mengatasi kasus di bawah ini.

Di sebuah perusahaan bus, tabel harga dapat dilihat dalam tabel di bawah ini.

		KELAS		
		EKSEKUTIF(1)	BISNIS(2)	EKONOMI(3)
TUJUAN	JAKARTA(1)	70000	40000	10000
	YOGYA(2)	80000	50000	**20000**
	SURABAYA(3)	**90000**	60000	30000

Karena sekarang masa promosi, maka khusus untuk surabaya-eksekutif dan yogya-ekonomi mendapatkan diskon sebesar 10%.

Buatlah program dengan data yang dimasukan adalah jenis kelas, tujuan dan banyak tiket yang dibeli. Data yang ingin ditampilkan adalah harga tiket dan total tiket, diskon dan besar pembayaran.

Contoh :

```
Pilih Jurusan :
1. Jakarta
2. Yogya
3. Surabaya
Jurusan yang anda pilih : 2 [input]
Pilih Kelas   :
1. Eksekutif
2. Bisnis
3. Ekonomi
Kelas yang anda pilih : 3 [input]
Banyak Tiket : 5 [input]
Harga Tiket  : Rp.  20000 [output]
Total Tiket  : Rp. 100000 [output]
Diskon       : Rp.  10000 [output]
Bayar        : Rp.  90000 [output]
```

Program untuk kasus di atas :

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

```
main()
{
   int kodejurusan,kodekelas,banyaktiket;
   long int hargatiket,total;
   float diskon,bayar;
   printf("Pilih Jurusan :\n");
   printf("---------------\n");
   printf("1. Jakarta\n2. Yogya\n3. Surabaya\n");
   printf("---------------\n");
   printf("Jurusan yang dipilih : ");scanf("%i",&kodejurusan);
   printf("Pilih Kelas :\n");
   printf("---------------\n");
   printf("1. Eksekutif\n2. Bisnis\n3. Ekonomi\n");
   printf("---------------\n");
   printf("Kelas yang dipilih : ");scanf("%i",&kodekelas);
   printf("Banyak Tiket : ");scanf("%i",&banyaktiket);
   if((kodejurusan==1)&&(kodekelas==1))
       hargatiket=70000;
   else
   if((kodejurusan==1)&&(kodekelas==2))
       hargatiket=40000;
   else
   if((kodejurusan==1)&&(kodekelas==3))
       hargatiket=10000;
   else
   if(kodejurusan==2)
   {
      if(kodekelas==1) hargatiket=80000; else
      if(kodekelas==2) hargatiket=50000; else
      if(kodekelas==3) hargatiket=20000;
   }
   else
   if(kodejurusan==3)
   {
      switch (kodekelas)
      {
         case 1:hargatiket=90000;break;
         case 2:hargatiket=60000;break;
         case 3:hargatiket=30000;
      }
   }
   printf("Harga Tiket   : Rp. %li\n",hargatiket);
   total=banyaktiket*hargatiket;
   printf("Total Tiket   : Rp. %li\n",total);
   if( ((kodejurusan==3)&&(kodekelas==1)) ||
       ((kodejurusan==2)&&(kodekelas==2))
     )
        diskon=0.1*total;
   else
        diskon=0;
   printf("Diskon 10%%    : Rp. %f\n",diskon);
   bayar=total-diskon;
   printf("Bayar         : Rp. %f\n",bayar);
   getch();
   return 0;
}
```

Diktat Mata Kuliah Pemrograman I

BAB VI
PERULANGAN (LOOP)

IF

Pendahuluan

Untuk memahami mengenai fungsi perulangan, coba lihatlah kasus sebagai berikut :

Buatlah suatu program untuk menampilkan angka dari 1 sampai dengan 5. Maka untuk kasus tersebut program yang buat adalah sebagai berikut :

```
#include <stdio.h>
#include <conio.h>
main()
{
      printf("1\n");
      printf("2\n");
      printf("3\n");
      printf("4\n");
      printf("5\n");
      getch();
      return 0;


}
```

Program di atas telah memenuhi yang diinginkan, tetapi jika bilangan yang akan ditampilkan misalkan dari 1 sampai 1000, maka sangatlah merepotkan jika harus menulis angka 1 sampai dengan 1000 secara manual. Oleh karena itu, di semua bahasa pemrograman terdapat suatu mekanisme yang bernama loop (perulangan).

Ada beberapa jenis perulangan yang dapat dilakukan oleh bahasa pemrograman C, yaitu :

- For
- While
- Do While
- Label

Perulangan Dengan Perintah *for*

Perulangan for mempunyai bentuk umum seperti berikut :

```
for(inisialisasi counter; kondisi perulangan; statement)
{
     statement;
}
```

Contoh berikut akan menampilkan angka 1 sampai 100, kemudian menampilkan angka 10 turun sampai 0 dengan perubahan nilainya adalah setengah (0.5).

```
#include <stdio.h>
#include <conio.h>
main()
{
     int i;
     int f;
     for (i=1;i<=1000;i++)
         printf("%i\n",i);
     for (f=10;f>=0;f-=0.5)
         printf("%6.2f\n",f);
     getch();
     return 0;


}
```

Dalam perulangan yang menggunakan for, perulangan dilakukan hanya jika kondisi perulangannya mempunyai nilai true (tidak 0).

Perulangan Dengan Perintah *while*

Bentuk umum dari while adalah seperti berikut :

```
while (kondisi)
{
    perintah;
    perintah;
}
```

Cara kerja dari perulangan while mirip dengan perulangan for. Tetapi dalam perulangan while ini, tidak ada jaminan bahwa program akan masuk ke dalam perulangan. Ini dikarenakan pemeriksaan kondisinya dilakukan di awal perulangan.

```
#include <stdio.h>
#include <conio.h>
main()
{
     int i;
     float f;
     i=1;
     while (i<=1000)
     {
         printf("%i\n",i);
         i++;
     }
     f=10;
     while (f>=0)
     {
         printf("%6.2f\n",f);
         f=f-0.5;
     }
     getch();
     return 0;


}
```

Perulangan Dengan Perintah *do while*

Bentuk umum dari do while adalah seperti berikut :

```
do
{
    perintah;
    perintah;
} while (kondisi);
```

Cara kerja dari perulangan do while mirip dengan perulangan while. Tetapi dalam perulangan do while ini, pengecekan kondisi dilakukan di akhir loop. Sehingga program **pasti** dapat masuk ke perulangan ini minimal 1 kali.

```
#include <stdio.h>
#include <conio.h>
main()
{
     int i;
     float f;
     i=1;
     do
     {
         printf("%i\n",i);
         i++;
     } while (i<=1000);
     f=10;
     do
     {
         printf("%6.2f\n",f);
         f=f-0.5;
     } while (f>=0);
     getch();
     return 0;


}
```

Perulangan Dengan Menggunakan label

Perulangan dengan menggunakan teknik label, merupakan teknik perulangan yang paling awal dikenal, biasanya ada dalam pemrograman berbahasa assembly. Tetapi perulangan seperti ini **tidak dianjurkan untuk dipakai** karena bisa membuat struktur program menjadi acak-acakan.

Untuk lebih jelas, perhatikan contoh program di bawah ini.

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

   int i;
   i=0;
   awal:
        i=i+1;
        printf("%i\n",i);
        if(i<10)
            goto awal;
        else
            goto akhir;
   printf("Perintah ini tak akan dieksekusi\n");
   printf("Perintah ini juga tak akan dieksekusi\n");
   akhir:
   getch();
   return 0;
}
```

Program di atas akan menampilkan angka 1 sampai 10. Perhatikan perintah di bawah ini :

```
printf("Perintah ini tak akan dieksekusi\n");
printf("Perintah ini juga tak akan dieksekusi\n");
```

Perintah tersebut tidak akan pernah dieksekusi, karena ketika program telah mencapai nilai 10 maka akan melewatkan perintah tersebut dan langsung loncat (goto) ke bagian akhir yang ditandai dengan perintah **akhir:**.

Tugas :

1. Buat program untuk menampilkan angka dari nilai awal k dengan keliapatan m dan jumlah angka sebanyak n. Contoh masukan dan keluaran:

   k = 10

   m = 3

   n = 4

   10 13 16 19

2. Buat program untuk menampilkan deret fibonanci yang sesuai dengan banyaknya angka yang ingin ditampilkan. Kemudian lakukan validasi untuk

angka yang dimasukkan. Angka yang dimasukan harus antara 2 sampai 10. Bila angka yang dimasukan adalah 0, maka program selesai.

Contoh masukan dan keluaran :

Deret Fibonanci

Masukan banyaknya angka yang diinginkan [2..10] : 8
Deret Fibonanci : 1, 1, 2, 3, 5, 8, 13, 21

Diktat Mata Kuliah Pemrograman I

BAB VII
FUNGSI (FUNCTION)

IF

Pendahuluan

Fungsi merupakan blok dari kode program yang dirancang untuk melaksanakan tugas khusus. Fungsi banyak dilibatkan dalam pembuatan suatu program, dengan tujuan :

- Program menjadi lebih tersetuktur, sehingga mudah dipahami dan mudah dikembangkan
- Dapat mengurangi pengulangan kode.

Bentuk umum suatu fungsi adalah sebagai berikut :

```
Tipedata namafungsi(daftarparameter)
{
    /*Badan Fungsi*/
    return nilaireturn; /* untuk tipe data bukan void */
}
```

Fungsi Bertipe *void*

Fungsi bertipe void, kalau dalam program pascal atau delphi disebut sebagai procedure. Fungsi ini tidak mempunyai nilai kembalian, jadi fungsi bertipe ini hanya merupakan sekumpulan kode program yang bekerja sesuai dengan parameter yang diberikan.

Contoh fungsi bertipe void :

```
void TampilNama()
{
     textcolor(RED);
     cprintf("Nama Saya : Tulis Nama Ada\n\r");
     cprintf("Alamat    : Jl. Can Di Aspal No. 70\n\r");
     cprintf("Telepon   : 022-2513709\n\r");
}

main()
{
     TampilNama();
     TampilNama();
     TampilNama();
}
```

Dalam program di atas, ada sebuah fungsi yang bernama TampilNama(), yang berguna untuk menampilkan data Nama, Alamat, dan Telepon. Dalam program utama (fungsi main()), cara pemanggilan fungsi tersebut adalah dengan menulis nama fungsinya (dalam hal ini TampilNama()). Jadi program di atas akan menampilkan isi fungsi TampilNama() sebanyak 3 kali.

Fungsi di atas merupakan fungsi yang dipanggil tanpa memakai parameter. Untuk melihat contoh fungsi berparameter, perhatikan program di bawah ini.

```
void Kotak(int X1,int Y1, int X2,int Y2,int Bingkai,int Latar)
{
     int i;
     textcolor(Bingkai);
     textbackground(Latar);
     gotoxy(X1,Y1);cprintf("┌"); /* alt+218 */
     gotoxy(X1,Y2);cprintf("%c",192);
     gotoxy(X2,Y1);cprintf("%c",191);
     gotoxy(X2,Y2);cprintf("%c",217);
     for (i=X1+1;i<=X2-1;i++)
     {
          gotoxy(i,Y1);cprintf("%c",196);
          gotoxy(i,Y2);cprintf("%c",196);
     }
     for(i=Y1+1;i<=Y2-1;i++)
     {
          gotoxy(X1,i);cprintf("%c",179);
          gotoxy(X2,i);cprintf("%c",179);
     }
}
main()
{
     Kotak(1,1,80,24,WHITE,BLUE);// Memanggil Procedur Kotak
     Kotak(2,2,15,23,WHITE,RED);
     getch();
     return 0;
}
```

Void Kotak merupakan sebuah fungsi yang akan membuat suatu kotak di layar sesuai dengan koordinat yang diberikan di bagian parameter. Koordinat tersebut adalah koordinat kiri atas (X1,Y1), dan koordinat titik kanan bawah (X2,Y2). Selain itu fungsi ini membutuhkan parameter Bingkai yang berguna untuk menentukan warna bingkai kotak, dan juga parameter Latar yang berguna untuk menentukan warna latar belakang kotak yang dibuat.

Pemanggilan Kotak(1,1,80,24,WHITE,BLUE) berguna untuk membuat kotak dengan posisi kiri atas pada koordinat (1,1) dan posisi kanan bawah pada koordinat (80,24) dengan warna bingkai kotak berwarna putih dengan latar belakang kotak berwarna biru.

Fungsi bertipe data

Dalam dunia matematika, kita mengenal fungsi. Contoh : $F(X)=X^2+3X+5$, yang berarti kita mempunyai sebuah fungsi bernama F yang membutuhkan parameter X sebagai data yang akan dihitung dengan persamaan X^2+3X+5 sehingga kalau kita menulis F(5), maka nilai dari fungsi tersebut adalah 5^2 + 3.5+ 5 = 45.

Bentuk umum dari Function ini dalam bahasa Pascal adalah :

```
typedata NamaFungsi(daftar parameter)
{
 Perintah;
 Perintah;
 Return NilaiHasilUntukFungsi;
}
```

Contoh 1 : Fungsi matematik untuk menghitung persamaan $F(X) = X^2+3X+5$;

```
float F(float X)
{
     return X*X+3*X+5;//Fungsi diisi hasil dari perhitungan X²+3*X+5
}
```

Contoh 2 : Fungsi untuk mencari Faktorial dari suatu nilai

```
float Faktorial(int N)
{
   int I;
   float Hasil;
   Hasil:=1;
   for(I=2;I<=N;I++)
         Hasil=Hasil * I;
   return Hasil;
}
```

Contoh 3: Fungsi untuk mencari Kombinasi dengan rumus :

Kombinasi (X,Y) = $\frac{Y!}{X!(Y - X)!}$

```
Float  Kombinasi(int X, int Y)
{
        return Faktorial(Y) / (Faktorial(X)*Faktorial(Y-X));
}
```

Diktat Mata Kuliah Pemrograman I

BAB VIII
ARRAY (LARIK)

IF

Apakah Array?

Contoh Kasus : Suatu universitas ingin mendata nilai mahasiswa di sutau kelas dengan banyak mahasiswa 10 orang. Dari semua nilai yang telah dimasukan tersebut ingin ditampilkan kembali dan dicari nilai rata-ratanya.

Untuk membuat program dengan ketentuan seperti diatas, ada beberapa cara untuk memecahkannya :

Program 1 : Tanpa menggunakan array

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n1,n2,n3,n4,n5,n6,n7,n8,n9,n10;
    float total, ratarata;
    // Pembacaan semua nilai dari keyboard
    printf("Pemasukan data nilai mahasiswa : \n");
    printf("Nilai mahasiswa Ke-1 : ");scanf("%d",&n1);
    printf("Nilai mahasiswa Ke-2 : ");scanf("%d",&n2);
    /* diulang dari nilai ke-3 sampai terakhir */
    printf("Nilai mahasiswa Ke-10: ");scanf("%d",&n10;
    // perhitungan total dan rata-rata
    total=n1+n2+n3+n4+n5+n6+n7+n8+n9+n10;
    ratarata=total/10;
    // Menampilkan data nilai yang telah dimasukan
    printf("Nilai mahasiswa Ke-1 : %3d\n",n1);
    printf("Nilai mahasiswa Ke-2: %3d\n",n2);
    /* diulang dari nilai ke-3 sampai terakhir */
    printf("Nilai mahasiswa Ke-10 : %3d\n",n10);
    // Menampilkan nilai rata-rata
    printf("Rata-rata kelas : %6.2f\n",ratarata);
    getch();
}
```

Dengan menggunakan cara diatas, sebenarnya programnya telah mencukupi, tetapi kalau nilai yang akan diolah menjadi lebih banyak, maka pendeklarasian variabel n harus dilakukan sebanyak yang diperlukan. Jadi kalau data yang akan diolah sebanyak 100 buah, maka pendeklarasian dan pembacaan datanya pun dilakukan sebanyak 100 kali. Dan perhitungannya juga. Rumus perhitungan total pun menjadi berubah. Pemrograman di atas sebenarnya sederhana tetapi bisa sangat merepotkan. Oleh karena

Array 1 dimensi

Solusi kedua dari kasus diatas adalah dengan menggunakan array. Array adalah suatu variabel yang dapat menampung lebih dari satu data dengan tipe data yang sama dan dibedakan berdasarkan nomor indexnya. Dalam bahasa C, array selalu dimulai dari index ke-0 (nol).

Contoh deklarasi array :

```
int N[10];
```

Deklarasi diatas berarti pendeklarasian variabel array bernama N yang mempunyai elemen sebanyak 10 buah dengan index dimulai dengan nomor 0 sampai 9. Dalam memori deklarasi tersebut dapat digambarkan seperti berikut :

N[0]	N[1]	N[2]	N[3]	N[4]	N5]	N[6]	N[7]	N[8]	N[9]

Untuk memasukan suaut elemen data dalam array, perintah yang dilakukan ditulis seperti pembacaan data variabel biasa hanya perbedaannya harus ditulis untuk index ke berapa.

Contoh untuk pengisian data ke elemen array :

```
scanf("%d", &N[2]);
```

Perintah diatas berarti pembacaan data dari keyboard untuk data bertipe integer (%d) dan dimasukan ke variabel array index ke-2 (urutan ke-3).

Contoh-contoh lain pengisian ke suatu elemen array :

```
I=5; // variabel I diisi dengan nilai 5
N[I] = 7; // data ke-I dari variabel N diisi dengan nilai 7
scanf("%d",&N[N[I]]);
// pembacaan data untuk variabel N pada index ke-N[I] (7)
```

Karena nomor elmeen dari array bisa diisi dengan variabel, berarti kita bisa melakukan perulangan (loop) untuk melakukan pembacaan data dari elemen pertama sampai elemen terakhir.

Untuk lebih jelas, lihat program pada halaman berikutnya.

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

```
void main()
{
    int Nilai[10];
    int index;
    float total,ratarata;

    // Pembacaan data dari keyboard
    printf("Pembacaan data nilai \n");
    for (index=0;index<10;index++)
    {
        printf("Nilai mahasiswa ke-%d = ",index+1);
        scanf("%d",&Nilai[index]);
    }

    // Perhitungan total dan rata-rata
    total=0;
    for (index=0;index<10;index++)
         total=total+Nilai[index];// atau total+=Nilai[index];
    ratarata=total/10;

    // Menampilkan data yang telah dimasukan dan rata-rata.
    for (index=0;index<10;index++)
        printf("Nilai mahasiswa ke-%d = %3d\n",index+1,Nilai[index]);
    printf("Rata-rata = %6.2f\n",ratarata);

    getch();

}
```

Array 2 Dimensi

Array 2 dimensi biasanya digunakan untuk menyimpan data dalam bentuk matrik. Index Array 2 dimensi terdiri dari index baris dan kolom.

Pendeklarasian array 2 dimensi adalah :

```
Tipedata  namaarray[b][k];
```

Dimana : b adalahbanyak baris dan k adalah banyak kolom.

Contoh

```
int matrik[5][5];
```

Perintah di atas akan membuat sebuah array 2 dimensi yang kalau digambarkan adalah sebagai berikut :

index	0	1	2	3	4
0	0,0	0,1	0,2	0,3	0,4
1	1,0	1,1	1,2	1,3	1,4
2	2,0	2,1	2,2	2,3	2,4
3	3,0	3,1	3,2	3,3	3,4
4	4,0	4,1	4,2	4,3	3,3

Cara pengaksesan elemen array 2 dimensi dapat dilihat pada contoh di bawah ini :

```
mat[0][0]=7;
printf("Masukan data : ");scanf("%d",&mat[2][1]);
printf("Data yang dimasukan : %d\n",mat[2][1]);
```

Keterangan :

- Baris pertama adalah mengisikan nilai 7 ke array mat pada baris 0 kolom 0.
- Baris kedua adalah perintah untuk membaca data elemen matrik pada baris 2 kolom ke 1.
- Baris ketiga adalah perintah untuk menampilkan data elemen matrik/array pada baris 2 dan kolom ke-1.

Pembacaan elemen-elemen array 2 dimensi melibatkan 2 perulangan. 1 perulangan baris dan 1 perulangan kolom. Untuk lebih jelas perhatikan program di bawah ini.

Contoh Program Array 2 Dimensi :

```
#include <stdio.h>
#include <conio.h>
#define maks 3
main()
{
    int mat[maks][maks];
    int b,k;
    printf("Pengisian Array : \n");
    for (b=0;b<maks;b++)
    {
        for (k=0;k<maks;k++)
        {
            printf("Matrik [%d,%d] : ",b,k);
            scanf("%d",&mat[b][k]);
        }
    }
    printf("Matrik yang telah dimasukan :\n")
    for (b=0;b<maks;b++)
    {
        for (k=0;k<maks;k++)
        {
            printf("%6d",mat[b][k]);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

Contoh Program Operasi pertambahan 2 matrik.

```
#include <stdio.h>
#include <conio.h>
#define maks 3
main()
{
    int mat1[maks][maks], mat2[maks][maks], mathasil[maks][maks];
    int b,k;
    printf("Pengisian Matrik 1 : \n");
    for (b=0;b<maks;b++)
    {
        for (k=0;k<maks;k++)
        {
            printf("Matrik [%d,%d] : ",b,k);
            scanf("%d",&mat1[b][k]);
        }
    }




    printf("Pengisian Matrik 2 : \n");
    for (b=0;b<maks;b++)
    {
```

```
        for (k=0;k<maks;k++)
        {
            printf("Matrik [%d,%d] : ",b,k);
            scanf("%d",&mat2[b][k]);
        }
    }
    // awal operasi pertambahan matrik

    for (b=0;b<maks;b++)
        for (k=0;k<maks;k++)
            mathasil[b][k]=mat1[b][k]+mat2[b][k];

    // akhir operasi perhitungan

    printf("Matrik 1 :\n")
    for (b=0;b<maks;b++)
    {
        for (k=0;k<maks;k++)
        {
            printf("%6d",mat1[b][k]);
        }
        printf("\n");
    }
    printf("Matrik 2 :\n")
    for (b=0;b<maks;b++)
    {
        for (k=0;k<maks;k++)
        {
            printf("%6d",mat2[b][k]);
        }
        printf("\n");
    }
    printf("Matrik Hasil :\n")
    for (b=0;b<maks;b++)
    {
        for (k=0;k<maks;k++)
        {
            printf("%6d",mathasil[b][k]);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

Kasus Array 2 Dimensi

1. Buatlah suatu program untuk menghitung jumlah anggota yang ada dalam suatu matrik.

   Contoh :

   2
   3
   4

   5
   6
   2

   1
   3
   4

   Total Elemen matrik adalah 30

   Catatan : Anggota elemen dimasukan dari keyboard

2. Buatlah suatu program untuk menampilkan total elemen per baris dan per kolom.
   Contoh :
   Input :

   2
   3
   4

   5
   6
   2

   1
   3
   4

   Output :

2
3
4
9

5
6
2
13

1
3
4
8

8
12
10

3. Buatlah suatu program untuk mengecek apakah suatu matrik simetris atau tidak.
   Contoh 1 :
   Input :
   2
   3
   4

   5
   6
   2

   1
   3
   4

   Matrik tidak simetris

   Contoh 2 :
   Input :

2
5
6

5
1
2

6
2
4

Matrik simetris

Catatan : Suatu matrik akan disebut simetris jika $M_{ij} = M_{ji}$, jadi satu elemen saja tidak terpenuhi berarti matrik tersebut tidak simetris.

4. Buatlah program untuk melakukan perhitungan perkalian matrik. Matrik pertama dan Matrik kedua dimasukan dari keyboard, kemudian lakukan operasi perkalian dan kemudian tampilkan hasilnya.

Diktat Mata Kuliah Pemrograman I

BAB X
STRUCT (RECORD)

IF

Apakah Struct ?

Turbo C tidak selalu menyediakan tipe data yang sesuai dengan tipe data yang diinginkan. Contoh kasus yaitu ketika kita ingin membuat suatu program mengolah data mahasiswa dimana data mahasiswa terdiri dari NIM, Nama, NilaiUTS, NilaiUAS, NilaiQuiz, NilaiAkhir dan Index Prestasinya. Turbo C tidak menyediakan tipe data untuk data tersebut. Oleh karena itu maka kita harus membuat suatu tipe data baru yang cocok dengan keperluan kita. Caranya adalah dengan menggunakan perintah struct.

Deklarasi tipe data baru (struct) untuk data mahasiswa dapat dilihat sebagai berikut :

```
struct TMhs
{
      char NIM[11];
      char Nama[21];
      int NilaiUTS,NilaiUAS,NilaiQuiz;
      float NilaiAkhir;
      char index;
};
```

Deklarasi diatas berarti kita telah membuat suatu tipe data yang bernama TMhs dimana setiap data bertipe TMhs mempunyai field NIM, Nama, NilaiUTS, NilaiUAS, NilaiQuiz, NilaiAkhir dan index.

Untuk mendeklarasikan sebuah variable yang bertipe TMhs caranya adalah seperti berikut :

```
TMhs Mhs1,Mhs2;
```

Deklarasi tersebut berarti bahwa kita membuat suatu variable bernama Mhs1 dan Mhs2 dimana tiap variable tersebut mempunyai field sesuai dengan TMhs.

Kalau digambarkan, maka struktur Mhs1 dan Mhs2 dapat dilihat seperti berikut :

MHS1							MHS2						
NIM	NAMA	NILAI UTS	NILAI UAS	NILAI QUIZ	NILAI AKHIR	INDEX	NIM	NAMA	NILAI UTS	NILAI UAS	NILAI QUIZ	NILAI AKHIR	INDEX

Untuk mengisi Nilai UTS dari Mhs1 maka perintahnya adalah :

```
Mhs1.NilaiUTS=50;
scanf("%i",&Mhs1.NilaiUTS);//membaca data dari keyboard
```

Contoh program yang menggunakan variable yang bertipe bentukan dapat dilihat di halaman berikutnya.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct TMhs
{
      char NIM[11];
      char Nama[21];
      int NilaiUTS,NilaiUAS,NilaiQuiz;
      float NilaiAkhir;
      char index;
};
main()
{
     TMhs mhs1,mhs2;
     printf("Pengisian Data");
     printf("NIM        : ");gets(mhs1.NIM);
     printf("NAMA       : ");gets(mhs1.Nama);
     printf("Nilai QUIZ : ");scanf("%d",&mhs1.NilaiQuiz);
     printf("Nilai UTS  : ");scanf("%d",&mhs1.NilaiUTS);
     printf("Nilai UTAS : ");scanf("%d",&mhs1.NilaiUAS);
     mhs1.NilaiAkhir=0.2*mhs1.NilaiQuiz+0.3*mhs1.NilaiUTS+0.5*mhs1.N
     ilaiUAS;
     if(mhs1.NilaiAkhir>=80) mhs1.index='A';else
     if(mhs1.NilaiAkhir>=60) mhs1.index='B';else
     if(mhs1.NilaiAkhir>=40) mhs1.index='C';else
     if(mhs1.NilaiAkhir>=20) mhs1.index='D';else
     if(mhs1.NilaiAkhir>=00) mhs1.index='E';

     mhs2=mhs1; // mengisikan semua data di mhs1 ke mhs2

     printf("Data yang telah dimasukan :");
     printf("NIM        : %s\n",mhs2.NIM);
     printf("NAMA       : %s\n",mhs2.Nama);
     printf("Nilai QUIZ : %i\n",mhs2.NilaiQuiz);
     printf("Nilai UTS  : %d\n",mhs2.NilaiUTS);
     printf("Nilai UTAS : %d\n",mhs2.NilaiUAS);
     printf("Nilai Akhir: %.2f\n",mhs2.NilaiAkhir);
     printf("Index      : %c\n",mhs2.index);

     getch();


}
```

Array Struct

Setiap tipe data dapat dibuat dalam bentuk array. Begitu juga dengan tipe data yang dibuat dengan perintah struct.

Contoh program di bawah ini dapat menjelaskan cara penggunaan array yang bertipe data buatan.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define maks 3
struct TMhs
{
      char NIM[9];
      char Nama[21];
      int NilaiUTS,NilaiUAS,NilaiQuis;
      float NilaiAkhir;
      char index;
};
main()
{
   TMhs mhs[maks]; // array struct
   int i;
   for(i=0;i<maks;i++)
   {
        printf("Pengisian Data Mahasiswa Ke-%i\n",i+1);
        printf("NIM        : ");fflush(stdin);gets(mhs[i].NIM);
        printf("NAMA       : ");fflush(stdin);gets(mhs[i].Nama);
        printf("Nilai QUIZ : ");scanf("%d",&mhs[i].NilaiQuis);
        printf("Nilai UTS  : ");scanf("%d",&mhs[i].NilaiUTS);
        printf("Nilai UTAS : ");scanf("%d",&mhs[i].NilaiUAS);
        mhs[i].NilaiAkhir=0.2*mhs[i].NilaiQuis+0.3*mhs[i].NilaiUTS+0.5*mhs[i].NilaiUAS;
        if(mhs[i].NilaiAkhir>=80) mhs[i].index='A';else
        if(mhs[i].NilaiAkhir>=60) mhs[i].index='B';else
        if(mhs[i].NilaiAkhir>=40) mhs[i].index='C';else
        if(mhs[i].NilaiAkhir>=20) mhs[i].index='D';else
        if(mhs[i].NilaiAkhir>=0)  mhs[i].index='E';
   };
   clrscr();
   printf("Data yang telah dimasukan adalah : \n");
   printf("-------------------------------------------------------------------------\n");
   printf("|    NIM     |       NAMA          | QUIS | UTS | UAS |  N A  | INDEX |\n");
   printf("-------------------------------------------------------------------------\n");
   for(i=0;i<maks;i++)
   {
      printf("| %-8s | %-20s |  %3i | %3i | %3i | %6.2f |   %c   |\n",
              mhs[i].NIM,mhs[i].Nama,mhs[i].NilaiQuis,mhs[i].NilaiUTS,
              mhs[i].NilaiUAS,mhs[i].NilaiAkhir,mhs[i].index);
   }
   printf("-------------------------------------------------------------------------\n");
   getch();
   return 0;
}
```

Kalau program tersebut dijalankan maka jalannya program dalam dilihat di bawah ini :

```
Pengisian Data Mahasiswa Ke-1
NIM        : 10197025
NAMA       : Andri Heryandi
Nilai QUIZ : 70
Nilai UTS  : 80
Nilai UTAS : 90
Pengisian Data Mahasiswa Ke-2
NIM        : 10197024
NAMA       : Hery Dwi Yulianto
Nilai QUIZ : 12
Nilai UTS  : 56
Nilai UTAS : 90
Pengisian Data Mahasiswa Ke-3
NIM        : 10197001
NAMA       : Irmayanti
Nilai QUIZ : 80
Nilai UTS  : 90
Nilai UTAS : 100

Data yang telah dimasukan adalah :
-------------------------------------------------------------------------
|   NIM    |        NAMA          | QUIS | UTS | UAS |   N A  | INDEX |
-------------------------------------------------------------------------
| 10197025 | Andri Heryandi       |   70 |  80 |  90 |  83.00 |   A   |
| 10197024 | Hery Dwi Yulianto    |   12 |  56 |  90 |  64.20 |   B   |
| 10197001 | Irmayanti            |   80 |  90 | 100 |  93.00 |   A   |
-------------------------------------------------------------------------
```

Diktat Mata Kuliah Pemrograman I

BAB XI
POINTER

IF

Pointer ?

Pointer adalah sebuah variabel yang isi datanya adalah alamat memori atau variabel lain. Sehingga pointer dapat juga disebut sebagai variabel alamat (address variable).

Deklarasi pointer

Untuk mendeklarasikan sebuah pointer, perintah dasarnya adalah :

```
Typedata *namavariabel;
```

Untuk lebih jelasnya adalah :

```
int *pint;
float *pfloat;
Tmhs *pmhs;
```

Pada contoh ke-1 kita mendeklarasikan sebuah pointer bernama pint yang menunjuk ke suatu alamat di memori yang menampung sebuah data bertipe integer. Contoh ke-2 mendeklarasikan sebuah variabel pointer bernama pfloat yang menunjuk ke suatu alamat yang menampung sebuah data bertipe float, begitu juga dengan contoh ke-3 yang mendeklarasikan suatu variabel bernama pmhs yang menunjuk ke suatu data yang bertipe Tmhs.

Pengisian data ke variabel pointer

Pengisian data ke variabel pointer bisa berarti pengisian alamat memori ke variabel tersebut atau pengisian data yang ditunjuk oleh pointer.

Untuk lebih jelasnya, perhatikan program dibawah ini :

```
01: #include <stdio.h>
02: #include <conio.h>
03: #include <string.h>
04:
05: main()
06: {
07:    char  c,*pc;
08:    int   i,*pi;
09:    float f,*pf;
10:    clrscr();
11:    c='A';i=7;f=6.25;
12:    printf("c : alamat=0x%p, isi=%c\n", &c, c);
13:    printf("x : alamat=0x%p, isi=%d\n", &i, i);
14:    printf("y : alamat=0x%p, isi=%5.2f\n", &f, f);
15:    pc=&c;
16:    pi=&i;
17:    pf=&f;
18:    printf("pc: alamat=0x%p, isi=%c\n", pc, *pc);
19:    printf("pi: alamat=0x%p, isi=%d\n", pi, *pi);
20:    printf("pf: alamat=0x%p, isi=%5.2f\n", pf, *pf);
21:    *pc='B';
22:    *pi=125;
23:    *pf=512.56;
24:    printf("c : isi=%c\n", c);
25:    printf("x : isi=%d\n", i);
26:    printf("y : isi=%5.2f\n", f);
27:    getch();
28:    return 0;
29: }
```

Jika dieksekusi, maka akan menghasilkan sebuah tampilan sebagai berikut :

```
c : alamat=0xFFF5, isi=A
x : alamat=0xFFF2, isi=7
y : alamat=0xFFEE, isi= 6.25
pc: alamat=0xFFF5, isi=A
pi: alamat=0xFFF2, isi=7
pf: alamat=0xFFEE, isi= 6.25
c : isi=B
x : isi=125
y : isi=512.56
```

Keterangan Program :

- Pada baris 7 : Pendeklarasian sebuah variabel **c** dengan tipe char, dan sebuah pointer **pc** yang merupakan pointer char.
- Pada baris 8 : Pendeklarasian sebuah variabel **i** dengan tipe int, dan sebuah pointer **pi** yang merupakan pointer int.
- Pada baris 9 : Pendeklarasian sebuah variabel **f** dengan tipe int, dan sebuah pointer **pf** yang merupakan pointer float.
- Pada baris 11 : Pengisian variabel **c** dengan karakter 'A', variabel **i** dengan nilai 7, dan variabel **f** dengan nilai 6.25.
- Pada baris 12 : Menampilkan alamat variabel **c** dan isinya.
- Pada baris 13 : Menampilkan alamat variabel **i** dan isinya.
- Pada baris 14 : Menampilkan alamat variabel **f** dan isinya.

- Pada baris 15 : Variabel pointer **pc** diisi dengan alamat variabel **c** sehingga kedua variabel mengacu ke tempat yang sama sehingga ketika isi **pc** diubah maka berarti merubah isi variabel **c** dan begitu juga sebaliknya.
- Pada baris 15 : Variabel pointer **pi** diisi dengan alamat variabel **i** sehingga kedua variabel mengacu ke tempat yang sama sehingga ketika isi **pi** diubah maka berarti merubah isi variabel **i** dan begitu juga sebaliknya.
- Pada baris 15 : Variabel pointer **pf** diisi dengan alamat variabel **f** sehingga kedua variabel mengacu ke tempat yang sama sehingga ketika isi **pf** diubah maka berarti merubah isi variabel **f** dan begitu juga sebaliknya.
- Baris 16, 17 dan 18 : Menampilkan data yang ditunjuk oleh pointer **pc, pi,** dan **pf**, yang hasilnya pasti sama dengan hasil baris 12, 13, dan 14.
- Baris 19 : Mengisi data ke alamat yang ditunjuk oleh **pc** dengan nilai B, yang berarti juga mengganti nilai variabel **c**.
- Baris 20 : Mengisi data ke alamat yang ditunjuk oleh **pi** dengan nilai 125, yang berate juga mengganti nilai variabel **i**.
- Baris 21 : Mengisi data ke alamat yang ditunjuk oleh **pf** dengan nilai 512.56, yang berarti juga mengganti nilai variabel **f**.
- Baris 22,23 dan 24 : Menampilkan isi nilai variabel **c, i** dan **f**, yang telah diubah oleh variabel pointernya.

Contoh Aplikasi Pointer

Salah satu penggunaan pointer adalah untuk membuat suatu array yang dinamis (banyaknya data yang bisa ditampung sesuai keperluan).

Sebagaimana kita ketahui jika kita membuat suatu program yang dapat menampung data nilai sebanyak 5 buah maka kita akan membuat suatu variabel array yang bertipe int dengan perintah **int data[5].** Dengan cara begitu maka program hanya akan berjalan dengan baik jika data yang diinputkan banyaknya di kurang atau sama dengan 5 buah. Apa yang terjadi ketika data yang akan diinputkan ternyata 10 buah, maka langkah yang dilakukan adalah harus mengubah programnya dan mengganti **int data[5]** menjadi **int data[10].**

Cara lain untuk membuat program tersebut adalah dengan menggunakan suatu variabel array yang dinamis dimana pemesanan tempat yang diperlukan untuk menyimpan data tidak dideklarasikan dalam program tapi dilakukan secara runtime (ketika program berjalan).

```
01: #include <stdio.h>
02: #include <conio.h>
03: #include <stdlib.h>
04: main()
05: {
06:    int *data;
07:    int i,banyakdata;
08:    printf("Banyak data yang akan diinputkan : ");scanf("%i",&banyakdata);
09:    data=(int *)malloc(sizeof(int)*banyakdata);
10:    for(i=0;i<banyakdata;i++)
11:    {
12:       printf("Pemasukan data ke-%i :",i+1);scanf("%i",(data+i));
13:    }
14:    printf("Data yang telah diinputkan adalah : \n");
15:    for(i=0;i<banyakdata;i++)
16:       printf("Data ke-%i : %i\n",i+1,*(data+i));
```

```
17:     getch();
18:     return 0;
19: }
```

Tampilan yang dihasilkan oleh program diatas adalah sebagai berikut :

```
Banyak data yang akan diinputkan : 5
Pemasukan data ke-1 :12
Pemasukan data ke-2 :3
Pemasukan data ke-3 :4
Pemasukan data ke-4 :5
Pemasukan data ke-5 :67
Data yang telah diinputkan adalah :
Data ke-1 : 12
Data ke-2 : 3
Data ke-3 : 4
Data ke-4 : 5
Data ke-5 : 67
```

Keterangan program :

- Baris 6 : Pendeklarasian sebuah variabel pointer **int** yang bernama **data**.
- Baris 7 : Pendeklarasian variabel **i** sebagai counter dan **banyakdata** untuk menyimpan banyaknya data.
- Baris 8 : Pengisian data banyak data.
- Baris 9 : Pemesanan alokasi di memori untuk variabel pointer data sebesar besarnya **int** (**sizeof(int)**) dikali dengan **banyakdata**.
- Baris 10 : Perulangan untuk membaca data dari data ke-0 sampai ke **banyakdata**-1.
- Baris 12 : Membaca data dari keyboard dan dimasukan ke alamat data pada urutan ke-**i**.
- Baris 15 – 16 : Menampilkan isi data yang ditunjuk oleh pointer

Diktat Mata Kuliah Pemrograman I

BAB XII
OPERASI FILE

IF

Membuat File Text

File text adalah suatu file yang pola penyimpanannya datanya dalam bentuk karakter. Sehingga kalau suatu variabel bertipe int (2 byte) dengan isi 10000, maka akan disimpan dalam bentuk karakter 10000 (5 karakter) sehingga file yang dibuat besarnya 5 byte.

Contoh program untuk membuat file text :

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  #include <String.h>
 4  main()
 5  {
 6      FILE *f;
 7      char st[81]="";
 8      f=fopen("strings.dat","w");
 9      if (f!=NULL)
10      {
11         do{
12             printf("Masukan string : ");gets(st);
13             fprintf(f,"%s\n",st);
14         }while (strlen(st)>0);
15         fclose(f);
16         printf("File selesai di buat");
17      }
18      else
19      {
20          printf("File gagal dibuat");
21      }
22      getch();
23      return 0;
24  }
```

Keterangan program :

- Baris 6 : Perintah FILE *f berarti deklarasi variabel f yang merupakan pointer ke suatu file .
- Baris 8 : membuat link / menghubungkan variabel file f dengan file strings.dat dengan mode operasi adalah w (write).
- Baris 9 : Jika file sukses di buat (variabel f menunjuk ke suatu file/tidak NULL) maka program akan mengerjakan perintah baris 11 – 16.
- Baris 11 dan 14: Perulangan selama string yang dimasukan panjangnya lebih dari 0.
- Baris 12 : Baca sebuah string, masukan ke variabel st

- Baris 13 : Tulis string st ke file f, dan kemudian diberikan tanda pindah baris (\n).
- Baris 15 : Menutup hubungan/link ke file f.
- Baris 18 : Jika file tidak bisa dibuat, maka akan menampilkan pesan File gagal dibuat.

Jika data yang dimasukan ingin disimpan di bagian belakang file, maka mode yang digunakan adalah mode "a"

Membaca File Text

Mode yang dipakai dalam membaca data text adalah mode "r". File yang dibaca bisa berekstensi apa saja.

Perhatikan perintah berikut :

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  #include <String.h>
 4  main()
 5  {
 6      FILE *f;
 7      char st[81]="";
 8      f=fopen("strings.dat","r");
 9      if (f!=NULL)
10      {
11         clrscr();
12         while (fscanf(f,"%s",st)!=EOF) //(fgets(st,80,f)!=NULL)
13         {
14             printf("%s",st);
15         }
16         fclose(f);
17         printf("File sudah dibaca");
18      }
19      else
10      {
21          printf("File gagal dibaca");
22      }
23      getch();
24      return 0;
25  }
26
```

Keterangan program :

- Baris 8 : Buka sebuah file (strings.dat) dalam mode "r" (read).
- Baris 9 – 18 : jika variabel F tidak NULL (file sukses di buka), maka program akan mengeksekusi program baris 11 – 17.
- Baris 12 : Perulangan selama fscanf belum mencapai EOF (end of file). Fscanf akan mengambil data dari file bertipe string (%s") disimpan di st. Alternatif lain adalah menggunakan perintah fgets.
- Baris 14 : Menuliskan data yang telah dibaca.

Membuat File Biner

File biner adalah suatu file yang pola penyimpanannya adalah dengan menuliskan data secara biner. Sebagai contoh, jika data yang disimpan ke file tersebut bertipe int (2 byte) dengan isi variabel 10000, maka akan disimpan sebesar 2 byte dengan isi : 2710_h. Tipe jenis ini sangat cocok untuk penyimpanan data dalam bentuk struct. Pembacaan file biner dilakukan perblok.

Contoh program :

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  main()
 4  {
 5      FILE *f;
 6      int i;
 7      f=fopen("dataint.abc","wb");
 8      if(f!=NULL)
 9      {
10        do{
11             printf("Masukan angka : ");scanf("%d",&i);
12             if(i!=0)
13                  fwrite(&i,sizeof(int),1,f);
14        }while (i!=0);
15        fclose(f);
16        printf("File telah selesai dibuat");
17      }
18      else
19      {
20         printf("Data tidak bisa dibuat");
21      }
22      getch();
23      return 0;
24  }
```

Keterangan program :

- Baris 7 : Buka file dalam mode "wb" (write binary).
- Baris 8 – 17 : Jika file bisa dibuat, maka akan melakukan perintah-perintah di baris 10 – 16.
- Baris 10 – 14 : Perulangan selama data yang dibaca dari keyboard tidak nol.
- Baris 11 : Baca suatu integer
- Baris 12-13 : Jika data tidak nol maka disimpan dalam file f sebanyak 1 data sebesar 2 byte (sizeof(int)) yang ada di alamat &i.
- Baris 15 : Menutup file f.

Membaca File Biner

Untuk membaca file biner, mode operasi filenya adalah "rb" (read binary). Pembacaan data untuk tipe biner dilakukan perblok, sehingga kita harus tahu besar data yang akan dibaca.

Perhatikan program di bawah ini :

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  main()
 4  {
 5      FILE *f;
 6      int i;
 7      f=fopen("dataint.abc","rb");
 8      if(f!=NULL)
 9      {
10        while(fread(&i,sizeof(int),1,f)==1)
11             printf("angka : %i\n",i);
12        fclose(f);
13        printf("File telah selesai dibaca");
14      }
15      else
16      {
17         printf("Data tidak bisa dibaca");
18      }
19      getch();
20      return 0;
21  }
```

Keterangan program :

- Baris 7 : Buka file dataint.abc dalam mode rb (read binary).
- Baris 8 – 13 : Jika file bisa dibuka, maka akan mengeksekusi perintah baris 10 – 13
- Baris 10 : Lakukan pembacaan data fread, simpan data ke alamat I (&i), sebesar 2 byte (sizeof(int)), sebanyak 1 buah integer, dari file f. Pembacaan ini diulang selama fread == 1. fread menghasilkan nilai banyak data yang dibaca.
- Baris 11 : Menuliskan data hasil pembacaan (i) ke layar
- Baris 12 : Menutup file f.

Membaca File Biner Secara Acak

Ada kalanya kita ingin mengambil data dari posisi tengah file. Untuk itu, ada suatu fungsi fseek, yang gunanya untuk memindahkan kursor file ke posisi tertentu.

Posisi awal pencarian ada 3 jenis yaitu SEEK_SET (dari awal file), SEEK_CUR (dari posisi sekarang), SEEK_END (dari akhir file).

Contoh perintah :

- fseek(f,10,SEEK_SET), pindahkan pointer file ke posisi (byte) ke-10 dari awal file.
- fseek(f,5,SEEK_CUR), pindahkan pointer file 5 byte dari posisi pointer sekarang.
- fseek(f,1,SEEK_END), pindahkan pointer posisi paling akhir (1 byte sebelum EOF).

Jadi kalau kita mempunyai file biner yang digunakan untuk menyimpan data int, maka kalau kita ingin mengambil data ke-5 perintah fseeknya adalah fseek(f,5*sizeof(int),SEEK_SET).

Contoh dibawah ini akan menampilkan data dari posisi 3 dan seterusnya.

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  main()
 4  {
 5      FILE *f;
 6      int i;
 7      f=fopen("dataint.abc","rb");
 8      if(f!=NULL)
 9      {
10        fseek(f,3*sizeof(int),SEEK_SET);
11        while(fread(&i,sizeof(int),1,f)==1)
12             printf("angka : %i\n",i);
13        fclose(f);
14        printf("File telah selesai dibaca");
15      }
16      else
17      {
18         printf("Data tidak bisa dibaca");
19      }
20      getch();
21      return 0;
    }
```

Diktat Mata Kuliah Pemrograman I

BAB XIII
GRAPHIC

IF

Inisialisasi Grafik

Untuk menggambar grafik dalam Turbo C, maka langkah yang pertama kali harus dilakukan adalah dengan mengubah mode layar dari mode text menjadi mode grafik, dengan cara :

```
#include <graphics.h>
main()
{
   int graphdriver, graphmode;
   graphdriver=VGA;
   graphmode=VGAHI;
   initgraph(&graphdriver,&graphmode,"C:\\TC\\BGI");
   //disinilah ditulis perintah penggambaran
   getch();
   closegraph();
   return 0;
}
```

Untuk menutup mode grafik dan kembali ke mode text, perintah yang diperlukan adalah closegraph.

Untuk melihat driver dan mode yang bisa dipakai, carilah keterangannya dengan menggunakan fasilitas help atau tempatkan kursor di posisi VGAHI kemudian tekan tombol Ctrl+F1.

Mengambil nilai Koordinat X dan Y

Ada beberapa perintah untuk mengambil posisi koordinat x atau y, diantaranya :

- **`getmaxx()`**

Perintah ini digunakan untuk mengambil nilai maksimal dari koordinat x. Fungsi ini mempunyai nilai integer.

- **`getmaxy()`**

Perintah ini digunakan untuk mengambil nilai maksimal dari koordinat x. Fungsi ini menghasilkan nilai integer.

- **`getx()`**

Perintah ini digunakan untuk mengambil posisi x dari kursor yang sedang aktif.

- **`gety()`**

Perintah ini digunakan untuk mengambil posisi y dari kursor yang sedang aktif.

Pembuatan Garis, Kotak

- **Membuat Kotak**

Untuk membuat/menggambar kotak, perintahnya adalah :

```
rectangle(x1,y1,x2,y2);
```

Dimana x1=koordinat kiri/kanan, y1=koordinat atas/bawah, x2=koordinat kiri/kanan, y2=koordinat atas/bawah.

Contoh :

```
rectangle(0,0,getmaxx(),getmaxy());
rectangle(10,10,getmaxx()-10,getmaxy()-10);
```

Membuat Garis

Untuk membuat/menggambar garis, perintahnya adalah :

```
line(x1,y1,x2,y2);
```

Dimana x1=koordinat kiri/kanan, y1=koordinat atas/bawah, x2=koordinat kiri/kanan, y2=koordinat atas/bawah.

Contoh :

```
line(0,0,getmaxx(),getmaxy());
line(getmaxx(),0,getmaxx()/2,getmaxy()/2);
```

- **Membuat garis dengan moveto, lineto dan linerel**

Perintah lineto digunakan untuk membuat suatu garis dari posisi kursor. Perintah moveto digunakan untuk memindahkan kursor ke posisi tertentu, fungsinya seperti gotoxy dalam mode text.

Bentuk dasar untuk perintah moveto adalah

```
moveto(x,y)
```

Dimana x adalah koordinat x yang dimaksud, y adalah koordinat y yang diinginkan.

Bentuk dasar untuk perintah lineto adalah

```
lineto(x,y);
```

Dimana x adalah koordinat x yang dimaksud, y adalah koordinat y yang diinginkan.

Contoh :

```
moveto(100,100);
lineto(125,150);
lineto(75,150);
lineto(100,100);
```

- **Membuat garis dengan moverel dan linerel**

Perintah linerel digunakan untuk menggambar suatu garis dari posisi kursor aktif ke posisi yang diinginkan berdasarkan jarak dari posisi kursor aktif. Perintah moverel digunakan untuk memindahkan kursor ke posisi tertentu dari posisi kursor aktif berdasarkan jarak dari posisi kursor aktif tersebut.

Bentuk dasar untuk perintah linerel adalah :

```
linerel(dx, dy);
```

Dimana dx adalah jarak koordinat x baru dari posisi x kursor aktif, dan dy adalah jarak koordinat y baru dari posisi y kursor aktif.

Bentuk dasar untuk perintah moverel adalah :

```
moverel(dx, dy);
```

Dimana dx adalah jarak koordinat x baru dari posisi x kursor aktif, dan dy adalah jarak koordinat y baru dari posisi y kursor aktif.

Contoh :

```
moveto(100,100);
linerel(25,50);
linerel(-50,0);
linerel(25,-50);
```

Pembuatan Lingkaran, Elips, Arc (Busur)

- **Membuat Lingkaran**

Untuk membuat/menggambar lingkaran, perintahnya adalah :

```
circle(x,y,r);
```

Dimana, x adalah koordinat x dari titik tengah, y adalah koordinat y dari titik tengah dan r adalah radius lingkaran.

Contoh :

```
circle(getmaxx()/2, getmaxy()/2,100);
circle(50,50,25);
```

- **Membuat Ellipse**

Untuk membuat/menggambar elip, perintah yang dipakai adalah :

```
ellipse(x,y,d1,d2,rx,ry);
```

dimana x adalah koordinat x dari titik tengah, y adalah koordinat y dari titik tengah, d1 dan d2 merupakan derajat awal ellips, rx adalah radius ellipse di sumbu x, dan ry adalah radius ellipse di sumbu y.

Contoh :

```
ellipse(getmaxx()/2,getmaxy()/2,0,180,100,50);
ellipse(200,200,90,270,150,25);
```

- **Membuat Busur**

Membuat busur mirip dengan membuat ellipse. Perbedaannya adalah tidak diperlukan radius x atau radius y, karena radius x dan radius y dalam busur adalah sama.

```
arc(x,y,d1,d2,r);
```

Contoh :

```
arc(getmaxx()/2, getmaxy()/2, 0,270,100);
arc(150,150,0,180,75);
```

Penulisan Text

Perintah-perintah yang dipakai untuk mengatur dan menuliskan suatu text ke layar dalam mode grafik, adalah :

- **settextstyle**

Perintah ini digunakan untuk mengatur karakteristik tulisan.

Bentuk dasar perintahnya adalah :

```
settextstyle(bentukfont,arahfont,besarfont);
```

Dimana bentuk font adalah bilangan yang menunjuk ke bentuk huruf, arah font menunjukan arah penulisan huruf apakah horizontal (0), atau vertikal (1) dan besar font adalah bilangan yang menunjukan besarnya font.

Contoh :

```
settextstyle(1,1,5);
settextstyle(4,0,5);
```

- **settextjustify**

Perintah ini digunakan untuk mengatur posisi penulisan text, apakah rata kanan, tengah, rata kiri serta posisi atas, tengah dan bawah.

Bentuk dasar adalah :

```
settextjustify(horizontal, vertikal)
```

Dimana horisontal adalah bilangan yang menunjukan perataan di bagian horizontal dengan nilai yang diperbolehkan adalah 0=kiri, 1=tengah, 2=kanan, dan vertikal menunjukan perataan di bagian verikal dimana bilangan yang diperbolehkan adalah 0=bawah, 1=tengah, 2=atas.

Contoh :

```
settextjustify(1,1);
settextjustify(1,0);
```

- **outtextxy**

Perintah diatas digunakan untuk menuliskan string ke layar dalam modus grafik di posisi tertentu, sesuai justifikasinya.

Bentuk perintah dasarnya :

```
outtextxy(x,y,text);
```

Dimana x dan y adalah posisi yang diinginkan dan text adalah tulisan yang ingin ditampilkan.

Contoh :

```
outtextxy(200,20,"UNIKOM");
outtextxy(400,50,"Universitas Komputer Indonesia");
```

- **outtext**

Perintah diatas digunakan untuk menulis string dalam posisi kursor.

Bentuk dasarnya adalah :

```
outtext(text);
```

Contoh :

```
outtext("Universitas Komputer Indonesia");
outtext("Teknik Informatika");
```

Pengarsiran

Perintah-perintah yang diperlukan untuk membuat suatu bentuk yang sudah terarsir adalah :

- **Setfillstyle**

Perintah setfillstyle digunakan untuk mengatur pengarsiran.

Bentuk dasarnya adalah :

```
setfillstyle(bentuk,warna);
```

Dimana bentuk adalah bilangan antara 0 .. 12 yang menandakan bentuk arsiran dan warna adalah warna dari arsirannya.

Contoh :

```
setfillstyle(1,RED);
```

- **Bar**

Perintah bar digunakan untuk membuat suatu bentuk kotak yang telah diarsir.

Untuk membuat/menggambar kotak, perintahnya adalah :

```
bar(x1,y1,x2,y2);
```

Dimana x1=koordinat kiri/kanan, y1=koordinat atas/bawah, x2=koordinat kiri/kanan, y2=koordinat atas/bawah.

Contoh :

```
bar(0,0,getmaxx(),getmaxy());
bar(10,10,getmaxx()-10,getmaxy()-10);
```

- **bar3d**

Perintah bar3d adalah perintah untuk membuat suatu bentuk kotak 3 dimensi.

Perintah dasarnya adalah :

```
bar3d(x1,y1,x2,y2,tebal,atas);
```

Dimana x1=koordinat kiri/kanan, y1=koordinat atas/bawah, x2=koordinat kiri/kanan, y2=koordinat atas/bawah, tebal adalah ketebalan kotak, dan parameter atas diisi 0(bar tidak ditutup) atau 1(bar ditutup).

Contoh :

```
bar3d(1,1,50,300,10,1);
bar3d(60,1,110,300,5,0);
```

- **pieslice**

Perintah diatas digunakan untuk membuat suatu lingkaran yang telah diarsir.

Perintah dasarnya adalah :

```
pieslice(x,y,d1,d2,radius);
```

Dimana x dan y adalah koordinat titik tengah, d1 dan d2 adalah derajat awal dan akhir, dan radius adalah radius lingkaran yang ingin dibuat.

Contoh :

```
pieslice(getmaxx()/2, getmaxy()/2,90,360,50);
pieslice(getmaxx()/2-10, getmaxy()/2-10,0,90,50);
```

Untuk perintah lain coba pelajari di bagian help dengan cara tempatkan kursor di graphics.h kemudian tekan Ctrl+F1.