

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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 Ritchie 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

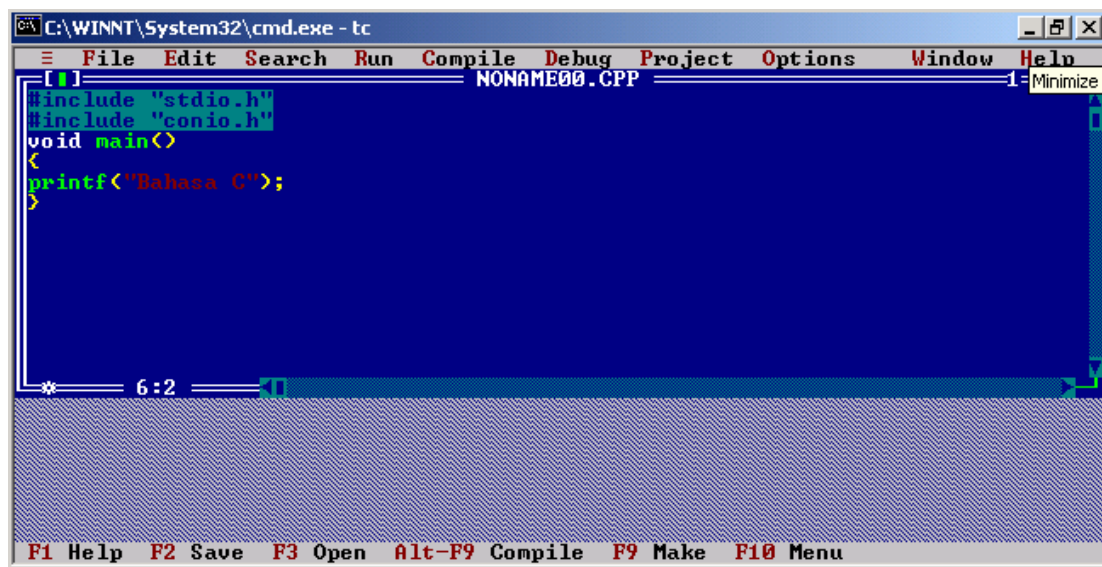
- ♦ Dukungan pustaka yang banyak.
 - ♦ C adalah bahasa yang terstruktur
 - ♦ Bahasa C termasuk bahasa tingkat menengah
- » **Kekurangan Bahasa C**
- ♦ Banyaknya Operator serta fleksibilitas penulisan program kadang-kadang membingungkan pemakai.
 - ♦ Bagi pemula pada umumnya akan kesulitan menggunakan pointer

3 Mengenal Editor Bahasa C

- » **Memulai Bahasa C**
Buka Editor Bahasa C yang ada, seperti Borland 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) **Contents**
 - (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

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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 variabel bisa diubah-ubah sesuai kebutuhan. Nama dari suatu variabel 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 variabel yang salah :
%nilai_mahasiswa, 80mahasiswa, rata-rata, ada spasi, penting!, dsb

4 Deklarasi

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

» Deklarasi Variabel

Bentuk umum pendeklarasian suatu variabel 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 : variabel "nilai" diisi dengan 80 dan variabel "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	$x \leq y$	Apakah x kurang dari sama dengan y
>	Lebih dari	$x > y$	Apakah x lebih dari y
>=	Lebih dari sama dengan	$x \geq y$	Apakah x lebih dari sama dengan y
==	Sama dengan	$x == y$	Apakah x sama dengan y
!=	Tidak 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 positif.

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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 layar 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

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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 :

	Nilai[0]	Nilai[1]	Nilai[2]	Nilai[3]	Nilai[4]
int Nilai[5];	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];
  clrscr();

  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];  
  clrscr();  
  
  // 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(" %0.2f", 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 },
      { 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

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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 memanggilya. 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 quot; // 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 fungsi 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 faktorial() */

void main()
{ int N;
  long int fak;


  printf("Berapa factorial ? "); scanf("%i", &N);
  fak = faktorial(N);              /* pemanggilan fungsi faktorial() */
  printf("%i factorial = %ld\n", N, fak);
  getch();
}

long int faktorial(int N)          /* definisi fungsi faktorial */
{ 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**

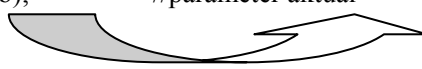
- ♦ **Parameter Formal** adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
- ♦ **Parameter Aktual** adalah variabel (parameter) yang dipakai dalam pemanggilan fungsi. Dalam contoh program pertambahan di atas parameter formal terdapat pada pendefinisian fungsi :

```
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 melewati 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 melewati 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 akan 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 dilakukan 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 register 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 faktorial ? "); scanf("%i", &N);

    printf("Faktorial dari %i = %ld\n", N, faktorial(N));
    getch();
}

long int faktorial(int N)          /* definisi fungsi faktorial */
{
    if(N==0)
        return(1);
    else
        return(N * faktorial(N - 1)); // fungsi faktorial() memanggil fungsi faktorial()
}
```

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 $g = 10 \text{ m/s}^2$ (Gunakan fungsi $\sin()$ dan $\cos()$)

Pemrograman Bahasa C dengan Turbo C

Achmad Solichin
Sh-001@plasa.com

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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 misteril(char *);

void main() {
    char string[] = "characters";
    printf("String sebelum proses adalah %s", string);
    misteril(string);
    printf("String setelah proses adalah %s", string);
}

void misteril(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 ?

- ```
#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);
}
```
- ```
#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

Lisensi Dokumen:

Copyright © 2003 IlmuKomputer.Com

Seluruh dokumen di **IlmuKomputer.Com** dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari **IlmuKomputer.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)
        putchar(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);
    }
    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 = getch();
    }
    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);  
}  
}
```