BAB II
DASAR-DASAR PEMROGRAMAN C

Tujuan :

1. Menjelaskan tentang beberapa tipe data dasar (jenis dan jangkauannya)
2. Menjelaskan tentang Variabel
3. Menjelaskan tentang konstanta
4. Menjelaskan tentang berbagai jenis operator dan pemakaiannya
5. Menjelaskan tentang instruksi I/O

2.1. Tipe Data Dasar

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

Data berdasarkan jenisnya dapat dibagi menjadi lima kelompok, yang dinamakan sebagai tipe data dasar. Kelima tipe data dasar adalah:

- Bilangan bulat (integer)
- Bilangan real presisi-tunggal
- Bilangan real presisi-ganda
- Karakter
- Tak-bertipe (*void*), keterangan lebih lanjut tentang void dijelaskan dalam Bab V.

Kata-kunci yang berkaitan dengan tipe data dasar secara berurutan di antaranya adalah *int* (*short int, long int, signed int* dan *unsigned int*), *float, double*, dan *char*.
Tabel 2-1 memberikan informasi mengenai ukuran memori yang diperlukan dan kawasan dari masing-masing tipe data dasar.

Tabel 2-1. Ukuran memori untuk tipe data

Tipe	Total bit	Kawasan	Keterangan
char	8	-128 s/d 127	karakter
int	32	-2147483648 s/d 2147483647	bilangan integer
float	32	1.7E-38 s/d 3.4E+38	bilangan real presisi-tunggal
double	64	2.2E-308 s/d 1.7E+308	bilangan real presisi-ganda

Untuk tipe data *short int, long int, signed int* dan *unsigned int*, maka ukuran memori yang diperlukan serta kawasan dari masint-masing tipe data adalah sebagai berikut :

Tabel 2-2 Ukuran memori untuk tipe data int

Tipe	Total bit	Kawasan	Keterangan
short int	16	-32768 s/d 32767	short integer
long int	32	-2147483648 s/d 2147483647	long integer
signed int	32	-2147483648 s/d 2147483647	biasa disingkat dengan int
unsigned int	32	0 s/d 4294967295	bilangan int tak bertanda

Catatan :

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

2.2 Variabel

2.2.1 Aturan Pendefinisan Variabel

Aturan penulisan pengenal untuk sebuah variabel, konstanta atau fungsi yang didefinisikan oleh pemrogram adalah sebagai berikut :

- Pengenal harus diawali dengan huruf (`A..Z`, `a..z`) atau karakter garis bawah (_).
- Selanjutnya dapat berupa huruf, digit (`0..9`) atau karakter garis bawah atau tanda dollar (`$`).
- Panjang pengenal boleh lebih dari 31 karakter, tetapi hanya 31 karakter pertama yang akan dianggap berarti.
- Pengenal tidak boleh menggunakan nama yang tergolong sebagai kata-kata cadangan (*reserved words)* seperti `int`, `if`, `while` dan sebagainya.

2.2.2 Mendeklarasikan Variabel

Variabel digunakan dalam program untuk menyimpan suatu nilai, dan nilai yang ada padanya dapat diubah-ubah selama eksekusi program berlangsung. Variabel yang akan digunakan dalam program haruslah dideklarasikan terlebih dahulu. Pengertian deklarasi di sini berarti memesan memori dan menentukan jenis data yang bisa disimpan di dalamnya.

Bentuk umum deklarasi variabel:

```
tipe daftar-variabel;
```

Pada pendeklarasian varibel, daftar-variabel dapat berupa sebuah variabel atau beberapa variabel yang dipisahkan dengan koma. Contoh:

```
int var_bulat1;
float var_pecahan1, var_pecahan2;
```

2.2.3 Memberikan Nilai ke Variabel

Untuk memberikan nilai ke variabel yang telah dideklarasikan, maka bentuk umum pernyataan yang digunakan adalah :

```
nama_variabel = nilai;
```

Contoh:

```
int var_bulat = 10;
double var_pecahan = 10.5;
```

2.2.4 Inisialisasi Variabel

Adakalanya dalam penulisan program, setelah dideklarasikan, variabel langsung diberi nilai awal. Sebagai contoh yaitu variabel **nilai** :

```
int nilai;

nilai = 10;
```

Dua pernyataan di atas sebenarnya dapat disingkat melalui pendeklarasian yang disertai penugasan nilai, sebagai berikut :

```
int nilai= 10;
```

Cara seperti ini banyak dipakai dalam program C, di samping menghemat penulisan pernyataan, juga lebih memberikan kejelasan, khususnya untuk variabel yang perlu diberi nilai awal (diinisialisasi).

2.3 Konstanta

Konstanta menyatakan nilai yang tetap. Berbeda dengan variabel, suatu konstanta tidak dideklarasikan. Namun seperti halnya variabel, konstanta juga memiliki tipe. Penulisan konstanta mempunyai aturan tersendiri, sesuai dengan tipe masing-masing.

- Konstanta karakter misalnya ditulis dengan diawali dan diakhiri dengan tanda petik tunggal, contohnya : `'A'` dan `'@'`.
- Konstanta integer ditulis dengan tanda mengandung pemisah ribuan dan tak mengandung bagian pecahan, contohnya : `-1` dan `32767`.
- Konstanta real (*float* dan *double*) bisa mengandung pecahan (dengan tanda berupa titik) dan nilainya bisa ditulis dalam bentuk eksponensial (menggunakan tanda `e`), contohnya : `27.5f` (untuk tipe *float*) atau `27.5` (untuk tipe *double*) dan `2.1e+5` (maksudnya 2,1 x 10^5).
- Konstanta string merupakan deretan karakter yang diawali dan diakhiri dengan tanda petik ganda (`"`), contohnya :`"Pemrograman Dasar C"`.

2.4 Operator

Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, seperti menjumlahkan dua buah nilai, memberikan nilai ke suatu variabel, membandingkan kesamaan dua buah nilai. Sebagian operator C tergolong sebagai operator binary, yaitu operator yang dikenakan terhadap dua buah nilai *(operand)*. Contoh :

```
a + b
```

Simbol + merupakan operator untuk melakukan operasi penjumlahan dari kedua *operand*-nya (yaitu a dan b). Karena operator penjumlahan melibatkan dua operator ini tergolong sebagai operator binary.

```
-c
```

Simbol - (minus) juga merupakan operator. Simbol ini termasuk sebagai operator unary, yaitu operator yang hanya memiliki sebuah operand (yaitu `c` pada contoh ini).

2.4.1. Operator Aritmatika

- Operator untuk operasi aritmatika yang tergolong sebagai operator binary adalah :

  `*` perkalian

  `/` pembagian

  `%` sisa pembagian

  `+` penjumlahan

  `-` pengurangan

- Adapun operator yang tergolong sebagai operator unary.

  `-` tanda minus

  `+` tanda plus

Contoh pemakaian operator aritmatika misalnya untuk memperoleh nilai diskriminan dari suatu persamaan kuadrat : **$D = b^2 - 4ac$**

```
/* File program : diskrim.c
 Menghitung diskriminan pers kuadrat  ax^2 + bx + c = 0 */

# include <stdio.h>

main()
{
    float a,b,c,d;
    a = 3.0f;
    b = 4.0f;
    c = 7.0f;

    d = b*b-4*a*c;
    printf("Diskriminan =%f\n",d);
}
```

Contoh eksekusi :

```
Diskriminan = -84.000000
```

Operator yang telah dituliskan di atas, yang perlu diberi penjelasan lebih lanjut adalah operator sisa pembagian. Beberapa contoh berikut kiranya akan memperjelas makna dari operator ini .

- Sisa pembagian bilangan 7 dengan 2 adalah 1 (`7 % 2` → `1`)
- Sisa pembagian bilangan 6 dengan 2 adalah 0 (`6 % 2` → `0`)

- Sisa pembagian bilangan 8 dengan 3 adalah 1 (`8 % 3` → `2`)

Kegunaan operator ini diantaranya bisa dipakai untuk menentukan suatu bilangan bulat termasuk ganjil atau genap, berdasarkan logika : "Jika bilangan habis dibagi dua (sisanya nol), bilangan termasuk genap. Sebaliknya, termasuk ganjil".

2.4.2. Operator Penurunan dan Penaikan

Masih berkaitan dengan operasi aritmatika, C menyediakan operator yang disebut sebagai operator penaikan dan operator penurunan, yaitu :

`++` operator penaikan
`--` operator penurunan

Operator penaikan digunakan untuk menaikkan nilai variabel sebesar satu. Penempatan operator terhadap variabel dapat dilakukan di muka atau di belakangnya, contohnya :

```
x = x+1;
y = y-1;
```

Bisa ditulis menjadi :

```
++x;
--y;
```

atau :

```
x++;
y--;
```

bergantung pada kondisi yang dibutuhkan oleh pemrogram. Di bawah ini adalah contoh yang akan menunjukkan perbedaan pemakaian dan hasil dari `++x` dengan `x++` (atau pemakaian `y--` dengan `--y`).

```
/* File program : pre_post.c
Contoh penggunaan pre & post Increment operator */

#include <stdio.h>

main()
{
    int count = 0, loop;

    loop = ++count;  /* count=count+1; loop=count; */
    printf("loop = %d, count = %d\n", loop, count);

    loop = count++;  /* loop=count; count=count+1; */
    printf("loop = %d, count = %d\n", loop, count);
}
```

Contoh eksekusi :

```
loop = 1, count = 1
loop = 1, count = 2
```

2.4.3. Prioritas Operator Aritmatika

Tabel di bawah ini memberikan penjelasan mengenai prioritas dari masing-masing operator. Operator yang mempunyai prioritas tinggi akan diutamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki prioritas lebih rendah.

Tabel 2.3 Tabel prioritas operator aritmatika dan urutan pengerjaannya

PRIORITAS	OPERATOR	URUTAN PENGERJAAN
Tertinggi	( )	dari kiri ke kanan
	! ++ -- + -	dari kanan ke kiri *)
	* / %	dari kiri ke kanan
	+ -	dari kiri ke kanan *)
Terendah	= += -= *= /= %=	dari kanan ke kiri

*) Bentuk **unary +** dan **unary –** memiliki prioritas yang lebih tinggi daripada bentuk **binary +** dan **binary –**

2.4.4. Operator Penugasan

Operator penugasan *(assignment operator)* digunakan untuk memindahkan nilai dari suatu ungkapan *(expression*) ke suatu pengenal. Operator pengerjaan yang umum digunakan dalam bahasa pemrograman, termasuk bahasa C adalah operator sama dengan (=). Contohnya :

```
fahrenheit = celcius * 1.8 + 32;
```

Maka '=' adalah operator penugasan yang akan memberikan nilai dari ungkapan : celcius `* 1.8 + 32` kepada variabel `fahrenheit.`

Bahasa C juga memungkinkan dibentuknya statemen penugasan menggunakan operator pengerjaan jamak dengan bentuk sebagai berikut :

pengenal1 = pengenal2 = … = ungkapan ;

Misalnya :

```
a = b = 15;
```

maka nilai variabel '`a` ' akan sama dengan nilai variabel '`b`' akan sama dengan `15`.

2.4.5 Operator Kombinasi (Pemendekan)

C menyediakan operator yang dimaksudkan untuk memendekkan penulisan operasi penugasan semacam

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

menjadi

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

Daftar berikut memberikan seluruh kemungkinan operator kombinasi dalam suatu pernyataan serta pernyataan padanannya.

Tabel 2.4 Seluruh kemungkinan operator kombinasi dan padanannya

`x +=  2;`	kependekan dari	`x = x + 2;`
`x -= 2;`	kependekan dari	`x = x - 2;`
`x *=  2;`	kependekan dari	`x = x * 2;`
`x /=  2;`	kependekan dari	`x = x / 2;`
`x %=  2;`	kependekan dari	`x = x % 2;`
`x <<= 2;`	kependekan dari	`x = x << 2;`
`x >>= 2;`	kependekan dari	`x = x >> 2;`
`x &=  2;`	kependekan dari	`x = x & 2;`
`x |=  2;`	kependekan dari	`x = x | 2;`
`x ^=  2;`	kependekan dari	`x = x ^ 2;`

2.5. Menampilkan Data ke Layar

Untuk keperluan menampilkan data/informasi, C menyediakan sejumlah fungsi. Beberapa di antaranya adalah berupa *printf()* dan *putchar()*.

2.5.1. Fungsi*printf()*

Fungsi *printf()* merupakan fungsi yang paling umum digunakan dalam menampilkan data. Berbagai jenis data dapat ditampilkan ke layar dengan memakai fungsi ini. Bentuk umum pernyataan *printf()* :

```
printf("string kontrol",argumen1, argumen2,...);
```

String kontrol dapat berupa keterangan yang akan ditampilkan pada layar beserta penentu format (seperti `%d`, `%f`,`%c`). Penentu format dipakai untuk memberi tahu kompiler mengenai jenis data yang akan ditampilkan. Argumen sesudah string kontrol (argumen1, argumen2,...)adalah data yang akan ditampilkan ke layar. Argumen ini dapat berupa variabel, konstanta dan bahkan ungkapan. Misal :

```
printf("%d",20);      /* argumen berupa konstanta */
printf("%d",a);       /*argumen berupa variabel */
printf("%d",a+20);    /*argumen berupa ungkapan */
```

Penentu format untuk data string atau karakter :

%c	untuk menampilkan sebuah karakter
%s	untuk menampilkan sebuah string

Untuk menampilkan data bilangan, penentu format yang dipakai berupa salah satu dari bentuk dalam Tabel 2.5.

Tabel 2.5 Penentu format pada *printf()*

`%u`	untuk menampilkan data bilangan tak bertanda *(unsigned)* dalam bentuk desimal.
`%d` `%i`	untuk menampilkan bilangan integer bertanda *(signed)* dalam bentuk desimal
`%o`	untuk menampilkan bilangan bulat tak bertanda dalam bentuk oktal.
`%x` `%X`	untuk menampilkan bilangan bulat tak bertanda dalam bentuk heksadesimal (`%x` → notasi yang dipakai : a, b, c, d, e dan f sedangkan `%X` → notasi yang dipakai : A, B, C, D, E dan F)
`%f`	untuk menampilkan bilangan real dalam notasi : dddd.dddddd
`%e` `%E`	untuk menampilkan bilangan real dalam notasi eksponensial
`%g` `%G`	untuk menampilkan bilangan real dalam bentuk notasi seperti `%f`, `%E` atau `%F` bergantung pada kepresisian data (digit `0` yang tak berarti tak akan ditampilkan)
l	merupakan awalan yang digunakan untuk `%d`, `%u`, `%x`, `%X`, `%o` untuk menyatakan long int (misal `%ld`). Jika diterapkan bersama `%e`, `%E`, `%f`, `%F`, `%g` atau `%G` akan menyatakan *double*
`L`	Merupakan awalan yang digunakan untuk `%f`, `%e`, `%E`, `%g` dan `%G` untuk menyatakan *long double*
`h`	Merupakan awalan yang digunakan untuk `%d`, `%i`, `%o`, `%u`, `%x`, atau `%X`, untuk menyatakan *short int.*

Contoh di bawah ini akan menjelaskan perbedaan format `%g`, `%e` dan `%f` dalam menampilkan bilangan real.

```
/*File program : form_efg.c
Perbedaan format %g, %e dan %f  */

#include <stdio.h>

main()
{
   float x = 251000.0f;

   printf("Format e  => %e\n", x);
   printf("Format f  => %f\n", x);
   printf("Format g  => %g\n", x);
}
```

Contoh eksekusi :

```
Format e => 2.510000e+005
Format f => 251000.000000
Format g => 251000
```

Tampak bahwa penentu format `%e` menampilkan bilangan dalam bentuk eksponensial. Jika penentu fomat yang digunakan berupa `%f`, bagian pecahan secara *default* akan ditampilkan dalam bentuk 6 digit. Sedangkan jika digunakan penentu format `%g`, maka digit yang tak berarti tak akan ditampilkan.

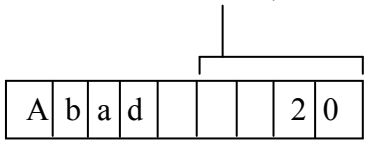
Untuk menentukan panjang medan yang disediakan bagi tampilan data, maka sesudah tanda `%` dalam penentu format dapat disisipi dengan bilangan bulat yang menyatakan panjang medan.

- Untuk data yang berupa bilangan bulat, misal pada :

  ```
  printf("Abad %4d", 20);
  ```

  `%4d` menyatakan medan untuk menampilkan bilangan **`20`** adalah sepanjang 4 karakter.

  ```
  printf("Abad %4d", 20);
  ```



- Untuk data yang berupa bilangan real, spesifikasi medannya berupa

m.n

  m = panjang medan

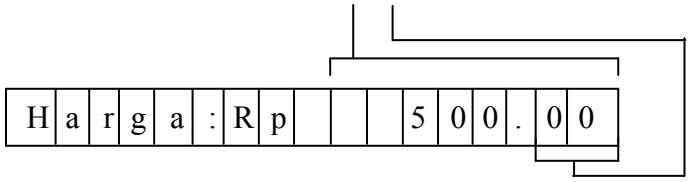
  n = jumlah digit pecahan

  Contoh pada pernyataan :

```
printf("Harga : Rp %8.2f\n", 500.0);
```

`%8.2f` menyatakan panjang medan dari bilangan real yang akan ditampilkan adalah 8 karakter dengan jumlah digit pecahan 2 buah.

```
printf("Harga : Rp %8.2f\n", 500.0);
```



Kalau hanya jumlah digit pecahan yang perlu ditentukan, panjang medan tak perlu disertakan, misal :

```
printf("%.2f\n", 600.0);
printf("%.2f\n", 7500.25);
```

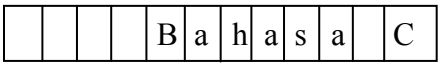
hasilnya :

```
600.00
7500.25
```

- Untuk data yang berupa string, contoh :

```
printf("%12s", "Bahasa C");
```

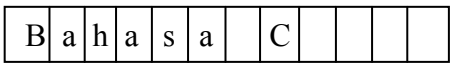
maka akan ditampilkan sebagai berikut



Tampak dalam berbagai jenis data di atas, penentu format yang mengandung panjang medan, secara *default* akan menampilkan data dalam bentuk rata kanan terhadap panjang medan yang diberikan. Untuk data string yang biasanya dikehendaki untuk ditampilkan dalam bentuk rata kiri, maka sesudah tanda `%` pada penentu format `%s` perlu disisipkan tanda − (minus), contoh :

```
printf("%-12s", "Bahasa C");
```

menyatakan bahwa string akan ditampilkan dalam medan dengan panjang 12 karakter dan diatur rata kiri. Sehingga tampilan di atas berubah menjadi :



```
/* File program : formatpjg.c
Contoh penggunaan format panjang medan data */

#include <stdio.h>

main()
{
     int nilai1 = 20;
     float nilai2 = 500.0f;

     printf("Abad %5d\n", nilai1);
     printf("%10.2f\n", nilai2);
     printf("%10s\n", "Bahasa C");
     printf("%-10s\n", "Bahasa C");
}
```

Contoh eksekusi :

```
Abad       20
       500.00
     Bahasa C
Bahasa C
```

2.5.2 Fungsi *putchar()*

Fungsi *putchar()* digunakan khusus untuk menampilkan sebuah karakter di layar. Penampilan karakter tidak diakhiri dengan perpindahan baris.

Contoh :

```
putchar('A');
```

menghasilkan keluaran yang sama dengan

```
printf("%c",'A');
```

2.6. Memasukan Data dari Keyboard

Data dapat dimasukan lewat keyboard saat eksekusi berlangsung. Untuk keperluan ini, C menyediakan sejumlah fungsi, di antaranya adalah *scanf()*, *getchar()*.

2.6.1. Fungsi*scanf()*

Fungsi *scanf()* merupakan fungsi yang dapat digunakan untuk memasukkan berbagai jenis data. Misalnya untuk memasukkan data jari-jari lingkaran pada contoh program **lingkaran.c**, maka penulisan

```
radius = 20;
```

dapat diganti menjadi

```
scanf("%f",&radius);
```

Selengkapnya, terlihat dalam contoh program di bawah ini.

```
/* File program : lingkaran.c
Menghitung keliling dan luas lingkaran */

#include <stdio.h>

main()
{
     double radius, keliling, luas;

     printf("Masukkan jari-jari lingkaran : ");
     scanf("%lf",&radius);

     keliling = 2 * 3.14 * radius; /* PI = 3.14 */

     luas = 0.5 * 3.14 * radius * radius;

     printf("\nData lingkaran\n");
     printf("Jari-jari = %8.2lf\n", radius);
     printf("Keliling  = %8.2lf\n", keliling);
     printf("Luas      = %8.2lf\n", luas);
}
```

Contoh eksekusi :

```
Masukkan jari-jari lingkaran = 5

Data lingkaran
Jari-jari =      5.00
Keliling  =     31.40
Luas      =     39.25
```

Bentuk *scanf()* sesungguhnya menyerupai fungsi *printf()*. Fungsi ini melibatkan penentu format yang pada dasarnya sama digunakan pada *printf()*. Secara umum bentuk *scanf()* adalah sebagai berikut :

```
scanf("string kontrol", daftar_argumen);
```

Dengan string kontrol dapat berupa :

- Penentu format
- Karakter spasi-putih *(white-space)*
- Karakter bukan spasi-putih

Penentu format menyatakan jenis data yang akan dibaca. Pada *scanf()* penentu format dapat berupa salah satu di antara yang ada pada daftar berikut :

Tabel 2.6 Penentu format *scanf()*

`%c`	membaca sebuah **karakter**
`%s`	membaca sebuah **string** (di bahas pada bab vii)
`%i` atau `%d`	membaca sebuah **integer desimal**
`%e` atau `%f`	membaca sebuah **bilangan real** (bisa dalam bentuk eksponensial)
`%o`	membaca sebuah **integer oktal**
`%x`	membaca sebuah **integer heksadesimal**
`%u`	membaca sebuah **integer tak bertanda**
`l`	awalan untuk membaca data long int (misal : `%ld`) atau untuk membaca data **double** (misal : `%lf`)
`L`	awalan untuk membaca data **long double** (misal : `%Lf`)
`h`	awalan untuk membaca data **short int**

Pada bentuk *scanf(),* **daftar_argumen** dapat berupa satu atau beberapa argumen dan haruslah berupa **alamat**. Misalnya hendak membaca bilangan real dan ditempatkan ke variabel radius, maka yang ditulis dalam *scanf()* adalah alamat dari **radius**. Untuk menyatakan alamat dari variabel, di depan variabel dapat ditambahkan tanda `&` (tanda `&` dinamakan sebagai operator alamat). Sehingga `&radius` menyatakan alamat dari **radius**. Dalam bentuk yang lengkap :

```
scanf("%f", &radius);
```

berarti (bagi komputer) : "bacalah sebuah bilangan real (`%f`) dan tempatkan ke alamat dari **radius** (`&radius`)".

2.6.3 Fungsi*getchar()*

Fungsi *getchar()* digunakan khusus untuk menerima masukan berupa sebuah karakter dari keyboard. Contoh :

```
c = getchar();
scanf("%c", &c);
```

maka variabel **c** akan berisi karakter yang diketikkan oleh user atau `EOF` *(end of file)* jika ditemui akhir dari file.

Kesimpulan :

- Data merupakan suatu nilai yang bisa dinyatakan dalam bentuk konstanta atau variabel.
- Konstanta menyatakan nilai yang tetap, sedangkan variabel menyatakan nilai yang dapat diubah-ubah selama eksekusi berlangsung,
- Variabel yang akan digunakan haruslah dideklarasikan terlebih dahulu, adakalanya langsung dideklarasikan sekaligus diberi nilai (diinisialisasi).
- Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi
- Operator yang terkait dengan operasi aritmatika antara lain adalah operator aritmatika, operator penurunan dan penaikan, operator penugasan *(assignment)* dan operator kombinasi (pemendekan)
- Untuk menampilkan data/informasi ke layar digunakan fungsi *printf()* dan *putchar()*.
- Untuk memasukkan data melalui keyboard saat eksekusi berlangsung digunakan fungsi *scanf()* dan *getchar()*.

Latihan :

1. Mengapa nama-nama variabel di bawah ini tidak valid ?

```
(a)  value$sum
(b)  exit flag
```

(c) 3lotsofmoney
(d) char

2. Berapakah hasil akhir dari program berikut :

```
#include <stdio.h>
main()
{
     int a = 22;

     a = a + 5;
     a = a-2;
     printf("a = %d\n", a);
}
```

3. Berapakah nilai x setelah pernyataan-pernyataan berikut dijalankan, apabila x bertipe *int* :

   (a) x = (2 + 3) – 10 * 2;
   (b) x = (2 + 3) – (10 * 2);
   (c) x = 10 % 3 * 2 + 1;

4. Nyatakan dalam bentuk pernyataan :

   (a) $y = bx^2 + 0{,}5x - c$
   (b) $y = \frac{0{,}3xz}{2a}$

5. Apa hasil eksekusi dari program berikut :

```
#include <stdio.h>
main()
{
     char kar = 'A';

     kar = kar + 32;
     printf("%c\n",kar);
}
```