

## Pre Lab P-03 Tipe Data, Operator dan Expresi

Sebagai penunjang untuk mengerjakan pdp-03 di lab. Maka anda harus mengacu pada rumus-rumus matematika dan statistic. Dibawah ini silahkan pelajari mengenai tipe data, operator dan expresi dalam bahasa C.

### Tipe Data Dasar

Tipe data dasar/sederhana terdiri seperti dalam table berikut, merupakan jenis data yang dikenakan pada variable. Konstanta, dan hasil balik fungsi.

	Nam Tipe Data	Dalam bahasa c
1.	Integer	int
2.	Character	char
3.	Floating Point	float
4.	Double precision floating point	double
5.	Void	void

### Contoh

Variable: `int Akar, char c, float ipk, double pecahan`

Konstanta: `const float PI=3.14;`

Hasil balik fungsi: `int tambah(int a, int b), void tukar(int *a,int *b)`

### Ukuran dan range tipe data dasar

<b>Tipe Data</b>	<b>Range Nilai</b>
char	-128 hingga 127
int	-32768 hingga +32767
float	3.4 e-38 hingga 3.4 e+38
double	1.7 e-308 hingga 1.7 e+308

Dari masing-masing tipe data di atas mereka memiliki jangkauan nilai, artinya setiap tipe mempunyai batasan nilai, artinya jika anda membuat variable bertipe int, maka hanya dapat diisi dengan nilai antara -32768 hingga +32767, perhatikan masalah tanda (-/+), jika nilai anda bertanda (bilangan bulat positif dan negatif), maka jenis tipe data dapat dispesifikasikan sebagai **signed int**, dan jika tidak bertanda (bilangan bulat positif), di tulis dengan **unsigned int**. Begitu juga untuk tipe data yang lain akan mengikuti.

## Ukuran dan Range tipe data pada computer 16 bit.

Secara lengkap maka tipe data, ukuran dan trange dalam computer 16 bit sebagai berikut :

<b>TYPE</b>	<b>UKURAN (Bits)</b>	<b>Range</b>
Char/Signed Char	8	-128 hingga 127
Unsigned Char	8	0 hingga 255
Int/Signed int	16	-32768 hingga 32767
Unsigned int	16	0 hingga 65535
Short int/Signed short int	8	-128 hingga 127
Unsigned short int	8	0 hingga 255
Long int/signed long int	32	-2147483648 hingga 2147483647
Unsigned long int	32	0 hingga 4294967295
Float	32	3.4 e-38 hingga 3.4 e+38
Double	64	1.7e-308 hingga 1.7e+308
Long Double	80	3.4 e-4932 hingga 3.4 e+4932

## Nama Tipe Data dan konversinya dalam C

### **Tipe Data**

Character  
Unsigned Character  
Signed Character  
Signed Integer  
Signed Short Integer  
Signed Long Integer  
UnSigned Integer  
UnSigned Short Integer  
UnSigned Long Integer  
Floating Point  
Double Precision Floating Point  
Extended Double Precision Floating Point

### **Equivalent dalam C**

char  
unsigned char  
signed char  
signed int/ int  
signed short int/ short int/ short  
signed long int/long int/ long  
unsigned int / unsigned  
unsigned short int/ unsigned short  
unsigned long int/ unsigned long  
float  
double  
long double

## sizeof

Jika kita akan melihat jumlah memory yang sebenarnya di pakai oleh suatu tipe data kita dapat menggunakan makro sizeof saat program berjalan formatnya :

```
sizeof object atau sizeof(type)
```

contoh :

```
size_t size;  
int i;
```

```
size = sizeof(i); // akan di set menjadi 4 ukuran memory-nya
sizeof(char)      // akan di set menjadi 1 ukuran memory-nya
```

### **Tipe Data yang didefinisikan sendiri/User**

Format : **typedef** <type> <nama>;

Contoh :

```
typedef int gaji;
typedef float rata2;
```

### **Deklarasi Kelas Penyimpanan (storage class)**

Dalam bahasa c variable tidak hanya merupakan tipe data, tetapi juga merupakan kelas penyimpanan yang menyediakan informasi mengenai lokasi dan keberadaannya. Kelas penyimpanan membagi bagian program dengan variable yang di kenal sebagai : **auto** : adalah variable local yang dikenal hanya pada fungsi yang di deklarasikan. Tipe ini merupakan default storage class.

**static** : adalah variable local yang ada dan tetap menghandle suatu nilai walaupun setelah control di transferkan ke suatu fungsi

**extern** : Adalah variable global yang di kenal ke seluruh fungsi dalam File

**register** : Adalah variable social yang disimpan dalam register prosesor.

### **Symbolik Konstak**

Format : **#define** <nama\_simbol> <nilai\_konstan>

Misal :

```
#define nilai 100
#define total 50
#define pi 3.1415
```

### **Variabel Konstan**

Contoh: **const** int class\_size = 40;

## Input dan Output dalam program

### **printf()**

Fungsi printf digunakan untuk menampilkan suatu keluaran pada layar. Perhatikan penggunaan printf berikut ini :

```
#include<stdio.h>
main()
{
    int nilai=3;
    printf("Bahasa C menyenangkan \n"); //pertama
    printf("nilai = %d \n",nilai);      //kedua
}
```

Fungsi printf yang *pertama* disertai dengan **escape sequence** (**\n**) yang berfungsi untuk pindah baris. Jadi dengan fungsi printf yang pertama maka kursor akan turun satu baris. Sedangkan fungsi printf yang *kedua* dimana terdapat format data **%d** berfungsi untuk menampilkan data dengan tipe integer. Jangan lupa dengan variabel yang menyimpan nilai tersebut harus disertakan setelah tanda petik terakhir.

Tampilan program tersebut :

```
Bahasa C menyenangkan
nilai = 3
```

### **puts**

Cara lain untuk menampilkan suatu keluaran ke layar adalah menggunakan fungsi puts. Tetapi fungsi puts hanya digunakan untuk menampilkan nilai **string** dan sudah mengandung **line feed** atau **escape sequence** ganti baris. Perhatikan contoh berikut :

```
#include
main()
{
    puts("Bahasa C menyenangkan ");
    puts("Belajar bahasa C ");
}
```

Kini tidak perlu lagi menggunakan "\n" untuk ganti baris baru. Tampilan program :

```
Bahasa C menyenangkan
Belajar bahasa C
```

### **putchar**

Fungsi ini digunakan untuk menampilkan sebuah **karakter** saja dan **tidak** mengandung **escape sequence** "\n". perhatikan contoh berikut :

```

#include
main()
{
    char a;
    printf("a = ");scanf("%c",&a);
    printf("Data yang anda masukkan ");
    putchar('\n');    //pertama
    putchar(a);      //kedua
}

```

Pada putchar yang **petama** kita ingin membuat program untuk ganti baris menggunakan "\n". kita menggunakan tanda petik tunggal (') karena karakter dalam program akan dikenal jika diberi tanda petik tunggal. Fungsi putchar ini hanya bisa menampilkan satu buah karakter saja sehingga apapun nilai yang kita masukkan hanya karakter pertama yang akan ditampilkan.

Berikut tampilan programnya :

```

a = 12
Data yang anda masukkan
1

```

### scanf

Jika kita ingin memasukkan data dari keyboard, kita dapat menggunakan fungsi scanf ini. Data selanjutnya akan didefinisikan sebagai data variabel. Jika fungsi scanf ini digunakan untuk membaca data dengan tipe array, karakter yang selanjutnya kita sebut dengan istilah string, maka fungsi ini hanya akan membaca data sampai ditemukan blank. Dengan demikian nilai setelah blank dianggap bukan lagi nilai dari variabel yang akan mengisi variabel tersebut. Fungsi scanf lebih cocok digunakan untuk data-data numerik.

Fungsi scanf jika digunakan maka harus disertai operator penanda alamat & didepan nama variabel yang digunakan untuk menyimpan data tersebut. Jika tidak mempergunakan operator alamat & ini maka ketika data diinputkan akan muncul pesan error Segmentation fault. Sedangkan untuk data string tidak perlu menggunakan operator ini. Tetapi jika digunakan tidak menjadi masalah. Fungsi scanf biasanya digunakan bersama-sama dengan fungsi printf. Perhatikan contoh berikut :

```

#include
main()
{
    char a[25];
    int b;
    printf("a = ");scanf("%s",a);
    printf("b = ");scanf("%d",&b);
}

```

```

printf("Data yang anda masukkan \n");
printf("a = %s \n",a);
printf("b = %d \n",b);
}

```

Setiap kali memasukkan data harus diikuti dengan menekan ENTER. Berikut adalah tampilan programnya.

```

a = wawan
b = 12
Data yang anda masukkan
a = wawan
b = 12

```

### gets

Jika kita menggunakan fungsi scanf untuk membaca data yang bertipe string, maka data tersebut hanya akan dibaca sampai ditemukan spasi. Misalnya nama "wawan darmawan" hanya kata "wawan" yang akan dibaca oleh program, sedangkan kata "darmawan" tidak terbaca karena sebelum kata tersebut program telah menemukan spasi yang mengakibatkan data ke variabel yang menyimpan nama dianggap selesai. Untuk keperluan pemasukan data string yang panjang dipisahkan dengan spasi, bahasa C menyediakan fungsi gets. Perhatikan contoh berikut ini :

```

#include
main()
{
    char nama[25];
    char alamat[50];
    printf("Nama : ");gets(nama);
    printf("Alamat : ");gets(alamat);
    printf("\n");
    printf("Data yang anda masukkan \n");
    printf("Nama : %s \n",nama);
    printf("Alamat : %s \n",alamat);
}

```

Tampilan program :

```

Nama : wawan darmawan
Alamat : bandung
Data yang anda masukkan
Nama : wawan darmawan
Alamat : bandung

```

## getchar

Jika ingin memasukkan sebuah nilai karakter ke variabel yang bertipe karakter maka kita dapat menggunakan perintah getchar. Perhatikan contoh program berikut ini :

```
#include
main()
{
    char nilai;
    printf("Nilai anda : ");
    nilai=getchar();
    printf("Nilai yang anda masukkan = %c \n",nilai );
}
```

Berikut tampilannya :

**Nilai anda : A**

**Nilai yang anda masukkan = A**

## FORMAT I/O

### Format karakter / specifier dalam Printf() dan scanf()

Karakter escape dalam format I/O

- \n (newline)
- \t (tab)
- \v (vertical tab)
- \f (new page)
- \b (backspace)
- \r (carriage return)
- \n (newline)

Format Specifier (%)	Type	Hasil Output
c	char	Karakter tunggal
i,d	int	Bilangan decimal
o	int	Bilangan octal
x,X	int	Bilangan Hexadesimal Notasi hrf besar/kecil
u	int	unsigned int
s	char *	Cetak string

f	double/float	format -m.ddd...
e,E	"	Format Scientific -1.23e002
g,G	"	e atau f sama saja
%	-	Cetak karakter %

**Contoh :**

```
printf("%-2.3f \n",17.23478);           //menjadi 17.235
printf("VAT=17.5%%\n");                //menjadi VAT=17.5%
scanf("%d",&i);
char string[80];
scanf("%s",string);
```

**contoh lain**

```
#include <stdio.h>
int main() {
int a = 47;
int b = 218;
int c = 12345;
float f = 27.876;
double d = 27.876;

printf("Out %8d\n", a) ;
printf("Out %08d\n", a) ;
printf("Out %-8d\n", a) ;
printf("Out %-08d\n", a) ;

printf("Out %8d\n", b) ;
printf("Out %08d\n", b) ;
printf("Out %-8d\n", b) ;
printf("Out %-08d\n", b) ;

printf("Out %8d\n", c) ;
printf("Out %08d\n", c) ;
printf("Out %-8d\n", c) ;
printf("Out %-08d\n", c) ;

printf("Out %8.2e\n", f*f) ;
printf("Out %-8f\n", f) ;
printf("Out %-8.5f\n", f) ;
printf("Out %8f\n", f) ;

printf("Out %8.2e\n", d) ;
printf("Out %-8f\n", d) ;
```

```
printf("Out %-8.5f\n", d) ;
printf("Out %8f\n", d) ;
return 0;
};
```

## Operator , Operasi dan Expresi

### Operator Unary

Operator	Arti
=====	=====
&	Alamat dari; value is the location of the operand
*	Isi dari; value is what is stored at the location
-	Negasi
+	Nilai dari operator
!	negasi Logical ((!E) equivalen dengan(0==E) )
~	komplemen Bit-wise

### Operator Arithmetic

Nama Operator		Syntax
<u>assignment</u> /Penugasan		a = b
<u>Addition</u> /Penjumlahan		a + b
<u>Subtraction</u> /Pengurangan		a - b
<u>Unary plus</u>		+a
<u>Unary minus</u>		-a
<u>Multiplication</u> /Perkalian		a * b
<u>Division</u> /Pembagian		a / b
<u>Modulo</u> /Sisa hasil bagi		a % b
<u>Increment</u>	Prefix	++a
	Suffix	a++
<u>Decrement</u>	Prefix	--a
	Suffix	a--

### Operator Perbandingan /Operator Relational

Nama Operator	Syntax
Sama Dengan	a == b

Tidak Sama Dengan	$a \neq b$
Lebih besar dari	$a > b$
Kurang dari	$a < b$
Lebih besar atau sama dengan	$a \geq b$
Kurang dari atau sama dengan	$a \leq b$

### Operator Logical

Nama Operator	Syntax
<u>Logical (NOT)</u>	$!a$
<u>Logical AND</u>	$a \&\& b$
<u>Logical OR</u>	$a \ \  b$

### Operator Bitwise

Operator name	Syntax
Bitwise NOT	$\sim a$
Bitwise AND	$a \& b$
Bitwise OR	$a \ \  b$
Bitwise XOR	$a \wedge b$
Bitwise geser kiri	$a \ll b$
Bitwise geser kanan	$a \gg b$

### Operator Assignment Komposit

Operator name	Syntax
Addition assignment	$a += b$
Subtraction assignment	$a -= b$
Multiplication assignment	$a *= b$
Division assignment	$a /= b$
Modulo assignment	$a \% = b$
Bitwise AND assignment	$a \& = b$
Bitwise OR assignment	$a \ \  = b$
Bitwise XOR assignment	$a \wedge = b$
Bitwise geser kiri assignment	$a \ll = b$
Bitwise geser kanan assignment	$a \gg = b$

## Operator Anggota(Member) dan pointer

Nama Operator	Syntax
<u>Array subscript</u>	<b>a[b]</b>
Indirection ("variable yg ditunjuk oleh <i>a</i> ")	<b>*a</b>
Reference ("alamat dari <i>a</i> ")	<b>&amp;a</b>
anggota <i>b</i> dari obyek yang di tunjuk oleh <i>a</i>	<b>a-&gt;b</b>
anggota <i>b</i> dari obyek <i>a</i>	<b>a.b</b>
Obyek yang ditunjuk oleh <i>a</i> , anggota yang ditunjuk oleh <i>b</i>	<b>a-&gt;*b</b>
Obyek <i>a</i> , anggota yang ditunjuk oleh <i>b</i>	<b>a.*b</b>

## Operator Lainnya

Operator name	Syntax
<u>Function call/Pemanggilan fungsi</u>	<b>a()</b>
<u>comma/koma</u>	<b>a, b</b>
<u>ternary conditional</u>	<b>a ? b : c</b>
<u>Scope resolution</u>	<b>a::b</b>
<u>size-of</u>	<b>sizeof(a)</b> <b>sizeof(type)</b>
Type identification	<b>typeid(a)</b> <b>typeid(type)</b>
<u>cast</u>	<b>(type) a</b>
Alokasi (c)	<b>malloc()</b> atau <b>calloc()</b>
Dealokasi (c)	<b>free</b>
<u>Allocate storage (c++)</u>	<b>new type</b>
<u>Allocate storage (array) (c++)</u>	<b>new type[n]</b>
<u>Deallocate storage (c++)</u>	<b>delete a</b>
<u>Deallocate storage (array)(c++)</u>	<b>delete[] a</b>