|  |  |  |
| --- | --- | --- |
| logounikombw | **Diktat Mata Kuliah Pemrograman I** | IF |
| BAB III  TIPE DATA, VARIABEL & OPERASI PERHITUNGAN |

Tipe Data

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

* Tipe data karakter
* Tipe data bilangan bulat.
* Tipe data bilangan pecahan.

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

Setiap tipe data mempunyai jangkauan nilai yang berbeda.

1. **Tipe data karakter**

Untuk tipe data karakter kita gunakan perintah char.

Contoh

char karakter;

char kar1,kar2,kar3;

char kar4=’A’;

char kar5=65;

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

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

Pemberian nilai kepada suatu karakter digunakan perintah sebagai berikut :

karakter=’A’;

Atau

**karakter=65;**

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

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

**printf(“Karakter dilihat dalam bentuk karakter = %c.\n”,karakter);**

**printf(“Karakter dilihat dalam bentuk angka = %d.\n”,karakter);**

Contoh program

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

Hasil dari eksekusi program adalah :

**Nilai variable K1 adalah A**

**Nilai variable K2 dalam bentuk angka = 65**

**Keterangan program Lat2.CPP**

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

Perintah “**k1=’A’;**” pada baris 7 adalah perintah untuk memasukan nilai karakter A kapital ke dalam variable k1 sehingga untuk baris berikutnya k1 berisi karakter A kapital atau angka 65.

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

Perintah printf pada baris 9 berarti perintah penulisan ke layar sesuai dengan format “Nilai variable k1 adalah %c\n”. Karakter %c tidak dicetak sebagai %c tetapi akan diganti dari variable yang sesuai dengan urutannya yaitu k1 dalam bentuk karakter. Perintah printf pada baris 10 cara kerjanya sama dengan perintah printf pada baris 9 bedanya hanya tanda %d berasal ditulis berdasarkan isi variable k2 yang dicetak dalam bentuk angka bukan karakter. Tanda %d digunakan untuk format pencetakan data dalam bentuk bilangan bulat.

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

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

1. **Tipe data bilangan bulat**

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

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

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

**int a;**

**unsigned int b;**

**char c;**

**long d;**

Contoh Program :

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

Perintah di atas akan menampilkan hasil seperti di bawah ini :

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

1. **Tipe data bilangan pecahan**

Tipe data untuk bilangan pecahan terdiri dari beberapa jenis yaitu :

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

Contoh Program

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

Hasil eksekusi program dapat dilihat di bawah ini :

|  |
| --- |
| Nilai a adalah : 1234567939550609410.00  Nilai d adalah : 1234567890123456770.00 |

1. **Tipe data string**

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

Cara pendeklarasian adalah :

**char nama[50];**

**char \*alamat;**

Contoh program :

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Catatan**  copy | Pemilihan tipe data harus hati-hati. Pertimbangkan jangkauan yang dimiliki oleh tipe data yang dipilih. Kesalahan dalam memilih tipe data akan menimbulkan suatu hasil yang tidak diperkirakan. Contoh :  int a=32000;  int b=769;  int c;  c=a+b;  printf(“%i + %i = %i\n”,a,b,c);  Jika program tersebut dijalankan, maka akan menghasilkan output seperti berikut :  32000 + 769 = -32767  Hal tersebut terjadi karena jangkuan nilai c sudah melebihi jangkauan nilai untuk sebuah tipe data int. Bila suatu variable telah melebihi jangkauan nilainya maka nilai variable tersebut akan berputar menjadi nilai minimalnya dan jika nilainya kurang dari minimal jangkauan nilainya maka variable tersebut akan terisi oleh bilangan maksimal tipe tersebut.   |  |  |  |  | | --- | --- | --- | --- | | Nilai yang diharapkan | 32767 | 32768 | 32769 | | Nilai pada variable C | 32767 | -32768 | -32767 | |

Operator-Operator Perhitungan

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

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

Contoh program :

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

Program di atas akan menampilkan hasil seperti berikut :

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

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

Operator-Operator Manipulasi Bit

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

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

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

Prioritas eksekusinya adalah

|  |  |
| --- | --- |
| Tertinggi  Terendah | ~  << >>  &  ^  | |

Tabel Kebenaran dari tiap operator manipulasi bit adalah :

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

Contoh :

|  |
| --- |
| **#include <stdio.h>**  **#include <conio.h>**  **main()**  **{**  **unsigned int x,y,and,or,not,xor;**  **x=78;// 41h**  **y=520;// 208h**  **and=x & y;**  **or=x | y;**  **xor=x ^ y;**  **not = ~y;**  **clrscr();**  **printf("x : %6u : %4Xh\n",x,x);**  **printf("y : %6u : %4Xh\n",y,y);**  **printf("--------------------------\n");**  **printf("x & y : %6u : %4Xh\n",and,and);**  **printf("x | y : %6u : %4Xh\n",or,or);**  **printf("x ^ y : %6u : %4Xh\n",xor,xor);**  **printf(" ~y : %6u : %4Xh\n",not,not);**  **getch();**  **return 0;**  **}** |

Jika di-Run akan menghasilkan :

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

Pembuktian :

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

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

Contoh :

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

Jika di-Run akan menghasilkan :

|  |
| --- |
| **Angka : 50 : 32h**  **x=angka << 2 : 200 : c8h**  **x=angka >> 2 : 12 : ch** |

Pembuktian :

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

Aturan-Aturan Perhitungan

Perhatikan perintah berikut :

|  |
| --- |
| **float a;**  **a= 9/5;** |

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

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

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

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

|  |
| --- |
| **float a,b;**  **a=(float)9/5;**  **b=(float)(9/5);** |

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

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

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

Jika dieksekusi, akan menghasilkan :

|  |
| --- |
| **F : 65.900002**  **I : 65**  **C : A** |

Keterangan :

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

Mendefinisikan Konstanta Simbolis

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

Contoh :

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

Jika dieksekusi, akan menghasilkan :

|  |
| --- |
| **Perhitungan Lingkaran**  **Dibuat Oleh : Andri Heryandi**  **======================**  **Radius : 10.00**  **Keliling : 62.80**  **Luas : 314.00** |

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