BAB 7. POINTER

Suatu pointer (variable penunjuk) adalah suatu variable yang berisi dengan alamat lokasi, yaitu suatu memori tertentu. Bahasa C menyediakan 2 buah operator untuk operasi pointer yaitu operator '*' dan operator '&'.

Operator alamat (Address operator (&))

Pada pendeklarasian variable, user tidak diharuskan menentukan lokasi sesungguhnya pada memory. Hal ini akan dilakukan secara otomatis oleh compiler dan operating system pada saat run-time. Jika ingin mengetahui di mana suatu variable disimpan, dapat dilakukan dengan memberikan tanda ampersand (&) di depan variable, yang berarti "address off".

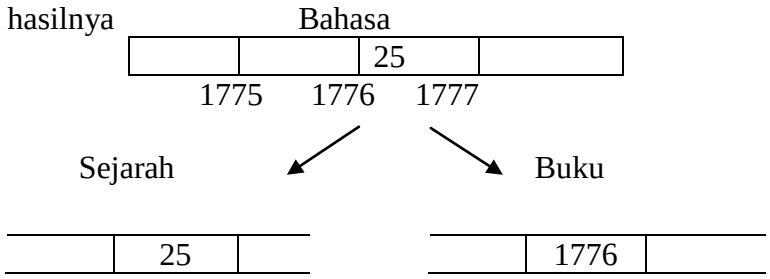
Contoh :

Buku = &bahasa;

Akan memberikan variabel buku alamat dari bahasa, karena variable bahasa diberi awalan ampersand (&), maka menjadi pokok disini adalah alamat dalam memory, bukan isi variable. Misalkan bahasa diletakkan di alamat 1776 kemudian dituliskan instruksi sbb:

Bahasa = 25;
Sejarah = bahasa;
buku = &bahasa;

Maka hasilnya



Operator Reference (*)

Dengan menggunakan pointer kita dapat mengakses nilai yang tersimpan secara langsung dengan memberikan awalan operator asterisk(*) pada identifier pointer, yang berarti "value pointed by".

Contoh ;

BhsC = *buku;

(dapat dikatakan bahwa BhsC sama dengan nilai yang ditunjuk oleh buku). BhsC = 25, karena buku di alamat 1776, dan nilai yang berda pada lamat 1776 adalah 25.

Deklarasi Pointer

Variabel pointer dideklarasikan dengan nama variabelnya ditulis dengan diawali karakter asterisk.

Bentuk umum :

Tipe-data *nama-variabel-pointer;

Contoh : int *data; berarti data adalah sebuah pointer yang menunjuk ke jenis data integer.

Tipe dari variable pointer menunjukkan tipe dari data yang ditunjuknya.

Inisialisasi Variabel Pointer

Pemberian harga awal pada variable-variabel pointer dapat sekaligus dilakukan pada saat variable-variabel tersebut dideklarasikan. Harga awal yang diberikan adalah alamat lokasi memory.

Contoh : int i,j; int *intptr = &i; atau bisa juga
int i,j, *intptr ; intptr = &i ;
intptr variable pointer ke jenis data integer dengan memberi harga awal berupa alamat variable i.

Contoh :

```
#include<stdio.h>

void main( )

{       int i,j, *intptr; i = 890;

        ptintf("variable i terletak pd alamat memory %p.\n",&i);

        printf("alamat i = %p,menerima nilai %d\n",&i, i);

        intptr = &i;      // intptr menunjuk alamat variable i

        printf('alamat intptr = %p, menunjuk ke nilai %d\n",intptr,*intptr);

        j = *intptr; printf("alamat j = %p,menerima nilai %d\n",&j, j);

}
```

output :

A. Pointer dan Fungsi

Contoh :

```
/*program pengriman argument dengan alamat*/

#include<stdio.h>

void main ( )

{       int fungsi(int *x);       /*prototipe fungsi atau deklarasi fungsi*/

        int x = 5, y =4;

        printf("Dalam fungsi main( ): x = %d, y = %d\n", x, y);

        y += fungsi(&x);       /*memanggil fungsi &mengirim alamat x */

        printf("Dalam fungsi main( ): x = %d, y = %d\n", x, y);  }

int fungsi(int *x)        /*pendefinisian fungsi*/

{       int y;

        y =  ++*x ;      // nilai x ditambah 1
```

```
    printf("Dalam fungsi1 : x = %d, y = %d\n", *x , y);
    return y ;          /* membalikkan nilai y ke fungsi utam  ke y+=fungsi(&x) */    }
```

output :

Catt : pengiriman alamat sebagai argumen menyebabkan perubahan nilai pada fungsi akan menyebabkan nilai pada fungsi utama juga berubah.

B. Pointer dan Array

Identifier suatu array equivalent dengan alamat dari elemen pertama, pointer equivalent dengan alamat elemen pertama yang ditunjuk. Perhatikan deklarasi berikut :

```
int numbers [20];
int *p;
```

Maka deklarasi di bawah ini juga benar : p = numbers;
p dan numbers equivalent, dan memiliki sifat (properties) yang sama. Perbedaanya, user dapat menentukan nilai lain untuk pointer p. dimana numbers akan selalu menunjuk nilai yang sama seperti yang telah didefiniskan. P, merupakan variable pointer, numbers adalah constant pointer. Karena walaupun instruksi di atas benar, tetapi tidak untuk instruksi di bawah ini :

```
numbers = p;
```

Karena numbers adalah array (constant pointer), dan tidak ada nilai yang dapat diberikan untuk identifier constant (constant identifier).
Operasi penambahan pointer merupakan suatu peningkatan nilai pointer yang menunjukkan lokasi nilai data berikutnya di memory. Misalkan variael pointer x menun jukkan alamat memori 1000, maka operasi penambahan **x+1** menunjukkan alamat **1000+sizeof(x).**

Misalkan suatu array dengan nama x dan pariabel pointer dengan nama p.Alamat elemen-elemen larik dimensi satu ini mulai elemen pertama sampai dengan ke n dapat ditunjukkan sbb :

Elemen ke 1 : &x[0] atau x atau x+0 atau p atau p+0
Elemen ke 2 : &x[1] atau x+1 atau p+1
Elemen ke 3 : &x[2] atau x+2 atau p+2
Elemen ke n : &x[n-1] atau x+(n-1) atau p+(n-1)

Sedangkan untuk mengakses nilai dari elemen array dapat diakses sbb :
Elemen ke 1 : x[0] atau *x atau *(x+0) atau *p atau *(p+0)
Elemen ke 2 : x[1] atau *(x+1) atau *(p+1)
Elemen ke 3 : x[2] atau *(x+2) atau *(p+2)
Elemen ke n : x[n-1] atau *(x+(n-1)) atau *(p+(n-1))

Contoh 1:

```
#include<stdio.h>
int main ( )
{       int a[5];
        int *p;
        p = a; *p = 10;
        p++; *p = 20;
        p = &a[2]; *p=30;
        p = a+3; *p = 40;
        p = a ; *(p+4) = 50;
        for(int n = 0; n< 5; n++)
        printf('%d,",a[n]);  return 0;  }
```

output :

penjelasan : p = a → p = &a[0] ; *p = 10 → a[0] = 10
p++ → p +1 → a + 1 →&a[1]; *p = 20 → a[1] = 20
p = &a[2] → p = a+2 ; *p = 30 → a[2] = 30
p = a+3 → p = &a[3] → *p =40 → a[3] = 40
p = a; *(p+4) = 50 → *(a+4) = 50 →&a[4] = 50

Contoh2 :

```
#include<stdio.h>
main ( )
{       int x[5];
        int *p;
        p = x; x[0] = 7;
        x[1] = x[0];
        x[2] = *p + 4 ;  // x[2] diisi dengan nilai x[0] + 4 = 7+4 = 11
        x[3] = *(p+2) – 2; // x[3] diisi dengan nilai x[2] – 2 = 11 – 2 =9
        x[4] = *(x+3);  // x[4] diisi dengan nilai x[3] = 9
        printf("%d %d %d %d %d", x[0], x[1], x[2], x[3], x[4]);
}  out :
```