Praktikum 10 (1/5)

POINTER

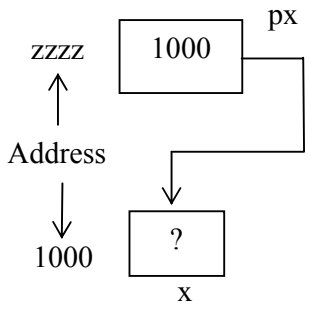
A. TUJUAN

1. Menjelaskan tentang konsep dari variabel pointer

B. DASAR TEORI

Konsep Dasar Pointer

Variabel pointer sering dikatakan sebagai variabel yang menunjuk ke obyek lain. Pada kenyataan yang sebenarnya, variabel pointer berisi alamat dari suatu obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer). Sebagai contoh, **px** adalah variabel pointer dan **x** adalah variabel yang ditunjuk oleh **px**. Kalau **x** berada pada alamat memori (alamat awal) 1000, maka **px** akan berisi 1000. Sebagaimana diilustrasikan pada gambar 8.1 di bawah ini



Gambar 8.1 Variabel pointer px menunjuk ke variabel x

Mendeklarasikan Variabel Pointer

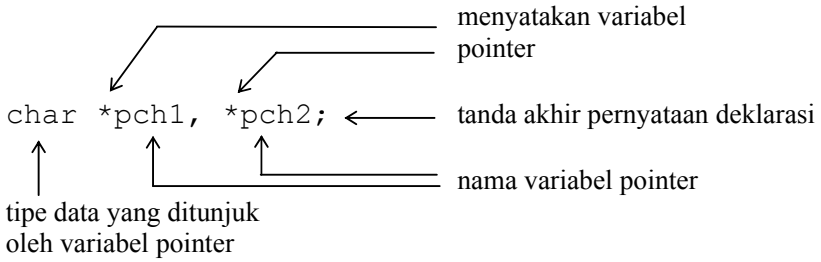
Suatu variabel pointer dideklarasikan dengan bentuk sebagai berikut :

```
tipe *nama_variabel
```

dengan **tipe** dapat berupa sembarang tipe yang sudah dibahas pada bab-bab sebelumnya, maupun bab-bab berikutnya. Adapun **nama_variabel** adalah nama dari variabel pointer. Sebagai contoh :

```
int *px;                    //contoh 1
char *pch1, *pch2;          //contoh 2
```

Contoh pertama menyatakan bahwa **px** adalah variabel pointer yang menunjuk ke suatu data bertipe *int*, sedangkan contoh kedua masing **pch1** dan **pch2** adalah variabel pointer yang menunjuk ke data bertipe *char*.



Gambar 8.2 Ilustrasi pendeklarasian variabel pointer

Mengatur Pointer agar Menunjuk ke Variabel Lain

Agar suatu pointer menunjuk ke variabel lain, mula-mula pointer harus diisi dengan alamat dari variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel, operator **&** (operator alamat, bersifat *unary*) bisa dipergunakan, dengan menempatkannya di depan nama variabel. Sebagai contoh, bila **x** dideklarasikan sebagai variabel bertipe *int*, maka

```
&x
```

berarti "alamat dari variabel **x**". Adapun contoh pemberian alamat **x** ke suatu variabel pointer px (yang dideklarasikan sebagai pointer yang menunjuk ke data bertipe *int*) yaitu :

```
px = &x;
```

Pernyataan di atas berarti bahwa **px** diberi nilai berupa alamat dari variabel **x**. Setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa **px** menunjuk ke variabel **x**.

Mengakses Isi Suatu Variabel Melalui Pointer

Jika suatu variabel sudah ditunjuk oleh pointer, variabel yang ditunjuk oleh pointer tersebut dapat diakses melalui variabel itu sendiri (pengaksesan langsung) ataupun melalui pointer (pengaksesan tak langsung). Pengaksesan tak langsung dilakukan dengan menggunakan operator *indirection* (tak langsung) berupa simbol ***** (bersifat *unary*).

Contoh penerapan operator * yaitu :

```
*px
```

yang menyatakan "isi atau nilai variabel/data yang ditunjuk oleh pointer **px**" . Sebagai contoh jika **y** bertipe *int*, maka sesudah dua pernyataan berikut

```
px = &x;
y = *px;
```

y akan berisi nilai yang sama dengan nilai **x**.

Kedua pernyataan di atas

```
px = &x;
y = *px;
```

sebenarnya dapat digantikan dengan sebuah pernyataan berupa

```
y = x;
```

Seandainya pada program di atas tidak terdapat pernyataan

```
px = &x;
```

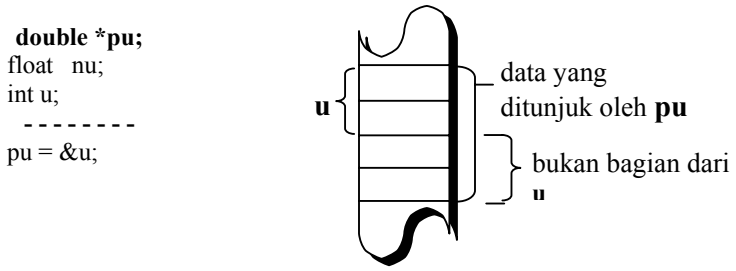
namun terdapat pernyataan

```
y = *px;
```

maka **y** tidaklah berisi nilai **x**, sebab **px** belum diatur agar menunjuk ke variabel **x.** Hal seperti ini harap diperhatikan. Kalau program melibatkan pointer, dan pointer belum diinisialisasi, ada kemungkinan akan terjadi masalah yang dinamakan "bug" yang bisa mengakibatkan komputer tidak dapat dikendalikan *(hang)*.

Selain itu tipe variabel pointer dan tipe data yang ditunjuk harus sejenis. Bila tidak sejenis maka akan terjadi hasil yang tidak diinginkan.

Untuk lebih jelasnya lihat gambar berikut.



Gambar 8.3 Ilustrasi kesalahan yang terjadi karena tipe tidak sejenis

C. PERCOBAAN

Untuk setiap program di bawah ini,

- gambarkan ilustrasi alokasi memori dari setiap baris pernyataan yang diproses
- perkirakan hasil eksekusinya

1. Mengakses isi suatu variabel melalui pointer.

```
#include <stdio.h>

main()
{
  int y, x = 87;
  int *px;

  px = &x;
  y = *px;

  printf("Alamat x    = %p\n", &x);
  printf("Isi px      = %p\n", px);
  printf("Isi x       = %d\n", x);
  printf("Nilai yang ditunjuk oleh px = %d\n", *px);
  printf("Nilai y     = %d\n", y);
}
```

2. Mengakses isi suatu variabel melalui pointer. Tipe variabel pointer dan tipe data yang ditunjuk harus sejenis.

```
#include <stdio.h>

main()
{
  int *pu;
  int nu;
```

```
    int u = 1234;

    pu = &u;
    nu = *pu;

    printf("Alamat dari u               = %p\n", &u);
    printf("Isi pu                      = %p\n", pu);
    printf("Isi u                       = %d\n", u);
    printf("Nilai yang ditunjuk oleh pu = %d\n", *pu);
    printf("Nilai nu                    = %d\n", nu);
}
```

3.
```
#include <stdio.h>

main()
{
  int z = 20, s = 30;
  int *pz, *ps;

  pz = &z;
  ps = &s;

  *pz += *ps;

  printf("z = %d\n", z);
  printf("s = %d\n", s);
}
```

4.
```
#include <stdio.h>

main()
{
  char c = 'Q';
  char *char_pointer = &c;

  printf("%c %c\n", c, *char_pointer);
  c = '/';
  printf("%c %c\n", c, *char_pointer);
  *char_pointer = '(';
  printf("%c %c\n", c, *char_pointer);
}
```

5.
```
#include <stdio.h>

main()
{
  int x = 1, y = 2;
  int *ip;

  ip = &x;
  y = *ip;
  x = ip;
  *ip = 3;

  printf("x = %d, y = %d", x, y);
}
```

D. LAPORAN RESMI

Kumpulkan listing program, ilustrasi alokasi memorinya beserta hasil eksekusinya.