PRAKTIKUM 24

Pointer 1

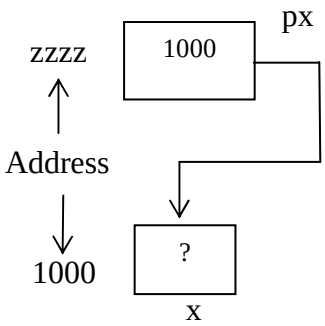
A. TUJUAN PEMBELAJARAN

1. Memahami konsep dari variabel pointer
2. Memahami cara Mengakses Isi Suatu Variabel Melalui Pointer
3. Memahami cara Mengakses dan Mengubah isi Suatu 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 24.1 di bawah ini



Gambar 24.1 Variabel pointer px menunjuk ke variabel x

Mendeklarasikan Variabel Pointer

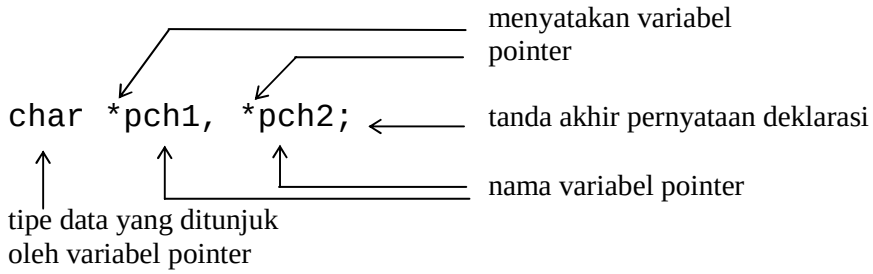
Suatu variabel pointer dideklarasikan dengan bentuk sebagai berikut :

```
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 19.2 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 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 "nilai/value yang ada pada alamat/address **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;
```

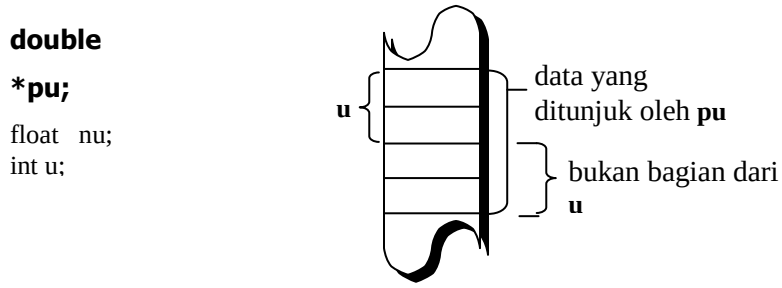
namun terdapat pernyataan

```
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 19.3 Ilustrasi kesalahan yang terjadi karena tipe tidak

Mengakses dan Mengubah isi Suatu Variabel Pointer

Contoh berikut memberikan gambaran tentang pengubahan isi suatu variabel secara tak langsung (yaitu melalui pointer). Mula-mula **pd** dideklarasikan sebagai pointer yang menunjuk ke suatu data bertipe *float* dan **d** sebagai variabel bertipe *float*. Selanjutnya

```
d = 54.5;
```

digunakan untuk mengisikan nilai 54,5 secara langsung ke variabel **d**. Adapun

```
pd = &d;
```

digunakan untuk memberikan alamat dari **d** ke **pd.** Dengan demikian **pd** menunjuk ke variabel **d.** Sedangkan pernyataan berikutnya

```
*pd = *pd + 10;
```
(atau: `*pd += 10;`)

merupakan instruksi untuk mengubah nilai variabel **d** secara tak langsung. Perintah di atas berarti "jumlahkan yang ditunjuk ***pd*** dengan 10 kemudian berikan ke yang ditunjuk oleh **pd**", atau identik dengan pernyataan

```
d = d + 10;
```

Akan tetapi, seandainya tidak ada instruksi

```
pd = &d;
```

maka pernyataan

```
*pd = *pd + 10;
```

tidaklah sama dengan

```
d = d + 10;
```

C. PERCOBAAN

Untuk setiap program di bawah ini,

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

```
1. 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. main(){
     float *pu, nu;
     double u = 1234.0;

     pu = &u;
     nu = *pu;

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

```
3. main(){
    float d = 54.5f, *pd;

    printf("Isi d mula-mula = %g\n", d);

    pd = &d;
    *pd += 10;

    printf("Isi d sekarang  = %g\n", d);
 }
```

4.
```
main(){
   int z = 20, s = 30, *pz, *ps;

   pz = &z;
   ps = &s;
   *pz += *ps;
   printf("z = %d\n", z);
   printf("s = %d\n", s);
}
```

5.
```
main(){
   char c = 'Q', *cp = &c;

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

6.
```
main() {
   int x = 1, y = 2, *ip;

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

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

7.
```
main(){
   int i1, i2, *p1, *p2;

   i1 = 9;
   p1 = &i1;
   i2 = *p1 / 2 – 2 * 3;
   p2 = p1;

   printf("i1=%d,i2=%d,*p1=%d,*p2=%d\n",i1,i2,*p1,*p2);
}
```

8.
```
main() {
   int  count = 10, *temp, sum = 7;

   temp = &count;
   *temp = 32;
   temp = &sum;
   *temp = count;
   sum = *temp * 4;
```

```
      printf("count=%d, *temp=%d, sum=%d\n", count,*temp, sum );
   }
```

```
9. main(){
      int  count = 13, sum = 9, *x, *y;

      x = &count;
      *x = 27;
      y = x;
      x = &sum;
      *x = count;
      sum = *x / 2 * 3;

      printf("count=%d, sum=%d, *x=%d, *y=%d\n", count,sum,*x,*y);
   }
```

```
10. int r, q = 7;
    int go_crazy(int *, int *);

    main()   {
      int *ptr1 = &q;
      int *ptr2 = &q;

      r = go_crazy(ptr1, ptr2);
      printf("q=%d, r=%d, *ptr1=%d,*ptr2=%d\n",q,r,*ptr1,*ptr2);

      ptr2 = &r;

      go_crazy(ptr2, ptr1);
      printf("q=%d, r=%d, *ptr1=%d, *ptr2=%d\n",q,r,*ptr1,*ptr2);
    }

    int go_crazy(int *p1, int *p2){
        int x = 5;

        r = 12;
        *p2 = *p1 * 2;
        p1 = &x;
        return *p1 * 3;
       }
```

D. LAPORAN RESMI

Kumpulkan listing program, ilustrasi alokasi memorinya beserta hasil eksekusinya.