PRAKTIKUM 11

POINTER 1

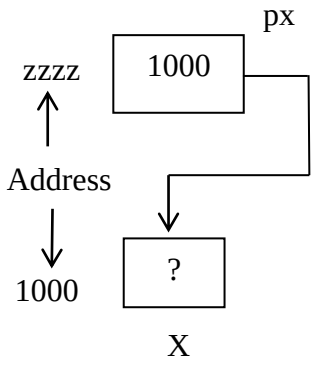
A. Tujuan

1. Menjelaskan tentang konsep dari variabel pointer
2. Menjelaskan tentang pointer array
3. Menjelaskan tentang pointer string

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 13.1 di bawah ini



Gambar 13.1 Variabel pointer px menunjuk ke variabel x

Mendeklarasikan Variabel Pointer

Suatu variabel pointer dideklarasikan dengan bentuk sebagai berikut :

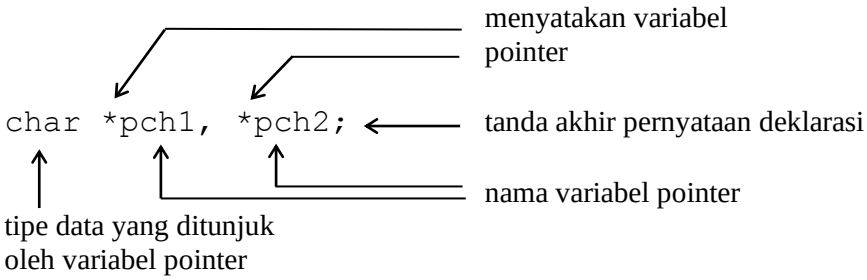
```
tipe *nama_variabel
```

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 13.2 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**.

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;
```

Pointer dan Array

Hubungan antara pointer dan array pada C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer. Pembahasan berikut akan memberikan gambaran hubungan antara pointer dan array. Misalnya dideklarasikan di dalam suatu fungsi

```
int tgl_lahir[3] = { 01, 09, 64 };
```

dan

```
int *ptgl;
```

Kemudian diberikan instruksi

```
ptgl = &tgl_lahir[0];
```

maka **ptgl** akan berisi alamat dari elemen array **tgl_lahir** yang berindeks nol. Instruksi di atas bisa juga ditulis menjadi

```
ptgl = tgl_lahir;
```

sebab nama array tanpa tanda kurung menyatakan alamat awal dari array. Sesudah penugasan seperti di atas,

```
*ptgl
```

dengan sendirinya menyatakan elemen pertama (berindeks sama dengan nol) dari array tgl_lahir.

Kesimpulan

- Tipe variabel pointer adalah tipe variabel yang berisi alamat dari variabel yang sebenarnya.
- Tipe variabel pointer harus sama dengan tipe variabel yang ditunjuk.
- Hubungan antara pointer dan array pada C sangatlah erat, sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer
- Variabel pointer bisa berupa string, array atau tipe variabel yang lainnya.

C. TUGAS PENDAHULUAN

1. . Buat program untuk menampilkan sebaris string seperti contoh berikut ;
   "Selamat Pagi" menggunakan variable pointer *(pointer to string)*.
2. Buat potongan program untuk mencetak huruf ketiga (L) dari kata :
   "P O L I T E K N I K " dengan menggunakan variabel pointer

D. PERCOBAAN

Untuk setiap program di bawah ini,

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

1.
```
#include <stdio.h>

main()
{
int x = 3, y = 4;
int *ip;

ip = &x;
y = *ip;
x = 10;
*ip = 3;

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

}
```

2.
```
#include <stdio.h>
main()
{
   int  count = 16, sum = 17, *x, *y;

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

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

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

int r, q = 8;

int go_crazy(int *, int *);

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

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

   ptr1 = &r;

   go_crazy(ptr1, ptr2);
   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;
   }
```

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

main()
{
 int var_x = 273;
int *ptr1;
int **ptr2;

ptr1 = &var_x;
ptr2 = &ptr1;
printf("Nilai var_x = *ptr1  = %d\n", *ptr1);
printf("Nilai var_x = **ptr2 = %d\n\n", **ptr2);

printf("ptr1 = &var_x = %p\n", ptr1);
printf("ptr2 = &ptr1  = %p\n", ptr2);
printf("       &ptr2  = %p\n", &ptr2);

}
```

5.
```
int array1[10], array2[10];
int *ip1, *ip2 = array2;
int *akhir = &array1[10];

for(ip1 = &array1[0]; ip1 < akhir; ip1++)
*ip2++ = *ip1;
```

E. LAPORAN RESMI

1. Kumpulkan listing program, ilustrasi alokasi memorinya beserta hasil eksekusinya