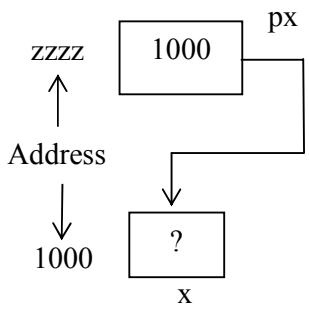
BAB VIII
POINTER

Tujuan :

1. Menjelaskan tentang konsep dari variabel pointer
2. Menjelaskan tentang pointer array
3. Menjelaskan tentang pointer string
4. Menjelaskan tentang array pointer
5. Menjelaskan tentang pointer dalam fungsi
6. Menjelaskan tentang pointer sebagai parameter fungsi
7. Menjelaskan tentang pointer yang menunjuk pointer.

8.1 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

8.2 Mendeklarasikan Variabel Pointer

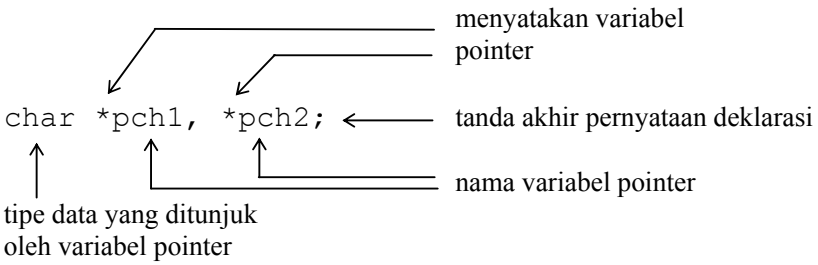
Suatu variabel pointer dideklarasikan dengan bentuk sebagai berikut :

```
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

8.3 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**.

8.4 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**. Untuk lebih jelasnya perhatikan contoh program **ptr1.c**

```
/* Program : ptr1.c */

#include <stdio.h>

main()
{
     int y, x = 87;                /* x & y bertipe int */
     int *px;
     /* var pointer yang menunjuk ke data yang bertipe int */

     px = &x;  /* px diisi dengan alamat dari variabel x */
     y = *px;  /* y diisi dengan nilai yg ditunjuk oleh 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);
}
```

Contoh eksekusi :

```
Alamat x       = 0012FF78
Isi px         = 0012FF78
Isi x          = 87
Nilai yang ditunjuk oleh px = 87
Nilai y        = 87
```

Pada program di atas, dua pernyataan

```
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. Lebih jelasnya perhatikan contoh program **ptr2.c**.

```
/* Program : ptr2.c */
```

```
#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);
}
```

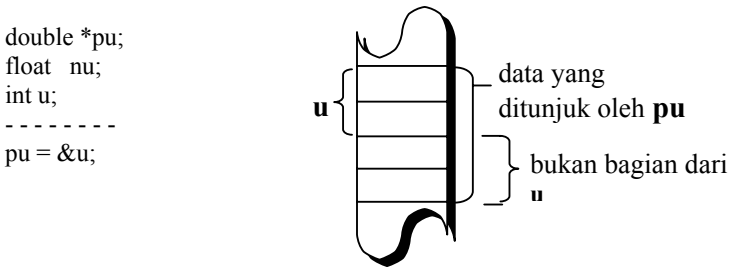
Pada contoh di atas, saat penugasan

```
pu = &u;
```

maka **pu** akan menunjuk data berukuran 4 byte (tipe *float*) sekalipun **u** berukuran 2 byte (tipe *int*). Oleh karena itu, pernyataan

```
nu = *pu;
```

tidak akan membuat **nu** berisi nilai **u**. untuk lebih jelasnya lihat gambar berikut.



Gambar 8.3 Ilustrasi kesalahan yang terjadi karena tipe tidak sejenis

8.5 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;
```

```
/* Program : ptr3.c */

#include <stdio.h>

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);
}
```

Contoh eksekusi :

```
Isi d mula-mula     = 54.5
Isi d sekarang = 64.5
```

8.5 Pointer dan Array (pointer to 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

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

dan

```
int *ptgl;
```

Kemudian diberikan instruksi

```
ptgl = &tgl_lahir[0]; //pointer to array of integer
```

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.** Hal ini bisa dilihat melalui pembuktian program berikut.

```
/* Program : ptr4.c */

#include <stdio.h>

main()
{
     static int tgl_lahir[] = {16, 4, 1974};
     int *ptgl;

     ptgl = tgl_lahir;

     printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
     printf("Nilai dari tgl_lahir[0] = %d\n", tgl_lahir[0]);
}
```

Contoh eksekusi :

```
Nilai yang ditunjuk oleh ptgl = 16
Nilai dari tgl_lahir[0]       = 16
```

```
/* Program : ptr5.c */

#include <stdio.h>

main()
{
     static int tgl_lahir[] = {16, 4, 1974};
     int *ptgl, i;

     ptgl = tgl_lahir;

     printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
     for (i=0; i<3; i++)
          printf("Nilai dari tgl_lahir[i] = %d\n",
               *(ptgl+i));
```

```
}
```

Contoh eksekusi:

```
Nilai yang ditunjuk oleh ptgl = 16
Nilai dari tgl_lahir[0]       = 16
Nilai dari tgl_lahir[1]       = 4
Nilai dari tgl_lahir[2]       = 1974
```

```
/* Program : ptr6.c */

#include <stdio.h>

main()
{
     static int tgl_lahir[] = {16, 4, 1974};
     int i;
     int *ptgl;

     ptgl = tgl_lahir;

     printf("Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
     for (i=0; i<3; i++)
          printf("Nilai dari tgl_lahir[i] = %d\n", *ptgl++);
}

Keterangan : tgl_lahir[i]  =  *(ptgl+i)  =  *ptgl++
```

8.6 Pointer dan String (pointer to string)

Contoh hubungan pointer dan string ditunjukkan pada program berikut.

```
/* Program : ptr4.c */

#include <stdio.h>

main()
{
    /* pkota menunjuk konstanta string "SEMARANG" */
    char *pkota = "SEMARANG";

    printf("String yang ditunjuk oleh pkota = ");
    puts(pkota);  // printf("%s\n", pkota);
}
```

Contoh eksekusi :

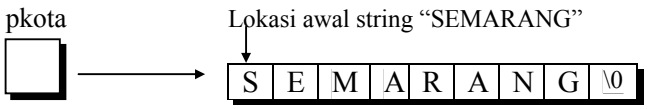
```
String yang ditunjuk oleh pkota = SEMARANG
```

Pada program di atas,

```
char *pkota = "SEMARANG";
```

akan menyebabkan kompiler

- mengalokasikan variabel **pkota** sebagai variabel pointer yang menunjuk ke obyek bertipe *char* dan menempatkan konstanta **"SEMARANG"** dalam suatu memori
- kemudian pointer **pkota** akan menunjuk ke lokasi string **"SEMARANG".**



Gambar 8.4 Pointer menunjuk data

Pernyataan di atas menyerupai pernyataan

```
char kota[] = "SEMARANG";
```

tetapi sebenarnya kedua pernyataan inisialisasi di depan tidaklah tepat sama. Sebab **pkota** adalah pointer (menyatakan alamat) yang dengan mudah dapat diatur agar menunjuk ke string lain (bukan string "SEMARANG"), sedangkan **kota** adalah array (array menyatakan alamat yang konstan, tak dapat diubah). Perhatikan dua program di bawah ini

```
/* Program : arrnama.c
 Menukarkan isi 2 string tanpa pemakaian pointer */

#include <stdio.h>
#include <string.h>

#define PANJANG 20

char nama1[PANJANG] = "JAMES BOND";
char nama2[PANJANG] = "HERCULE POIROT";

main()
{
     char namax[PANJANG];
```

```
     puts("SEMULA : ");
     printf("nama1 --> %s\n", nama1);
     printf("nama2 --> %s\n", nama2);

     strcpy(namax, nama1);
     strcpy(nama1, nama2);
     strcpy(nama2, namax);

     puts("KINI : ");
     printf("nama1 --> %s\n", nama1);
     printf("nama2 --> %s\n", nama2);
}
```

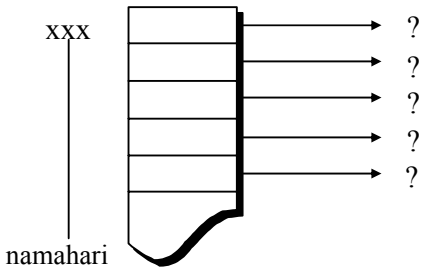
```
/* Program : ptrnama.c
 Menukarkan isi 2 string dengan fasilitas pointer */

#include <stdio.h>
#include <string.h>

char *nama1 = "JAMES BOND";
char *nama2 = "HERCULE POIROT";

main()
{
    char *namax;

    puts("SEMULA : ");
    printf("nama1 --> %s\n", nama1);
     /* nama1:pointer yg menunjuk ke string JAMES BOND */
    printf("nama2 --> %s\n", nama2);
     /* nama2:pointer yg menunjuk ke string HERCULE POIROT */

    namax = nama1;
    nama1 = nama2;
    nama2 = namax;

    puts("KINI : ");
    printf("nama1 --> %s\n", nama1);
    printf("nama2 --> %s\n", nama2);
}
```

8.7 Array dari Pointer (Array of Pointer)

- Suatu array bisa digunakan untuk menyimpan sejumlah pointer. Sebagai contoh:

```
char *namahari[10];
```

merupakan pernyataan untuk mendeklarasikan array pointer. Array **namahari** terdiri dari 10 elemen berupa pointer yang menunjuk ke data bertipe *char*.



Gambar 8.5 Array pointer

- Array pointer bisa diinisialisasi sewaktu pendeklarasian. Sebagai contoh:

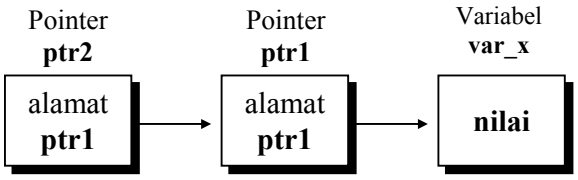
```
static char *namahari[] =
{"Senin",
"Selasa",
"Rabu",
"Kamis",
"Jumat",
"Sabtu",
"Minggu"};
```

Pada contoh ini,

namahari[0] berisi alamat yang menunjuk ke string "Senin".
namahari[1] berisi alamat yang menunjuk ke string "Selasa".
namahari[2] berisi alamat yang menunjuk ke string "Rabu".
dan sebagainya.

8.8 Pointer menunjuk Pointer (Pointer to Pointer)

Suatu pointer bisa saja menunjuk ke pointer lain. Gambar berikut memberikan contoh mengenai pointer menunjuk pointer.



Gambar 8.6 Pointer yang menunjuk pointer

- Untuk membentuk rantai pointer seperti pada gambar di atas, pendeklarasian yang diperlukan berupa

```
int var_x;
int *ptr1;
int **ptr2;
```

Perhatikan pada deklarasi di depan:

- **var_x** adalah variabel bertipe int.
- **ptr1** adalah variabel pointer yang menunjuk ke data bertipe *int.*
- **ptr2** adalah variabel pointer yang menunjuk ke pointer *int.*
  (itulah sebabnya deklarasinya berupa *int **ptr2;*)
- Agar **ptr1** menunjuk ke variabel **var_x**, perintah yang diperlukan berupa

```
ptr1 = &var_x;
```

- Sedangkan supaya **ptr2** menunjuk ke **ptr1**, instruksi yang diperlukan adalah

```
ptr2 = &ptr1;
```

- Contoh berikut memberikan gambaran cara pengaksesan nilai pada **var_x** melalui pointer **ptr2** dan **ptr1**.

```
/* File program : ppointer.c
Contoh program untuk pointer yang menunjuk pointer */

#include <stdio.h>

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

     ptr1 = &var_x;
     ptr2 = &ptr1;

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

Contoh eksekusi :

```
Nilai var_x  =  273
```

```
Nilai var_x  =  273
```

8.9 Pointer dalam Fungsi

Pointer dan kaitannya dengan fungsi yang akan dibahas berikut meliputi :

- Pointer sebagai parameter fungsi
- Pointer sebagai keluaran fungsi

8.9.1 Pointer Sebagai Parameter Fungsi

- Penerapan pointer sebagai parameter yaitu jika diinginkan agar nilai suatu variabel internal dapat diubah oleh fungsi yang dipanggil.
- Sebagai contoh dapat dilihat pada fungsi berikut.

```
void naikkan_nilai (int *x, int *y)
{
    *x = *x + 2;
    *y = *y + 2;
}
```

- Fungsi di atas dimaksudkan agar kalau dipanggil, variabel yang berkenaan dengan parameter aktual dapat diubah nilainya, masing-masing dinaikkan sebesar 2. Contoh pemanggilan :

```
naikkan_nilai(&a, &b);
```

- Perhatikan, dalam hal ini variabel **a** dan **b** harus ditulis diawali operator alamat (**&**) yang berarti menyatakan alamat variabel, sebab parameter fungsi dalam pendefinisian berupa pointer.

```
/* Program : argptr.c
Fungsi dengan argumen berupa pointer  */

#include <stdio.h>

void naikkan_nilai(int *, int *);

main()
{
     int a = 3, b = 7;

     printf("SEMULA : a = %d  b = %d\n", a, b);

     naikkan_nilai(&a, &b);
     printf("KINI   : a = %d  b = %d\n", a, b);
}

void naikkan_nilai(int *x, int *y)
{
     *x = *x + 2;
     *y = *y + 2;
}
```

Contoh eksekusi :

```
Semula : a = 3   b = 7
Kini   : a = 5   b = 9
```

8.9.2 Pointer Sebagai Keluaran Fungsi *(return value)*

- Suatu fungsi dapat dibuat agar keluarannya berupa pointer. Misalnya, suatu fungsi menghasilkan keluaran berupa pointer yang menunjuk ke string nama_bulan, seperti pada contoh berikut.

```
char *nama_bulan(int n)
{
    static char *bulan[]=
    {"Kode bulan salah", "Januari", "Februari", Maret",
          "April", "Mei", "Juni", "Juli", "Agustus",
          "September", "Oktober", "Nopember","Desember"
    };
    return ( (n<1  | |  n>12) ? bulan[0] : bulan[n] );
}
```

- Pada definisi fungsi di atas,

```
char *nama_bulan()
```

  menyatakan bahwa keluaran fungsi **nama_bulan()** berupa pointer yang menunjuk ke obyek char (atau string).

- Dalam fungsi **nama_bulan()**, mula-mula array bernama **bulan** dideklarasikan dan sekaligus diinisialisasi agar menunjuk sejumlah string yang menyatakan nama bulan. Di bagian akhir fungsi, pernyataan

```
return ( (n<1 || n>12) ? bulan[0] : bulan[n] );
```

  menyatakan bahwa hasil fungsi berupa pointer yang menunjuk ke

  → string "Kode bulan salah" (**bulan[0]**) jika masukan fungsi n<1 atau n>12

  → **bulan[n]** untuk n yang terletak antara 1 sampai dengan 12.

```
/* File program : pbulan.c
Fungsi dengan keluaran berupa pointer yang menunjuk string */

#include <stdio.h>

char *nama_bulan(int n); //prototype function

main()
{
     int bl;
     char *pch;

     printf("Masukkan Bulan 1..12 : ");
     scanf("%d", &bl);
     pch = nama_bulan(bl);
     printf("Bulan ke-%d adalah %s\n", bl, nama_bulan(bl));
}

char *nama_bulan(int n)
{
     static char *bulan[] = {
          "Kode bulan salah",
               "Januari",
               "Februari",
               "Maret",
               "April",
               "Mei",
               "Juni",
               "Juli",
               "Agustus",
               "September",
               "Oktober",
               "November",
               "Desember"
     };

     return ((n<1||n>12) ? bulan[0] : bulan[n]);
}
```

Kesimpulan

- Tipe variabel pointer adalah tipe variabel yang berisi alamat dari variabel yang sebenarnya.

- Tipe variabel pointer harus sama dengan tipe varibel yang ditunjuk.
- Hubungan antara pointer dan array pada C sangatlah erat, sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer
- Varibel pointer bisa berupa string, array atau tipe variabel yang lainnya.
- Suatu pointer bisa saja menunjuk ke pointer lain *(pointer to pointer)*
- Variabel pointer bisa digunakan sebagai parameter dalam sebuah fungsi, sebagaimana juga bisa dijadikan sebagai nilai balik *(return value)* dari sebuah fungsi.

Latihan :

Buatlah potongan program untuk soal-soal di bawah ini

1.Berapa nilai dari z dan s pada output program dibawah ini.

```
#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);
}
```

2. Bagaimana bentuk output dari program dibawah ini ?

```
#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);
}
```

3. Buat program untuk menampilkan sebaris string seperti contoh berikut ;

   "Selamat Pagi"

   menggunakan variable pointer *(pointer to string)*.

4. Buat potongan program untuk mencetak huruf ketiga (L) dari kata :

   "P O L I T E K N I K " dengan menggunakan variabel pointer .