Bab 11. Pointer 1

Konsep Pemrograman

Politeknik Elektronika Negeri Surabaya

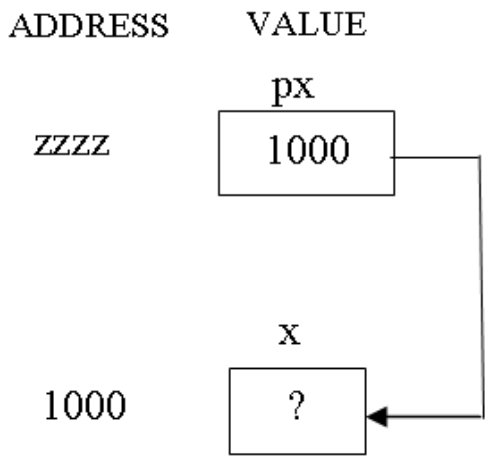
2017

Overview

- Konsep Dasar Pointer
- Deklarasi Variabel Pointer
- Mengatur Pointer agar Menunjuk ke Variabel Lain
- Akses INDIRECT melalui pointer

Konsep Dasar Pointer

- Pointer adalah variabel yang **khusus digunakan untuk menampung address.**
- Pointer sering dikatakan sebagai variabel yang menunjuk ke obyek/variabel lain.
- Kenyataan sebenarnya, variabel pointer berisi alamat dari suatu obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer).
- Misalnya:
  - **px** adalah variabel pointer
  - **x** adalah variabel yang ditunjuk oleh **px**.
  - Kalau **x** berada pada alamat memori 1000, maka **px** akan berisi 1000



Deklarasi Variabel Pointer

- Pointer dideklarasikan dengan bentuk sbb:

  tipe_data *nama_variabel;

- **tipe_data** dapat berupa sembarang tipe yang sudah dibahas pada bab-bab sebelumnya, maupun bab-bab berikutnya.
- **nama_variabel** adalah nama dari variabel pointer.
- Jika dideklarasikan :

```
int *px;
```

→nama var = `px`; tipenya = *pointer to int*

→menyatakan bahwa `px` adalah variabel pointer yang menunjuk ke suatu data tertentu yang bertipe *int*

Mengarahkan Pointer ke Variabel Lain

- Mula-mula pointer diisi dengan alamat dari variabel yang akan ditunjuk.
- Untuk menyatakan alamat dari suatu variabel, digunakan operator `&` (operator alamat, bersifat *unary*) yang ditempatkan di depan nama variabel.
- Jika dideklarasikan :

```
int *px, x = 10;
```

Maka `&x` berarti "alamat dari variabel **x**".

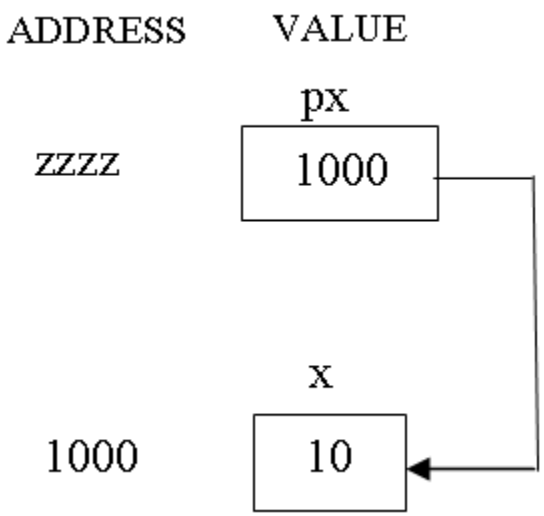
```
px = &x;
```

berarti bahwa `px` diberi nilai berupa alamat dari variabel **x**.

- Setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa **px** menunjuk ke variabel **x**.

Mengarahkan Pointer ke Variabel Lain

- Hasilnya :



- **Suatu lokasi/address yg telah ditunjuk oleh sebuah pointer, maka lokasi tsb value-nya bisa diakses baik secara DIRECT maupun INDIRECT melalui pointernya**

Akses INDIRECT melalui pointer

- Pengaksesan tak langsung dilakukan dengan menggunakan operator *indirection* (tak langsung) berupa simbol ***** (bersifat *unary*).

  ```
  *px
  ```

  menyatakan "nilai atau value dari variabel/data yang ditunjuk oleh pointer **px**" .

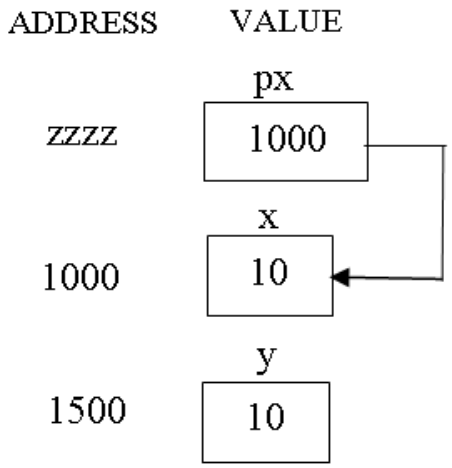
- Contoh

  ```
  int *px, x = 10, y;

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

  maka `y` akan berisi `10` yaitu

  nilai yang sama dengan nilai `x`



Akses INDIRECT melalui pointer

ATURAN PEMBACAAN

- **`&x` = alamat/address dari variabel x**
- **`*px` = nilai/value yang ada pada address/lokasi `px`**

  →Baca ulang dengan cara mengganti `px` dengan value yang ada pada `px` (dalam contoh di atas `px` berisi `1000`)

  →merupakan cara akses indirect

Keterangan:

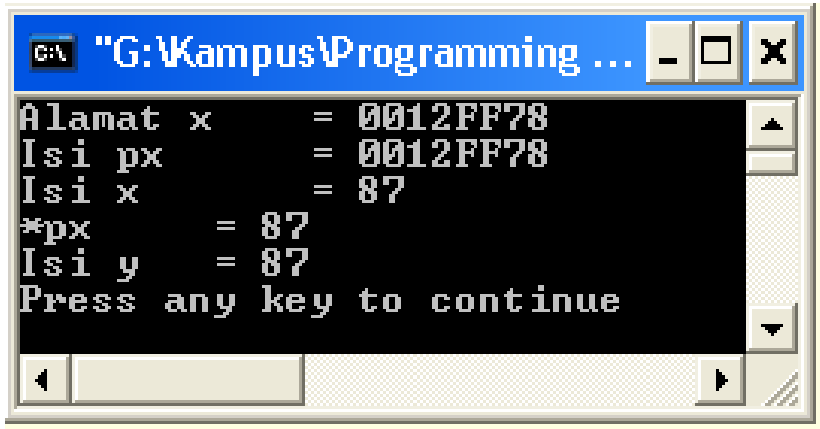
- `x` adalah nama variabel yang bertipe *int*
- `px` adalah nama variabel yang bertipe *pointer to int*

Contoh

```
#include <stdio.h>
main()
{
   int y, x = 87, *px;

   px = &x;
   y = *px;

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



Akses INDIRECT melalui pointer

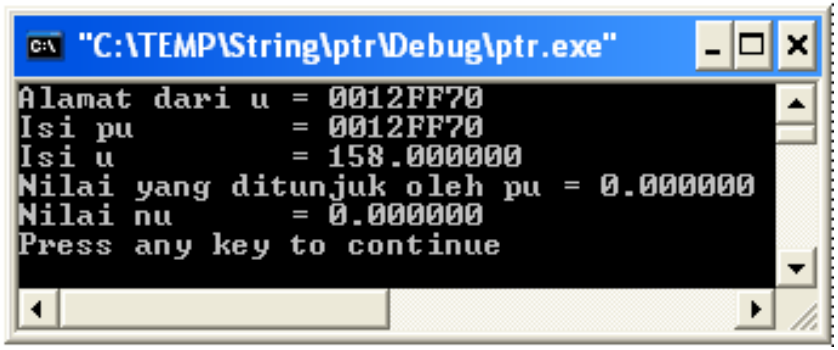
- Tipe variabel pointer dan tipe data yang ditunjuknya harus sejenis. Bila tidak sejenis, akan terjadi hasil yang tidak diinginkan

Contoh

```
main(){
   float *pu, nu;
   double u = 158.0;

   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);
}
```



Latihan

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);
   }
```

Latihan

2.
```
main(){
        int z = 20, s = 30, *pz, *ps;
        pz = &z;
        ps = &s;
        *pz += *ps;
        printf("z = %d\n", z);
        printf("s = %d\n", s);
   }
```

3.
```
main(){
        char c = 'Q';
        char *cp = &c;
        printf("%c %c\n", c, *cp);
        c = '/';
        printf("%c %c\n", c, *cp);
        *cp = '(';
        printf("%c %c\n", c, *cp);
}
```

Latihan

4.
```
main() {
    int x = 1, y = 2, *ip;
    ip = &x;
    y = *ip;
    *ip = 3;
    printf("x = %d, y = %d", x, y);
}
```

5.
```
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);
}
```

Latihan

6.
```
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 );

}
```

7.
```
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);
}
```