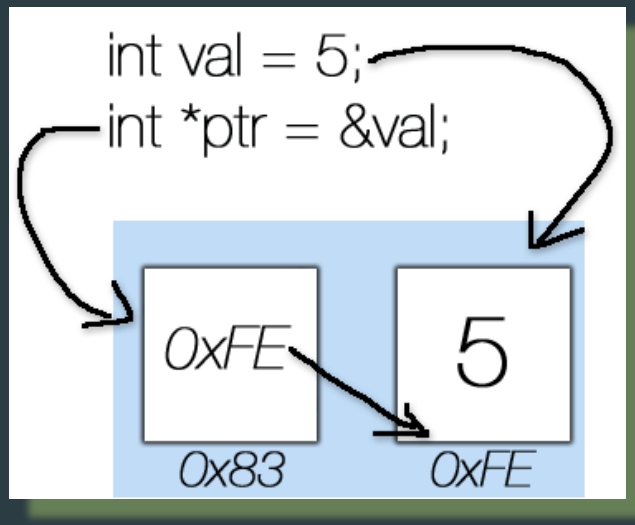
Algoritma & Struktur Data

Pointer



Modul ke : 10

Definisi Pointer

- Pointer adalah variabel yang nilainya merupakan alamat (address) memori variabel lain.
- Misal v adalah variabel yang mewakili data. Maka kompilator akan menentukan alamat sel memori untuk v yaitu &v. Jika alamat v diberikan kepada pv maka pv = &v
- pv disebut pointer ke v karena ia menunjuk ke lokasi yang ditempati oleh v.

Definisi Pointer

- Data yang diwakili oleh v diakses menggunakan operator * yang ditulis di depan pv sehingga *pv dan v mewakili data yang sama.
- Isi yang ditunjuk oleh pv adalah *pv (= v).
- * Disebut content of operator. Ia juga disebut sebagai indirection operator karena dapat digunakan untuk mengisi nilai suatu variabel secara tidak langsung yaitu melalui pointer.
- & Disebut address of operator

Manfaat Pointer

- Mengirim kembali lebih dari satu nilai.
- Mengolah *array* dengan cara yang lebih efisien.
- Membentuk tipe data yang lebih kompleks (linked-list, tree, graph).
- Menghasilkan informasi tentang memori.

Deklarasi & Inisialisasi Pointer

- Sintaks :

  <DATA_TYPE> *PTR_NAME ;

- Contoh :

  INT *PTR;

- Pointer dapat diberi nilai awal alamat variabel lain. Ia juga dapat diberi nilai 0, disebut null pointer yang biasa digunakan untuk mengakhiri linked-list.

  #DEFINE NULL 0

  …

  INT U, V;

  INT *PV = &V, *PU = NULL;

Cara Penulisan Pointer

- int *p; Old C
- int * p; Clarity
- int* p; Has A Type *int

Operator Dan Deskripsi Pointer

- Simbol operator
  - Content of operator *
  - address of operator &

- Deskripsi
  - Simbolik pv $\Rightarrow$ v
  - Logikal pv menunjuk v
  - Aktual pv berisi alamat v, yaitu pv = &v

Contoh 1

```
#include <conio.h>
#include <iostream.h>
Void main() {
   int u1, u2, v = 3;
   int *pv;
   u1 = 2 * (v + 5);
   pv = &v;
   u2 = 2 * (*pv + 5);
   cout << “u1 = “ << u1 << “, u2 = “ << u2 << “\n”;
   cout << “pv = “ << pv << “\n”;
   getch();
}
```

u1=16, u2=16

Contoh 2


```
#include <conio.h>
#include <iostream.h>
Void main() {
    int v = 3;
    int *pv = &v;
    cout << “*pv = “ << *pv << “, v = “ << v << “\n”;
    *pv = 0;
    cout << “*pv = “ << *pv << “, v = “ << v << “\n”;
    getch();
}
```

*pv = 3, v = 3

*pv = 0, v = 0

Contoh 3

```
#include <conio.h>
#include <iostream.h>
void main() {
    int u = 3, v;
    int *pu, *pv;
    pu = &u;
    v = *pu;
    pv = &v;
    cout << “u=“<< u <<“,&u=“<< &u <<“,pu=“ <<pu<< “,*pu=”<<*pu;
    cout << “v=“<< v <<“,&v=“<< &v <<“,pv=“ <<pv<< “,*pv=”<<*pv;
    getch();
}
```

Operasi Pada Pointer (1)

- Operasi hitung tidak berlaku di pointer kecuali ‘-’ yang ditafsirkan sebagai jarak yang memisahkan kedua alamat. Tidak ada maknanya alamat di tambah alamat.
- Pointer bisa ditambah atau dikurangi dengan nilai integer tertentu. Misalkan ptr=ptr+1 ditafsirkan sebagai geser pointer satu posisi ke depan.

Operasi Pada Pointer (2)

- Pointer Bisa Di-*assign* Ke Pointer Lain.
  Misalnya, ptr1=ptr2, Ditafsirkan sebagai berikan Ptr2 Ke Ptr1, Sehingga kedua Pointer sekarang menunjuk Ke alamat yang sama yaitu alamat yang sebelumnya ditunjuk oleh ptr2.
- Pointer bisa dibandingkan dengan Pointer lain.
  Sebagai contoh, If(ptr1 < Ptr2) ...

Pointer ke Pointer

- char c = ‘x’;
- char* pc = &c;
- char** ppc = &pc;
- char*** pppc = &ppc;
- ***pppc = ‘X’;
- 

Transfer Parameter

- By value:
  Yang dikirim ke fungsi lain adalah nilai data (r-value).
- By location / address:
  Yang dikirim ke fungsi lain adalah alamat (l-value).

Contoh 4: By Value

```
void garis(char x) {
   int i;
   for(i=1; i<=10; i++)
    cout << x;
}

Void main() {
   char c = '-';
   garis( c );
}
```

Contoh 5: By Location

```
void hitung(int x, int y, int *p, int *q) {
   *p = x + y;
   *q = x * y;
}

void main() {
   int x = 10, y = 5, p, q;
   hitung(x, y, &p, &q);
   printf("x + y = %d, x * y = %d", p, q);
}
```

Contoh 6: By Value/Location?

```
void tukar(char a, char b) {
   char c;
   c = a; a = b; b = c;
}

void main() {
   char x = 'X', y = 'Y';
   tukar(x, y);
   cout << "X = " << x << ", Y =" << y;
}
```

Pertukaran data yang diharapkan antara x dan y tidak terjadi. Mengapa?

Contoh 7: By Value /Location

```
#include<stdio.h>
void f1(int u, int v) {
    u=0; v=0;
    cout << “Di dalam f1: u = “ << u << “ v = ” << v << “\n”;
    return 0;
}

void f2(int *pu, int *pv) {
    *pu=0; *pv=0;
    cout << “Di dalam f2: *pu = “ << *pu “, *pv = “ << *pv << “\n”;
    return 0;
}
```

Besambung ke halaman selanjutnya …

Contoh 7: By Value/Location

```
void main() {
    int u=1, v=3;
    cout << “\nSebelum panggil f1: u = “ << u << “,v =” << v;
    f1(u,v);
    cout << “\nSesudah panggil f1: u = “ << u << “,v =” << v;
    cout << “\nSebelum panggil f2: u = “ << u << “,v =” << v;
    f2(&u,&v);
    cout << “\nSesudah panggil f2: u = “ << u << “,v =” << v;
    getch();
}
```

Reference Variabel

- Only on c++
- Variabel reference adalah alias atau sinonim variabel lain. Jika variabel lain diubah maka variabel reference juga berubah, dan sebaliknya.
- Ia dibentuk dengan menambahkan operator & di *type specifier*.
- Ia harus diinisialisasi.
- Contoh:
  - Int n = 33;
  - Int& r = n; //r adalah alias dari n

Fungsi Swap()

- Fungsi swap() berikut akan banyak anda temui di sort data.

```
Void swap(int& x, int& y) {
   int temp;
   X = temp;
   X = y;
   Y = temp;
}
```

Call By Value Vs Call By Reference(1)

Call By Value

- int x;
- Parameter formal x adalah variabel lokal.
- Ia adalah duplikasi parameter aktual.
- Ia tidak dapat mengubah parameter aktual.
- Parameter aktual mungkin konstan, variabel atau ekspresi.
- Parameter aktual adalah read only.

Call By Reference

- int &x;
- Parameter formal adalah referensi lokal.
- Ia adalah sinonim parameter aktual.
- Ia dapat mengubah parameter aktual.
- Parameter aktual harus variabel.
- Parameter aktual adalah read-write.

Call By Value Vs Call By Reference (2)

```
int sqByVa(int a) {
  return a*a;
}

void sqByRe(int &yRe) {
  yRe *= yRe;
}
```

Asumsi: prototipe fungsi int sqByVa(int) Dan void sqByRe(int&) ada di atas MAIN()

```
int main() {
  int x=2, y=4;
  printf("Sebelum sqByVa x:%d",x);
  printf("Nilai balik sqByVa :  %d",sqByVa(x));
  printf("Sesudah sqByVa x:%d",x);
  printf("Sebelum sqByRe y:%d",y);
  sqByRe(y);
  printf("Sesudah sqByRe y:%d",y);
  return 0;
}
```

Pointer Sebagai Parameter

```
void swap( int &a, int & b) {
   int temp;
   temp = a;
   a = b;
   b = temp;
}

int main() {
   int x=4, y=8;
   swap(x,y);
   printf("%d  %d",x,y);
   return 0;
}
```

```
void swap( int * a, int * b) {
   int temp;
   temp = *a;
   *a = *b;
   *b = temp;
}

int main() {
   int x=4, y=8;
   swap(&x,&y);
   printf("%d  %d",x,y);
   return 0;
}
```

Pointer Sebagai Return Value

```
int *fungsi(int *p) {
   int i, im , m = 0;
   for(i=0; i<5; i++)
        if (*(p+i)>m) {
            m = *(p+i);
            im = i;
     }
   return(p+im);
}
```

```
main(){
  int a[] = {20, 40, 10, 50,
    30};
  int *ptr;
  ptr =  fungsi(a);
  cout << " m = " << *ptr;
}
```

APA RETURN VALUE DARI *fungsi()?

THE END