

PRAKTIKUM 3

POINTER

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Stuktur Program menggunakan bahasa C++

B. Peralatan

1. PC Desktop
2. Windows 7
3. Notepad++ dan MinGW atau Dev++

C. Teori

Pointer merupakan tipe data primitif seperti umumnya. Bila variabel pada umumnya menyimpan data tertentu maka pointer hanya dapat menyimpan alamat memori. Dengan kata lain pointer adalah sebuah objek dalam memori yang menunjuk ke sebuah nilai yang tersimpan di memori berdasarkan alamat nilai tersebut.

Operator pointer yang disediakan c++ yaitu :

- a. Operator Dereference (&) -> ampersand
- b. Operator Reference (*) -> asterix

Array sebagai Pointer

Secara internal array juga menyatakan alamat, dimana pengenalan array sama dengan alamat pada elemen pertama pada array. Ketika array dibuat, compiler secara otomatis membuat sebuah internal pointer constant untuk itu dan menyimpan alamat dari array didalam pointer. Nama dari array sebenarnya sudah menyatakan konstanta pointer, sehingga operator & tidak diperlukan.

Inisialisasi :

```
int x[] {8, 2, 9, 4, 6, 3, 1};  
int* p = x;           // p menunjuk x[0]  
p++;                  // p menunjuk x[1]
```

Pengaksesan :

```
cout << *p << '\n';    // 2  
cout << *(p + 2);      // 4
```

Pengaksesan dapat dilakukan satu persatu langsung merujuk pada indeks atau serentak. Contoh pengaksesan secara serentak pada array 2 dimensi :

```
for(auto i = 0; i < 7; ++i)
    cout << *p++ << ' ';
```

Fungsi dan Pointer

Secara default ketika ada sebuah nilai yang dilewatkan (sebagai parameter atau argumen) ke fungsi maka yang terjadi adalah proses penyalinan (copy) dari nilai asli. Dengan kata lain jika terjadi manipulasi terhadap nilai yang masuk tersebut tidak akan mempengaruhi nilai aslinya.

Sebagai ilustrasi berikut ini adalah kode untuk fungsi swap() yang digunakan untuk menukar 2 nilai yang dilewatkan

Percobaan 1:

```
void swap(int x, int y)
{
    int tmp = x;
    x = y;
    y = tmp;
}

int main()
{
    int a{3};
    int b{4};
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    swap(a, b);
    cout << "setelah di swap" << endl;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
}
```

Di main() terdapat 2 variabel yaitu a dengan nilai 3 dan b dengan nilai 4. Saat terjadi pemanggilan terhadap fungsi swap() dengan a dan b sebagai argumennya akan terjadi penyalinan nilai a ke x dan b ke y. Proses penukaran terjadi di dalam fungsi swap() melibatkan variabel lokalnya (x, y, dan tmp). Ketika pemanggilan selesai dan proses dikembalikan ke pemanggil (fungsi main()) maka seluruh variabel lokal di fungsi swap() akan dihapus.

Berdasar kondisi tersebut dapat dikatakan bahwa proses penukaran justru terjadi terhadap x dan y sebagai variabel lokal swap(), bukan pada a dan b sebagai nilai asli yang hendak ditukar.

Pelewatan argumen dengan menyalin nilai asli ke dalam fungsi disebut **passing argument by value** atau sering disingkat **pass by value**. Cara lain untuk melewati argumen ke fungsi adalah dengan **pass by pointer**. Cara ini 'hanya'

melewatkan alamat nilai asli kedalam fungsi. Berikut adalah contoh penggunaan pointer pada *pass by value* dalam fungsi swap() :

Percobaan 2 :

```
#include <iostream>
using namespace std;
void swap(int* x, int* y)
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}
int main()
{
    int a{3};
    int b{4};
    swap(&a, &b);
}
```

Argumen formal dalam fungsi swap() (x dan y) yang merupakan pointer yang menunjuk ke alamat yang dilewatkan. Saat terjadi pemanggilan fungsi swap() dengan argumen berupa alamat dari variabel a dan b maka x akan menjadi 'handle' untuk a dan y untuk b. Proses penukaran dalam fungsi swap() benar – benar terjadi pada a dan b.

Referensi

Berguna untuk membuat variable yang menjadi argument fungsi dapat diganti di dalam suatu fungsi. Sebuah nilai akan dialokasikan pada alamat memori tertentu. Setiap alamat dapat direferensi oleh pointer. Berbeda dari bahasa c di c++ terdapat cara lain untuk melakukan referensi selain menggunakan pointer yaitu dengan reference. Berbeda dengan pointer yang bisa menunjuk ke alamat yang berbeda – beda maka reference hanya bisa ke satu alamat saja.

Sekali reference diinisiasi terhadap alamat tertentu maka selama daur hidup tidak boleh berpindah ke alamat yang lain. Selain itu pengoperasian reference lebih natural, seperti halnya variabel biasanya; tidak seperti pointer yang menggunakan notasi asterik untuk menunjuk ke nilai yang direferensi. Potongan kode berikut ini adalah contoh pendeklarasian reference:

```
Int x{8};
Int& r = x;
```

Deklarasi di atas menjelaskan bahwa r (ditandai dengan ampersand) merupakan reference ke x. Perbedaan mendasar antara pointer dengan reference adalah pada lokasinya. Pada pointer alamat antara pointer itu sendiri dengan alamat yang direferensi berbeda. Reference memiliki alamat yang sama dengan alamat yang direferensinya sehingga dapat dikatakan bahwa reference adalah alias dari sebuah

alamat. Untuk membuktikan perhatikan kode berikut yang merupakan lanjutan deklarasi sebelumnya:

Percobaan 3 :

```
Void swap(int& x, int& y)
{
    int tmp = x;
    x = y;
    y = tmp;
}

Int main()
{
    int a{3};
    int b{4};
    swap(a, b);
}
```

Ukuran data

Ukuran sebuah variabel dapat dihitung menggunakan operator sizeof. Besarnya ukuran tergantung dari besarnya tipe data yang digunakan. Sebagai contoh ukuran tipe int dapat dihitung dengan sizeof(int). Berikut ini adalah contoh potongan kode untuk menampilkan beberapa ukuran tipe data menggunakan operator sizeof.

Percobaan 4 :

```
Cout << "ukuran int : " << sizeof(int) << endl;
Cout << "ukuran char : " << sizeof(char) << endl;
Cout << "ukuran float : " << sizeof(float) << endl;
Cout << "ukuran double : " << sizeof(double) << endl;
```

Satuan ukuran data untuk potongan kode diatas adalah byte. Sebuah array berukuran 5 elemen bertipe float akan memiliki ukuran sebesar 20 byte (4 byte x 5). Potongan kode berikut ini adalah contoh lain untuk perhitungan ukuran array :

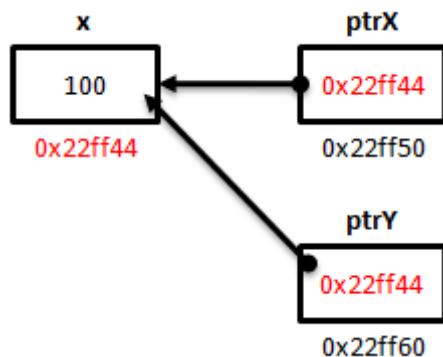
```
Float a[5];
Cout << "ukuran array a : " << sizeof(a) << endl;
```

D. Praktikum

Source Code #1 – Basic pointer

Pointer adalah variabel yang digunakan untuk menunjuk alamat memory variabel lain. Jadi isi dari pointer adalah alamat memory bukan nilai yang sebenarnya.

Berdasarkan ilustrasi berikut, tuliskan program lengkap yang mengimplementasikan penggunaan pointer.



Simpan dengan nama **pointer-1-xxxx** dan untuk melihat hasilnya klik menu **Execute -> Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output

:

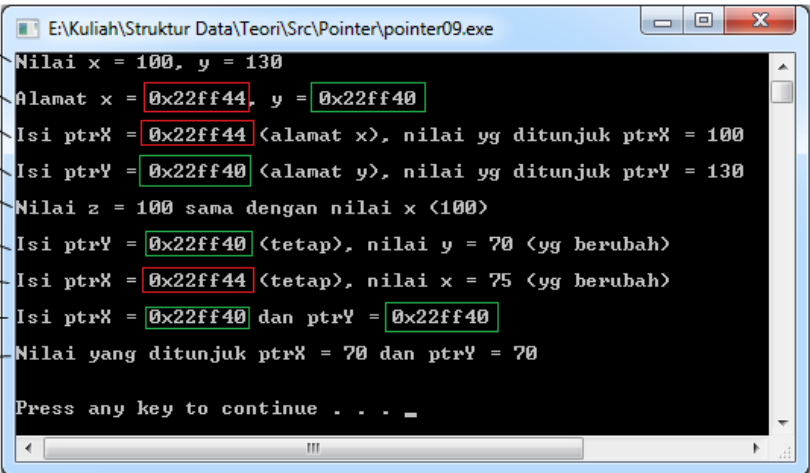
```
E:\Kuliah\Struktur Data\Teori\Src\Pointer\pointer08.exe
Isi variabel x = 100 ada di alamat 0x22ff44
Isi variabel ptrX = 0x22ff44 sama dengan alamat x <0x22ff44>
Isi variabel ptrY = 0x22ff44 sama dengan alamat x <0x22ff44>
Nilai yg ditunjuk ptrX = 100 sama dengan nilai x <100>
Nilai yg ditunjuk ptrY = 100 sama dengan nilai x <100>
Press any key to continue . . .
```

Source Code #2 – Operasi penugasan/assignment pada pointer

Lengkapi baris perintah yang ada komentar `// TODO :`

```
5 int main()
6 {
7     int x = 100, y = 130, z; // deklarasi var non pointer
8
9     int *ptrX, *ptrY; // deklarasi var pointer
10
11     // TODO : tampilkan nilai x dan y
12
13     // TODO : tampilkan alamat x dan y
14
15     ptrX = &x; // ptrX menunjuk alamat x
16     // TODO : tampilkan nilai ptrX dan *ptrX
17
18     ptrY = &y; // ptrY menunjuk alamat y
19     // TODO : tampilkan nilai ptrY dan *ptrY
20
21     z = *ptrX; // z berisi nilai yg ditunjuk ptrX
22     // TODO : tampilkan nilai z dan x
23
24     *ptrY = 70; // merubah nilai yg ditunjuk ptrY
25     // TODO : tampilkan nilai ptrY dan y
26
27     *ptrX = *ptrY + 5; // merubah nilai yg ditunjuk ptrX
28     // TODO : tampilkan nilai ptrX dan x
29
30     ptrX = ptrY; // ptrX menunjuk apa yg ditunjuk ptrY
31     // TODO : tampilkan nilai ptrX dan ptrY
32     // TODO : tampilkan nilai *ptrX dan *ptrY
33
34     cout << endl;
35     system("pause");
36     return 0;
37 }
```

Simpan dengan nama **pointer-2-xxxx** dan untuk melihat hasilnya klik menu **Execute -> Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output :



The screenshot shows a Windows command prompt window titled "E:\Kuliah\Struktur Data\Teori\Src\Pointer\pointer09.exe". The output of the program is displayed, with lines of output corresponding to the line numbers 11, 13, 16, 19, 22, 25, 28, 31, and 32 of the source code. The output shows the values of variables x, y, z, and the pointers ptrX and ptrY, along with the memory addresses they point to. The program ends with a "Press any key to continue" prompt.

```
11 Nilai x = 100, y = 130
13 Alamat x = 0x22ff44, y = 0x22ff40
16 Isi ptrX = 0x22ff44 (alamat x), nilai yg ditunjuk ptrX = 100
19 Isi ptrY = 0x22ff40 (alamat y), nilai yg ditunjuk ptrY = 130
22 Nilai z = 100 sama dengan nilai x (100)
25 Isi ptrY = 0x22ff40 (tetap), nilai y = 70 (yg berubah)
28 Isi ptrX = 0x22ff44 (tetap), nilai x = 75 (yg berubah)
31 Isi ptrX = 0x22ff40 dan ptrY = 0x22ff40
32 Nilai yang ditunjuk ptrX = 70 dan ptrY = 70

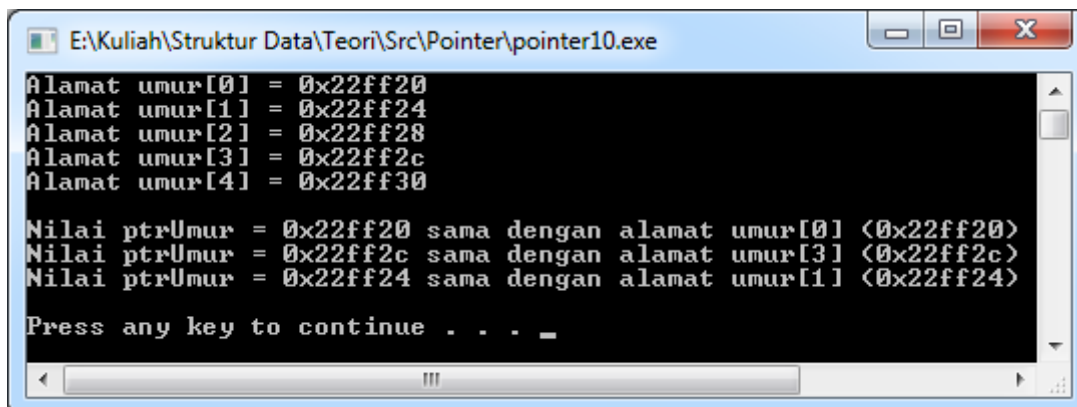
Press any key to continue . . .
```

Source Code #3 – Operasi Aritmatika pada pointer 1

Operasi aritmatika pada variabel pointer berbeda dengan variabel biasa. Operasi penambahan dengan suatu nilai menunjukkan lokasi data berikutnya (index selanjutnya) dalam memori, begitu juga operasi pengurangan menunjukkan lokasi data sebelumnya.

```
5 int main()
6 {
7     int umur[] = { 21, 22, 23, 24, 25 };
8
9     int *ptrUmur = umur; // menunjuk element pertama dari array umur
10
11     cout << "Alamat umur[0] = " << &umur[0] << endl;
12     cout << "Alamat umur[1] = " << &umur[1] << endl;
13     cout << "Alamat umur[2] = " << &umur[2] << endl;
14     cout << "Alamat umur[3] = " << &umur[3] << endl;
15     cout << "Alamat umur[4] = " << &umur[4] << endl << endl;
16
17     // nilai ptrUmur sebelum operasi penambahan dan pengurangan
18     cout << "Nilai ptrUmur = " << ptrUmur << " sama dengan alamat umur[0] (" << &umur[0] << ")" << endl;
19
20     // operasi penambahan
21     ptrUmur += 3;
22     cout << "Nilai ptrUmur = " << ptrUmur << " sama dengan alamat umur[3] (" << &umur[3] << ")" << endl;
23
24     // operasi pengurangan
25     ptrUmur -= 2;
26     cout << "Nilai ptrUmur = " << ptrUmur << " sama dengan alamat umur[1] (" << &umur[1] << ")" << endl;
27
28     cout << endl;
29     system("pause");
30     return 0;
31 }
```

Simpan dengan nama **pointer-3-xxxx** dan untuk melihat hasilnya klik menu **Execute -> Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output :



```
E:\Kuliah\Struktur Data\Teori\Src\Pointer\pointer10.exe
Alamat umur[0] = 0x22ff20
Alamat umur[1] = 0x22ff24
Alamat umur[2] = 0x22ff28
Alamat umur[3] = 0x22ff2c
Alamat umur[4] = 0x22ff30

Nilai ptrUmur = 0x22ff20 sama dengan alamat umur[0] (0x22ff20)
Nilai ptrUmur = 0x22ff2c sama dengan alamat umur[3] (0x22ff2c)
Nilai ptrUmur = 0x22ff24 sama dengan alamat umur[1] (0x22ff24)

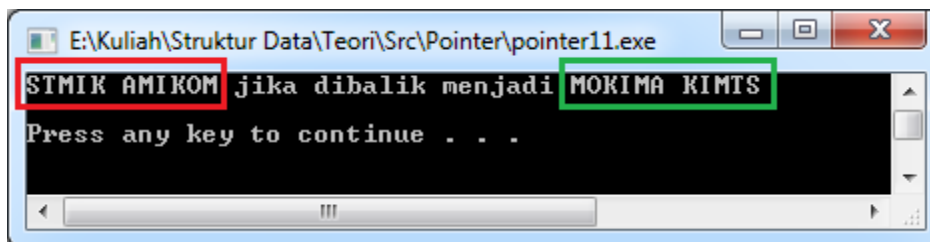
Press any key to continue . . .
```

Source Code #4 - Operasi Aritmatika pada pointer 2

Program pembalik huruf dengan memanfaatkan variabel pointer dan operasi pengurangan. Silahkan dilengkapi baris perintah yang ada komentar *// TODO :*

```
5 int main()
6 {
7     char kampus[] = "STMIK AMIKOM";
8
9     // hitung jumlah element array kampus
10    int length = sizeof(kampus) / sizeof(char);
11
12    // tampilkan nilai array kampus
13    for (int i = 0; i < length; i++)
14    {
15        cout << kampus[i];
16    }
17    cout << "jika dibalik menjadi";
18
19    // TODO : tampilkan nilai kampus secara terbalik menggunakan pointer
20
21    cout << endl << endl;
22    system("pause");
23    return 0;
24 }
```

Simpan dengan nama **pointer-4-xxxx** dan untuk melihat hasilnya klik menu **Execute -> Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output :



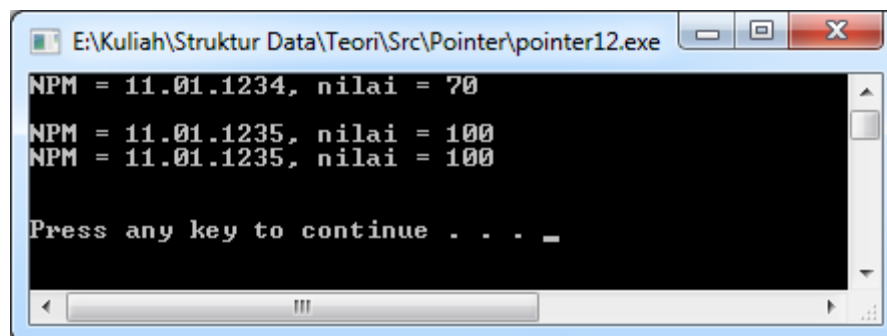
Source Code #5 – Pointer pada Struct

Pointer selain dapat diterapkan pada tipe data primitif dan array juga dapat diterapkan pada struct. Intinya pointer dapat bertipe apa saja yang didukung oleh bahasa c/c++.

```
5 struct Mahasiswa
6 {
7     char npm[11];
8     int nilai;
9 };
10
11 int main()
12 {
13     Mahasiswa mhs; // deklarasi var non-pointer dg tipe Mahasiswa
14
15     // mengakses elemen struct untuk var non-pointer
16     // menggunakan notasi titik
17     strcpy(mhs.npm, "11.01.1234");
18     mhs.nilai = 70;
19
20     cout << "NPM = " << mhs.npm << ", nilai = " << mhs.nilai << endl << endl;
21
22     Mahasiswa *ptrMhs; // deklarasi var pointer dg tipe Mahasiswa
23     ptrMhs = &mhs; // ptrMhs menunjuk alamat mhs
24
25     // mengakses elemen struct untuk var pointer
26     // menggunakan notasi tanda panah (->)
27     strcpy(ptrMhs->npm, "11.01.1235");
28     ptrMhs->nilai = 100;
29
30     cout << "NPM = " << mhs.npm << ", nilai = " << mhs.nilai << endl;
31     cout << "NPM = " << ptrMhs->npm << ", nilai = " << ptrMhs->nilai << endl;
32
33     cout << endl << endl;
34     system("pause");
35     return 0;
36 }
```

Simpan dengan nama **pointer-5-xxxx** dan untuk melihat hasilnya klik menu **Execute -> Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output

:



```
E:\Kuliah\Struktur Data\Teori\Src\Pointer\pointer12.exe
NPM = 11.01.1234, nilai = 70
NPM = 11.01.1235, nilai = 100
NPM = 11.01.1235, nilai = 100
Press any key to continue . . . _
```

E. Latihan

Note: kerjakan semua latihan diatas baru anda mengerjakan tugas dibawah ini:

1. Buatlah sebuah program yang didalamnya terdapat function untuk menghitung jumlah kata dari string yang diberikan dengan fasilitas pointer, dengan program bersifat aktif atau adanya input.
2. Buatlah program yang didalamnya terdapat function untuk membuat setiap huruf diberikan dengan fasilitas pointer dari string yang di inputkan, huruf pertama menjadi huruf besar, huruf besar semua dan huruf kecil semua.
3. Buatlah program yang bisa menyimpan nilai yang dimasukkan oleh user, bisa mengurutkan nilai tersebut secara descending, dan coba temukan nilai yang dimasukan oleh user dengan fasilitas pointer.