FAKULTAS
TEKNOLOGI
INFORMASI
Universitas Advent Indonesia

Lecture 03 – Pointers

Friday, September 8, 2023

Data Structure

Passing and Returning Structure

- Variabel struktur dapat diteruskan (passing) ke fungsi dengan cara yang sama seperti argumen normal.
- Structure juga dapat dijadikan sebagai tipe kembalian dari sebuah fungsi.
- *Lihat contoh program…*

Pointers

- Pointer merupakan salah satu fitur dari C/C++ yang membedakannya dari Bahasa pemrograman lain.
- Pointer adalah sebuah variabel yang nilainya adalah **alamat** dari variabel yang lain.
- Bagaimana data disimpan di dalam komputer?

How data is stored in our computer?

- Setiap variabel yang diciptakan dalam program kita akan diberikan lokasi dalam memori komputer. Nilai yang diberikan pada variabel tersebut akan disimpan dilokasi yang telah di tetapkan.

- Untuk mengetahui dimana data tersebut disimpan, C/C++ memiliki sebuah operator referensi (*reference operator*) yang disimbolkan dengan **&**.

Reference Operator (&)

- Operator referensi ini berperan sebagai "referensi" untuk variabel lain. Referensi dalam ini adalah alamat yang ditempati oleh suatu variabel.
- Penggunaan:

```
string makanan = "Nasi Goreng";
string &sarapan = makanan;

cout << makanan << "\n"; //output = Nasi Goreng
cout << sarapan << "\n"; //output = Nasi Goreng
```

Reference Operator (&)

- Operator referensi ini berperan sebagai **"referensi" untuk variabel lain**, begitu juga memberikan **alamat** yang ditempati oleh suatu variabel.
- **Penggunaan:**

```
string makanan = "Nasi Goreng";

cout << &makanan; //output = alamat memori dalam hexadecimal
```

Why is it useful to know the memory address?

- Referensi (*reference*) dan Pointers penting dalam C/C++ karena memberikan kemampuan bagi programmer untuk memanipulasi data dalam memori komputer sehingga dapat mengurangi kode dan meningkatkan kinerja.

Pointers

Variabel Pointer

- C++ memungkinkan programmer untuk memanipulasi data yang ada dalam memori komputer secara langsung. Hal ini dilakukan menggunakan variabel pointer.
- Variabel pointer adalah variabel yang menunjuk (*points*) ke sebuah alamat spesifik dalam memori yang ditunjukkan oleh variabel lain.
- Sebuah variabel pointer menunjuk ke tipe data (seperti int atau string) dari tipe yang sama, dan dibuat dengan menggunakan operator *.

Pointers

Deklarasi pointer

```
<datatype>  *var_name;  atau  <datatype>*  var_name
```

Contoh:

```
int *ptr;  atau  int* ptr;
```

Pointers

Contoh:

```
int *ptr;  atau  int* ptr;
```

- Contoh di atas mendefinisikan sebuah variabel pointer `ptr` yang menyimpan alamat memori.
- Simbol bintang (*asterisk*) adalah operator deferensi yang berarti **penunjuk ke (*pointer to*)**.
- Dalam contoh ini, pointer `ptr` adalah penunjuk ke `int`, dengan kata lain menunjuk ke suatu nilai integer di alamat memori.

Operator Dereferensi (*)

- Operator referensi (**&**) memberikan alamat dari suatu variabel. Untuk memperoleh nilai yang disimpan di dalam alamat memori tersebut, dapat menggunakan operator dereferensi (*).

Contoh:

```
int bilangan1 = 5;
```

Jika variabel bilangan disimpan dalam alamat memori 0x123, maka operator referensi (**&**) akan memberikan informasi mengenai **alamat memori** yakni **0x123** sementara operator dereferensi (*)memberikan informasi mengenai **nilai yang disimpan** didalam alamat tersebut yakni **5**.

Don’t be confused

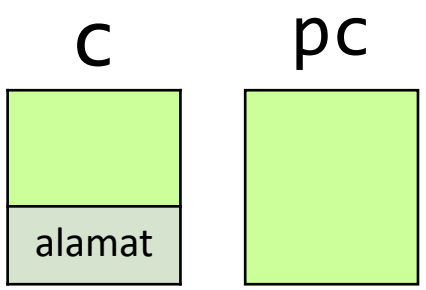
Perhatikan bahwa tanda * dapat membingungkan di sini, karena ia melakukan dua hal berbeda dalam kode program:

1. Ketika digunakan dalam deklarasi (string *ptr), berarti kita menciptakan variabel pointer.
2. Ketika tidak digunakan dalam deklarasi, ia bertindak sebagai operator dereferensi.

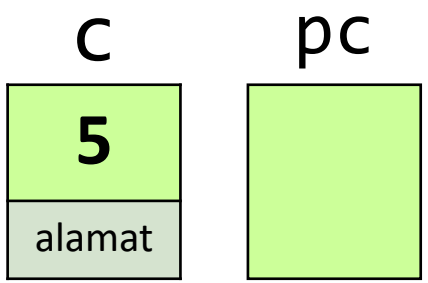
ContohProgramPointer - Explanation

```
int *pc, c;
```

alamat = 0x23fe34



```
c = 5;
```




```
cout << "Alamat dari c (&c): " << &c << endl;
cout << "Nilai dari c (c): " << c << endl << endl;
```

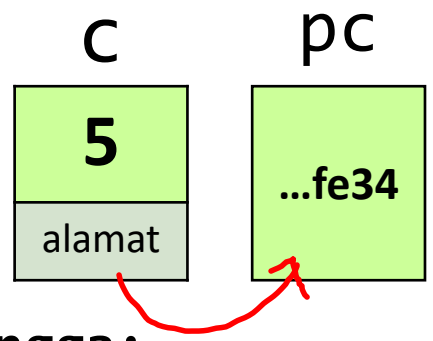
Output yg dihasilkan:

```
Alamat dari c (&c): 0x23fe34
Nilai dari c (c): 5
```

ContohProgramPointer - Explanation

```
pc = &c;
```

```
pc = 0x23fe34
*pc = 5
```



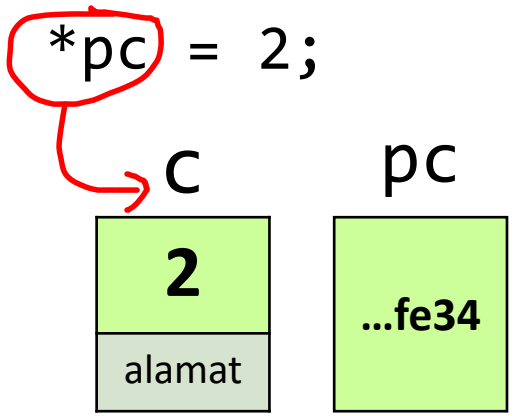
Sehingga:

```
cout << "Alamat yang dipegang oleh pointer pc (pc): "<< pc << endl;
cout << "Isi dari alamat yang dipegang oleh pointer pc (*pc): " << *pc << endl << endl;
```

Output:

```
Alamat yang dipegang oleh pointer pc (pc): 0x23fe34
Isi dari alamat yang dipegang oleh pointer pc (*pc): 5
```

ContohProgramPointer - Explanation




```
&c = 0x23fe34
c = 2
```

Sehingga:

```
cout << "Alamat dari (&c): " << &c << endl;
cout << "Nilai dari (c): " << c << endl << endl;
```

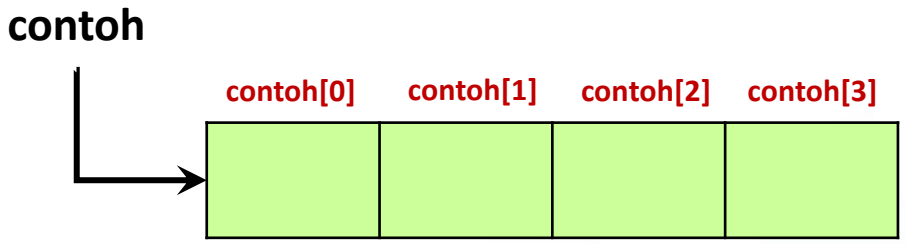
Output:

```
Alamat dari (&c): 0x23fe34
Nilai dari (c): 2
```

Pointers and Array

```
int contoh[4] = {8, 2, 1, 6};
```

- Ketika array dideklarasikan pertama kali, secara implisit array akan dikonversi menjadi pointer yang menunjuk ke elemen pertama array.



- Dengan kata lain, nama arraynya sendiri sudah merupakan alamat dari dari array yang telah dideklarasikan.

Pointers and Array

- Saat menetapkan alamat array ke pointer jangan menggunakan operator referensi (&).

```
int contoh[4] = {8, 2, 1, 6};
int *ptrContoh;
ptrContoh = contoh; // bukan ptrContoh = &contoh;
```

Pointers and Array

- Namun, untuk menetapkan suatu pointer untuk menunjuk ke salah satu elemen array dilakukan dengan menggunakan operator referensi (&) diikuti dengan indeks array. Contoh:

```
int contoh[4] = {8, 2, 1, 6};
int *ptrContoh;
ptrContoh = &contoh[2];
```

- Lalu, bagaimana jika kita ingin pointer `ptrContoh` menunjuk ke elemen yang keempat?

Pointers and Array

```
ptrContoh = &contoh[2];
ptrContoh = ptrContoh + 1;
```

Oleh karena sebelumnya `ptrContoh` menunjuk ke elemen yang ke tiga, makan `ptrContoh + 1` akan menunjuk ke elemen yang ke empat. Dengan kata lain, pointer `ptrContoh` tidak menunjuk ke byte berikutnya dari `ptrContoh`.

Pointers and Array

Sehingga:

```
int contoh[4] = {8, 2, 1, 6};
int *ptrContoh;
ptrContoh = contoh;
```

Maka:

&contoh[0] == ptrContoh dan contoh[0] == *ptrContoh

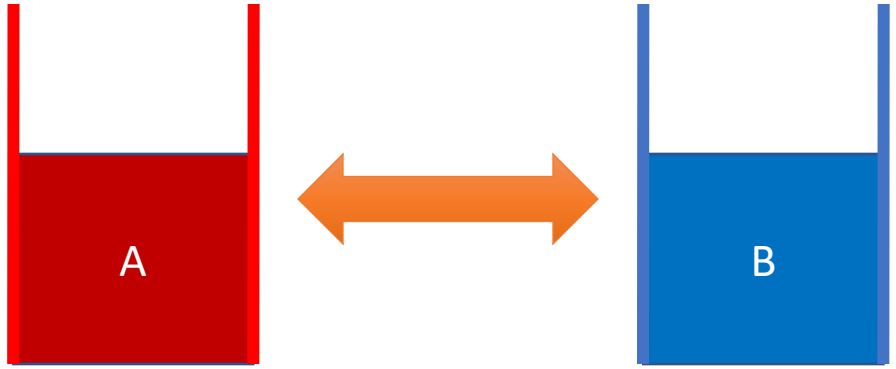
&contoh[1] == ptrContoh + 1 dan contoh[1] == *(ptrContoh + 1)

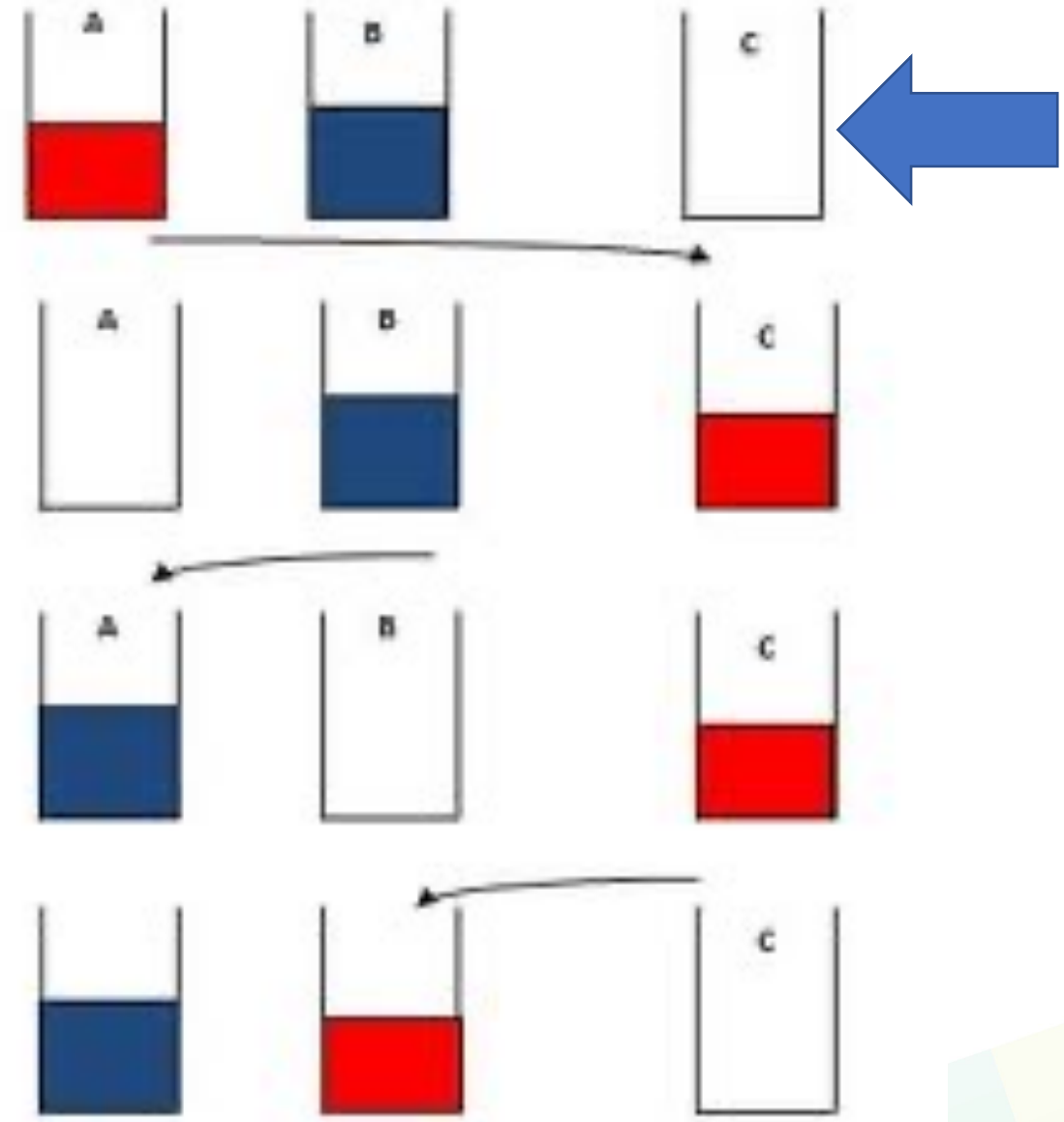
Dst…

Pointers and Function

- Informasi dapat diteruskan ke fungsi sebagai parameter. Parameter bertindak sebagai variabel di dalam fungsi.
- Terdapat 2 cara untuk melakukan *passing arguments* dalam sebuah fungsi.
  - *Pass by value* : nilai dari argument tersebut akan disalin (copy) ke nilai parameter fungsi yang sesuai.
  - *Pass by reference* : menggunakan referensi (*reference*) dari suatu nilai.
- Cara lain adalah menggunakan **pointer**.
- Contoh: penjumlahan dua bilangan menggunakan fungsi.

Swapping Algorithm





Temporary place

FAKULTAS TEKNOLOGI INFORMASI
Universitas Advent Indonesia

Swapping Function Using Pointer

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

Pointer to Pointer

Ketika pointer to pointer (double pointer) didefinisikan, pointer yang pertama digunakan untuk menyimpan alamat dari variabel yang telah dideklarasikan. Pointer yang kedua kemudian akan menyimpan alamat dari pointer yang pertama.

Syntax:

```
<datatype> **pointer_name;
int **ptr;
```

Pointers to Structure

Sebuah variabel pointer dapat dideklarasikan/didefinisikan bukan hanya menggunakan tipe data primitif seperti int, float, double, dsb, namun dapat juga didefinisikan dengan tipe berupa struktur (*struct/structures*).

Contoh sederhana:

```
#include <iostream>
using namespace std;

struct temp {
    int i;
    float f;
};
```

```
int main() {
    temp *ptr;
    return 0;
}
```