Modul 2 Programming

Array, Function, dan Pointer

1. Array

Array merupakan kumpulan dari beberapa elemen dengan **tipe data yang sama**. Artinya, tidak seperti di python dimana kita bisa buat array yang isinya integer campur string.

Misal, kita ingin menyimpan 3 string yaitu “apel”, “jeruk”, dan “anggur”. Daripada kita mendeklarasi 3 buah variabel, kita dapat membuat sebuah array dengan nama ‘buah’ yang isinya 3 string. Setiap elemen dalam array ini akan ada indeks yaitu angka untuk mengakses elemen dalam array. Indeks mulai dari 0 yak, bukan 1!

Berikut adalah contoh **deklarasi array, mengakses elemen, dan mengganti nilai dalam array**.

```
#include <iostream>
using namespace std;
int main(){
    // Deklarasi Array of String dengan size 3
    string buah[3] = {"apel", "jeruk", "anggur"};

    // Akses Elemen di Array (Contohnya kita ingin melakukan print)
    cout<<buah[0]<<endl; // print "apel"
    cout<<buah[1]<<endl; // print "jeruk"
    cout<<buah[2]<<endl; // print "anggur"

    // Mengganti Nilai Suatu Data di Array
    buah[1] = "pir"; // Array sekarang berisi ["apel", "jeruk",
"anggur"]
    cout<<buah[1]<<endl; // print "pir"
}
```

Kita juga dapat melakukan looping terhadap array. Misal, kita punya array nilai ujian suatu kelas yang ukurannya adalah 10. Daripada kita menulis cout 10 kali, kita dapat membuat for loop yang melakukan looping terhadap indeksnya. (Bayangkan jika size arranya 1000, yakali kita cout 1000 kali)

```cpp
#include <iostream>
using namespace std;
int main(){
    int nilai[10];

    // Looping input
    for(int i=0;i<10;i++){
        cout<<"Elemen ke-: ";
        cin>>nilai[i];
    }
    cout<<endl;

    // Looping print
    for(int i=0;i<10;i++){
        cout<<nilai[i]<<" ";
    }
    cout<<endl;

    // Looping untuk Mendapat Total Nilai
    int sum = 0;
    for(int i=0;i<10;i++){
        sum += nilai[i];
    }
    cout<<sum<<endl; // Jumlah dari nilai berdasar input
}
```

Apakah anda tahu bahwa string itu adalah array of character?

Misal, saya ingin mengganti suatu huruf dalam string.

```cpp
#include <iostream>
using namespace std;
int main(){
    string x = "Hello World";
    cout<<x<<endl; // Hello World
    x[4] = '0';
    cout<<x<<endl; // Hell0 World
}
```

Menariknya, memperlakukan string sebagai array of character diatas tidak dapat dilakukan di python! Mengapa? Karena string di python immutable object. Baca ini kalau penasaran https://www.geeksforgeeks.org/mutable-vs-immutable-objects-in-python/.

Tidak hanya integer atau string yang dapat kita jadikan array, kita juga dapat membentuk array of array! Jadinya ya seperti matrix. Contohnya seperti berikut:

```cpp
#include <iostream>
using namespace std;
int main(){
    // Deklarasi Array of Array
    int matrix[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};
    // 1 2 3
    // 4 5 6
    // 7 8 9

    // Akses Elemen (Misal baris ke-2 dan kolom ke-3 yaitu 6)
    cout<<matrix[1][2]<<endl;

    // Looping Matrix (urutan seperti nilai elemennya)
    for(int i=0;i<3;i++){ // Looping baris
        for(int j=0;j<3;j++){ // Looping kolom
            cout<<matrix[i][j]<<" ";
        }
        cout<<endl; // Ganti baris
    }
}
```

Kita juga bisa mendapatkan size dari array. Tidak semudah len() *ferguso*!

```cpp
#include <iostream>
using namespace std;
int main(){
    int arr[3] = {1,2,3};
    // sizeof mengembalikan jumlah byte di array, bukan jumlah
elemen
    cout<<sizeof(arr)/sizeof(int)<<endl;
    // 12 / 4 = 3
}
```

2. Function dan Procedure

Function di C++ mirip seperti *def* di python. Fungsi adalah blok subprogram yang dapat mengembalikan nilai (fungsi) atau tidak (prosedur).

Misal, dalam matematika, kita ingin membuat fungsi f(x) = 3x+5. Dengan memasukan x = 3, kita mendapat 3*3 + 5. Implementasi di C++ juga sama:

```
#include <iostream>
using namespace std;
// Jangan lupa beri tipe data keluaran fungsi dan parameter fungsi!
float f(float x){
    float ans = 3*x + 5;
    return ans; // "return untuk mengembalikan nilai"
}
int main(){
    cout<<f(3)<<endl; // 14
    cout<<f(3.14)<<endl; // 14.42
}
```

Fungsi tidak diwajibkan untuk mengeluarkan suatu nilai. Inilah disebut prosedur. Keluaran dari prosedur adalah void (tidak ada) sehingga untuk deklarasinya, kita pakai void.

```
#include <iostream>
using namespace std;
void sayHello(){
    cout<<"Hello"<<endl;
}
void sayHello2(string nama){
    cout<<"Hello "<<nama<<endl;
}
int main(){
    sayHello(); // Hello
    sayHello(); // Hello
    sayHello2("World"); // Hello World
    sayHello2("Pak Budi"); // Hello Pak Budi
}
```

Perlu diingat, sekali fungsi melakukan return, kode dalam fungsi akan selesai!

```cpp
#include <iostream>
using namespace std;
bool lebihBesar(int a, int b){
    if(a <= b){return 0;}
    // code dibawah tidak akan berjalan jika
    // if diatas terpenuhi dan terjadi return
    cout<<"test"<<endl;
    return 1;
}
int main(){
    cout<<lebihBesar(2, 3)<<endl;
}
```

Rekursif adalah fungsi yang memanggil fungsi itu sendiri. Contohnya saat kita ingin membuat fungsi fibonacci (derete fibonacci : 1,1,2,3,5,8,11,...)

```cpp
#include <iostream>
using namespace std;
// Fibonaci => fib(n)
// fib(0) = 1
// fib(1) = 1
// fib(n) = fib(n-1) + fib(n-2)
int fib(int n){
    if(n==0 || n==1){return 1;}
    return fib(n-1) + fib(n-2);
}
int main(){
    cout<<fib(4)<<endl;
    // fib(4) = fib(3) + fib(2) = 3 + 2 = 5
    // fib(3) = fib(2) + fib(1) = 2 + 1 = 3
    // fib(2) = fib(1) + fib(0) = 1 + 1 = 2
    // fib(1) = fib(0) = 1
}
```

Oh ya, jangan lupa bahwa array di C++ bersifat statis. Artinya apa? Artinya adalah size array tidak dapat berubah. Beda dengan python dimana array bersifat dinamis (tinggal pake append untuk memperbesar array).

Nah, kalo memang perlu, bisa memakai namanya vector. Opsional untuk dipelajari, tapi kalo dipelajari sangat membantu (apalagi kalo Anda anak *competitive programming*).

Referensi : https://en.cppreference.com/w/cpp/container/vector

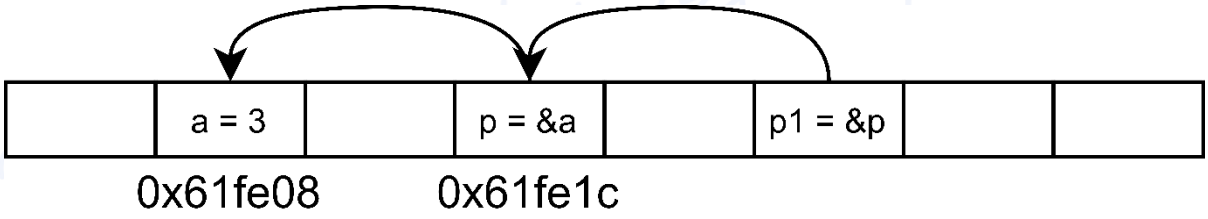
```cpp
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int>v;
    v.push_back(100); // push_back ekuivalen append di python
    v.push_back(200);
    v.push_back(300);
    cout<<v[1]<<endl; // 200
    cout<<v.size()<<endl; // 3 (size vector)
    // Ada clear, insert, erase, pop_back, dll
    // https://en.cppreference.com/w/cpp/container/vector
}
```

3. Pointer

Sebuah *nightmare* bagi pemula yang baru belajar C/C++ :). Tapi tenang, kita tidak akan langsung terjun mendalami seperti menonton video tentang pointer 3 jam 47 menit (https://youtu.be/zuegQmMdy8M).

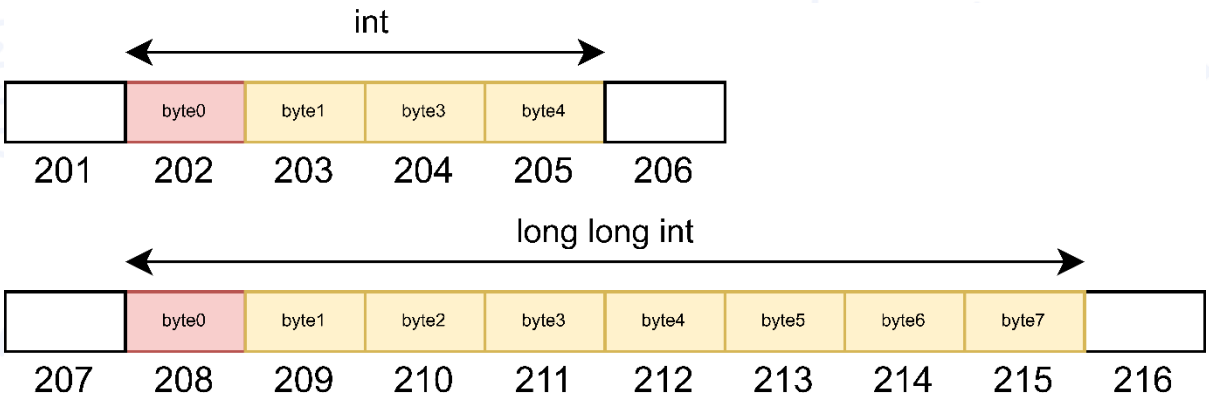
Sebelum mengerti pointer, kita harus paham apa itu memory address. Setiap kali kita membentuk suatu variabel, ada alamat di RAM komputer yang menyimpan data tersebut.

Misal, kita deklarasi integer a dengan value 3. Integer a tersebut disimpan di alamat 0x61fe08 pada memory kita. Pointer adalah variabel yang menyimpan alamat memory itu.




```
#include <iostream>
using namespace std;
int main(){
    int  a;   a = 3;   // Buat variabel
    int* p;   p = &a;  // Buat pointer yang menyimpan alamat a

    cout<<&a<<endl; // 0x61fe1c (Alamat dari a => Pakai '&')
    cout<<p<<endl; // 0x61fe1c (p menyimpan alamat a / p menunjuk
alamat a)

    cout<<*p<<endl; // 3 (Value dari data yang ditunjuk yaitu a)
    cout<<a<<endl; // 3 (Sama saja dengan akses a langsung)

    // Jadi disini, pointer p menunjuk address a dan kita bisa
mengakses data yang disimpan di address yang ditunjuk.


    // Bahkan kita juga bisa membuat pointer untuk pointer
    int** p1;
    p1 = &p;
    cout<<p1<<endl; // 0x61fe08 (Alamat dari p*)
    cout<<*p1<<endl; // 0x61fe1c (Value dari p*)
}
```

Lah, kenapa variabel pointer saja perlu datatipe (int or *whatever*)? Karena pointer ini perlu tahu size dari variable yang kita tunjuk (ukuran byte-nya). Saat kita menunjuk integer, berarti kita perlu tahu address byte pertamanya dan 3 byte kedepannya (ukuran int = 4 byte). Saat kita menunjuk long long int, berarti kita perlu tahu address byte pertamanya dan 7 byte kedepannya (ukuran long long int = 8 byte).



Yang diwarnakan merah adalah alamat yang ditunjuk oleh pointer, dan pointer harus tau jika dipanggil *pointer, maka berapa byte yang harus dibaca.