Nama Fazaliqa Achmad Farezi

Nim 5180711031

Pointer

Pointer adalah sebuah variabel atau object yang menunjuk ke variabel atau object lainnya. Sebelumnya pernah dijelaskan mengenai variabel, menyatakan bahwa. “Variabel merupakan sebuah representasi dari alamat memori pada komputer”.Dan pointer hanyalah variabel yang menyimpan alamat memori, memori tersebut dapat berasal dari variabel, obyek dan lain-lain. Dengan pointer kita dimungkinkan untuk menunjuk suatu memori, mendapatkan isi dari memori dan mengubah isi dari memori yang di tunjuk.

Macam macam pointer

**Address-of Operator (&)**, adalah operator yang memungkinkan kita untuk mendapatkan/melihat alamat memori yang dimiliki oleh variabel tersebut. Cara menggunakanya adalah dengan meletakan tanda & di depan identitas saat pemanggilan variabel. Hal itu akan membuat compiler memberikan alamat memori bukan isi/nilai dari memori tersebut.

Contoh Program

#include <iostream>

using namespace std;

int main()

{

    int var = 2;

    cout<<&var<<" memiliki nilai "<<var<<endl;

    return 0;

}

**Dereference Operator (\*)**, adalah operator yang memungkinkan mendapatkan isi/nilai dari sebuah memori berdasarkan alamat memori.

Contoh Program

#include <iostream>

using namespace std;

int main()

{

    int var = 2;

    cout<<var<<endl;

    cout<<&var<<endl;

    cout<<\*&var<<endl;

    return 0;

}

Pointer adalah sebuah variabel yang menyimpan alamat memori dari variabel lainnya, kita dimungkinkan untuk mengubah nilai dari variabel yang ditunjuk oleh pointer dan menyalin. Selama program berjalan kita bebas mengubah tujuan dari pointer, mengubah untuk menunjuk ke alamat memori variabel lain atau menunjuk ke alamat memori yang bukan merupakan variabel.

Pointer

### Mendirikan Pointer

Setelah anda mengerti fungsi-fungsi dari dua operator yang telah di jelaskan di atas. Sekarang anda dapat mendirikan pointer dengan menggunakan dua operator tersebut

Bentuk Penulisan

tipeData \*identitas;

//atau

tipeData \*identitas = &var;

Contoh Penulisan

int \*pInt;

double \*pDouble = &myVar;

pada umumnya pointer sebenarnya adalah variabel, peraturan yang dimiliki variabel juga berlaku pada pointer, jadi tidak jauh beda dengan variabel. pointer hanya mendapatkan beberapa perbedaan yaitu penambahan dua operator yang akan membuat variabel menjadi variabel pointer.

Untuk mendirikan sebuah variabel pointer kita hanya menambahkan dereference operator sebelum identitas. Operator dereference tidak harus melekat pada identitas, operator tersebut juga bisa di letakan setelah tipe data atau di antara tipe data dan identitas. dari berbagai cara penulisan tersebut memiliki makna yang sama yaitu satu operator dereference hanya akan berlaku pada saru variabel.

Pointer tidak hanya berlaku pada variabel, kita juga dapat melakukanya pada function, objek dan lain-lain. Dan cara implementasi pointer selain pada variabel, caranya masih sama seperti kita melakukanya pada variabel.

### Cara Mengakses Pointer

Variabel pointer adalah variabel yang memiliki alamat memori sebagai nilai dari variabel pointer tersebut. Dan pada pointer kita dimungkinkan untuk mengakses nilai dari pointer itu sendiri dan mengakses nilai dari alamat memori yang dimiliki(ditunjuk) oleh pointer.

Pointer merupakan variabel, untuk mengakses pointer tidak jauh beda dengan cara mengakses variabel. Untuk mengakses nilai dari pointer kita hanya cukup memanggil identiatas dari pointer tersebut.

pInt

pemanggilan itu akan menghasilkan nilai dari pointer yang berupa alamat memori dari variabel yang ditunjuk oleh pointer tersebut.

Karena pointer hanya dapat mememiliki nilai berupa alamat memori, untuk mengubah nilai dari pointer atau mengubah tujuan dari pointer kita membutuhkan operator address-of (&) pada operand sumber.

pInt = &myVar

operand sumber akan menghasilkan alamat memori dari myVar, dan hal itu merupakan nilai yang dibutuhkan oleh variabel pointer.

Contoh Program :

#include <iostream>

using namespace std;

int main()

{

    int var1 = 2, var2 = 5;

    int \*pVar = &var1;

    cout<<"var1 = "<<&var1<<endl;

    cout<<"pVar = "<<pVar<<endl;

    cout<<"====="<<endl<<endl;

    pVar=&var2;

    cout<<"var2 = "<<&var2<<endl;

    cout<<"pVar = "<<pVar<<endl;

    return 0;

}

Variabel pointer dapat menyimpan alamat memori bukan berarti anda dapat bebas memberi alamat memori secara langsung pada pointer.

int \*pInt = 0x012345;

Hal di atas tidak dimungkinkan untuk dilakukan.

Karena isi/nilai dari variable pointer merupakan sebuah alamat memori maka untuk mengakses nilai dari variabel yang dutunjuk oleh pointer kita membutuhkan operator dereference (\*).

\*pInt

Dalam pointer kita dimugkinkan untuk menyalin dan mengubah nilai pada variabel yang ditunjuk oleh pointer.

\*pInt = 2

Sebelum anda mencoba mengubah nilai dari memori yang ditunjuk oleh pointer anda harus yakin bahwa memori itulah yang ingin anda ubah nilainya. Jangan pernah mengubah memori yang bukan milik program anda, karena hal itu terlalu memiliki resiko yang cukup serius untuk komputer anda. Dan satu hal yang perlu anda ingat adalah, berilah nilai awal di saat deklarasi pointer. Itu dapat menimalisir kemungkinan kecelakaan, hal itu harus dilakukan karena saat deklarasi variabel pointer tanpa inisialisasi maka pointer akan memiliki nilai sampah.

Contoh Program :

#include <iostream>

using namespace std;

int main()

{

    int var = 2;

    int \*pVar = &var;

    cout<<"var = "<<var<<endl;

    cout<<"pVar = "<<\*pVar<<endl;

    cout<<"====="<<endl<<endl;

    \*pVar= 109;

    cout<<"var = "<<var<<endl;

    cout<<"pVar = "<<\*pVar<<endl;

    return 0;

}

**Ukuran pointer**  
Setiap kita medirikan pointer, pointer itu akan membutuhkan memori. Dan besar memori itu sama pada setiap tipe data yang digunakanBesar memori dari pointer tergantung pada mesin kompiler. jika kompiler merupakan 32bit maka pointer akan memakan memori sebanyak 4 bytes, Jika menggunakan 64bit maka pointer memakan memori sebanyak 8 bytes.

#include <iostream>

using namespace std;

int main()

{

    int var = 2;

    int \*pVar = &var;

    cout<<sizeof(pVar)<<endl;

    return 0;

}