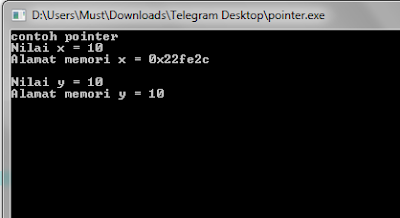
NAMA = Rasyi fazri hartono

NIM = 5180711111

**Materi Pointer C++**

**Pengertian Pointer di C++**  
  
Dalam ilmu komputer, pointer adalah tipe data bahasa pemrograman yang nilainya mengacu untuk menunjuk langsung nilai lain yang tersimpan di tempat lain dalam suatu memori komputer menggunakan alamat yang ada. Untuk bahasa pemrograman tingkat tinggi, pointer secara efektif dapat mengambil tempat pada general purpose registers di bahasa tingkat rendah seperti bahasa assembly atau kode mesin, tetapi bila memori tersedia.  
  
Contoh: jika variabel A berisi alamat dari variabel B, maka variabel A dikatakan menunjuk ke variabel B.  
  
  
**Kegunaan Pointer Di C++ :**  
  
Kegunaan pointer yang utama adalah untuk menyimpan alamat memori dari sebuah variabel (data type atau object dari class). Selain menyimpan alamat dari variabel, pointer juga dapat digunakan untuk menyimpan alamat dari sebuah fungsi (function pointer).  
Function pointer telah digunakan sejak dikenalkannya bahasa C,dan banyak digunakan untuk sebuah fungsi callback atau untuk meningkatkan readability dari sebuah code  
Anda dapat memperlakukan function pointer seperti pointer biasa (pointer ke datatype/object), anda dapat menyimpan,mengirimkan, merubah address, atau meng-evaluasi address dari pointer ke fungsi ini dengan sifat tambahan anda dapat memanggil fungsi yang ditunjuk oleh function pointer.  
  
Pointer bias juga berguna untuk :  
1. Mengirimkan “Parameter yang berupa variabel” ke dalam fungsi, artinya nilai variabel bisa diubah di dalam fungsi.  
2.Untuk membuat variabel DINAMIS (Bukan variabel Statis)  
penggunaan function pointer pada C++ dibatasi, yaitu function pointer tidak boleh menunjuk pada function yang berada dalam sebuah class (class member function) kecuali function tersebut berjenis static.  
  
  
**Tipe Data Pointer :**  
  
Dimana Tipe\_data merupakan tipe dari data yang ditunjuk, bukan tipe dari pointer-nya. Contoh :  
1. Mensubstitusikan address sebuah variabel ke pointer dengan memakai address operator &  
int x;  
int \*ptr;  
ptr = &x;  
2. Mensubstitusikan address awal sebuah array ke pointer  
char t[5];  
char \*ptr;  
ptr = t;  
3. Mensubstitusikan address salah satu elemen array dengan address operator  
char t[5] ;  
char \*ptr; ptr = &t[3];  
4. Mensubstitusikan address awal character string ke pointer char  
char \*ptr; ptr = "jakarta"  
5. Mensubstitusikan NULL pada pointer. NULL ada pointer kosong, menunjukkan suatu status dimana pointer itu belum diinisialisasikan dengan sebuah address tertentu.  
6.Memakai fungsi MALLOC.  
  
  
**Operator Pointer**  
  
Ada 2 operator pointer yang dikenal secara luas, yaitu operator & dan operator \*.  
  
Operator &  
Operator & merupakan operator alamat. Pada saat pendeklarasian variable, user tidak diharuskan menentukan lokasi sesungguhnya pada memory, hal ini akan dilakukan secara otomatis oleh kompiler dan operating sysem pada saat run-time. Jika ingin mengetahui dimana suatu variable akan disimpan, dapat dilakukan dengan memberikan tanda ampersand (&) didepan variable , yang berarti "address of".  
Contoh :  
ted = &andy;  
Penulisan tersebut berarti akan memberikan variable ted alamat dari variable andy. Karena variabel andy diberi awalan karakter ampersand (&), maka yang menjadi pokok disini adalah alamat dalam memory, bukan isi variable. Misalkan andy diletakkan pada alamat 1776 kemudian dituliskan instruksi sbb : andy = 25; fred = andy; ted = &andy;  
  
Operator \*  
perator \* merupakan operator reference. Dengan menggunakan pointer,kita dapat mengakses nilai yang tersimpan secara langsung dengan memberikan awalan operator asterisk (\*) pada identifier pointer, yang berarti "value pointed by".  
Contoh : beth = \*ted;  
(dapat dikatakan:"beth sama dengan nilai yang ditunjuk oleh ted") beth = 25, karena ted dialamat 1776, dan nilai yang berada pada alamat 1776 adalah 25. Ekspresi dibawah ini semuanya benar, perhatikan :  
andy = 25; &andy = 1776;  
ted = 1776; \*ted = 25;  
Ekspresi pertama merupakan assignation bahwa andy = 25;. Kedua, menggunakan operator alamat (address/derefence operator (&)), sehingga akan mengembalikan alamat dari variabel andy. Ketiga bernilai benar karena assignation untuk ted adalah ted = &andy;. Keempat menggunakan reference operator (\*) yang berarti nilai yang ada pada alamat yang ditunjuk oleh ted, yaitu 25. Maka ekspresi dibawah ini pun akan bernilai benar : \*ted = andy;  
  
  
**Deklarasi pointer**  
  
Seperti halnya variabel yang lain, variabel pointer juga harus dideklarasikan terlebih dahulu sebelum digunakan. Bentuk Umum :  
Tipe\_data \*nama\_pointer;  
  
Contoh : int \*nilai; ==> char \*huruf;  
  
  
**Ada dua cara yang dapat dilakukan untuk alokasi space di memori bagi pointer, yaitu:**  
  
1. Menempati space variable lain yang sudah punya space.  
Variabel lain tersebut dapat berupa variabel biasa (bukan pointer) atau pointer yang tentunya sudah punya alokasi space di memori.   
Syntaxnya :  
var\_pointer = &var\_biasa;  
  
contoh :  
#include <iostream>  
#include <conio.h>  
  
using namespace std;  
  
main() {  
  int x,\*y;  
  char \*judul="contoh pointer";  
  y=&x;  
  x=10;  
  cout<<judul<<endl;  
  cout<<"Nilai x = "<<x;  
  cout<<"\nAlamat memori x = "<<&x;  
  cout<<"\n\nNilai y = "<<\*y;  
  cout<<"\nAlamat memori y = "<<\*y;  
  getch();  
  }  
hasil output sebagai berikut :



2. Dialokasikan tersendiri di memori (Memori dinamis).  
Pointer tidak menempati space variabel lain, tetapi dialokasikan space tersendiri di memori dengan instruksi new.   
Syntaxnya:  
var\_pointer = new tipe\_data-pointernya;  
  
contoh :  
int \*a;  
float \*x;  
a=new int;  
x=new float;  
  
contoh program :  
 #include <iostream>  
#include <conio.h>  
  
using namespace std;  
main() {  
  int x,\*y;   y=new int;  
  x=15;        \*y=20;  
  cout<<"Nilai x = "<<x;  
  cout<<"\nAlamat memori x = "<<&x;  
  cout<<"\n\nNilai y = "<<\*y;  
  cout<<"\nAlamat memori y = "<<y;  
  delete y;  
  cout<<"\n\nSetelah di delete";  
  cout<<"\nNilai y = "<<\*y;  
  cout<<"\nAlamat memori y = "<<y;  
  getch();}  
hasil outputnya sebagai berikut :

