```
/*****************************************************************
* Author: Muhammad Rafi                                          *
* Purpose: Explaining Virtual Functions & RTTI (Example Class)   *
* Dated: August 18, 2007                                         *
* Version: 1.0                                                   *
)
* Last modified: August 20, 2007                                 *
*****************************************************************/
#include<iostream>
#include<typeinfo>
#include<cstring>


using namespace std;


class Mammal{
      public:
      virtual void Speaks(){ cout<< "Mammal Speaks..." <<endl;}

};

class Cat: public Mammal{
      public:
      void Speaks(){ cout<< "Meow.. Meow..." <<endl;}
};

class Dog: public Mammal{
      public:
       void Speaks(){ cout<< "Boow .. Boof..." <<endl;}
};

class Horse: public Mammal{
      public:
       void Speaks(){ cout<< "Winne .. Winne..." <<endl;}
};

Mammal * Build(){

     switch(rand()%3)
     {
        case 0: return new Cat;
        case 1: return new Dog;
        case 2: return new Horse;

     }

}

int main()
{
    Mammal * ptr;
    Cat * c1;
    Dog *d1;
    Horse * h1;
// typeid and polymorphic behaviour of derived class

for(int i=0; i < 100 ; i++)
{
   ptr= Build();
   cout<< typeid(ptr).name() <<endl;
   ptr->Speaks();
}

 //Checking typeid of derived class objects

  c1= new Cat;
  d1= new Dog;
  h1= new Horse;

  cout<< "\n\n\n\t " << typeid(c1).name() <<endl;
  cout<< "\t " << typeid(d1).name() <<endl;
  cout<< "\t " << typeid(h1).name() <<endl <<endl<<endl;


   system("pause");

}
```