#include <typeinfo> - datanin tipini tapmaq ucun library

int i = 5;

float j = 1.5f;

char c = 'a';

const type\_info& t1 = typeid(i \* j);

const type\_info& t2 = typeid(i \* c);

const type\_info& t3 = typeid(c);

cout << "t1 is of type " << t1.name() << endl;

cout << "t2 is of type " << t2.name() << endl;

cout << "t3 is of type " << t3.name() << endl;

//////////////////////////////////////////////////////////////////

Bir klassin polimorfik klass olmasi ucun en azi 1 virtual acar sozu olmalidir

//////////////////////////////////////////////////////////////////

mutable int roll2; // const daxilinde deyismek ucun mutable istifade olunu

//////////////////////////////////////////////////////////////////

C++ supports 4 types of casting

1) static cast

2) dynamic cast

3) Const cast

4) Reinterpret cast

1) static cast - (static\_cast)

Kompliyasiya vaxtinda cevirmeni yoxlayir

Cevire bilirse error cixartmit

Cevire bilmirse error cixartir (compile error)(red line)

castinge qarantiya verir

2) dynamic cast - (dynamic\_cast)

Run time merhelesinde cevrilib cevrilmeyeceyini bilirik

cevire bilmedikde nullptr qaytarir

toreme classin ozelliklerine catmaq ucun

dynamic\_cast istifade oluna biler

upcast(derived -> base)

downcast(base -> derived)

3) Const cast

Const cast constant olan pointeri constantliqdan cixarmaq ucundur

Example :

class Student

{

int roll;

mutable int roll2; // const daxilinde deyismek ucun mutable istifade olunu

public:

Student(int r) : roll(r)

{

}

void Fun() const

{

//this->roll = 20 // error

this->roll2 = 200; // okay because roll2 is mutable

(const\_cast<Student\*>(this))->roll = 20;

}

};

4) Reinterpret cast

Elinden gelen her seyi edir ki cevirsin

Neyise cevire bilirse cevirir, cevire bilmirse qeribe data qaytarir;

yaxsi ceheti : cevire bilmese bele datani qoruyub saxlayir