**Enumeration (enum)**

is a type whose values are user-defined named constants called enumerator unscoped and scoped enu prefer scoped enum than unscoped enum use enumerations when our object is to have one value out of a set of predefinednamed valuesenum MyEnum unscoped enum

#include <iostream>

using namespace std;

{

myfirstvalue = 69,

mysecondvalue = 6969,

mythirdvalue

};

enum class My\_Enum : int // scoped enum with underlying type

{

first = 100,

second,

third

};

int main()

{

MyEnum a\_enum = myfirstvalue; // create an enum type variable

a\_enum = mysecondvalue; // change the value of enum type variable

cout << a\_enum << '\n';

My\_Enum b\_enum = My\_Enum::first;

cout << (int)b\_enum << '\n';

}

#include <iostream>

using namespace std;

class Base

{

public:

virtual ~Base() {}

};

class Derive : public Base

{

};

int main()

{

int i = 33;

double d = 33.3;

i = d; // convert double to int

char c1 = 3;

char c2 = 3;

auto result = c1 + c2; // result is of int type

bool b = i; // convert int to boolean

int \*p = &i;

void \*vptr = p; // int pointer convert to void pointer

int \*p2 = static\_cast<int \*>(vptr); // cast void pointer to int pointer

cout << "int pointer: " << \*p2 << '\n';

cout << "I is: " << i << '\n';

cout << "B is: " << b << '\n';

Base \*base = new Derive;

Base \*base1 = new Base;

Derive \*derive = new Derive;

if (dynamic\_cast<Derive \*>(base)) // check if base can be cast to Derive

{

cout << "Yes" << '\n';

}

else

{

cout << "No" << '\n';

}

if (dynamic\_cast<Derive \*>(base1)) // check base1 can cast to Derive

{

cout << "Yes" << '\n';

}

else

{

cout << "No" << '\n';

}

if (dynamic\_cast<Base \*>(derive)) // check derive can cast to Base

{

cout << "Yes" << '\n';

}

else

{

cout << "No" << '\n';

}

delete base;

delete derive;

}

**conversions**

types can be convert into other types

**implicit conversions**

convert double to int can cause some infomation lost(the decimal part) when smaller intergers types(char) used in arithmetic operation,they promoted to integers any type can convert to boolean: if they are not 0 they become true and otherwise an array get convert to pointer of the first element -> prefer vector and array containers

**explicit conversions**

dynamic\_cast: converts pointers of base class to pointers to derived class, and vice versa up the inheritance chain reintrepret\_cast is avoided as it does not guarantee of any kind use static\_cast most of the time