Пример 05.01. Вызов методов по lvalue и rvalue ссылкам.

```
# include <iostream>

using namespace std;

class A
{
private:
    int a;
    mutable int b;

public:
//    A() : a(1), b(1) {}
    int f()& { cout << "int()&" << endl; return ++a; }
    int f() const&
    {
        cout << "int() const&" << endl;
//        ++a; // Error!

        return ++b;
    }
    int f()&& { cout << "int()&&" << endl; return b += a; }
    //  int f() {} // Error!
};

A func(const A& obj)
{
    return obj;
}

int main()
{
    A obj1;
    const A obj2{};

    obj1.f();
    obj2.f();
    move(obj1).f();

    A().f();
    func(obj1).f();
}
```

Пример 05.02. Создание и уничтожение объектов.

```
# include <iostream>

using namespace std;

class A
{
private:
    int value = 1;

public:
    A() { value *= 2; }
    A(const A&) { value *= 3; }
    A(A&&) noexcept { value *= 4; }
    ~A() { cout << value << endl; }
};

A f(A obj) { return obj; }

A f1() { return A(); }

A f2()
{
    A obj;

    return obj;
```

```
}

int main()
{
    cout << "prim 1" << endl;
    {
        A obj;

        f(obj);
    }
    cout << "prim 2" << endl;
    {
        A obj = f1();
    }
    cout << "prim 3" << endl;
    {
        A obj = f2();
    }
}
```

Пример 05.03. Конструкторы копирования и переноса.

```
# include <iostream>

using namespace std;

class Array
{
private:
    double* arr;
    int count;

public:
    Array() = default;
    Array(int cnt) : count(cnt) { arr = new double[count] {}; }
    Array(const Array& ar);
    Array(Array&& ar) noexcept;
    ~Array();

    bool equals(Array ar);

    static Array minus(const Array& ar);
};

Array::Array(const Array& ar) : count(ar.count)
{
    arr = new double[count];

    for (int i = 0; i < count; ++i)
        arr[i] = ar.arr[i];
}

Array::Array(Array&& ar) noexcept : count(ar.count)
{
    arr = ar.arr;

    ar.arr = nullptr;
}

Array::~Array()
{
    delete[] arr;
}

bool Array::equals(Array ar)
{
    if (count != ar.count) return false;

    int i;
    for (i = 0; i < count && arr[i] == ar.arr[i]; ++i);
```

```
    return i == count;
}

Array Array::minus(const Array& ar)
{
    Array temp(ar);

    for (int i = 0; i < temp.count; ++i)
        temp.arr[i] *= -1;

    return temp;
}


int main()
{
    Array mas{ 10 };

    if (mas.equals(Array::minus(mas)))
    {
        cout << "true" << endl;
    }
    else
    {
        cout << "false" << endl;
    }
}
```