# Код

#include <iostream>

using namespace std;

class Array5 {

private:

int array[5];

public:

Array5();

Array5(int, int, int, int, int);

~Array5();

friend ostream& operator<<(ostream&, Array5);

friend istream& operator>>(istream&, Array5&);

void operator\*=(int);

friend Array5 operator-(Array5&);

};

Array5::Array5() {

array[0] = 0;

array[1] = 0;

array[2] = 0;

array[3] = 0;

array[4] = 0;

}

Array5::Array5(int e0, int e1, int e2, int e3, int e4) {

array[0] = e0;

array[1] = e1;

array[2] = e2;

array[3] = e3;

array[4] = e4;

}

Array5::~Array5() {

}

ostream& operator<<(ostream& os, Array5 myArray) {

for (int i = 0;i < 5;i++) {

os << myArray.array[i] << " ";

}

os << endl;

return os;

}

istream& operator>>(istream& is, Array5& myArray) {

for (int i = 0;i < 5;i++) {

is >> myArray.array[i];

}

return is;

}

void Array5::operator\*=(int m) {

for (int i = 0;i < 5;i++) {

array[i] \*= m;

}

}

Array5 operator-(Array5& myArray) {

for (int i = 0;i < 5;i++) {

myArray.array[i] \*= -1;

}

return myArray;

}

int main()

{

Array5 a(1, 2, 4, 8, 16);

cout << "example array a:\n" << a;

a \*= 5;

cout << "same array after multiplying by 5:\n" << a;

Array5 b;

cout << "enter array b:\n";

cin >> b;

cout << "array b:\n" << b;

cout << "array b after changing sign of every element:\n";

cout << -b << endl;

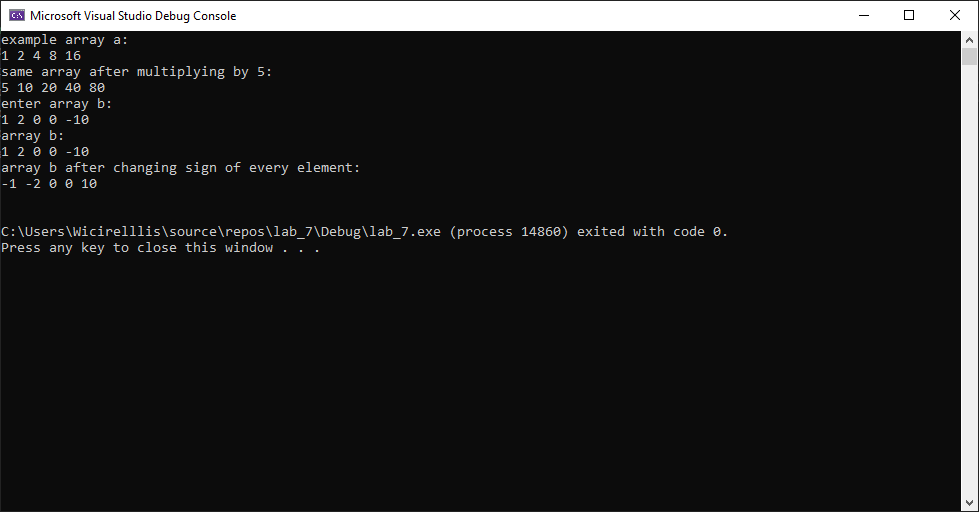
return 0;

}

# 2.

Проверим работу программы на каком-нибудь входе. Пусть ввод слеующий:

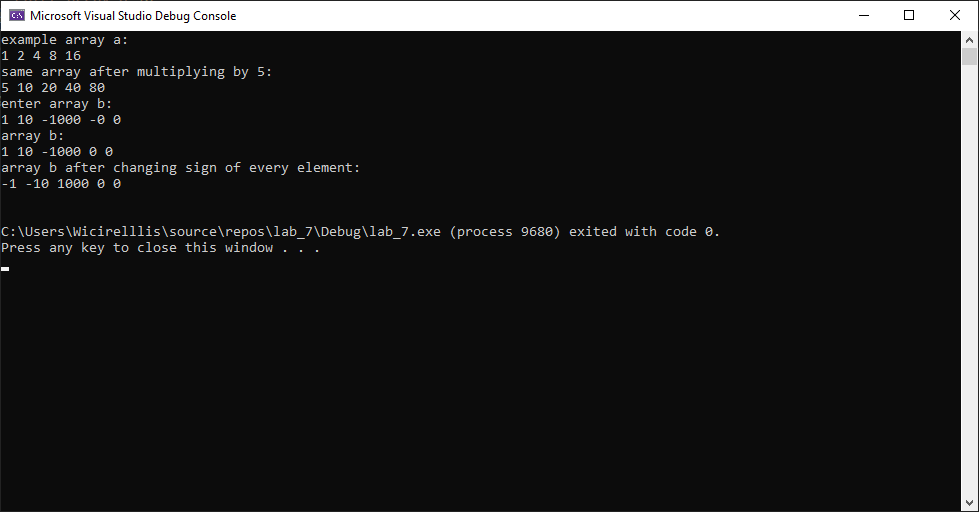
1 2 0 0 -10

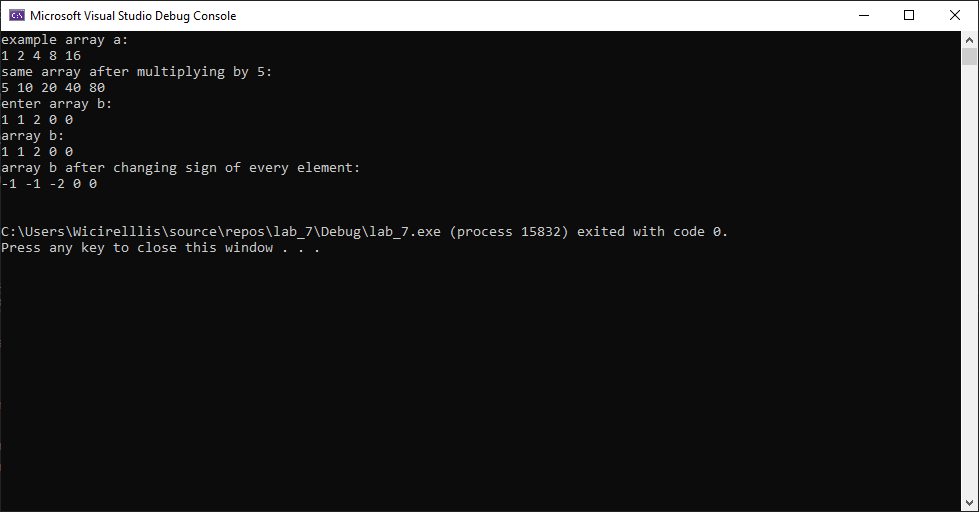


Работает (по крайней мере на данном входе) верно.

# 3.

Несколько скринов вывода





# 4. Вопросы

1) Что такое перегрузка оператора? Какие есть ограничения?

В плюсах есть станартные операторы, вроде +, +=, <<. Перегрузка – это до-\переопределение поведения этих операторов. Например + складывает два числа, но для произвольного класса + не определен. Мы не можем написать

class myClass {

//...

};

myClass a, b, c;

c = a + b;

Т.к. компилятор не знает, как складывать объекты. Перегрузив + мы сможем складывать и объекты (данного класса).

Нельзя перегрузить :: . .\* и тернарный. Нельзя создать новые операторы, т.е. перегрузить &| нельзя т.к. его изначально нет. Аргументы сохраняются при перегрузке, т.е. если + бинарный оператор, то нельзя его перегрузить и получить унарный.

2) Чем отличается friend-функция от обычный функции

Друзья имеют доступ к прайват полям\методам классов аргументов. Друга можно мыслить как метод нескольких классов одновременно.

int myFunction(myClassA a, myClassB b)

Можно посмотреть на такой пример. Есть функция, которая принимает два объекта (инстанса классов). Но у функции нет доступа к закрытым частям объектов a, b. Если добовать friend вначале, то этот доступ будет.

friend int myFunction(myClassA a, myClassB b)