ЛАБОРАТОРНАЯ РАБОТА № 5

ИССЛЕДОВАНИЕ ПЕРЕГРУЗКИ ОПЕРАТОРОВ

1 ЦЕЛЬ РАБОТЫ

Исследование назначения и средств создания перегруженных операторов при написании объектно-ориентированных программ.

2 ПОСТАНОВКА ЗАДАЧИ

Для заданного по варианту класса выполнить следующие действия:

- описать конструкторы и деструктор (по необходимости);

- переопределить оператор вывода в поток <<;

- переопределить оператор ввода из потока >>;

- переопределить заданные по варианту операторы;

Создать два объекта заданного по варианту класса и на их примере продемонстрировать корректную работу всех перегруженных операторов.

Вариант 3

Создать класс вектор Vector (float \*). Перегрузить операторы: 1) ++ (префиксный) как унарный метод класса, увеличивающий элементы вектора вдвое; 2) -- (префиксный) как унарную дружественную функцию, уменьшающую значение каждого элемента вектора на 6; 3) + как бинарный метод класса, суммирующий два вектора и сохраняющий результат в первом; 4) > как бинарную дружественную функцию, определяющую, существует ли в первом векторе такой элемент, который превосходит по величине все элементы второго вектора.

3 ТЕКСТ ПРОГРАММЫ

#include <conio.h>

#include <stdlib.h>

#include <iostream>

using namespace std;

class Vector

{

int n;

float \*V;

public:

Vector(int n1)

{

n=n1;

V = new float [n];

cout<<"Введите значения вектора: "<<endl;

for(int i=0; i<n; i++)

cin>>V[i];

cout<<endl;

for(int i=0; i<n; i++)

cout<<V[i]<<" ";

cout<<endl<<endl;

}

Vector &operator ++ ()

{

for(int i=0; i<n; i++)

V[i]+=V[i];

for(int i=0; i<n; i++)

cout<<V[i]<<" ";

return \*this;

}

Vector &operator + (Vector &ob2)

{

for(int i=0; i<n; i++)

V[i]+=ob2.V[i];

for(int i=0; i<n; i++)

cout<<V[i]<<" ";

return \*this;

}

~Vector()

{

delete V;

}

friend Vector &operator -- (Vector &ob1);

friend Vector &operator > (Vector &ob1, Vector &ob2);

friend ostream & operator << (ostream& output, const Vector & ob);

friend istream & operator >> (istream& input, Vector &ob);

};

Vector &operator -- (Vector &ob1)

{

for(int i=0; i<ob1.n; i++)

ob1.V[i]-=6;

for(int i=0; i<ob1.n; i++)

cout<<ob1.V[i]<<" ";

}

Vector &operator > (Vector &ob1, Vector &ob2)

{

int c=0;

for(int i=0; i<ob1.n; i++)

{

for(int j=0; j<ob2.n; j++)

if(ob1.V[i]>ob2.V[j]) c++;

if(c==ob1.n) break;

else c=0;

}

if(c==ob1.n) cout<<"Имеется";

else cout<<"Не имеется";

}

ostream & operator << (ostream& output, const Vector &ob)

{

output<<endl<<"Результат работы оператора вывода"<<endl;

for(int i=0; i<ob.n; i++)

output<<'|'<<ob.V[i]<<'|';

return output;

}

istream & operator >> (istream& input, Vector &ob)

{

for(int i=0; i<ob.n; i++)

input>>ob.V[i];

return input;

}

int main ()

{

setlocale(0, "RUS");

Vector ob1(4), ob2(4);

cout<<"Оператор ++: "<<endl;

++ob1;

cout<<endl;

cout<<"Оператор --: "<<endl;

--ob1;

cout<<endl;

cout<<"Оператор +: "<<endl;

ob1+ob2;

cout<<endl;

cout<<"Оператор >: "<<endl;

ob1>ob2;

cout<<endl;

cin>>ob2;

cout<<ob2;

return 0; }

4 ТЕСТИРОВАНИЕ

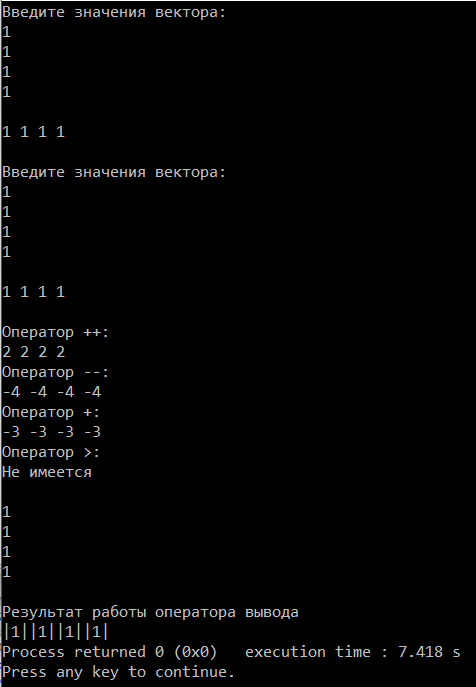


Рисунок 1 – Тестирование корректности работы перегруженных операторов.

ВЫВОД

В ходе выполнения лабораторной работы были исследованы назначения и средства создания перегруженных операторов при написании объектно-ориентированных программ.