«Отчет по П. З. № 4 ПЕРЕГРУЗКА ОПЕРАТОРОВ. ШАБЛОНЫ ФУНКЦИЙ» Сергей Учаев. Вариант 20.

Создать класс вещественных чисел. Определить оператор ++, как функцию-член и \* как дружественную функцию.

В одномерном массиве, состоящем из n элементов, вычислить сумму элементов массива, расположенных после последнего минимального элемента.

1. Постановка задачи:

№ 1:

Перегруженные операторы класса RealNumber:

void operator++() // Перегрузка оператора ++

Дружественная функция класса RealNumber:

friend void operator\*(RealNumber tmp) // Дружественная перегрузка \*

Входные данные + ограничения:

А) Поле представителя класса zxc.number. Тип double. Число типа double.

Исключения: буквы, строка.

№ 2:

Функции:

void InputINT(int\*); // Прототип функции ввода целого значения

void InputDOU(double\*); // Прототип функции ввода double значения

void InputFLO(float\*); // Прототип функции ввода float значения

void InputINTS(int\*); // Прототип функции ввода целого значения

double Proccess(T\* array, int count, double\* sum, int\* counter) // Функция обработки элемен-тов

Входные данные + ограничения

А) Переменная size. Тип int. Размерность матрицы (количество строк).

Исключения: буква, буква с цифрой, строка, вещественное, меньше или равно 0, большие числа (зависит от наличия свободной оперативной памяти компьютера).

Б) Переменная array1[]. Тип int. Элементы матрицы.

Исключения: буква, буква с цифрой, строка, дробное число.

В) Переменная array2[]. Тип floar. Элементы матрицы.

Исключения: буква, буква с цифрой, строка.

Г) Переменная array3[]. Тип double. Элементы матрицы.

Исключения: буква, буква с цифрой, строка.

Выходные данные + ограничения:

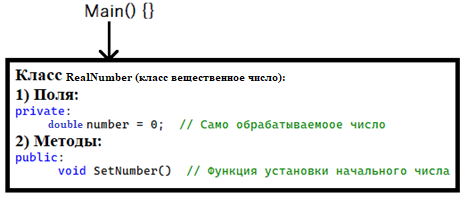
А) Перемення sum. Тип double. Сумма элементов от последнего отрицательного элемента.

Ограничения: буква, буква с цифрой, строка.

Среда разработки: Visual Studio 2022.

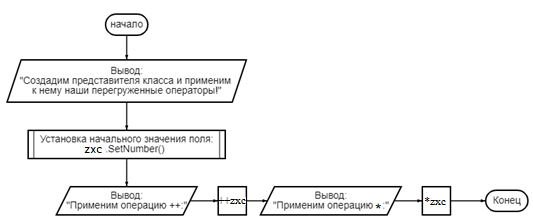
2. Разработка программы:

2.1 Описание классов, полей и методов классов.

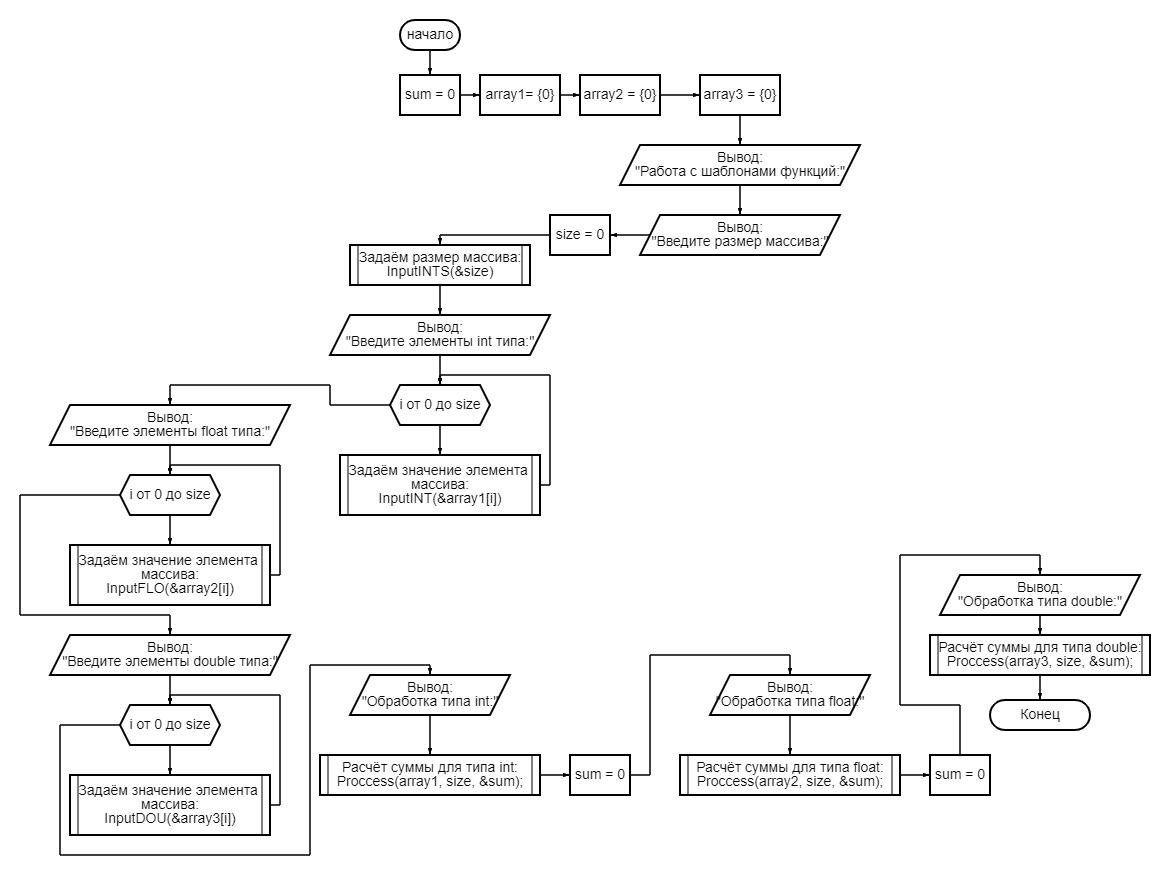


2.2Разработка схемы алгоритма.

№ 1:



№ 2:



2.3 Описание пользовательского интерфейса.

«Взаимодействие пользователя с программой осуществляется с помощью диалога».

Примеры диалогов:

№ 1:

* Вводимое число – 5,5

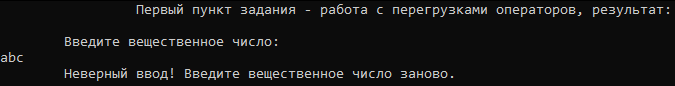
№ 2:

* Вводи размера массива – 4;
* Ввод элементов типа int – 1\_-2\_3\_4;
* Ввод элементов типа float – 1\_2\_-3.4\_4.5;
* Ввод элементов типа double – 1\_-2\_3\_-4.99

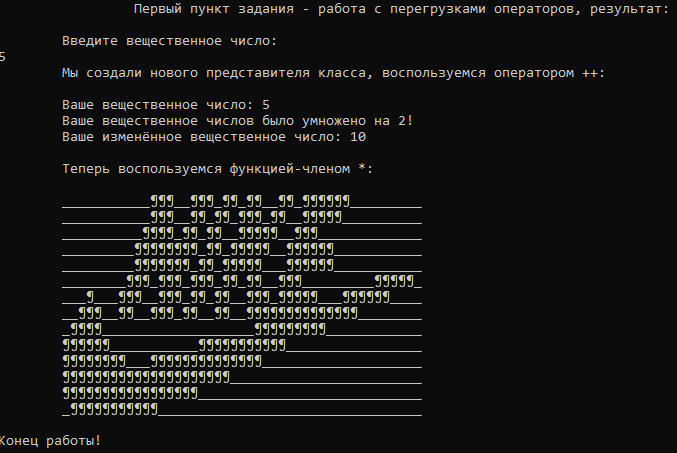
№ 1:

1. Ввод вещественного значения:

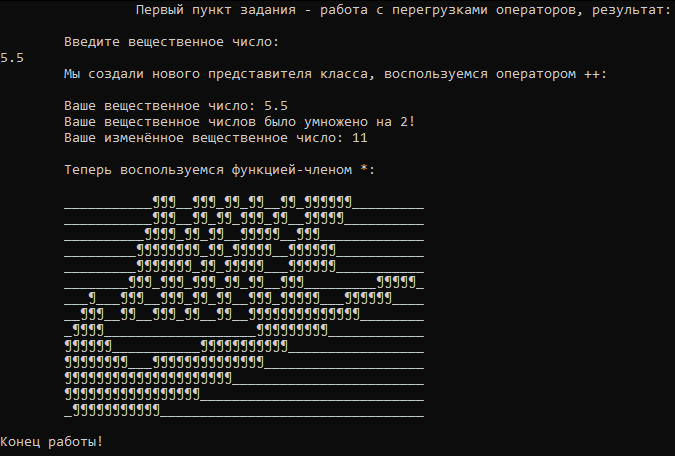
Вводимое значение – “abc”



Вводимое значение – 5



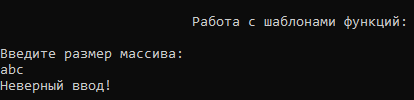
Вводимое значение – 5.5



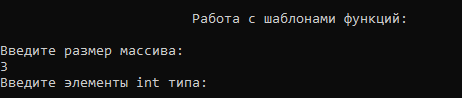
№ 2:

1. Ввод размера массива:

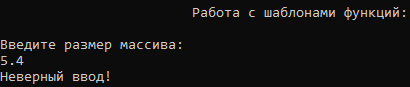
Вводимое значение – “abc”



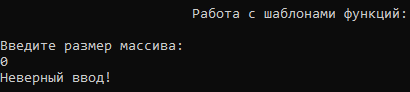
Вводимое значение – 3



Вводимое значение – 5.4



Вводимое значение – 0

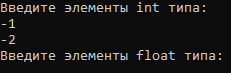


1. Ввод значения массива int;

Вводимое значение – “abc”



Вводимое значение – -2



Вводимое значение – 5.5



3) Ввод значения массива float;

Вводимое значение – “abc”



Вводимое значение – -2.5

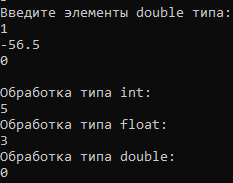


4) Ввод значения массива double;

Вводимое значение – “abc”



Вводимое значение – -56.5



3. Реализация и тестирование программы

3.1. Описание разработанной программы

№ 1:

Класс имеет одно приватное поле типа double (number) – само вещественное значение класса и один публичный метод (SetNumber) – установка этого начального значения, перегруженный оператор ++, который умножает число на 2, а также дружественный метод \*, который выводит рисунок.

В функции main мы создаём представителя класса zxc и задаём значение его полю при помощи функции SetNumber, После чего два перегруженных нами операторы.

№ 2:

Заранее определён шаблон для функции Process, которая выполняет основную логику программы.

После в main функции мы создаём три статических массива разных типов: int, float, double. И вызываем трижды метод Process, в которые в качестве аргументов суём массивы разных типов.

3.2. Тестирование программы

№ 1:

|  |  |  |
| --- | --- | --- |
| Входные данные | Результат программы | Смысл |
| zxc.SetNumber();  (abc) | Неверный ввод! | Проверка на строку |
| zxc.SetNumber();  (5) | 2  \*Рисунок\* | Проверка на корректность работы программы |

№ 2:

|  |  |  |
| --- | --- | --- |
| Входные данные | Результат программы | Смысл |
| InputINTS(&size)  (abc) | Неверный ввод! | Проверка на строку |
| InputINTS(&size)  (2.4) | Неверный ввод! | Проверка на дробное число |
| InputINTS(&size)  (2) | 2 | Проверка на корректность работы программы |
| InputINTS(&size)  (-2) | Неверный ввод! | Проверка на отрицательную размерность |
| InputINT(&array1[i])  (abc) | Неверный ввод! | Проверка на строку |
| InputINT(&array1[i])  (5.5) | Неверный ввод! | Проверка на дробное число |
| InputINT(&array1[i])  (4) | 4 | Проверка на корректность работы программы |
| InputFLO(&array2[i])  (abc) | Неверный ввод! | Проверка на строку |
| InputFLO(&array2[i])  (5.5) | 5.5 | Проверка на корректность работы программы |
| InputDOU(&array3[i])  (abc) | Неверный ввод! | Проверка на строку |
| InputDOU(&array3[i])  (5.5) | 5.5 | Проверка на корректность работы программы |
| InputINTS(&size)  (2)  InputINT(&array1[i])  (0)  InputINT(&array1[i+1])  (0)  InputFLO(&array2[i])  (5.5)  InputFLO(&array2[i+1])  (5.5)  InputDOU(&array3[i])  (-6.4)  InputDOU(&array3[i+1])  (-4) | Обработка типа int:  0  Обработка типа float:  0  Обработка типа double:  -4 | Проверка на корректность работы программы |

Возможный диалог:

№ 1:

Ввод начального значения поля:

5

№ 2:

Ввод размера массива – ввод элементов целого типа – ввод элементов float типа – ввод элементов double типа:

3 \_ -1 4 -3 \_ 2.5 -9.4 4 \_ -64.1 -4 5

Код:

№ 1:

#include <iostream> // Стандартная библиотек ввода-вывода

using namespace std; // Использование стандартного пространства имён

void InputDOU(double\* tmp) // функция ввода вещественного числа

{

while (!(cin >> \*tmp) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "\tНеверный ввод! Введите вещественное число заново." << endl;

}

}

class RealNumber // Класс вещественного числа

{

private:

double number = 0; // Вещественное поле

public:

void SetNumber() // Функция установки начального значения

{

cout << "\tВведите вещественное число: " << endl;

InputDOU(&number);

}

void operator++() // Описание определение оператора ++

{

cout << "\tВаше вещественное число: " << number << endl;

number \*= 2;

cout << "\tВаше вещественное числов было умножено на 2!" << endl;

cout << "\tВаше изменённое вещественное число: " << number << endl;

}

friend void operator\*(RealNumber tmp) // Описание перегрузки дружественной функции

{

cout << "\t\_\_\_\_\_\_\_\_\_\_\_¶¶¶\_\_¶¶¶\_¶¶\_¶¶\_\_¶¶\_¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_\_\_\_\_\_\_\_\_\_\_¶¶¶\_\_¶¶\_¶¶\_¶¶¶\_¶¶\_\_¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_\_\_\_\_\_\_\_\_\_¶¶¶¶\_¶¶\_¶¶\_\_¶¶¶¶¶\_\_¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_\_\_\_\_\_\_\_\_¶¶¶¶¶¶¶¶\_¶¶\_¶¶¶¶¶\_\_¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_\_\_\_\_\_\_\_\_¶¶¶¶¶¶¶\_¶¶\_¶¶¶¶¶\_\_\_¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_\_\_\_\_\_\_\_¶¶¶\_¶¶¶\_¶¶¶\_¶¶\_¶¶\_\_¶¶¶\_\_\_\_\_\_\_\_\_¶¶¶¶¶\_" << endl;

cout << "\t\_\_\_¶\_\_\_¶¶¶\_\_¶¶¶\_¶¶\_¶¶\_\_¶¶¶\_¶¶¶¶¶\_\_\_¶¶¶¶¶¶\_\_\_\_" << endl;

cout << "\t\_\_¶¶¶\_\_¶¶\_\_¶¶¶\_¶¶\_\_¶¶\_\_¶¶¶¶¶¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_¶¶¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t¶¶¶¶¶¶¶¶\_\_\_¶¶¶¶¶¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t\_¶¶¶¶¶¶¶¶¶¶¶\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

};

void main(void)

{

setlocale(LC\_ALL, "Russian"); // Установка русского языка в консоль

cout << "\t\t Первый пункт задания - работа с перегрузками операторов, результат:" << endl << endl;

RealNumber zxc; // Создание представителя класса

zxc.SetNumber(); // Вызов функции установки начального значения вещественного поля

cout << "\tМы создали нового представителя класса, воспользуемся оператором ++:" << endl << endl;

++zxc; // Вызов перегруженного оператора

cout << endl << "\tТеперь воспользуемся функцией-членом \*:" << endl << endl;

\*zxc; // Вызов дружественной функции

cout << endl << "Конец работы!\n";

}

№ 2:

#define SIZE 10 // Фиксированная размерность массивов

#include <iostream> // Стандартная библиотека ввода-вывода

#include <cmath> // Математическая библиотека

using namespace std; // Использование стандратного пространства имён

void InputINT(int\*); // Прототип функции ввода элементов целого типа

void InputDOU(double\*); // Прототип функции ввода элементов типа double

void InputFLO(float\*); // Прототип функции ввода элементов типа float

void InputINTS(int\*); // Прототип функции ввода размера массивов

template <typename T> // Прототип шаблонна

double Proccess(T\* array, int count, double\* sum, int\* counter) // Основная функция обработки

{

int I = -1; // Индекс последнего отрицательного элемента

for (int i = count - 1; i >= 0; --i)

{

if ((array[i]) < 0)

{

I = i;

break;

}

}

if (I == -1) // Проверка найденного индекса последнего отрицательного элемента

{

cout << "У вас нет отрицательных элементов" << endl;

\*counter = -1;

}

else

{

for (int i = ++I; i < count; ++i)

{

\*sum += array[i];

++\* counter;

}

}

return \*sum;

}

void main(void)

{

setlocale(LC\_ALL, "Russian"); // Установка русского языка в консоли

double sum = 0; // Сумма подходящих элементов

int counter = 0; // Счётчик подходящих элементов

int array1[SIZE] = { 0 };

float array2[SIZE] = { 0 };

double array3[SIZE] = { 0 };

cout << "\n\t\t\tРабота с шаблонами функций:\n" << endl;

cout << "Введите размер массива:" << endl;

int n; // Размерность массивов

InputINTS(&n); // Вызов функции ввода размерности массивов

cout << "Введите элементы int типа:" << endl;

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

{

InputINT(&array1[i]); // Заполнение массива типа int

}

cout << "Введите элементы float типа:" << endl;

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

{

InputFLO(&array2[i]); // Заполнение массива типа float

}

cout << "Введите элементы double типа:" << endl;

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

{

InputDOU(&array3[i]); // Заполнение массива типа double

}

double TMP = 0; // Временная переменная

cout << "\nОбработка типа int:" << endl;

TMP = Proccess(array1, n, &sum, &counter); // Вызов функции-обработки для первого массива

if (counter > 0)

{

cout << TMP;

}

if (counter == 0)

{

cout << "У вас нет элементов после последнего отрицательного числа!" << endl;

}

counter = 0;

sum = 0;

cout << "\nОбработка типа float:" << endl;

TMP = Proccess(array2, n, &sum, &counter); // Вызов функции-обработки для второго массива

if (counter > 0)

{

cout << TMP;

}

if (counter == 0)

{

cout << "У вас нет элементов после последнего отрицательного числа!" << endl;

}

counter = 0;

sum = 0;

cout << "\nОбработка типа double:" << endl;

TMP = Proccess(array3, n, &sum, &counter); // Вызов функции-обработки для третьего массива

if (counter > 0)

{

cout << TMP;

}

if (counter == 0)

{

cout << "У вас нет элементов после последнего отрицательного числа!" << endl;

}

}

void InputINTS(int\* tmp) // Опасание функции ввода размера массивов

{

while (!(cin >> \*tmp) || (cin.peek() != '\n') || (\*tmp <= 0))

{

cin.clear();

while (cin.get() != '\n');

cout << "Неверный ввод!" << endl;

}

}

void InputINT(int\* tmp) // Описание функции ввода числа типа int

{

while (!(cin >> \*tmp) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "Неверный ввод!" << endl;

}

}

void InputDOU(double\* tmp) // Описание функции ввода числа типа double

{

while (!(cin >> \*tmp) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "Неверный ввод!" << endl;

}

}

void InputFLO(float\* tmp) // Описание функции ввода числа типа float

{

while (!(cin >> \*tmp) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "Неверный ввод!" << endl;

}

}