МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Прикладная математика и информатика»

ОТЧЕТ

по лабораторной работе

СОЗДАНИЕ КЛАССА «ВЕКТОР»

Выполнил: студент группы

3822Б1ПМ2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М. М. Хамков

Подпись

Проверил: к.ф.-м.н., доц.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. А. Черных

Подпись

Нижний Новгород  
2023

Содержание.

[Введение](#_Toc532934514) 3

[Постановка задачи](#_Toc532934515) 4

[Описание алгоритмов.](#_Toc532934532) 5

[*1.* Изменение размера вектора](#_Toc532934533) 5

[*2.* Добавление координаты в конец вектора](#_Toc532934534) 6

[*3.* Добавление координаты в начало вектора](#_Toc532934534) 7

[Описание программы.](#_Toc532934537) 8

[Результаты работы.](#_Toc532934537) 11

[Заключение.](#_Toc532934539) 12

[Литература.](#_Toc532934540) 13

[Приложение.](#_Toc532934541) 14

**Введение**

Вектор – это математический объект, который описывает направление и величину физической величины. Векторы широко используются в различных областях науки и техники, таких как физика, механика, электротехника, компьютерная графика и многое другое. В данной лабораторной работе будет реализован класс «Вектор» с помощью средств языка программирования C++.

**Постановка задачи**

В этой работе я реализую класс «Вектор» с помощью средств языка программирования C++. Также я создам класс «Exception» для обработки ошибок в векторных вычислениях.

Так же в работе будут использованы шаблоны.

Шаблоны в C++ - это механизм, который позволяет определять обобщенные типы данных и алгоритмы, которые могут работать с этими типами. Шаблоны позволяют создавать классы, функции и структуры данных, которые могут принимать различные типы данных в качестве параметров.

Основная идея шаблонов заключается в том, что код, который работает с обобщенными типами данных, может быть написан один раз и использоваться для работы с различными типами данных. Шаблоны позволяют создавать универсальные решения, которые могут быть применены к различным задачам.

Шаблоны в C++ могут быть параметризованы типами данных или значениями. Параметры типа данных могут быть любым типом данных, включая пользовательские классы. Параметры значения могут быть любым константным значением, таким как целочисленная константа или перечисление.

**Описание алгоритмов**

1. *Изменение размера вектора*

В алгоритме реализована возможность менять размер уже созданного вектора. При уменьшении этого значения, лишние координаты будут удаляться, а при увеличении добавленные ячейки будут заполняться нулями.

1. *Добавление координаты в конец вектора*

В алгоритме реализована возможность добавления с помощью специального метода координаты в конец вектора, с увеличением его размера на 1. Остальные координаты вектора сохраняются на своих прежних местах.

1. *Добавление координаты в начало вектора*

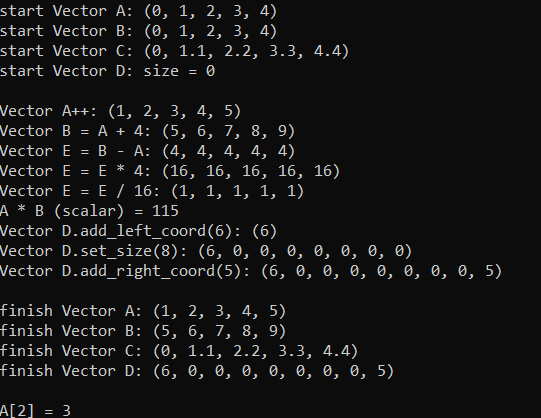
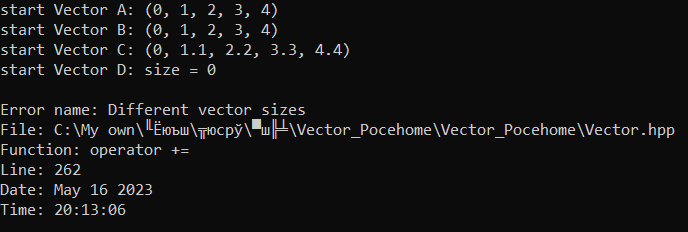
В алгоритме реализована возможность добавления с помощью специального метода координаты в начало вектора, с увеличением его размера на 1. Остальные координаты вектора смещаются на одну позицию вправо.

**Описание программы**

Необходимо реализовать класс «Вектор», с возможность выполнять такие операции над элементами этого класса, как: +, -, \*, /, =, ==, !=, +=, ++, <<, >>, []. Также необходимо сделать конструкторы, деструктор, сеттеры и геттеры. При реализации функций должны использоваться шаблоны, для удобной работы с разными типами данных.

1. Vector()   
   Конструктор по умолчанию. Размер равен 0, а список координат – нулевой указатель.
2. Vector(size\_t size)  
   Конструктор. Размер равен size, а список координат – массив из 0 размера size.
3. Vector(size\_t size, T\* coords)  
   Конструктор . Размер равен size, а список координат – coords.
4. Vector(const Vector& vector)  
   Конструктор копирования. Создаёт копию переданного вектора.
5. ~Vector()  
   Деструктор. Удаляет массив cords из памяти.
6. size\_t get\_size() const  
   Возвращает размер вектора.
7. void set\_size(size\_t size)  
   Изменяет размер вектора на size.
8. T& get\_coord(size\_t index) const  
   Возвращает координату вектора, находящуюся под индексом index.
9. void set\_coord(size\_t index, T coord)  
   Меняет значение координаты вектора, находящейся под индексом index, на coord.
10. void add\_right\_coord(T coord)  
    Добавляет координату coord в конец вектора, увеличивая его размер на 1.
11. void add\_left\_coord(T coord)  
    Добавляет координату coord в начало вектора, увеличивая его размер на 1.
12. T& operator[](size\_t index) const  
    Возвращает координату вектора, находящуюся под индексом index.
13. Vector<T>& operator=(const Vector<T>& vector)  
    Делает исходный вектор равным вектору vector.
14. friend std::ostream& operator<<(std::ostream& out, const Vector& vector)  
    Позволяет выводить вектор с помощью библиотеки <iostream>.
15. friend std::istream& operator>>(std::istream& in, const Vector& vector)  
    Позволяет вводить вектор пользователю с помощью библиотеки <iostream>.
16. bool operator==(Vector<T1> vector1, Vector<T2> vector2)  
    Оператор равенства.
17. bool operator!=(Vector<T1> vector1, Vector<T2> vector2)  
    Оператор неравенства.
18. Vector<T> operator+(const Vector<T>& vector, const T& n)  
    Прибавляет к каждой координате вектора vector число n.
19. Vector<T> operator-(const Vector<T>& vector, const T& n)  
    Вычитает из каждой координаты вектора vector число n.
20. Vector<T> operator\*(const Vector<T>& vector, const T& n)  
    Умножает каждую координату вектора vector на число n.  
    Возможна также обратная запись: n \* vector
21. Vector<T> operator/(const Vector<T>& vector, const T& n)  
    Делит каждую координату вектора vector на число n.
22. Vector<T> operator+(const Vector<T>& vector1, const Vector<T>& vector2)  
    Складывает вектора.
23. Vector<T> operator-(const Vector<T>& vector1, const Vector<T>& vector2)  
    Вычитает из вектора vector1 вектор vector2.
24. T operator\*(const Vector<T>& vector1, const Vector<T>& vector2)  
    Скалярное умножение векторов
25. Vector<T>& operator+=(Vector<T>& vector1, const Vector<T>& vector2)  
    Увеличивает вектор vector1 на вектор vector2
26. Vector<T>& operator+=(Vector<T>& vector, const T& n)  
    Увеличивает координаты вектора vector на число n.
27. Vector<T>& operator++(Vector<T>& vector)  
    Увеличивает координаты вектора vector на 1.
28. Vector<T>& operator++(Vector<T>& vector, int)   
    Увеличивает координаты вектора vector на 1.
29. std::ostream& operator<<(std::ostream& out, const Exception& exception)  
    Выводит ошибку в консоль.

**Результаты работы**

1. Результат запуска тестового файла:
2. Результат вывода ошибки:

**Заключение**

В рамках данной работы был реализован класс «Вектор» с помощью средств языка программирования C++. С элементами этого класса можно совершать ряд функций, необходимых для вычислений с векторами. Также реализованы операторы ввода и вывода и собственная обработка ошибок для удобной работы с этим классом.

**Литература**

* Записи с лекций по предмету «Языки и методы программирования»;
* Записи с практических занятий по предмету «Языки и методы программирования».

**Приложения**

<https://github.com/Pocehome/Vector_Pocehome.git>