Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа по классам № 7**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**семестр** 2

Выполнил студент гр. ИВТ-21-1б

Балдин Алексей Игоревич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

г. Пермь - 2022

**Цель**

Реализация шаблона класса-контейнера.

**Постановка задачи**

1. Определить шаблон класса-контейнера (см. лабораторную работу №6).
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Инстанцировать шаблон для стандартных типов данных (int, float, double).
5. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы стандартных типов данных.
6. Реализовать пользовательский класс (см. лабораторную работу №3).
7. Перегрузить для пользовательского класса операции ввода-вывода.
8. Перегрузить операции необходимые для выполнения операций контейнерного класса.
9. Инстанцировать шаблон для пользовательского класса.
10. Написать тестирующую программу, иллюстрирующую выполнение операций для контейнера, содержащего элементы пользовательского класса.

**Исходные данные**

Вариант 5:

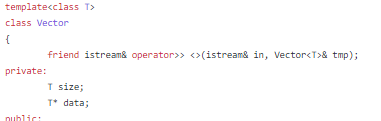
Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:

[] – доступа по индексу;

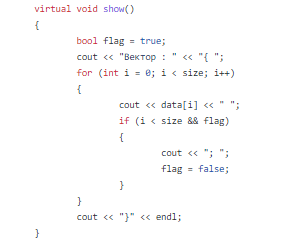
int() – определение размера вектора;

\* вектор – умножение элементов векторов a[i]\*b[i];

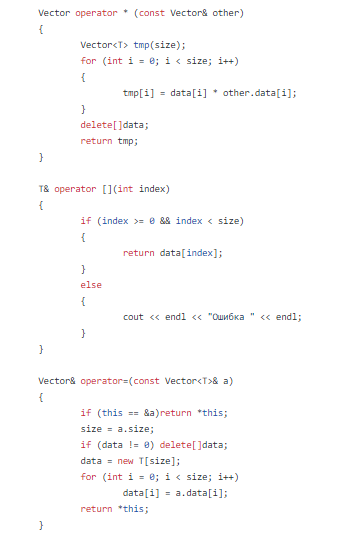
Пользовательский класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием.

Создаю шаблонный класс Vector, в полях есть дружественный перегруженный оператор ввода, также поля private: тоже шаблонного типа size, \*data.  


Виртуальный метод в этом же классе. Создан вместо перегрузки оператора вывода, посчитал этот вариант рациональней.

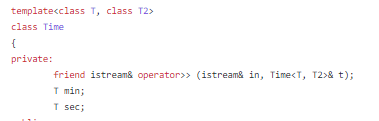


Далее в этом же классе идет перегрузка операторов:

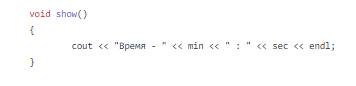


Класс Time, тоже шаблонный, но принимает два параметра.

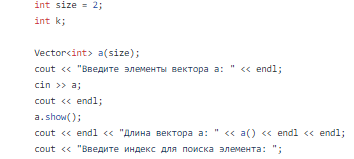
В полях private есть дружественный оператор вывода, также поля (минуты и секунды):

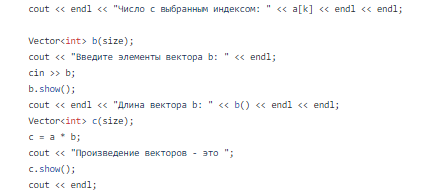


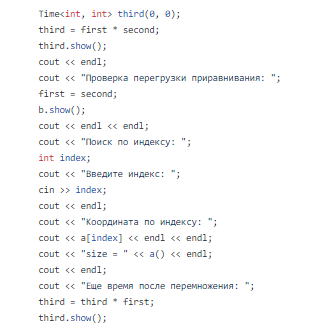
Метод show – виртуальная функция, будет показывать нам минуты и секунды через двоеточие:



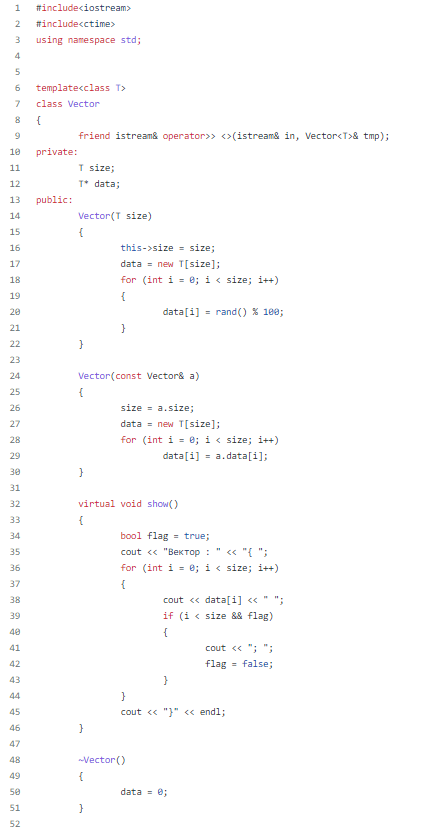
В main() создаю поля типа Vector, Time. Так как есть виртуальная функция, то вызываю функцию show для объектов класса и вывожу по условию.

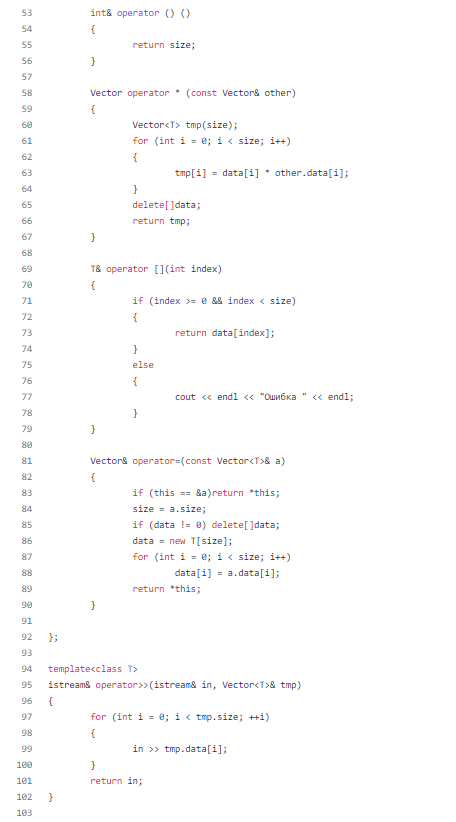


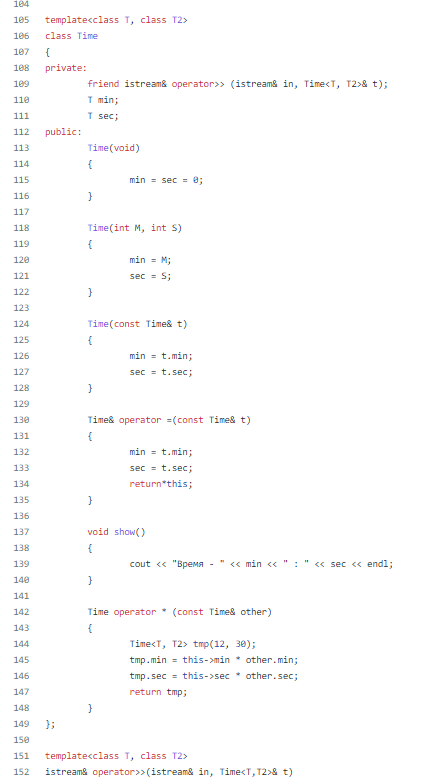


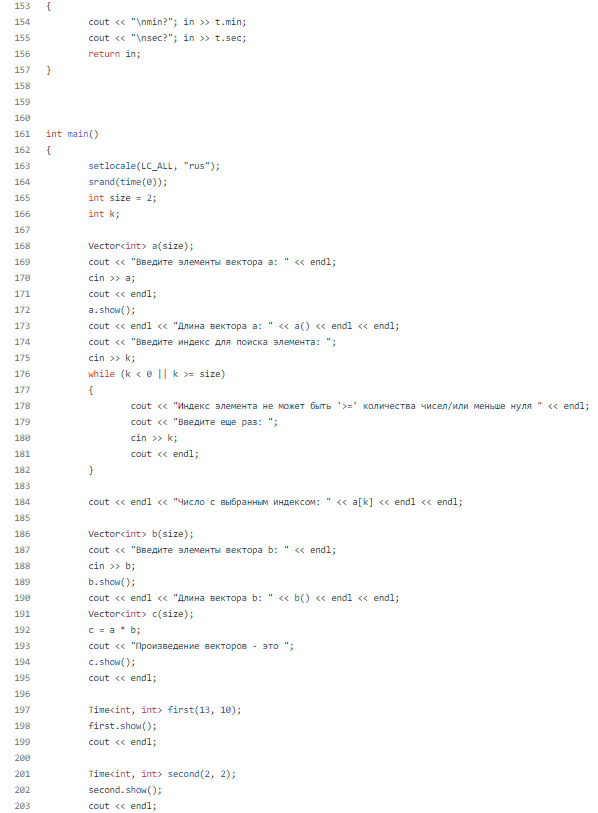


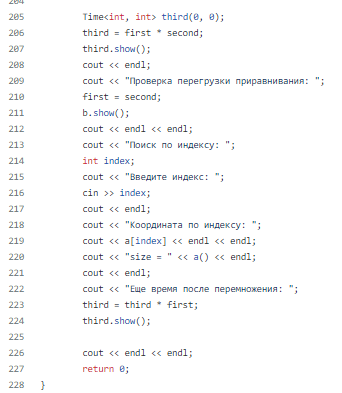
Код программы:











Результат программы:

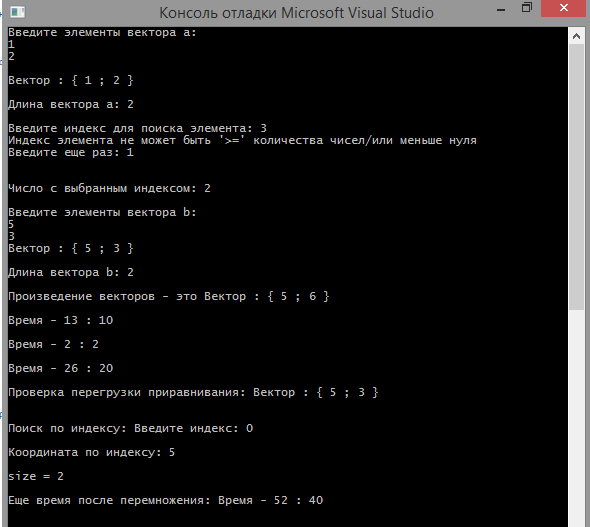


Диаграмма к задаче:

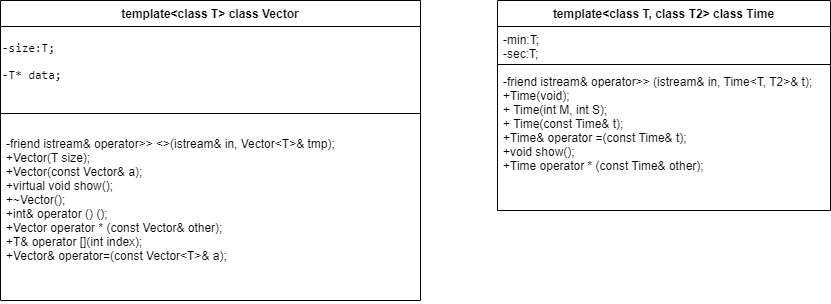


Рисунок 1

**Контрольные вопросы:**

* 1. В чем смысл использования шаблонов?
  2. Каковы синтаксис/семантика шаблонов функций?
  3. Каковы синтаксис/семантика шаблонов классов?
  4. Что такое полиморфные функции?
  5. Что такое параметры шаблона функции?
  6. Перечислите основные свойства параметров шаблона функции.
  7. Как записывать параметр шаблона?
  8. Можно ли перегружать параметризованные функции?
  9. Перечислите основные свойства параметризованных классов.
  10. Все ли компонентные функции параметризованного класса являются параметризованными?
  11. Являются ли дружественные функции, описанные в параметризованном классе, параметризованными?
  12. Могут ли шаблоны классов содержать виртуальные компонентные функции?
  13. Как определяются компонентные функции параметризованных классов вне определения шаблона класса?
  14. Что такое инстанцирование шаблона?
  15. На каком этапе происходит генерирование определения класса по шаблону?

**Ответы на вопросы:**

1. Шаблоны позволяют определить конструкции (функции, классы), которые используют определенные типы, но на момент написания кода точно не известно, что это будут за типы. Иными словами, шаблоны позволяют определить универсальные конструкции, которые не зависят от определенного типа.
2. Продемонстрированно в решении лаб.р.
3. Все также продемонстрированно в решении лаб.р. :

template<список\_параметров\_шаблона> определение\_класса

1. Параметры шаблона являются его формальными аргументами, а типы тех аргументов, которые используются в конкретных обращениях к функции, служат фактическими аргументами шаблона.
2. Имена параметров шаблона должны быть уникальными во всем определении шаблона.

Список параметров шаблона функций не может быть пустым, так как при этом теряется возможность параметризации и шаблон функций становится обычным определением конкретной функции.

В списке параметров шаблона функций может быть несколько параметров. Каждый из них должен начинаться со служебного слова class. Например, допустим такой заголовок шаблона

*Имя параметра шаблона (в наших примерах* type1, type2*и т.д.) имеет в определяемой шаблоном функции все права имени типа*, т.е. с его помощью могут специализироваться формальные параметры, определяться тип возвращаемого функцией значения и типы любых объектов, локализованных в теле функции. Имя параметра шаблона видно во всем определении и скрывает другие использования того же идентификатора в области, глобальной по отношению к данному шаблону функций. Если внутри тела определяемой функции необходим доступ к внешним объектам с тем же именем, нужно применять операцию изменения области видимости. Следующая программа иллюстрирует указанную особенность имени параметра шаблона функций:

1. Каждый формальный параметр шаблона обозначается служебным словом class, за которым следует имя параметра (идентификатор). Пример определения шаблона функций, вычисляющих абсолютные значения числовых величин разных типов: template<class type> type
2. Да, меняя тип данных параметра, либо меняя параметры местами, в том случае, если они разного типа, а также можно выполнять перегрузку функции добавляя или исключая параметры.
3. Компонентные функции параметризованного класса автоматически являются параметризованными. Их не обязательно объявлять как параметризованные с помощью *template.* Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, т.е. по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону. Если *friend*-функция содержит в своем описании параметр типа параметризованного класса, то для каждого созданного по данному шаблону класса имеется собственная *friend*-функция. В рамках параметризованного класса нельзя определить *friend*-шаблоны (дружественные параметризованные классы). С одной стороны, шаблоны могут быть производными (наследоваться) как от шаблонов, так и от обычных классов, с другой стороны, они могут использоваться в качестве базовых для других шаблонов или классов. Шаблоны функций, которые являются членами классов, нельзя описывать как*virtual.* Локальные классы не могут содержать шаблоны в качестве своих элементов. Статические переменные шаблонов классов необходимо инициализировать для каждого используемого типа данных.
4. Компонентные функции параметризованного класса автоматически являются параметризованными. Их не обязательно объявлять как параметризованные с помощью template. Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, т.е. по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону.
5. Дружественные функции, которые описываются в параметризованном классе, не являются автоматически параметризованными функциями, т.е. по умолчанию такие функции являются дружественными для всех классов, которые организуются по данному шаблону. Если friend-функция содержит в своем описании параметр типа параметризованного класса, то для каждого созданного по данному шаблону класса имеется собственная friend-функция
6. В С++ шаблон функции не является функцией, поэтому шаблон не может быть виртуальным, а содержать виртуальную функцию может.

Однако когда дело доходит до виртуальной функции-члена класса шаблона, она кажется другой, потому что весь класс сам "создается" с помощью параметра шаблона, когда создается экземпляр переменной класса шаблона.

1. Компонентные функции параметризованных классов Реализация компонентной функции шаблона класса, которая находится вне определения шаблона класса, должна включать дополнительно следующие два элемента: • Определение должно начинаться с ключевого слова template, за которым следует такой же список\_параметров\_типов в угловых скобках, какой указан в определении шаблона класса.
2. Инстанцирование шаблона – это генерация кода функции или класса по шаблону для конкретных параметров.
3. Будет создан в время компиляции когда компилятор узнает, что вы использовали его для <int>, (Все остальное зависит от реализации компилятора)