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 № 6**

**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 класса-контейнера.

**Постановка задачи**

1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.
3. Перегрузить операции, указанные в варианте.
4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа.
5. Написать тестирующую программу, иллюстрирующую выполнение операций.

**Исходные данные**

Вариант 5:

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

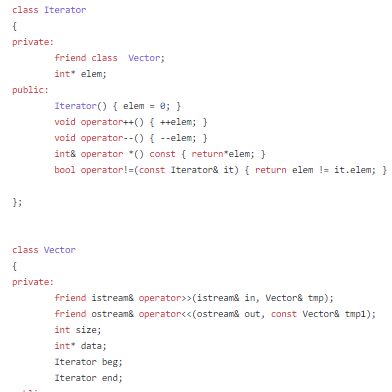
[] – доступа по индексу;

int() – определение размера вектора;

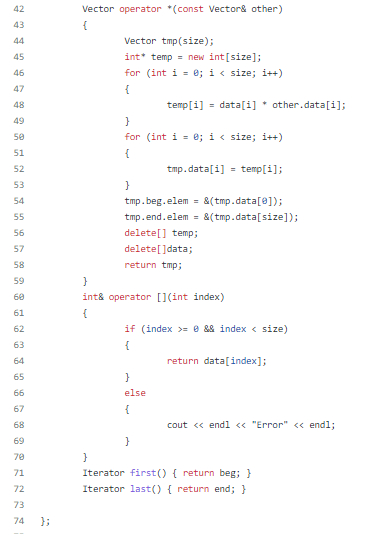
\* вектор – умножение элементов векторов a[i]\*b[i];

+ n – переход вправо к элементу с номером n (с помощью класса-итератора).

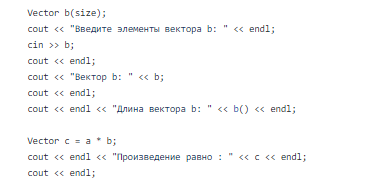
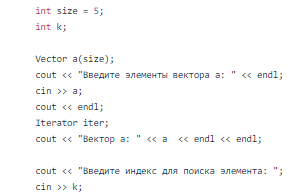
Создаю класс Iterator, у него в полях есть дин. массив, в public: прототипы перегруженных операторов и в private – ключевое слово friend для класса Vector, которой создаю сразу после первого класса. В классе Vector – присутствуют поля size, дин. Массив, и дружественные операторы ввода и вывода.

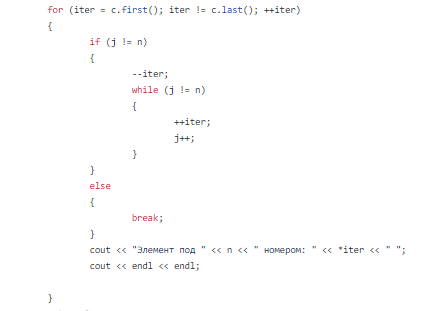


Здесь перегружаю мои операторы(Происходит в классе Vector):

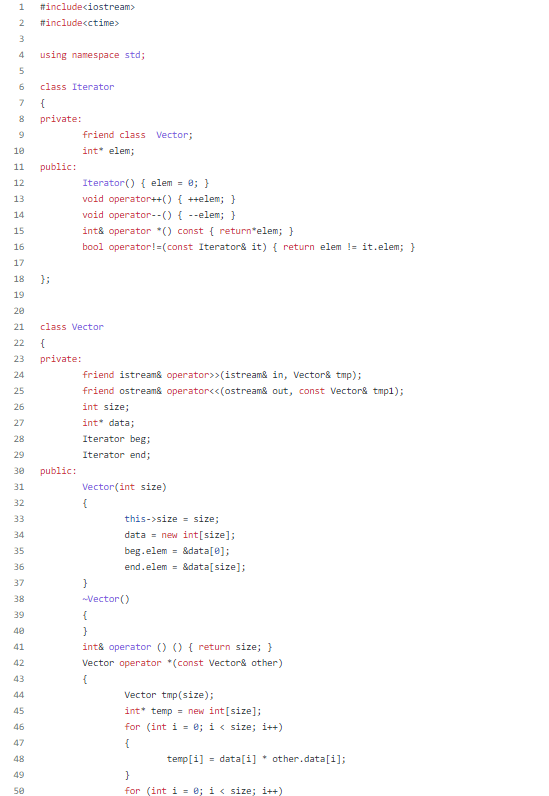


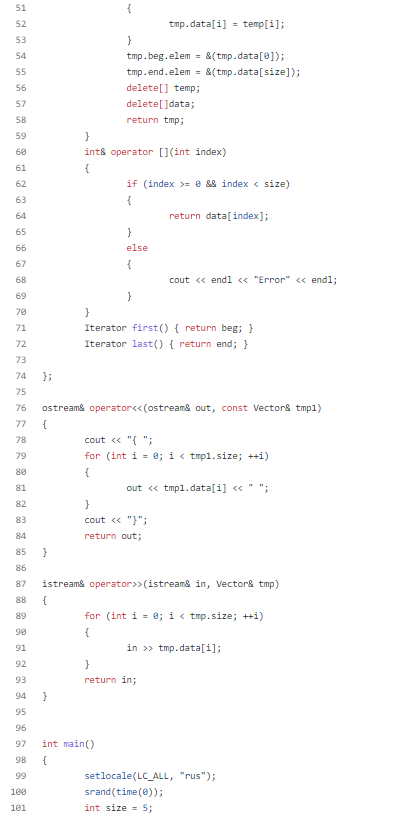
В main() создаю объекты типа Vector, Iterator, и вывожу по условию.



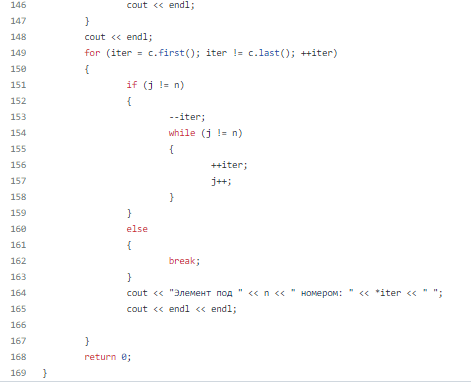


Код программы:









Результат программы:

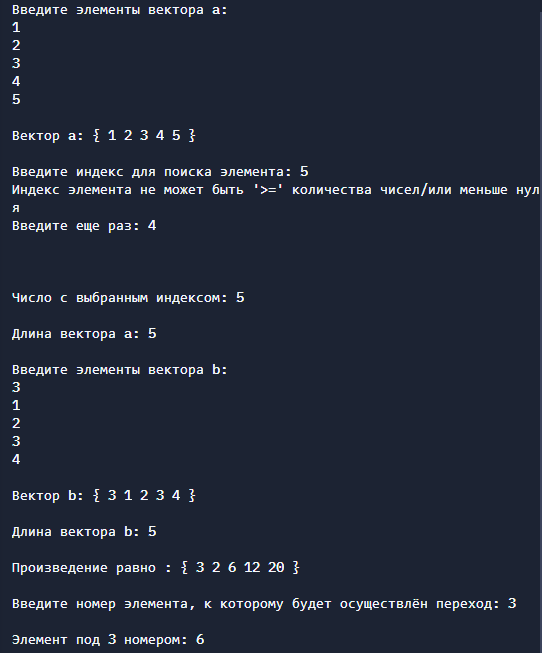


Диаграмма к задаче:

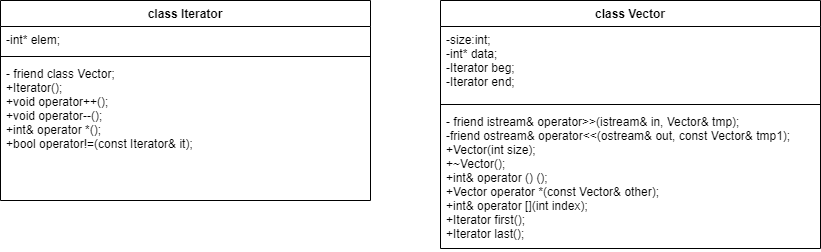


Рисунок 1

**Контрольные вопросы:**

* 1. Что такое абстрактный тип данных? Привести примеры АТД.
  2. Привести примеры абстракции через параметризацию.
  3. Привести примеры абстракции через спецификацию.
  4. Что такое контейнер? Привести примеры.
  5. Какие группы операций выделяют в контейнерах?
  6. Какие виды доступа к элементам контейнера существуют? Привести примеры.
  7. Что такое итератор?
  8. Каким образом может быть реализован итератор?
  9. Каким образом можно организовать объединение контейнеров?
  10. Какой доступ к элементам предоставляет контейнер, состоящий из элементов

«ключ-значение»?

* 1. Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?
  2. Какой из объектов (a,b,c,d) является контейнером?
     1. int mas=10;
     2. 2. int mas;
     3. 3. struct {char name[30]; int age;} mas;
     4. 4. int mas[100];
  3. Какой из объектов (a,b,c,d) не является контейнером? a. int a[]={1,2,3,4,5};

1. 2. int mas[30];
2. 3. struct {char name[30]; int age;} mas[30];
3. 4. int mas;
   1. Контейнер реализован как динамический массив, в нем определена операция доступ по индексу. Каким будет доступ к элементам контейнера?
   2. Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера

**Ответы на вопросы:**

1. Абстрактный тип данных (АТД) - это тип данных (набор значений и совокупность операций для этих значений), доступ к которому осуществляется только через интерфейс. Программу, которая использует АТД, будем называть клиентом, а программу, содержащую спецификацию этого типа данных - реализацией.
2. Абстракция через параметризацию позволяет, используя параметры, представить фактически неограниченный набор различных вычислений одной программой, которая есть абстракция этих наборов. Абстракцией через параметризацию почти все Вы пользовались много раз, даже не замечая этого. Возьмем наш пример с поиском максимального значения в массиве. Использовав эту процедуру однажды, вероятно, где-то в другом месте программы необходимо будет найти максимальный элемент в другом массиве, причем его имя может быть вовсе не а, а скажем z.

int aMax, a[100];

int zMax, z[100];

…

aMax = GetMaxValue (a);…Maz = GetMaxValue (z);

Следовательно, мы используем абстракцию через параметризацию, обобщая этим процедуру и делая ее более универсальной.

1. Абстракция через параметризацию позволяет, используя параметры, представить фактически *неограниченный набор различных вычислений одной программой*, которая есть абстракция этих наборов.

Абстракцией через параметризацию почти все Вы пользовались много раз, даже не замечая этого. Возьмем наш пример с поиском максимального значения в массиве. Использовав эту процедуру однажды, вероятно, где-то в другом месте программы необходимо будет найти максимальный элемент в другом массиве, причем его имя может быть вовсе не а, а скажем z.

int aMax, a[100];  
int zMax, z[100];

…

aMax = GetMaxValue (a);  
…  
zMaz = GetMaxValue (z);

Следовательно, мы используем абстракцию через параметризацию, обобщая этим процедуру и делая ее более универсальной.

1. Контейнер — это набор некоторого количества обязательно однотипных элементов, упакованных в контейнер определённым образом. Пример был в решении лаб.р.

Простейшим прототипом контейнера в классическом языке C++ является массив.

1. Доступ (чтение/запись) к элементу по его номеру (индексу) Добавление нового элемента в конец контейнера, начало контейнера, вставка нового элемента в середину. Удаление элемента из конца контейнера, из начала контейнера, из середины контейнера. Поиск элемента по его значению. Перебор элементов контейнера с выполнением какой-либо операции над каждым (итерирование)
2. Доступ к элементам контейнеров осуществляется с помощью итераторов. Итерация элементов в контейнере и доступ **к** отдельным элементам осуществляются с помощью итераторов. Итераторы можно использовать явно, используя функции и операторы их членов и глобальные функции.
3. **Итератор** — это объект, который может перебирать элементы в контейнере стандартной библиотеки **С++** и предоставлять доступ к отдельным элементам.
4. Аналог const указателя, аналог обычных указателей, может двигаться только в одну сторону, в обе стороны, и может предоставлять доступ к произвольным элементам последовательности.
5. Ассоциативные контейнеры можно объединить в два подмножества: сопоставления (set) и наборы (map). Контейнер map, который иногда называют словарем, состоит из пар "ключ-значение".
6. Этот контейнер хранит пары “ключ/значение”, в которых каждый ключ является уникальным. Это облегчает извлечение значения по заданному ключу. Доступ к элементам — в произвольном порядке
7. Если вставка и удаление выполняются только на одном конце контейнера, то такой контейнер называется стеком, а работает он в соответствии с дисциплиной обслуживания LIFO (Last In First Out — последним вошел, первым вышел).
8. d
9. d
10. С помощью итератора по индексу.
11. Он не поддерживает произвольный доступ к элементу по индексу. Из каждой ячейки можно обратиться только к соседней. То есть для списка не определена операция [].