Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

УТВЕРЖДАЮ
Зав.кафедрой,
к. ф.-м. н.
_____ М. В. Огнева

ОТЧЕТ О ПРАКТИКЕ

студента 1 курса 142 группы факультета КНиИТ
Бурдавицына Артёма Андреевича

вид практики: учебная
кафедра: информатики и программирования
курс: 1
семестр: 2(3)
продолжительность: 2 нед., с 29.06.2018 г. по 12.07.2018 г.

Руководитель практики от университета,

доцент, к. ф.-м. н. _____ К. П. Вахлаева

Тема практики:«Алгоритмы и структуры данных STL»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Библиотека стандартных шаблонов (STL) (англ. Standard Template Library) — набор согласованных обобщённых алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++.

В ходе практики были изучены такие инструменты библиотеки, как:

- контейнеры (ассоциативные, последовательные и адаптеры);
- алгоритмы;
- итераторы;
- функторы.

Библиотека STL является неотъемлемой частью языка C++, позволяющей сократить количество строк кода и затрат времени благодаря включенным в неё методам и классам. STL широко применяется в программировании, предоставляя программисту большой спектр возможностей, которыми удобно воспользоваться во многих ситуациях. Библиотека проста в применении, удобна и многофункциональна, поэтому разобраться в её работе полезно для программиста любого уровня.

1 История и общие сведения о библиотеке

Библиотека стандартных шаблонов до включения в стандарт C++ была сторонней разработкой, вначале — фирмы HP, а затем SGI. Стандарт языка не называет её «STL», так как эта библиотека стала неотъемлемой частью языка, однако многие люди до сих пор используют это название, чтобы отличать её от остальной части стандартной библиотеки.

Архитектура библиотеки была разработана Александром Степановым и Менг Ли. [1].

В библиотеке выделяют пять основных компонентов:

1. Контейнер (англ. container) — хранение набора объектов в памяти.
2. Итератор (англ. iterator) — обеспечение средств доступа к содержимому контейнера.
3. Алгоритм (англ. algorithm) — определение вычислительной процедуры.
4. Адаптер (англ. adaptor) — адаптация компонентов для обеспечения различного интерфейса.
5. Функциональный объект (англ. functor) — сокрытие функции в объекте для использования другими компонентами.

Здесь приведем пример, ссылок на литературу.

Первый список [2].

Второй [3]

Третий [4]

Четвертый [5]

Пятый [6]

Шестой [7]

Седьмой [8]

Восьмой [9]

Ничего страшного если вы вновь ссылаетесь на туже литературу. Он вас поймет [1]

2 Последовательные контейнеры

Последовательные контейнеры библиотеки STL помогают осуществлять хранение данных и доступ к ним. Их особенность заключается том, что они поддерживают указанный пользователем порядок вставляемых элементов. Среди последовательных контейнеров различают векторы, списки и деки (двусторонние очереди).

2.1 Вектор

Вектор, по сути, является стандартной реализацией двумерного массива, что позволяет хранить неограниченное количество элементов в одном массиве. Принцип работы вектора аналогичен принципу работы динамического массива. В векторе организован произвольный доступ к элементам, т.е к любому элементу можно обратиться по его индексу.

Подробная схема работы вектора представлена на рисунке 2.1.

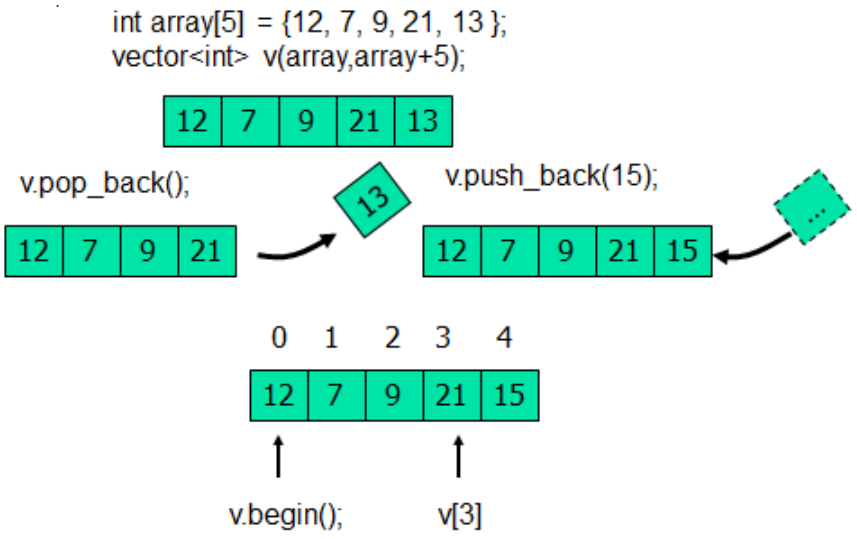

Рисунок 2.1 – Схема Вектора

Для работы с векторами существуют стандартные функции.

«Перечень функций представлены в таблице» 2.1.

Задача 1 (А). Дана последовательность целых чисел. Продублировать k-ый элемент

Устанавливаем итератор на k-й элемент. С помощью функции insert вставляем на k-ю позицию значение k-го элемента.

```
.....
 it = vector1.begin() + k;
```

Таблица 2.1 – Функции для работы с контейнером vector

Функция	Описание
.empty()	Проверка вектора на пустоту. Возвращает значение true, если вектор пуст
.front()	Возвращает значение первого элемента
.back()	Возвращает значение последнего элемента
.push-back()	Осуществляет вставку элемента в конец массива
.clear()	Очистка вектора
.pop-back()	Удаление последнего элемента
.size()	Получение количества элементов вектора
.begin()	Получение индекса начала вектора

```
.....
 vector1.insert(it - 1, *(it - 1));
.....
```

Полный код программы приведен в приложении А.

Задача 9 (Б). Дана последовательность целых чисел. Вставить после первого и перед последним элементы, получающиеся из исходных перестановкой цифр в обратном порядке.

Создаем вспомогательную функцию, переставляющую цифры числа в обратной порядке. Устанавливаем итератор на элемент, следующий после первого. Вставляем туда число полученное перестановкой цифр для первого числа.

```
.....
 it = vector1.begin() + 1;
                int a = vector1[0];
                a = invert(a);
                vector1.insert(it, a);
                ++n;
.....
```

Аналогично для вставки числа перед последним элементом.

Полный код программы приведен в приложении А.

2.2 Список

Список (list) представляет собой последовательный контейнер, который поддерживает быструю вставку и удаление элементов из любой позиции в кон-

тейнере. Список отличается от вектора тем, что доступ к элементам осуществляется последовательно, т.е чтобы добраться к нужному элементу, необходимо перебрать все элементы до него.

Подробная схема работы списка представлена на рисунке 2.2.

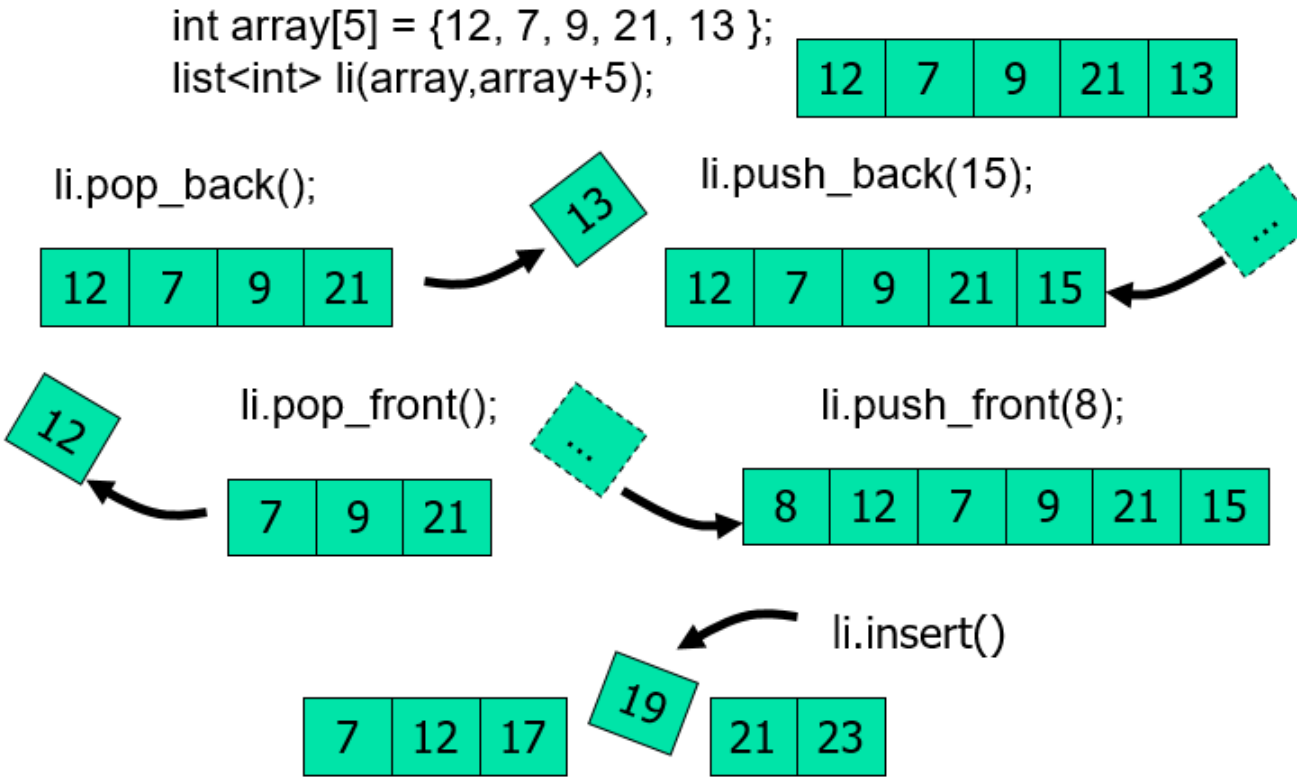

Рисунок 2.2 – Схема Списка

Для работы с списками существуют стандартные функции.
«Перечень функций представлены в таблице» 2.2.

Таблица 2.2 – Функции для работы с контейнером list

Функция	Описание
`.empty()`	`Проверка на пустоту. Возвращает значение true, если список пуст`
`.front()`	`Возвращает значение первого элемента`
`.back()`	`Возвращает значение последнего элемента`
`.push-back()`	`Осуществляет вставку элемента в конец`
`.push-front()`	`Осуществляет вставку элемента в начало`
`.pop-front()`	`Удаление первого элемента`
`.clear()`	`Очистка списка`
`.pop-back()`	`Удаление последнего элемента`
`.size()`	`Получение количества элементов списка`
`.erase()`	`Удаление выбранного элемента`

ЗАДАЧА 8 (А). Заменить средний элемент на сумму первого и последнего,

если количество элементов нечетное.

Находим сумму первого и последнего элементов как сумму указателей на соответствующие итераторы.

```
.....
 it1 = --list1.end();
 it2 = list1.begin();
 int a = *it1 + *it2;
.....
```

С помощью функции erase удаляем средний элемент, на его места вставляем ранее найденную сумму.

```
.....
 it = list1.begin();
 advance(it, n / 2);
 list1.erase(it);
 it = list1.begin();
 advance(it, n / 2);
 list1.insert(it, a);
.....
```

Полный код программы приведен в приложении Б.

Задача 4 (Б). Заменить последний и первый элементы на средний

Устанавливаем итератор на средний элемент. Находим значение среднего элемента как указатель на данный итератор.

```
.....
 it = list1.begin();
 advance(it, n / 2);
 int a = *it;
.....
```

Устанавливаем итератор на первый элемент. С помощью erase удаляем его. На его места вставляем нужное значение.

```
.....
 it = list1.begin();
                 list1.erase(it);
                 it = list1.begin();
                 list1.insert(it, a);
.....
```

Аналогично для замены последнего элемента.

Полный код программы приведен в приложении Б.

2.3 Дек

Дек (deque - Double-Ended-Queue) — двусторонняя очередь, т.е очередь, каждая ячейка которой содержит в себе указатель как на предыдущий, так и на следующий элемент. Это позволяет очень быстро добавлять новые элементы как в конец так и в начало контейнера.

Принцип работы дека аналогичен принципу работы списка.

Для работы с деками существуют стандартные функции.

Перечень функций представлены в таблице 2.3.

Таблица 2.3 – Функции для работы с контейнером deque

Функция	Описание
.empty()	Проверка на пустоту. Возвращает значение true, если дек пуст
.front()	Возвращает значение первого элемента
.back()	Возвращает значение последнего элемента
.push-back()	Осуществляет вставку элемента в конец
.push-front()	Осуществляет вставку элемента в начало
.pop-front()	Удаление первого элемента
.clear()	Очистка списка
.pop-back()	Удаление последнего элемента
.size()	Получение количества элементов списка
.erase()	Удаление выбранного элемента

Задача 17. Создать список из слов. Исключить из списка повторяющиеся слова.

С помощью двух вложенных циклов проходим по списку и ищем повторяющиеся слова. При нахождении каждого из них с помощью erase удаляем второе повторяющееся слово, уменьшаем общее количество элементов и счетчик второго цикла на 1, с помощию переменной flag типа bool обозначаем,

что необходимо удалить первый из повторяющихся элементов. Аналогично удаляем его.

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

         deque <string>::iterator it;
         bool flag = false;

         for (int j = i+1; j < n; ++j) {
                 if (deque1[i] == deque1[j]) {
                         it = deque1.begin();
                         advance(it, j);
                         deque1.erase(it);
                         --n;
                         --j;
                         flag = true;
                 }
         }
.....
```

Полный код программы приведен в приложении B.

ЗАДАЧА 18. Создать список из целых чисел. Создать новый список, записав в него вначале все положительные числа, а затем все отрицательные числа из исходного списка.

Создаем второй список. С помощью функций pushback и pushfront записываем во второй список отрицательные и положительные числа соответственно.

```
  .....
   it = deque1.begin();

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

                   int a = *it;
                   if (a > 0) deque2.push_front(a);
                   else if (a < 0) deque2.push_back(a);
```

```
                advance(it, 1);

        }
.....
```

Полный код программы приведен в приложении В.

3 Итераторы, алгоритмы, функторы

Итератор — это объект, который может выполнять итерацию элементов в контейнере STL и предоставлять доступ к отдельным элементам. Все контейнеры STL предоставляют итераторы, чтобы алгоритмы могли получить доступ к своим элементам стандартным способом, независимо от типа контейнера, в котором сохранены элементы.

Вы можете использовать итераторы явно, с помощью члена и глобальных функций, таких как begin() и end(), а также операторов ++ и – для перемещения вперед или назад. Вы можете также использовать итераторы неявно, с циклом range-for или (для некоторых типов итераторов) подстрочным оператором [].

В STL началом последовательности или диапазона является первый элемент. Конец последовательности или диапазона всегда определяется как элемент, следующий за последним элементом.

Подробная схема работы списка представлена на рисунках 3.1, 3.2, 3.3.

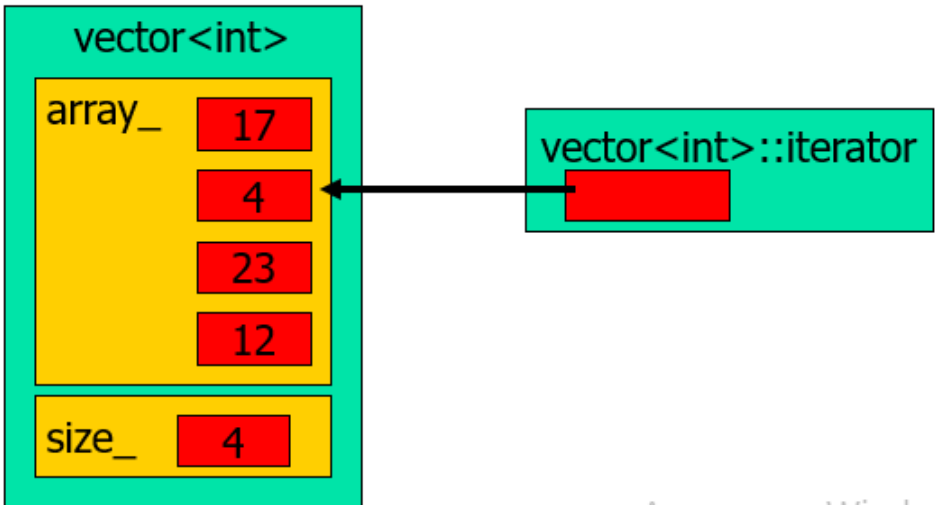

Рисунок 3.1 – Схема Итератора 1

Алгоритмы STL универсальны, поскольку они могут работать с различными структурами данных. Структуры данных, с которыми они могут работать, включают в себя не только классы контейнеров STL, но также определенные программой структуры данных и массивы элементов, удовлетворяющие требованиям определенного алгоритма. Алгоритмы STL достигают такого уровня универсальности путем получения доступа к элементам контейнера и их просмотра опосредованным образом через итераторы.

Алгоритмы STL обрабатывают диапазоны итератора, которые обычно

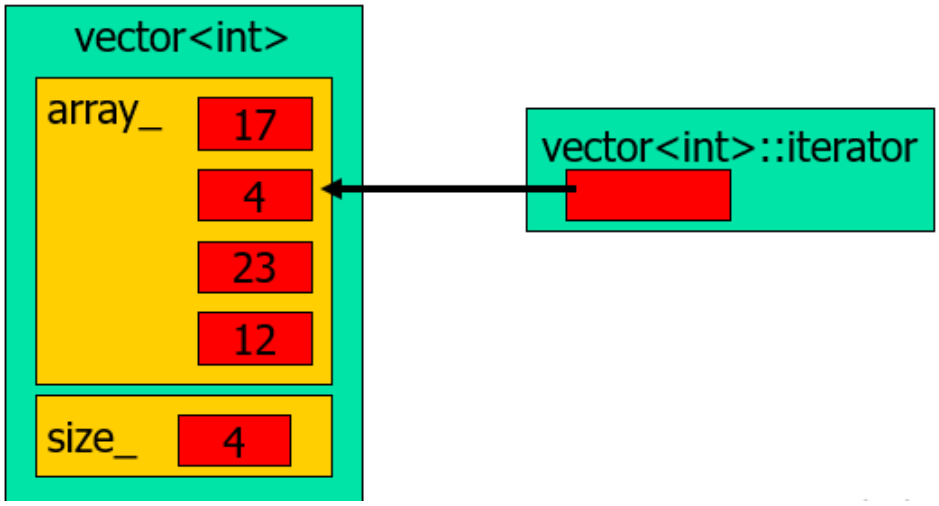

Рисунок 3.2 – Схема Итератора 2

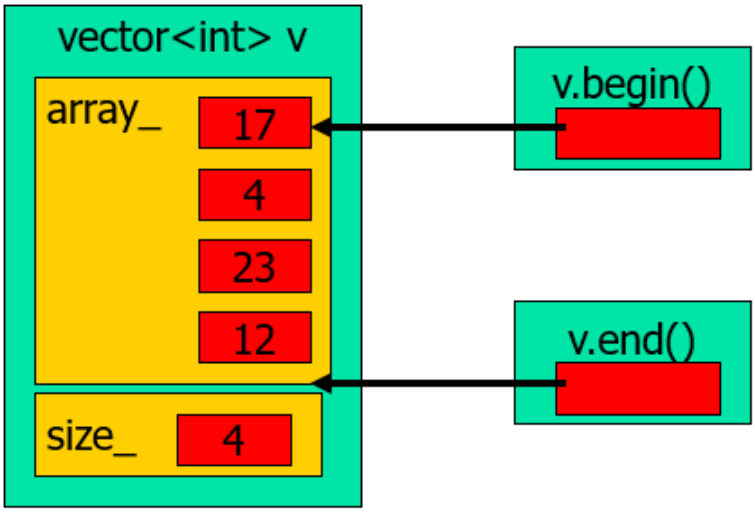

Рисунок 3.3 – Схема Итератора 3

определяются их начальными или конечными позициями. Указанные диапазоны должны быть допустимы в том смысле, что все указатели в диапазонах должны поддерживать удаление ссылок и в рамках последовательностей каждого диапазона последняя позиция должна быть доступна из первой путем приращения.

Алгоритмы STL расширяют действия, поддерживаемые операциями и функциями-членами каждого контейнера STL, и позволяют работать, например, с различными типами объектов контейнера одновременно.

Перечень наиболее часто использующихся алгоритмов представлен в таблице 3.1.

Для большинства алгоритмов существуют их аналоги с суффиксами if и copy.

Таблица 3.1 – Некоторые функции для работы с algorithm

Функция	Описание
.sort()	Сортировка контейнера по возрастанию некоторого поля
.erase()	Очищает элементы из заданного диапазона
.count()	Считает количество элементов данного диапазона
.remove()	Удаляет из диапазона все значения, равные заданному
.binary-search()	Возвращает true, если заданное значение входит в интервал
.lower_bound()	Возвращает итератор, указывающий на первую позицию, в которую можно вставить заданное значение без изменения порядка следования объектов
.upper_bound()	Возвращает итератор, указывающий на последнюю позицию, в которую можно вставить заданное значение без изменения порядка следования объектов
set_difference()	Создает упорядоченную разность множеств, заданных диапазонами 1 и 2
set_intersection()	Создает упорядоченное пересечение множеств, заданных диапазонами 1 и 2

Суффикс if указывает, что алгоритм работает с объектами функций, действующих для значений элементов, а не с самими значениями элементов. Например, алгоритм find-if выполняет поиск элементов, значения которых удовлетворяют критерию, заданному объектом функции, а алгоритм find ищет определенное значение.

Суффикс copy означает, что этот алгоритм не только манипулирует значениями элементов, но также копирует измененные значения в диапазон назначения. Например, алгоритм reverse изменяет порядок элементов в диапазоне на обратный, а алгоритм reverse-copy также копирует результат в диапазон назначения.

Для их использования необходимо добавить предикат, принимающий булевые значения, как дополнительный параметр.

Функтор — это сокращение от функциональный объект, представляющий собой конструкцию, позволяющую использовать объект класса как функцию. В C++ для определения функтора достаточно описать класс, в котором переопределена операция ().

Задача 1. Точки на плоскости заданы парами целочисленных координат.

а) удалить все точки из I четверти

б) подсчитать количество точек, у которых x и y совпадают

в) найти наиболее удалённую от начала координат точку

г) расположить в порядке возрастания координаты x

Создадим структуру Point для хранения координат введенных точек.

```
.....
 struct Point {
 public:
         int x;
         int y;

         Point(int x, int y) {

                 this->x = x;
                 this->y = y;

         }

 };
 .....
```

Для пункта а) необходимо использовать функции removeif и erase. Removeif удаляет из заданного диапазона элементы, соответствующие некоторому условию, и возвращает итератор, указывающий на начало. Erase удаляет элементы из заданного диапазона.

```
.....
 point.erase(remove_if(point.begin(), point.end(), f), point.end());
 .....
```

Для пункта б) необходимо использовать функцию countif, возвращающую количество элементов из заданного диапазона, удовлетворяющие некоторому условию.

```
.....
 s = count_if(point.begin(), point.end(), f1);
 .....
```

Для пункта в) необходимо использовать функцию maxelement, устанавливающую итератор на элемент из заданного диапазона, удовлетворяющие некоторому условию.

```
.....
 i = max_element(point.begin(), point.end(), compare);
 .....
```

Для пункта г) необходимо использовать функцию sort, сортирующую элементы из заданного диапазона по возрастанию первой координаты.

```
.....
 i = max_element(point.begin(), point.end(), compare);
 .....
```

Полный код программы приведен в приложении Г.

4 Контейнеры-адаптеры

Контейнер-адаптер — это разновидность последовательного или ассоциативного контейнера, который ограничивает интерфейс для простоты и ясности. Контейнеры-адаптеры не поддерживают итераторы, поэтому их нельзя использовать для работы с алгоритмами STL. Среди контейнеров-адапткров различают стек и очередь.

4.1 Стек

Стек (stack — стопка) — абстрактный тип данных, имеющих одну точку доступа (голову), т.е представляющий собой список элементов, организованных по принципу LIFO (last in — first out, «последним пришёл — первым вышел»).

Подробная схема работы стека представлена на рисунке 4.1.

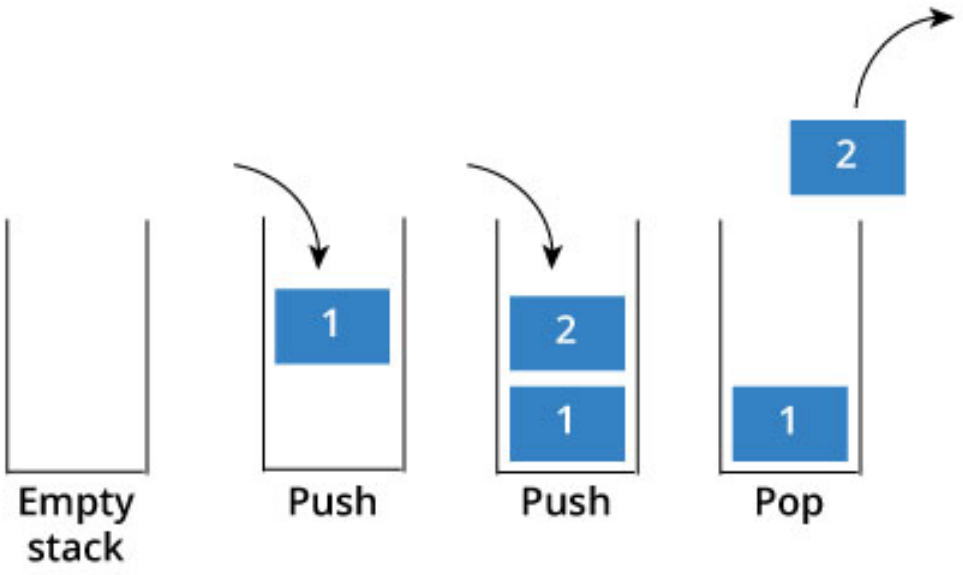

Рисунок 4.1 – Схема Стека

Для работы со стеками существуют стандартные функции.
Перечень функций представлен в таблице 4.1.

Задача 1 (A). Удалить первый кратный трём элемент.

Создаем 2 стека. В первый стек записываем все числа. Для того, чтобы удалить нужный элемент, с помощью переменной типа bool отмечаем первый кратный трем элемент, чтобы удалить только его.

Таблица 4.1 – Функции для работы с контейнером stack

Функция	Описание
.empty()	Проверка на пустоту. Возвращает значение true, если стек пуст
.top()	Возвращает значение первого элемента
.pop()	Удаляет первый элемент
.push()	Осуществляет вставку элемента в начало
.size()	Получение количества элементов стека

```
.....
 if (a % 3 == 0 && f == false) {

                        num = i;
                        f = true;

                }
.....
```

Переписываем второй стек только нужные элементы из первого стека. Полный код программы приведен в приложении Д.

Задача 17 (Б). Заменить первый, средний и последний элемент на единицы, если они простые.

Создаем вспомогательную функцию для проверки числа на простоту. В первый стек записываем все числа. В цикле переписываем элементы из первого стека во второй и если номер элемента соответствует условию, то меняем его на единицу.

```
.....
   while (!stack1.empty()) {

                    a = stack1.top();
                    stack1.pop();

                    if ((i == 1 || i == n || i == mid_num) && ifNotPrime(a) == false)
                            a = 1;

                            stack2.push(a);

                    ++i;
```

```
            }
.....
```

Полный код программы приведен в приложении Д.

4.2 Очередь

Очередь (queue) — абстрактный тип данных, имеющих 2 точки доступа (голову и хвост), т.е представляющий собой список элементов, организованных по принципу FIFO (first in — first out, «первым пришёл — первым вышел»).

Подробная схема работы стека представлена на рисунках 4.2.

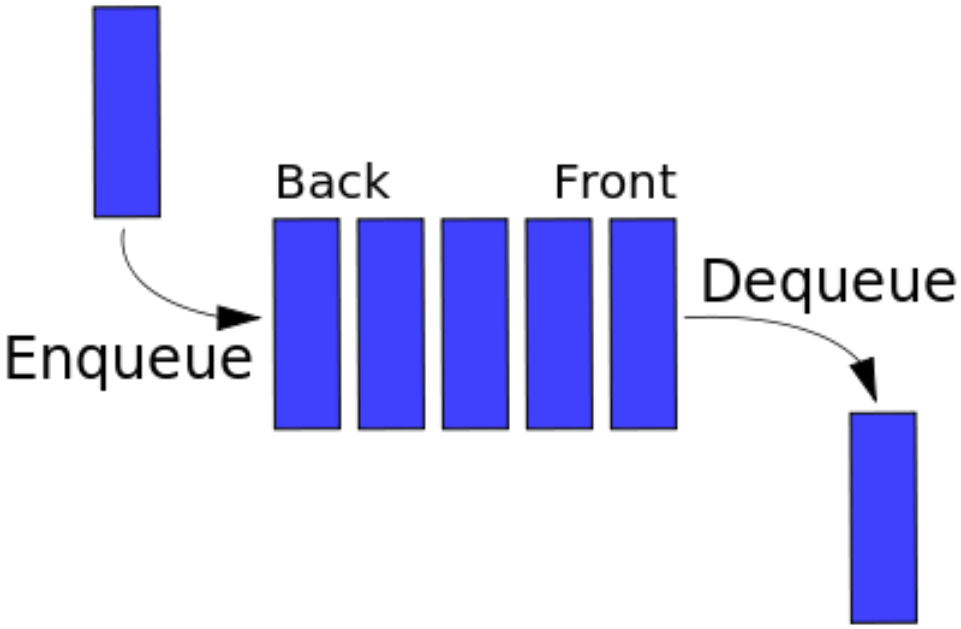

Рисунок 4.2 – Схема Очереди

Для работы с очередью существуют стандартные функции.

Перечень функций представлен в таблице 4.2.

Задача 10. Дано N очередей. С ними выполняют K операций PUSH(I, V), TOP(I), POP(I) – добавить в хвост очереди I число V, показать число в голове очереди I и удалить число из головы очереди I (при этом результат показывается). Результат должен содержать столько чисел, сколько операций TOP и POP было сделано.

Создаем динамический массив очередей.

Таблица 4.2 – Функции для работы с контейнером queue

Функция	Описание
.empty()	Проверка на пустоту. Возвращает значение true, если очередь пуста
.pop()	Удаляет последний элемент
.push()	Осуществляет вставку элемента в начало
.size()	Получение количества элементов очереди
.front()	Возвращает первый элемент
.back()	Возвращает последний элемент

```
.....
    queue<int> **QArray = new queue<int> *[a]; // массив указателей на очереди
            for (int i = 0; i < a; i++)
            {
                    QArray[i] = new queue<int>; // создание объекта очередь
            }
.....
```

Из входного файла считываем команды. По нескольким первым буквам распознаем тип команды (PUSH, POP или TOP). В зависимости от типа команды посимвольно считываем одно или два числа. С помощью функции atoi преобразуем переменные с числами типа string к int. Делаем с соответствующей очередью соответствующие действия.

Например, для функции POP:

```
.....
    if (str1 == "POP") {

                            string str2;
                            int i = 4;

                            while (i < str.length() && str[i] != ')') {

                                    str2 += str[i];
                                    ++i;

                            }

                            int num = 0;
                            num = atoi(str2.c_str());

                            out << QArray[num-1]->front() << " ";
```

```
                QArray[num-1]->pop();

            }
```

. . . . .

Аналогично для других функций.

Полный код программы приведен в приложении Д.

5 Ассоциативные контейнеры

В ассоциативных контейнерах элементы вставляются в предварительно определенном порядке — например, с сортировкой по возрастанию. Также доступны неупорядоченные ассоциативные контейнеры. Ассоциативные контейнеры можно объединить в два подмножества: сопоставления (set) и наборы (map).

5.1 Множество

Set и multiset реализуют такие сущности как множество и мультимножество. Они содержат некоторое количество отсортированных элементов. При добавлении нового элемента в множество он сразу становится на свое место так, чтобы не нарушать порядка сортировки. Потому как в множестве и мультимножестве все элементы сортируются автоматически. Множества содержат только уникальные элементы, а мультимножества могут содержать дубликаты.

Подробная схема работы сета представлена на рисунке 5.1.

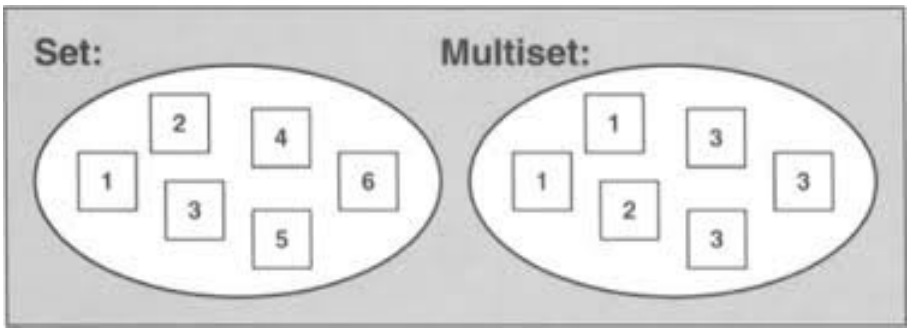

Рисунок 5.1 – Схема Сета

Для работы с сетами существуют стандартные функции.

Перечень функций представлен в таблице 5.1.

Таблица 5.1 – Функции для работы с контейнером set

Функция	Описание
`.empty()`	`Проверка на пустоту. Возвращает значение true, если сет пуст`
`.size()`	`Получение количества элементов сета`
`.erase()`	`Удаление элемента из сета`
`.insert()`	`Вставляет элемент в сет`
`.clear()`	`Очищает сет`
`.count()`	`Возвращает количество элементов с заданным ключем`
`.find()`	`Возвращает элемент с заданным ключем`

ЗАДАЧА 3. Найти все такие цифры, которые встречаются только в кратных 10 числах.

Создаем вектор, в котором будут храниться исходные числа. Создаем 2 сета. В первый будем записывать цифры всех чисел, не кратных 10.

```
.....
 for (int i = 0; i < v.size(); i++)
         {

                 if (v[i] % 10 != 0) {

                         int k = abs(v[i]);
                         while (k > 0)
                         {
                                 int a = k % 10;
                                 k = k / 10;
                                 set1.insert(a);

                         }
                 }
         }
 .....
```

Во второй сет будем записывать все цифры из чисел не кратных 10, не встречающиеся в первом сете.

```
.....
for (int i = 0; i < v.size(); ++i) {

                if (v[i] % 10 == 0) {

                        int k = abs(v[i]);

                        while (k != 0) {

                                int a = k % 10;
                                bool f = false;

                                for (auto it = set1.begin(); it != set1.end(); ++it)
```

```
                                        if (*it == a) {

                                                f = true;
                                                break;

                                        }

                                }

                                if (f == false) set2.insert(a);

                                k /= 10;

                        }

                }

        }
.....
```

Полный код программы приведен в приложении E.

ЗАДАЧА 7. Дан текст, состоящий из предложений, разделённых знаками препинания из набора «.?!». Предложения в свою очередь состоят из слов, отделённых друг от друга пробелами. Найти слова (без учёта регистра) и их количество, которые встречаются только в повествовательных и вопросительных предложениях.

Создаем 3 сета. В первый сет будем складывать слова из восклицательных предложений без учёта регистра.

```
.....
if (str[str.length() - 1] == '!') {

                        string str_buf;
                        int j = 0;
                        while (j < str.length()) {

                                if (isalpha(str[j])) {
                                        str[j] = tolower(str[j]);
                                        str_buf += str[j];
```

```
                        }
                        else {
                                set1.insert(str_buf);
                                str_buf.clear();
                        }
                        ++j;
                }
        }
.....
```

Во второй сет будем складывать слова из повествовательных и вопросительных предложений.

```
.....
else {

                string str_buf;
                int j = 0;
                while (j < str.length())  {

                        if (isalpha(str[j])) {
                                str[j] = tolower(str[j]);
                                str_buf += str[j];
                        }
                        else {
                                set2.insert(str_buf);
                                str_buf.clear();
                        }
                        ++j;
                }
        }
.....
```

С помощью функции set-difference помещаем в третий сет уникальные слова из втрого сета (слова из второго сета, не встречающиеся в первом).

```
.....
set_difference(set2.begin(), set2.end(), set1.begin(), set1.end(),
inserter(set3, set3.begin()));
.....
```

Полный код программы приведен в приложении E.

5.2 Отображение

Отображение (map) — отсортированный ассоциативный контейнер, который содержит пары ключ-значение с неповторяющимися ключами. Это отличие от остальных контейнеров делает его пригодным для создания хеш-таблиц.

Подробная схема работы map представлена на рисунке 5.2.

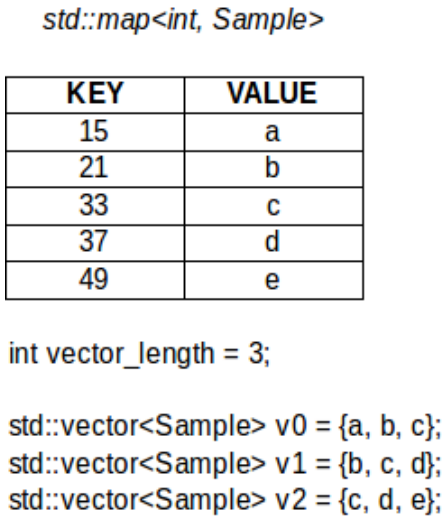

Рисунок 5.2 – Схема Map

Для работы с map существуют стандартные функции, аналогичные функциям работы с сетами.

Задача 10. Во входном файле задан набор слов и целых чисел, разделённых пробелами. Найти все слова, встречающиеся столько же раз, сколько последнее число.

Создаем 1 map. Увеличиваем на 1 значение ячейки с номером, соответствующим очередному входному элементу, и запоминаем последнее число.

```
 .....
while (in >> tmp)
        {
                words[tmp]++;
                if (isdigit(tmp[0]))
                        str_num = tmp;
        }

        int num = atoi(str_num.c_str());
 .....
```

С помощью итераторов в цикле проходим по map и ищем очередное слово, встречающееся необходимое количество раз. Кладем его в вектор.

```
  .....
for (auto i = words.begin(); i != words.end(); ++i) {

                string st = i->first;
                int st2 = i->second;
                cout << st << " " << st2 << endl;

                if (isalpha(st[0]) && st2==num) {

                        vec_ans.push_back(st);

                }

        }
  .....
```

Если вектор не пуст, то выводим его.

Полный код программы приведен в приложении Ж.

6 Использование контейнеров и алгоритмов STL

Здесь важно написать условие задачи. Как планируете реализовать, ну и соответственно фрагменты кода.

ЗАКЛЮЧЕНИЕ

В ходе практики были изучены элементы стандартной библиотеки шаблонов. Были использованы такие контейнеры как `vector`, `list`, `deque`, `stack`, `queue`, `set`, `map`, `multiset`, `multimap`. Использование данных контейнеров существенно упростило и ускорило процесс решения задач, что делает библиотеку STL необходимой для изучения.

Таким образом, библиотека STL — это применяемые на практике знания, которые помогают значительно уменьшить размер кода программы и быстро оптимизировать его. Универсальность STL является одновременно преимуществом и слабой стороной, т.к.....

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Аммерааль,* . STL для программистов на C++ / . Аммерааль. — Москва: «ДМК», 1999. — С. 240.

2 The C++ Resources Network [Электронный ресурс]. — URL: http://www.cplusplus.com/reference/stl/ (Дата обращения 15.07.2016).

3 *Джосаттис,* . . Стандартная библиотека C++: справочное руководство / . . Джосаттис. — Москва: ООО «И. Д. Вильямс», 2014. — С. 1136.

4 *Саттер,* . Новые сложные задачи на C++ / . Саттер. — Москва: Издательский дом «Вильямс», 2005. — С. 272.

5 *Саттер,* . Новые сложные задачи на C++. Серия C++ In-Depth / . Саттер. — Москва: Издательский дом «Вильямс», 2008. — С. 400.

6 *Скотт Мейерс,*. Эффективное использование C++. Библиотека программиста / Скотт Мейерс. — СПб: Питер, 2002. — С. 224.

7 *Огнёва,* . . Основы программирования на C++: Учеб. пособие в 2 ч. Часть 2. / . . Огнёва, . . Кудрина. — Саратов: ООО Издательский Центр «Наука», 2009. — С. 100.

8 *Огнёва,* . . Структуры даннх и алгоритмы:программирование на языке C++: Учеб. пособие в 2 ч. Часть 1. / . . Огнёва, . . Кудрина. — Саратов: ООО Издательский Центр «Наука», 2013. — С. 88.

9 Электронный курс: Ознакомительная (учебная) практика МОАИС [Электронный ресурс]. — URL: https://course.sgu.ru/course/view.php?id=1022 (Дата обращения 15.07.2016).

ПРИЛОЖЕНИЕ А

Вектор

Задача 1 (А). Дана последовательность целых чисел. Продублировать k-ый элемент

Листинг vector1A.cpp

```
#include<iostream>
#include<vector>

using namespace std;

int main()
{
        int n; cin >> n;
        vector <int> vector1;
        for (int i = 0; i < n; ++i) {

                int a; cin >> a;
                vector1.push_back(a);



        }
        vector <int>::iterator it;
        cout << "K = ";
        int k; cin >> k;

        it = vector1.begin() + k;

        if (k < n) {

                vector1.insert(it - 1, *(it - 1));
                ++n;

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

```
                    cout << vector1[i] << " ";

                }
        }
        else cout << "error" << endl;

        return 0;
}
c
```

Пример:

Вход	Выход
9 1 2 3 4 5 6 7 8 9 K = 3	1 2 3 3 4 5 6 6 7 8 9 9
9 1 2 3 4 5 6 7 8 9 K = 1	1 1 2 3 4 5 6 6 7 8 9 9

ЗАДАЧА 9 (Б). Дана последовательность целых чисел. Вставить после первого и перед последним элементы, получающиеся из исходных перестановкой цифр в обратном порядке.

Листинг `vector9B.cpp`

```
#include<iostream>
#include<vector>

using namespace std;

int invert(int a) {

        int a1 = a, s = 0;

        while (a1 != 0) {

                a1 /= 10;
```

```
                ++s;

        }

        --s;
        a1 = 0;

        while (a != 0) {

                a1 += a % 10 * pow(10, s);
                a /= 10;
                --s;

        }

        return a1;

}

int main()
{
        int n; cin >> n;
        vector <int> vector1;
        for (int i = 0; i < n; ++i) {

                int a; cin >> a;
                vector1.push_back(a);


        }
        vector <int>::iterator it;

        if (n != 0 && n != 1) {
                it = vector1.begin() + 1;
```

```
        int a = vector1[0];
        a = invert(a);
        vector1.insert(it, a);
        ++n;

        it = vector1.end() - 1;
        a = vector1[n - 1];
        a = invert(a);
        vector1.insert(it, a);
        ++n;
}
else if (n == 1) {

        it = vector1.begin() + 1;
        int a = vector1[0];
        a = invert(a);
        vector1.insert(it, a);
        ++n;

        it = vector1.begin();
        a = vector1[0];
        a = invert(a);
        vector1.insert(it, a);
        ++n;

}

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

        cout << vector1[i] << " ";

}

return 0;
```

```
}
c
```

Пример:

Вход	Выход
9 12 2 3 4 5 6 7 8 93	12 21 3 4 5 6 7 8 39 93
9 98 8 7 6 5 4 3 2 12	98 89 8 7 6 5 4 3 21 12

ПРИЛОЖЕНИЕ Б
Список

ЗАДАЧА 8 (А). Заменить средний элемент на сумму первого и последнего, если количество элементов нечетное.

Листинг `list8A.cpp`

```
#include<iostream>
#include<list>
#include <iterator>

using namespace std;

int main()
{
        int n; cin >> n;
        list <int> list1;

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

                int a; cin >> a;
                list1.push_back(a);

        }

        if (n % 2 != 0 && n!=1) {

                list <int>::iterator it;
                it = list1.begin();
                advance(it, n / 2);
                list <int>::iterator it1;
                it1 = --list1.end();
                list <int>::iterator it2;
                it2 = list1.begin();
```

```
                int a = *it1 + *it2;

                list1.erase(it);

                it = list1.begin();
                advance(it, n / 2);
                list1.insert(it, a);

        }

        else if (n == 1) {

                list <int>::iterator it;
                it = list1.begin();
                int a = 2 * *it;

                list1.erase(it);

                it = list1.begin();
                list1.insert(it, a);


        }

        copy(list1.begin(), list1.end(), ostream_iterator<int>(cout, " "

        return 0;
}
```

Пример:

Задача 4 (Б). Заменить последний и первый элементы на средний

Листинг `list4B.cpp`

```
#include<iostream>
#include<list>
```

Вход	Выход
5 5 4 7 0 1	5 4 6 0 1
6 6 8 0 6 3 1	6 8 0 6 3 1
1 1	2

```
#include <iterator>

using namespace std;

int main()
{
        int n; cin >> n;
        list <int> list1;

        if (n != 0) {

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

                        int a; cin >> a;
                        list1.push_back(a);

                }

                list <int>::iterator it;
                it = list1.begin();

                advance(it, n / 2);

                int a = *it;

                it = list1.begin();
                list1.erase(it);
                it = list1.begin();
                list1.insert(it, a);
```

```
                it = --list1.end();
                list1.erase(it);
                it = list1.end();
                list1.insert(it, a);


                copy(list1.begin(), list1.end(), ostream_iterator<int>(c
        }

        return 0;
}
c
```

Пример:

Вход	Выход
5 9 6 89 5 1	89 6 89 5 89
6 42 3 5 2 1 4	2 3 5 2 1 2

ПРИЛОЖЕНИЕ В

Дек

ЗАДАЧА 17. Создать список из слов. Исключить из списка повторяющиеся слова.

Листинг deque17.cpp

```
#include <iostream>
#include <deque>
#include <iterator>
#include <fstream>
#include <string>
#include <algorithm>

using namespace std;

ifstream in("input.txt");

int main()
{
        deque <string> deque1;
        int n = 0;

        while (in.peek() != EOF) {

                string str;
                in >> str;
                deque1.push_back(str);
                ++n;

        }

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

        deque <string>::iterator it;
```

```
        bool flag = false;

        for (int j = i+1; j < n; ++j) {
                if (deque1[i] == deque1[j]) {
                        it = deque1.begin();
                        advance(it, j);
                        deque1.erase(it);
                        --n;
                        --j;
                        flag = true;
                }
        }

        if (flag == true) {
                it = deque1.begin();
                advance(it, i);
                deque1.erase(it);
                --n;
                i = -1;
                ++s;
                flag = false;
        }
}

        copy(deque1.begin(), deque1.end(), ostream_iterator<string>(cout
        cout << endl << endl;

        in.close();

        return 0;
}
```

Пример:

ЗАДАЧА 18. Создать список из целых чисел. Создать новый список, запи-

Вход	Выход
cat dog pie cat cat	dog pie
cat dog pie	cat dog pie
cat cat	

сав в него вначале все положительные числа, а затем все отрицательные числа из исходного списка.

Листинг deque18.cpp

```
#include <iostream>
#include <deque>
#include <iterator>
#include <fstream>

using namespace std;

ifstream in("input.txt");

int main()
{
        deque <int> deque1, deque2;
        int n = 0;

        while (in.peek() != EOF) {

                int a; in >> a;
                deque1.push_back(a);
                ++n;

        }

        copy(deque1.begin(), deque1.end(), ostream_iterator<int>(cout, "
        cout << endl << endl;
        int s = 0;
```

```
        deque <int>::iterator it;
        it = deque1.begin();


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

                int a = *it;
                if (a > 0) deque2.push_front(a);
                else if (a < 0) deque2.push_back(a);

                advance(it, 1);

        }

        copy(deque2.begin(), deque2.end(), ostream_iterator<int>(cout, "
        cout << endl << endl;

        in.close();

        return 0;
}
```

Пример:

Вход	Выход
1 -1 2 -2 3 -3 4 -4 5 -5 6 -6	6 5 4 3 2 1 -1 -2 -3 4- 5 -6
-1 0 1	1 -1

ПРИЛОЖЕНИЕ Г
Алгоритмы

ЗАДАЧА 1. Точки на плоскости заданы парами целочисленных координат.

а) удалить все точки из I четверти

б) подсчитать количество точек, у которых x и y совпадают

в) найти наиболее удалённую от начала координат точку

г) расположить в порядке возрастания координаты x

Листинг `algoritm1.cpp`

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <fstream>
#include <climits>

using namespace std;

ifstream in("input.txt");

struct Point {
public:
        int x;
        int y;

        Point(int x, int y) {

                this->x = x;
                this->y = y;

        }

};

double max_dist = INT_MIN;
```

```
double distance1(int x, int y) {

        return (sqrt(x*x + y*y));

}

void print(vector<pair<int, int>> point) {

        vector<pair<int, int>>::iterator i = point.begin();

        for (i; i != point.end(); i++) {

                cout << "(" << (*i).first << " , " << (*i).second << ")"

        }

        cout << endl;

}
bool f(pair <int, int> A) {

        if (A.first > 0 && A.second > 0) return true;
        return false;

}

bool f1(pair <int, int> A) {

        if (A.first == A.second) return true;
        return false;

}
```

```
bool compare(pair<int,int> A, pair<int,int> B)
{
        if (distance1(A.first, A.second) < distance1(B.first, B.second))

                return true;

        }
        return false;
}

int main()
{
        setlocale(LC_ALL, "rus");

        vector<pair<int, int>> point;

        int x, y, n = 0;

        while (in >> x) {

                in >> y;
                point.push_back(make_pair(x, y));
                ++n;

        }

        vector<pair<int, int>>::iterator i = point.begin();

        print(point);


        cout << "a) Удалить все точки из I четверти: " << endl;

        point.erase(remove_if(point.begin(), point.end(), f), point.end(
```

```
        print(point);

        cout << "б) Подсчитать количество точек, у которых x и y совпада
        i = point.begin();
        int s = 0;

        s = count_if(point.begin(), point.end(), f1);

        cout << s << endl << endl;

        cout << "в) Найти наиболее удалённую от начала координат точку"

        i = max_element(point.begin(), point.end(), compare);

        cout << "(" << (*i).first << " , " << (*i).second << ")" << endl

        cout << "г) Расположить в порядке возрастания координаты x" << e

        sort(point.begin(), point.end());

        print(point);

        return 0;
}
```

Пример:

Вход	Выход
1 1 -1 -1 -10 10 47 -89 -47 89	а) удалить все точки из I четверти (-1,-1), (-10,10), (47,-89), (-47,89) б) подсчитать количество точек, у которых x и y совпа 1 в) найти наиболее удалённую от начала координат точку (47,-89) г) расположить в порядке возрастания координаты x (-47,89), (-10,10), (-1,-1), (47,89)

ПРИЛОЖЕНИЕ Д
Стек и очередь

ЗАДАЧА 1 (А). Удалить первый кратный трём элемент.

Листинг stack1A.cpp

```
#include <iostream>
#include <stack>

using namespace std;

int main()
{
        stack <int> stack1, stack2;
        int a, n, i = 1, num = 0;
        cin >> n;

        bool f = false;

        while (i <= n) {

                cin >> a;
                stack1.push(a);
                if (a % 3 == 0 && f == false) {

                        num = i;
                        f = true;

                }

                ++i;

        }

        num = n - num + 1;
```

```
        i = 1;

        while (!stack1.empty()) {

                a = stack1.top();
                stack1.pop();

                if (num != i)
                        stack2.push(a);

                ++i;

        }

        while (!stack2.empty()) {

                cout << stack2.top() << " ";
                stack2.pop();

        }

        return 0;
}
```

Пример:

Вход	Выход
6 -1 15 8 4 9 1	 -1 8 4 9 1

Задача 17 (Б). Заменить первый, средний и последний элемент на единицы, если они простые.

Листинг stack17B.cpp

```
#include <iostream>
#include <stack>
```

```
using namespace std;

bool ifNotPrime(int a) {

        for (int i = 2; i <= sqrt(a); ++i) {

                if (a % i == 0) return true;

        }

        return false;

}

int main()
{
        stack <int> stack1, stack2;
        int a, n, i = 1, num = 0, mid_num=0;
        cin >> n;

        while (i <= n) {

                cin >> a;
                stack1.push(a);
                ++i;

        }

        i = 1;
        mid_num = n / 2;

        while (!stack1.empty()) {
```

```
            a = stack1.top();
            stack1.pop();

            if ((i == 1 || i == n || i == mid_num) && ifNotPrime(a)
                    a = 1;

                    stack2.push(a);

            ++i;

        }

        cout << endl;

        while (!stack2.empty()) {

                cout << stack2.top() << " ";
                stack2.pop();

        }


        return 0;
}
```

Пример:

Вход	Выход
6 11 2 3 4 5 6	1 2 3 4 5 6
5 5 4 3 2 7	1 4 1 2 1

Задача 10. Дано N очередей. С ними выполняют K операций PUSH(I, V), TOP(I), POP(I) – добавить в хвост очереди I число V, показать число в

голове очереди I и удалить число из головы очереди I (при этом результат показывается). Результат должен содержать столько чисел, сколько операций TOP и POP было сделано.

Листинг queue10.cpp

```
#include <iostream>
#include <queue>
#include <string>
#include <fstream>

using namespace std;

ifstream in("input.txt");
ofstream out("output.txt");

int main()
{
        int a, b;
        in >> a >> b;

        queue<int> **QArray = new queue<int> *[a]; // массив указателей
        for (int i = 0; i < a; i++)
        {
                QArray[i] = new queue<int>; // создание объекта очередь
        }

        for (int i = 0; i < b; ++i) {

                string str;
                in >> str;

                string str1;
                str1 += str[0];
                str1 += str[1];
```

```
str1 += str[2];

if (str1 == "PUS") {

        string str2, str3;
        int i = 5;

        while (i < str.length() && str[i] != ',') {

                str2 += str[i];
                ++i;

        }

        ++i;

        while (i < str.length() && str[i] != ')') {

                str3 += str[i];
                ++i;

        }


        int num = 0;
        num = atoi(str2.c_str());
        int n = 0;
        n = atoi(str3.c_str());

        QArray[num-1]->push(n);

}

else if (str1 == "POP") {
```

```
                string str2;
                int i = 4;

                while (i < str.length() && str[i] != ')') {

                        str2 += str[i];
                        ++i;

                }

                int num = 0;
                num = atoi(str2.c_str());

                out << QArray[num-1]->front() << " ";
                QArray[num-1]->pop();

        }

        else if (str1 == "TOP") {

                string str2;
                int i = 4;

                while (i < str.length() && str[i] != ')') {

                        str2 += str[i];
                        ++i;

                }

                int num = 0;
                num = atoi(str2.c_str());
```

```
                        out << QArray[num-1]->front() << " ";

                }
        }
        return 0;
}
```

Пример:

Вход	Выход
100 6 PUSH(100,1) PUSH(100,3) POP(100) PUSH(50,3) TOP(50) TOP(100)	1 3 3

ПРИЛОЖЕНИЕ Е
Множество

ЗАДАЧА 3. Найти все такие цифры, которые встречаются только в кратных 10 числах.

Листинг set3.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include <set>
#include <iterator>

using namespace std;
ifstream in("input.txt");
ofstream out("output.txt");

int main()
{
        int c = 0;
        setlocale(LC_ALL, "rus");
        int i;
        vector <int> v;
        set <int> set1, set2;
        while (in >> i)
        {
                v.push_back(i);
        }

        copy(v.begin(), v.end(), ostream_iterator <int>(cout, " "));

        cout << endl;
        for (int i = 0; i < v.size(); i++)
        {
```

```
        if (v[i] % 10 != 0) {

                int k = abs(v[i]);
                while (k > 0)
                {
                        int a = k % 10;
                        k = k / 10;
                        set1.insert(a);

                }
        }
}

for (int i = 0; i < v.size(); ++i) {

        if (v[i] % 10 == 0) {

                int k = abs(v[i]);

                while (k != 0) {

                        int a = k % 10;
                        bool f = false;

                        for (auto it = set1.begin(); it != set1.

                                if (*it == a) {

                                        f = true;
                                        break;

                                }
```

```
                                }

                                if (f == false) set2.insert(a);

                                k /= 10;

                        }

                }

        }

        if (!set2.empty()) {
                copy(set2.begin(), set2.end(), ostream_iterator <int>(co
                cout << endl;
        }
        else cout << "No elements found" << endl;

        return 0;
}
```

Пример:

Вход	Выход
100 6 130 40 50 60 13	4 5
10 20 30 123 40 50 45 890	8 9

Задача 7. Дан текст, состоящий из предложений, разделённых знаками препинания из набора «.?!». Предложения в свою очередь состоят из слов, отделённых друг от друга пробелами. Найти слова (без учёта регистра) и их количество, которые встречаются только в повествовательных и вопросительных предложениях.

Листинг `set7.cpp`

```
#include <iostream>
#include <fstream>
```

```
#include <vector>
#include <string>
#include <set>
#include <iterator>
#include <algorithm>

using namespace std;
ifstream in("input.txt");
ofstream out("output.txt");

int main()
{
        int c = 0;
        setlocale(LC_ALL, "rus");
        string i;
        vector <string> v;
        set <string> set1, set2, set3;
        while (in.peek()!=EOF)
        {
                getline(in, i);
                v.push_back(i);
        }

        for (int i = 0; i < v.size(); ++i)
                cout << v[i] << endl;
        cout << endl;

        for (int i = 0; i < v.size(); ++i) {

                string str = v[i];

                if (str[str.length() - 1] == '!') {

                        string str_buf;
```

```
                int j = 0;
                while (j < str.length()) {

                        if (isalpha(str[j])) {
                                str[j] = tolower(str[j]);
                                str_buf += str[j];
                        }
                        else {
                                set1.insert(str_buf);
                                str_buf.clear();
                        }
                        ++j;
                }
        }

        else {

                string str_buf;
                int j = 0;
                while (j < str.length())  {

                        if (isalpha(str[j])) {
                                str[j] = tolower(str[j]);
                                str_buf += str[j];
                        }
                        else {
                                set2.insert(str_buf);
                                str_buf.clear();
                        }
                        ++j;
                }
        }
```

```
        }

        copy(set1.begin(), set1.end(), ostream_iterator <string>(cout, "
        cout << endl;
        copy(set2.begin(), set2.end(), ostream_iterator <string>(cout, "
        cout << endl << endl;

        set_difference(set2.begin(), set2.end(), set1.begin(), set1.end(

        if (!set3.empty())
                copy(set3.begin(), set3.end(), ostream_iterator <string>
        else cout << "Not unique words found" << endl;


        return 0;
}
```

Пример:

Вход	Выход
aaa bbb jjj. aaa ccc! bbb ccc hhh?	bbb jjj hhh

ПРИЛОЖЕНИЕ Ж
Отображения

ЗАДАЧА 10. Во входном файле задан набор слов и целых чисел, разделённых пробелами. Найти все слова, встречающиеся столько же раз, сколько последнее число.

Листинг `map10.cpp`

```
#include <map>
#include <iterator>
#include <string>
#include <fstream>
#include <iostream>
#include <vector>

using namespace std;

ifstream in("input.txt");
ofstream out("output.txt");

int main()
{
        setlocale(LC_ALL, "rus");
        map <string, int> words;
        string tmp, str_num;
        vector <string> vec_ans;

        while (in >> tmp)
        {
                words[tmp]++;
                if (isdigit(tmp[0]))
                        str_num = tmp;
        }

        int num = atoi(str_num.c_str());
```

```
        cout << num << endl;
        string ans;

        for (auto i = words.begin(); i != words.end(); ++i) {

                string st = i->first;
                int st2 = i->second;
                cout << st << " " << st2 << endl;

                if (isalpha(st[0]) && st2==num) {

                        vec_ans.push_back(st);

                }

        }
        cout << endl;

        if (!vec_ans.empty()) {
                for (int i = 0; i < vec_ans.size(); ++i) {

                        cout << vec_ans[i] << " ";

                }
                cout << endl;
        }
        else cout << "No words found" << endl;

        return 0;
}
```

Пример:

Вход	Выход
12 a f d f a f a ra a a a 7 a 6 8 a e d s a e a e 4 a e 6 8 a e d s a e a e 15 a e	a a e No words found

ПРИЛОЖЕНИЕ 3

Использование контейнеров и алгоритмов STL

ЗАДАЧА 3. Свой стек, стек на основе массива, сравнить со стеком STL.

Листинг `final.cpp`

```
#include <iostream>
#include <string>
#include <fstream>
#include <stack>

using namespace std;

ifstream in("input.txt");
ofstream out("output.txt");

template <typename T>
class myStack
{
        struct Element
        {
                T inf;
                Element *next;
                Element(T x, Element *p) : inf(x), next(p) {}
        };
        Element *head; //указатель на вершину стека
public:
        myStack() : head(0) {} //конструктор стека
        bool Empty() //проверка стека на пустоту
        {
                return head == 0;
        }

        T Pop() //взять элемент из стека
        {
```

```
                if (Empty()) //если стек пуст, то ничего не делать
                {
                        return 0;
                }
                Element *r = head; //иначе запоминаем указатель на верши
                T i = r->inf; //запоминаем информацию из верхнего элемен
                head = r->next; //передвигаем указатель стека на следующ
                                                //от вершины элемент
                delete r; //освобождаем память, на которую указывает r
                return i; //возвращаем значение
        }
        void Push(T data) //добавить элемент в стек
        {
                head = new Element(data, head);
        }
        T Top() //просмотреть элемент на вершине стека
        {
                if (Empty()) //если стек пуст, то возвращаем 0
                {
                        return 0;
                }
                else //иначе возвращаем информацию из вершины стека
                {
                        return head->inf;
                }
        }
};

template <typename T>
class ArrStack {
private:
        int size;
        T *array;
```

```
public:
        ArrStack(int maxSize) {
                size = 0;
                array = new T[maxSize];
        }

        bool Empty() {

                return size == 0;

        }

        void Push(const T newElement) {
                array[size] = newElement;
                size++;
        }

        void Pop() {
                size--;
        }

        T Top() {
                return array[size - 1];
        }
};

int main()
{

        int n = 0, buf;
        // исследовать с разными типами, разным количеством элементов
        //kazachkova.anna@gmail.com

        //для самописного stack
```

```
        myStack <int> stack1;

        while (in >> buf)
                stack1.Push(buf);

        while (!stack1.Empty())
                cout << stack1.Pop() << " ";
        cout << endl << endl;

        //для stack на основе массива

        ArrStack <int> stack2(100);

        while (in >> buf)
                stack2.Push(buf);

        while (!stack2.Empty()) {
                cout << stack2.Top() << " ";
                stack2.Pop();
        }
        cout << endl;

        return 0;
}
```