Методические указания

к выполнению лабораторных работ по дисциплине

«Объектно-ориентированное программирование»

2*-го* курса бакалавриата ИВТ

1, 2 модуль

**Содержание**

[Лабораторная работа №1 (3 части)](#_i28eihwo7qny)

[1. Классы](#_ro5utn8otcgy)

[Структура класса](#_tyjcwt)

[Тестирование](#_1t3h5sf)

[Оформление кода](#_2s8eyo1)

[Эффективность кода](#_3rdcrjn)

[2. Коллекции](#_dis1hkvp44qr)

[Структура коллекции](#_1ksv4uv)

[Работа с файлами](#_2jxsxqh)

[Тестирование](#_z337ya)

[Оформление кода](#_3j2qqm3)

[Эффективность кода](#_1y810tw)

[3. Наследование](#_2xcytpi)

[Реализация наследования](#_3whwml4)

[Графический интерфейс](#_qsh70q)

[Тестирование](#_3as4poj)

[Оформление кода](#_1pxezwc)

[Оформление интерфейса](#_2p2csry)

[Эффективность кода](#_147n2zr)

[Эффективность интерфейса](#_23ckvvd)

[Задания ЛР №1](#_32hioqz)

[Проектная работа. Часть 1](#_dvvk3kfcwf65)

[Лабораторная работа №2 (2 части)](#_sx1n8t26wgj2)

[1. Шаблонная коллекция](#_xp5upmurnht1)

[2. Заключительный этап](#_91gjyxzdk3pb)

[Задания ЛР №2](#_q0mvrzexjbq2)

[Проектная работа. Часть 2](#_yb2azp4ey06)

## 

# Лабораторная работа №1 (3 части)

### 1. Классы

**Методика оценки**

| **Структура класса**  *Класс содержит перечисленные пункты*   * конструктор по умолчанию * конструктор инициализации * конструктор копирования * деструктор * поля класса, требуемые по условию задачи * методы доступа (селекторы и модификаторы) для скрытых (private) полей класса * Если в классе окажутся открытые (public) поля, это должно быть обосновано   *Класс не допускает хранение ошибочных значений* | **40%** |
| --- | --- |
| **Эффективность кода**  *Методы, не изменяющие поля класса, объявлены как константные*   * **подсказка**: методы, возвращающие значения полей   *Ненужное копирование избегается передачей значений по ссылке*   * **допустимые исключение**: базовые типы данных (int, double, char, bool и т.п.) | **15%** |
| **Оформление кода**  *Код программы выдержан в одном стиле*   * на протяжении всего кода программы, выбранные пользователем имена переменных придерживаются одного и того же стиля написания (например, UpperCamelCase для классов, lowerCamelCase для полей и методов, SCREAMING\_SNAKE\_CASE для констант)   *Объявление классов в .h файле, реализация в .cpp файле разделены*  *Имена переменных осмыслены*   * имена классов, методов и полей отражают их конкретную роль в программе * **исключение**: переменные в циклах, временные переменные   *Длины строк не превышают 80 символов*   * **исключение**: строки с комментариями | **15%** |
| **Тестирование**  *Тесты отвечают следующим требованиям*   * все тесты вызываются из функции main() (не в коде созданных классов) * для тестирования используется функция assert() из библиотеки assert.h * исключить дублирование данных при проверке * при отсутствии ошибок выводится сообщение об успешном прохождении тестов   *Перечисленные функции успешно проходят тестирование*   * конструктор по умолчанию * конструктор инициализации * конструктор копирования * методы доступа полей класса (селекторы и модификаторы) * методы манипуляций с полями класса (при наличии) | **30%** |

**Описание требований**

Сложной программе вовсе необязательно выглядеть сложно. И у объектно-ориентированных языков программирования есть несколько способов упрощения кода. В структурных языках программирования последовательность команд, связанных общей задачей, обычно объединяют в подпрограмму (т.е. функцию или процедуру). Где бы и сколько бы раз она ни понадобилась, теперь достаточно написать одно её имя, а не копировать одни и те же строки кода – чем не шаг к упрощению? Можно пойти ещё дальше, сгруппировав переменные и функции, для чего существует *пространство имён* (namespace). Однако это пространство только одно; как быть, если нужно несколько одинаковых по структуре наборов данных?

На помощь приходят классы. Один раз описав класс с его *полями* (переменными класса) и *методами* (функциями класса), можно создать любое количество объектов этого класса, обладающих такой же структурой, но хранящих свои значения. Однако на этом их удобство не заканчивается.

Класс позволяет задать специальные методы, которые объект этого класса самостоятельно вызовет в самом начале и самом конце своего существования. *Конструктор* чаще всего используется для инициализации полей объекта и, при необходимости, выделения памяти, а *деструктор* – для очистки выделенной памяти. Поскольку объект может быть создан разными способами, существует несколько разновидностей конструктора:

**Конструктор по умолчанию** не принимает аргументов

**Конструктор инициализации** принимает от одного до нескольких аргументов и на их основании устанавливает значения полей **Конструктор копирования** принимает ссылку (обычно константную) на объект того же класса, и копирует значения его полей

Sample()

Sample(int i, char c)

Sample(const Sample &other)

Также классы позволяют установить доступность своих полей и методов:

class Sample { public:

//доступ везде, где определён этот класс

protected:

//доступ только в методах этого класса или его наследниках

private:

//доступ только в методах этого класса

}

Для предоставления регулируемого доступа к недоступному полю, в классе могут быть использован

т.н. *селектор*, возвращающий значение поля и *модификатор*, присваивающий полю новое значение (переданное в аргументе), если оно пройдёт некоторую проверку. В ряде объектно-ориентированных языков, селектор и модификатор являются обычными методами.

#### Структура класса

Класс должен включать в себя:

* + - * конструктор по умолчанию
      * конструктор инициализации
      * конструктор копирования
      * поля класса, требуемые по условию задачи
      * методы доступа (селекторы и модификаторы) для скрытых (private) полей класса

Если в классе окажутся открытые (public) поля, это должно быть обосновано.

В классе должны присутствовать перечисленные поля и методы.

Класс не должен допускать хранение ошибочных значений.

#### Тестирование

Чтобы убедиться в том, что методы класса работают как положено, их необходимо подвергнуть проверке. Для этого мы воспользуемся функцией assert() из библиотеки assert.h. Эта функция остановит выполнение программы и выдаст ошибку, если её аргумент окажется ложным (false) или равным нулю. Если несколько раз вызвать эту функцию с различными предположениями вида ожидаемое значение == рассчитанное значение в качестве аргумента, тогда завершение программы без ошибки будет означать, что во всех вызовах функции assert() аргументы были истинными (true), а все наши предположения оказались верны.

Тестированию должны подвергнуться следующие функции:

* + - * конструктор по умолчанию: (поле == значение по умолчанию)
      * конструктор инициализации: (поле == подходящее значение инициализации) или (поле == значение по умолчанию, если значение инициализации ошибочно)
      * конструктор копирования: (поле копии == поле оригинала)
      * методы доступа полей класса: (поле == подходящее новое значение) или (поле != ошибочное новое значение)

Перечисленные функции должны успешно проходить тестирование.

#### Оформление кода

Хотя у всех могут быть свои предпочтения касательно размера отступов, расположения скобок и других мелочей, придерживаясь одного стиля оформления кода можно существенно упростить чтение, а иногда и понимание кода своей программы. Работая с библиотеками stl и Qt следует принять во внимание, что принятые ими стили немного различаются:

|  | stl | Qt |
| --- | --- | --- |
| Имена переменных и функций | [snake\_case](https://en.wikipedia.org/wiki/Snake_case) | [camelCase](https://en.wikipedia.org/wiki/Camel_case) |
| Имена классов | snake\_case | [PascalCase](https://en.wikipedia.org/wiki/PascalCase) |

Код программы должен быть выдержан в одном стиле.

Должна прослеживаться логика в именах переменных и классов.

#### Эффективность кода

Для проверки работоспособности исходного кода программы требуется преобразовать его в исполняемый машинный код. Этот нетривиальный процесс называется *компиляцией*, а выполняющие его программы, соответственно, называются *компиляторами*. Компиляторы различаются в своей способности упрощать и оптимизировать исходники, генерируя эффективный машинный код, однако они не всесильны и подсказки от программиста могут им здорово помочь.

Одной из таких подсказок может быть использование модификатора const. Поля и объекты, объявленные с таким модификатором не могут быть изменены в течение своего существования, а методы класса при вызове не изменяют значения полей.

const int Class::method(int value) **возвращает неизменяемое значение**

int Class::method(const int value) не изменяет значение **аргумента**

int Class::method(int value) const не изменяет значения **полей класса**

Другой способ подразумевает использование модификатора &. В отличие от оператора &, который используется в *выражениях* и возвращает адрес переменной, этот модификатор используется в *объявлениях* полей, методов и аргументов, чтобы показать, что мы передаём его значение *по ссылке*, обращаясь напрямую к области памяти, в которой лежит этот параметр. Без использования модификатора & параметр передаётся *по значению*, т.е. копируется, а действия производятся над его независимой локальной копией.

void Class::increase(int value){ value++; } Увеличивает **копию** переменной value

void Class::increase(int &value){ value++; } Увеличивает переменную на единицу

int Class::getValue(){ return this->value; } Возвращает **копию поля** value

int& Class::getValue(){ return this->value; } Возвращает поле value (и потенциальную возможность его изменить)

Хотя копирование переменных базовых типов почти не стоит ни памяти, ни процессорного времени, на копировании больших объектов обычно можно здорово сэкономить. Правильное применение модификаторов const и & может повысить эффективность работы программы, а неправильное – привести к лишней трате ресурсов или дырам в безопасности.

bool Data::isEqual(Data other) { ... } Неоправданное копирование!

Решения:

bool Data::isEqual(Data &other) работает с оригиналом объекта

bool Data::isEqual(const Data &other) гарантирует неизменность оригинала

int& Class::value() { return \_value; } Возможность изменить приватное поле!

Решения:

int Class::value() возвращает копию поля

const int& Class::value() const возвращает неизменное значение поля

Методы, не изменяющие объекты, должны быть объявлены как константные (сonst).

Применить передачу по ссылке (&), чтобы избежать ненужного копирования.

### 

### 2. Коллекции

**Методика оценки**

| **Структура класса**  *Коллекция реализована самостоятельно на основе:*  *Для варианта 1-3:*   * [стека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) по аналогии с [std::stack](https://en.cppreference.com/w/cpp/container/stack) или [QStack](https://doc.qt.io/qt-5/qstack.html)   *Для варианта 4-6:*   * [очереди](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) по аналогии с [std::queue](https://en.cppreference.com/w/cpp/container/queue) или [QQueue](https://doc.qt.io/qt-5/qqueue.html)   *Для варианта 7-9:*   * [двусвязного списка](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA#%D0%94%D0%B2%D1%83%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B4%D0%B2%D1%83%D0%BD%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA)) по аналогии с [std::list](https://en.cppreference.com/w/cpp/container/list) или [QLinkedList](https://doc.qt.io/qt-5/qlinkedlist.html)   *Для варианта 10-12:*   * [одномерного массива](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2) по аналогии с [std::vector](https://en.cppreference.com/w/cpp/container/vector) или [QVector](https://doc.qt.io/qt-5/qvector.html)   *Для варианта 13-15:*   * [двухмерного массива](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85))   *Для всех вариантов:*   * в *коллекции не используются шаблонные классы сторонних библиотек (stl, Qt, boost и т.п.)* * *Класс содержит пункты (по вариантам):*   *Для варианта 1-12:*   * *конструктор по умолчанию*   *Для варианта 13-15:*   * *конструктор инициализации (размер коллекции как параметр)* * *Класс содержит перечисленные пункты*   + *конструктор копирования*   + *деструктор*   + *метод, возвращающий количество хранящихся в коллекции объектов*   + *методы добавления и удаления элементов (варианты 1-12)*   + *метод, удаляющий все хранимые объекты (варианты 1-12)*   + *класс элемента коллекции, хранящий объект класса из предыдущего задания (варианты 1-9)*   + *класс итератора для обхода всех элементов коллекции в обоих направлениях (варианты 1-9)*   + *методы, возвращающие итераторы на первый и последний элемент коллекции (варианты 1-9)*   + *метод доступа к элементу по индексу (варианты 10-15)*   + *прочие методы, если таковые требуются по условию задачи* * *Класс устойчив к ошибкам*   + *вся динамически выделенная память успешно освобождается*   + *публичные методы коллекции не принимают и не возвращают указатели на свои поля* | **30%** |
| --- | --- |
| * **Работа с файлами** * *Класс содержит методы работы с файлами*   + метод, принимающий путь к файлу и сохраняющий в него содержимое коллекции   + метод, принимающий путь к файлу и загружающий из него содержимое коллекции * *Используется один из способов представления данных*   + **Читаемый текст:** ([std::fstream](https://en.cppreference.com/w/cpp/io/basic_fstream) или [QTextStream](https://doc.qt.io/qt-5/qtextstream.html))   + **Двоичные данные:** (std::fstream + std::ios::binary или [QDataStream](https://doc.qt.io/qt-5/qdatastream.html)   + **Другие форматы:** XML ([QXmlStreamWriter](https://doc.qt.io/qt-5/qxmlstreamwriter.html) + [QXmlStreamReader](https://doc.qt.io/qt-5/qxmlstreamreader.html)), JSON ([QJsonDocument](https://doc.qt.io/qt-5/qjsondocument.html)) и т.п. * *Нет ошибок при записи/чтении файла*   + Содержимое коллекции записывается в произвольный (указанный) файл и читается из того же файла без изменений | **15%** |
| * **Тестирование** * *Тесты отвечают следующим требованиям*   + все тесты вызываются из функции main()   + для тестирования используется функция assert() из библиотеки assert.h   + при отсутствии ошибок выводится сообщение об успешном прохождении тестов * *Реализованы две вспомогательные функции*   + Функция проверки двух коллекций на равенство, попарно сравнивающая их объекты (может быть методом коллекции)   + Функция, последовательного и пронумерованного вывода элементов коллекции на экран (не может быть методом коллекции)   + обе функции пользуются итератором или индексами для организации цикла по коллекции * *Перечисленные функции успешно проходят тестирование*   + конструктор копирования   + функция проверки двух коллекций на равенство   + методы чтения и записи в файл   + методы добавления и удаления объектов (варианты 1-12)   + метод, удаляющий все хранимые объекты (варианты 1-12)   + конструктор инициализации (варианты 10-15) | **30%** |
| * **Оформление кода** * *Код программы выдержан в одном стиле*   + на протяжении нового кода программы, выбранные пользователем имена переменных всё ещё придерживаются одного и того же стиля написания * *Имена переменных осмыслены*   + имена классов, методов и полей отражают их конкретную роль в программе   + **исключение:** переменные в циклах, временные переменные * *Длины строк не превышают 80 символов*   + **исключение:** строки с комментариями * *Классы, от которых зависит работоспособность коллекции (включая итератор) являются членами класса-коллекции (варианты 1-9, рекомендательный характер)* | **15%** |
| * **Эффективность кода** * *Методы, не изменяющие поля класса, объявлены как константные*   + *метод, возвращающий количество хранящихся значений*   + *метод, сохраняющий содержимое коллекции в произвольный файл и прочие методы, не изменяющие хранящиеся в коллекции данные* * *Ненужное копирование избегается передачей значений по ссылке*   + ***исключение:*** *базовые типы данных (int, double, char, bool и т.п.)* * *Методы класса-коллекции реализованы с минимальным количеством циклов* | **15%** |

**Описание требований**

Даже при написании простой программы можно столкнуться с обработкой больших объёмов информации. Хотя для хранения однородных данных можно использовать статические массивы, при удалении и добавлении элементов хотелось бы избежать лишних операций копирования и выделения памяти. Специально для выполнения этой задачи и хранения объектов других классов, существуют так называемые классы-*контейнеры* или *коллекции*, каждый со своим способами добавления, удаления и расположения объектов в памяти. В рамках этой лабораторной работы потребуется написать собственную реализацию одного из этих классов:

| **Коллекция** | **Добавление** | **Удаление** | **Доступ** (чтение/запись) |
| --- | --- | --- | --- |
| Стек  однонаправленный список | в конец | с конца | с конца или по итератору |
| Очередь однонаправленный список | в конец | с начала | с начала или по итератору |
| Двусвязный список  двунаправленный список | в произвольном месте | | с начала, с конца или по итератору |
| Одномерный массив динамический массив | в произвольном месте | | по индексу |
| Двумерный массив динамический массив | в произвольном месте | | по двум индексам |

#### Структура коллекции

В классе-коллекции должны быть определены

* + - * конструктор по умолчанию (варианты 1–12)
      * конструктор инициализации (варианты 10–15, размер коллекции как параметр)
      * конструктор копирования
      * методы добавления и удаления элементов (варианты 1–12)
      * класс элемента коллекции (итератор) (варианты 1–9)

**–** объект класса из предыдущего задания

* + - * метод доступа к элементу по итератору (варианты 1–9) или индексу (варианты 10–15)
      * метод, удаляющий все хранимые объекты
      * метод, возвращающий число хранящихся объектов
      * прочие методы, если таковые требуются по условию задачи

Класс должен соответствовать перечисленным пунктам.

Не должно быть ошибок при выделении и очистке памяти.

Не должно быть возможности нарушить работоспособность коллекции.

#### Работа с файлами

В классе-коллекции также должны присутствовать два метода – для записи данных в произвольный файл и загрузки данных из произвольного файла. Можно выбрать один из способов представления данных:

**Читаемый текст** [std::fstream](https://en.cppreference.com/w/cpp/io/basic_fstream) / [QTextStream](http://doc.qt.io/archives/qt-4.8/qtextstream.html)

**Двоичные данные** [std::fstream (std::ios::binary)](https://en.cppreference.com/w/cpp/io/basic_fstream) / [QDataStream](http://doc.qt.io/archives/qt-4.8/qdatastream.html)

**Другие форматы**: XML, JSON и т.д. [QXmlStreamWriter](http://doc.qt.io/qt-5/qxmlstreamwriter.html) + [QXmlStreamReader](http://doc.qt.io/qt-5/qxmlstreamreader.html); [QJsonDocument](http://doc.qt.io/qt-5/qjsondocument.html)

Содержимое коллекции должно записываться в произвольный файл и читаться из файла без ошибок.

#### Тестирование

Как и в предыдущей работе, тестирование производится при помощи функции assert(). Для тестирования потребуются две дополнительные функции:

1. Функция проверки двух коллекций на равенство, поэлементно проверяющая их соответствующие объекты

(может быть членом класса-коллекции)

1. Функция последовательного и пронумерованного вывода элементов коллекции на экран (**не** может быть членом класса-коллекции)

Тестированию должны подвергнуться

* конструктор копирования: (копия == оригинал) *1* , затем изменение копии и проверка (копия != оригинал)
* конструктор инициализации (варианты 10–15):

(размер коллекции == введённый параметр) для массива или

(размер коллекции == произведение введённых параметров) для двумерного массива

* методы добавления и удаления объектов (варианты 1–12):

(размер коллекции == размер коллекции перед добавлением одного объекта + 1) и (размер коллекции == размер коллекции перед удалением одного объекта - 1)

* метод, удаляющий все хранимые объекты:

(размер непустой коллекции после удаления всех объектов == 0)

* метод доступа к элементу: вывод на экран всех хранимых объектов *02*
* методы чтения и записи в файл:

(непустая коллекция, сохранённая в новый файл == коллекция, прочитанная из того же файла)*01*

Перечисленные функции должны успешно проходить проверку.

#### Оформление кода

Код программы должен быть выдержан в одном стиле.

Должна прослеживаться логика в именах переменных и классов.

#### Эффективность кода

Методы, не изменяющие объекты, должны быть объявлены как константные (сonst).

Применить передачу по ссылке (&), чтобы избежать ненужного копирования.

### 3. Наследование

**Методика оценки**

| * **Реализация наследования** * *Правильно выполнено наследование*   + все общие для классов методы и поля объявлены в родительском классе, а уникальные – только в классах-наследниках * *В иерархии классов присутствует метод идентификации*   + метод является виртуальным, перегруженным в каждом наследнике и возвращающим различные значения в зависимости от класса * *Коллекция адаптирована к появлению нового класса*   + коллекция хранит объекты обоих классов без потери предыдущего функционала | **25%** |
| --- | --- |
| * **Графический интерфейс** * *В графическом интерфейсе присутствуют все элементы для проверки выполнения поставленной задачи*   + элемент интерфейса, отображающий всё содержимое коллекции   + поля для ввода значений объектов обоих классов (варианты 1-12)   + кнопки или пункты меню, позволяющие добавлять объекты с введёнными значениями в коллекцию и удалять их из коллекции (варианты 1-12)   + кнопки или пункты меню, позволяющие сохранить коллекцию в произвольный файл и загрузить её из произвольного файла и прочие элементы интерфейса для обработки данных, требуемых по условию задачи | **20%** |
| * **Тестирование** * *Программа не допускает ошибок и правильно решает задачу* * *Новые и измененные методы программно протестированы* * *Протестирован программно ключевой процесс работы с графическим интерфейсом* | **25%** |
| * **Оформление кода** * *Код программы выдержан в одном стиле*   + на протяжении нового кода программы, выбранные пользователем имена переменных всё ещё придерживаются одного и того же стиля написания * *Имена переменных осмыслены*   + имена классов, методов и полей отражают их конкретную роль в программе   + **исключение:** переменные в циклах, временные переменные * *Длины строк не превышают 80 символов*   + **исключение:** строки с комментариями | **7%** |
| * **Оформление интерфейса** * *Каждый из элементов интерфейса выполняет свою определённую задачу*   + поля, кнопки и прочие интерактивные элементы подписаны в соответствии со своей ролью в приложении * *Элементы интерфейса подстраиваются к размерам окна*   + для каждого окна приложения задана своя компоновка ([layout](http://doc.qt.io/qt-5/layout.html))   + в компоновке окна находятся все элементы интерфейса (т.е. виджеты) и другие объекты компоновки | **8%** |
| * **Эффективность кода** * *Методы, не изменяющие поля класса, объявлены как константные*   + метод, сохраняющий содержимое коллекции в произвольный файл и прочие методы, не добавляющие и не удаляющие хранящиеся в коллекции данные | **7%** |
| * **Эффективность интерфейса** * *Элементы интерфейса адаптируются к ситуации*   + элемент интерфейса, выводящий содержимое коллекции, автоматически обновляется при её изменении   + пользователь получает обратную связь (напр. всплывающая подсказка, диалоговое окно) при совершении ошибки   + интерактивные элементы интерфейса выключаются или скрываются в моменты, когда нет возможности или смысла с ними взаимодействовать. Примеры: кнопка удаления записей (в случае пустой коллекции), кнопка добавления записей (в случае пустых или неверно заполненных полей ввода) и т.п. | **8%** |

**Описание требований**

#### Реализация наследования

Все общие для классов методы и поля должны быть объявлены в родительском классе, а уникальные – только в классах-наследниках.

Должен быть реализован виртуальный метод, позволяющий отличить классы между собой. Класс-коллекция должна хранить объекты обоих классов без потери функционала.

#### Графический интерфейс

Для вывода сообщений и ошибок обычно используются всплывающие окна. Простейшее решение – диалоговое окно, хотя любой другой виджет тоже может привлечь внимание пользователя

Элементы интерфейса, которые не могут быть использованы в данный момент, выключаются setEnabled(bool), setDisabled(bool) или скрываются setVisible(bool), setHidden(bool) до тех пор, пока не появится возможность их использования. Пример: кнопка «удалить», когда контейнер пуст

Хотя внешний вид любых компонент можно настроить по своему усмотрению, не стоит шокировать пользователя буйством красок или вводить его в заблуждение. Почему бы не вдохновиться видом любимых программ или сайтов?

GUI должен давать возможность работы с объектами обоих классов, предоставлять доступ ко всем функциям контейнера и давать возможность выполнить поставленную задачу.

A user interface is like a joke. If you have to explain it, it’s not that good

#### Тестирование

Программа не должна допускать ошибок и правильно решать поставленную задачу.

#### Оформление кода

Код программы должен быть выдержан в одном стиле.

Должна прослеживаться логика в именах переменных и классов.

#### Оформление интерфейса

Каждый из элементов интерфейса должен выполнять свою определённую задачу.

Интерфейс должен предостерегать пользователя от ошибок и рационально использовать пространство.

#### Эффективность кода

Методы, не изменяющие объекты, должны быть объявлены как константные (сonst).

Применить передачу по ссылке (&), чтобы избежать ненужного копирования.

#### Эффективность интерфейса

Интерфейс должен автоматически адаптироваться к изменениям.

Интерактивные элементы интерфейса должны выключаться или скрываться в моменты, когда нет возможности или смысла с ними взаимодействовать.

## Задания ЛР №1

Ниже приведены тексты заданий с описанием организации хранения данных и методов работы с ними для каждой части лабораторной работы:

* 1. Для первой части лабораторной работы. Класс: его поля и специфичные методы
  2. Для второй части лабораторной работы. Контейнер (его [структура данных](#_1ksv4uv)): хранение объектов классов из п. 1, п. 3 и реализация своих специфичных методов
  3. Для третьей части лабораторной работы. Класс: его поля и методы, частично совпадающие с таковыми у класса из п. 1.

1. На одном из столов офиса растёт стопка макулатуры и одноразовой посуды. Сотрудники опасаются, что в определённый момент равновесие нарушится и стопка упадёт. Разработайте для них программу, позволяющую воссоздать стопку и определить, находится ли она в равновесии. Стопка падает если центр масс стопки находится вне границы первого листа (или тарелки).
   1. Лист бумаги: ширина, высота, толщина, координаты по двум осям
   2. Стопка макулатуры ([стек](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) кусков бумаги): метод проверки на наличие равновесия
   3. Бумажная тарелка: радиус, толщина, координаты по двум осям. Наследовать класс Тарелки и Листа от класса Позиция, содержащего поля: координаты, толщина.
2. Некоторых писателей отталкивает сложность современных текстовых редакторов. Разработайте для них программу с минимальным функционалом: хранение и редактирование текста с возможностью отмены и возвращения изменений.
   1. Операция редактирования текста со свойствами: тип операции (вставка или удаление типа enum), текст и позиция изменённой подстроки
   2. История правок ([стек](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) операций над текстом): методы отмены и восстановления правок в заданном тексте
   3. Замена текста: заменяемая подстрока, замененная подстрока и позиция изменяемой подстроки. Наследовать класс Замены от класса Операций 1го типа; если строка замены пустая - значит операция не замены, а вставки или удаления.
3. Бакалавры с факультета математики открыли для себя множество комплексных чисел и теперь нуждаются в новом калькуляторе. Калькулятор должен поддерживать простейшие арифметические операции и производить вычисления, записанные в стеке.
   1. Комплексное число с 3мя свойствами: действительная, мнимая часть, арифметическая операция (сложение, вычитание, умножение или деление)
   2. Калькулятор ([стек](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) объектов класса Комплексное число): вычисление значения хранящегося выражения
   3. Операция: знак операции (+, -, \*, /, R, I, =), количество операндов. Где R - операция взятия действительной части числа, а I - мнимой.
4. Престижная сеть ресторанов быстрого питания решила существенно сократить расходы, заменив часть персонала роботами. Для автоматизации процессов на кухне были закуплены программируемые роботы-повара. Для работы повара нужно представить процесс приготовления каждого блюда как очередь из ингредиентов и действий над ними. Разработайте программу для хранения и редактирования таких процессов.
   1. Ингредиент: название, единица измерения и количество (в этих единицах)
   2. Рецепт ([очередь](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ингредиентов и действий над ними)
   3. Операция над ингредиентом: действие (нарезать, упаковать, пожарить, положить и т.п.), длительность действия
5. Студенты с кафедры робототехники запускают радиоуправляемый автомобиль. На автомобиль ещё не успели поставить GPS-модуль, поэтому его местоположение приходится вычислять по переданным с пульта командам. Напишите программу, позволяющую определить координаты автомобиля на основании переданных с пульта команд.
   1. Команда движения: Шаг с параметрами текущее время, скорость
   2. Последовательность команд ([очередь](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) команд движения и поворота): расчёт координат автомобиля в заданное время
   3. Команда поворота: текущее время, радиус поворота (положительный - против часовой стрелки, отрицательный - по часовой)
6. Филолог Филипп уже долгое время работает на кассе ресторана быстрого питания, однако с появлением автоматов самообслуживания он начал беспокоиться за своё рабочее место и искать способы получить одобрение начальства. Одним из таких способов является сбор пожертвований в различные фонды, например, фонд поддержки филологов. Чтобы похвастаться своей продуктивностью, он хочет составить подробный отчёт с ежедневной выручкой. Поскольку с математикой Филипп не в ладах, ему нужна программа, рассчитывающая выручку за произвольный период времени.
   1. Покупка: время и общая сумма покупки
   2. Кассовый аппарат ([очередь](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) покупок и пожертвований): расчёт выручки за заданный период времени
   3. Пожертвование в фонд: время и сумма пожертвования, название фонда
7. Профессора кафедры прикладной математики любят заходить в студенческую столовую после долгого дня ведения лекций и написания научных статей. При выборе обеда каждый профессор уделяет особое внимание калорийности, поскольку он хочет восполнить дневные затраты на умственную деятельность. Напишите программу, позволяющие хранить блюда и их сочетания в порядке увеличения энергетической ценности и производить поиск ближайшего блюда или обеда по заданной калорийности.
   1. Блюдо: название, вес, содержание на 100 грамм продукта жиров, белков, углеводов, органических кислот и пищевых волокон, метод расчёта [энергетической ценности](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D0%B5%D1%80%D0%B3%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%86%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) на основе этих данных
   2. Меню ([двусвязный список](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA#%D0%94%D0%B2%D1%83%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B4%D0%B2%D1%83%D0%BD%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA)) блюд): автоматическая сортировка блюд по калорийности (при добавлении), поиск ближайшего блюда по заданной калорийности
   3. Блюдо для сотрудников (наследуется от обычного Блюда, добавляя): базовая цена, метод расчёта скидки (студентам - 20%, преподавателям -10%, остальным 0%).
8. Коммивояжёр в перерывах между разъездами тратит массу времени на сбор чемодана. Ему бы очень пригодился список вещей, отсортированный по их удельной ценности (цена/объём). Напишите программу, позволяющую автоматически составить такой список.
   1. Предмет: название, объём, цена
   2. Список вещей ([двусвязный список](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA#%D0%94%D0%B2%D1%83%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B4%D0%B2%D1%83%D0%BD%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA)) предметов): автоматическая сортировка предметов по удельной ценности (при добавлении)
   3. Стопка монет (от прародительского класса Предмет): цена, функция расчёта объёма (Геттер объема, возвращает минимальное число монет достоинством в 1, 5, 10, 50 и 100 у.е., каждая из которых занимает единичный объём).
9. Сотрудники института биоинформатики хотят восстановить первичную структуру белка, для чего требуется объединить его (частично пересекающиеся) фрагменты. Требуется написать программу, которая хранит белковые последовательности (в виде списков аминокислот) и умеет их объединять, если аминокислоты в конце одной последовательности совпадут с аминокислотами в начале другой.
   1. [Протеиногенная аминокислота](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B8%D0%BD%D0%BE%D0%BA%D0%B8%D1%81%D0%BB%D0%BE%D1%82%D1%8B#%D0%9F%D1%80%D0%BE%D1%82%D0%B5%D0%B8%D0%BD%D0%BE%D0%B3%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B0%D0%BC%D0%B8%D0%BD%D0%BE%D0%BA%D0%B8%D1%81%D0%BB%D0%BE%D1%82%D1%8B): способ показывать (и принимать) своё полное название и однобуквенное обозначение
   2. Белок ([двусвязный список](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA#%D0%94%D0%B2%D1%83%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_(%D0%B4%D0%B2%D1%83%D0%BD%D0%B0%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA)) аминокислот): операция сложения, возвращающая результат сложения (с пересечением) двух белков
   3. [Нестандартная аминокислота](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D1%8B%D0%B5_%D0%B0%D0%BC%D0%B8%D0%BD%D0%BE%D0%BA%D0%B8%D1%81%D0%BB%D0%BE%D1%82%D1%8B): способ показывать и принимать своё полное название
10. Блогер Боря собирается публиковать плейлисты для изменения настроения. Для каждого музыкального произведения в своей коллекции он рассчитал настроение по шкалам грустное-весёлое и спокойное-интенсивное. Ему нужна программа, которая принимает начальное и конечное настроение и выдаёт список музыкальных композиций с минимальными скачками настроения (между композициями).
    1. Музыкальная композиция: название, исполнитель, настроение
    2. Сборник ([массив](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2)) музыкальных композиций): метод, принимающий начальное и конечное настроение и возвращающий новый список композиций, меняющий своё настроение от начального до конечного с минимальным средним размером скачка
    3. Попурри: название, начальное и конечное настроение композиции. Добавить Попурри в коллекцию к композициям.
11. В новом автосалоне требуется расположить несколько автомобилей на автоподиумах, причём так, чтобы автоподиумы не пересекались между собой, а на подиумах стояли только автомобили соответствующих марок. Напишите приложение, проверяющее выполнение этих условий.
    1. Автомобиль: название марки, габариты, координаты, угол поворота
    2. План автосалона ([массив](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2)) автомобилей и подиумов): габариты, координаты, метод проверки на выезда автомобилей за границы автоподиумов
    3. Автоподиум: название марки, радиус, координаты
12. Магистров с факультета математики застукали за распитием кофе в лаборатории. В связи с этим дежурного сотрудника обязали вести электронный журнал учёта кофе. В помощь лаборатории, напишите программу для работы с таким журналом, проверяющую правильность заполнения журнала.
    1. Запись в журнале: время доступа к кофеварке, информация о сотруднике (Ф.И.О., должность и учёная степень) и оказанное влияние на объём кофе (больше нуля, если прибавилось и меньше, если убавилось)
    2. Журнал ([массив](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2)) записей): ёмкость колбы, метод проверки журнала на наличие ошибок (кофе было меньше нуля или больше, чем вмещает колба)
    3. Запись дежурного: время наблюдения, текущий объём кофе
13. До квадратного королевства, в котором каждый день была одна и та же погода, добрался первый циклон. Опасаясь возможной паники среди населения, король издал указ построить метеостанции для измерения погодных условий. В казне хватило денег, чтобы застроить всё королевство метеостанциями с шагом в одну королевскую милю, но часть средств загадочным образом пропала, потому вместо нескольких метеостанций пришлось поставить дешёвые флюгеры. Напишите программу, рассчитывающую погоду в произвольной точке, интерполируя данные с ближайших метеостанций и флюгеров.
    1. Показания метеостанции: температура, давление, а также направление и скорость ветра
    2. Метеоцентр ([двумерный массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) метеостанций): расчёт погодных показателей в произвольной точке королевства.
    3. Флюгер: направление и скорость ветра. Класс метеостанций наследует класс флюгеров.
14. У кафедры электротехники внезапно закончилась лицензия на всё программное обеспечение. Разработайте программу, позволяющую рассчитать напряжения в узлах заданной цепи.
    1. Резистор: номер, сопротивление
    2. Матрица сопротивлений ([двумерный массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) элементов электрической цепи): частота тока (*f* ), список источников тока, расчёт потенциалов в узлах цепи [методом узловых потенциалов](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%83%D0%B7%D0%BB%D0%BE%D0%B2%D1%8B%D1%85_%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D0%BE%D0%B2#%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5)
    3. Индуктивность/ёмкость: номер, реактивное сопротивление, метод расчёта полного сопротивления (по частоте тока *f* )
15. Будущих бухгалтеров отталкивает сложность современных электронных таблиц. Разработайте для них программу с минимальным функционалом: хранение числовых и текстовых данных в таблице с изменяемым размером.
    1. Ячейка электронной таблицы: числовое или текстовое значение, может быть пустой
    2. Таблица ([двумерный массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) ячеек): методы вычисления суммы и среднего арифметического всех числовых данных в произвольном столбце
    3. Ячейка с формулой: диапазон ячеек (адреса первой и последней ячейки) и операция над диапазоном (сумма, произведение, среднее значение), метод вывода результата операции (или ошибки, если расчёт невозможен)



## 

# Проектная работа. Часть 1

**Методика оценки**

| **Название темы**  *Название темы соответствует решаемой задаче* | **25%** |
| --- | --- |
| **Постановка задачи**  *Определена решаемая программой задача и критерии, по которым задача будет считаться решённой* | **25%** |
| **Метод решения**  *Приведён примерный алгоритм решения задачи*  *Указан выбранный язык программирования и целевая платформа, для которой разрабатывается приложение*  *Для каждой коллекции данных приведено обоснование их выбора с точки зрения эффективного выделения памяти и скорости работы*  *Для командных работ: определены роли (области ответственности) участников* | **25%** |
| **Иерархия классов**  *Приведена* [*диаграмма классов*](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2)  *Наследование классов обосновано методом решения задачи* | **25%** |

# Лабораторная работа №2 (2 части)

### 1. Шаблонная коллекция

**Методика оценки**

| **Структура класса**  *Класс является шаблонным и соответствует своему варианту*  *Коллекция реализована на основе:*  *Для варианта 1-3:*   * *Множества,* [хэш-таблицы](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0) по аналогии с [std::unordered\_set](http://www.cplusplus.com/reference/unordered_set/unordered_set/) или [QSet](https://doc.qt.io/qt-5/qset.html)   *Для варианта 4-9:*   * *Ассоциативного массива,* [хэш-таблицы](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0) по аналогии с [std::unordered\_map](http://www.cplusplus.com/reference/unordered_map/unordered_map/) или [QHash](https://doc.qt.io/qt-5/qhash.html)   *Для варианта 10-12:*   * [бинарного дерева](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0) по аналогии с [std::map](http://www.cplusplus.com/reference/map/map/) или [QMap](https://doc.qt.io/qt-5/qmap.html)   *Для варианта 13-15:*   * [префиксного дерева](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE) с использованием любого ассоциативного массива   *Требования к реализации:*  *Для варианта 1-9:*   * для хэш-функции можно использовать как свой алгоритм, так и уже [готовый](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F#%D0%92%D0%B8%D0%B4%D1%8B_%C2%AB%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B9%C2%BB); в коллекции реализовано [разрешение коллизий](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0#%D0%A0%D0%B0%D0%B7%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D0%B9)   *Для варианта 10-12:*   * считать, что для типа K реализован оператор >   *Для варианта 13-15:*   * считать, что для типа K реализован оператор [];   *Дополнительные требования:*  *Только для варианта 4-6:*   * тип значения V является не шаблонным, а целым неотрицательным   *Класс содержит перечисленные пункты*   * конструктор по умолчанию и конструктор копирования * деструктор * метод, удаляющий все хранящиеся значения из коллекции * метод, возвращающий количество хранящихся в коллекции значений * метод, принимающий путь к файлу и сохраняющий в него содержимое коллекции * метод, принимающий путь к файлу и загружающий из него содержимое коллекции * оператор ==, принимающий другую коллекцию и возвращающий истину, если обе коллекции хранят одинаковые значения T (или пары ключ-значение K, V) и ложь в противном случае   *Для варианта 1-3:*   * оператор <<, добавляющий в коллекцию заданное значение T   *Для варианта 4-6:*   * оператор <<, добавляющий в коллекцию заданный ключ K. Если ключа не было в коллекции, его значение V будет равно 1. Если ключ уже был в коллекции, значение увеличится на единицу.   *Для варианта 7-15:*   * метод, добавляющий в коллекцию заданную пару ключ-значение K, V   *Для варианта 1-3:*   * метод, удаляющий из коллекции заданное значение T при его наличии   *Для варианта 4-15:*   * метод, удаляющий из коллекции пару ключ-значение по произвольному ключу K   *Для варианта 1-3:*   * оператор [], принимающий значение T и возвращающий при наличии значения в коллекции истину, а при отсутствии – ложь   *Для варианта 4-6:*   * оператор [], принимающий ключ K и возвращающий его значение V   *Для варианта 7-15:*   * оператор [], принимающий ключ K и возвращающий его значение V по ссылке   *Для варианта 1-3:*   * оператор &&, принимающий другую коллекцию и возвращающий все общие значения T в виде новой коллекции   *Для варианта 4-15:*   * метод, принимающий ключ K и возвращающий при его наличии в коллекции истину, а при отсутствии – ложь   *Для варианта 4:*   * метод, возвращающий ключ K с минимумом ошибок * метод, возвращающий ключ K с максимумом ошибок   *Для варианта 5:*   * метод, возвращающий список из трёх ключей K с максимальными значениями   *Для варианта 6:*   * метод, принимающего число и возвращающим количество ключей K, значения которых больше или равны этому числу   *Для варианта 7:*   * метод, принимающий значение V, и возвращающий список ключей K с этим значением   *Для варианта 8:*   * метод, возвращающий сумму всех значений V   *Для варианта 9:*   * метод, принимающий список ключей K и возвращающий список соответствующих значений V   *Для варианта 10-12:*   * метод, возвращающий список всех значений V в порядке возрастания ключа K   *Для варианта 13-15:*   * метод, возвращающий список хранящихся ключей K, начинающихся с заданной последовательности   *Класс устойчив к ошибкам*   * в деструкторе коллекции освобождается вся выделенная ей память * публичные методы коллекции не принимают и не возвращают указатели на свои поля   *Для варианта 4-12:*   * При обращении к несуществующим ключам предусмотрено исключение | **20%** |
| --- | --- |
| **Тестирование**  *Тесты отвечают следующим требованиям*   * все тесты вызываются из функции main() (не в коде созданных классов) * для тестирования используется функция assert() из библиотеки assert.h * исключить дублирование данных при проверке * при отсутствии ошибок выводится сообщение об успешном прохождении тестов * больше на экран ничего не выводится   *Перечисленные функции успешно проходят тестирование*   * Размер коллекции, созданной конструктором по умолчанию, равен нулю * Непустая коллекция после удаления всех объектов равна коллекции, созданной при помощи конструктора по умолчанию) * Непустая коллекция, сохранённая в файл, равна новой коллекции, загруженной из того же файла   *Для варианта 1-3:*   * При добавлении нового значения T размер коллекции увеличивается на единицу, а при добавлении имеющегося значения T размер остаётся прежним   *Для варианта 4-6:*   * При добавлении нового ключа K размер коллекции увеличивается на единицу * При добавлении имеющегося ключа K размер остаётся прежним, а значение V увеличивается на единицу   *Для варианта 7-15:*   * При добавлении новой пары ключ-значение K, V размер коллекции увеличивается на единицу, а при добавлении имеющейся пары ключ-значение K, V размер не изменяется   *Для варианта 1-3:*   * Оператор [] принимает добавленное значение T и возвращает истину   *Для варианта 4-15:*   * Метод, проверяющий наличие ключа принимает добавленный ключ K и возвращает истину * Оператор [] принимает добавленный ключ K и возвращает его значение V   *Для варианта 1-3:*   * При удалении существующего значения T размер коллекции уменьшается на единицу, а при удалении отсутствующего значения T размер не изменяется   *Для варианта 4-15:*   * При удалении существующего ключа K размер коллекции уменьшается на единицу, а при удалении отсутствующего ключа K размер не изменяется     *Для варианта 1-3:*   * Оператор [] принимает удалённое значение T и возвращает ложь   *Для варианта 4-15:*   * Метод, принимающий ключ K и возвращающий при его наличии в коллекции истину, а при отсутствии - ложь   *Для варианта 1-3:*   * При пересечении любой коллекции с пустой, оператор && возвращает пустую коллекцию (any && empty == empty) * При пересечении одинаковых коллекций, оператор && возвращает новую коллекцию, равную любой из них (any && any == any) * Копия непустой коллекции равна оригиналу, после вставки или удаления T равенство нарушается   *Для варианта 4-15:*   * Копия непустой коллекции равна оригиналу, после изменения одного из значений V равенство нарушается | **30%** |
| **Собственная реализация**  *Класс не использует готовые решения*  *Для варианта 1-3:*   * в коллекции не используется шаблонный класс множества или ассоциативного массива из сторонних библиотек (stl, Qt, boost и т.п.)   *Для варианта 4-12:*   * в коллекции не используется шаблонный класс ассоциативного массива из сторонних библиотек (stl, Qt, boost и т.п.)   *Для варианта 13-15:*   * в коллекции не используется шаблонный класс префиксного дерева из сторонних библиотек (stl, Qt, boost и т.п.) | **20%** |
| **Оформление кода**  *Код программы выдержан в одном стиле*   * на протяжении всего кода программы, выбранные пользователем имена переменных придерживаются одного и того же стиля написания (например, [UpperCamelCase](https://ru.wikipedia.org/wiki/CamelCase) для классов, lowerCamelCase для полей и методов, [SCREAMING\_SNAKE\_CASE](https://ru.wikipedia.org/wiki/Snake_case) для констант)   *Имена переменных осмыслены*   * имена классов, методов и полей отражают их конкретную роль в программе * **исключение**: переменные в циклах, временные переменные   *Длины строк не превышают 80 символов*   * **исключение**: строки с комментариями | **10%** |
| **Эффективность кода**  *Временная сложность не превышает ожидаемую*   * Подробнее о [временнОй сложности](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0)   *Для варианта 1-9:*   * время вставки, удаления и поиска значения не зависит от количества значений в коллекции.   *Для варианта 10-12:*   * время вставки, удаления и поиска значения линейно зависит от глубины дерева   *Для варианта 13-15:*   * время вставки, удаления и поиска значения линейно зависит от длины ключа K   *Методы, не изменяющие поля класса, объявлены как константные*   * метод, возвращающий количество хранящихся значений * метод, сохраняющий содержимое коллекции в произвольный файл * и прочие методы, не добавляющие и не удаляющие хранящиеся в коллекции данные   *Ненужное копирование избегается передачей значений по ссылке*   * допустимо исключение: базовые типы данных (int, double, char, bool и т.п.) | **20%** |

### 2. Заключительный этап

**Методика оценки**

| **Графический интерфейс**  *Присутствуют все элементы интерфейса для работы с коллекцией*   * есть возможность добавлять и удалять элементы коллекции, а также сохранять коллекцию в указанный пользователем файл и загружать коллекцию из указанного пользователем файла * присутствуют элементы интерфейса, необходимые для проверки выполнения задания (в зависимости от варианта)   *Есть элемент интерфейса, отображающий всё содержимое коллекции*   * для вывода коллекции используется [QListWidget](https://doc.qt.io/qt-5/qlistwidget.html), [QTableWidget](https://doc.qt.io/qt-5/qtablewidget.html) или [QTreeWidget](https://doc.qt.io/qt-5/qtreewidget.html) | **30%** |
| --- | --- |
| **Тестирование**  *Тесты из предыдущей части продолжают выполняться*  *Программа работает с конкретными типами данных, указанными в варианте*  *Программа решает поставленную задачу, не допуская при этом ошибок* | **40%** |
| **Оформление кода**  *Код программы выдержан в одном стиле*   * на протяжении всего кода программы, выбранные пользователем имена переменных придерживаются одного и того же стиля написания (например, [UpperCamelCase](https://ru.wikipedia.org/wiki/CamelCase) для классов, lowerCamelCase для полей и методов, [SCREAMING\_SNAKE\_CASE](https://ru.wikipedia.org/wiki/Snake_case) для констант)   *Имена переменных осмыслены*   * имена классов, методов и полей отражают их конкретную роль в программе * **исключение**: переменные в циклах, временные переменные   *Длины строк не превышают 80 символов*   * **исключение**: строки с комментариями | **5%** |
| **Оформление интерфейса**  *Каждый из элементов интерфейса выполняет свою определённую задачу*   * поля, кнопки и прочие интерактивные элементы подписаны в соответствии со своей ролью в приложении   *Элементы интерфейса подстраиваются к размерам окна*   * для каждого окна приложения задана своя компоновка ([layout](http://doc.qt.io/qt-5/layout.html)) * в компоновке окна находятся все элементы интерфейса (т.е. виджеты) и другие объекты компоновки | **5%** |
| **Эффективность кода**  *Уместно используются auto и typedef*   * **auto** применяется для вывода типа (например, при совмещении объявления с инициализацией) * **typedef** используется для присвоения нового имени для готовой конструкции (например, для шаблонной коллекции)   *Методы, не изменяющие поля класса, объявлены как константные*   * метод, сохраняющий содержимое коллекции в произвольный файл * и прочие методы, не добавляющие и не удаляющие хранящиеся в коллекции данные | **10%** |
| **Эффективность интерфейса**  *Элементы интерфейса адаптируются к ситуации*   * элемент интерфейса, выводящий содержимое коллекции, автоматически обновляется при её изменении * пользователь получает обратную связь (напр. всплывающая подсказка, диалоговое окно) при совершении ошибки * интерактивные элементы интерфейса выключаются или скрываются в моменты, когда нет возможности или смысла с ними взаимодействовать   **примеры**: кнопка удаления записей (в случае пустой коллекции), кнопка добавления записей (в случае пустых или неверно заполненных полей ввода) и т.п. | **10%** |

## 

## Задания ЛР №2

1. Сотрудники офиса устали от растущих гор макулатуры и решили собраться на выходных в T. Поскольку вкусы у всех оказались разные, было решено поступить следующим образом: каждый из сотрудников подготовит список предпочтительных T, после чего из всех списков будут сформирован один общий список, содержащий только T, которые устраивают всех без исключения. Напишите программу, составляющий такой список из нескольких предложенных.
   1. Коллекция: множество <T>
   2. T = ресторан (строка)
2. Некоторые писатели, не добившиеся самостоятельной публикации, решили объединить усилия и выпустить сборник со своими работами. Оказалось, что все пишут в разных T, что может плохо сказаться на качестве сборника. Для повышения шансов на успешную публикацию, писатели решили найти все T, в которых у каждого из них есть опыт. Напишите для них программу, находящую в заданных списках все общие T и формирующую из них новый список.
   1. Коллекция: множество <T>
   2. T = жанр (строка)
3. Бакалавры математического факультета заметили, что вместо подготовки к приближающейся сессии они тратят слишком много времени в столовой. Чтобы сэкономить время, не оставшись при этом голодными, студенты решили заранее составить список всех T, которые достаточно питательны, всегда есть в достатке и которые можно съесть в пределах одной минуты. Требуется написать программу, позволяющую составить списки (3 коллекции) T по каждой из заданных категорий и определить T, принадлежащие одновременно всем трём категориям.
   1. Коллекция: множество <T>
   2. T = блюдо (строка)
4. Ресторан быстрого питания производит сокращение штата, заменяя обслуживающий персонал роботами. Чтобы определить кандидатов на повышение и увольнение, отдел кадров начал считать ошибки, совершённые каждым K. По окончании некоторого времени самых нерасторопных K уволят, а самых ответственных – повысят. Напишите программу, ведущую учёт ошибок каждого K и способную в любой момент определить кандидатов на увольнение и повышение.
   1. Коллекция: хэш-таблица <K, число ошибок> с методом, возвращающим ключ K с минимумом ошибок и методом, возвращающим ключ K с максимумом ошибок.
   2. K = сотрудник (строка)
5. K с кафедры робототехники собираются устроить соревнования с участием радиоуправляемых автомобилей. Призы, купленные на средства с последнего гранта, получат три K, победившие в большинстве соревнований. Напишите программу, ведущую учёт побед каждого K и способную составить список из трёх победителей.
   1. Коллекция: хэш-таблица <K, количество побед> c методом, возвращающим три ключа K с максимальными значениями (победами)
   2. K = студент (строка)
6. Филолог Филипп решил выучить иностранный язык и стать переводчиком. Грамматика ему даётся без особых проблем, но для обогащения словарного запаса нужно записывать новые K, и периодически повторять старые K. Напишите программу, позволяющую хранить количество повторений для каждого K и возвращающую количество всех выученных K, повторённых не меньше заданного числа раз.
   1. Коллекция: хэш-таблица <K, количество повторений> с методом, принимающего число и возвращающим количество ключей K, значения (повторения) больше или равны этому числу
   2. K = иностранное слово (строка)
7. Скучающие K с кафедры прикладной математики нашли себе необычное развлечение. Заметив однажды, что V уплетают свои обеды за считанные секунды, K устроили тотализатор и ежедневно пытаются угадать, кто быстрее справится со своей порцией. Напишите программу, позволяющую записывать ставки и выводить список K, сделавших ставку на заданного V.
   1. Коллекция: хэш-таблица <K, V> с методом, принимающим некоторое значение V, и возвращающим список ключей K с тем же значением
   2. K = профессор (строка); V = студент (строка)
8. Коммивояжёр собрал чемодан и приступил к выбору маршрута. Для этого нужно выяснить, кто из его постоянных клиентов заинтересован в приобретении хотя бы одной из взятых им K и готов предложить за неё V. Напишите программу, позволяющую составить список K и соответствующих им V, а также рассчитывающую потенциальную выручку от продажи всех K.
   1. Коллекция: хэш-таблица <K, V> с методом расчёта суммы всех значений V
   2. K = вещь (строка); V = наибольшая цена (число)
9. Огромный скачок напряжения вывел из строя сервера института биоинформатики. После восстановления энергоснабжения оказалось, что была потеряна база данных с K и V. Напишите программу, которая может хранить соответствия K различным V и воссоздавать последовательность V по заданной последовательности K.
   1. Коллекция: хэш-таблица <K, V> с методом, принимающим список ключей K и возвращающую соответствующий список значений V
   2. K = [кодон](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%BE%D0%BD) (строка); V = аминокислота (строка)
10. Блогер Боря хочет поблагодарить подписчиков за спонсирование своего путешествия. Для этого во время своей поездки он собирается сделать альбом из фотографий уникальных V. Чтобы ничего не пропустить, ему нужна программа, позволяющая записывать K и соответствующие им V, а также составляющая список V в порядке возрастания K.
    1. Коллекция: бинарное дерево <K, V> с методом, возвращающим список значений V в порядке возрастания ключа K
    2. K = дата; V = астрономическое явление (строка)
11. До начала распродажи мебели и бытовой техники, магазину кухонных гарнитуров нужно срочно провести инвентаризацию. Для этого требуется программа, позволяющая записывать K и соответствующие V, а также составляющая список V в порядке возрастания K.
    1. Коллекция: бинарное дерево <K, V> с методом, возвращающим список значений V в порядке возрастания ключа K
    2. K = артикул (число); V = название кухонного гарнитура (строка)
12. Бакалавров с факультета математики, ухаживающих за бинарным деревом, не устраивает существующий журнал, поскольку они хотят знать заранее, кому в какой день приходить. Для этого им нужна программа, позволяющая вводить соответствия K и V, а также составляющая список V в порядке возрастания K.
    1. Коллекция: бинарное дерево <K, V> с методом, возвращающим список значений V в порядке возрастания ключа K
    2. K = дата; V = студент (строка)
13. С появлением почтовой службы в квадратном королевстве, наконец, появилась возможность обмена сообщениями. Уйма различных почтовых индексов обескуражила почтальонов, поэтому они решили составить справочник, в котором каждому K, состоящему из нескольких цифр, соответствовал некоторый V. В помощь почтальонам, напишите программу, эффективно реализующую такой справочник, а также предлагающую функцию автодополнения K при неполном вводе.
    1. Коллекция: префиксное дерево <K, V> с методом, возвращающим список хранящихся ключей K, начинающихся с заданной последовательности
    2. K = последовательность цифр (строка); V = адрес почтового отделения (строка)
14. Студенты с кафедры электротехники обнаружили, что свет от новых ламп кафедры виден из их общежития, располагающегося в нескольких километрах. Они решили попробовать передавать закодированные сообщения, последовательно включая и выключая лампы, находящиеся на кафедре. Теперь студентам, следящим из общежития, нужна программа, позволяющая записывать K и соответствующие им V, а также предлагающая функцию автодополнения K при неполном вводе.
    1. Коллекция: префиксное дерево <K, V> с методом, возвращающим список хранящихся ключей K, начинающихся с заданной последовательности
    2. K = последовательность цветов (вектор/строка); V = буква (символ)
15. Юных бухгалтеров отправляют на курсы повышения квалификации для обучения языку программирования 1С. Для облегчения работы им нужен электронный справочник, позволяющий записывать K и соответствующие V, а также предлагающий функцию автодополнения K при неполном вводе.
    1. Коллекция: префиксное дерево <K, V> с методом, возвращающим список хранящихся ключей K, начинающихся с заданной последовательности
    2. K = слово, т.е. последовательность букв (строка); V = тип (перечисление: служебное слово/функция/константа)

# 

# Проектная работа. Часть 2

**Методика оценки**

| **Допуск**  *Программа выполняет поставленную задачу*  *Студент способен самостоятельно ответить на вопросы по своему коду* | **10%** |
| --- | --- |
| **Инкапсуляция**  *Выполнен* ***предыдущий*** *пункт*  *Проявлено знание модификаторов доступа и типов наследования классов*   * при отсутствии подходящей ситуации в работе, можно использовать собственный пример | **30%** |
| **Наследование**  *Выполнен* ***предыдущий*** *пункт*  *Демонстрируется знание принципов наследования, особенностей абстрактных классов и виртуальных функций*   * для демонстрации используется готовый код курсовой работы либо гипотетический пример | **30%** |
| **Полиморфизм**  *Выполнен* ***предыдущий*** *пункт*  *Проявлено знание шаблонных классов и функций, а также умение перегружать методы и операторы*   * если шаблоны и перегрузка не потребовались в ходе курсовой работы, допускается приведение своего примера их использования | **30%** |