**Черкасов Александр 1 вариант Лабораторная работа №2-3 А-08-19**

**Содержание**

[**Задание (2.1)** 1](#_Toc51623285)

[**Задание (2.2)** 1](#_Toc51623286)

[**1. Постановка задачи** 1](#_Toc51623287)

[**2. Разработка программы** 1](#_Toc51623288)

[2.1 Разработка функций и методов классов 1](#_Toc51623289)

[2.2 Разработка интерфейса 3](#_Toc51623290)

[**3. Реализация и тестирование программы** 4](#_Toc51623291)

[3.1 Описание разработанной программы 4](#_Toc51623292)

[3.2 Тестирование программы 5](#_Toc51623293)

[**Вывод** 9](#_Toc51623294)

[**Приложение. Код программы** 9](#_Toc51623295)

# **Задание (2.1)**

Описать класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне.

Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, — и свойство, позволяющее получить его текущее состояние.

При выходе за границы диапазона выбрасываются исключения. Написать программу, демонстрирующую все разработанные элементы класса.

# **Задание (2.2)**

Создать дочерний класс Счетчик1, который может увеличивать или уменьшать свое значение на любое введенное число в заданном диапазоне.

# **1. Постановка задачи**

Разработать объектно-ориентрованную программу в соответствии с заданиями 2.1 и 2.2

**Функции**: Ввод и вывод данных о показаниях счётчиков, реализация методов по увеличению или уменьшению показаний в заданном диапазоне.

**Входные данные**:

Для конструктора: нет/начальное состояние счетчика

Для функции увеличения/уменьшения: значение для увеличения/уменьшения счетчика

**Выходные данные**: Измененные показания счетчиков

**Ограничения**:

Показания счетчиков – целое число в диапазоне [-2048 2048]. При увеличении или уменьшении нельзя выходить за рамки диапазона.

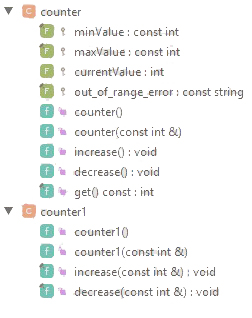
**Вид приложения** - консольное приложение на языке C++.

**Среда разработки** – CLion

# **2. Разработка программы**

2.1 Разработка функций и методов классов

Необходимо создать класс counter, содержащий в себе методы увеличения и уменьшение показания счетчика, получение текущего значения. Также необходимо разработать дочерний класс counter1, в котором требуются переписать методы увеличения и уменьшения, чтобы могли принимать значение, на которое требуется увеличить показания. В таблицах \* \* приведены поля, конструкторы и методы классов, а диаграмма приведена на рис\*



|  |  |
| --- | --- |
| **counter** | |
| **Конструкторы** | **Описание** |
| counter() | Конструктор по умолчанию. Задает значение currentValue = 0 |
| explicit counter(const int&); | Конструктор с параметром. Проверяет входное значение на вхождение в диапазон {minValue, maxValue] Если входит – записывает его в currentValue, иначе – выбрасывает runtime\_error |
| **Поля** |  |
| const int minValue = -2048; | Нижняя граница счетчика |
| const int maxValue = 2048; | Верхняя граница счетчика |
| int currentValue; | Текущее показание счетчика |
| const std::string out\_of\_range\_error = "counter::out\_of\_range"; | Сообщение, содержащееся в выбрасываемой runtime\_error |
| **Методы** |  |
| virtual void increase(); | Увеличивает показание счетчика на 1. Если показания выходят за границы – изменение отменяется, выбрасывается runtime\_error |
| virtual void decrease(); | Уменьшает показание счетчика на 1. Если показания выходят за границы – изменение отменяется, выбрасывается runtime\_error |
| [[nodiscard]] int get() const; | Возвращает показания счетчика |

|  |  |
| --- | --- |
| **counter1** | |
| **Конструкторы** | **Описание** |
| counter1() | Конструктор по умолчанию. Вызывает соответствующий конструктор родительского класса counter |
| explicit counter1(const int&); | Конструктор с параметром. Вызывает соответствующий конструктор родительского класса counter |
| **Методы** |  |
| void increase(const int&); | Увеличивает показание счетчика на заданное значение. Если показания выходят за границы – изменение отменяется, выбрасывается runtime\_error |
| void decrease(const int&); | Уменьшает показание счетчика на заданное значение. Если показания выходят за границы – изменение отменяется, выбрасывается runtime\_error |

2.2 Разработка интерфейса

Разработано консольное приложение, предоставляющее пользователю диалоговое меню, с помощью которого пользователь может выбрать начальное значение счетчика, увеличить или уменьшить его. Имеется 2 «режима работы»: с классом counter и классом counter1. Программа выводит минимальное и максимальное значения счетчика и его текущее значение, чтобы пользователь не вышел за границы диапазона.

Любые исключения, выбрасываемые программой (runtime\_error от счетчика и invalid\_argument от неправильно введенного значения в процессе работы) отлавливаются в блоке *try catch*, чтобы программа не сломалась во время работы.

# **3. Реализация и тестирование программы**

3.1 Описание разработанной программы

Для реализации программы в качестве подхода выбрано объектно-ориентированное программирование.

Разработано консольное приложение на языке С++

Среда программирования: CLion

В заголовочных файлах .h содержится интерфейс классов, в .cpp файлах – реализация классов

*Класс counter*. cодержит два конструктора по умолчанию и параметризированный. Первый устанавливает значение счетчика в 0. Второй проверяет полученное значение на вхождение в диапазон и при верном значении вписывает его как значение счетчика, иначе выбрасывается runtime\_error. Реализовано 3 метода: get(), increase(), decrease(). Первый возвращает текущее показание счетчика, следующие на единицу соответственно увеличивают или уменьшают текущее значение счетчика с проверкой на выход за пределы диапазона. При выходе за диапазон – изменение отменяется, выбрасывается runtime\_error

*Класс counter1*. Класс-наследник от класса counter. Своими двумя конструкторами (по умолчанию и параметризированным) вызывают конструкторы родительского класса. Переопределены методы increase и decrease для изменения показания счетчика не на 1, а на введенное значение.

3.2 Тестирование программы

Табл. 3.1. Тесты

|  |  |  |
| --- | --- | --- |
| **№ теста** | **Смысл теста** | **Результат** |
| **1** | Проверка **стандартных конструкторов** | counter::get = 0  counter1::get = 0 |
| **2** | Проверка **конструкторов с параметром** | counter::get = 10  counter1::get = 10 |
| **3** | Проверка функции **increase** | counter::get = 1  counter::get = 5 |
| **4** | Проверка функции **decrease** | counter::get = -1  counter1::get = -4 |
| **5** | Проверка выхода за **верхний предел** | В всех случаях  std::runtime\_error  (“counter::out\_of\_range”) |
| **6** | Проверка выхода за **нижний предел** |  |
| **7** | Проверка выхода за **верхний предел** в **конструкторе** |  |
| **8** | Проверка выхода за **нижний предел** в **конструкторе** |  |
| **9** | Проверка **отрицательного параметра** для **increase** | counter1::get = -1 |
| **10** | Проверка **отрицательного параметра** для **decrease** | counte1::get = 1 |
| **11** | Попытка ввести **строку** в параметры **конструктора** | std:: runtime\_error |
| **12** | Попытка ввести **строку** в параметры **increase** | Во всех случаях  std::invalid\_argument |
| **13** | Попытка ввести **строку** в параметры **decrease** |  |

# **Вывод**

Проделанная работа познакомила с принципами ООП:

1)Инкапсуляция – ограничили доступ к полям класса

2)Наследование – объекты дочернего класса используют методы родительского(базового) класса

3)Полиморфизм – переопределенные методы базового класса в дочернем

# **Приложение. Код программы**