Черкасов Александр Андреевич А-08-19

**Задание**

**Вариант 1** Описать класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. **Принимаем диапазон [-2048, 2048]** Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. **Считаем значением по умолчанию 0**. Счетчик имеет два метода: увеличения и уменьшения, — и свойство, позволяющее получить его текущее состояние. При **выходе за границы** диапазона **выбрасываются исключения**, а **значения не изменяются**. Создать дочерний класс Счетчик1, который может увеличивать или уменьшать свое значение на любое введенное число в заданном диапазоне.

**Функции:**

***//----Конструкторы----//***

counter() / counter1()

***Входные данные***: нет

***Задается***начальное значение 0

counter(const int&) / counter1(const int&)

***Входные данные****:* начальное значение счетчика

***Ограничения****:* значение лежит в диапазоне [-2048 2048]

***Задается***начальное значение по входному

***Реакция на неверное начальное значение****:* выбрасывается runtime\_error, в интерфейсе вызов конструктора по-умолчанию

***Реакция на неверные входные данные****:* обрабатывается в интерфейсе, выбрасывается invalid\_argument

***//----Функции увеличения и уменьшения счетчика----//***

virtual void increase(); / void increase(const int&)

virtual void decrease(); / void decrease(const int&)

**Для counter**

***Входные данные***: нет

***Реакция на выход за переделы диапазона***: выбрасывается runtime\_error, значение не изменяется

**Для counter1**

***Входные данные***: значение для увеличения/уменьшения счетчика

***Ограничения***: целые числа

***Реакция на выход за переделы диапазона***: выбрасывается runtime\_error, значение не изменяется

***Реакция на неверные входные данные***: обрабатывается в интерфейсе, выбрасывается invalid\_argument

***//----Получение значения счетчика----//***

[[nodiscard]] int get() const;

***Входные данные***: значение для увеличения/уменьшения счетчик

**Вид приложения** – консольное. **Разработано** в среде CLion, **собрано** компилятором MinGW

**Разработка структуры приложения**

**Класс counter:**

**protected:**

***//----Поля, содержащие текущее значение и ограничения счетчика----//***

int minValue;

int maxValue;

int currentValue;

***//----Сообщения ошибок----//***

const std::string out\_of\_range\_error = "counter::out\_of\_range";

const std::string wrong\_range = "counter::wrong\_range";

**public:**

***//----Конструкторы----//***

counter();

counter(const int&);

***//----Виртуальные функции увеличения и уменьшения счетчика----//***

virtual void increase();

virtual void decrease();

***//----Получение значения счетчика----//***

[[nodiscard]] int get() const;

**Дочерний класс counter1:**

**public:**

**//----Конструкторы дочернего класса----//**

counter1();

counter1(const int&);

**//----Перезаписанные функции увеличения и уменьшения счетчика----//**

void increase(const int&);

void decrease(const int&);

3. Реализация и тестирование программы

Описание разработанной программы

Разработан класс счетчика, конструктор по умолчанию устанавливает начальное значение в 0. Конструктор, принимающий в целое число, устанавливает начальное значение по входному параметру. При выходе за диапазон выбрасывается ошибка runtime\_error. Методы класса: increase() decrease() get(). Увеличение, уменьшение и получение значения счетчика соответственно. При попытке выйти за границы, выбрасывается ошибка runtime\_error.

На основе счетчика, разработан дочерний класс счетчик1. Его конструкторы вызывают конструкторы родительского класса. Переписаны функции increase() и decrease() на increase(const int&) и decrease(const int&) для возможности задания значения, на которое нужно увеличить / уменьшить значения счетчика.

Класс «обернут» в интерфейс, с помощью которого пользователь может вызывать методы классов. В нем отображаются тип счетчика, диапазон и его текущее значение. Интерфейс способен обрабатывать входные данные, поэтому при попытке ввести вместо числа какую-либо строку, программа не сломается, а выведет сообщение о неверных входных данных.

3.2. Тестирование программы

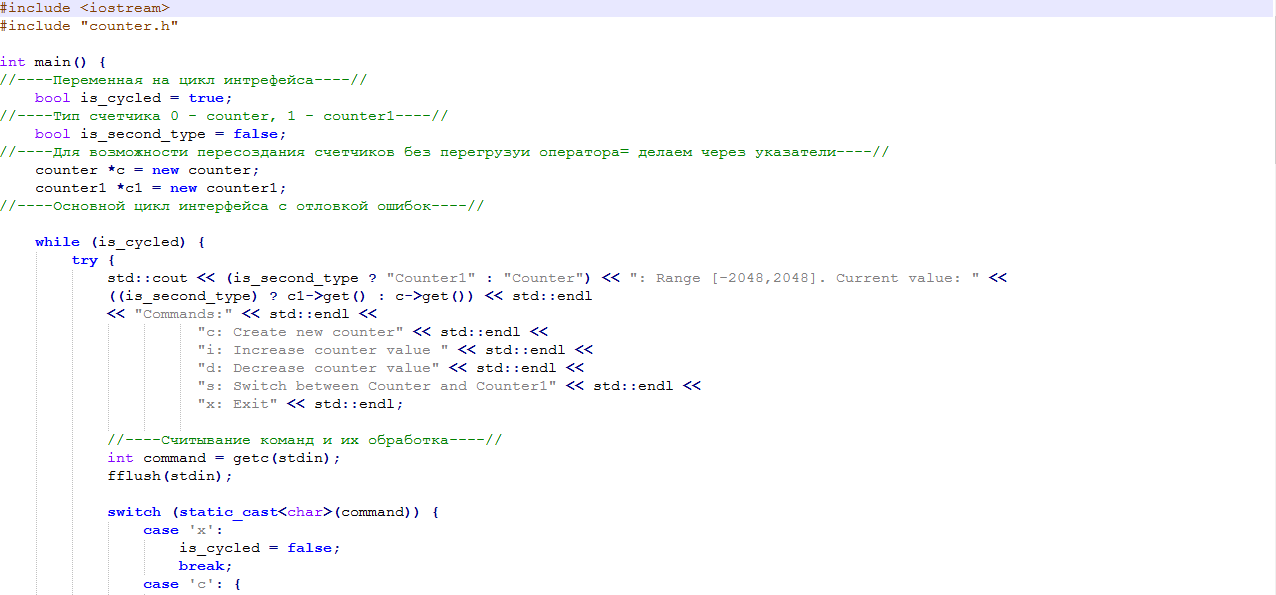
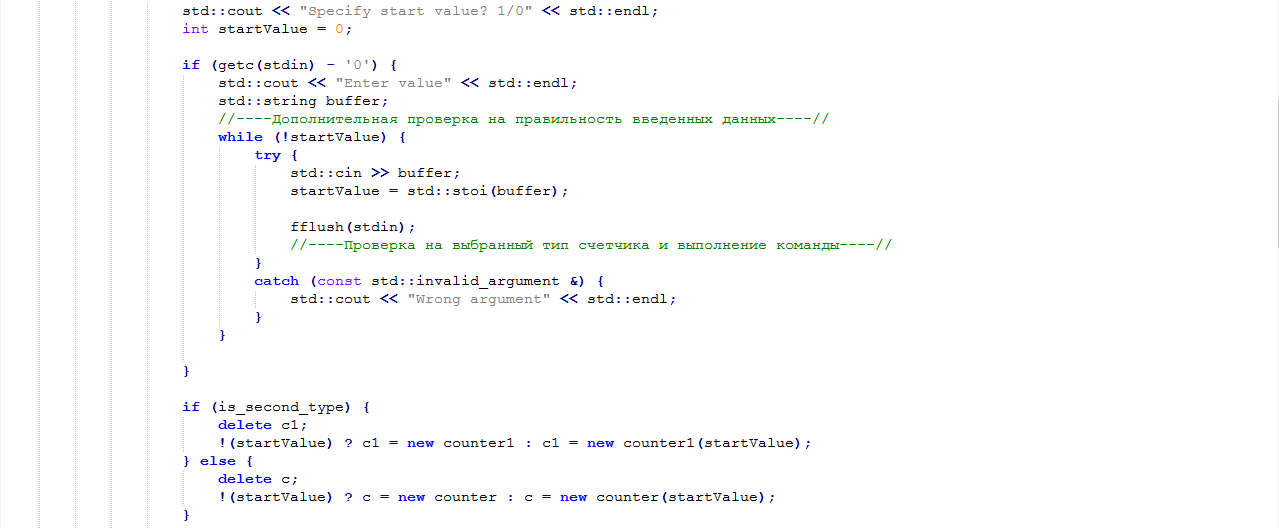
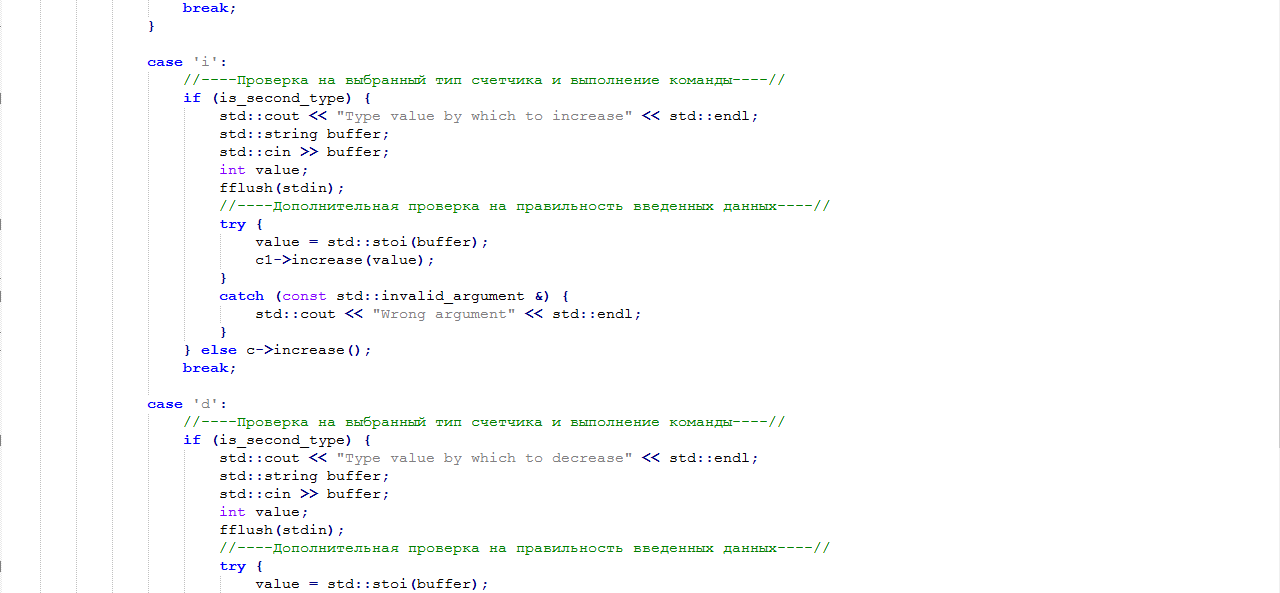
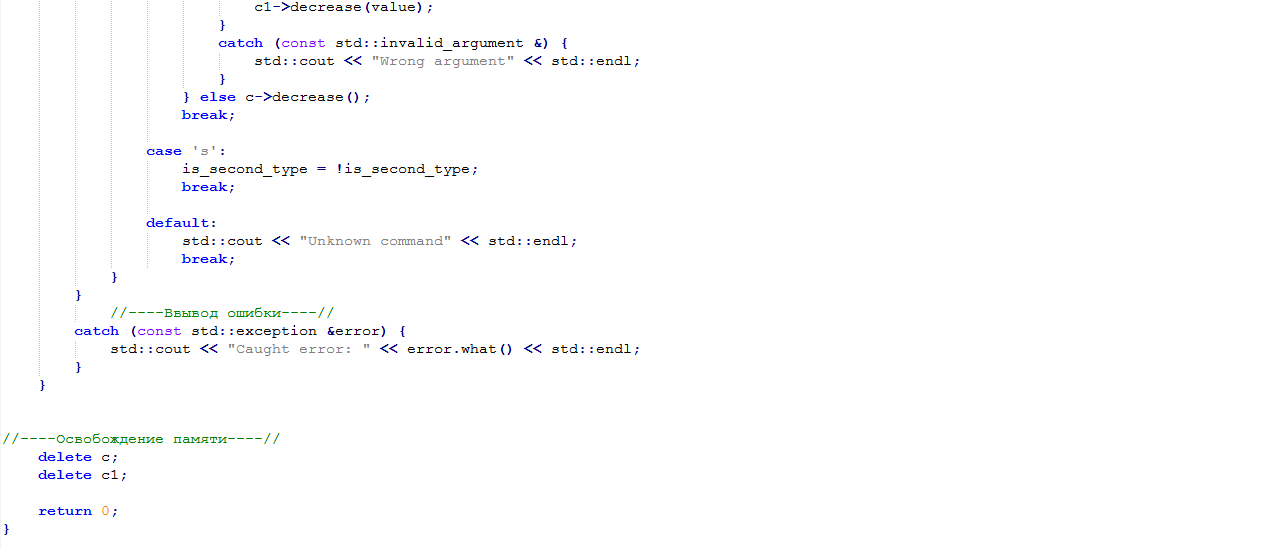
|  |  |  |  |
| --- | --- | --- | --- |
| **№ теста** | **Исходные данные** | **Ожидаемый  результат** | **Смысл теста** |
| **1** | counter()  counter1() | counter::get = 0  counter1::get = 0 | Проверка **стандартных конструкторов** |
| **2** | counter(10)  counter1(10) | counter::get = 10  counter1::get = 10 | Проверка **конструкторов с параметром** |
| **3** | counter()  counter1()  counter::increase()  counter1::increase(5) | counter::get = 1  counter::get = 5 | Проверка функции **increase** |
| **4** | counter()  counter1()  counter::decrease()  countet1::decrease(4) | counter::get = -1  counter1::get = -4 | Проверка функции **decrease** |
| **5** | сounter(2048)  counter::increase()  counter1(5)  counter1::increase(2045) | В всех случаях  std::runtime\_error  (“counter::out\_of\_range”) | Проверка выхода за **верхний предел** |
| **6** | counter(-2048)  counter::decrease()  counter1(-5)  counter1::decrease(-2048) | Проверка выхода за **нижний предел** |
| **7** | counter(2049)  counter1(4000) | Проверка выхода за **верхний предел** в **конструкторе** |
| **8** | counter(-4049)  counter1(-2048) | Проверка выхода за **нижний предел** в **конструкторе** |
| **9** | counter1()  counter1::increase(-1) | counter1::get = -1 | Проверка **отрицательного параметра** для **increase** |
| **10** | counter1()  counter1::decrease(-1) | counte1::get = 1 | Проверка **отрицательного параметра** для **decrease** |
| **11** | //Вызов в интерфейсе пересоздание счетчика  //Ввод случайной строки на предложении интерфейса ввести значение | std:: runtime\_error | Попытка ввести **строку** в параметры **конструктора** |
| **12** | //Вызов в интерфейсе increase для counter1  //Ввод случайной строки на предложении интерфейса ввести значение | Во всех случаях  std::invalid\_argument | Попытка ввести **строку** в параметры **increase** |
| **13** | //Вызов в интерфейсе decrease для counter1  //Ввод случайной строки на предложении интерфейса ввести значение | Попытка ввести **строку** в параметры **decrease** |

Вывод

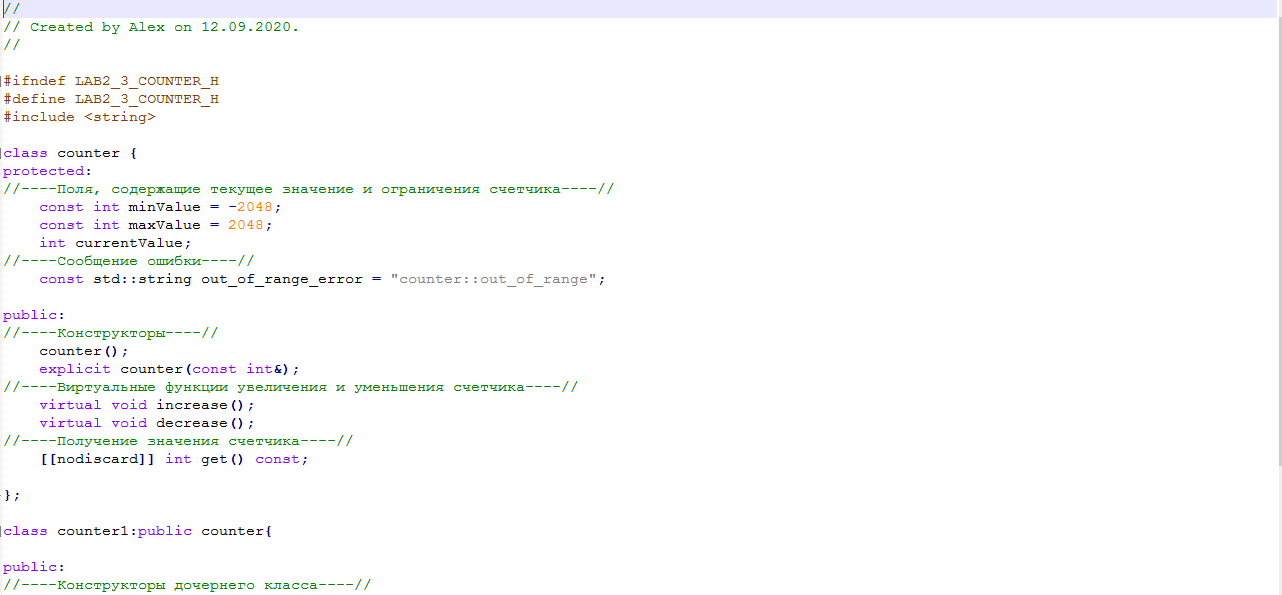
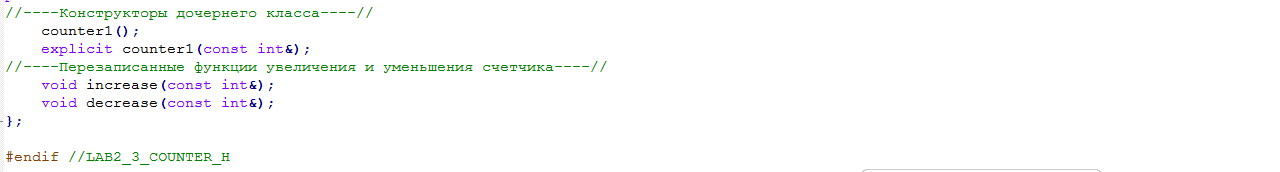
Научились созданию классов, использовать основные принципы ООП: инкапсуляцию, наследованию, полиморфизму.

Приложение. Листинг программы

**main.cpp**

**counter.h**

**сounter.cpp**

