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

**Содержание**

[**Задание (2.1)** 1](#_Toc51623285)

[**Задание (2.2)** 1](#_Toc51623286)

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

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

[2.1 Разработка функций и методов классов 2](#_Toc51623289)

[2.2 Разработка интерфейса 3](#_Toc51623290)

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

[3.1 Описание разработанной программы 3](#_Toc51623292)

[3.2 Тестирование программы 3](#_Toc51623293)

[**Вывод** 7](#_Toc51623294)

[**Приложение. Код программы** 7](#_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++.

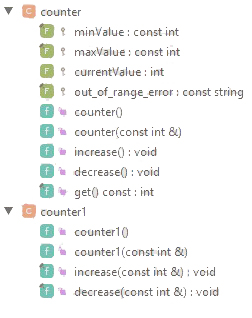
**Среда разработки** – CLion

# 

# **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 2.1 и 2.2 приведены поля, конструкторы и методы классов, а диаграмма приведена на рис.2.1

рис2.1

Табл.2.1

|  |  |
| --- | --- |
| **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 |

Табл.2.2

|  |  |
| --- | --- |
| **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

*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

|  |  |  |
| --- | --- | --- |
| **№ теста** | **Смысл теста** | **Результат** |
| **1** | Проверка **стандартных конструкторов** |  |
| **2** | Проверка **конструкторов с параметром** |  |
| **3** | Проверка функции **increase** |  |
| **4** | Проверка функции **decrease** |  |
| **5** | Проверка выхода за **верхний предел** |  |
| **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** |  |
| **10** | Проверка **отрицательного параметра** для **decrease** |  |
| **11** | Попытка ввести **строку** в параметры **конструктора** |  |
| **12** | Попытка ввести **строку** в параметры **increase** |  |
| **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**

**main.cpp**

*/\*  
 \* Лабораторная работа 2-3 Вар.1 Черкасов Александр А-08-19  
 \*/*#include **<iostream>**#include **"counter.h"  
  
int** main() {  
*//----Переменная на цикл интрефейса----//* **bool** is\_ui\_cycled = **true**;  
*//----Тип счетчика 0 - counter, 1 - counter1----//* **bool** is\_second\_type = **false**;  
*//----Для возможности пересоздания счетчиков без перегрузки оператора= делаем через указатели----//* counter \*c = **new** counter;  
 counter1 \*c1 = **new** counter1;  
*//----Основной цикл интерфейса с отловкой ошибок----//* **while** (is\_ui\_cycled) {  
 **try** {  
 std::cout << (is\_second\_type ? **"Counter1"** : **"Counter"**) << **": Range [-2048,2048]. Current value: "** <<  
 ((is\_second\_type) ? c1->get() : c->get()) << std::endl  
 << **"Commands:"** << std::endl <<  
 **"c: Create new counter"** << std::endl <<  
 **"i: Increase counter value "** << std::endl <<  
 **"d: Decrease counter value"** << std::endl <<  
 **"s: Switch between Counter and Counter1"** << std::endl <<  
 **"x: Exit"** << std::endl;  
  
 *//----Считывание команд и их обработка----//* **int** command = getc(**stdin**);  
 fflush(**stdin**);  
  
 **switch** (**static\_cast**<**char**>(command)) {  
 **case 'x'**:  
 is\_ui\_cycled = **false**;  
 **break**;  
 **case 'c'**: {  
 std::cout << **"Specify start value? 1/0"** << std::endl;  
 **int** startValue = 0;  
  
 **if** (getc(**stdin**) - **'0'**) {  
 std::cout << **"Enter value"** << std::endl;  
 std::string buffer;  
 *//----Дополнительная проверка на правильность введенных данных----//* **while** (!startValue) {  
 **try** {  
 std::cin >> buffer;  
 startValue = std::stoi(buffer);  
  
 fflush(**stdin**);  
 *//----Проверка на выбранный тип счетчика и выполнение команды----//* }  
 **catch** (**const** std::invalid\_argument &) {  
 std::cout << **"Wrong argument"** << std::endl;  
 }  
 }  
  
 }  
  
 **if** (is\_second\_type) {  
 **delete** c1;  
 **try** {  
 !(startValue) ? c1 = **new** counter1 : c1 = **new** counter1(startValue);  
 }  
 **catch** (...) {  
 std::cout << **"Out of range. Value set to zero\n"**;  
 c1 = **new** counter1;  
 }  
 } **else** {  
 **delete** c;  
 **try** {  
 !(startValue) ? c = **new** counter : c = **new** counter(startValue);  
 }  
 **catch** (...) {  
 std::cout << **"Out of range. Value set to zero\n"**;  
 c = **new** counter;  
 }  
 }  
  
 **break**;  
 }  
  
 **case 'i'**:  
 *//----Проверка на выбранный тип счетчика и выполнение команды----//* **if** (is\_second\_type) {  
 std::cout << **"Type value by which to increase"** << std::endl;  
 std::string buffer;  
 std::cin >> buffer;  
 **int** value;  
 fflush(**stdin**);  
 *//----Дополнительная проверка на правильность введенных данных----//* **try** {  
 value = std::stoi(buffer);  
 c1->increase(value);  
 }  
 **catch** (**const** std::invalid\_argument &) {  
 std::cout << **"Wrong argument"** << std::endl;  
 }  
 } **else** c->increase();  
 **break**;  
  
 **case 'd'**:  
 *//----Проверка на выбранный тип счетчика и выполнение команды----//* **if** (is\_second\_type) {  
 std::cout << **"Type value by which to decrease"** << std::endl;  
 std::string buffer;  
 std::cin >> buffer;  
 **int** value;  
 fflush(**stdin**);  
 *//----Дополнительная проверка на правильность введенных данных----//* **try** {  
 value = std::stoi(buffer);  
 c1->decrease(value);  
 }  
 **catch** (**const** std::invalid\_argument &) {  
 std::cout << **"Wrong argument"** << std::endl;  
 }  
 } **else** c->decrease();  
 **break**;  
  
 **case 's'**:  
 is\_second\_type = !is\_second\_type;  
 **break**;  
  
 **default**:  
 std::cout << **"Unknown command"** << std::endl;  
 **break**;  
 }  
 }  
 *//----Ввывод ошибки----//* **catch** (**const** std::exception &error) {  
 std::cout << **"Caught error: "** << error.what() << std::endl;  
 }  
 }  
  
  
*//----Освобождение памяти----//* **delete** c;  
 **delete** c1;  
 **return** 0;  
}

**counter.h**

#ifndef **LAB2\_3\_COUNTER\_H**#define **LAB2\_3\_COUNTER\_H**#include **<string>  
  
class** counter {  
**protected**:  
*//----Поля, содержащие текущее значение и ограничения счетчика----//* **const int** minValue = -2048;  
 **const int** maxValue = 2048;  
 **int** currentValue;  
*//----Сообщение ошибки----//* **const** std::string out\_of\_range\_error = **"counter::out\_of\_range"**;  
  
**public**:  
*//----Конструкторы----//* counter();  
 **explicit** counter(**const int**&);  
*//----Виртуальные функции увеличения и уменьшения счетчика----//* **virtual void** increase();  
 **virtual void** decrease();  
*//----Получение значения счетчика----//* [[nodiscard]] **int** get() **const**;  
  
};  
  
**class** counter1:**public** counter{  
  
**public**:  
*//----Конструкторы дочернего класса----//* counter1();  
 **explicit** counter1(**const int**&);  
*//----Перезаписанные функции увеличения и уменьшения счетчика----//* **void** increase(**const int**&);  
 **void** decrease(**const int**&);  
};  
  
#endif *//LAB2\_3\_COUNTER\_H*

**counter.cpp**

#include **"counter.h"**#include **<stdexcept>***//----Стандратный констуктор----//*counter::counter():currentValue(0) {  
}  
*//----Конструктор с указанием текущего значения для счетчика----//*counter::counter(**const int**& curVal):currentValue(curVal) {  
 **if** (currentValue<minValue || currentValue>maxValue) {  
 currentValue=0;  
 **throw** std::runtime\_error(out\_of\_range\_error);  
 }  
}  
  
*//----Увеличение и уменьшение счетчика на единицу с проверкой на выход за пределы диапазона----//***void** counter::increase() {  
 **if**(++currentValue>maxValue) {  
 --currentValue;  
 **throw** std::runtime\_error(out\_of\_range\_error);  
 }  
}  
  
**void** counter::decrease() {  
 **if** (--currentValue<minValue) {  
 ++currentValue;  
 **throw** std::runtime\_error(out\_of\_range\_error);  
 }  
}  
*//----Получение значения----//***int** counter::get() **const** {  
 **return** currentValue;  
}  
  
*//----Констукторы дочернего класса, вызывающий родительский----//*counter1::counter1():counter(){  
  
}  
  
counter1::counter1(**const int**& curVal) : counter(curVal){  
}  
  
*//----Увеличение и уменьшение на произвольное значение с проверкой на выход за пределы----//***void** counter1::increase(**const int**& value) {  
 currentValue+=value;  
 **if** (currentValue<minValue || currentValue>maxValue) {  
 currentValue-=value;  
 **throw** std::runtime\_error(out\_of\_range\_error);  
 }  
}  
  
**void** counter1::decrease(**const int**& value) {  
 currentValue-=value;  
 **if** (currentValue<minValue || currentValue>maxValue) {  
 currentValue+=value;  
 **throw** std::runtime\_error(out\_of\_range\_error);  
 }  
}