Министерство цифрового развития, связи и массовых коммуникаций

Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования «Сибирский государственный университет

телекоммуникаций и информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа

«Шаблон класса процессор»

Выполнил:

Студент группы ИП-017

Костин А.В.

Работу проверил:

ассистент кафедры ПМиК

Агалаков А.А.

Новосибирск 2023 г.

Содержание

[1. Задание 3](#_Toc153303987)

[2. Исходный код программы 7](#_Toc153303988)

[2.1. Код программы 7](#_Toc153303989)

[2.2. Код тестов 12](#_Toc153303990)

[3. Результаты модульных тестов 17](#_Toc153303991)

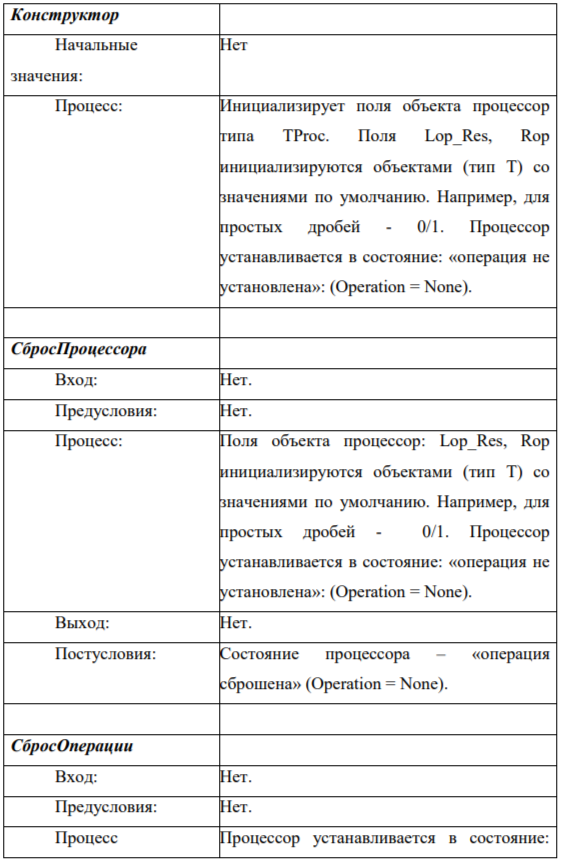
[4. Вывод 18](#_Toc153303992)

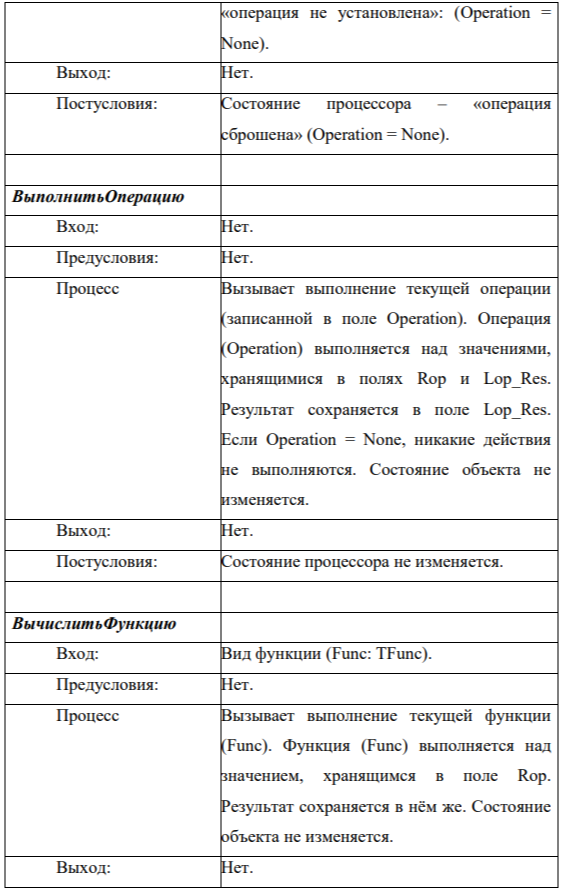
1. **Задание**
2. В соответствии с приведенной ниже спецификацией реализовать параметризованный абстрактный тип данных «Процессор», используя шаблон классов С++.
3. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.
4. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

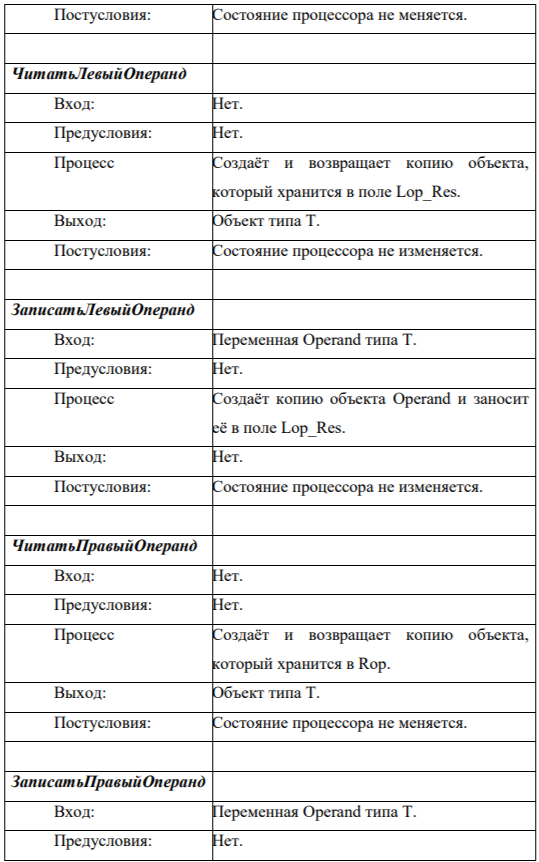
Данные

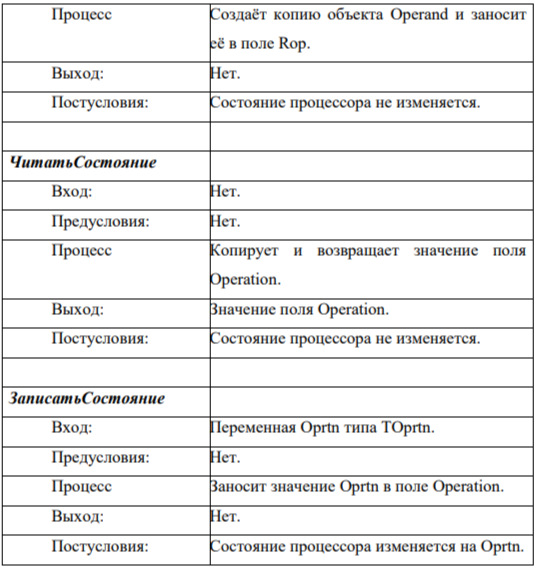
Процессор (тип TProc) выполняет двухоперандные операции TOprtn = (None, Add, Sub, Mul, Dvd) и однооперандные операции - функции TFunc = (Rev, Sqr) над значениями типа T. Левый операнд и результат операции хранится в поле Lop\_Res, правый - в поле Rop. Оба поля имеют тип T. Процессор может находиться в состояниях: «операция установлена» - поле Operation не равно None (значение типа TOprtn) или в состоянии «операция не установлена» - поле Operation = None. Значения типа TProc - изменяемые. Они изменяются операциями: «Сброс операции» (OprtnClear), «Выполнить операцию» (OprtnRun), «Вычислить функцию» (FuncRun), «Установить операцию» (OprtnSet), «Установить левый операнд» (Lop\_Res\_Set), «Установить правый операнд» (Rop\_Set), «Сброс калькулятора» (ReSet). На значениях типа T должны быть определены указанные выше операции и функции.

Операции









1. **Исходный код программы**
   1. **Код программы**

**TProc.h**

#pragma once

#include <algorithm>

enum EOporation {

none,

add,

sub,

mul,

dvd

};

enum EFunction {

rev,

sqr

};

template<class T>

class TProc

{

private:

int operation = 0;

T accumulator;

T parametr;

public:

TProc();

void resetProcessor();

void resetOperation();

void runOperation();

void runFunction(int func);

T getAccumulator();

T getParametr();

int getOperation();

void setAccumulator(T a);

void setParametr(T a);

void setOperation(int a);

};

template<class T>

TProc<T>::TProc()

{

resetProcessor();

}

template<class T>

void TProc<T>::resetProcessor()

{

operation = none;

accumulator = T();

parametr = T();

}

template<class T>

void TProc<T>::resetOperation()

{

operation = none;

}

template<class T>

void TProc<T>::runOperation()

{

switch (operation) {

case none:

return;

case add:

accumulator = accumulator + parametr;

return;

case sub:

accumulator = accumulator - parametr;

return;

case mul:

accumulator = accumulator \* parametr;

return;

case dvd:

accumulator = accumulator / parametr;

return;

default:

return;

}

}

template<class T>

void TProc<T>::runFunction(int func)

{

switch (func) {

case rev:

accumulator = -accumulator;

return;

case sqr:

accumulator = sqrt(accumulator);

return;

default:

return;

}

}

template<class T>

T TProc<T>::getAccumulator()

{

return accumulator;

}

template<class T>

T TProc<T>::getParametr()

{

return parametr;

}

template<class T>

int TProc<T>::getOperation()

{

return operation;

}

template<class T>

void TProc<T>::setAccumulator(T a)

{

accumulator = a;

}

template<class T>

void TProc<T>::setParametr(T a)

{

parametr = a;

}

template<class T>

void TProc<T>::setOperation(int a)

{

operation = a;

}

* 1. **Код тестов**

**UnitTest.cpp**

#include "pch.h"

#include "CppUnitTest.h"

#include "../ModernCodingProcessor/TProc.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest)

{

public:

TEST\_METHOD(TestResetProcessor)

{

TProc<int> a;

a.resetProcessor();

int result = a.getOperation();

int expected = none;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestResetOperation)

{

TProc<int> a;

a.resetOperation();

int result = a.getOperation();

int expected = none;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestAcumulator)

{

TProc<int> a;

a.setAccumulator(5);

int result = a.getAccumulator();

int expected = 5;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TesParametr)

{

TProc<int> a;

a.setParametr(5);

int result = a.getParametr();

int expected = 5;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestOperation)

{

TProc<int> a;

a.setOperation(add);

int result = a.getOperation();

int expected = add;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestRev)

{

TProc<int> a;

a.setAccumulator(25);

a.runFunction(rev);

int result = a.getAccumulator();

int expected = -25;

Assert::AreEqual(expected, result);

}

TEST\_METHOD(TestSqr)

{

TProc<int> a;

a.setAccumulator(25);

a.runFunction(sqr);

int result = a.getAccumulator();

int expected = 5;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestAdd)

{

TProc<int> a;

a.setAccumulator(25);

a.setParametr(5);

a.setOperation(add);

a.runOperation();

int result = a.getAccumulator();

int expected = 30;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestSub)

{

TProc<int> a;

a.setAccumulator(25);

a.setParametr(5);

a.setOperation(sub);

a.runOperation();

int result = a.getAccumulator();

int expected = 20;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestMul)

{

TProc<int> a;

a.setAccumulator(25);

a.setParametr(5);

a.setOperation(mul);

a.runOperation();

int result = a.getAccumulator();

int expected = 125;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestDvd)

{

TProc<int> a;

a.setAccumulator(25);

a.setParametr(5);

a.setOperation(dvd);

a.runOperation();

int result = a.getAccumulator();

int expected = 5;

Assert::AreEqual(result, expected);

}

TEST\_METHOD(TestNone)

{

TProc<int> a;

a.setAccumulator(25);

a.setParametr(5);

a.setOperation(none);

a.runOperation();

int result = a.getAccumulator();

int expected = 25;

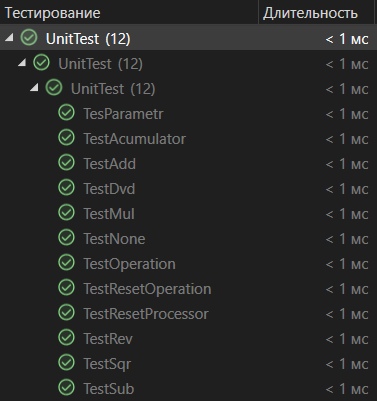
Assert::AreEqual(result, expected);

}

};

}

1. **Результаты модульных тестов**



1. **Вывод**

По итогам данной лабораторной работе были сформированы практические навыки реализации шаблона класса процессор с помощью классов С++ и их модульного тестирования.