Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

**Лабораторная работа № 12**

**по дисциплине «Современные технологии программирования»**

**«Параметризованный абстрактный тип данных “Процессор”»**

Выполнил: студент 4 курса ф. ИВТ, гр. ИП-813

Пещеров Вячеслав Александрович

Проверил: ассистент

к. ПМиК Агалаков А.А.

Новосибирск, 2021

Оглавление

[**Задание** 3](#_Toc90052181)

[**Реализация** 8](#_Toc90052182)

[**Демонстрация работы** 9](#_Toc90052183)

[**Вывод** 10](#_Toc90052184)

[**Список литературы** 11](#_Toc90052185)

[**Приложение** 12](#_Toc90052186)

[Листинг 1. TProcUnitTests.cpp 12](#_Toc90052187)

[Листинг 2. TProc.h 17](#_Toc90052188)

[Листинг 3. TProc.cpp 18](#_Toc90052189)

# **Задание**

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

Спецификация типа данных «Процессор». ADT **TProc**

**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

|  |  |
| --- | --- |
| ***Конструктор*** |  |
| Начальные  значения: | Нет |
| Процесс: | Инициализирует поля объекта процессор типа TProc. Поля Lop\_Res, Rop инициализируются объектами (тип T) со значениями по умолчанию. Например, для простых дробей - 0/1. Процессор устанавливается в состояние: «операция не  установлена»: (Operation = None). |
|  |  |
| ***СбросПроцессора*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Поля объекта процессор: Lop\_Res, Rop инициализируются объектами (тип T) со значениями по умолчанию. Например, для простых дробей - 0/1. Процессор устанавливается в состояние: «операция не  установлена»: (Operation = None). |
| Выход: | Нет. |
| Постусловия: | Состояние процессора – «операция  сброшена» (Operation = None). |
|  |  |
| ***СбросОперации*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Процессор устанавливается в состояние: |

|  |  |
| --- | --- |
|  | «операция не установлена»: (Operation =  None). |
| Выход: | Нет. |
| Постусловия: | Состояние процессора – «операция  сброшена» (Operation = None). |
|  |  |
| ***ВыполнитьОперацию*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Вызывает выполнение текущей операции (записанной в поле Operation). Операция (Operation) выполняется над значениями, хранящимися в полях Rop и Lop\_Res. Результат сохраняется в поле Lop\_Res. Если Operation = None, никакие действия не выполняются. Состояние объекта не  изменяется. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора не изменяется. |
|  |  |
| ***ВычислитьФункцию*** |  |
| Вход: | Вид функции (Func: TFunc). |
| Предусловия: | Нет. |
| Процесс | Вызывает выполнение текущей функции (Func). Функция (Func) выполняется над значением, хранящимся в поле Rop. Результат сохраняется в нём же. Состояние  объекта не изменяется. |
| Выход: | Нет. |

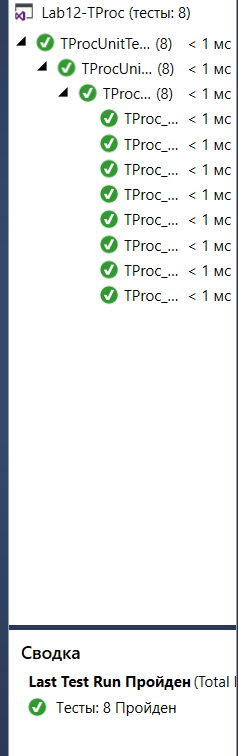
|  |  |
| --- | --- |
| Постусловия: | Состояние процессора не меняется. |
|  |  |
| ***ЧитатьЛевыйОперанд*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает копию объекта,  который хранится в поле Lop\_Res. |
| Выход: | Объект типа T. |
| Постусловия: | Состояние процессора не изменяется. |
|  |  |
| ***ЗаписатьЛевыйОперанд*** |  |
| Вход: | Переменная Operand типа T. |
| Предусловия: | Нет. |
| Процесс | Создаёт копию объекта Operand и заносит  её в поле Lop\_Res. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора не изменяется. |
|  |  |
| ***ЧитатьПравыйОперанд*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает копию объекта,  который хранится в Rop. |
| Выход: | Объект типа T. |
| Постусловия: | Состояние процессора не меняется. |
|  |  |
| ***ЗаписатьПравыйОперанд*** |  |
| Вход: | Переменная Operand типа T. |
| Предусловия: | Нет. |

|  |  |
| --- | --- |
| Процесс | Создаёт копию объекта Operand и заносит  её в поле Rop. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора не изменяется. |
|  |  |
| ***ЧитатьСостояние*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Копирует и возвращает значение поля  Operation. |
| Выход: | Значение поля Operation. |
| Постусловия: | Состояние процессора не изменяется. |
|  |  |
| ***ЗаписатьСостояние*** |  |
| Вход: | Переменная Oprtn типа TOprtn. |
| Предусловия: | Нет. |
| Процесс | Заносит значение Oprtn в поле Operation. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора изменяется на Oprtn. |

# **Реализация**

В данной работе мы создали параметризованный абстрактный тип данных “Процессор” и все необходимые для работы с ним операции, такие как: конструктор, сброс процессора, сброс операции, выполнить операцию, вычислить функцию, читать левый операнд, запись левого операнда, считывание правого операнда и его запись, считывание состояние и запись этого состояния.

# **Демонстрация работы**



# **Вывод**

Мы научились работать в среде Visual Studio, а именно разрабатывать в ней модульные тесты для тестирования наших функции и классов на языке C#. Данная среда отлично подходит для выполнения модульного тестирования и автоматизации.

# **Список литературы**

1. Подбельский В.В., Фомин С.С.инт Курс программирования на языке Си: учебник. – М.:ДМК Пресс, 2012 – 384 с.
2. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. - СПб. : Питер, 2014 - 432 с. : ил. - (Серия "Учебник для вузов").
3. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4 на языке C# . 3-е изд.: - СПб.:Питер, 2012 - 928 с. : ил.
4. Котляров, В. П. Основы тестирования программного обеспечения : учебное пособие для СПО / В. П. Котляров. — Саратов : Профобразование, 2019 — 335 c. — ISBN 978-5-4488-0364-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/86202.html (дата обращения: 21.08.2020). — Режим доступа: для авторизир. пользователей

# **Приложение**

## Листинг 1. TProcUnitTests.cpp

#include "pch.h"

#include "../TFrac.h"

#include "../TProc.cpp"

#include "CppUnitTest.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TProcUnitTests

{

TEST\_CLASS(TProcUnitTests)

{

public:

TEST\_METHOD(TProc\_tfrac\_constructor\_0\_1\_None)

{

TProc<TFrac> test = TProc<TFrac>();

int expected\_rop = 1;

int expected\_lop\_res = 0;

TOprtn expected\_operation = None;

int actual\_rop = test.get\_rop().getznam();

int actual\_lop\_res = test.get\_rop().getchisl();

TOprtn actual\_operation = test.get\_operation();

Assert::IsTrue(expected\_rop== actual\_rop);

Assert::IsTrue(expected\_lop\_res== actual\_lop\_res);

Assert::IsTrue(expected\_operation== actual\_operation);

}

TEST\_METHOD(TProc\_tfrac\_set\_rop\_lop\_res\_and\_operation\_1\_2\_Add)

{

TProc<TFrac> test = TProc<TFrac>();

TFrac d(2, 1);

test.set\_rop(d);

test.set\_lop\_res(d);

test.set\_operation(Add);

int expected\_rop = 1;

int expected\_lop\_res = 2;

TOprtn expected\_operation = Add;

int actual\_rop = test.get\_rop().getznam();

int actual\_lop\_res = test.get\_rop().getchisl();

TOprtn actual\_operation = test.get\_operation();

Assert::IsTrue(expected\_rop== actual\_rop);

Assert::IsTrue(expected\_lop\_res== actual\_lop\_res);

Assert::IsTrue(expected\_operation== actual\_operation);

}

TEST\_METHOD(TProc\_tfrac\_execute\_operation\_add\_2\_1\_2\_1\_4\_1)

{

TProc<TFrac> actual = TProc<TFrac>();

TFrac d(2, 1);

TFrac expected(4, 1);

actual.set\_rop(d);

actual.set\_lop\_res(d);

actual.set\_operation(Add);

actual.execute\_operation();

Assert::IsTrue(expected.getchisl()== actual.get\_lop\_res().getchisl());

Assert::IsTrue(expected.getznam()== actual.get\_lop\_res().getznam());

}

TEST\_METHOD(TProc\_tfrac\_execute\_operation\_sub\_2\_1\_2\_1\_0\_1)

{

TProc<TFrac> actual = TProc<TFrac>();

TFrac d(2, 1);

TFrac expected(0, 1);

actual.set\_rop(d);

actual.set\_lop\_res(d);

actual.set\_operation(Sub);

actual.execute\_operation();

Assert::IsTrue(expected.getchisl()==actual.get\_lop\_res().getchisl());

Assert::IsTrue(expected.getznam()== actual.get\_lop\_res().getznam());

}

TEST\_METHOD(TProc\_tfrac\_execute\_operation\_mul\_2\_1\_2\_1\_4\_1)

{

TProc<TFrac> actual = TProc<TFrac>();

TFrac d(2, 1);

TFrac expected(4, 1);

actual.set\_rop(d);

actual.set\_lop\_res(d);

actual.set\_operation(Mul);

actual.execute\_operation();

Assert::IsTrue(expected.getchisl()== actual.get\_lop\_res().getchisl());

Assert::IsTrue(expected.getznam()== actual.get\_lop\_res().getznam());

}

TEST\_METHOD(TProc\_tfrac\_execute\_operation\_div\_2\_1\_2\_1\_1\_1)

{

TProc<TFrac> actual = TProc<TFrac>();

TFrac d(2, 1);

TFrac expected(1, 1);

actual.set\_rop(d);

actual.set\_lop\_res(d);

actual.set\_operation(Div);

actual.execute\_operation();

Assert::IsTrue(expected.getchisl()== actual.get\_lop\_res().getchisl());

Assert::IsTrue(expected.getznam()== actual.get\_lop\_res().getznam());

}

TEST\_METHOD(TProc\_tfrac\_execute\_function\_reverse\_2\_1\_1\_2)

{

TProc<TFrac> actual = TProc<TFrac>();

TFrac d(2, 1);

TFrac expected(1, 2);

actual.set\_rop(d);

actual.set\_lop\_res(d);

actual.set\_operation(Mul);

actual.execute\_function(Rev);

Assert::IsTrue(expected.getchisl()== actual.get\_rop().getchisl());

Assert::IsTrue(expected.getznam()== actual.get\_rop().getznam());

}

TEST\_METHOD(TProc\_tfrac\_execute\_operation\_square\_2\_1\_4\_1)

{

TProc<TFrac> actual = TProc<TFrac>();

TFrac d(2, 1);

TFrac expected(4, 1);

actual.set\_rop(d);

actual.set\_lop\_res(d);

actual.set\_operation(Div);

actual.execute\_function(Sqr);

Assert::IsTrue(expected.getchisl()== actual.get\_rop().getchisl());

Assert::IsTrue(expected.getznam()==actual.get\_rop().getznam());

}

};

}

## Листинг 2. TProc.h

#pragma once

enum TOprtn

{

None,

Add,

Sub,

Mul,

Div

};

enum TFunc

{

Rev,

Sqr

};

template <class T>

class TProc

{

private:

T lop\_res;

T rop;

TOprtn operation;

public:

TProc();

void reset\_proc();

void reset\_operation();

void execute\_operation();

void execute\_function(enum TFunc func);

T get\_lop\_res() const;

void set\_lop\_res(const T& set);

T get\_rop() const;

void set\_rop(const T& set);

TOprtn get\_operation() const;

void set\_operation(TOprtn set);

};

## Листинг 3. TProc.cpp

#include "pch.h"

#include "TProc.h"

#include <iostream>

template <class T>

TProc<T>::TProc()//конструктор

{

rop = T();

lop\_res = T();

operation = None;

}

template<class T>

void TProc<T>::reset\_proc()//сбросить процессор

{

rop = T();

lop\_res = T();

operation = None;

}

template<class T>

void TProc<T>::reset\_operation()//сброс операции

{

operation = None;

}

template<class T>

void TProc<T>::execute\_operation()//выполнение операции

{

switch (operation)

{

case Add:

lop\_res = lop\_res.add(rop);

break;

case Sub:

lop\_res = lop\_res.sub(rop);

break;

case Mul:

lop\_res = lop\_res.mul(rop);

break;

case Div:

lop\_res = lop\_res.div(rop);

break;

case None:

default:

break;

}

}

template<class T>

void TProc<T>::execute\_function(enum TFunc func)//выполненние функции

{

switch (func)

{

case Rev:

rop = rop.reverse();

break;

case Sqr:

rop = rop.square();

break;

default:

break;

}

}

template<class T>

T TProc<T>::get\_lop\_res() const//считываем левый операнд

{

return lop\_res;

}

template<class T>

void TProc<T>::set\_lop\_res(const T& set)//записать левый операнд

{

lop\_res = set;

}

template<class T>

T TProc<T>::get\_rop() const//получаем правый операнд

{

return rop;

}

template<class T>

void TProc<T>::set\_rop(const T& set)//записываем правый операнд

{

rop = set;

}

template<class T>

TOprtn TProc<T>::get\_operation() const//считывание состояния

{

return operation;

}

template<class T>

void TProc<T>::set\_operation(TOprtn set)//запись состояния

{

operation = set;

}