

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 память на одно число»

Выполнил:
Студент группы ИП-017
Костин А.В.
Работу проверил:
ассистент кафедры ПМиК
Агалаков А.А.

Новосибирск 2023 г.

Содержание

1. Задание	3
2. Исходный код программы.....	6
2.1. Код программы	6
2.2. Код тестов.....	8
3. Результаты модульных тестов	11
4. Вывод	12

1. Задание

1. В соответствии с приведенной ниже спецификацией реализовать параметризованный абстрактный тип данных «память», для хранения одного числа – объекта типа T, используя шаблон классов C++.
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

Данные

Память (тип TMemory, в дальнейшем - память) - это память для хранения «числа» объекта типа T в поле FNumber, и значения «состояние памяти» в поле FState. Объект память - изменяемый. Он имеет два состояния, обозначаемых значениями: «Включена» (_On), «Выключена» (_Off). Её изменяют операции: Записать (Store), Добавить (Add), Очистить (Clear).

Операции

Конструктор	
Начальные значения:	Нет.
Процесс:	Инициализирует поле FNumber объекта «память» (тип TMemory) объектом «число» (тип T) со значением по умолчанию.

	Например для числа типа TFrac со значением 0/1. Память устанавливается в состояние «Выключена», в поле FState «состояние памяти» заносится значение (_Off).
Записать	
Вход:	(E) – объект тип T.
Предусловия:	Нет.
Процесс:	В объект «память» (тип TMemory) в поле FNumber записывается копия объекта E. Память устанавливается в состояние «Включена», в поле FState «состояние памяти» заносится значение (_On).
Выход:	Нет.
Постусловия:	Состояние памяти поле FState – «Включена» (_On).
Взять	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт и возвращает копию объекта хранящегося в объекте «память» (тип TMemory) в поле FNumber.
Выход:	Объект типа T.
Постусловия:	Состояние памяти поле FState – «Включена» (_On).
Добавить	
Вход:	(E) – число объект типа T.

Предусловия:	Нет.
Процесс:	В поле FNumber объекта «память» (тип TMemory) записывается объект типа T, полученный в результате сложения числа (E) и числа, хранящегося в памяти в поле FNumber.
Выход:	Нет.
Постусловия:	Состояние памяти поле FState – «Включена» (_On).
Очистить	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	В поле числа (FNumber) объекта «память» (тип TMemory) записывается объект типа T со значением по умолчанию. Например, для простой дроби - 0/1. Память (поле FState) устанавливается в состояние «Выключена» (_Off).
Выход:	Нет.
Постусловия:	Состояние памяти поле FState – «Выключена» (_Off).
Читать Состояние Памяти	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Копирует и возвращает значение поля FState «состояние памяти» объекта «память» (тип TMemory) в формате строки.

Выход:	Значение поля «состояния памяти» (типа String).
Постусловия:	Нет.
<i>ЧитатьЧисло</i>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Копирует и возвращает значение поля «число» (FNumber) объекта «память» (тип TMemory).
Выход:	Объект число (тип T).
Постусловия:	Нет.

2. Исходный код программы

2.1. Код программы

TMemory.h

```
#pragma once
#pragma once
#include <string>

#define _On true
#define _Off false

template<class T>
class TMemory
{
    T FNumber;
    bool FState;
public:
    TMemory();
    void write(T E);
    T read();
    void add(T E);
    void clear();
    std::string getState();
};

template<class T>
```

```

TMemory<T>::TMemory()
{
    FNumber = T();
    FState = _Off;
}

template<typename T>
void TMemory<T>::write(T E)
{
    if (FState == _Off) {
        FNumber = E;
        FState = _On;
    }
}

template<typename T>
T TMemory<T>::read()
{
    return FNumber;
}

template<typename T>
void TMemory<T>::add(T E)
{
    FNumber = FNumber + E;
    FState = _On;
}

template<typename T>
void TMemory<T>::clear()
{
    FNumber = T();
    FState = _Off;
}

template<typename T>
std::string TMemory<T>::getState()
{
    std::string result;
    if (FState == _On) {
        result = "On";
    }
    else {
        result = "Off";
    }
}

```

```

    }
    return result;
}

```

2.2. Код тестов

UnitTest.cpp

```

#include "pch.h"
#include "CppUnitTest.h"
#include "../ModernCodingMemory/TMemory.h"
#include <string>

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest
{
    TEST_CLASS(UnitTest)
    {
    public:

        TEST_METHOD(TestWrite)
        {
            TMemory<int> memory;
            memory.write(5);
            int result = memory.read();
            int expected = 5;
            Assert::AreEqual(result, expected);
        }

        TEST_METHOD(TestAdd)
        {

```



```

    TMemory<int> memory;
    memory.add(5);
    int result = memory.read();
    int expected = 5;
    Assert::AreEqual(result, expected);
}

```

```

TEST_METHOD(TestClear)
{
    TMemory<int> memory;
    memory.write(5);
    memory.clear();
    int result = memory.read();
    int expected = 0;
    Assert::AreEqual(result, expected);
}

```

```

TEST_METHOD(TestState)
{
    TMemory<int> memory;
    std::string result = memory.getState();
    std::string expected = "Off";
    Assert::AreEqual(result, expected);
}

```

```

TEST_METHOD(TestState2)
{
    TMemory<int> memory;
    memory.write(5);

```

```

        std::string result = memory.getState();
        std::string expected = "On";
        Assert::AreEqual(result, expected);
    }

    TEST_METHOD(TestState3)
    {
        TMemory<int> memory;
        memory.write(5);
        memory.clear();
        std::string result = memory.getState();
        std::string expected = "Off";
        Assert::AreEqual(result, expected);
    }
};
}

```

3. Результаты модульных тестов

Тестирование	Длительность
▲ ✓ UnitTest (6)	< 1 мс
▲ ✓ UnitTest (6)	< 1 мс
▲ ✓ UnitTest (6)	< 1 мс
✓ TestAdd	< 1 мс
✓ TestClear	< 1 мс
✓ TestState	< 1 мс
✓ TestState2	< 1 мс
✓ TestState3	< 1 мс
✓ TestWrite	< 1 мс

4. Вывод

По итогам данной лабораторной работе были сформированы практические навыки реализации шаблона память на одно число с помощью классов С++ и их модульного тестирования.