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 № 14**

**по дисциплине «Современные технологии программирования»**

**«Абстрактный тип данных (ADT) “Полином”»**

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

Пещеров Вячеслав Александрович

Проверил: ассистент

к. ПМиК Агалаков А.А.

Новосибирск, 2021

Оглавление

[**Задание** 4](#_Toc90054445)

[**Реализация** 8](#_Toc90054446)

[**Демонстрация работы** 9](#_Toc90054447)

[**Вывод** 10](#_Toc90054448)

[**Список литературы** 11](#_Toc90054449)

[**Приложение** 12](#_Toc90054450)

[Листинг 1. TMember.cpp 12](#_Toc90054451)

[Листинг 2. TMemberUnitTests.cpp 14](#_Toc90054452)

[Листинг 3. TPoly.cpp 20](#_Toc90054453)

# **Задание**

1. Реализовать тип «полином», в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования Visual Studio.
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

Спецификация абстрактного типа данных «Полином». ADT TPoly

**Данные**

Полиномы Tpoly - это неизменяемые полиномы с целыми коэффициентами.

Операции

Операции могут вызываться только объектом «полином» (тип TPoly), указатель на который передаётся в них по умолчанию. При описании операций этот объект в разделе «Вход» не указывается.

**Таблица 1**. Описание операций на ADT TPoly

|  |  |
| --- | --- |
| Наименование операции | Описание |
| ***Конструктор*** |  |
| Начальные  значения: | Коэффициент (с) и степень (n)  одночленного полинома |
| Процесс: | Создаёт одночленный полином с коэффициентом (с) и степенью (n), или ноль-полином, если коэффициент (с) равен 0 и возвращает указатель на него.  Например: Конструктор(6,3) = 6x^3 Конструктор(3,0) = 3  Конструктор() = 0 |
| Выход: | Нет. |
| Постусловия: | Объект инициализирован. |
|  | |
| ***Степень*** |  |

|  |  |
| --- | --- |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Отыскивает степень n полинома, т.е.  наибольшую степень при ненулевом коэффициенте (c). Степень нулевого полинома равна 0.  Например:  a = (x^2+1), a.Степень = 2 a = (17), a. Степень = 0 |
| Выход: | n - целое число - степень полинома. |
| Постусловия: | Нет. |
|  | |
| ***Коэффициент*** |  |
| Вход: | n - целое число - степень полинома. |
| Предусловия: | Полином – не нулевой. |
| Процесс: | Отыскивает коэффициент (c) при члене  полинома со степенью n (c\*x^n). Возвращает коэффициент (c) найденного члена или 0, если n больше степени полинома.  Например:  p = (x^3+2x+1), p.Coeff (4) = 0 p = (x^3+2x+1), p.Coeff (1) = 2 |
| Выход: | Целое число. |
| Постусловия: | Нет. |
|  | |
| ***Очистить (Clear)*** |  |
| Вход: | Нет. |
| Предусловия: | Нет |
| Процесс: | Удаляет члены полинома. |
| Выход: | Нет. |
| Постусловия: | this – нуль-полином. |
|  | |
| ***Сложить*** |  |
| Вход: | q - полином. |
| Предусловия: | Нет |
| Процесс: | Создаёт полином, являющийся  результатом сложения полинома с полиномом q и возвращает его. |
| Выход: | Полином. |
| Постусловия: | Нет. |

|  |  |
| --- | --- |
|  | |
| ***Умножить*** |  |
| Вход: | q - полином. |
| Предусловия: | Нет. |
| Процесс: | Создаёт полином, являющийся  результатом умножения полинома на полином q и возвращает его. |
| Выход: | Полином. |
| Постусловия: | Нет. |
|  | |
| ***Вычесть*** |  |
| Вход: | q - полином. |
| Предусловия: | Нет. |
| Процесс: | Создаёт полином, являющийся  результатом вычитания из полинома полинома q, и возвращает его. |
| Выход: | Полином. |
| Постусловия: | Нет. |
|  | |
| ***Минус*** |  |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт полином, являющийся  разностью ноль-полинома, и полинома и возвращает его. |
| Выход: | Полином. |
| Постусловия: | Нет. |
|  | |
| ***Равно*** |  |
| Вход: | q - полином. |
| Предусловия: | Нет. |
| Процесс: | Сравнивает полином с полиномом q на равенство. Возвращает значение True, если полиномы равны, т.е. имеют одинаковые коэффициенты при соответствующих членах, и значение  False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
|  | |
| ***Дифференцировать*** |  |

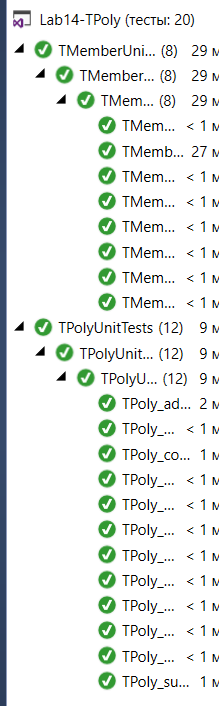
|  |  |
| --- | --- |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт полином, являющийся  производной полинома и возвращает его.  Например:  a = (x3+7x+5),  a.Дифференцировать = 3x2+7 |
| Выход: | Полином. |
| Постусловия: | Нет. |
|  | |
| ***Вычислить*** |  |
| Вход: | x – действительное число. |
| Предусловия: | Нет. |
| Процесс: | Вычисляет значение полинома в точке x  и возвращает его. Например:  a = (x2+3x), а.Вычислить(2) = 10 |
| Выход: | Действительное число. |
| Постусловия: | Нет. |
|  | |
| ***Элемент*** |  |
| Вход: | i - целое число - номер члена полинома. |
| Предусловия: | Нет. |
| Процесс: | Обеспечивает доступ к члену полинома  с индексом i для чтения его коэффициента (c) и степени (n) так, что если изменять i от 0 до количества членов в полиноме минус один, то  можно просмотреть все члены полинома. |
| Выход: | Коэффициент – целое число, степень –  целое число. |
| Постусловия: | Полином не модифицируется. |

end Tpoly

# **Реализация**

В данной работе мы создали абстрактный тип данных полином и все необходимые для работы с ним операции, такие как: конструктор, читать степень, писать степень, писать коэффициент, проверка равенства, функция взятия производной, вычисление в точке x, вывод одночлена в строку.

# **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. TMember.cpp

#include "TMember.h"

TMember::TMember()

{

FCoeff = 0;

FDegree = 0;

}

TMember::TMember(int c, int n)

{

FCoeff = c;

FDegree = n;

}

TMember::TMember(const TMember& p)

{

FCoeff = p.FCoeff;

FDegree = p.FDegree;

}

int TMember::get\_coeff(void) const

{

return FCoeff;

}

void TMember::set\_coeff(const int& n)

{

FCoeff = n;

}

int TMember::get\_degree(void) const

{

return FDegree;

}

void TMember::set\_degree(const int& c)

{

FDegree = c;

}

TMember TMember::diff(void)

{

return TMember(FCoeff \* FDegree, FDegree - 1);

}

double TMember::count\_poly(double x)

{

return FCoeff \* pow(x, FDegree);

}

string TMember::get\_tmember\_as\_string(void) const

{

return string(FCoeff == 0 ? "0" : to\_string(FCoeff) + (FDegree != 0 ? "\*x^" + to\_string(FDegree) : ""));

}

bool TMember::operator==(const TMember& q) const

{

return (FCoeff == q.FCoeff && FDegree == q.FDegree);

}

TMember TMember::operator-() const

{

return TMember(-FCoeff, FDegree);

}

TMember TMember::operator+(const TMember& q) const

{

if (FDegree != q.FDegree)

{

throw - 1;

}

return TMember(FCoeff + q.FCoeff, FDegree);

}

TMember TMember::operator\*(const TMember& q) const

{

return TMember(FCoeff \* q.FCoeff, FDegree + q.FDegree);

}

## Листинг 2. TMemberUnitTests.cpp

#include "pch.h"

#include "../TPoly.h"

#include "../TPoly.cpp"

#include "../TMember.h"

#include "../TMember.cpp"

#include "CppUnitTest.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TMemberUnitTests

{

TEST\_CLASS(TMemberUnitTests)

{

public:

TEST\_METHOD(TMember\_constructor\_tests)

{

TMember actual(0, 2);

string expected = "0";

Assert::IsTrue(expected == actual.get\_tmember\_as\_string());

actual = TMember(-3, 2);

expected = "-3\*x^2";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = TMember(-3, -2);

expected = "-3\*x^-2";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = TMember(3, -2);

expected = "3\*x^-2";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = TMember();

expected = "0";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

}

TEST\_METHOD(TMember\_get\_coeff\_tests)

{

TMember actual(0, 2);

int expected = 0;

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember(-3, 2);

expected = -3;

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember(-3, -2);

expected = -3;

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember(3, -2);

expected = 3;

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember();

expected = 0;

Assert::AreEqual(expected, actual.get\_coeff());

}

TEST\_METHOD(TMember\_get\_degree\_tests)

{

TMember actual(0, 2);

int expected = 2;

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember(-3, 2);

expected = 2;

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember(-3, -2);

expected = -2;

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember(3, -2);

expected = -2;

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember();

expected = 0;

Assert::AreEqual(expected, actual.get\_degree());

}

TEST\_METHOD(TMember\_set\_coeff\_tests)

{

TMember actual(0, 2);

int expected = 2;

actual.set\_coeff(expected);

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember(-3, 2);

expected = -2;

actual.set\_coeff(expected);

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember(-3, -2);

expected = -3;

actual.set\_coeff(expected);

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember(3, -2);

expected = 0;

actual.set\_coeff(expected);

Assert::AreEqual(expected, actual.get\_coeff());

actual = TMember();

expected = -5;

actual.set\_coeff(expected);

Assert::AreEqual(expected, actual.get\_coeff());

}

TEST\_METHOD(TMember\_set\_degree\_tests)

{

TMember actual(0, 2);

int expected = 5;

actual.set\_degree(expected);

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember(-3, 2);

expected = -2;

actual.set\_degree(expected);

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember(-3, -2);

expected = 2;

actual.set\_degree(expected);

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember(3, -2);

expected = 0;

actual.set\_degree(expected);

Assert::AreEqual(expected, actual.get\_degree());

actual = TMember();

expected = -5;

actual.set\_degree(expected);

Assert::AreEqual(expected, actual.get\_degree());

}

TEST\_METHOD(TMember\_equals\_tests)

{

TMember m(5, 3);

TMember m2(5, 3);

TMember m3(1, 2);

bool expected = true;

bool actual = (m == m2);

Assert::AreEqual(expected, actual);

expected = false;

actual = (m == m3);

Assert::AreEqual(expected, actual);

}

TEST\_METHOD(TMember\_diff\_tests)

{

TMember m(0, 2);

TMember m1(-3, 2);

TMember m2(-3, -2);

TMember m3(3, -2);

TMember m4(2, 0);

TMember actual = m.diff();

string expected = "0";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = m1.diff();

expected = "-6\*x^1";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = m2.diff();

expected = "6\*x^-3";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = m3.diff();

expected = "-6\*x^-3";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

actual = m4.diff();

expected = "0";

Assert::AreEqual(expected, actual.get\_tmember\_as\_string());

}

TEST\_METHOD(TMember\_count\_tests)

{

TMember actual(5, 5);

double expected = 5;

Assert::AreEqual(expected, actual.count\_poly(1));

expected = 160;

Assert::AreEqual(expected, actual.count\_poly(2));

expected = 0;

Assert::AreEqual(expected, actual.count\_poly(0));

expected = -5;

Assert::AreEqual(expected, actual.count\_poly(-1));

expected = -15625;

Assert::AreEqual(expected, actual.count\_poly(-5));

}

};

}

## Листинг 3. TPoly.cpp

#include "TPoly.h"

TPoly::TPoly()

{}

TPoly::TPoly(int c, int n)//конструктор

{

tpoly.emplace(n, TMember(c, n));

}

TPoly::TPoly(const TPoly& q)//конструктор

{

tpoly = q.tpoly;

}

int TPoly::get\_max\_degree()//взятие степени

{

return tpoly.rbegin()->first;

}

int TPoly::get\_coeff(int n)//взятие коэффицента

{

if (tpoly.count(n))

{

return tpoly[n].get\_coeff();

}

return 0;

}

void TPoly::clear()//очистка

{

tpoly.clear();

}

TPoly TPoly::operator+(const TPoly& q)//сложение

{

TPoly result = \*this;

for (auto& elem : q.tpoly)

{

if (result.tpoly.count(elem.first))

{

result.tpoly[elem.first] = result.tpoly[elem.first] + elem.second;

}

else

{

result.tpoly.emplace(elem.first, elem.second);

}

}

return result;

}

TPoly TPoly::operator\*(const TPoly& q)//умножение

{

TPoly result(0, 0);

for (auto& it1 : tpoly)

{

for (auto& it2 : q.tpoly)

{

TMember m = it1.second \* it2.second;

if (result.tpoly.count(m.get\_degree()))

{

result.tpoly[m.get\_degree()] = result[m.get\_degree()] + m;

}

else

{

result.tpoly[m.get\_degree()] = m;

}

}

}

return result;

}

TPoly TPoly::operator-(const TPoly& q)//вычитание полинома

{

return (\*this) + (-q);

}

TPoly TPoly::operator-() const//смена знака

{

TPoly result;

for (auto& elem : tpoly)

{

result.tpoly[elem.first] = -elem.second;

}

return result;

}

bool TPoly::operator==(const TPoly& q)//проверка равенстива

{

return tpoly == q.tpoly;

}

TPoly TPoly::diff(void)//производная

{

TPoly result;

for (auto& it : tpoly)

{

result.tpoly.emplace(it.first == 0 ? 0 : it.first - 1, it.second.diff());

}

return result;

}

double TPoly::count\_poly(double x)//вычисление

{

double sum = 0.0;

for (auto& it : tpoly)

{

sum += it.second.count\_poly(x);

}

return sum;

}

string TPoly::get\_poly\_as\_string()//взять полином как строку

{

string result;

for (auto elem = tpoly.rbegin(); elem != tpoly.rend(); elem++)

{

result += (elem->second.get\_tmember\_as\_string() != "0" ? elem->second.get\_tmember\_as\_string() + "+" : "");

}

if (result.size() != 0)

{

if (result[result.size() - 1] == '+')

{

result.erase(result.size() - 1, 1);

}

}

int index = result.find("+-");

while (index > 0)

{

result.replace(index, 2, "-");

index = result.find("+-");

}

return result;

}

TMember TPoly::operator[](int i) //доступ к i элементу

{

if (i < 0 || i >(int)tpoly.size())

{

return TMember();

}

auto it = tpoly.begin();

for (int in = 0; in < i; in++)

{

++it;

}

return (\*it).second;

}