Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

**Отчёт по лабораторной работе**

**«Арифметические операции с полиномами»**

**Выполнил:** студент группы 381603-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Капустин Д.О.

**Проверил:** к.т. н., старший преподаватель кафедры МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кустикова В.Д.

Нижний Новгород

2018 г.

**Содержание**

[Введение 3](#_Toc514109797)

[1 Постановка задачи 4](#_Toc514109798)

[2 Руководство пользователя 5](#_Toc514109799)

[3 Руководство программиста 6](#_Toc514109800)

[3.1 Структура программы 6](#_Toc514109801)

[3.2 Структуры данных 6](#_Toc514109802)

[3.3 Алгоритмы 7](#_Toc514109803)

[3.4 Программная реализация 10](#_Toc514109804)

Заключение [1](#_Toc514109806)3

[Литература 1](#_Toc514109807)4

[Приложения 1](#_Toc514109808)5

[Приложение 1. Pol\_Link.h 1](#_Toc514109809)5

Приложение 2. Pol\_List.h. [1](#_Toc514109810)5

[Приложение 3. Pol\_Mon.h 2](#_Toc514109811)3

[Приложение 4. Pol.h 23](#_Toc514109812)

[Приложение 5. Pol\_Mon.cpp 27](#_Toc514109813)

[Приложение 6. Pol.cpp 2](#_Toc514109814)8

[Приложение 7. Polinoms.cpp 36](#_Toc514109815)

[Приложение 8. Test\_pol.cpp 44](#_Toc514109816)

# Введение

Целью лабораторной работы является создание программы, способной выполнять арифметические операции (сложение, вычитание, умножение) с полиномами трех переменных со степенями от 0 до 9. Программа реализуется на языке С++. В качестве структуры данных используется циклический список с головой.

Разработанная программа может быть использована как самостоятельный продукт непосредственно для нахождения результата выполнения заданных пользователем арифметических операций с полиномами, так и в качестве составной части при решении более сложных задач.

# Постановка задачи

Разработать программу, которая позволяет пользователю задать полиномы, состоящие из мономов трех переменных x, y, z со степенями от 0 до 9 с вещественными коэффициентами, выполнить над ними заданные арифметические операции (сложения, вычитания, умножения двух полиномов, умножение полинома на константу) и вывести полученный результат вычислений.

# Руководство пользователя

Для начала работы необходимо открыть файл Polinoms.exe.

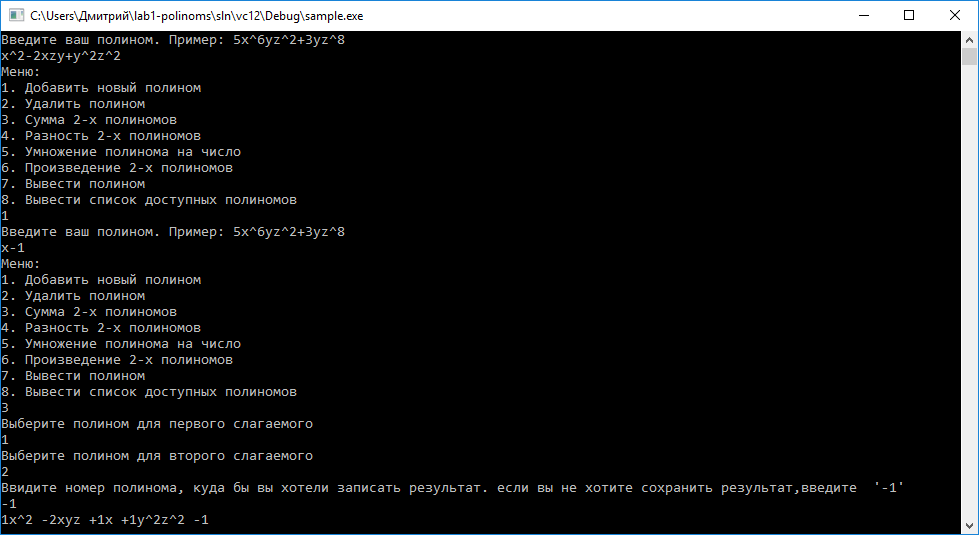
Затем на запрос программы ввести первый полином в соответствии с предложенным примером (допустимые символы: x, y, z,^,0,1,2,3,4,5,6,7,8,9,+.-).

После чего выбрать желаемую операцию из предложенного меню:

1. Добавить новый полином
2. Удалить полином
3. Сумма 2-х полиномов
4. Разность 2-х полиномов
5. Умножение полинома на число
6. Произведение 2-х полиномов
7. Вывести полином
8. Вывести список доступных полиномов

В случае выбора пунктов 3,4,6 меню осуществить ввод второго полинома. При выборе пункта 5 меню ввести число.

Далее программа производит вывод результата выполнения арифметической операции и (по желанию пользователя) сохраняет или не сохраняет его, продолжает либо завершает свою работу.



1. Пример

# Руководство программиста

## Структура программы

Pol\_Link.h

Pol\_List.h

Pol\_Mon.h

Pol.h

Pol\_Mon.cpp

Pol.cpp

Polinoms.cpp

Test\_pol.cpp

Программа состоит из четырех проектов.

*gtest*

Содержит в себе gtest.h и gtest-all.cpp. Файлы содержат googleтесты, которые необходимы для проверки корректности реализованных классов.

*polinoms*

Содержит в себе:

1. Pol\_Link.h– реализация шаблонного класса звена.
2. Pol\_Mon.h– объявление полей и методов, которые отвечают за степень и коэффициент звена(монома)
3. Pol.h – объявление полей и методов класса полином
4. Pol\_List.h– реализация шаблонного класса список
5. Pol\_Mon.cpp– реализация полей и методов класса Monom
6. Pol.cpp – реализация полей и методов класса полином

*tests*

Содержит в себе test\_main.cpp, test\_pol.cpp. Реализация тестов для проверки корректности реализованных классов.

*sample*

Содержит Polinoms.cpp– реализация меню консольного приложения.

## Структуры данных

В программе реализована структура Monom (моном), класс TLink(звено), класс, TRingList (список), класс TPolinom (полином). Полином состоит из зацикленного списка с головой. Список состоит из звеньев. Полями списка является звено – голова и звено – текущий элемент в списке. ­Звенья хранят в себе указатель на звено и моном. Моном состоит из числа двойной точности (double) – коэффициент и беззнакового целого числа(unsigned int) – степень монома.

## Алгоритмы

Описание алгоритмов:

Сложение полиномов. На вход поступает второй полином. Создается результирующий пустой полином. Далее алгоритм сложения аналогичен алгоритму слияния двух отсортированных списков: Оба полинома просматриваются одновременно в цикле, пока один из них не кончился. Если степень текущего монома первого меньше, чем степень текущего монома второго, в результат вставляется текущий первого, иначе второго. В случае, если мономы одинаковые, а сумма их коэффициентов не равна нулю, вставляется моном с коэффициентом, равным сумме коэффициентов первого и второго монома. После выхода из цикла оставшиеся мономы вставляются в конец. Возвращается результат сложения

Умножение полиномов. На вход поступает второй полином. Создается результирующий пустой полином. В цикле с первого по последний мономы просматривается первый полином. В этом цикле вложен второй такой же цикл по второму полиному. В результат вставляются мономы, являющиеся результатом произведения каждого монома из первого полинома с каждым мономом из второго полинома, если только коэффициент результирующего монома не равен нулю. Возвращается результирующий полином.

## Программная реализация

### Структура Monom

Поля:

double cf–коэффициент монома

unsigned int stmonom – степень монома

Методы:

Monom(const Monom&\_Tmonom) –конструктор копирования

Monom(const double d=0.0; const int m=0) –конструктор с параметрами

~Monom() – деструктор

Monom operator + (constTMonom&monom) –сумма мономов

Monom operator = (constTMonom&monom) –оператор копирования

bool operator == (constTMonom&monom) const –сравнение мономов

bool operator != (constTMonom&monom) const –сравнение мономов

bool operator > (constTMonom&monom) const –сравнение мономов

bool operator < (constTMonom&monom) const –сравнение мономов

### Класс TLink

Поля:

Tdata –данные

TLink<T>\* pNext – указатель на следующее звено

Методы:

TLink() – конструктор по умолчанию

TLink(const T datum, TLink<T>\* link = nullptr) – конструктор по параметрам

~TLink() –деструктор

TLink<T>& operator = (const TLink<T>& \_TLink) –оператор копирования

bool operator == (const TLink<T>& link) const –оператор сравнения

### Класс TRingList

Поля:

TLink<T> \*head – голова списка

TLink<T> \*current – текущий элемент списка

Методы:

TRingList() –конструктор по умолчанию

TRingList(constTRingList<T>&) –конструктор копирования

~TRingList() –деструктор

TRingList<T>& operator = (constTRingList<T>& TRingList2) –оператор копирования

void Clean() –очистка списка

void Insert(const T \_data) – упорядоченная вставка в список

void InsertInEnd(const T datum) – вставка в конец

void Reset() – текущий элемент указывает следующее звено после головы

void Delete(const T & d) – удаление данных

TLink<T>Search(const TLink<T>& d) – поиск в списке данных

void GetNext() – двигаем указатель текущего элемента на следующий

bool IsEnded() const – проверка на конце списка

TLink<T>\* GetCurrent() – вернуть текущее звено

bool operator == (constTRingList<T>& list2) –оператор сравнения

bool operator > (constTRingList<T>& list2) –оператор сравнения

bool operator < (constTRingList<T>& list2) –оператор сравнения

bool operator != (constTRingList<T>& list2) –оператор сравнения

### Класс TPolinom

Поля:

TRingList<Monom>curmon – список

Методы:

TPolinom() – конструктор по умолчанию

TPolinom(const string &s) – конструктор по строке

TPolinom(const TPolinom&) –конструткор копирования

~TPolinom() –деструктор

TPolinom operator + (constTPolinom& \_pol) –оператор суммы

TPolinom& operator += (constTPolinom& \_pol) –оператор суммы

TPolinom operator - (constTPolinom& \_pol) –оператор разности

TPolinom operator \* (constTPolinom& \_pol) – оператор произведения

TPolinomoperator \* (double d) – оператор умножения на константу

friendTPolinomoperator \* (doublecf2, constTPolinom&poli2) – оператор произведения константы и полинома

TPolinom& operator = (constTpolinom&\_pol) –оператор копирвоания

bool operator == (constTPolinom& curpol) –оператор сравнения

bool operator != (constTPolinom& curpol) –оператор сравнения

bool operator > (constTPolinom& curpol) –оператор сравнения

bool operator < (constTPolinom& curpol) –оператор сравнения

voidPrivPod() – функция приведения подобных звеньев

friendostream& operator <<(ostream&out, constTPolinom& \_pol) –перегрузка вывода полинома

# Заключение

Представленная программа позволяет пользователю задать полиномы, состоящие из мономов трех переменных x, y, z со степенями от 0 до 9 с вещественными коэффициентами, выполнить над ними заданные арифметические операции (сложения, вычитания, умножения двух полиномов, умножение полинома на константу) и вывести полученный результат вычислений. Таким образом, цель лабораторной работы достигнута.

# Литература

1. *Гергель В.П.* Рабочие материалы студента по общему курсу «Методы программирования» - Н.Новгород, 2015.

# Приложения

**Приложение 1. Pol\_Link.h**

#pragma once;

template <typename T>

class TLink {

public:

T data;

TLink<T>\* pNext;

TLink() {};

~TLink() {};

TLink(const T datum, TLink<T>\* link = nullptr) { data = datum; pNext = link; }

TLink<T>& operator=(const TLink<T>& \_TLink)

{

data = \_TLink.data;

pNext = \_TLink.pNext;

return \*this;

};

bool operator == (const TLink<T>& link) const

{

return \*this->pNext == link.pNext;

};

## };

## Приложение 2. Pol\_List.h

#pragma once;

#include "Pol\_Link.h"

template <typename T>

class TRingList {

private:

TLink <T> \*head;

TLink <T> \*current;

public:

TRingList();

TRingList(const TRingList <T> &);

~TRingList();

void Insert(const T \_data);

void InsertInEnd(const T datum);

void Reset();

void Clean();

bool IsEnded() const;

TRingList<T>& operator=(const TRingList<T>& \_TRingList);

bool operator==(const TRingList<T>& list2) const;

bool operator<(const TRingList<T>& list2) const;

bool operator>(const TRingList<T>& list2) const;

bool operator!=(const TRingList<T>& list2) const { return !((\*this) == list2); }

TLink<T>\* GetNext();

void GetNext() { current = current->pNext; };

//TLink<T>\* Search(const TLink<T>& d);

void Delete(const T & d);

TLink<T>\* GetCurrent() { return current; }

};

template <typename T>

TRingList<T>::TRingList()

{

head = new TLink<T>();

head->pNext = head;

current = head;

};

template <typename T>

TLink<T>\* TRingList<T>::Search(const TLink<T> & d)

{

TLink<T>\* tmp = head;

while ((tmp->pNext != head) && (tmp->pNext->data != d.data))

{

tmp = tmp->pNext;

}

if (tmp->pNext == head)

{

return nullptr;

}

return tmp->pNext;

};

template<typename T>

void TRingList<T>::Delete(const T & d)

{

TLink<T>\* tmp = head;

TLink<T>\* tmp2 = head->pNext;

if (tmp != tmp2)

{

while ((tmp2->pNext != head) && (tmp2->data != d))

{

tmp = tmp2;

tmp2 = tmp2->pNext;

}

if (tmp2->data == d)

{

tmp->pNext = tmp2->pNext;

delete tmp2;

}

else throw "element does not exist in your list";

}

else throw "element does not exist in your list";

};

template <typename T>

TRingList<T>::TRingList(const TRingList <T> & Ring)

{

if (Ring.head->pNext != Ring.head)

{

head = new TLink<T>(Ring.head->data);

TLink<T> \*temp = head;

TLink<T> \*temp2 = Ring.head->pNext;

while (temp2 != Ring.head)

{

temp->pNext = new TLink<T>(temp2->data, head);

temp = temp->pNext;

temp2 = temp2->pNext;

}

}

else {

head = new TLink<T>(Ring.head->data);

current = head;

head->pNext = head;

}

};

template<typename T>

TRingList<T>::~TRingList()

{

Clean();

delete head;

};

template <typename T>

void TRingList<T>::Insert(const T datum)

{

if (head->pNext == head)

{

TLink<T>\* tmp = new TLink<T>(datum, head);

head->pNext = tmp;

}

else

{

TLink<T>\* prev = head;

TLink<T>\* tmp = head->pNext;

while ((tmp != head) && (datum < tmp->data))

{

prev = tmp;

tmp = tmp->pNext;

}

prev->pNext = new TLink<T>(datum, tmp);

}

};

template <typename T>

void TRingList<T>::InsertInEnd(const T datum)

{

TLink<T>\* tmp = head;

while (tmp->pNext != head)

tmp = tmp->pNext;

tmp->pNext = new TLink<T>(datum, head);

};

template <typename T>

void TRingList<T>::Reset()

{

current = head->pNext;

};

template<typename T>

void TRingList<T>::Clean()

{

TLink<T> \*temp = head->pNext;

TLink<T> \*temp2 = head;

while (temp != head)

{

temp2 = temp->pNext;

delete temp;

temp = temp2;

}

head->pNext = head;

};

template<typename T>

bool TRingList<T>::IsEnded() const

{

return (current == head);

};

template<typename T>

TRingList<T>& TRingList<T>::operator=(const TRingList<T> & Ring)

{

if ((\*this) != Ring)

{

Clean();

if (Ring.head->pNext != Ring.head)

{

head = new TLink<T>(Ring.head->data);

TLink<T> \*temp = head;

TLink<T> \*temp2 = Ring.head->pNext;

while (temp2 != Ring.head)

{

temp->pNext = new TLink<T>(temp2->data, head);

temp = temp->pNext;

temp2 = temp2->pNext;

}

}

else {

head = new TLink<T>(Ring.head->data);

current = head;

head->pNext = head;

}

}

return \*this;

};

template<typename T>

bool TRingList<T>::operator==(const TRingList<T> & list2) const

{

TLink<T> \*temp = head;

TLink<T> \*temp2 = list2.head;

bool f = true;

if ((temp->pNext == head) && (temp2->pNext == list2.head))

f = true;

else if (((temp->pNext != head) && (temp2->pNext == list2.head)) || ((temp2->pNext != list2.head) && (temp->pNext == head)))

f = false;

while ((temp->pNext != head) && (temp2->pNext != list2.head) && (f))

{

if (temp->data != temp2->data)

f = false;

temp = temp->pNext;

temp2 = temp2->pNext;

}

return f;

};

template<typename T>

bool TRingList<T>::operator<(const TRingList<T> & list2) const

{

TLink<T> \*temp = head;

TLink<T> \*temp2 = list2.head;

bool f = true;

if ((temp->pNext == head) && (temp2->pNext == list2.head))

f = false;

else if ((temp->pNext != head) && (temp2->pNext == list2.head))

f = false;

else if ((temp2->pNext != list2.head) && (temp->pNext == head))

f = true;

while ((temp->pNext != head) && (temp2->pNext != list2.head) && (f))

{

if ((temp->data > temp2->data) || (temp->data == temp2->data))

f = false;

temp = temp->pNext;

temp2 = temp2->pNext;

}

return f;

};

template<typename T>

bool TRingList<T>::operator>(const TRingList<T> & list2) const

{

TLink<T> \*temp = head;

TLink<T> \*temp2 = list2.head;

bool f = true;

if ((temp->pNext == head) && (temp2->pNext == list2.head))

f = true;

else if ((temp->pNext != head) && (temp2->pNext == list2.head))

f = true;

else if ((temp2->pNext != list2.head) && (temp->pNext == head))

f = false;

while ((temp->pNext != head) && (temp2->pNext != list2.head) && (f))

{

if ((temp->data < temp2->data) || (temp->data == temp2->data))

f = false;

temp = temp->pNext;

temp2 = temp2->pNext;

}

return f;

};

template <typename T>

TLink<T>\* TRingList<T>::GetNext()

{

current = current->pNext;

return current;

};

===============================================================

## Приложение 3. Pol\_Mon.h

#pragma once

struct Monom {

double cf;

unsigned int stmonom;

Monom(const double d = 0.0, const int m = 0) { cf = d; stmonom = m; };

Monom(const Monom& \_Tmonom);

const Monom& operator = (const Monom& \_Tmonom);

bool operator < (const Monom& \_tmonom) const;

bool operator != (const Monom& \_tmonom) const;

bool operator == (const Monom& \_tmonom) const;

bool operator > (const Monom& \_tmonom) const;

Monom operator+ (const Monom& \_tmonom);

};

===============================================================

## Приложение 4. Pol.h

#include <iostream>

#include <cstdio>

#include <string>

#include "Pol\_List.h"

#include "Pol\_Mon.h"

using namespace std;

class TPolinom {

private:

TRingList<Monom> curmon;

public:

TPolinom() { }

TPolinom(const string &s);

TPolinom(const TPolinom &);

~TPolinom();

TPolinom operator+ (const TPolinom & \_pol);

TPolinom operator- (const TPolinom & \_pol)

{

TPolinom res = (\*this) + (-1.0) \* \_pol;

return res;

}

TPolinom& operator+= (const TPolinom & \_pol);

TPolinom operator\* (const TPolinom & \_pol);

TPolinom operator\* (double d);

TPolinom& operator = (const TPolinom& \_pol);

bool operator == (const TPolinom& curpol) const { return (\*this).curmon == curpol.curmon; };

bool operator != (const TPolinom& curpol) const { return !((\*this) == curpol); };

bool operator < (const TPolinom& curpol) const { return (\*this).curmon <curpol.curmon; };

bool operator > (const TPolinom& curpol) const { return (\*this).curmon > curpol.curmon; };

friend ostream& operator<< (ostream &out, const TPolinom& \_pol)

{

TPolinom copy\_of\_polinom(\_pol);

copy\_of\_polinom.curmon.Reset();

cout << copy\_of\_polinom.curmon.GetCurrent()->data.cf;

if ((copy\_of\_polinom.curmon.GetCurrent()->data.cf==1) && (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100 != 0) && (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10 != 0 && (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom) % 10 != 0)

out << copy\_of\_polinom.curmon.GetCurrent()->data.cf;

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100) != 0) {

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100) == 1)

out << "x";

else

out << "x^" << (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100);

}

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10 != 0) {

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10 == 1)

out << "y";

else out << "y^" << (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10;

}

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom) % 10 != 0) {

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom % 10) == 1)

out << "z";

else out << "z^" << (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom) % 10;

}

TLink<Monom>\* cur = copy\_of\_polinom.curmon.GetNext();

for (cur; !copy\_of\_polinom.curmon.IsEnded(); copy\_of\_polinom.curmon.GetNext())

{

if (copy\_of\_polinom.curmon.GetCurrent()->data.cf > 0)

{

out << " +" << copy\_of\_polinom.curmon.GetCurrent()->data.cf;

}

else out << " " << copy\_of\_polinom.curmon.GetCurrent()->data.cf;

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100) != 0) {

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100) == 1)

out << "x";

else

out << "x^" << (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 100);

}

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10 != 0)

{

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10 == 1)

out << "y";

else out << "y^" << (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10) % 10;

}

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom) % 10 != 0) {

if ((copy\_of\_polinom.curmon.GetCurrent()->data.stmonom % 10) == 1)

out << "z";

else out << "z^" << (copy\_of\_polinom.curmon.GetCurrent()->data.stmonom) % 10;

}

}

out << endl;

return out;

}

friend TPolinom operator\* (double d, const TPolinom &p)

{

TPolinom res(p);

res.curmon.Reset();

if (d == 0)

res.curmon.Clean();

else {

for (res.curmon.GetCurrent(); !res.curmon.IsEnded(); res.curmon.GetNext())

{

res.curmon.GetCurrent()->data.cf \*= d;

}

}

return res;

};

};

===============================================================

## 

## Приложение 5. Pol\_Mon.cpp

#include "Pol\_Mon.h"

Monom::Monom(const Monom& \_Tmonom)

{

cf = \_Tmonom.cf;

stmonom = \_Tmonom.stmonom;

}

const Monom& Monom::operator = (const Monom& \_Tmonom)

{

cf = \_Tmonom.cf;

stmonom = \_Tmonom.stmonom;

return \*this;

}

bool Monom::operator < (const Monom& \_tmonom) const

{

return stmonom < \_tmonom.stmonom;

}

bool Monom::operator != (const Monom& \_tmonom) const

{

return stmonom != \_tmonom.stmonom;

}

bool Monom::operator == (const Monom& \_tmonom) const

{

return stmonom == \_tmonom.stmonom;

}

bool Monom::operator > (const Monom& \_tmonom) const

{

return stmonom > \_tmonom.stmonom;

}

Monom Monom::operator+ (const Monom& \_tmonom)

{

Monom res;

if ((\*this).stmonom == \_tmonom.stmonom)

{

res.stmonom = (\*this).stmonom;

res.cf = (\*this).cf + \_tmonom.cf;

}

else throw "unable to sum monoms!";

return res;

}

===============================================================

## Приложение 6. Pol.cpp

#include "Pol.h"

TPolinom::TPolinom(const std::string& s)

{

string str = s;

if (str[0] != '\0')

{

while (str.length() > 0)

{

string monom = "";

Monom temp;

string coeff = "";

double number = 0;

int i = 0, j = 1;

j = 1;

while (j < str.length() && str[j] != '+' && str[j] != '-')

j++;

monom = str.substr(0, j);

str.erase(0, j);

j = 0;

while (j < monom.length() && !isalpha(monom[j]))

j++;

coeff = monom.substr(0, j);

if (coeff == "" || coeff == "+")

temp.cf = 1;

else if (coeff == "-")

temp.cf = -1;

else temp.cf = atof(coeff.c\_str());

monom.erase(0, j);

j = 0;

while (j < monom.length())

{

if (isalpha(monom[j]) && monom[j + 1] == '\0')

{

switch (monom[j])

{

case 'x':

temp.stmonom += 1 \* 100;

break;

case 'y':

temp.stmonom += 1 \* 10;

break;

case 'z':

temp.stmonom += 1;

break;

default:

throw "Input error:end of string";

break;

}

j++;

}

else if (isalpha(monom[j]) && isalpha(monom[j + 1]))

{

switch (monom[j])

{

case 'x':

temp.stmonom += 1 \* 100;

break;

case 'y':

temp.stmonom += 1 \* 10;

break;

case 'z':

temp.stmonom += 1;

break;

default:

throw "Input error:end of string";

break;

}

j++;

}

else if (isalpha(monom[j]) && monom[j + 1] == '^' && monom[j + 2] != '\0' && isdigit(monom[j + 2]))

{

switch (monom[j])

{

case 'x':

temp.stmonom += atoi(&monom[j + 2]) \* 100;

j += 3;

break;

case 'y':

temp.stmonom += atoi(&monom[j + 2]) \* 10;

j += 3;

break;

case 'z':

temp.stmonom += atoi(&monom[j + 2]) \* 1;

j += 3;

break;

}

}

}

if ((temp.stmonom != 0 || temp.stmonom == 0) && temp.cf != 0) {

TLink<Monom>\* link = curmon.Search(temp);

if (link != NULL)

{

if (link->data.cf + temp.cf != 0)

{

link->data.cf += temp.cf;

}

else curmon.Delete(link->data);

}

else

curmon.Insert(temp);

}

}

}

}

TPolinom::TPolinom(const TPolinom & \_polinom)

{

curmon = \_polinom.curmon;

}

TPolinom::~TPolinom()

{

}

TPolinom TPolinom::operator+ (const TPolinom & \_polinom)

{

(\*this).curmon.Reset();

TPolinom copy\_of\_polinom(\_polinom);

copy\_of\_polinom.curmon.Reset();

TPolinom res;

res.curmon.Reset();

while (!copy\_of\_polinom.curmon.IsEnded() && !(\*this).curmon.IsEnded())

{

if (copy\_of\_polinom.curmon.GetCurrent()->data < (\*this).curmon.GetCurrent()->data)

{

res.curmon.InsertInEnd((\*this).curmon.GetCurrent()->data);

(\*this).curmon.GetNext();

}

else if (copy\_of\_polinom.curmon.GetCurrent()->data == (\*this).curmon.GetCurrent()->data)

{

if ((\*this).curmon.GetCurrent()->data.cf + copy\_of\_polinom.curmon.GetCurrent()->data.cf != 0.0)

{

res.curmon.InsertInEnd((\*this).curmon.GetCurrent()->data + copy\_of\_polinom.curmon.GetCurrent()->data);

}

(\*this).curmon.GetNext();

copy\_of\_polinom.curmon.GetNext();

}

else

{

res.curmon.InsertInEnd(copy\_of\_polinom.curmon.GetCurrent()->data);

copy\_of\_polinom.curmon.GetNext();

}

}

while (!copy\_of\_polinom.curmon.IsEnded())

{

res.curmon.InsertInEnd(copy\_of\_polinom.curmon.GetCurrent()->data);

copy\_of\_polinom.curmon.GetNext();

}

while (!(\*this).curmon.IsEnded())

{

res.curmon.InsertInEnd((\*this).curmon.GetCurrent()->data);

(\*this).curmon.GetNext();

}

return res;

}

TPolinom & TPolinom::operator+=(const TPolinom & \_polinom)

{

TPolinom copy\_of\_polinom(\_polinom);

copy\_of\_polinom.curmon.Reset();

for (copy\_of\_polinom.curmon.GetCurrent(); !copy\_of\_polinom.curmon.IsEnded(); copy\_of\_polinom.curmon.GetNext())

{

TLink<Monom>\* tmp2 = (\*this).curmon.Search(copy\_of\_polinom.curmon.GetCurrent()->data);

if (tmp2 != NULL)

{

if (tmp2->data.cf + copy\_of\_polinom.curmon.GetCurrent()->data.cf != 0)

{

tmp2->data.cf += copy\_of\_polinom.curmon.GetCurrent()->data.cf;

}

else

(\*this).curmon.Delete(tmp2->data);

}

else

(\*this).curmon.Insert(copy\_of\_polinom.curmon.GetCurrent()->data);

}

return \*this;

}

TPolinom TPolinom::operator\* (const TPolinom & \_polinom)

{

(\*this).curmon.Reset();

TPolinom copy\_of\_polinom(\_polinom);

copy\_of\_polinom.curmon.Reset();

TPolinom res;

res.curmon.Reset();

for ((\*this).curmon.GetCurrent(); !(\*this).curmon.IsEnded(); (\*this).curmon.GetNext())

{

for (copy\_of\_polinom.curmon.GetCurrent(); !copy\_of\_polinom.curmon.IsEnded(); copy\_of\_polinom.curmon.GetNext())

{

if (((\*this).curmon.GetCurrent()->data.stmonom + copy\_of\_polinom.curmon.GetCurrent()->data.stmonom < 1000) &&

(((\*this).curmon.GetCurrent()->data.stmonom / 10 % 10 + copy\_of\_polinom.curmon.GetCurrent()->data.stmonom / 10 % 10) < 10) &&

(((\*this).curmon.GetCurrent()->data.stmonom % 10 + copy\_of\_polinom.curmon.GetCurrent()->data.stmonom % 10) < 10))

{

unsigned int tmpabc = (\*this).curmon.GetCurrent()->data.stmonom + copy\_of\_polinom.curmon.GetCurrent()->data.stmonom;

Monom tmp((\*this).curmon.GetCurrent()->data.cf \* copy\_of\_polinom.curmon.GetCurrent()->data.cf, tmpabc);

TLink<Monom>\* tmp2 = res.curmon.Search(tmp);

if (tmp2 != NULL)

if ((tmp2->data.cf += (\*this).curmon.GetCurrent()->data.cf \* copy\_of\_polinom.curmon.GetCurrent()->data.cf) == 0)

{

res.curmon.Delete(tmp2->data);

}

else tmp2->data.cf += (\*this).curmon.GetCurrent()->data.cf \* copy\_of\_polinom.curmon.GetCurrent()->data.cf;

else

res.curmon.Insert(tmp);

}

else throw "Power is more then expected";

}

copy\_of\_polinom.curmon.Reset();

}

return res;

}

TPolinom TPolinom::operator\* (double d)

{

TPolinom res = (\*this);

res.curmon.Reset();

if (d == 0)

res.curmon.Clean();

else {

while (!res.curmon.IsEnded())

{

res.curmon.GetCurrent()->data.cf \*= d;

res.curmon.GetNext();

}

}

return res;

}

TPolinom& TPolinom::operator =(const TPolinom& \_polinom)

{

curmon = \_polinom.curmon;

return \*this;

}

===============================================================

## Приложение 7. Polinoms.cpp

#include <cstdio>

#include <string>

#include <iostream>

#include "Pol.h"

#include "Pol\_Mon.h"

int main() {

setlocale(LC\_ALL, "Russian");

TPolinom mass[10];

TPolinom res;

std::string s;

std::cout << "Введите ваш полином. Пример: 5х^6yz^2+3yz^8" << std::endl;

TPolinom zero("");

int menu = 1;

int i = 1;

std::cin >> s;

TPolinom First(s);

mass[1]= First;

i++;

int a, b, c;

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

while (menu < 9 && menu > 0)

{

switch (menu)

{

case 1:

{

std::cout << "Введите ваш полином. Пример: 5х^6yz^2+3yz^8" << std::endl;

std::cin >> s;

TPolinom temp1(s);

mass[i] = temp1;

i++;

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

}

case 2:

do {

std::cout << "Введите номер полинома, который вы хотите удалить " << std::endl;

std::cin >> a;

string s = "0xyz";

TPolinom Null(s);

mass[a] = Null;

} while (a < 0 || a > 10);

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

case 3:

do {

std::cout << "Выберите полином для первого слагаемого" << std::endl;

std::cin >> a;

} while (a < 0 || a > 10);

do {

std::cout << "Выберите полином для второго слагаемого " << std::endl;

std::cin >> b;

} while (b < 0 || b > 10);

do {

std::cout << "Ввидите номер полинома, куда бы вы хотели записать результат. если вы не хотите сохранить результат,введите '-1' " << std::endl;

std::cin >> c;

} while (c < -1 || c > 10);

if (c == -1)

{

TPolinom res;

res = mass[a] + mass[b];

std::cout << res;

}

else if (c > -1)

mass[c] = mass[a] + mass[b];

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

case 4:

do {

std::cout << "Выберите полином, который является уменьшаемым " << std::endl;

std::cin >> a;

} while (a < 0 || a > 10);

do {

std::cout << "Выберите полином, который является вычитаемым " << std::endl;

std::cin >> b;

} while (b < 0 || b > 10);

do {

std::cout << "Ввидите номер полинома, куда бы вы хотели записать результат. если вы не хотите сохранить результат,введите '-1' " << std::endl;

std::cin >> c;

} while (c < -1 || c >10);

if (c == -1)

{

TPolinom res;

res = mass[a] - mass[b];

std::cout << res;

}

else if (c > -1)

mass[c] = mass[a] - mass[b];

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

case 5:

double d;

do {

std::cout << "Выберите полином для первого множителя" << std::endl;

std::cin >> a;

} while (a < 0 || a > 10);

std::cout << "Введите число, на которые Вы хотите умножить полином" << std::endl;

std::cin >> d;

do {

std::cout << "Ввидите номер полинома, куда бы вы хотели записать результат. если вы не хотите сохранить результат,введите '-1' " << std::endl;

std::cin >> c;

} while (c < -1 || c > 10);

if (c == -1)

{

TPolinom res;

res = mass[a] \* d;

std::cout << res;

}

else if (c > -1)

mass[c] = mass[a] \* d;

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

case 6:

do {

std::cout << "Выберите полином для первого множителя" << std::endl;

std::cin >> a;

} while (a < 0 || a > 10);

do {

std::cout << "Выберите полином для второго множителя" << std::endl;

std::cin >> b;

} while (b < 0 || b > 10);

do {

std::cout << "Ввидите номер полинома, куда бы вы хотели записать результат. если вы не хотите сохранить результат,введите '-1' " << std::endl;

std::cin >> c;

} while (c < -1 || c > 10);

if (c == -1)

{

TPolinom res;

res = mass[a] \* mass[b];

std::cout << res;

}

else if (c > -1)

mass[c] = mass[a] \* mass[b];

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

case 7:

do {

std::cout << "Выберите полином, который вы хотите вывести " << std::endl;

std::cin >> a;

cout << mass[a];

} while (a < 0 || a >= 10);

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Умножение полинома на число" << endl;

std::cout << "6. Произведение 2-х полиномов" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

case 8:

do {

std::cout << "Список доступных полиномов " << std::endl;

string s = "0xyz";

TPolinom Null(s);

for (int j = 1; j < 10; j++)

if (mass[j]!=Null)

cout << j <<' '<< mass[j] << '\n';

break;

} while (a < 0 || a >= 10);

std::cout << "Меню:" << endl;

std::cout << "1. Добавить новый полином" << endl;

std::cout << "2. Удалить полином" << endl;

std::cout << "3. Сумма 2-х полиномов" << endl;

std::cout << "4. Разность 2-х полиномов" << endl;

std::cout << "5. Произведение 2-х полиномов" << endl;

std::cout << "6. Умножение полинома на число" << endl;

std::cout << "7. Вывести полином" << endl;

std::cout << "8. Вывести список доступных полиномов" << endl;

std::cin >> menu;

break;

default:exit(1);

}

}

return 0;

}

===============================================================

## Приложение 8. Test\_pol.cpp

#include "Pol\_List.h"

#include "Pol.h"

#include "POl\_Mon.h"

#include "Pol\_Link.h"

#include <gtest.h>

TEST(TPolinom, Create\_Pol)

{

ASSERT\_NO\_THROW(TPolinom a(""));

}

TEST(TPolinom, Two\_Empty\_Pol\_Equal)

{

TPolinom b("");

TPolinom a("");

EXPECT\_EQ(a, b);

}

TEST(TPolinom, Build\_Pol)

{

ASSERT\_NO\_THROW(TPolinom a("5x^3"));

}

TEST(TPolinom, Create\_Pol\_After\_Existing)

{

TPolinom a("5x^3");

ASSERT\_NO\_THROW(TPolinom b(a));

}

TEST(TPolinom, Create\_Pol\_With\_EQ\_Mon)

{

TPolinom a("2x^5+3x^5");

TPolinom b("5x^5");

ASSERT\_NO\_THROW(a, b);

}

TEST(TPolinom, Summa\_1)

{

TPolinom a("10x^2");

TPolinom b("12y^2");

ASSERT\_NO\_THROW(a + b);

}

TEST(TPolinom, Summa\_2)

{

TPolinom a("x^3+2y^2-z");

TPolinom b("y^3-3xy+2xy^3");

ASSERT\_NO\_THROW(a + b);

}

TEST(TPolinom, Summa\_is\_Correct\_1)

{

TPolinom a("5x^2");

TPolinom b("6x^7");

TPolinom c("6x^7+5x^2");

EXPECT\_EQ(c, a + b);

}

TEST(TPolinom, Summa\_is\_Correct\_2)

{

TPolinom a("x+y+z");

TPolinom b("x-2y+z^2");

TPolinom c("2x-y+z^2+z");

EXPECT\_EQ(c, a + b);

}

TEST(TPolinom, Substract\_1)

{

TPolinom a("x+1");

TPolinom b("x-1");

ASSERT\_NO\_THROW(a + b);

}

TEST(TPolinom, Substract\_2)

{

TPolinom a("x^2+5");

TPolinom b("x^2-4");

ASSERT\_NO\_THROW(a + b);

}

TEST(TPolinom, Substract\_is\_Correct\_1)

{

TPolinom a("x+1");

TPolinom b("x-1");

TPolinom c("-2");

EXPECT\_EQ(c, b - a);

}

TEST(TPolinom, Substract\_is\_Correct\_2)

{

TPolinom a("x+2y");

TPolinom b("x-z");

TPolinom c("2y+z");

EXPECT\_EQ(c, a - b);

}

TEST(TPolinom, Mult\_Two\_Pol\_1)

{

TPolinom a("x^3");

TPolinom b("y^4");

ASSERT\_NO\_THROW(a \* b);

}

TEST(TPolinom, Mult\_Two\_Pol\_2)

{

TPolinom a("10x^2");

TPolinom b("12y^2");

ASSERT\_NO\_THROW(a \* b);

}

TEST(TPolinom, Mult\_Two\_Pol\_is\_Correct\_1)

{

TPolinom a("2x^2");

TPolinom b("2y^2");

TPolinom c("4x^2y^2");

EXPECT\_EQ(c, a \* b);

}

TEST(TPolinom, Mult\_Two\_Pol\_is\_Correct\_2)

{

TPolinom a("x+1");

TPolinom b("x-1");

TPolinom c("x^2-1");

EXPECT\_EQ(c, a \* b);

}

TEST(TPolinom, ERROR\_Mult\_St\_Over\_10\_1)

{

TPolinom a("3x^4");

TPolinom b("x^8");

ASSERT\_ANY\_THROW(a \* b);

}

TEST(TPolinom, ERROR\_Mult\_St\_Over\_10\_2)

{

TPolinom a("y^2");

TPolinom b("y^8");

ASSERT\_ANY\_THROW(a \* b);

}

TEST(TPolinom, Group\_Equal\_St\_1)

{

TPolinom a("xyz+xy");

TPolinom b("z+2");

TPolinom c("xyz^2+xyz+2xyz+2xy");

TPolinom d("xyz^2+3xyz+2xy");

EXPECT\_EQ(c, a \* b);

EXPECT\_EQ(c, d);

}

TEST(TPolinom, Group\_Equal\_St\_2)

{

TPolinom a("x+yz");

TPolinom b("2x+yz");

TPolinom c("2x^2+2xyz+xyz+y^2z^2");

TPolinom d("2x^2+3xyz+y^2z^2");

EXPECT\_EQ(c, a \* b);

EXPECT\_EQ(c, d);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_1)

{

TPolinom a("x^2");

double b = 20.9;

ASSERT\_NO\_THROW(a \* b);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_2)

{

TPolinom a("x^2+y-z");

double b = 5.5;

ASSERT\_NO\_THROW(a \* b);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_is\_Correct\_1)

{

TPolinom a("100x^3");

double b = 19.5;

TPolinom c("1950x^3");

EXPECT\_EQ(c, a \* b);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_is\_Correct\_2)

{

TPolinom a("10x^2-2y+4z");

double b = 3.5;

TPolinom c("35x^2-7y+14z");

EXPECT\_EQ(c, a \* b);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_OtherSide\_1)

{

double b = 20.9;

TPolinom a("x^2");

ASSERT\_NO\_THROW(b \* a);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_OtherSide\_2)

{

double b = 5.5;

TPolinom a("x^2+y-z");

ASSERT\_NO\_THROW(b \* a);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_OtherSide\_is\_Correct\_1)

{

double b = 19.5;

TPolinom a("100x^3");

TPolinom c("1950x^3");

EXPECT\_EQ(c, b\* a);

}

TEST(TPolinom, Mult\_Pol\_On\_Double\_OtherSide\_is\_Correct\_2)

{

double b = 3.5;

TPolinom a("10x^2-2y+4z");

TPolinom c("35x^2-7y+14z");

EXPECT\_EQ(c, b \* a);

}

TEST(TMonom, can\_create\_node)

{

ASSERT\_NO\_THROW(Monom a);

}

TEST(TRingList, can\_create\_list)

{

ASSERT\_NO\_THROW(TRingList<int> l);

}

class EmptyList : public testing::Test

{

protected:

TRingList<int> Example;

public:

EmptyList() {};

~EmptyList() {};

};

TEST\_F(EmptyList, Copy\_Empty)

{

ASSERT\_NO\_THROW(TRingList<int>variable(Example));

}

TEST\_F(EmptyList, Copy\_Empty\_Correct)

{

TRingList<int>variable(Example);

variable.Reset();

EXPECT\_EQ(Example.GetCurrent(), Example.GetNext());

}

TEST\_F(EmptyList, Eq\_Two\_Empty)

{

TRingList<int> variable;

EXPECT\_EQ(true, Example == variable);

}

TEST\_F(EmptyList, Assign\_Two\_Empty)

{

TRingList<int> variable;

ASSERT\_NO\_THROW(Example = variable);

}

TEST\_F(EmptyList, Assign\_Two\_Emptyi\_Correct1)

{

TRingList<int> variable;

Example = variable;

EXPECT\_EQ(Example.GetCurrent(), Example.GetNext());

}

TEST\_F(EmptyList, Insert\_To\_Tail\_For\_Empty)

{

Example.InsertInEnd(5);

Example.Reset();

EXPECT\_EQ(5, Example.GetCurrent()->data);

EXPECT\_EQ(Example.GetCurrent(), Example.GetCurrent()->pNext->pNext);

}

TEST\_F(EmptyList, Search\_For\_Empty)

{

ASSERT\_NO\_THROW(Example.Search(5));

}

TEST\_F(EmptyList, Search\_For\_Empty\_Correct)

{

TLink<int>\* p = Example.Search(6);

EXPECT\_EQ(NULL, p);

}

TEST\_F(EmptyList, Clean\_Empty)

{

ASSERT\_NO\_THROW(Example.Clean());

}

TEST\_F(EmptyList, Can\_Not\_Del\_From\_Empty)

{

ASSERT\_ANY\_THROW(Example.Delete(5));

}

TEST\_F(EmptyList, Del\_From\_Empty\_Correct)

{

Example.InsertInEnd(5);

Example.Delete(5);

Example.Reset();

EXPECT\_EQ(Example.GetCurrent(), Example.GetCurrent()->pNext);

}

TEST\_F(EmptyList, Can\_Not\_Del\_Not\_Memb)

{

Example.InsertInEnd(5);

ASSERT\_ANY\_THROW(Example.Delete(6));

}

TEST\_F(EmptyList, Clean\_Empty\_Correct)

{

Example.Clean();

Example.Reset();

EXPECT\_EQ(Example.GetCurrent(), Example.GetCurrent()->pNext);

}

class ThreeNodesList : public testing::Test

{

protected:

TRingList<int> Example;

public:

ThreeNodesList()

{

Example.InsertInEnd(1);

Example.InsertInEnd(2);

Example.InsertInEnd(3);

}

~ThreeNodesList() {};

};

TEST\_F(ThreeNodesList, 3\_Nodes\_Correct)

{

Example.Reset();

EXPECT\_EQ(1, Example.GetCurrent()->data);

EXPECT\_EQ(2, Example.GetCurrent()->pNext->data);

EXPECT\_EQ(3, Example.GetCurrent()->pNext->pNext->data);

EXPECT\_EQ(Example.GetCurrent()->pNext->pNext->pNext, Example.GetCurrent()->pNext->pNext->pNext->pNext->pNext->pNext->pNext);

}

TEST\_F(ThreeNodesList, Copy\_Not\_Empty)

{

ASSERT\_NO\_THROW(TRingList<int> variable(Example));

}

TEST\_F(ThreeNodesList, Copy\_Of\_Not\_Empty\_Correct)

{

TRingList<int> variable(Example);

variable.Reset();

EXPECT\_EQ(1, variable.GetCurrent()->data);

EXPECT\_EQ(2, variable.GetCurrent()->pNext->data);

EXPECT\_EQ(3, variable.GetCurrent()->pNext->pNext->data);

EXPECT\_EQ(variable.GetCurrent()->pNext->pNext->pNext, variable.GetCurrent()->pNext->pNext->pNext->pNext->pNext->pNext->pNext);

}

TEST\_F(ThreeNodesList, Insert\_In\_End\_For\_Not\_Empty)

{

Example.InsertInEnd(5);

Example.Reset();

EXPECT\_EQ(5, Example.GetCurrent()->pNext->pNext->pNext->data);

EXPECT\_EQ(Example.GetCurrent()->pNext->pNext->pNext->pNext, Example.GetCurrent()->pNext->pNext->pNext->pNext->pNext->pNext->pNext->pNext->pNext);

}

TEST\_F(ThreeNodesList, Search\_For\_Not\_Empty)

{

ASSERT\_NO\_THROW(Example.Search(3));

}

TEST\_F(ThreeNodesList, sSearch\_For\_Not\_Empty\_Correct)

{

Example.Reset();

TLink<int> \* p = Example.Search(3);

TLink<int> \* ptrue = Example.GetCurrent()->pNext->pNext;

EXPECT\_EQ(ptrue, p);

}

TEST\_F(ThreeNodesList, Search\_Not\_a\_Member\_For\_Not\_Empty)

{

ASSERT\_NO\_THROW(Example.Search(5));

}

TEST\_F(ThreeNodesList, Search\_Not\_a\_Member\_for\_Not\_Empty\_Correct)

{

TLink<int> \* p = Example.Search(7);

EXPECT\_EQ(nullptr, p);

}

TEST\_F(ThreeNodesList, can\_clean\_not\_empty\_list)

{

ASSERT\_NO\_THROW(Example.Clean());

}

TEST\_F(ThreeNodesList, Clean\_Not\_Empty\_Correct)

{

Example.Clean();

Example.Reset();

EXPECT\_EQ(Example.GetCurrent(), Example.GetCurrent()->pNext);

}

TEST\_F(ThreeNodesList, Assign\_2\_Not\_Empty)

{

TRingList<int> variable;

variable.InsertInEnd(22);

variable.InsertInEnd(33);

ASSERT\_NO\_THROW(Example = variable);

}

TEST\_F(ThreeNodesList, Assign\_2\_Not\_Empty\_Of\_EQ\_SIZE)

{

TRingList<int> variable;

variable.InsertInEnd(11);

variable.InsertInEnd(22);

variable.InsertInEnd(33);

Example = variable;

Example.Reset();

EXPECT\_EQ(11, Example.GetCurrent()->data);

EXPECT\_EQ(22, Example.GetCurrent()->pNext->data);

EXPECT\_EQ(33, Example.GetCurrent()->pNext->pNext->data);

EXPECT\_EQ(Example.GetCurrent()->pNext->pNext->pNext, Example.GetCurrent()->pNext->pNext->pNext->pNext->pNext->pNext->pNext);

}

TEST\_F(ThreeNodesList, Compare\_2\_EQ\_Correct)

{

TRingList<int> variable(Example);

EXPECT\_EQ(true, Example == variable);

}

class TwoListsTest : public testing::Test

{

protected:

TRingList<int> Example;

TRingList<int> variable;

TRingList<int> z;

public:

TwoListsTest()

{

Example.InsertInEnd(1);

Example.InsertInEnd(2);

Example.InsertInEnd(3);

variable.InsertInEnd(11);

variable.InsertInEnd(22);

}

~TwoListsTest() {}

};

TEST\_F(TwoListsTest, Assign\_2\_Not\_Empty\_Of\_Not\_EQ\_SIZE)

{

ASSERT\_NO\_THROW(Example = variable);

}

TEST\_F(TwoListsTest, Assign\_2\_Not\_Empty\_Of\_Not\_EQ\_SIZE\_Correct)

{

Example = variable;

Example.Reset();

EXPECT\_EQ(11, Example.GetCurrent()->data);

EXPECT\_EQ(22, Example.GetCurrent()->pNext->data);

EXPECT\_EQ(Example.GetCurrent()->pNext->pNext, Example.GetCurrent()->pNext->pNext->pNext->pNext->pNext);

}

TEST\_F(TwoListsTest, Assign\_To\_Empty\_Correct)

{

Example = z;

Example.Reset();

EXPECT\_EQ(Example.GetCurrent()->pNext, Example.GetCurrent());

}

TEST\_F(TwoListsTest, Assign\_Empty\_To\_Not\_Empty\_Correct)

{

z = Example;

EXPECT\_EQ(Example, z);

}

TEST\_F(TwoListsTest, Compare\_2\_Not\_Empty)

{

ASSERT\_NO\_THROW(Example == variable);

}

TEST\_F(TwoListsTest, Compare\_2\_Not\_EQ\_of\_EQ\_SIZE)

{

variable.Insert(3);

EXPECT\_EQ(false, Example == variable);

}

TEST\_F(TwoListsTest, Compare\_2\_Not\_EQ\_With\_Bigger\_Left)

{

EXPECT\_EQ(false, Example == variable);

}

TEST\_F(TwoListsTest, Compare\_2\_Not\_EQ\_With\_Bigger\_Right)

{

EXPECT\_EQ(false, variable == Example);

}

TEST\_F(TwoListsTest, Compare\_With\_Empty\_Left)

{

EXPECT\_EQ(false, z == variable);

}

TEST\_F(TwoListsTest, Compare\_With\_Empty\_Right)

{

int s;

EXPECT\_EQ(false, variable == z);

}