МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

**(ННГУ)**

**Институт информационных технологий, математики и механики**

Направление подготовки: «Программная инженерия»

**ОТЧЕТ**

по лабораторной работе

**Аналитические преобразования полиномов от нескольких переменных (списки)**

Выполнил:

студент группы 3822Б1ПР2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кондратьев Я.С.

Подпись

Нижний Новгород

2023 г.

Содержание

[1. Введение 3](#_Toc153310549)

[2. Постановка задачи 4](#_Toc153310550)

[3. Руководство пользователя 5](#_Toc153310551)

[4. Руководство программиста 6](#_Toc153310552)

[4.1. Описание структуры программы 6](#_Toc153310553)

[4.2. Описание структур данных 6](#_Toc153310554)

[4.2.1. TList](#_Toc153310555) 6

[4.2.2. THeadList (шаблонный класс, наследованный от TList) 6](#_Toc153310556)

[4.2.3. TMonom 6](#_Toc153310557)

[4.2.4. TPolinom (наследуется от специализации THeadList<TMonom>) 7](#_Toc153310558)

[4.2.5. TNode 7](#_Toc153310559)

[5. Эксперименты 8](#_Toc153310563)

[6. Заключение 10](#_Toc153310564)

[7. Литература 11](#_Toc153310565)

[8. Приложение 1 12](#_Toc153310566)

[9. Приложение 2 13](#_Toc153310567)

[10. Приложение 3 17](#_Toc153310568)

[11. Приложение 4 19](#_Toc153310569)

[12. Приложение 5 20](#_Toc153310570)

[13. Приложение 6 24](#_Toc153310571)

[14. Приложение 7 25](#_Toc153310572)  
  
[15. Приложение 8 26](#_Toc153310572)

1. Введение

Анализ данных – важный аспект использования компьютеров в современном мире. Компьютеры применяются не только для вычислительных задач, но и для таких задач, как компьютерное доказательство теорем, логический вывод информации и анализ текстовых данных. Примером такой задачи является обработка полиномов, представленных в аналитической форме. Полиномы широко используются в математике для различных приложений, включая аппроксимацию экспериментальных данных и построение функциональных зависимостей.

Цель данной лабораторной работы заключается в изучении методов обработки полиномов с помощью компьютера. Для этого мы проведем анализ различных способов хранения полиномов и разработаем соответствующие программы для их обработки. Основное внимание будет уделено практическому использованию структур данных на основе списков. В процессе выполнения работы мы создадим универсальное представление линейных списков и программа для работы с этими списками, которую можно будет применять и в других областях.

1. Постановка задачи

В рамках лабораторной работы ставится задача создания программных средств, поддерживающих эффективное хранение полиномов и выполнение основных операций над ними:

1. сложение/вычитание;
2. умножение
3. копирование/присваивание;
4. сравнение.

В процессе выполнения лабораторной работы требуется использовать систему контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

Выполнение работы предполагает решение следующих задач:

1. Реализация односвязных списков:
   1. Реализация класса TDatValue;
   2. Реализация класса TRootLink;
   3. Реализация класса TDatLink;
   4. Реализация класса TDatList;
   5. Реализация класса THeadRing.
2. Реализация класса TMonom.
3. Реализация класса TPolinom.
4. Обеспечение работоспособности тестов и примера использования.
5. Реализация визуального интерфейса на основе Visual C++ WinForms.
6. Руководство пользователя

При запуске программы автоматически выводится 6 уже заданных мономов, а также пример сложения и вычитания полиномов. Пользователю никаких действий не предлагается.

1. Руководство программиста

## Описание структуры программы

Программа состоит из следующих модулей:

* Модуль Polinom, содержащий реализацию классов THeadList, TList, TNode, TMonom, TPolinom (файлы в директориях ./include/ и ./src/).
* Модуль Polinom\_test. Набор тестов для класса TPolinom. Включает в себя файл ./test/test\_polinom.cpp. Реализованы они с помощью использования фреймворка Google Test.
* Примеры использования классов (файлы в директории ./samples/).

## Описание структур данных

### TList

protected:

TNode<T>\* pFirst - первое звено

TNode<T>\* pLast - последнее звено

TNode<T>\* pCurrent - текущее звено

TNode<T>\* pPrevious - звено перед текущим

TNode<T>\* pStop - значение указателя, означающего конец списка

int length - количество звеньев в списке

public:

TList () - конструктор

~TDatList(void) – деструктор  
int GetLength() – получить длину списка  
void InsertFirst(T item) – вставить звено перед первым

void InsertLast(T item) - вставить звено последним

void InsertCurrent(T item) – вставить звено перед текущим

void DeleteFirst() - удалить первое звено

void DeleteCurrent() - удалить текущее звено

void GoNext() - сдвиг вправо текущего звена

void Reset() - установить на начало списка

T GetCurrentItem() - установить текущее звено

void SetCurrentItem(T item) - получить номер текущего звена

bool IsEmpty() – проверка списка на пустоту

### THeadList (шаблонный класс, наследованный от TList)

protected:

TNode<T>\* pHead - заголовок, pFirst - звено за pHead

public:

THeadList( - конструктор

~THeadList() – деструктор

void InsertFirst(T item) - после заголовка

void DeleteFirst() - удалить первое звено

### TMonom

double Coeff - коэффициент монома

int index – индекс

TMonom() – конструктор

TMonom(double coef1, int degX, int degY, int degZ) - конструктор

void SetCoef(int cval) – задать коэффициент

int GetCoef(void) – получить коэффициент

void SetIndex(int ival) – задать индекс

int GetIndex(void) – получить индекс

void SetDeg(const int\* deg) – задать переменные

int\* GetDeg(void) – получить переменные

bool operator==(const TMonom& other) - оператор сравнения равенства

bool operator<(const TMonom& other) - оператор сравнения меньше

bool operator>(const TMonom& other) – оператор сравнения больше

string ToString() – перевод монома в строку

### TPolinom (наследуется от специализации THeadList<TMonom>)

const int nonDisplayedZeros - Количество неотображаемых нулей при выводе коэффициента полинома

public:

TPolinom() – конструктор

TPolinom(TPolinom& other); - конструктор копирования

TPolinom(string str) – Разбивает строку str на мономы  
 TPolinom& operator=(const TPolinom& other) – оператор присваивания

TPolinom operator+(TPolinom& q) - оператор сложения  
 void AddMonom(TMonom newMonom) – добавить моном в полином  
 TPolinom AddPolinom(TPolinom& other) - добавление полинома  
string ToString() - получить полином в виде строки

### TNode

T value - значение элемента списка  
 TNode<T>\* pNext - указатель на следующий узел списка

1. Эксперименты

Эксперименты проводились на ПК с следующими параметрами:

1. Операционная система: Windows 10
2. Процессор: Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz 3.60 GHz
3. Версия Visual Studio: 2022

В таблицах ниже под временем работы подразумевается усредненное время работы, основанное на 10 последовательных запусках на различных данных, полученных генерацией псевдослучайных чисел.

|  |  |
| --- | --- |
| Количество переменных и размер полинома | Время работы оператора сложения (в млс) |
| 100 | 1.2 |
| 500 | 13.5 |
| 1 000 | 23.1 |
| 2 000 | 80.9 |
| 3 000 | 197.5 |
| 5 000 | 432.8 |

Таблица 1 – Измерение работы оператора сложения

Рис. 1 – График времени работы оператора сложения

|  |  |
| --- | --- |
| Количество переменных и размер полинома | Время работы оператора умножения (в млс) |
| 20 | 2 |
| 30 | 8,5 |
| 40 | 12,5 |
| 50 | 43,2 |
| 75 | 104,5 |
| 100 | 444,7 |
| 150 | 2789,2 |
| 200 | 13205, 9 |

*Таблица 2 – Измерение времени работы оператора умножения*

Рис. 2– График времени работы оператора умножения

Сложность оператора умножения примерно соответствует экспоненциальной сложности.

1. Заключение

В заключение, можно отметить, что проведенный анализ показал, что хранение полиномов в виде односвязного списка является наиболее эффективным способом. Это позволяет легко выполнять операции над полиномами, такие как сложение, вычитание и умножение.

Таким образом, цель лабораторной работы достигнута, мы изучили методы компьютерной обработки полиномов и разработали программы для их обработки. В процессе выполнения лабораторной работы мы также разработали универсальный способ представления линейных связных списков и программу для работы со списками, которая может быть применена в различных областях приложений.

В заключение стоит отметить, что полученные результаты могут быть использованы для дальнейшего развития методов обработки полиномов. Можно исследовать возможность создания более эффективных алгоритмов обработки полиномов и разработки программ, оптимизированных под различные вычислительные устройства.

1. Литература
   * + 1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.
       2. Сысоев А.В. Полиномы / Сысоев А.В. [Электронный ресурс] // cloud.unn.ru : [сайт]. — URL: [https://cloud.unn.ru/s/x33MEa9on8HgNgw](https://cloud.unn.ru/s/x33MEa9on8HgNgw%20) (дата обращения: 10.12.2023).).
2. Приложение 1

THeadList.h

#pragma once

#include "TList.h"

using namespace std;

template<class T>

class THeadList : public TList<T> {

protected:

TNode<T>\* pHead; // заголовок, pFirst - звено за pHead

public:

THeadList();

~THeadList();

void InsertFirst(T item); // вставка звеньев после заголовка

void DeleteFirst(); // удалить первое звено

};

template<class T>

THeadList<T>::THeadList() : TList<T>()//cоздается заголовочный узел pHead и устанавливается связь между pHead и первым узлом списка.

//Это делается путем установки указателя pNext заголовочного узла на первый узел списка, а также установкой связи между последним

// узлом списка и заголовочным узлом, чтобы обеспечить циклическую структуру списка.

{

pHead = new TNode<T>;

pHead->pNext = this->pFirst;

this->pStop->pNext = pHead;

}

template<class T>

THeadList<T>::~THeadList()//очистка памяти

{

if (pHead != nullptr)

delete pHead;

pHead = nullptr;

}

template <class T>

void THeadList<T>::InsertFirst(T item)//вставка нового узла

{

TList<T>::InsertFirst(item);

pHead->pNext = this->pFirst;

}

template <class T>

void THeadList<T>::DeleteFirst()//удаление первого узла списка

{

TList<T>::DeleteFirst();

pHead->pNext = this->pFirst;

}

1. Приложение 2

TList.h

#pragma once

#include "TNode.h"

#include <iostream>

using namespace std;

template<class T>

class TList

{

protected:

TNode<T>\* pFirst; // первое звено

TNode<T>\* pCurrent; // текущее звено

TNode<T>\* pPrevious; // звено перед текущим

TNode<T>\* pLast; // последнее звено

TNode<T>\* pStop; // значение указателя, означающего конец списка

int length; // количество звеньев в списке

public:

TList();

~TList();

int GetLength() { return length; }

bool IsEmpty(); // список пуст ?

// вставка звеньев

void InsertFirst(T item); // перед первым

void InsertCurrent(T item); // перед текущим

void InsertLast(T item); // вставить последним

// удаление звеньев

void DeleteFirst(); // удалить первое звено

void DeleteCurrent(); // удалить текущее звено

void GoNext(); // сдвиг вправо текущего звена

// (=1 после применения GoNext для последнего звена списка)

void Reset(); // установить на начало списка

bool IsEnd(); // список завершен ?

T GetCurrentItem();

void SetCurrentItem(T item) { pCurrent->value = item; }

};

template <class T>

TList<T>::TList()//инициализируются поля класса

{

pFirst = nullptr;

pCurrent = nullptr;

pPrevious = nullptr;

pLast = nullptr;

pStop = new TNode<T>;

length = 0;

}

template <class T>

TList<T>::~TList() // освобождение памяти

{

while (length > 0) {

DeleteFirst();

}

if (pStop != nullptr) {

delete pStop;

pStop = nullptr;

}

pFirst = nullptr;

pPrevious = nullptr;

pCurrent = nullptr;

pLast = nullptr;

}

template <class T>

bool TList<T>::IsEmpty()

{

return length == 0;

}

template <class T>

void TList<T>::InsertFirst(T item)

{

if (length == 0) {

pFirst = new TNode<T>;//создается новый узел pFirst

pFirst->value = item;

pFirst->pNext = pStop;//устанавливается связь с заглушкой pStop

pLast = pFirst;// указывают на новый узел pFirst

pCurrent = pFirst;

}

else {

TNode<T>\* prev = pFirst;

pFirst = new TNode<T>;

pFirst->value = item;

pFirst->pNext = prev;

}

length++;

}

template <class T>

void TList<T>::InsertLast(T item)

{

if (length == 0)

InsertFirst(item);

else {

TNode<T>\* prev = pLast;

pLast = new TNode<T>;

pLast->value = item;

pLast->pNext = pStop;

prev->pNext = pLast;//Указатель pNext предыдущего узла prev устанавливается на новый узел pLast

length++;

}

}

template <class T>

void TList<T>::InsertCurrent(T item)

{

if (length == 0)

InsertFirst(item);

else {

TNode<T>\* pPrevios = pCurrent;

pCurrent = new TNode<T>;//создается новый узел pCurrent, в котором хранится значение нового элемента

pCurrent->value = item;

pCurrent->pNext = pPrevios->pNext;

pPrevios->pNext = pCurrent;

length++;

}

}

template <class T>

void TList<T>::DeleteFirst()

{

if (pFirst != nullptr) {

TNode<T>\* prev = pFirst;

pFirst = pFirst->pNext;

if (pCurrent == prev)

pCurrent = pFirst;

if (pLast == prev)

pLast = pFirst;

if (pPrevious == prev)

pPrevious = nullptr;

if (prev != nullptr) {

delete prev;

prev = nullptr;

}

length--;

}

}

template <class T>

void TList<T>::DeleteCurrent()//Если текущий элемент не является заглушкой pStop, то устанавливаются новые

//значения для указателей, чтобы пропустить текущий элемент. Затем происходит удаление текущего элемента и

// корректировка указателя pPrevious.

{

if (pCurrent != nullptr) {

TNode<T>\* prev = pCurrent;

pCurrent = pPrevious;

pCurrent->pNext = prev->pNext;

if (prev != nullptr)

delete prev;

pPrevious = nullptr;

length--;

}

}

template <class T>

T TList<T>::GetCurrentItem()

{

if (pCurrent == pStop)

throw " ";

return pCurrent->value;//возвращает значение текущего элемента

}

template <class T>

void TList<T>::Reset()//сбрасывает текущий элемент списка на начало

{

pPrevious = nullptr;

pCurrent = pFirst;

}

template <class T>

void TList<T>::GoNext()//переходит к следующему элементу списка

{

if (pCurrent->pNext == nullptr)

throw - 1;

pPrevious = pCurrent;

pCurrent = pCurrent->pNext;

}

template <class T>

bool TList<T>::IsEnd()//проверяет, достигнут ли конец списка

{

return pCurrent->pNext == pStop;

}

1. Приложение 3

TMonom.h

#pragma once

#include <string>

using namespace std;

struct TMonom

{

double coef; // коэффициент монома

int index; // индекс (свертка степеней переменных X, Y и Z)

TMonom() {

coef = 0;

index = 0;

}

TMonom(double coef1, int degX, int degY, int degZ) {

coef = coef1;

index = degX \* 100 + degY \* 10 + degZ;

}

void SetCoef(int cval) { coef = cval; }

int GetCoef(void) { return coef; }

void SetIndex(int ival) { index = ival; }

int GetIndex(void) { return index; }

bool operator==(const TMonom& other) {

return index == other.index && coef == other.coef;

}

bool operator>(const TMonom& other) {

return index >= other.index || coef > other.coef;

}

bool operator<(const TMonom& other) {

return index <= other.index || coef < other.coef;

}

string ToString() {

string result;

int degX = index / 100;

int degY = index / 10 % 10;

int degZ = index % 10;

if (coef != 1)

result += to\_string(round(coef \* 100000) / 100000);

if (degX != 0) {

result += "X";

if (degX != 1)

result += "^" + to\_string(degX);

}

if (degY != 0) {

result += "\*Y";

if (degY != 1)

result += "^" + to\_string(degY);

}

if (degZ != 0) {

result += "\*Z";

if (degZ != 1)

result += "^" + to\_string(degZ);

}

return result;

}

};

1. Приложение 4

TNode.h

#pragma once

template<class T>

struct TNode

{

T value;// значение элемента списка

TNode<T>\* pNext;//указатель на следующий узел списка

};

1. Приложение 5

TPolinom.h

#pragma once

#include "THeadList.h"

#include "TMonom.h"

#include <string>

#include <sstream>

const int nonDisplayedZeros = 4;

// Количество неотображаемых нулей при выводе коэффициента полинома

// Кол-во символов после запятой = 6 - nonDisplayedZeros

class TPolinom : public THeadList<TMonom>

{

public:

TPolinom();

TPolinom(TPolinom& other);

TPolinom(string str);

TPolinom& operator=(const TPolinom& other); // присваивание

TPolinom operator+(TPolinom& q); // сложение полиномов

void AddMonom(TMonom newMonom); // добавление монома

TPolinom AddPolinom(TPolinom& other); // добавление полинома

string ToString(); // перевод в строку

};

TPolinom::TPolinom() : THeadList<TMonom>::THeadList()//инициализирует список пустым

{

;

}

TPolinom::TPolinom(TPolinom& other) : THeadList<TMonom>::THeadList()//создает копию полинома other

{

other.Reset();

for (int i = 0; i < other.length; i++) {

AddMonom(other.pCurrent->value);

other.GoNext();

}

}

TPolinom::TPolinom(string str)

{

stringstream ss(str);

string monom;

while (getline(ss, monom, '+')) {//Разбивает строку str на мономы, каждый из которых представляет

//собой число типа double, за которым следует набор переменных (X, Y, Z) со своими степенями

stringstream ss1(monom);

string tmp;

double coef;

int degX = 0, degY = 0, degZ = 0;

getline(ss1, tmp, '\*');

coef = stod(tmp);

while(getline(ss1, tmp, '\*')) {

switch (tmp[0])

{

case 'X':

if (tmp.size() >= 3)

degX = tmp[2] - '0';

else

degX = 1;

break;

case 'Y':

if (tmp.size() >= 3)

degY = tmp[2] - '0';

else

degY = 1;

break;

case 'Z':

if (tmp.size() >= 3)

degZ = tmp[2] - '0';

else

degZ = 1;

break;

default:

break;

}

}

AddMonom(TMonom(coef, degX, degY, degZ));

}

}

TPolinom& TPolinom::operator=(const TPolinom& other)//обеспечивает присваивание одного полинома другому

{

if (this == &other)

return \*this;

if (pFirst != nullptr) {

while (length > 0) {

DeleteFirst();

}

}

TNode<TMonom>\* tmp = other.pFirst;

for (int i = 0; i < other.length; i++) {

AddMonom(tmp->value);

tmp = tmp->pNext;

}

return \*this;

}

void TPolinom::AddMonom(TMonom m)//добавляет новый моном newMonom в полином

{

if (m.GetCoef() == 0)

return;

if (pFirst == nullptr)

InsertFirst(m);

else {

int i = 0;

Reset();

while (i <= length && pCurrent->value.GetIndex() != m.GetIndex() && pCurrent->value > m)

{

i++;

if (i == length) {

InsertLast(m);

return;

}

GoNext();

}

if (pCurrent->value.GetIndex() == m.GetIndex()) {

double coef = pCurrent->value.GetCoef() + m.GetCoef();

pCurrent->value.SetCoef(coef);

if (coef == 0)

DeleteCurrent();

}

else {

if (pCurrent == pFirst) {

InsertFirst(m);

}

else if (pPrevious != nullptr) {

pCurrent = pPrevious;

pPrevious = nullptr;

InsertCurrent(m);

}

}

Reset();

}

}

TPolinom TPolinom::operator+(TPolinom& other)//выполняет сложение двух полиномов

{

TPolinom res(\*this);

other.Reset();

for (int i = 0; i < other.length; i++) {

res.AddMonom(other.pCurrent->value);

other.GoNext();

}

return res;

}

TPolinom TPolinom::AddPolinom(TPolinom& other)//добавляет полином other к текущему полиному

{

return \*this + other;

}

string TPolinom::ToString()//преобразует полином в строку

{

string result;

TNode<TMonom>\* tmp = pFirst;

if (length == 0) {

return "0";

}

if (tmp != nullptr)

{

result += tmp->value.ToString();

tmp = tmp->pNext;

}

while (tmp != nullptr && tmp != pStop)

{

if (tmp->value.GetCoef() > 0) {

result += " + " + tmp->value.ToString();

}

else if (tmp->value.GetCoef() < 0) {

tmp->value.SetCoef(-1 \* tmp->value.GetCoef());

result += " - " + tmp->value.ToString();

tmp->value.SetCoef(-1 \* tmp->value.GetCoef());

}

tmp = tmp->pNext;

}

return result;

}

1. Приложение 6

Polinom\_sample.cpp

#include "TPolinom.h"

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

try

{

cout << "Мономы:" << endl;

TMonom m1(3, 5, 2, 5);

TMonom m2(-5, 4, 3, 3);

TMonom m3(7, 3, 5, 1);

TMonom m4(4, 3, 2, 6);

TMonom m5(0, 2, 7, 3);

TMonom m6(-6, 2, 1, 8);

cout << "m1: " << m1.ToString() << endl;

cout << "m2: " << m2.ToString() << endl;

cout << "m3: " << m3.ToString() << endl;

cout << "m4: " << m4.ToString() << endl;

cout << "m5: " << m5.ToString() << endl;

cout << "m6: " << m6.ToString() << endl;

TPolinom p1;

p1.AddMonom(m1);

p1.AddMonom(m2);

p1.AddMonom(m3);

cout << endl << "Первый полином от m1, m2, m3: " << p1.ToString() << endl;

TPolinom p2;

p2.AddMonom(m4);

p2.AddMonom(m5);

p2.AddMonom(m6);

cout << endl << "Воторой полином от m4, m5, m6: " << p2.ToString() << endl;

TPolinom res(p1.ToString());

res = p1 + p2;

cout << endl << "res = p1 + p2: " << res.ToString() << endl;

cout << "Длина: " << res.GetLength() << endl << endl;

res = p2;

cout << "res = p2: " << res.ToString() << endl << endl;

}

catch (exception e)

{

cout << e.what();

}

return 0;

}

1. Приложение 7

test\_main.cpp

#include <gtest.h>

int main(int argc, char \*\*argv) {

::testing::InitGoogleTest(&argc, argv);

return RUN\_ALL\_TESTS();

}

1. Приложение 8

test\_polinom.cpp

#include <gtest.h>

#include "TPolinom.h"

using namespace std;

TEST(Polinom, Created\_polinom\_is\_empty) {

TPolinom a;

ASSERT\_EQ(0, a.GetLength());

}

TEST(Polinom, Created\_polinom\_with\_one\_monom\_is\_not\_empty) {

TPolinom a;

a.AddMonom(TMonom(1, 1, 1, 1));

ASSERT\_NE(0, a.GetLength());

}

TEST(Polinom, Coefficients\_add\_up\_if\_degrees\_are\_equal) {

TPolinom a;

a.AddMonom(TMonom(1, 1, 1, 1));

a.AddMonom(TMonom(1, 1, 1, 1));

ASSERT\_EQ("2.000000X\*Y\*Z", a.ToString());

}

TEST(Polinom, Coefficients\_dont\_add\_up\_if\_degrees\_are\_different) {

TPolinom a;

a.AddMonom(TMonom(1, 1, 1, 1));

a.AddMonom(TMonom(1, 2, 1, 1));

ASSERT\_EQ("X^2\*Y\*Z + X\*Y\*Z", a.ToString());

}

TEST(Polinom, Monom\_delete\_if\_coefficient\_equal\_zero) {

TPolinom a;

a.AddMonom(TMonom(0, 1, 1, 1));

ASSERT\_EQ(0, a.GetLength());

}

TEST(Polinom, Correct\_adding\_operator) {

TMonom m1(3, 5, 2, 5);

TMonom m2(-5, 4, 3, 3);

TMonom m3(7, 3, 5, 1);

TMonom m4(4, 3, 2, 6);

TMonom m5(0, 2, 7, 3);

TMonom m6(-6, 2, 1, 8);

TPolinom p1, p2, res;

p1.AddMonom(m1);

p1.AddMonom(m2);

p1.AddMonom(m3);

p2.AddMonom(m4);

p2.AddMonom(m5);

p2.AddMonom(m6);

res.AddMonom(m1);

res.AddMonom(m2);

res.AddMonom(m3);

res.AddMonom(m4);

res.AddMonom(m5);

res.AddMonom(m6);

ASSERT\_EQ(res.ToString(), (p1 + p2).ToString());

}