

## Содержание

<b>Практическая работа. Абстрактный тип данных (ADT) «Полином» .....</b>	<b>2</b>
<b>Тема:</b> Классы C++, библиотека STL, библиотечный класс Tlist C++ Builder. ....	2
<b>Задание</b> .....	2
<b>Рекомендации к выполнению</b> .....	5
<b>Порядок выполнения</b> .....	13
<b>Содержание отчета</b> .....	14
<b>Контрольные вопросы</b> .....	14

## **Практическая работа. Абстрактный тип данных (ADT) «Полином»**

**Тема:** Классы C++, библиотека STL, библиотечный класс Tlist C++ Builder.

**Цель:** Сформировать практические навыки реализации абстрактных типов данных с помощью классов.

### **Задание**

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

### **Спецификация абстрактного типа данных «Полином».**

#### **ADT TPolу**

### **Данные**

Полиномы TPolу - это неизменяемые полиномы с целыми коэффициентами.

### **Операции**

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

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

Наименование операции	Описание
<b>Конструктор</b>	
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:	Указатель на созданный полином.
Постусловия:	Нет.

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

Выход:	Полином.
Постусловия:	Нет.
<b>Умножить</b>	
Вход:	q - полином.
Предусловия:	Нет.
Процесс:	Создаёт полином, являющийся результатом умножения полинома на полином q и возвращает его.
Выход:	Полином.
Постусловия:	Нет.
<b>Вычесть</b>	
Вход:	q - полином.
Предусловия:	Нет.
Процесс:	Создаёт полином, являющийся результатом вычитания из полинома полинома q, и возвращает его.
Выход:	Полином.
Постусловия:	Нет.
<b>Минус</b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт полином, являющийся разностью ноль-полинома, и полинома и возвращает его.
Выход:	Полином.
Постусловия:	Нет.
<b>Равно</b>	
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.

<b>Дифференцировать</b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт полином, являющийся производной полинома и возвращает его. Например: $a = (x^3 + 7x + 5)$ , а.Дифференцировать = $3x^2 + 7$
Выход:	Полином.
Постусловия:	Нет.
<b>Вычислить</b>	
Вход:	$x$ – действительное число.
Предусловия:	Нет.
Процесс:	Вычисляет значение полинома в точке $x$ и возвращает его. Например: $a = (x^2 + 3x)$ , а.Вычислить(2) = 10
Выход:	Действительное число.
Постусловия:	Нет.
<b>Элемент</b>	
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**

### Рекомендации к выполнению

1. Тип данных реализовать, используя классы C++ и библиотеку STL.

2. Полином можно рассматривать как список одночленных полиномов, поэтому для реализации полинома полезно реализовать абстрактный вспомогательный тип данных одночленный полином. Спецификация для него приведена ниже.

### **Спецификация абстрактного типа данных Одночлен. ADT TMember**

#### **Данные**

Одночлен TMember - это изменяемые одночленные полиномы с целыми коэффициентами. Коэффициент и степень хранятся в полях целого типа FCoeff и FDegree соответственно.

#### **Операции**

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

**Таблица 2.** Описание операций на ADT TMember.

Наименование операции	Описание
<b><i>Конструктор</i></b>	
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:	Указатель на созданный одночленный полином.
Постусловия:	Нет.
<b><i>ЧитатьСтепень</i></b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает степень n одночленного полинома (содержимое поля FDegree). Степень нулевого полинома равна 0. Например:

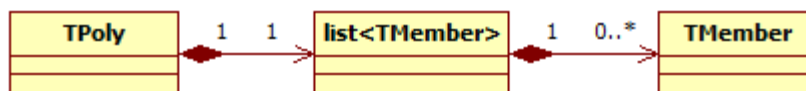
	$a = (1x^2)$ , а.Степень = 2
Выход:	n - целое число - степень полинома.
Постусловия:	Нет.
<b>ПисатьСтепень</b>	
Вход:	n - целое число - степень полинома.
Предусловия:	Нет.
Процесс:	Записывает степень n одночленного полинома в поле FDegree.
Выход:	Нет.
Постусловия:	Поле FDegree = n.
<b>ПисатьКоэффициент</b>	
Вход:	c - целое число - коэффициент полинома.
Предусловия:	Нет.
Процесс:	Записывает коэффициент c одночленного полинома в поле FCoeff.
Выход:	Нет.
Постусловия:	Поле FCoeff = c.
<b>Равно</b>	
Вход:	q - одночлен.
Предусловия:	Нет.
Процесс:	Сравнивает одночлен с одночленом q на равенство. Возвращает значение True, если одночлены равны, т.е. имеют одинаковые коэффициенты и степени, и значение False - в противном случае.
Выход:	Булевское значение.
Постусловия:	Нет.
<b>Дифференцировать</b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт одночлен, являющийся производной одночлена и возвращает его. Например:

	$a = (x^3)$ , а.Дифференцировать = $3x^2$ .
Выход:	Одночлен.
Постусловия:	Нет.
<b>Вычислить</b>	
Вход:	$x$ – действительное число.
Предусловия:	Нет.
Процесс:	Вычисляет значение одночлена в точке $x$ и возвращает его. Например: $a = (1x^2)$ , а.Вычислить(2) = 4.
Выход:	Действительное число.
Постусловия:	Нет.
<b>ОдночленВСтроку</b>	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Формирует строковое представление одночлена.
Выход:	Строка.
Постусловия:	Нет.

**end TMember**

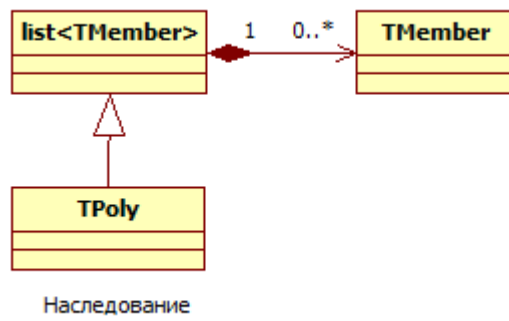
- Члены полинома храните в контейнере STL. Проанализируйте операции на полиноме и выберите тип контейнера.
- Для реализации одночленного полинома (одночлена) создайте класс Tmember, в который вынесите все операции на членах полинома. После выполнения каждой операции приводите полином к нормализованному виду: упорядочить, привести подобные, удалить нулевые члены (члены с нулевыми коэффициентами).
- Тип данных реализуйте в отдельном модуле UPoly.

**Ниже приведены два примера классов, реализующих абстракцию данных полином.**



Вложение





### Пример реализация полинома с помощью классов C++ Builder.

Для реализации полинома на C++ Builder можно использовать его библиотечный класс TList, который представляет собой список указателей на объекты класса TObject. Класс TObject является предком всех классов в модели ООП реализованной в C++ Builder.

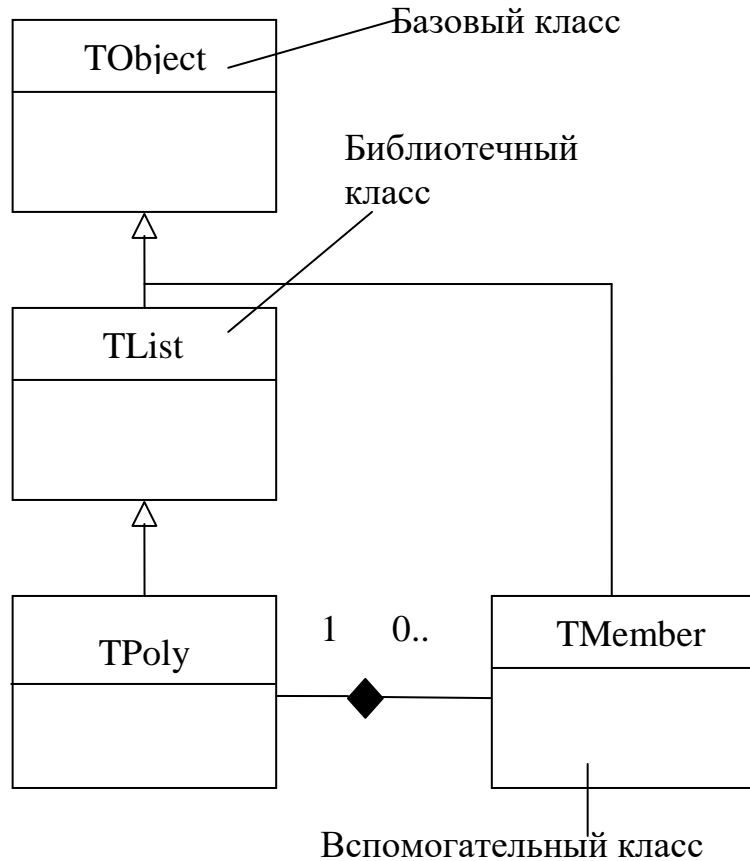


Рисунок 1. Диаграмма UML – описание реализации полинома.

Реализация полинома будет представлена модулем, состоящим из двух файлов: заголовочного файла UPoly.h с описанием классов TPoly, TMember и файла UPoly.cpp, содержащего описание методов классов. Заголовочный файл UPoly.h с описанием классов TPoly, TMember и файл головной программы для тестирования представлены ниже.

```
//---файл UPoly.h-----
#ifndef UPolyH
#define UPolyH
#include <vcl.h>
#include <math.h>
//-----
#endif
```

```

//-----
class TMember : public TObject
{
    int c,d;//Коэффициент и степень одночлена
    String Get(void) const;
public:
    TMember(int nc = 0, int nd = 0):
c(nc),d(nd){};//Конструктор
    TMember(const TMember& P);//Конструктор копирования
    bool operator == (const TMember& T);//Сравнение
//двух объектов
    void operator - (void){ Coeff = -
Coeff;};//Смена знака //одночлена на обратный
    TMember* Diff(void);//Дифференцирование одночлена
    double Eval(double r){return
c*row(r,d);};//Вычисление //одночлена
    __property String Member = {read = Get};//Читает объект
//в формате строки
    __property int Degree = {read = d, write = d};//Читать
//и писать степень
    __property int Coeff = {read = c, write = c};//Читать и
//писать коэффициент
};

//-----
class TPoly : public TList
{
    String Get(void) const;
    TMember* GetMember(int i) const;
    void Likeness(void);//Приведение подобных
    void RemoveZero(void);//Удаление нулевых
    void Invar(void);//Приведение полинома к
//нормализованному виду
    TPoly* MulNum(TMember* Num);
public:
    TPoly(int nc = 0, int nd = 0);//Конструктор
    TPoly(const TPoly& P);//Копирующий конструктор
    virtual __fastcall ~TPoly(void);//Деструктор
    int Degree(void) const;//Степень полинома
    double Eval(double x);//Вычисление значения полинома
    void operator - (void);//Изменение знака полинома
    TPoly* Diff(void);//Дифференцирование полинома

```

```

    TPoly* operator + (const TPoly& T); //Сложение
//полиномов
    TPoly* operator - (const TPoly& T); //Вычитание
//полиномов
    TPoly* operator * (const TPoly& B); //Умножение
//полиномов
    TMember* operator [] (int i); //Одночлен по индексу
    bool operator == (const TPoly& T); //Сравнение двух
//полиномов
    __property String Poly = {read = Get}; //Чтение полинома
//в формате строки
    __property TMember* Member[int index] = {read =
GetMember};
};

```

```

//-----файл UMain.cpp-----
#include <vcl.h>
#pragma hdrstop
#include "UPoly.h"
#include <iostream.h>
#include <stdarg.h>
//-----
#pragma argsused
int main(int argc, char* argv[])
{
    int i = 0;
    TMember* M;
    TPoly* P0 = new TPoly();
    TPoly* P1 = new TPoly(1,0);
    TPoly* P2 = new TPoly(1,1);
    TPoly* P4 = new TPoly(2,2);
    TPoly* P3 = *P1 + *P2; //1X^0 + 1X^1
    cout << "1X^0 + 1X^1" << P3->Poly.c_str() << endl; //1X^0 +
1X^1
    TPoly* P5 = (*P1) + (*P2); //1X^0 + 1X^1
    cout << "1X^0 + 1X^1" << P5->Poly.c_str() << endl;
    -*P5;
    cout << "- 1X^0 - 1X^1" << P5->Poly.c_str() << endl;
    TPoly* P6 = (*P5) * (*P2);
    cout << "- 1X^0 - 1X^2" << P6->Poly.c_str() << endl;
    -*P5;
    P6 = (*P5) * (*P3);
}

```

```

cout << "1X^0 + 2X^1 + 1X^2" << P6->Poly.c_str()<<endl;
TPoly* P7 = (*P5).Diff();
cout << "0X^0 + 1X^0" << P7->Poly.c_str()<<endl;
double r = (*P5).Eval(1);
cout << "P8.Eval(1) = 2 " << r <<endl;
//M = static_cast<TMember*>(P4->Items[i]);
//TPoly* P5 = P3->MulNum(M );
//M = static_cast<TMember*>(Poly->Items[0]);
//cout <<M->Member.c_str()<<endl;
//Poly->_Add(new TMember(6,6));
//M = static_cast<TMember*>(Poly->Items[1]);
cin >> i;
return 0;
}
//-----

```

### Порядок выполнения

В режиме консольного приложения

- опишите класс TMember, реализуйте класс и протестируйте каждый метод;
- опишите класс TPoly, реализуйте класс и протестируйте каждый метод.

Тестовые наборы поместите в таблицу 3 следующего вида:

**Таблица 3.** Тестовый набор для тестирования операции «Умножить» на типе «Полином»

Тестовый набор для тестирования операции «Умножить»				
Номер теста	Исходные данные		Ожидаемый результат	
	Вход	Полином	Возвращаемое значение	Полином
1	$0 \cdot X^0$	$0 \cdot X^0$	$0 \cdot X^0$	$0 \cdot X^0$
2	$0 \cdot X^0$	$1 \cdot X^0$	$0 \cdot X^0$	$1 \cdot X^0$
3	$1 \cdot X^0$	$1 \cdot X^0$	$1 \cdot X^0$	$1 \cdot X^0$
4	$1 \cdot X^0$	$2 \cdot X^1$	$2 \cdot X^1$	$2 \cdot X^1$
5	$1 \cdot X^0$	$2 \cdot X^1 + 3 \cdot X^2$	$2 \cdot X^1 + 3 \cdot X^2$	$2 \cdot X^1 + 3 \cdot X^2$
6	$1 \cdot X^0 + 1 \cdot X^1$	$1 \cdot X^0 - 1 \cdot X^1$	$1 \cdot X^0 - 1 \cdot X^2$	$1 \cdot X^0 - 1 \cdot X^1$

## **Содержание отчета**

1. Задание.
2. Текст программы на Object Pascal, C++ Builder.
3. Тестовые наборы данных для тестирования типа данных.

## **Контрольные вопросы**

1. Чем определяется размер памяти, выделяемой под экземпляр класса?
2. Что такое RTTI класса?
3. Как и когда происходит связывание объекта с RTTI класса?
4. Как описываются и переопределяются виртуальные и динамические методы?
5. Что такое раннее связывание, и для каких методов оно выполняется?
6. Что такое позднее связывание, и для каких методов оно выполняется?
7. Когда для класса необходимо описать собственный деструктор?