Министерство Образования Российской Федерации

СибГУТИ

Современные технологии программирования 2.
Методические указания к выполнению практических работ

Новосибирск

2018

Практикум содержит методические указания к выполнению практических работ по курсу Современные технологии разработки программирования 2.

Составитель: канд.ф.-м.наук, доц. М.Г.Зайцев

Рецензент: канд.техн.наук, доц. В.Г.Кобылянский

Работа подготовлена кафедрой прикладной математики и кибернетики

СибГУТИ, 2018 г.

Содержание

Введение

Целями данного практикума является формирование практических навыков:

- *проектирования программ в технологии «объектно-ориентированного программирования»;*
- *реализации объектно-ориентированных проектов с помощью классов C++;*
- *использования библиотеки визуальных компонентов для построения интерфейса.*

Результатом выполнения предлагаемых в практикуме лабораторных работ станет приложение под Windows «Калькулятор универсальный».

Задание

1. Разработайте Универсальный калькулятор с интерфейсом в стиле Windows, который позволил бы вычислять выражения с p-ичными числами, простыми дробями, комплексными числами.
2. Калькулятор необходимо снабдить системой справочной.
3. Для установки калькулятора необходимо создать инсталлятор.

Спецификация

1. Калькулятор должен обеспечить вычисление выражений в одном из режимов:
   - p-ичные числа,
   - комплексные числа,
   - рациональными дроби.
2. Остальные требования: работа с памятью, работа с буфером обмена, прецеденты использования те же, что и для калькуляторов p-ичных чисел, простых дробей и комплексных чисел [1].

Рекомендации к выполнению

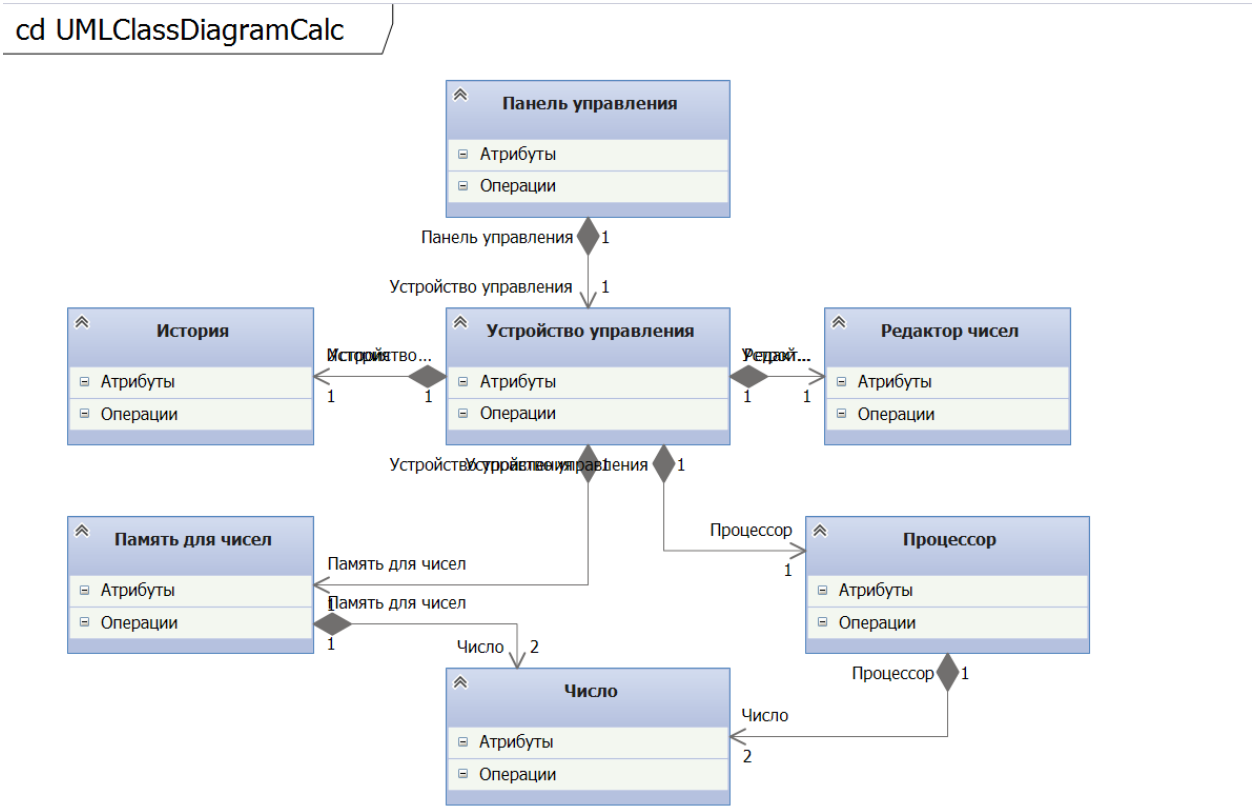
Реализуйте универсальный калькулятор, используя результаты предыдущих лабораторных работ [1]. Разработку выполните в следующем порядке:

1. Разработайте иерархию классов TANumber (число), в основу которой положите абстрактный класс TANumber, от него должны будут наследовать действительные числа TPNumber(p-ичные числа с основанием 2..16), простые дроби TFrac и комплексные числа TComp; в классе TANumber объявите общие для всех приведённых типов чисел операции, как абстрактные методы;
2. Разработайте абстрактный тип данных TMemory (память), которая могла бы хранить и обрабатывать действительные числа TPNumber(p-ичные числа с основанием 2..16), простые дроби TFrac и комплексные

числа TComp; для этого тип хранимого числа определите как TANumber;

3. Разработайте абстрактный тип данных TProc (процессор), который мог бы обрабатывать действительные числа TPNumber(p-ичные числа с основанием 2..16), простые дроби TFrac и комплексные числа TComp; для этого тип обрабатываемого числа определите как TANumber;
4. Разработайте иерархию классов TAEditor (редактор), в основу которой положите абстрактный класс TAEditor, от него должны будут наследовать редакторы действительных чисел TREditor(p-ичные числа с основанием 2..16), простых дробей TFEditor и комплексных чисел TCEditor;
5. Разработайте класс TCtrl (управление калькулятором);
6. Разработайте TClcPnl (класс интерфейс калькулятора).
7. Выполните тестирование приложения «универсальный калькулятор».

Диаграмма классов UML для калькулятора представлена на рисунке.



Практическая работа. Иерархия классов «Число»

Тема: технология ООП

Цель: Сформировать практические навыки реализации иерархии классов средствами объектно-ориентированного программирования языка C++.

Задание

1. Разработать и реализовать иерархию классов «Число», обеспечивающую выполнение операций над p-ичными числами, простыми дробями и комплексными числами.

На унифицированном языке моделирования UML (Unified Modeling Language) диаграмма класса TANumber (Число) выглядит следующим образом:

TANumber (Число)
строка: String (свойство) числоЕстьНоль(B: TANumber): Boolean; virtual; копировать: TANumber; ; virtual; сложить (B: TANumber): TANumber; virtual; вычесть (B: TANumber): TANumber; virtual; перемножить (B: TANumber): TANumber; virtual; поделить (B: TANumber): TANumber; virtual; равенствоЧисел (B: TANumber): Boolean; virtual; квадрат: TANumber; virtual; обратное: TANumber; virtual; читатьЧислоВформатеСтроки: String; virtual; (метод свойства) писатьЧислоВформатеСтроки(a: String) ; virtual; (метод свойства)
Обязанность: выполнение арифметических операций над действительными числами, простыми дробями и комплексными числами

2. Класс должен отвечать за выполнение следующих операций над действительными числами, простыми дробями и комплексными числами:
   - сложение;
   - вычитание;
   - умножение;
   - деление;
   - эквивалентность двух чисел;
   - равенство числа нулю;
   - запись строкового представления числа;
   - чтение строкового представления числа;
   - создание копии числа

3. Иерархия классов «Число» представлена на (Рис. 1, Рис. 2. Рис. 3). Они отличаются тем, что первая иерархия не обеспечивает повторного использования кода, в отличие от второй. Здесь TANumber – число, TFrac – постая дробь, TPNumber – p-ичное число, TComplex – комплексное число. Третья иерархия обеспечивает работу со всеми типами чисел в системе счисления с выбранным основанием.
4. Протестировать каждый класс иерархии.

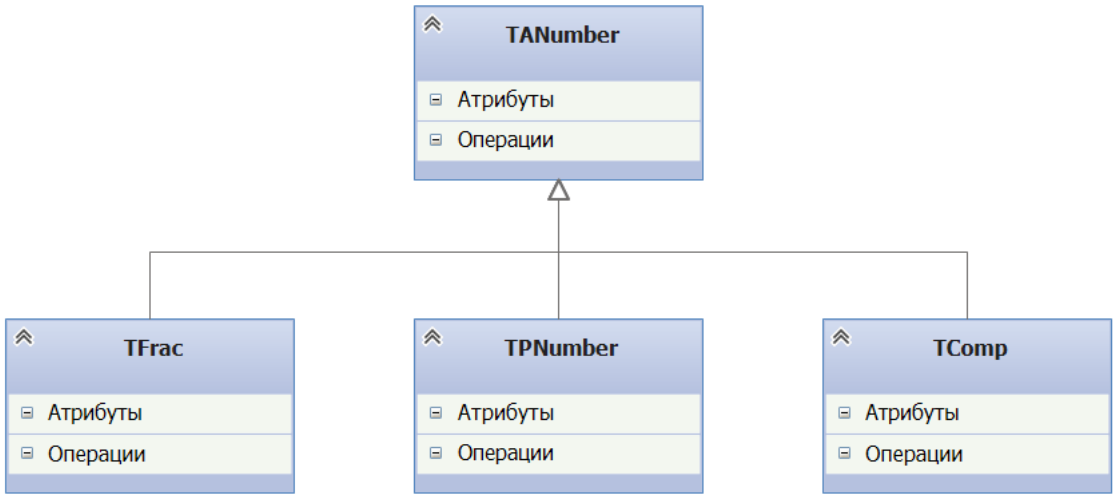


Рис. 1. Иерархия классов «число» без повторного использования кода

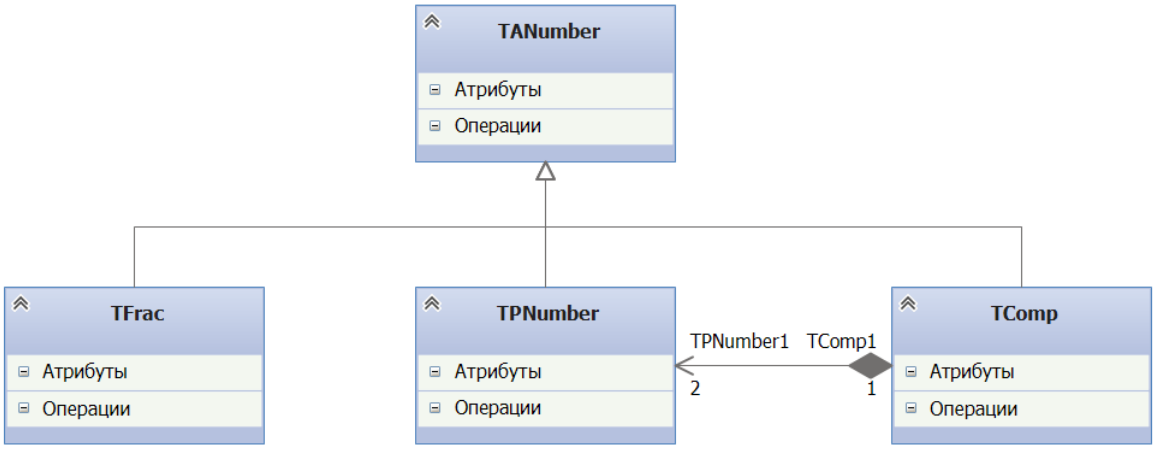


Рис. 2. Иерархия классов «число» с повторным использованием кода

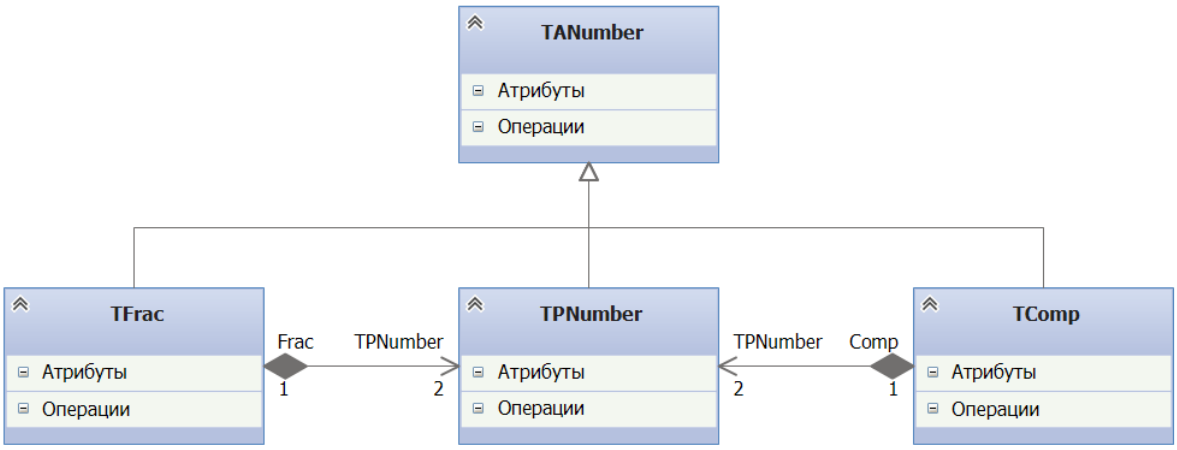


Рис. 3. Иерархия классов «число» для работы с числами в системе счисления с выбранным основанием.

Рекомендации к выполнению

1. В классе TANumber опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление числа.
2. В классе TANumber опишите следующие операции:
   - «числоЕстьНоль», операция возвращает булевское значение True, если число равного 0, False – в противном случае;
   - «равенствоЧисел», операция возвращает булевское значение True, если число равного заданному, False – в противном случае;
   - «сложить», операция возвращает число полученное сложением числа с заданным;
   - «вычесть», операция возвращает число полученное вычитанием из числа заданного;
   - «сложить», операция возвращает число полученное сложением числа с заданным;
   - «перемножить», операция возвращает число полученное перемножением числа на заданное;
   - «поделить», операция возвращает число полученное делением числа на заданное;
   - «квадрат», операция возвращает квадрат числа;
   - «обратное», операция возвращает обратное число;
   - «читатьЧислоВформатеСтроки» - строкового типа (метод свойства), возвращает значение числа в формате строки;
   - «писатьЧислоВформатеСтроки», получает значение строкового типа (метод свойства) и заносит его в «число»;
3. Классы иерархии реализуйте в отдельном модуле UANumber. В разделе описания констант опишите следующие константы:
   - «разделитель целой и дробной частей» строкового типа;
   - «строковое представление нуля» строкового типа.

Ниже представлены примеры объявления классов на C++ и C#. В примере 1 на C++ реализован вариант, представленный на рис.1. В примере 2 на C# реализован вариант, представленный на рис. 3.

Пример 1. Объявление иерархии классов "число" на C++

```
//----Пример реализации иерархии классов «число» на C++ Builder
//---------------------------------------------------
#include <iostream.h>
#include <vcl.h>

#ifndef UNumberH
#define UNumberH
//---------------------------------------------------
//-----------------------------------абстрактный класс число
//---------------------------------------------------
const AnsiString  cCSeparator = " i* ";
class TANumber
{
   virtual AnsiString GetNumber(void) const = 0;
   virtual void SetNumber(AnsiString newn)  = 0;
 public:
   virtual TANumber* Copy() = 0;
   virtual const TANumber& operator = (TANumber& B) = 0;
   virtual TANumber& operator + (TANumber& B) = 0;
   virtual TANumber& operator - (TANumber& B) = 0;
   virtual TANumber& operator * (TANumber& B) = 0;
   virtual TANumber& operator / (TANumber& B) = 0;
   virtual bool     operator ==(TANumber& B) = 0;
   virtual TANumber& operator - (void)       = 0;
   virtual TANumber& Sqr(void)              = 0;
   virtual TANumber& Rev(void)              = 0;
   virtual bool EqZero(void)               = 0;
   __property AnsiString Number = {read = GetNumber, write = SetNumber};
};
//---------------------------------------------------
//-----------------------------------класс простых дробей
class TFrac : public TANumber
{
   long Numerator,Denominator;//числитель и знаменатель
   virtual void SetNumber(AnsiString newn);
   void Reduce(void);//сократить дробь
   void Sign(void);//знак дроби поместить перед числителем
   long Gcd(long Nr,long Dr);//наибольший общий делитель
   virtual AnsiString GetNumber(void) const ;
 public:
```

```
   TFrac(long Nr = 0,long Dr = 1);//конструктор
   TFrac(AnsiString S);//конструктор
   TFrac(TFrac& C);//конструктор копирования
   TANumber* Copy();
         TFrac&  operator + (TANumber& B);
   virtual TFrac&  operator - (TANumber& B);
   virtual TFrac&  operator * (TANumber& B);
   virtual TFrac&  operator / (TANumber& B);
   virtual bool    operator == (TANumber& B);
   virtual const TFrac&  operator = (TANumber& B);
   virtual TFrac&  operator - (void);
   virtual TFrac&  Sqr(void);
   virtual TANumber&  Rev(void);
   virtual bool EqZero(void);
};
//-------------------------------------------------------
//---------------------------------------класс вещественных чисел
class : public TANumber
{
   double Num,Base,Cor;
   virtual void SetNumber(AnsiString newn);
   virtual AnsiString GetNumber(void) const ;
 public:
   TPNumber (double Nr = 0);
   TPNumber (AnsiString S);
   TPNumber (TPNumber & C);
   TANumber* Copy();
   virtual const TPNumber &  operator = (TANumber& B);
   virtual TPNumber &  operator + (TANumber& B);
   virtual TPNumber &  operator - (TANumber& B);
   virtual TPNumber &  operator * (TANumber& B);
   virtual TPNumber &  operator / (TANumber& B);
   virtual bool    operator == (TANumber& B);
   virtual TPNumber &  operator - (void);
   virtual TPNumber &  Sqr(void);
   virtual TPNumber &  Rev(void);
   virtual bool    EqZero(void);
};
//-------------------------------------------------------
//-----------------------------------------класс комплексных чисел
class TComp : public TANumber
{
   TPNumber  Re,Im;
   virtual void SetNumber(AnsiString newn);
```

```
  virtual AnsiString GetNumber(void) const ;
 public:
  TComp(TPNumber & R,TPNumber & I);
  TComp(double R = 0,double I = 0);
  TComp(AnsiString S);
  TComp(TComp& C);
  TANumber* Copy();
  virtual const TComp&  operator = (TANumber& B);
  virtual TComp&  operator + (TANumber& B);
  virtual TComp&  operator - (TANumber& B);
  virtual TComp&  operator * (TANumber& B);
  virtual TComp&  operator / (TANumber& B);
  virtual bool    operator ==(TANumber& B);
  virtual TComp&  operator - (void);
  virtual TComp&  Sqr(void);
  virtual TComp&  Rev(void);
  virtual bool    EqZero(void);
};
#endif
```

Пример 2. C#

```
namespace Number
{
    abstract class Number
    {
        public abstract Number Add(Number b);
        public abstract Number Mul(Number b);
    }
    class PNumber: Number
    {
        public double number { get; }
        public PNumber(double n)
        {
            number = n;
        }
        public override Number Add(Number b)
        {
            double c = this.number + (b as PNumber).number;
            return new PNumber(c);
        }
        public override Number Mul(Number b)
        {
            double c = this.number * (b as PNumber).number;
            return new PNumber(c);
        }
        public override string ToString()
        {
            return number.ToString();
```

```
        }
    }
    class Frac: Number
    {
        PNumber num;
        PNumber dnom;
        public Frac(double n, double dn)
        {
            num = new PNumber(n);
            dnom = new PNumber(dn);
        }
        public override Number Add(Number b)
        {
            double c = this.num.number * (b as Frac).dnom.number +
this.dnom.number * (b as Frac).num.number;
            double d = this.num.number * (b as Frac).dnom.number;
            return new Frac(c,d);
        }
        public override Number Mul(Number b)
        {
            double c = this.num.number * (b as Frac).dnom.number +
this.dnom.number * (b as Frac).num.number;
            double d = this.num.number * (b as Frac).dnom.number;
            return new Frac(c, d);
        }
        public override string ToString()
        {
            return num+"/"+dnom;
        }
    }
    class Comp: Number
    {
        PNumber re;
        PNumber im;
        public Comp(double n, double dn)
        {
            re = new PNumber(n);
            im = new PNumber(dn);
        }
        public override Number Add(Number b)
        {
            double c = this.re.number + (b as Comp).re.number;
            double d = this.im.number * (b as Comp).im.number;
            return new Comp(c, d);
        }
        public override Number Mul(Number b)
        {
            double c = this.re.number - (b as Comp).re.number;
            double d = this.im.number * (b as Comp).im.number;
            return new Comp(c, d);
        }
```

```
        public override string ToString()
        {
            return re + "i*" + im;
        }
    }

}
```

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. Что такое отношение обобщение и как оно изображается на диаграммах UML?
2. Что такое отношение ассоциация и как оно изображается на диаграммах UML?
3. Что такое чисто виртуальная функция (абстрактный метод)?
4. Как переопределяются виртуальные методы в описании класса?
5. Что такое полиморфизм?

Практическая работа. Иерархия классов «Редактор»

Тема: технология ООП

Цель: Сформировать практические навыки реализации иерархии классов средствами объектно-ориентированного программирования языка C++.

Задание

1. Разработать и реализовать иерархию классов «Редактор», наследующих от AEditor, которая обеспечивала бы возможность редактирования p-ичных чисел, простых дробей и комплексных чисел.

На унифицированном языке моделирования UML (Unified Modeling Language) диаграмма класса AEditor выглядит следующим образом:

AEditor (Редактор)
строка: String числоЕстьНоль: Boolean добавитьЗнак: String добавитьP-ичную цифру(a: Integer): String добавитьНоль: String забойСимвола: String очистить: String читатьСтрокаВформатеСтроки: String (метод свойства) писатьСтрокаВформатеСтроки(a: String) (метод свойства) редактировать(a: Integer): String
Обязанность: ввод, хранение и редактирование строкового представления чисел

2. Класс должен отвечать за ввод и редактирование строкового представления чисел. Класс должен обеспечивать:
   - добавление символов;
   - добавление и изменение знака;
   - добавление разделителя;
   - забой символа, стоящего справа (BackSpace);
   - установку нулевого значения числа (Clear);
   - чтение строкового представления;
   - запись строкового представления;
3. Иерархия классов «Редактор» представлена на (Рис. 4, Рис. 5, Рис. 6, Рис. 7). Они отличаются тем, что первая иерархия не обеспечивает повторного использования кода, в отличие от второй. Третья и четвёртая иерархии обеспечивает работу со всеми типами чисел в системе счисления с выбранным основанием. Здесь AEditor – абстрактный редактор, PEditor – редактор p-ичных чисел, FEditor – редактор дробей, CEditor – редактор комплексных чисел.
4. Протестировать каждый класс иерархии.

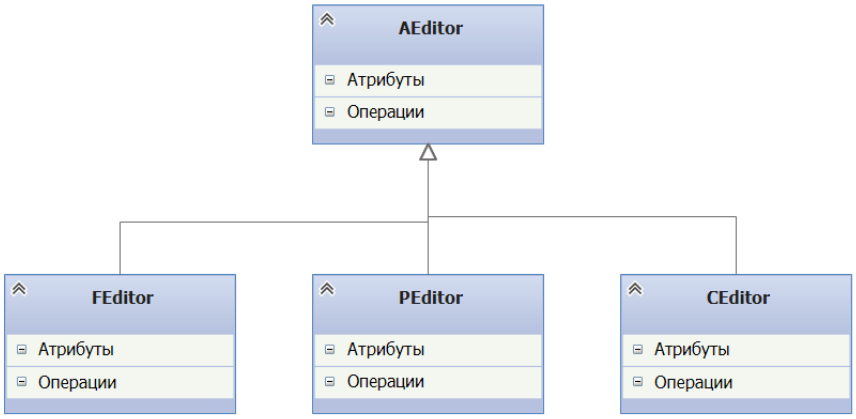


Рис. 4. Иерархия классов редактор без повторного использования кода

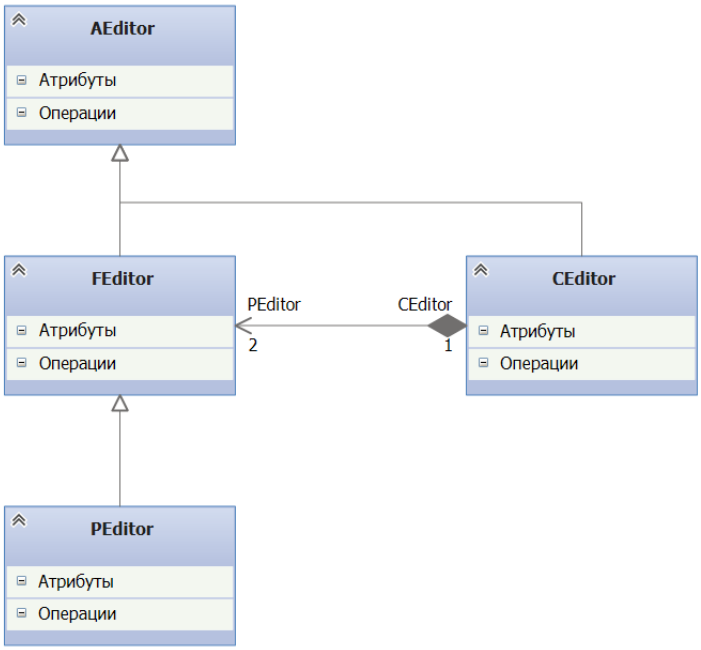


Рис. 5. Иерархия классов редактор с повторным использованием кода

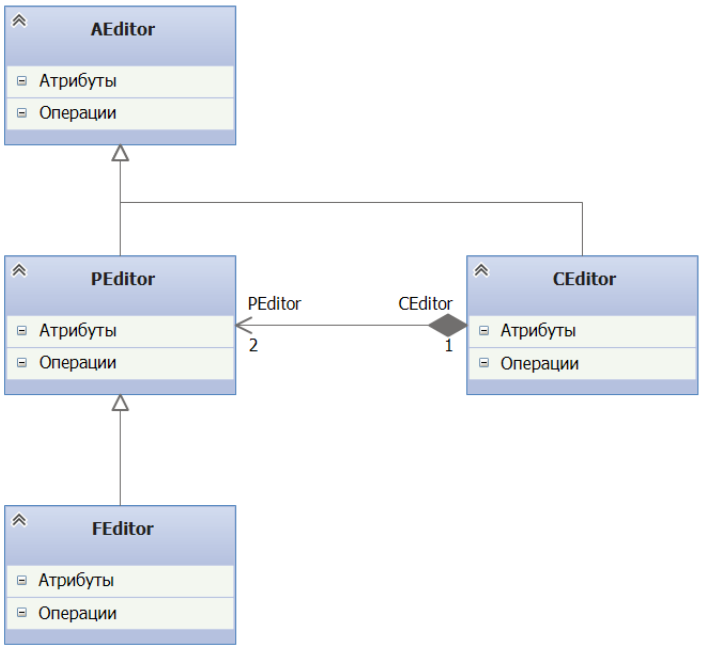


Рис. 6. Иерархия классов редактор с повторным использованием кода для работы с числами в заданной системе счисления.

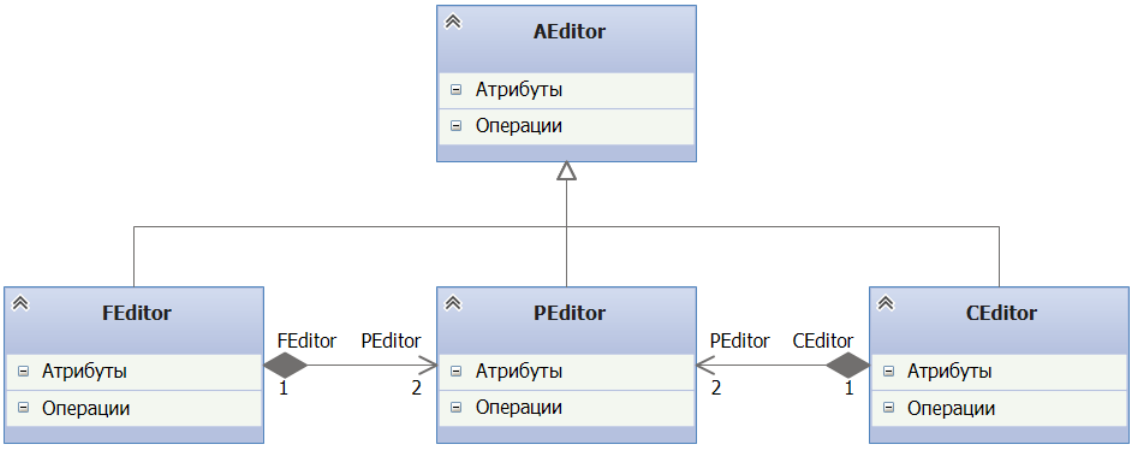


Рис. 7. Иерархия классов редактор с повторным использованием кода для работы с числами в заданной системе счисления.

Рекомендации к выполнению

1. В классе AEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемого.
2. В классе AEditor опишите следующие операции:
   - «число есть ноль», операция возвращает булевское значение True, если «строка» содержит изображение числа равного 0, False – в противном случае;

- «добавить знак», операция добавляет или удаляет знак «-» из «строка» и возвращает значение «строка»;
- «добавить цифру», операция получает целое число (числовое обозначение цифры), преобразует его в символ и добавляет к «строка», если это допускает формат, возвращает значение «строка»;
- «добавить ноль», операция добавляет ноль к «строка», если это допускает формат, возвращает значение «строка»;
- «забой символа», операция удаляет крайний правый символ «строка» и возвращает значение «строка»;
- «очистить», операция устанавливает в «строка» строку, изображающую 0, возвращает значение «строка»;
- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «читать «строка» в формате строки» - строкового типа (метод свойства), возвращает значение «строка» в заданном пользователем формате;
- «писать «строка» в формате строки», получает значение строкового типа (метод свойства) и заносит его в «строка»;

3. Классы иерархии реализуйте в отдельном модуле UAEditor. В разделе описания констант опишите следующие константы:
   - «разделитель целой и дробной частей» строкового типа;
   - «строковое представление нуля» строкового типа.

   Ниже представлены примеры объявления классов на C++.

Пример 3. Объявление иерархии классов "редактор" на C++

```
//  команды управления редактором
const unsigned cZero        = 0; //добавить цифру 0 к редактируемому числу
const unsigned cOne         = 1; //добавить цифру 1 к редактируемому числу
const unsigned cTwo         = 2; //добавить цифру 2 к редактируемому числу
const unsigned cThree       = 3; //добавить цифру 3 к редактируемому числу
const unsigned cFour        = 4; //добавить цифру 4 к редактируемому числу
const unsigned cFive        = 5; //добавить цифру 5 к редактируемому числу
const unsigned cSix         = 6; //добавить цифру 6 к редактируемому числу
const unsigned cSeven       = 7; //добавить цифру 7 к редактируемому числу
const unsigned cEight       = 8; //добавить цифру 8 к редактируемому числу
const unsigned cNine        = 9; //добавить цифру 9 к редактируемому числу
const unsigned cSign        = 10;//знак
const unsigned cSeparatorFR  = 11;//разделитель для дроби и вещественного
числа
const unsigned cSeparatorC   = 12;//разделитель для комплексного числа
const unsigned cBS          = 13;//забой символа
const unsigned CE           = 14;//очистить редактируемое число
```

```
//---------------------------------------------------------------------
//  редактор чисел - абстракный класс
class AEdit
{
  protected:
//---------------------------------------------------------------------
    String FSeparator;
//  изображение разделителя
//---------------------------------------------------------------------
    bool FSeparatorIs ;
//  наличие разделителя в числе
//---------------------------------------------------------------------
    virtual void SetNum (String n) = 0;
//  писать в поле FNumber
//---------------------------------------------------------------------
    virtual String GetNum(void) const = 0;
//  прочитать поле FNumber
//---------------------------------------------------------------------
    virtual  void AddDigitLS (int a) = 0;
//  добавить цифру слева от разделителя
//---------------------------------------------------------------------
    virtual  void AddDigitRS (int a) = 0;
//  добавить цифру справа от разделителя
//---------------------------------------------------------------------
  public:
//---------------------------------------------------------------------
    __property String Number = { read = GetNum, write = SetNum };
//  читать и писать редактируемое число в формате строки
//---------------------------------------------------------------------
    __property   bool SeparatorIs = {read = FSeparatorIs, write = FSeparatorIs};
//  читать наличие разделителя
//---------------------------------------------------------------------
      virtual  bool IsZero(void) const = 0;
//  редактируемое число равно нулю
//---------------------------------------------------------------------
      virtual  String AddSeparator (int a) = 0;
//  добавить разделитель
//---------------------------------------------------------------------
      virtual  String AddDigit (int a) = 0;
//  добавить цифру
//---------------------------------------------------------------------
      virtual  String AddSigne (int a)= 0;
//  изменить знак на противоположный
//---------------------------------------------------------------------
```

```
     virtual  String BackSpace(void) = 0;
//  удалить крайний правый символ
//---------------------------------------------------------------
     virtual  String CE(void) = 0;
//  очистить ввод: установить в редактируемое число нулевое значение
//---------------------------------------------------------------
};
//---------------------------------------------------------------
//-------------------------------  редактор простых дробей
//---------------------------------------------------------------
class FEdit : public AEdit
{
  protected:
//---------------------------------------------------------------
   String FZero;
//  изображение нуля
//---------------------------------------------------------------
   String FSign;
//  изображение знака
//---------------------------------------------------------------
   String FNumber;
//  редактируемое число; нулю соответствует пустая строка
//---------------------------------------------------------------
   virtual void SetNum (String n);
//  писать в поле FNumber
//---------------------------------------------------------------
   virtual String GetNum(void) const;
//  прочитать поле FNumber
//---------------------------------------------------------------
   virtual  void AddDigitLS (int a);
//  добавить цифру слева от разделителя
//---------------------------------------------------------------
   virtual  void AddDigitRS (int a);
//  добавить цифру справа от разделителя
//---------------------------------------------------------------
  public:
//---------------------------------------------------------------
     FEdit();
//---------------------------------------------------------------
     virtual  bool IsZero(void) const;
//  редактируемое число равно нулю
//---------------------------------------------------------------
     virtual  String AddSeparator (int a);
//  добавить разделитель
```

```
//---------------------------------------------------------------------
     virtual  String AddDigit (int a);
//  добавить цифру
//---------------------------------------------------------------------
     virtual  String AddSigne (int a);
//  изменить знак на противоположный
//---------------------------------------------------------------------
     virtual  String BackSpace(void);
//  удалить крайний правый символ
//---------------------------------------------------------------------
     virtual  String CE(void);
//  очистить ввод: редактируемое число равно нулю
//---------------------------------------------------------------------
};
//---------------------------------------------------------------------
//  редактор вещественных чисел
//  наследует от редактора простых дробей
class PEdit : public FEdit
{
  protected:
//---------------------------------------------------------------------
   virtual  void AddDigitRS (int a);
//  добавить цифру
//---------------------------------------------------------------------
  public:
//---------------------------------------------------------------------
     PEdit();
//---------------------------------------------------------------------
};
//---------------------------------------------------------------------
//  редактор комплексных чисел
class CEdit : public AEdit
{
//---------------------------------------------------------------------
   String FZero;
//  изображение нуля
//---------------------------------------------------------------------
   PEdit FRe;//редактор действительной части
//---------------------------------------------------------------------
   PEdit FIm;//редактор мнимой части
//---------------------------------------------------------------------
  protected:
//---------------------------------------------------------------------
   virtual void   SetNum (String n );
```

```
//  писать в поле FNumber
//---------------------------------------------------------------------
   virtual String GetNum(void) const;
//  прочитать поле FNumber
//---------------------------------------------------------------------
   virtual void   AddDigitLS (int a);
//  добавить цифру слева от разделителя
//---------------------------------------------------------------------
   virtual void   AddDigitRS (int a);
//  добавить цифру справа от разделителя
//---------------------------------------------------------------------
  public:
//---------------------------------------------------------------------
   virtual bool    IsZero(void) const;
//  редактируемое число равно нулю
//---------------------------------------------------------------------
   virtual String    AddDigit(int a);
//  добавить цифру в изображение комплексного числа
//---------------------------------------------------------------------
   virtual String    AddSeparator (int a = 0);
//  добавить разделитель действительной и мнимой частей
//---------------------------------------------------------------------
   virtual String    AddSigne (int a = 0);
//  изменить знак комплексного числа на противоположный
//---------------------------------------------------------------------
   virtual String    BackSpace(void);
//  удалить крайний правый символ
//---------------------------------------------------------------------
   virtual String    CE(void);
//  очистить ввод: редактируемое число равно нулю
//---------------------------------------------------------------------
   CEdit();
//---------------------------------------------------------------------
};
```

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. Что такое наследование?
2. Какие директивы используют при описании наследования и в чём их смысл?

3. Какой класс называется абстрактным?
4. Какое отношение называется агрегацией и как оно изображается на диаграммах UML?
5. Какое отношение называется композицией и как оно изображается на диаграммах UML?

Практическая работа. Абстрактный тип данных (ADT) «Память»

Тема: технология ООП

Цель: Сформировать практические навыки: реализации абстрактного типа данных с помощью классов C++.

Задание

1. В соответствии с приведенной ниже спецификацией реализовать абстрактный тип данных «память для Число», используя класс.
2. Оттестировать каждую операцию, определенную на типе данных одним из методов тестирования.
3. Оттестировать тип данных в целом.

Спецификация типа данных «память для Число».

ADT TMemory

Данные

Память для Число(тип TMemory, в дальнейшем - память) - это память для хранения указателя (ссылки) на объект «Число» (тип ANumber) и значения «состояние памяти». Объект «память для Число» - изменяемый. Он имеет два состояния, обозначаемых значениями: «Включена» (_On), «Выключена» (_Off). Её изменяют операции: Записать (Store), Добавить (Add), Очистить (Clear).

Операции

Конструктор	
Начальные значения:	Указатель (ссылка) на Число (тип ANumber со значением 0)
Процесс:	Создаёт объект «память» типа TMemory. Инициирует поле Mem (тип ANumber) объекта «память» (тип TMemory) указателем на копию объекта «Число» (тип TPNumber), полученным через параметр. Память устанавливается в состояние «Выключена», в поле «состояние памяти» заносится значение (_Off).
Записать	
Вход:	(E) – указатель (ссылка) на объект «Число» (тип ANumber).
Предусловия:	Нет.
Процесс:	В объект «память» (тип TMemory) в поле указатель (ссылка) на «Число» записывается копия объекта (E). Память

	устанавливается в состояние «Включена», в поле «состояние памяти» заносится значение (_On).
Выход:	Нет.
Постусловия:	Состояние памяти – «Включена» (_On).
Взять	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт копию объекта типа ANumber, хранящуюся в объекте «память» (тип TMemory) и возвращает ссылку (указатель) на неё.
Выход:	Ссылка (указатель) на объект типа ANumber.
Постусловия:	Состояние памяти – «Включена» (_On).
Добавить	
Вход:	(E) – ссылка (указатель) на Число (объект типа ANumber).
Предусловия:	Нет.
Процесс:	В поле ссылка (указатель) на «Число» объекта «память» (тип TMemory) записывается объект «Число» (тип ANumber), полученный в результате сложения «Числа» по ссылка (указателю) (E) и «Числа», ссылка (указатель) на которое хранится в памяти.
Выход:	Нет.
Постусловия:	Состояние памяти – «Включена» (_On).
Очистить	
Вход:	Нет.
Предусловия:	Состояние памяти – «Включена» (_On).
Процесс:	В поле ссылка (указатель) на «Число» объекта «память» (тип TMemory) записывается объект «Число» (тип ANumber), инициализированный значением 0. Память устанавливается в состояние «Выключена» (_Off).
Выход:	Нет.
Постусловия:	Состояние памяти – «Выключена» (_Off).

ЧитатьСостояниеПамяти	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Копирует и возвращает значение поля «состояние памяти» объекта «память» (тип TMemory) в формате строки.
Выход:	Значение типа AnsiString.
Постусловия:	Нет.
ЧитатьЧисло	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Возвращает значение Числа, хранящегося в памяти, в формате строки.
Выход:	Значение типа AnsiString.
Постусловия:	Нет.

end TMemory

Рекомендации к выполнению

1. Тип данных реализовать, используя класс С.
2. Ссылка (указатель) на Число храните в поле типа ANumber (см. лаб. Раб. Абстрактный тип данных - Число).
3. Для чтения состояния памяти и хранимого значения используйте свойство (property).
4. Тип данных реализовать в отдельном модуле UMemory.
5. Примеры описания класса TMemory на C++Builder и Delpi, а также диаграмма классов приведёны ниже.

Пример описания класса C++Builder

```
//---------------------------------------------------------------------
#include "UNumber.h"
#ifndef UMemoryH
#define UMemoryH
//---------------------------------------------------------------------
enum  MemoryState {_OFF,_ON};
class  TMemory
{
 private:
   MemoryState MemState;
   TANumber* Mem;
   AnsiString GetMemState(void) const;
   AnsiString GetNumber(void) const;
 public:
   TMemory&  operator +=(TANumber& n);
```

```
   void  MemStore(TANumber& n);
   TANumber&  MemRestore(void);
   void MemClear(void);
   TMemory(TANumber& n);
   virtual ~TMemory();
   __property AnsiString MemOn = { read = GetMemState} ;
   __property AnsiString Number = { read = GetNumber} ;
   friend ostream& operator << (ostream& , TMemory& d);
};
#endif
```

Ниже приведены диаграмма классов и состояний для класса память.

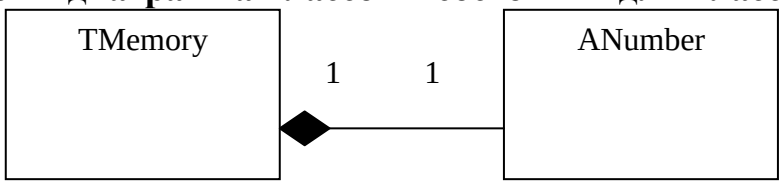


Рис. Диаграмма классов Память.

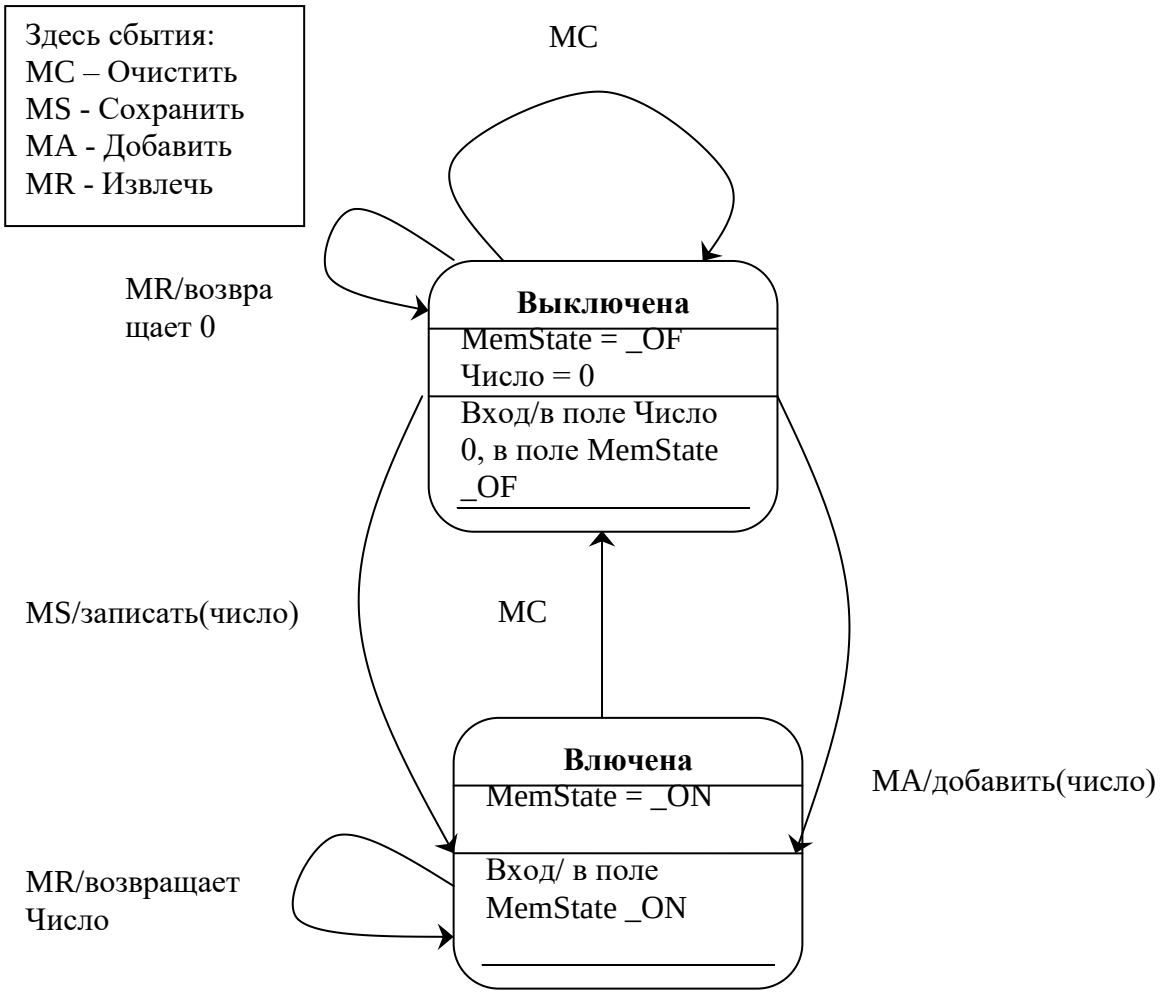


Рис. Диаграмма состояния Память

Содержание отчета

1. Задание.
2. Текст программы на языке программирования.

3. Тестовые наборы данных для тестирования типа данных.

Контрольные вопросы

1. Какое отношение называется зависимостью и как оно изображается на диаграммах UML?
2. В чём отличие между композицией и агрегацией в UML?
3. Для чего используют диаграммы состояний?
4. Перечислите основные элементы диаграммы состояний?

Практическая работа. Абстрактный тип данных: Процессор.

Тема: технология ООП

Цель: Сформировать практические навыки: реализации абстрактного типа данных с помощью классов C++.

Задание

1. В соответствии с приведенной ниже спецификацией реализовать абстрактный тип данных «Процессор», используя класс.
2. Оттестировать каждую операцию, определенную на типе данных одним из методов тестирования.
3. Оттестировать тип данных в целом.

Спецификация типа данных «Процессор».

ADT TProc

Данные

Процессор (тип TProc) выполняет двухоперандные операции TOprtn = (None, Add, Sub, Mul, Dvd) и однооперандные операции - функции TFunc = (Rev, Sqr). Если операция или функция не может быть выполнена, в поле Error типа String заносится сообщение об ошибке. Левый операнд и результат операции хранится в поле Lop_Res, правый - в поле Rop. Оба поля имеют тип TANumber. Процессор может находиться в состоянии «операция установлена»: поле Operation не равно None (значение типа TOprtn) или в состоянии операция не установлена: поле Operation = None. Значения типа TProc - изменяемые. Они изменяются операциями: Сброс операции (OprtnClear), Выполнить операцию (OprtnRun), Вычислить функцию (FuncRun), Установить операцию (OprtnSet), Установить левый операнд (Lop_Res_Set), Установить правый операнд (Rop_Set), Сброс калькулятора (ReSet).

Операции

Конструктор	
Начальные значения:	два указателя TANumber* L, TANumber* R
Процесс:	Создаёт объект Процессор типа TProc. Поля Lop_Res, Rop инициализируются копиями объектов (L, R). В поле Error заносится пустая строка. Процессор устанавливается в состояние: «операция не установлена»: (Operation = None).
Сброс процессора	
Вход:	Нет.

Предусловия:	Нет.
Процесс:	Поля объекта процессор: Lop_Res, Rop инициализируются объектами текущего типа со значением 0. В поле Error заносится пустая строка. Процессор устанавливается в состояние: «операция не установлена»: (Operation = None).
Выход:	Нет.
Постусловия:	Состояние процессора – «операция сброшена» (Operation = None).
Сброс операции	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Процессор устанавливается в состояние: «операция не установлена»: (Operation = None).
Выход:	Нет.
Постусловия:	Состояние процессора – «операция сброшена» (Operation = None).
Выполнить операцию	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Вызывает выполнение текущей операции (записанной в поле Operation). Операция (Operation) выполняется над значениями, хранящимися в полях Rop и Lop_Res. Результат сохраняется в поле Lop_Res. Если Operation = None, никакие действия не выполняются. Если операция не может быть выполнена, в поле Error заносится сообщение об ошибке. Состояние объекта не изменяется.
Выход:	Нет.
Постусловия:	Изменяется состояние поля Lop_Res (если текущая операция выполнена) или Error – в противном случае.
Вычислить функцию	
Вход:	Вид функции (Func: TFunc).
Предусловия:	Нет.
Процесс:	Вызывает выполнение текущей функции (Func). Функция (Func) выполняется над

	значением, хранящимся в поле Rop. Результат сохраняется в нём же. Если операция не может быть выполнена, в поле Error заносится сообщение об ошибке. Состояние объекта не изменяется.
Выход:	Нет
Постусловия:	Изменяется состояние поля Rop (если текущая операция выполнена) или Error – в противном случае.
Читать левый операнд	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт копию Lop_Res и возвращает указатель на неё.
Выход:	Указатель на Lop_Res.
Постусловия:	Состояние процессора не изменяется.
Записать левый операнд	
Вход:	Переменная (Operand) типа TANumber.
Предусловия:	Нет.
Процесс:	Создаёт копию Operand и заносит указатель на неё в поле Lop_Res.
Выход:	Нет.
Постусловия:	Значение поля Lop_Res меняется на значение (Operand).
Читать правый операнд	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Создаёт копию Rop и возвращает указатель на неёё.
Выход:	Указатель на Rop.
Постусловия:	Состояние процессора не меняется.
Записать правый операнд	
Вход:	Переменная (Operand) типа TANumber.
Предусловия:	Нет.
Процесс:	Создаёт копию Operand и заносит указатель на неё в поле Rop.

Выход:	Нет.
Постусловия:	Значение поля Rop меняется на значение (Operand).
Читать состояние	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Копирует и возвращает значение поля Operation.
Выход:	Значение поля Operation.
Постусловия:	Состояние процессора не изменяется.
Записать состояние	
Вход:	Переменная (Oprtn) типа TOprtn.
Предусловия:	Нет.
Процесс:	Заносит значение Oprtn в поле Operation.
Выход:	Нет.
Постусловия:	Значение поля Operation изменяется на Oprtn.
Читать ошибку	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Копирует и возвращает значение поля Error.
Выход:	Значение поля Error.
Постусловия:	Состояние процессора не изменяется.
Сброс ошибки	
Вход:	Нет.
Предусловия:	Нет.
Процесс:	Очищает поле Error.
Выход:	Нет.
Постусловия:	Изменяется значение поля Error.

Рекомендации к выполнению

1. Тип данных TProc реализовать, используя класс Object Pascal,C++Builder.
2. Число храните как поле типа TANumber (см. лаб. Раб. Абстрактный тип данных - Числа).
3. Для чтения состояния процессора, полей: «левый операнд-результат» (Lop_Res), «правый операнд» (Rop), «ошибка» ()Error используйте свойство (property).
4. Тип данных реализовать в отдельном модуле UProcssr.

5. В приведённой ниже таблице показана последовательность изменения состояния процессора при вычислении выражения:

$2 + 3 * 4^2$

Шаг	Вход	Метод	Rop	Lop_Res	Operation
0		Create	0	0	None
1	2		0	0	None
2	+	Lop_Res_Set; OprtnSet	0	2	Add
3	3		0	2	Add
4	*	Rop_Set; OprtnRun; OprtnSet;	3	2+3	Mul
5	4		4	2+3	Mul
6	Sqr	Rop_Set; FuncRun	4^2	2+3	Mul
7	=	OprtnRun	4^2	2+3* 4^2	Mul
8	C	ReSet	0	0	None

6. Пример описания класса TProc на Delphi и C++Builder приведены ниже:

Пример на C++Builder

```
//---------------------------------------------------------------------

#ifndef UProcssrH
#define UProcssrH
#include "UNumber.h"
//---------------------------------------------------------------------
//---------------------------------------------------------------------
typedef enum {None, Add, Sub, Mul, Dvd} TOprtn;//двухоперандные операции
//---------------------------------------------------------------------------
typedef enum  {Rev, Sqr} TFunc;//однооперандные операции - функции
//---------------------------------------------------------------------------
class TProc
{
  private:
//    AnsiString FError ;
    TOprtn Oprtn;//установленная операция
    TANumber* Fropd;//правый операнд lopd_Res <- lopd_Res + rOpd
    TANumber* Flopd_Res;//левый операнд и результат операции
    void SetROpnd(TANumber* r);//занести операнд 2
    void SetLOpnd(TANumber* r);//занести операнд 1
//    void ClearError( AnsiString r = "");//очистить поле FError
    void OprtnClear(void);
    TANumber* GetRopd(void);
    TANumber* GetLopd_Res(void);
  public:
```

```
    __property TANumber* ropd = {read = GetRopd, write = SetROpnd};
    //правый операнд lopd_Res <- lopd_Res + rOprnd
    __property TANumber* lopd_Res = { read = GetLopd_Res, write = SetLOpnd};
    //левый операнд и результат операции
    __property TOprtn Operation = {read = Oprtn, write = Oprtn};
//    __property AnsiString Error = {read = FError, write = ClearError};
    void OprtnRun();//выполнить операцию
    void FuncRun(TFunc FuncNew);//выполнить функцию
    void ReSet();//установить начальное состояние
    TProc(TANumber* L, TANumber* R);
    ~TProc();
};
#endif
//---------------------------------------------------------------------
```

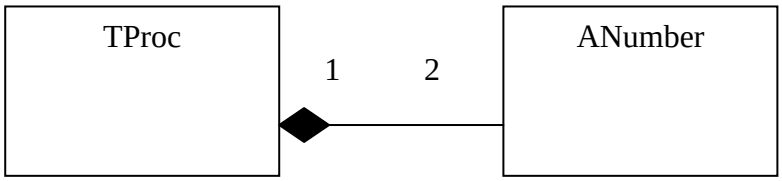


Рис. Диаграмма классов Процессор.

Содержание отчета

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

Контрольные вопросы

1. Как отношение зависимости может быть реализовано в языке программирования?
2. Как отношение агрегации может быть реализовано в языке программирования?
3. Как отношение композиции может быть реализовано в языке программирования?
4. Как отношение ассоциации может быть реализовано в языке программирования?
5. Что означает мощность отношения между классами и как она изображается на языке UML?

Практическая работа. Управление калькулятором

Тема: технология ООП

Цель

Сформировать практические навыки реализации классов средствами объектно-ориентированного программирования C++.

Задание

1. Разработать и реализовать класс «Управление универсальным калькулятором» тип TCtrl, используя класс C++.

На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно обозначить следующим образом:

УправлениеКалькуляторомПростыхДробей (тип TCtrl)	
состояниеКалькулятора:	TCtrlState
редактор:	TEditor
процессор:	TProc
память:	TMemory
число:	TANumber
выполнитьКомандуКалькулятора(a: Integer; var b, MState: String): String выполнитьКомандуРедактора(a: Integer): String выполнитьОперацию(a: Integer): String выполнитьФункцию(a: Integer): String вычислитьВыражение(a: Integer): String установитьНачальноеСостояниеКалькулятора(a: Integer): String выполнитьКомандуюПамяти(a: Integer; var MState: String): String читатьПисатьСостояниеКалькулятора: TCtrlState выполнитьКомандуБуфераОбмена(a: Integer; var b: String): String	

конструктор деструктор
Обязанность: управление выполнением команд калькулятора

2. Класс должен отвечать за управление выполнением команд калькулятора. Он распределяет команды калькулятора между объектами («редактор», «процессор», «память», «буфер обмена»), которые должны эти команды выполнять.
3. Протестировать каждый метод класса.

Рекомендации к выполнению

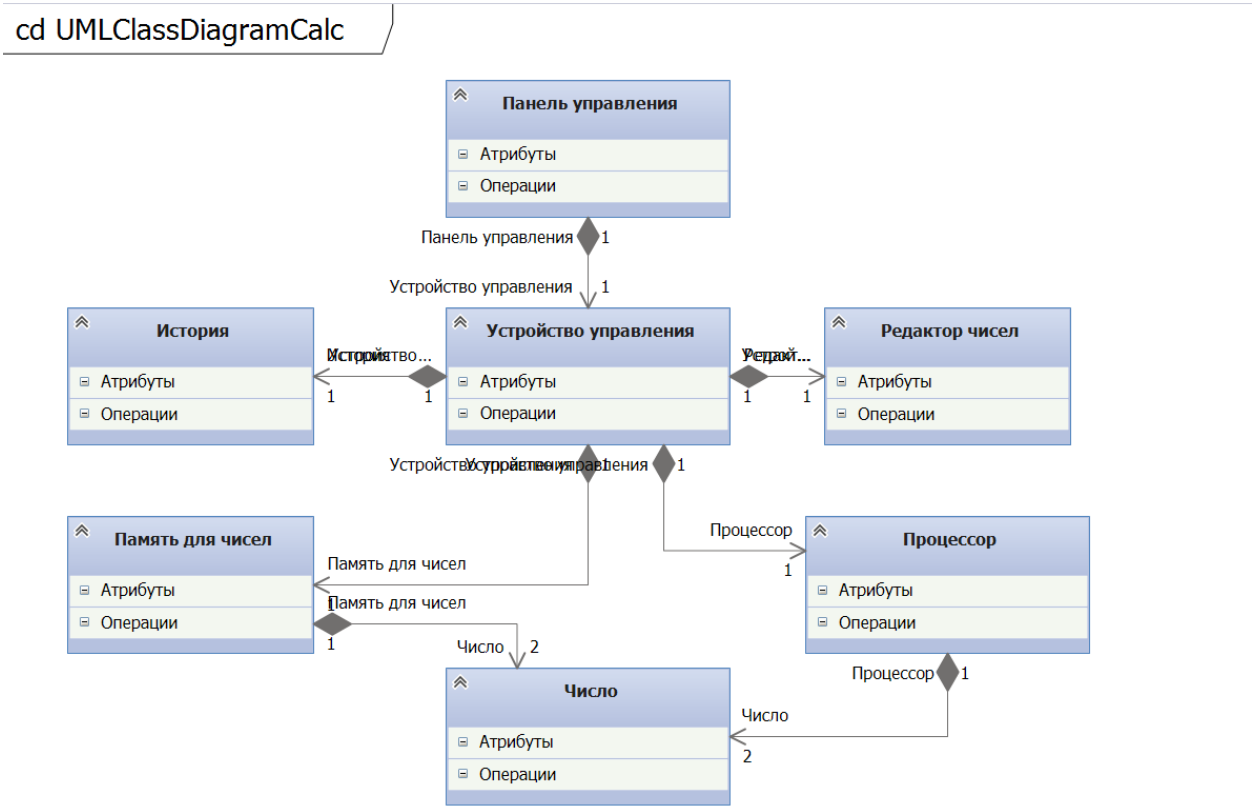
1. Класс TCtrl реализуйте в отдельном модуле UControl.
2. В модуле опишите перечисляемый тип TCtrlState = (cStart, cEditing, FunDone, cValDone, cExpDone, cOpChange, cError) для обозначения состояний калькулятора: cStart (Начальное), cEditing (Ввод и редактирование), cExpDone (Выражение вычислено), cOpDone (Операция выполнена), cValDone (Значение введено), cOpChange (Операция изменена),. cError (Ошибка).
3. В классе опишите следующие атрибуты:
   - «редактор» - объект типа TEditior;
   - «процессор» - объект типа TProc (TCalc в предыдущей редакции);
   - «память» - объект типа TMemory;
   - «состояние калькулятора» - тип TCtrlState;
   - «число» - объект типа TANumber (результат выполнения последней команды).
4. Набор основных операций класса определяется набором команд калькулятора, заданных пользователем. Кроме того, в классе будут определены вспомогательные операции, обеспечивающие реализацию основных операций. В классе опишите следующие операции:

- «выполнитьКомандуКалькулятора» (управляет вызовом операций по работе с объектами: редактор (поле TEditor), процессор (поле TProc), память (поле TMemory), буфер обмена (глобальный объект ClipBoard)), операция получает целое число (номер команды пользователя), строку для буфера обмена, строку со значением состояния памяти и возвращает строку для буфера обмена, строку состояния памяти и строку результата;
- «выполнитьКомандуРедактора» (управляет вызовом методов объекта редактор (тип TEditor)), операция получает целое число (номер команды пользователя и возвращает строку результата;
- «выполнитьОперацию» (управляет вызовом методов объекта процессор (поле TProc)), операция получает целое число (номер команды пользователя и возвращает строку результата;
- «выполнитьФункцию» (управляет вызовом методов объекта процессор (поле TProc)), операция получает целое число (номер команды пользователя и возвращает строку результата;
- «вычислитьВыражение» (управляет вызовом методов объекта процессор (поле TProc)), операция получает целое число (номер команды пользователя и возвращает строку результата;
- «установитьНачальноеСостояниеКалькулятора» (управляет вызовом методов для перевода объекта типа TCalc в состояние **Start** (см. ниже), операция получает целое число (номер команды пользователя и возвращает строку результата;
- «выполнитьКомандуПамяти» (управляет вызовом методов объекта типа **TCtrl**, обеспечивающих выполнение команд памяти), операция получает целое число (номер команды пользователя), строку со значением состояния памяти и возвращает строку состояния памяти и строку результата;

- «выполнитьКомандуБуфераОбмена» (управляет вызовом методов объекта типа **TClipBoard**, обеспечивающих выполнение команд буфера обмена), операция получает целое число (номер команды пользователя), строку со значением буфера обмена и возвращает строку со значением буфера обмена и строку результата;
- «читать | писать состояние калькулятора», возвращает значение типа TCtrlState (свойство, опирающееся на поле);
- «конструктор», осуществляет создание объектов и инициализацию полей класса;
- «деструктор», осуществляет освобождение памяти, занимаемой объектом класса и объектами, указатели на которые хранятся в полях объекта: «Редактор», «Процессор», «Память», «Число».

5. Логика работы объекта «управление калькулятором» класс TCtrl может быть описана с помощью таблицы переходов, которая отражает изменение состояния калькулятора и результат работы объекта под действием команд пользователя. Таблица переходов строится на основе анализа прецедентов (вариантов использования), приведённых в спецификации. Для построения таблицы переходов необходимо:
    - проанализировать спецификацию, приведённую в задании для калькулятора простых дробей и выделить состояния калькулятора, например: Start (Начальное), Editing (Ввод и редактирование), ExpDone (Выражение вычислено), FunDone (Функция выполнена), ValDone (Значение введено), OpChange (смена операрации), Error (ошибка);
    - построить диаграмму состояний.

Диаграмма классов для Управления калькулятором представлена на рисунке ниже.



Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. В чём сущность отношения ассоциации между классами?
2. Как изображается отношение ассоциации между классами на языке UML?
3. В чём сущность отношения агрегации между классами?
4. Как изображается отношение агрегации между классами на языке UML?
5. В чём сущность отношения композиции между классами?
6. Как изображается отношение композиции между классами на языке UML?

Лабораторная работа. Интерфейс

Тема: технология ООП

Цель

Сформировать практические навыки реализации классов средствами объектно-ориентированного программирования C++.

Задание

1. Разработать и реализовать класс «Интерфейс универсального калькулятора» тип TClcPnl наследник TForm, используя C++.

На Унифицированном языке моделирования UML (Unified Modeling Language) наш класс можно обозначить следующим образом:

ИнтерфейсКалькулятораПростыхДробей	
строкаЧисло:	TStaticText
состояниеПамяти:	TStaticText
кнопки ввода:	TBitButton
FormCreate(Sender: TObject)	
ButtonClick(Sender: TObject)	
FormKeyPress(Sender: TObject; var Key: Char)	
Методы для обработки команд меню	
Обязанность:	
Обеспечить пользователю возможность управления калькулятором через клавиатуру и командные кнопки для выполнения вычислений	

2. Класс должен отвечать:

   2.1.за ввод:

   - команд редактирования,
   - команд памяти,
   - команд процессора;

   2.2.отображение:

   - вводимого числа,

- результата вычисления,
- состояния памяти;

2.3. класс должен обеспечить возможность:

- ввода перечисленных команд с помощью командных кнопок и клавиатуры;
- выполнение команд для работы с буфером обмена:
  - копировать,
  - вставить;
- настройки на в зависимости от варианта- типа чисел, обрабатываемых калькулятором.

3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. Класс TClcPnl реализуйте в отдельном модуле UClcPnl.
2. Панель управления реализуйте как форму.
3. В классе формы используйте следующие визуальные компоненты:
   - для отображения строки - простых дробей и состояния памяти- компоненты типа TStaticText;
   - для ввода символов и выполняемых операций - компоненты типа TBitButton;
   - для выбора команд при работе с буфером обмена, настройки параметра режима работы (простая дробь, р-ичное, комплексное), вызова справки вставьте главное меню: Правка с подменю: Копировать, Вставить; Вид с подменю: Целое, Целое и дробь; Справка – компонент класса TMainMenu.
4. В классе формы опишите следующие событийные процедуры:
   - «создание формы» CreateForm для создания объекта TClcCtrl и инициализации компонента отображения строки ввода/вывода;

- «нажатие кнопки» (ButtonClick) - для преобразования нажатия кнопки в соответствующее целое число и вызова метода «выполнить команду калькулятора» объекта TClcCtrl;
- «нажатие клавиши на клавиатуре» (FormKeyPress) - для преобразования нажатия клавиши в соответствующее целое число и вызова метода «выполнить команду калькулятора» объекта TClcCtrl;
- методы для обработки команд меню.

Содержание отчета

1. Задание.
2. Текст программы.
3. Тестовые наборы данных для тестирования класса.

Контрольные вопросы

1. В чём сущность отношения зависимости между классами?
2. Как изображается отношение зависимости между классами на языке UML?
3. В чём сущность отношения обобщения между классами?
4. Как изображается отношение обобщения между классами на языке UML?
5. Назовите основные принципы построения объектно-ориентированной модели?
6. Назовите основные элементы объектно-ориентированной модели?

Литература

1. Технология разработки программных систем. Ч. 1 : методические указания к выполнению лабораторных работ по специальности 010503 "Математическое обеспечение и администрирование информационных систем", 010501 "Прикладная математика и информатика", 080801 "Прикладная информатика" / Новосиб. гос. техн. ун-т ; [сост. М. Г. Зайцев]. - Новосибирск, 2011. - 49, [2] с. : ил., табл.

2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. Учебник. — 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005. 544 с.: ил.
3. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем: Учебное пособие для вузов. СПб.: Питер, 2003. – 472с.: ил.
4. Г. Буч. Обьектно-ориентированное проектирование с примерами приложений на С++, 2-ое издание. Учебник/: Пер. с англ. - М.: Издательство Бином, СПб.: Невский Диалект, 1999. - 560с.