СибГУТИ

# Современные технологии программирования 2

Методические указания к выполнению

расчётно-графического задания

Новосибирск

2018

Практикум содержит методические указания к выполнению расчётно-графического задания по курсу Современные технологии программирования 2.

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

Работа подготовлена кафедрой прикладной математики и кибернетики

СибГУТИ, 2018 г.

18 марта 2018 г. Зайцев М.Г.

# Оглавление

***Введение***

Целями данного практикума является формирование практических навыков:

- *проектирования программ в технологии «абстрактных типов данных»;*
- *реализации абстрактных типов данных с помощью классов C++;*
- *использования библиотеки визуальных компонентов для построения интерфейса.*

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

Задание

Реализовать калькулятор для выполнения вычислений над заданными в соответствии с вариантом числами, используя классы *C++* и библиотеку визуальных компонентов для построения интерфейса.

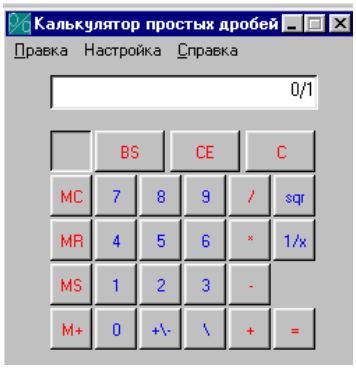
Варианты чисел:

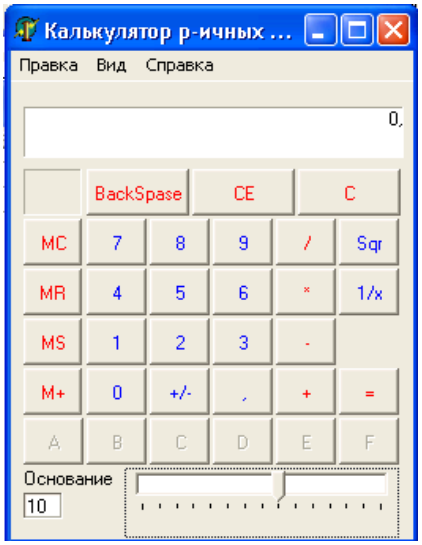
- действительные числа, представленные в системе счисления с основанием p (p-ичные числа),
- простые дроби,
- комплексные числа.

Общие требования

1. Калькулятор обеспечивает вычисление выражений с использованием операций: +, -, *. / и функций: Sqr (возведение в квадрат), Rev (1/x - вычисление обратного значения) без учёта приоритета операций. Приоритет функций одинаковый, выше приоритета операций.
2. Предусмотреть возможность ввода операндов в выражение:
   - с клавиатуры,
   - с помощью командных кнопок,
   - из буфера обмена,
   - из памяти.

3. Необходимо реализовать команду (=). которая завершает вычисление выражения. Она выполняет текущую операцию.
4. Необходимо реализовать команду C (начать вычисление нового выражения), которая устанавливает калькулятор в начальное состояние. Она сбрасывает текущую операцию и устанавливает нулевое значение для отображаемого числа и операндов.
5. Интерфейс выполнить в стиле стандартного калькулятора Windows (вид - обычный).




6. Приложение должно иметь основное окно для ввода исходных данных, операций и отображения результата и окно для вывода сведений о разработчиках приложения.
7. Основное окно должно содержать список из трёх меню:
    - Правка:

    Содержит два пункта: копировать и вставить. Эти команды используются для работы с буфером обмена;

    - Настройка:

    Содержит команды выбора режима работы приложения;

    - Справка:

    Этот команда для вызова справки о приложении.
4. Калькулятор должен обеспечивать возможность ввода исходных данных с помощью:

- командных кнопок (мышью),
- клавиатуры: цифровой и алфавитно-цифровой.

5. Вводимые числа выравнивать по правому краю.
6. Калькулятор должен быть снабжён памятью. Для работы с памятью необходимы команды:
    - MC («Очистить»),
    - MS («Сохранить»),
    - MR («Копировать»),
    - $M^+$ («Добавить к содержимому памяти»).

Память может находиться в двух состояниях, которые отображаются на панели:

- «Включена» (M). В памяти храниться занесённое значение
- «Выключена» ( ). В памяти находится ноль.

Состояние памяти меняется командами Записать и Добавить.

7. Для редактирования вводимых значений необходимы команды:
    - BackSpase (удалить крайний справа символ отображаемого числа),
    - CE (заменить отображаемое число нулевым значением)
    - Добавить символ, допустимый в изображении числа (арабские цифры, знак, разделители).
8. Снабдите компоненты интерфейса всплывающими подсказками.

Варианта 1 – «Калькулятор р-ичных чисел».

Требования.

1. Калькулятор обеспечивает работу с числами в системах счисления с основанием в диапазоне от 2 до 16.
2. Основание системы счисления – настраиваемый параметр. Настройку можно установить в основном окне или добавить в меню «Настройка».
3. Исходные числа и результат вводятся и выводятся в формате фиксированная точка

[-]<p - ичное целое без знака><разделитель>[< p - ичная дробь без знака >]

Необходимо обеспечить возможность работы в режимах:

- «целые» (вводятся только p-ичные целые числа),
- «действительные» (вводятся p-ичные числа с целой и дробной частями).

4. Кнопки для ввода цифровой информации необходимо связать с используемой системой счисления. Для пользователя необходимо сделать доступными кнопки только для ввода цифр используемой системы счисления.
5. При смене системы счисления отображаемое число должно выражаться в новой системе счисления.

**Необходимо предусмотреть следующие варианты (прецеденты) использования калькулятора:**

- Выполнение одиночных операций:
  «операнд1» «оператор» «операнд2» «=» «результат»

Пример. 5 + 2 = 7 (p = 10)

- Выполнение операций с одним операндом:
  «операнд» «оператор» «=» «результат»

Пример. 5 * = 25 (p = 10)

- Повторное выполнение последней операции:
  «=»«результат» «=» «результат»

Пример. 5 + 4 = 9 = 13 = 17(p = 10)

- Выполнение операции над отображаемым значением в качестве обоих операндов:
  «результат» «оператор» «=» «результат»

Пример. 2 + 3 = 5 = 8 + = 16(p = 10)

- Вычисление функций:
  «операнд» «Sqr» «результат»

Пример. 5 «Sqr» 25 (p = 10)

- Вычисление выражений:
  «операнд1» «функция1» «оператор1» «операнд2» «функция2» «оператор2» …«операндN» «операторN» «=»«результат»

Пример.

| ввод | 6 | Sqr | + | 2 | Sqr | / | 10 | + | 6 | = |
|---|---|---|---|---|---|---|---|---|---|---|
| Отображаемый результат | 6 | 36 | 36 | 2 | 4 | 40 | 10 | 4 | 6 | 10 |

Операнды выражений могут копироваться из памяти, буфера обмена или вводиться с помощью кнопок и клавиатуры.

Варианта 2 – «Калькулятор простых дробей».

Требования.

1. Калькулятор должен обеспечить ввод и редактирование целых чисел в обычной записи и рациональных дробей в записи:

[-]<целое без знака>|[-]<числитель><разделитель><знаменатель>.

<числитель>::= <целое без знака>

<знаменатель>::= <целое без знака>

<разделитель>::= ‘/’ | ‘|’

2. Предусмотреть настройку калькулятора на отображение результата в двух форматах: «дробь» или «число». В формате «дробь» результат всегда отображается в виде дроби. В формате «число» результат отображается в виде числа, если дробь может быть сокращена, так что знаменатель равен 1.

**Необходимо предусмотреть следующие варианты использования (прецеденты) калькулятора:**

- Выполнение одиночных операций:
  «операнд1» «оператор» «операнд2» «=» «результат»

Пример. 5/1 + 2/1 = 7/1.

- Выполнение операций с одним операндом:
  «операнд» «оператор» «=» «результат»

Пример. 5/1 * = 25/1.

- Повторное выполнение операции:
  «=»«результат» «=» «результат»

Пример. 5/1 + 4/1 = 9/1 = 13/1 = 17.

- Выполнение операции над отображаемым значением в качестве обоих операндов:
  «результат» «оператор» «=» «результат»

Пример. 2/1 + 3/1 = 5/1 = 8/1 + = 16/1.

- Вычисление функций:
  «операнд» «Sqr» «результат»

Пример. 5/1 «Sqr» 25/1.

- Вычисление выражений:
  «операнд1» «функция1» «оператор1» «операнд2» «функция2» «оператор2» …«операндN» «операторN» «=»«результат»

Пример.

| ввод | 6/1 | Sqr | + | 2/1 | Sqr | / | 10/1 | + | 6/1 | = |
|---|---|---|---|---|---|---|---|---|---|---|
| Отображаемый результат | 6/1 | 36/1 | 36/1 | 2/1 | 4/1 | 40/1 | 10/1 | 4/1 | 6/1 | 10/1 |

Операнды могут выражений могут копироваться из памяти, из буфера обмена или вводиться с помощью кнопок и клавиатуры.

Варианта 3 – «Калькулятор комплексных чисел».

Требования.

1. Калькулятор обеспечивает ввод комплексных чисел в записи:

[-]<действительная часть><разделитель>[-] <мнимая часть>

<действительная часть>::= <действительное число без знака с целой и\или дробной частями>

<мнимая часть>::= <действительное число без знака с целой и\или дробной частями>

<разделитель>::= ‘i*’

2. Предусмотреть настройку калькулятора на отображение результата в двух форматах: “комплексное” или “действительное” число. В формате «комплексное» результат всегда отображается в виде комплексного числа. В формате «действительное» результат отображается в виде действительного, если мнимая часть равна 0.
3. Калькулятор должен вычислять функции: Pwr - возведение в целую степень, Root - извлечение целого корня (Предусмотреть возможность вывода всех корней), Mdl - вычисление модуля комплексного числа, Cnr - вычисление аргумента комплексного числа в градусах, Cnr - вычисление аргумента комплексного числа в радианах. Предусмотреть ввод показателя степени для возведения в степень и извлечения корня. Результат вычисления указанных выше функций отображайте в отдельных компонентах. Эти операции вычисляются отдельно, а не в составе выражения.

**Необходимо предусмотреть следующие варианты использования калькулятора (прецеденты):**

- Выполнение одиночных операций:
  «операнд1» «оператор» «операнд2» «=» «результат»

  Пример. 5 + 2 = 7.
- Выполнение операций с одним операндом:
  «операнд» «оператор» «=» «результат»

  Пример. 5 * = 25.
- Повторное выполнение операции:
  «=» «результат» «=» «результат»

  Пример. 5 + 4 = 9 = 13 = 17.

- Выполнение операции над отображаемым значением в качестве обоих операндов:

  «результат» «оператор» «=» «результат»

Пример. 2 + 3 = 5 = 8 + = 16.

- Вычисление функций:

  «операнд» «Sqr» «результат»

Пример. 5 «Sqr» 25 (p = 10)

- Вычисление выражений:

  «операнд1» «функция1» «оператор1» «операнд2» «функция2» «оператор2» …«операндN» «операторN» «=»«результат»

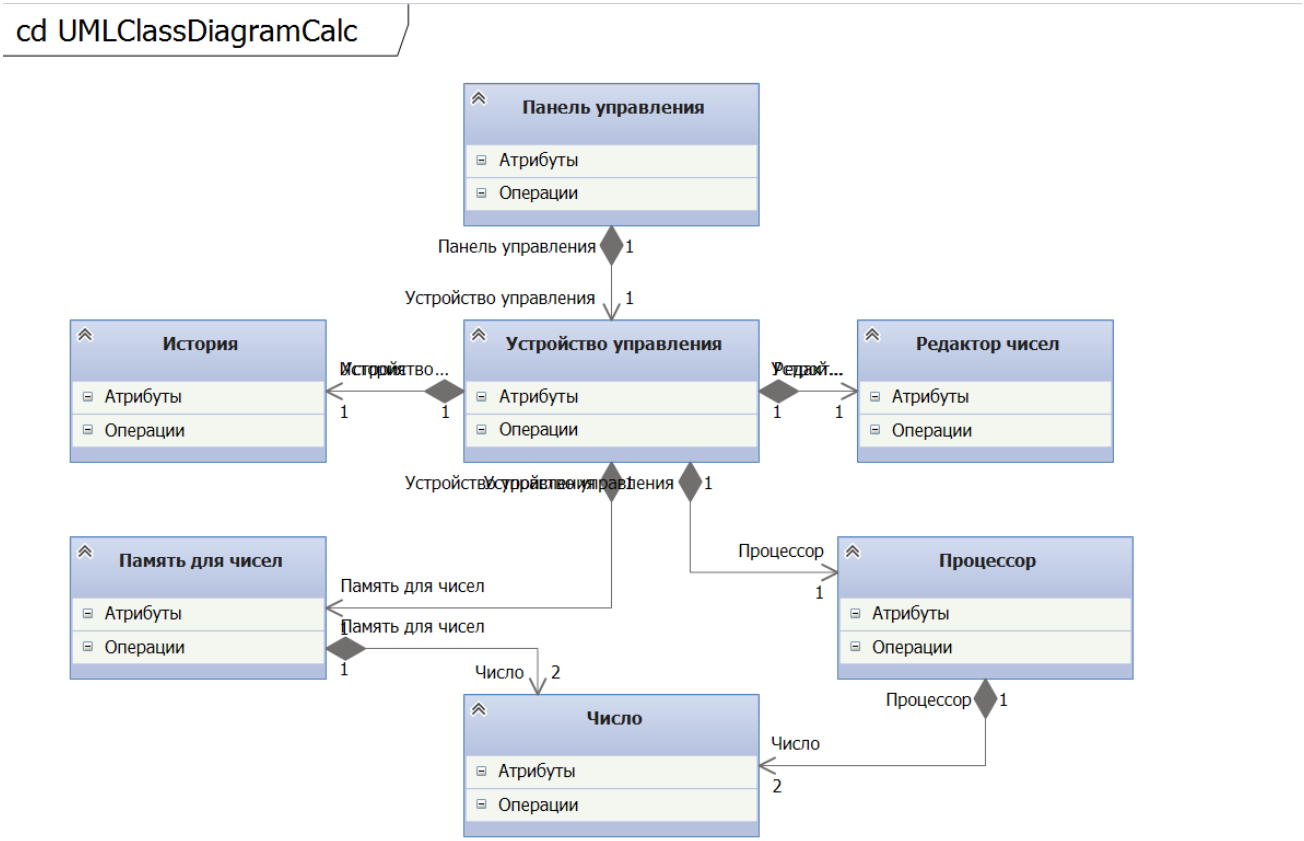
Пример.

| ввод | 6 | Sqr | + | 2 | Sqr | / | 10 | + | 6 | = |
|---|---|---|---|---|---|---|---|---|---|---|
| Отображаемый результат | 6 | 36 | 36 | 2 | 4 | 40 | 10 | 4 | 6 | 10 |

Операнды выражений могут копироваться из памяти, буфера обмена или вводиться с помощью кнопок и клавиатуры.

Рекомендации к выполнению

1. В качестве буфера обмена используйте глобальный объект ClipBoard класса TClipBoard, доступный во всех работающих приложениях.
2. Диаграмма классов UML для калькулятора представлена на рисунке.




Здесь класс число в зависимости от варианта может быть: p-ичное число, простая дробь, комплексное число.

3. Для выполнения задания вам необходимо выполнить практические работы, приведённые в методических указаниях. Рекомендуемый порядок выполнения:

- Вариант - калькулятор p-ичных чисел:
    a. Практическая работа. Абстрактный тип данных p - ичное число
    b. Практическая работа. Редактор p - ичных чисел
    c. Практическая работа. Параметризованный абстрактный тип данных «Память»
    d. Практическая работа. Параметризованный абстрактный тип данных «Процессор»
    e. Практическая работа. Управление калькулятором
    f. В Практическая работа. Интерфейс калькулятора

- Вариант - калькулятор простых дробей:
    a. Практическая работа. Абстрактный тип данных  простая дробь
    b. Практическая работа. Редактор простых дробей
    c. Практическая работа. Параметризованный абстрактный тип данных «Память»
    d. Практическая работа. Параметризованный абстрактный тип данных «Процессор»
    e. Практическая работа. Управление калькулятором
    f. Практическая работа. Интерфейс калькулятора
- Вариант - калькулятор комплексных чисел:
    a. Практическая работа. Абстрактный тип данных  комплексное число
    b. Практическая работа. Редактор комплексных чисел
    c. Практическая работа. Параметризованный абстрактный тип данных «Память»
    d. Практическая работа. Параметризованный абстрактный тип данных «Процессор»
    e. Практическая работа. Управление калькулятором
    f. Практическая работа. Интерфейс калькулятора

## *Практическая работа. Абстрактный тип данных р - ичное число*

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C++.

Задание

1. Реализовать абстрактный тип данных «р-ичное число», используя класс C++, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, одним из методов тестирования.

Спецификация типа данных «р-ичное число».

**ADT TPNumber**

**Данные**

Р-ичное число TPNumber - это действительное число (n) со знаком в системе счисления с основанием (b) (b в диапазоне 2..16), содержащее целую и дробную части. Точность представления числа с (с >= 0). Р-ичные числа изменяемые.

**Операции**

Операции могут вызываться только объектом р-ичное число (тип TPNumber), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «само число».

| ***Операция*** | ***Описание*** |
|---|---|
| ***Конструктор*** | |
| Начальные значения: | Вещественное число (a) во внутреннем формате, система счисления (b), точность представления числа (c) |
| Процесс: | Инициализирует поля р-ичного числа: система счисления (b), точность |

| | |
|---|---|
| | представления (c). В поле (n) созданного числа заносится (a).<br>Например:<br>Конструктор(a,3,3) = число a в системе счисления 3 с тремя разрядами после троичной точки.<br>Конструктор(a,3,2) = число a в системе счисления 3 с двумя разрядами после троичной точки. |
| | |
| ***Конструктор*** | |
| Начальные значения: | Строковое представление p–ичного числа (a), система счисления (b), точность представления числа (c) |
| Процесс: | Инициализирует поля p-ичного числа: система счисления (b), точность представления (c). В поле (n) созданного числа заносится результат преобразования строки (a) в числовое представление. b-ичное число (a) и основание системы счисления (b) представлены в формате строки.<br>Например:<br>Конструктор(‘20’,’3’,’6’) = 20 в системе счисления 3, точность 6 знаков после запятой.<br>Конструктор(‘0’,’3’,’8’) = 0 в системе счисления 3, точность 8 знаков после |

| | |
|---|---|
| | запятой. |
| | |
| **Копировать:** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт копию самого числа (тип TPNumber). |
| Выход: | p-ичное число. |
| Постусловия: | Нет. |
| | |
| ***Сложить*** | |
| Вход: | P-ичное число d с основанием и точностью такими же, как у самого числа. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает p-ичное число (тип TPNumber), полученное сложением полей (n) самого числа и числа d. |
| Выход: | p-ичное число. |
| Постусловия: | Нет |
| | |
| ***Умножить*** | |
| Вход: | P-ичное число d с основанием и точностью такими же, как у самого числа. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает p-ичное число (тип TPNumber), полученное умножением полей (n) самого числа и числа d. |
| Выход: | P-ичное число (тип TPNumber). |

| | |
|---|---|
| Постусловия: | Нет. |
| | |
| ***Вычесть*** | |
| Вход: | P-ичное число d с основанием и точностью такими же, как у самого числа. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает p-ичное число (тип TPNumber), полученное вычитанием полей (n) самого числа и числа d. |
| Выход: | P-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
| | |
| ***Делить*** | |
| Вход: | P-ичное число d с основанием и точностью такими же, как у самого числа. |
| Предусловия: | Поле (n) числа (d) не равно 0. |
| Процесс: | Создаёт и возвращает p-ичное число (тип TPNumber), полученное делением полей (n) самого числа на поле (n) числа d. |
| Выход: | P-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
| | |
| ***Обратить*** | |
| Вход: | Нет. |
| Предусловия: | Поле (n) самого числа не равно 0. |
| Процесс: | Создаёт p-ичное число, в поле (n) которого заносится значение, полученное как 1/(n) самого числа. |
| Выход: | P-ичное число (тип TPNumber). |

| | |
|---|---|
| Постусловия: | Нет. |
| | |
| ***Квадрат*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт p-ичное число, в поле (n) которого заносится значение, полученное как квадрат поля (n) самого числа. |
| Выход: | P-ичное число (тип TPNumber). |
| Постусловия: | Нет. |
| | |
| ***ВзятьРЧисло*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (n) самого числа. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьРСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает p-ичное число (q) в формате строки, изображающей значение поля (n) самого числа в системе счисления (b) с точностью (c). |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***ВзятьОснованиеЧисло*** | |

| | |
|---|---|
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (b) самого числа (q). |
| Выход: | Целочисленное значение |
| Постусловия: | Нет. |
| | |
| ***ВзятьОснованиеСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (b) самого числа в формате строки, изображающей (b) в десятичной системе счисления. |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***ВзятьТочностьЧисло*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (c) самого числа . |
| Выход: | Целое значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьТочностьСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение поля (c) самого числа в формате строки, изображающей (c) в десятичной системе счисления. |

| | |
|---|---|
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***УстановитьОснованиеЧисло*** | |
| Вход: | Целое число (newb). |
| Предусловия: | 2 <= newb <= 16. |
| Процесс: | Устанавливает в поле (b) самого числа значение (newb). |
| Выход: | Нет. |
| Постусловия: | Нет. |
| | |
| ***УстановитьОснованиеСтрока*** | |
| Вход: | Строка (bs), изображающая основание (b) р-ичного числа в десятичной системе счисления. |
| Предусловия: | Допустимый диапазон числа, изображаемого строкой (bs) - 2,,16. |
| Процесс: | Устанавливает значение поля (b) самого числа значением, полученным в результате преобразования строки (bs). |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***УстановитьТочностьЧисло*** | |
| Вход: | Целое число (newc). |
| Предусловия: | newc >= 0. |

| | |
|---|---|
| Процесс: | Устанавливает в поле (c) самого числа значение (newc). |
| Выход: | Нет. |
| Постусловия: | Нет. |
| | |
| ***УстановитьТочностьСтрока*** | |
| Вход: | Строка (newc). |
| Предусловия: | Строка (newc) изображает десятичное целое >= 0. |
| Процесс: | Устанавливает в поле (c) самого числа значение, полученное преобразованием строки (newc). |
| Выход: | Нет. |
| Постусловия: | Нет. |

***end TPNumber***

Рекомендации к выполнению

1. Тип данных реализовать, используя класс C++.
2. Число храните как поле вещественного типа.
3. Основание системы счисления храните как поле целочисленного типа.
4. Тип данных реализовать в отдельном модуле UPNumber.

Содержание отчета

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

Контрольные вопросы

1. Что такое инкапсуляция?

2. Как синтаксически представлено поле в описании класса?
3. Как синтаксически представлен метод в описании класса?
4. В чём состоит назначение конструктора?

***Практическая работа. Абстрактный тип данных простая дробь***

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C++.

Задание

1. Реализовать абстрактный тип данных «простая дробь», используя класс C++ в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных одним из методов тестирования.

Спецификация типа данных «простые дроби».

**ADT TFrac**

**Данные**

Простая дробь (тип TFrac) - это пара целых чисел: числитель и знаменатель (a/b). Простые дроби изменяемые.

**Операции**

Операции могут вызываться только объектом простая дробь (тип **TFrac**), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «сама дробь».

| ***Конструктор*** | |
|---|---|
| Начальные значения: | Пара целых чисел (a) и (b). |
| Процесс: | Инициализирует поля простой дроби (тип TFrac): числитель значением a, |

| | |
|---|---|
| | знаменатель - (b). В случае необходимости дробь предварительно сокращается.<br>Например:<br>*Конструктор*(6,3) = (2/1)<br>*Конструктор*(0,3) = (0/3). |
| | |
| ***Конструктор*** | |
| Начальные значения: | Строковое представление простой дроби . Например: ‘7/9’. |
| Процесс: | Инициализирует поля простой дроби (тип TFrac) строкой f =’a/b’. Числитель значением a, знаменатель - b. В случае необходимости дробь предварительно сокращается.<br>Например:<br>*Конструктор*(‘6/3’) = 2/1<br>*Конструктор* (‘0/3’) = 0/3 |
| | |
| **Копировать:** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт копию самой дроби (тип TFrac) с числителем, и знаменателем такими же, как у самой дроби. |
| Выход: | Простая дробь (тип TFrac).<br>Например:<br>c = 2/1, Копировать(c) = 2/1 |

| | |
|---|---|
| Постусловия: | Нет. |
| | |
| ***Сложить*** | |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную сложением самой дроби q = a1/b1 с d =a2/b2: ((a1/b1)+(a2/b2)=(a1*b2 + a2*b1)/( b1* b2)).<br>Например:<br>q = 1/2, d = -3/4<br>q.Сложить(d) = -1/4. |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
| | |
| ***Умножить*** | |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Создаёт простую дробь (тип TFrac), полученную умножением самой дроби q = a1/b1 на d = a2/b2 ((a1/b1)*(a2/b2)=(a1* a2)/( b1* b2)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
| | |
| ***Вычесть*** | |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |

| | |
|---|---|
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную вычитанием d = a2/b2 из самой дроби q = a1/b1: ((a1/b1)-(a2/b2)=(a1* b2-a2*b1)/(b1*b2)).<br>Например:<br>q = (1/2), d = (1/2)<br>q.Вычесть(d) = (0/1). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет |
| | |
| ***Делить*** | |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Числитель числа d не равно 0. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученное делением самой дроби q = a1/b1 на дробь d = a2/b2: ((a1/b1)/(a2/b2)=(a1* b2)/( a2*b1)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
| | |
| ***Квадрат*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученную умножением самой дроби на себя: ((a/b)*(a/b)=(a* a)/( b* b)). |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |

| | |
|---|---|
| | |
| ***Обратное*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает простую дробь (тип TFrac), полученное делением единицы на саму дробь: 1/((a/b) = b/a. |
| Выход: | Простая дробь (тип TFrac) |
| Постусловия: | Нет. |
| | |
| ***Минус*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт простую дробь, являющуюся разностью простых дробей z и q, где z - простая дробь (0/1), дробь, вызвавшая метод. |
| Выход: | Простая дробь (тип TFrac). |
| Постусловия: | Нет. |
| | |
| ***Равно*** | |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет |
| Процесс: | Сравнивает саму простую дробь q и d. Возвращает значение True, если q и d - тождественные простые дроби, и значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |

| | |
|---|---|
| | |
| ***Больше*** | |
| Вход: | Простая дробь d (тип TFrac). |
| Предусловия: | Нет. |
| Процесс: | Сравнивает саму простую дробь q и d. Возвращает значение True, если q > d, - значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьЧислительЧисло*** | |
| Вход: | |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение числителя дроби в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьЗнаменательЧисло*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение знаменателя дроби в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьЧислительСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |

| | |
|---|---|
| Процесс: | Возвращает значение числителя дроби в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***ВзятьЗнаменательСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение знаменателя дроби в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***ВзятьДробьСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Возвращает значение простой дроби, в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |

***end TFracRatio***

Рекомендации к выполнению

1. Тип данных реализовать, используя класс C++.
2. Для записи и считывания полей простой дроби использовать свойства (property).
3. Тип данных реализовать в отдельном модуле UFrac.

Содержание отчета

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

Контрольные вопросы

1. Особенности описания методов класса?
2. Особенности описания и назначение конструктора класса?
3. Видимость идентификаторов в описании класса?
4. Особенности вызова методов применительно к объектам класса?
5. Что такоё абстрактный тип данных?

### *Практическая работа. Абстрактный тип данных комплексное число*

Цель

Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов C++.

Задание

1. Реализовать абстрактный тип данных «комплексное число», используя класс C++, в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных одним из методов тестирования.

Спецификация типа данных «комплексное число».

**ADT TComplex**

**Данные** Комплексное число TComplex - это изменяемая пара вещественных чисел, представляющие действительную и мнимую части комплексного числа (a + i*b).

**Операции**

Операции могут вызываться только объектом комплексное число (тип TComplex), указатель на который в них передаётся по умолчанию. При описании операций этот объект называется «само число».

| | |
|---|---|
| ***Конструктор*** | |
| Начальные значения: | Пара вещественных чисел (a) и (b). |
| Процесс: | Инициализирует поля комплексного числа (тип TComplex) значениями: действительную часть - a), мнимую - b.<br>Например:<br>*Конструктор*(6,3)=6 + i*3<br>*Конструктор*(3,0)=3 + i*0<br>*Конструктор*(0,0)=0 + i*0 |
| | |
| ***Конструктор*** | |
| Начальные значения: | Строка, представляющая комплексное число. |
| Процесс: | Инициализирует поля комплексного числа (тип TComplex) значениями представленными строкой f = ’a + i*b’: действительную частью значением a, комплексную часть - b.<br>Например:<br>*Конструктор*(‘6+i*3’) = 6+i*3<br>*Конструктор*(‘0+i*3’) = 0+i*3 |
| | |
| **Копировать:** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает собственную |

| | |
|---|---|
| | копию - комплексное число (тип TComplex) с действительной и мнимой частями такими же как у самого числа. |
| Выход: | Комплексное число (тип TComplex).<br>Например:<br>c = 6+i3, Копировать(c) = 6+i3 |
| Постусловия: | Нет. |
| | |
| ***Сложить*** | |
| Вход: | Комплексное число d (тип TComplex). |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает комплексное число, полученное сложением самого числа q = a1+i*b1 с числом d = a2+i*b2: ((a1+i*b1)+(a2+i*b2)=(a1+a2)+i*(b1+b2)).<br>Например:<br>q = (2 +i*1), d = (2 +i*1),<br>q.Сложить(d) = (4 +i*2). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Умножить*** | |
| Вход: | Комплексное число d (тип TComplex). |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает комплексное число, полученное умножением самого числа q = a1+i*b1 на число d = a2+i*b2: ((a1+i*b1)*(a2+i*b2)=(a1*a2 - |

| | |
|---|---|
| | b1*b2)+i*(a1*b2+ a2*b1)). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Квадрат*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает комплексное число (тип TComplex), полученное умножением числа на самого себя: ((a1+i*b1)*(a1+i*b1)=(a1*a1 - b1*b1)+i*(a1*b1+ a1*b1)). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Обратное*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает комплексное число (тип TComplex), полученное делением единицы на само число 1/((a1+i*b1) = a1/(a1**2 + b1**2) - i* b1/( a1**2 + b1**2 )). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Вычесть*** | |
| Вход: | Комплексное число d (тип TComplex).. |
| Предусловия: | Нет. |

| | |
|---|---|
| Процесс | Создаёт и возвращает комплексное число (тип TComplex), полученное вычитанием d = a2 + i b2 из самого себя q = (a1+i*b1): (a1+i*b1)-(a2+i*b2)=(a1-a2)+i*(b1-b2).<br>Например:<br>q = (2 +i*1), d = (2 +i*1))<br>q.Вычесть(d) = (0 + i0). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Делить*** | |
| Вход: | Комплексное число (d). |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает комплексное число (тип TComplex), полученное делением самого числа (q) на число (d) ((a1+i*b1)/(a2+i*b2)=(a1*a2 + b1*b2)/(a2**2 + b2**2)+i*(a2*b1 – a1*b2)/(a2**2 + b2**2)). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Минус*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает комплексное число (тип TComplex), являющееся разностью комплексных чисел z и и |

| | |
|---|---|
| | самого числа, где z – комплексное число (0+i0). |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Модуль*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Вычисляет и возвращает модуль самого комплексного числа (q).<br>Например:<br>q = (2 +i*1), q. Модуль = √(2*2+1*1).<br>q = (i*17), q. Модуль = √(0*0+17*17). |
| Выход: | Вещественное число. |
| Постусловия: | Нет. |
| | |
| ***УголРад*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Возвращает аргумент fi самого комплексного числа q (в радианах). fi = (arcTg(b/a), a>0; pi/2, a = 0, b > 0; arcTg(b/a) + pi, a < 0; -pi/2, a = 0, b <0 ).<br>Например:<br>q = (1 +i*1), q. УголРад = 0,79. |
| Выход: | Вещественное число. |
| Постусловия: | Нет. |
| | |

| | |
|---|---|
| ***УголГрад*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Возвращает аргумент самого комплексного числа q (в градусах). Например: q = (1 +i*1), q. Град = 45. |
| Выход: | Вещественное число. |
| Постусловия: | Нет. |
| | |
| ***Степень*** | |
| Вход: | Целое (n). |
| Предусловия: | Нет. |
| Процесс | Возвращает целую положительную степень n самого комплексного числа q. $q^n = r^n(\cos(n*fi)+ i*\sin(n*fi))$. |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Корень*** | |
| Вход: | Целое (n), целое (i). |
| Предусловия: | Нет. |
| Процесс | Возвращает i-ый корень целой положительной степени n самого комплексного числа q. $\sqrt[n]{(q)} = \sqrt[n]{(r)}*(\cos((fi + 2*k*pi)/n)+ i*\sin((fi + 2*k*pi)/n))$. При этом коэффициенту k придается последовательно n значений: k = 0,1,2…, n - 1 и получают n значений корня, т.е. |

| | |
|---|---|
| | ровно столько, каков показатель корня. |
| Выход: | Комплексное число (тип TComplex). |
| Постусловия: | Нет. |
| | |
| ***Равно*** | |
| Вход: | Комплексное число (d). |
| Предусловия: | Нет. |
| Процесс | Сравнивает само комплексное число с числом (d). Возвращает значение True, если они - тождественные комплексные числа, и значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
| | |
| ***НеРавно*** | |
| Вход: | Комплексное число (d). |
| Предусловия: | Нет. |
| Процесс | Сравнивает само комплексное число с числом (d). Возвращает значение True, если само число <> d, - значение False - в противном случае. |
| Выход: | Булевское значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьReЧисло*** | |
| Вход: | Нет |
| Предусловия: | Нет. |
| Процесс | Возвращает значение действительной |

| | |
|---|---|
| | части самого комплексного числа в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьImЧисло*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Возвращает значение мнимой части самого комплексного числа в числовом формате. |
| Выход: | Вещественное значение. |
| Постусловия: | Нет. |
| | |
| ***ВзятьReСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Возвращает значение вещественной части самого комплексного числа в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***ВзятьImСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Возвращает значение мнимой части самого комплексного числа в строковом формате. |

| | |
|---|---|
| Выход: | Строка. |
| Постусловия: | Нет. |
| | |
| ***ВзятьКомплексноеСтрока*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Возвращает значение самого комплексного числа в строковом формате. |
| Выход: | Строка. |
| Постусловия: | Нет. |

***end TComplex***

Рекомендации к выполнению

1. Тип данных реализовать, используя класс.
2. Для описания полей комплексного числа использовать свойства (property).
3. Тип данных реализовать в отдельном модуле UComplex.

Содержание отчета

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

Контрольные вопросы

1. Что такое интерфейс класса?
2. Что такое реализация класса?
3. Особенности описания и назначение деструктора класса?
4. Особенности описания и назначение конструктора копирования?
5. Особенности вызова методов применительно к объектам класса?
6. Что такое сообщение применительно к объектам класса?

***Практическая работа. Редактор р-ичных чисел***

Цель

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

Задание

1. Разработать и реализовать класс TEditor «Редактор р-ичных чисел», используя класс С++.

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

| **РедакторР-ичныхЧисел** |
|---|
| строка: String<br>числоЕстьНоль: Boolean<br>добавитьЗнак: String<br>добавитьР-ичную цифру(a: Integer): String<br>добавитьНоль: String<br>забойСимвола: String<br>очистить: String<br>конструктор<br>читатьСтрокаВформатеСтроки: String (метод свойства)<br>писатьСтрокаВформатеСтроки(a: String) (метод свойства)<br>редактировать(a: Integer): String |
| Обязанность:<br>ввод, хранение и редактирование строкового представления р-ичных чисел |

2. Класс должен отвечать за ввод и редактирование строкового представления р-ичных чисел. Значение р-ичного нуля - ‘0,’. Класс должен обеспечивать:

- добавление символов, соответствующих p-ичным цифрам (p от 2 до 16);
- добавление и изменение знака;
- добавление разделителя целой и дробной частей;
- забой символа, стоящего справа (BackSpace);
- установку нулевого значения числа (Clear);
- чтение строкового представления p-ичного числа;
- запись строкового представления p-ичного числа;

3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. В классе TEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемого p- ичного числа, .
2. В классе 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 p-ичную цифру», операция получает целое число (числовое обозначение p-ичной 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 p-ичный 0, возвращает значение «строка»;

- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «конструктор», создаёт объект типа TEditor;
- «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. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы:
   - «разделитель целой и дробной частей» строкового типа;
   - «строковое представление нуля» строкового типа.

Содержание отчета

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

Контрольные вопросы

1. В чём состоит особенность раздела описания класса с уровнем доступа protected?
2. В чём состоит особенность раздела описания класса с уровнем доступа private?
3. В чём состоит особенность раздела описания класса с уровнем доступа public?
4. В чём состоит особенность инициализации полей ссылочного типа и констант в конструкторе?
5. Что такое указатель this?
6. Что такое статические элементы класса?

***Практическая работа. Редактор простых дробей***

Цель

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

Задание

1. Разработать и реализовать класс TEditor «Ввод и редактирование простых дробей», используя класс C++.

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

| **РедакторПростыхДробей** |
|---|
| строка: String |
| дробьЕстьНоль: Boolean<br>добавитьЗнак: String<br>добавитьЦифру(a: Integer): String<br>добавитьНоль: String<br>забойСимвола: String<br>очистить: String<br>конструктор<br>читатьСтрокаВформатеСтроки: String (метод свойства)<br>писатьСтрокаВформатеСтроки(a: String) (метод свойства)<br>редактировать(a: Integer): String |
| Обязанность:<br>ввод, хранение и редактирование строкового представления простых дробей. |

2. Класс должен отвечать за посимвольный ввод, хранение и редактирование строкового представления простых дробей. Значение нуля - ‘0|1’. Класс должен обеспечивать:
   - добавление цифры;

- добавление и изменение знака;
- добавление разделителя целой и дробной частей;
- забой символа, стоящего справа (BackSpace);
- установку нулевого значения числа (Clear);
- чтение строкового представления простой дроби;
- запись строкового представления простой дроби.

3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. В классе TEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемой простой дроби.
2. В классе опишите следующие операции:
   - «дробь есть ноль», операция возвращает булевское значение True, если «строка» содержит изображение дроби равной 0/1, 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 арабской 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 дробь 0/1, возвращает значение «строка»;

- «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
- «конструктор», создаёт объект типа TEditor;
- «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. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы:
   - «разделитель числителя и знаменателя» строкового типа;
   - «строковое представление нуля» строкового типа.

Содержание отчета

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

Контрольные вопросы

1. В чём состоят особенности статических полей?
2. В чём состоят особенности статических методов?
3. В чём состоит перегрузка операций для класса?
4. В чём состоит особенность константных методов?

### *Практическая работа. Редактор комплексных чисел*

Цель

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

Задание

1. Разработать и реализовать класс «Ввод и редактирование комплексных чисел» (TEditor), используя класс C++.

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

| **РедакторКомплексныхЧисел** |
|---|
| строка: String<br>комплексноеЧислоЕстьНоль: Boolean<br>добавитьЗнак: String<br>добавитьЦифру(a: Integer): String<br>добавитьНоль: String<br>забойСимвола: String<br>очистить: String<br>конструктор<br>читатьСтрокаВформатеСтроки: String (метод свойства)<br>писатьСтрокаВформатеСтроки(a: String) (метод свойства)<br>редактировать(a: Integer): String |
| Обязанность:<br>ввод, хранение и редактирование строкового представления комплексных чисел |

2. Класс должен отвечать за посимвольный ввод, хранение и редактирование строкового представления комплексных чисел. Значение комплексного нуля - ‘0, i* 0,’. Класс должен обеспечивать:
   - добавление цифры;
   - добавление и изменение знака действительной и мнимой частей;
   - добавление разделителя целой и дробной частей действительной и мнимой частей комплексного числа;
   - добавление разделителя мнимой и действительной частей комплексного числа

- забой символа, стоящего справа (BackSpace);
- установку нулевого значения комплексного числа (Clear);
- чтение строкового представления комплексного числа;
- запись строкового представления комплексного числа.

3. Протестировать каждый метод класса.

Рекомендации к выполнению

1. В классе TEditor опишите следующие атрибуты:
   - «строка» - строкового типа, содержит строковое представление редактируемого комплексного числа, .
2. В классе опишите следующие операции:
   - «число есть ноль», операция возвращает булевское значение True, если «строка» содержит изображение комплексного числа равного 0, +i 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 арабской 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 комплексное число 0, +i 0,, возвращает значение «строка»;
   - «редактировать», операция получает номер команды редактирования, выполняет действия по её выполнению и возвращает значение «строка»;
   - «конструктор», создаёт объект типа TEditor;

- «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. Класс реализуйте в отдельном модуле UEditor. В разделе описания констант опишите следующие константы:
   - «разделитель целой и дробной частей действительной и мнимой частей комплексного числа» - строкового типа;
   - «разделитель действительной и мнимой частей комплексного числа» - строкового типа;
   - «строковое представление нуля» - строкового типа.

Содержание отчета

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

Контрольные вопросы

1. Когда в классе необходимо явным образом описать деструктор?
2. Что такое конструктор по умолчанию?
3. Когда в классе необходимо явным образом описать конструктор копирования?

## *Практическая работа. Параметризованный абстрактный тип данных «Память»*

Цель

Сформировать практические навыки реализации параметризованного абстрактного типа данных с помощью шаблона классов C++.

Задание

1. В соответствии с приведенной ниже спецификацией реализовать параметризованный абстрактный тип данных «память», для хранения одного числа – объекта типа T, используя шаблон классов *C++*.
2. Протестировать каждую операцию, определенную на типе данных одним из методов тестирования.

Спецификация типа данных «память».

**ADT** TMemory

**Данные**

Память (тип TMemory, в дальнейшем - память) - это память для хранения «числа» объекта типа T в поле FNumber, и значения «состояние памяти» в поле FState. Объект память - изменяемый. Он имеет два состояния, обозначаемых значениями: «Включена» (_On), «Выключена» (_Off). Её изменяют операции: Записать (Store), Добавить (Add), Очистить (Clear).

**Операции**

| | |
|---|---|
| ***Конструктор*** | |
| Начальные значения: | Нет. |
| Процесс: | Инициализирует поле FNumber объекта «память» (тип TMemory) объектом «число» (тип T) со значением по умолчанию. Например для числа типа TFrac со значением 0/1. Память устанавливается в состояние «Выключена», в поле FState «состояние памяти» заносится значение (_Off). |
| ***Записать*** | |
| Вход: | (E) – объект тип T. |
| Предусловия: | Нет. |

| | |
|---|---|
| Процесс: | В объект «память» (тип TMemory) в поле FNumber записывается копия объекта E. Память устанавливается в состояние «Включена», в поле FState «состояние памяти» заносится значение (_On). |
| Выход: | Нет. |
| Постусловия: | Состояние памяти поле FState – «Включена» (_On). |
| | |
| ***Взять*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Создаёт и возвращает копию объекта хранящегося в объекте «память» (тип TMemory) в поле FNumber. |
| Выход: | Объект типа T. |
| Постусловия: | Состояние памяти поле FState – «Включена» (_On). |
| | |
| ***Добавить*** | |
| Вход: | (E) – число объект типа T. |
| Предусловия: | Нет. |
| Процесс: | В поле FNumber объекта «память» (тип TMemory) записывается объект типа T, полученный в результате сложения числа (E) и числа, хранящегося в памяти в поле FNumber. |
| Выход: | Нет. |
| Постусловия: | Состояние памяти поле FState – |

| | |
|---|---|
| | «Включена» (_On). |
| | |
| ***Очистить*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | В поле числа (FNumber) объекта «память» (тип TMemory) записывается объект типа T со значением по умолчанию. Например, для простой дроби - 0/1. Память (поле FState) устанавливается в состояние «Выключена» (_Off). |
| Выход: | Нет. |
| Постусловия: | Состояние памяти поле FState – «Выключена» (_Off). |
| | |
| ***ЧитатьСостояниеПамяти*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Копирует и возвращает значение поля FState «состояние памяти» объекта «память» (тип TMemory) в формате строки. |
| Выход: | Значение поля «состояния памяти» (типа String). |
| Постусловия: | Нет. |
| | |
| ***ЧитатьЧисло*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Копирует и возвращает значение поля |

| | |
|---|---|
| | «число» (FNumber) объекта «память» (тип TMemory). |
| Выход: | Объект число (тип T). |
| Постусловия: | Нет. |

***end*** TCMemory

Рекомендации к выполнению

1. Тип данных реализуйте, используя параметризованный класс C++.
   template <class T>
2. Число храните в поле FNumber типа T.
3. Для чтения состояния памяти и хранимого значения используйте свойство (property).
4. Тип данных реализуйте в отдельном модуле UMemory.

**Ниже приведены диаграмма классов и диаграмма состояний для класса «Память».**

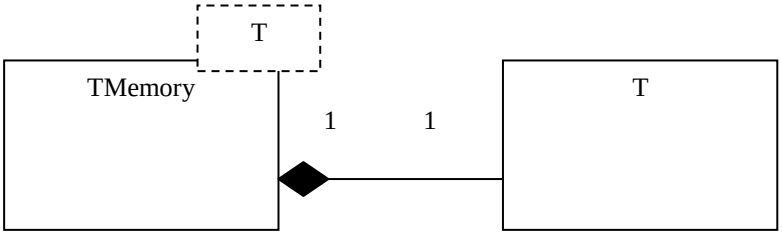



Рис. Диаграмма классов для класса «Память».

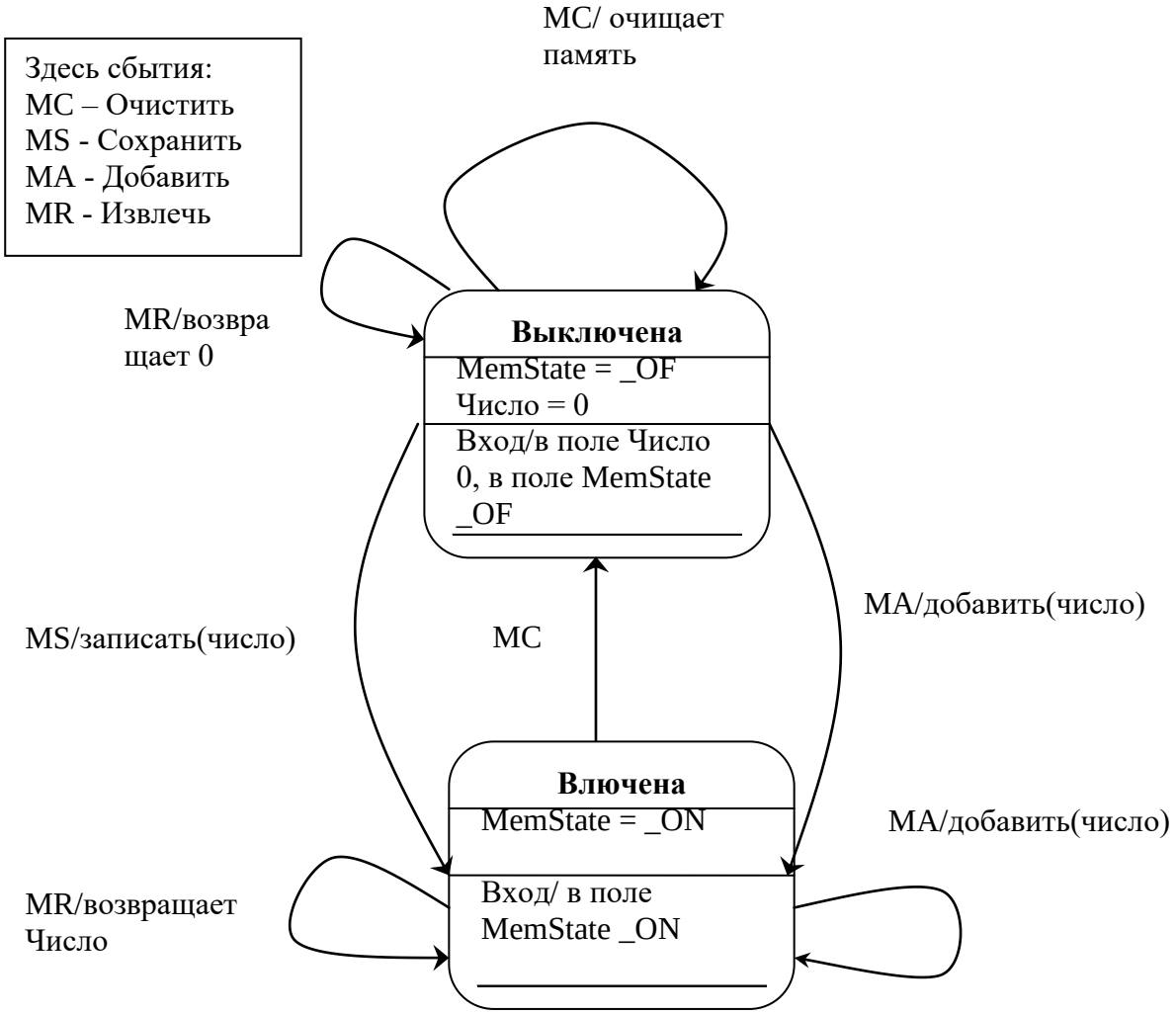



Рис. Диаграмма состояния для класса «Память»

Содержание отчета

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

Контрольные вопросы

1. Когда в классе необходимо явным образом описать конструктор?
2. Что можно использовать в качестве параметров шаблона?
3. Можно ли использовать шаблоны в качестве параметров шаблона?

***Практическая работа. Параметризованный абстрактный тип данных «Процессор».***

Цель

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

Задание

1. В соответствии с приведенной ниже спецификацией реализовать параметризованный абстрактный тип данных «Процессор», используя шаблон классов *C++*.
2. Протестировать тип данных.

Спецификация типа данных «Процессор».

**ADT** TProc

**Данные**

Процессор (тип TProc) выполняет двухоперандные операции TOprtn = (None, Add, Sub, Mul, Dvd) и одноопернадные операции - функции TFunc = (Rev, Sqr) над значениями типа T. Левый операнд и результат операции хранится в поле Lop_Res, правый - в поле Rop. Оба поля имеют тип T. Процессор может находиться в состояниях: «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). На значениях типа T должны быть определены указанные выше операции и функции.

**Операции**

| | |
|---|---|
| ***Конструктор*** | |

| | |
|---|---|
| Начальные значения: | Нет |
| Процесс: | Инициализирует поля объекта процессор типа TProc. Поля Lop_Res, Rop инициализируются объектами (тип T) со значениями по умолчанию. Например, для простых дробей - 0/1. Процессор устанавливается в состояние: «операция не установлена»: (Operation = None). |
| | |
| ***СбросПроцессора*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс: | Поля объекта процессор: Lop_Res, Rop инициализируются объектами (тип T) со значениями по умолчанию. Например, для простых дробей - 0/1. Процессор устанавливается в состояние: «операция не установлена»: (Operation = None). |
| Выход: | Нет. |
| Постусловия: | Состояние процессора – «операция сброшена» (Operation = None). |
| | |
| ***СбросОперации*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Процессор устанавливается в состояние: «операция не установлена»: (Operation = None). |

| | |
|---|---|
| Выход: | Нет. |
| Постусловия: | Состояние процессора – «операция сброшена» (Operation = None). |
| | |
| ***ВыполнитьОперацию*** | |
| 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. Состояние объекта не изменяется. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора не изменяется. |
| | |
| ***ВычислитьФункцию*** | |
| 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. Состояние объекта не изменяется. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора не меняется. |
| | |

| | |
|---|---|
| ***ЧитатьЛевыйОперанд*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает копию объекта, который хранится в поле Lop_Res. |
| Выход: | Объект типа T. |
| Постусловия: | Состояние процессора не изменяется. |
| | |
| ***ЗаписатьЛевыйОперанд*** | |
| Вход: | Переменная Operand типа T. |
| Предусловия: | Нет. |
| Процесс | Создаёт копию объекта Operand и заносит её в поле Lop_Res. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора не изменяется. |
| | |
| ***ЧитатьПравыйОперанд*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Создаёт и возвращает копию объекта, который хранится в Rop. |
| Выход: | Объект типа T. |
| Постусловия: | Состояние процессора не меняется. |
| | |
| ***ЗаписатьПравыйОперанд*** | |
| Вход: | Переменная Operand типа T. |
| Предусловия: | Нет. |
| Процесс | Создаёт копию объекта Operand и заносит её в поле Rop. |

| | |
|---|---|
| Выход: | Нет. |
| Постусловия: | Состояние процессора не изменяется. |
| | |
| ***ЧитатьСостояние*** | |
| Вход: | Нет. |
| Предусловия: | Нет. |
| Процесс | Копирует и возвращает значение поля Operation. |
| Выход: | Значение поля Operation. |
| Постусловия: | Состояние процессора не изменяется. |
| | |
| ***ЗаписатьСостояние*** | |
| Вход: | Переменная Oprtn типа TOprtn. |
| Предусловия: | Нет. |
| Процесс | Заносит значение Oprtn в поле Operation. |
| Выход: | Нет. |
| Постусловия: | Состояние процессора изменяется на Oprtn. |

Рекомендации к выполнению

1. Тип данных TProc реализовать, используя шаблон классов C++, template <class T>.
2. Числа храните как поля типа T.
3. Для чтения состояния процессора, полей: «левый операнд-результат» (Lop_Res), «правый операнд» (Rop), используйте свойство (property).
4. Тип данных реализовать в отдельном модуле UProc.
5. В приведённой ниже таблице показана последовательность изменения состояния процессора, если T = TFrac, при вычислении выражения:

$$2/1 + 3/1 * (4/1)^2$$

| Шаг | Вход | Метод | Rop | Lop_Res | Operation |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 0 | | Create | 0/1 | 0/1 | None |
| 1 | 2 | | 0/1 | 0/1 | None |
| 2 | + | Lop_Res_Set; OprtnSet | 0/1 | 2/1 | Add |
| 3 | 3 | | 0/1 | 2/1 | Add |
| 4 | * | Rop_Set; OprtnRun; OprtnSet; | 3/1 | 2/1+3/1 | Mul |
| 5 | 4 | | 4/1 | 2/1+3/1 | Mul |
| 6 | Sqr | Rop_Set; FuncRun | (4/1)^2 | 2/1+3/1 | Mul |
| 7 | = | OprtnRun | (4/1)^2 | 2/1+3/1* (4/1)^2 | Mul |
| 8 | C | ReSet | 0/1 | 0/1 | None |

**Ниже приведена диаграмма классов для класса память.**

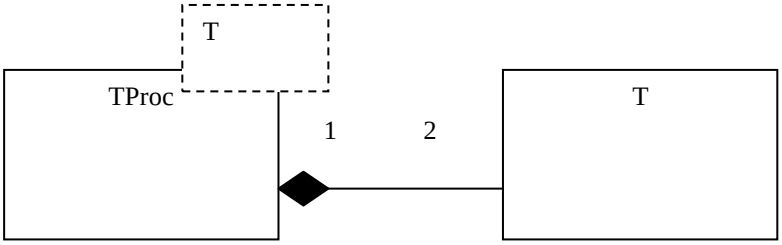



Рис. Диаграмма классов Процессор.

Содержание отчета

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

Контрольные вопросы

1. Как использовать тип в качестве параметра шаблона?
2. Как использовать переменную в качестве параметра шаблона?

3. Какие существуют отношения между классами?

***Практическая работа. Управление калькулятором***

Цель

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

Задание

1. Разработать и реализовать класс «Управление калькулятором чисел» тип TCtrl, используя класс C++. Тип чисел, которые обрабатывает калькулятор, зависит от варианта.

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

| **УправлениеКалькуляторомПростыхДробей (тип TCtrl)** |
|---|
| состояниеКалькулятора: TCtrlState<br>редактор: TEditor<br>процессор: TProc<br>память: TMemory<br>число: TFrac |
| выполнитьКомандуКалькулятора(a: Integer; var b, MState: String): String<br>выполнитьКомандуРедактора(a: Integer): String<br>выполнитьОперацию(a: Integer): String<br>выполнитьФункцию(a: Integer): String<br>вычислитьВыражение(a: Integer): String<br>установитьНачальноеСостояниеКалькулятора(a: Integer): String<br>выполнитьКомандуюПамяти(a: Integer; var MState: String): String<br>читатьПисатьСостояниеКалькулятора: TCtrlState |

| выполнитьКомандуБуфераОбмена(a: Integer; var b: String): String<br>конструктор<br>деструктор |
|---|
| Обязанность:<br>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

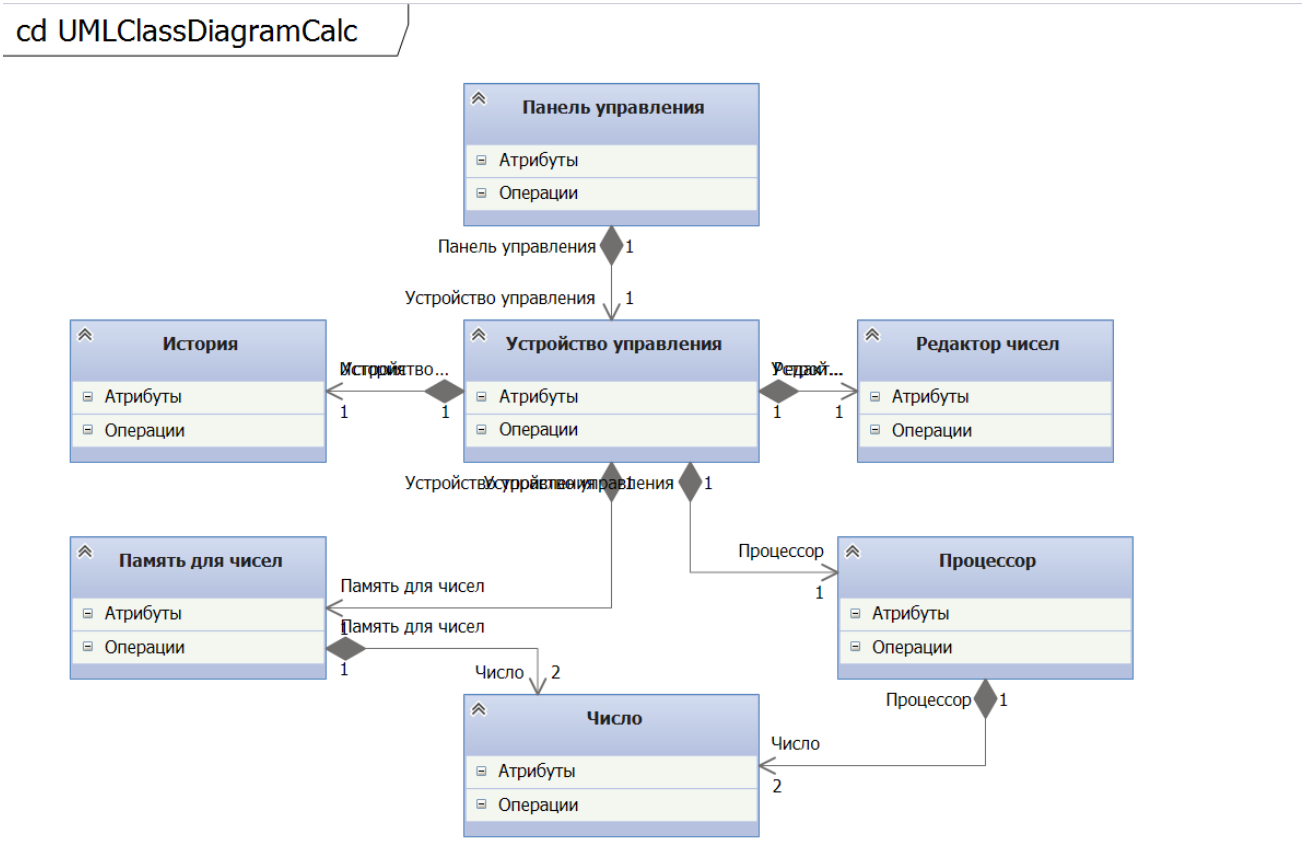
1. Класс TCtrl реализуйте в отдельном модуле UControl.
2. В модуле опишите перечисляемый тип TCtrlState = (cStart, cEditing, FunDone, cValDone, cExpDone, cOpChange, cError) для обозначения состояний калькулятора: cStart (Начальное), cEditing (Ввод и редактирование), cExpDone (Выражение вычислено), cOpDone (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 TFrac (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:
6. 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);
7. построить диаграмму состояний.

Диаграмма классов для Управления калькулятором простых дробей представлено 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?

***Практическая работа. Интерфейс***

Цель

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

Задание

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

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

| **ИнтерфейсКалькулятораПростыхДробей** | |
|---|---|
| строкаПростаяДробь: | 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 (действительное, 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

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. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. Учебник. — 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2005. 544 с.: ил.

2. Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем: Учебное пособие для вузов. СПб.: Питер, 2003. – 472с.: ил.
3. Г. Буч. Обьектно-ориентированное проектирование с примерами приложений на C++, 2-ое издание. Учебник/: Пер. с англ. - М.: Издательство Бином, СПб.: Невский Диалект, 1999. - 560с.