Предмет:

Современные технологии программирования

Язык: С++

## Лабораторная работа 1. Абстрактный тип данных простая дробь

### Цель

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

### Задание

1. Реализовать абстрактный тип данных «простая дробь», используя класс С++ 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. Например:*Конструктор*(6,3) = (2/1)*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.Например:*Конструктор*(‘6/3’) = 2/1*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).Например: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)). Например:q = 1/2, d = -3/4q.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)). Например:q = (1/2), d = (1/2)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. 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?

## Лабораторная работа 2. Абстрактный тип данных «комплексное число»

### Тема: Классы С++

*Цель: Сформировать практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов С++. Синтаксис классов: инкапсуляция, простые свойства.*

### Задание

1. Реализовать абстрактный тип данных «комплексное число», используя класс С++, 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 неизменяемая 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».

|  |  |
| --- | --- |
| ***КонструкторЧисло*** |  |
|  Начальные значения: | Пара вещественных чисел (a) и (b). |
|  Процесс: | Создаёт комплексное число (тип TComplex) с действительной частью (a) и мнимой (b). Например:*КонструкторЧисло* (6,3)=6 + i\*3*КонструкторЧисло* (3,0)=3 + i\*0*КонструкторЧисло* (0,0)=0 + i\*0 |
|  |
| ***КонструкторСтрока*** |  |
|  Начальные значения: | Строка, представляющая комплексное число.  |
|  Процесс: | Создаёт комплексное число (тип TComplex), представленное строкой f = ’a + i\*b’, с действительной частью (a) и комплексной частью (b).Например:*КонструкторСтрока*(‘6+i\*3’) = 6+i\*3*КонструкторСтрока*(‘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).Например: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)). Например:q = (2 +i\*1), d = (2 +i\*1),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). Например:q = (2 +i\*1), d = (2 +i\*1))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). Например:q = (2 +i\*1), q. Модуль = √(2\*2+1\*1).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 ). Например: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. qn = rn(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. √n(q) = √n(r)\* (cos ((fi + 2\*k\*pi)/n)+ i\* sin ((fi + 2\*k\*pi)/n)). При этом коофициенту 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. Тип данных реализовать в отдельном модуле Ucomplex в консольном приложении.

### 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. Как синтаксически представлено простое свойство в описании класса?
5. Особенности описания методов класса?
6. Особенности описания и назначение конструктора класса?
7. Видимость идентификаторов в описании класса?
8. Особенности вызова методов применительно к объектам класса?

## Лабораторная работа 3. Абстрактный тип данных (ADT) p - ичное число

### Тема: Классы Object Pascal, С++

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

### Задание

1. Реализовать абстрактный тип данных «р-ичное число», используя класс, 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) (в диапазоне 2..16), содержащее целую и дробную части. Точность представления числа – (c >= 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».

|  |  |
| --- | --- |
| ***КонструкторЧисло*** |  |
|  Начальные значения: | Вещественное число (a) во внутреннем формате, система счисления (b), точность представления числа (c) |
|  Процесс: | Создаёт p-ичное число: система счисления (b), точность представления (c). В поле (n) созданного числа заносится (a).Например:NCreate(a,3,3) = число a в системе счисления 3 с тремя разрядами после троичной точки.NCreate(a,3,2) = число a в системе счисления 3 с двумя разрядами после троичной точки. |
|  |
| ***КонструкторСтрока*** |  |
|  Начальные значения: | Строковое представление р–ичного числа (a), система счисления (b), точность представления числа (c) |
|  Процесс: | Создаёт р-ичное число: 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.Например: SCreate(‘20’,’3’,’6’) = 20 в системе счисления 3, точность 6 знаков после запятой.SCreate(‘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: | р-ичное число. |
|  Постусловия: | Нет. |
|  |
| ***Сложить*** |  |
|  Вход: | Р-ичное число d с основанием и точностью такими же, как у самого числа. |
|  Предусловия: | Нет. |
|  Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное сложением полей (n) самого числа и числа d. |
|  Выход: | р-ичное число. |
|  Постусловия: | Нет |
|  |
| ***Умножить*** |  |
|  Вход: | Р-ичное число d с основанием и точностью такими же, как у самого числа. |
|  Предусловия: | Нет. |
|  Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное умножением полей (n) самого числа и числа d. |
|  Выход: | Р-ичное число (тип TPNumber). |
|  Постусловия: | Нет. |
|  |
| ***Вычесть*** |  |
|  Вход: | Р-ичное число d с основанием и точностью такими же, как у самого числа. |
|  Предусловия: | Нет. |
|  Процесс: | Создаёт и возвращает р-ичное число (тип TPNumber), полученное вычитанием полей (n) самого числа и числа d. |
|  Выход: | Р-ичное число (тип TPNumber). |
|  Постусловия: | Нет. |
|  |
| ***Делить*** |  |
|  Вход: | Р-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 р-ичное число (тип TPNumber), полученное делением полей (n) самого числа на поле (n) числа d. |
|  Выход: | Р-ичное число (тип TPNumber). |
|  Постусловия: | Нет. |
|  |
| ***Обратить*** |  |
|  Вход: | Нет. |
|  Предусловия: | Поле (n) самого числа не равно 0. |
|  Процесс: | Создаёт р-ичное число, в поле (n) которого заносится значение, полученное как 1/(n) самого числа. |
|  Выход: | Р-ичное число (тип TPNumber). |
|  Постусловия: | Нет. |
|  |
| ***Квадрат*** |  |
|  Вход: | Нет. |
|  Предусловия: | Нет. |
|  Процесс: | Создаёт р-ичное число, в поле (n) которого заносится значение, полученное как квадрат поля (n) самого числа.  |
|  Выход: | Р-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 р-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) p-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 С++.
2. Число храните как поле вещественного типа.
3. Основание системы счисления храните как поле целочисленного типа.
4. Тип данных реализовать в отдельном модуле UPNumber в режиме консольного приложения.

### 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. Как синтаксически представлено простое свойство в описании класса?
5. Особенности описания методов класса?
6. Особенности описания и назначение конструктора класса?
7. Видимость идентификаторов в описании класса?
8. Особенности вызова методов применительно к объектам класса?

## Контрольная работа

### Тема: Последовательные контейнеры STL и модульное тестирование

Цель: Сформировать практические навыки разработки абстракций данных на основе контейнеров STL и модульного тестирования средствами Visual Studio.

### Задание

Реализовать обработку данных пользовательского типа (объектов класса) с помощью контейнера в соответствии с вариантом задания и со следующей спецификацией:

* приложение заполняет контейнер данными, которые вводятся пользователем с консоли;
* выводит содержимое контейнера на консоль для контроля ввода;
* выполняет сортировку контейнера в порядке возрастания значений объектов с помощью алгоритма или метода контейнера;
* выводит содержимое контейнера на консоль для контроля операции;
* выполняет сортировку контейнера в порядке убывания значений объектов с помощью алгоритма или метода контейнера;
* выводит содержимое контейнера на консоль для контроля операции;
* вычисляет сумму значений объектов с помощью алгоритма и выводит значение на консоль.

Протестировать методы класса объектов, помещаемых в контейнер, используя средства модульного тестирования Visual Studio. Тестовые наборы необходимо построить на основе критериев тестирования C0,C1,C2 в зависимости от варианта задания.

### Варианты задания

|  |  |  |  |
| --- | --- | --- | --- |
| №Варианта | Тип контейнера | Класс объектов | Критерий тестирования |
| 3 | deque | Р-ичное число | С0,C1 |

### Рекомендации к выполнению

1. Заданную обработку данных реализуйте как консольное приложение, используя классы языка программирования и библиотеку шаблонов STL.
2. Добавьте в класс объектов (в соответствии с вариантом задания), помещаемых в контейнер перегруженные операторы < и > для выполнения сортировки. В контрольной работе используются классы, разработанные вами в лабораторных работах.
3. Выполняйте сортировку контейнера с помощью алгоритма sort или метода контейнера в зависимости от варианта.
4. Вычисление суммы значений объектов контейнера выполняйте с помощью алгоритма accumulate и функционального объекта plus<A>(), здесь А – класс ваших объектов;
5. Для выполнения описанных в задании операций по обработке данных, разработайте класс со следующим описанием:

class InOutDo

{

public:

 static void Input(…)

 {

 //Вводим данные с клавиатуры и заносим в контейнер.

 }

 static void Output(…)

 {

//Выводим содержимое контейнера на монитор.

}

 static A Sum(…)

 {

 //Находим сумму

 }

 static void SortUp(…)

 {

 //Сортируем вектор по возрастанию

 }

 static void SortDown(…)

 {

 //Сортируем вектор по убыванию

 }

};

Тогда функция main примет примерно такой вид:

int \_tmain(int argc, \_TCHAR\* argv[])

{

 SetConsoleCP(1251);

 SetConsoleOutputCP(1251);

 //Описание вашего Контейнера с объектами вашего класса А.

 //Контейнер<A> m;

 InOutDo::Input(m);

 InOutDo::Output(m);

 InOutDo::SortUp(m);

 InOutDo::Output(m);

 InOutDo::SortDown(m);

 InOutDo::Output(m);

 cout << InOutDo::Sum(m).get() << endl;

 system("PAUSE");

 return 0;

}

Примеры программ обработки данных с помощью деки, вектора и списка приведены ниже.

**Пример 1. Обработка данных с декой.**

// ConsoleDeque.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <string>

#include <sstream>

#include <deque>

#include <algorithm>

#include <numeric>//Определяет шаблонны функций контейнера, которые выполняют алгоритмы числовой обработки.

#include <functional>

#include "windows.h"

using namespace std;

//Класс объектов, помещаемых в контейнер.

class A

{

 int n, d;

public:

 A(int n = 0, int d = 1) : n(n), d(d){};

 A operator+(const A b)const

 {

 return A((n\*b.d + b.n\*d), d\*b.d);

 };

 string get()

 {

 string a;

 ostringstream os;

 os << n << "/" << d;

 return os.str();

 };

 bool operator>(const A b)const

 {

 return n\*b.d > d\*b.n;

 };

 bool operator<(const A b)const

 {

 return n\*b.d < d\*b.n;

 };

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

 SetConsoleCP(1251);

 SetConsoleOutputCP(1251);

 //Дека простых дробей

 deque<A> m;

 //Итератор для деки

 deque<A>::iterator iter;

 //Проталкиваем дроби в деку

 m.push\_back(A(2));

 m.push\_back(A(3));

 m.push\_back(A(1));

 m.push\_back(A(5));

 m.push\_back(A(9));

 m.push\_back(A(7));

 //Сортируем деку по возрастанию

 sort(m.begin(), m.end(), less<A>());

 for (iter = m.begin(); iter != m.end(); iter++)

 cout << iter->get() << " ";

 cout << endl;

 //Сортируем деку по убыванию

 sort(m.begin(), m.end(), greater<A>());

 for (int j = 0; j != m.size(); j++)

 cout << m[j].get() << " ";

 cout << endl;

 //Находим сумму

 A sum = accumulate(m.begin(), m.end(), A(), plus<A>());

 cout << sum.get() << endl;

 system("PAUSE");

 return 0;

 return 0;

}

**Пример 2. Обработка данных с вектором.**

// ConsoleVector.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <string>

#include <sstream>

#include <vector>

#include <algorithm>

#include <numeric>//Определяет шаблонны функций контейнера, которые выполняют алгоритмы числовой обработки.

#include <functional>

#include "windows.h"

using namespace std;

//Класс объектов, помещаемых в контейнер.

class A

{

 int n, d;

public:

 A(int n = 0, int d = 1) : n(n), d(d){};

 A operator+(const A b)const

 {

 return A((n\*b.d + b.n\*d), d\*b.d);

 };

 string get()

 {

 string a;

 ostringstream os;

 os << n << "/" << d;

 return os.str();

 };

 bool operator>(const A b)const

 {

 return n\*b.d > d\*b.n;

 };

 bool operator<(const A b)const

 {

 return n\*b.d < d\*b.n;

 };

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

 SetConsoleCP(1251);

 SetConsoleOutputCP(1251);

 //Вектор простых дробей

 vector<A> m;

 //Итератор для вектора

 vector<A>::iterator iter;

 //Проталкиваем дроби в вектор

 m.push\_back(A(2));

 m.push\_back(A(3));

 m.push\_back(A(1));

 m.push\_back(A(5));

 m.push\_back(A(9));

 m.push\_back(A(7));

 //Сортируем вектор по возрастанию

 sort(m.begin(), m.end(), less<A>());

 for (iter = m.begin(); iter != m.end(); iter++)

 cout << iter->get() << " ";

 cout << endl;

 //Сортируем вектор по убыванию

 sort(m.begin(), m.end(), greater<A>());

 for (int j = 0; j != m.size(); j++)

 cout << m[j].get() << " ";

 cout << endl;

 //Находим сумму

 A sum = accumulate(m.begin(), m.end(), A(), plus<A>());

 cout << sum.get() << endl;

 system("PAUSE");

 return 0;

}

**Пример 3. Обработка данных со списком.**

// ConsoleList.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <string>

#include <sstream>

#include <list>

#include <algorithm>

#include <numeric>//Определяет шаблонны функций контейнера, которые выполняют алгоритмы числовой обработки.

#include <functional>

#include "windows.h"

using namespace std;

//Класс объектов, помещаемых в контейнер.

class A

{

 int n, d;

public:

 A(int n = 0, int d = 1) : n(n), d(d){};

 A operator+(const A b)const

 {

 return A((n\*b.d + b.n\*d), d\*b.d);

 };

 string get()

 {

 string a;

 ostringstream os;

 os << n << "/" << d;

 return os.str();

 };

 bool operator>(const A b)const

 {

 return n\*b.d > d\*b.n;

 };

 bool operator<(const A b)const

 {

 return n\*b.d < d\*b.n;

 };

};

int \_tmain(int argc, \_TCHAR\* argv[])

{

 SetConsoleCP(1251);

 SetConsoleOutputCP(1251);

 //Список простых дробей

 list<A> m;

 //Итератор для списка

 list<A>::iterator iter;

 //Проталкиваем дроби в вектор

 m.push\_back(A(2));

 m.push\_back(A(3));

 m.push\_back(A(1));

 m.push\_back(A(5));

 m.push\_back(A(9));

 m.push\_back(A(7));

 //Сортируем список по возрастанию

 //sort(m.begin(), m.end(), );

 m.sort(less<A>());

 for (iter = m.begin(); iter != m.end(); iter++)

 cout << iter->get() << " ";

 cout << endl;

 //Сортируем список по убыванию

 //sort(m.begin(), m.end(), );

 m.sort(greater<A>());

 for (iter = m.begin(); iter != m.end(); iter++)

 cout << iter->get() << " ";

 cout << endl;

 //Находим сумму

 A sum = accumulate(m.begin(), m.end(), A(), plus<A>());

 cout << sum.get() << endl;

 system("PAUSE");

 return 0;

 return 0;

}

**Пример модульного теста для тестирования перегруженных операторов +, == класса А приведён ниже.**

#include "stdafx.h"

#include "CppUnitTest.h"

#include "..\ConsoleDeque\A.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest1

{

 TEST\_CLASS(UnitTest1)

 {

 public:

 TEST\_METHOD(Add)

 {

 A a = A(1, 2);

 A b = A(1, 2);

 A c = a + b;

 string s = "4/4";

 Assert::AreEqual(s, c.get());

 }

 TEST\_METHOD(Equal)

 {

 A a = A(1, 2);

 A b = A(2, 4);

 A c = a + b;

 string s = "4/4";

 Assert::AreEqual(true, a == b);

 }

 };

}

### Порядок выполнения

Реализуйте задание с помощью проекта «консольное приложение»:

1. Создайте консольное приложение и сохраните его под именем CJob\_1.
2. Добавьте к исходному тексту консольного приложения описание вашего класса (объекты которого вы будете обрабатывать с помощью контейнера) в соответствии с вариантом задания.
3. Добавьте в описание класса перегруженные операторы отношения (<,>) и, если необходимо, недостающие методы.
4. Добавьте в ваше решение класс модульного теста C++. Разработайте тестовый набор данных для тестирования методов вашего класса по критериям тестирования в соответствии с вариантом и протестируйте их, используя средства модульного тестирования Visual Studio.
5. Добавьте к исходному тексту консольного приложения описание класса InOutDo в соответствии с заданием.
6. Используя методы класса InOutDo, выполните требуемую обработку данных.

Таблица. Тестовый набор для тестирования методов класса объектов, помещаемых в контейнер.

|  |
| --- |
| Тестовый набор для тестирования операции Сложить для класса простых дробей |
| Номер теста | Исходные данные | Ожидаемый результат |
| Вход | Вход | Возвращаемое значение |
| 1 | (1/2) | (1/2) | (4/4) |

### Контрольные вопросы

1. В чём состоит сущность критерия C0?
2. В чём состоит сущность критерия C1?
3. В чём состоит сущность критерия C2?
4. Что такое УПГ?
5. Что такое путь в УПГ?
6. Что такое ветвь УПГ?
7. В каком файле описан последовательный контейнер deque?
8. В каком файле описан последовательный контейнер vector?
9. В каком файле описан последовательный контейнер list?
10. Что означает имя iterator в области видимости последовательного контейнера?
11. Что такое функциональный объект?
12. Назначение и параметры алгоритма sort() для последовательных контейнеров?
13. Назначение и параметры алгоритма accumulate () для последовательных контейнеров?
14. Назначение метода size() последовательного контейнера?
15. В чём особенности статических методов?
16. В чём особенности последовательных контейнеров?

### Содержание отчета

1. Задание.
2. Текст программы.
3. Скриншот.

## Литература

1. Написание модульных тестов для C/C++ в Visual Studio [Электронный ресурс]  URL: <https://docs.microsoft.com/ru-ru/visualstudio/test/writing-unit-tests-for-c-cpp> (дата обращения 21.03.18).
2. Руководство по программированию на C# [Электронный ресурс]  URL: https://metanit.com/cpp/tutorial/1.1.php (дата обращения 20.03.18).