Архитектура вычислительных систем.

Вариант 305

Задание 11, Функция 22

Проломов Виталий Алексеевич БПИ208

**Задача**

**Сущности**

1. Комплексные (действительная и мнимая части – пара действительных чисел)
2. Простые дроби (числитель, знаменатель – пара целых чисел)
3. Полярные координаты (угол [радиан] – действительное; координаты конечной точки на плоскости)

### **Общие для всех альтернатив функции:**

Приведение каждого значения к действительному числу, эквивалентному записанному

### Обобщенная структура - число

Общих полей у классов нет

Задание

1) Зафиксировать для отчета основные характеристики программы, такие как:

число интерфейсных модулей (заголовочных файлов) и

модулей реализации (файлов с определением программных объектов),

общий размер исходных текстов, полученный размер исполняемого кода (если он формируется),

время выполнения программы для различных тестовых наборов данных.

2) В соответствии с вариантом задания разработать программный продукт с

использованием процедурного подхода и статической типизацией.

Разработку осуществить на языке программирования C или на языке C++ в стиле C.

3) Провести отладку и тестирование разработанной программы на заранее

подготовленных тестовых наборах данных.

Количество тестовых наборов данных – не менее пяти.

Число уникальных элементов в тестовых наборах должно варьироваться от нуля до 10000.

При необходимости, программа должна правильно обрабатывать переполнение по данным.

Тестовые наборы до 20 элементов должны вводиться из заранее подготовленных тестовых файлов. Тестовые данные с большим числом элементов должны порождаться программно с использованием генераторов случайных наборов данных. Данные, формируемые генератором случайных наборов, должны поддерживать допустимые значения. Управление вводом данных задается из командной строки.

4) Описать структуру используемой ВС с наложением на нее обобщенной схемы

разработанной программы.

--1--.

1. Интерфейсные модули:

number.h

complex\_number.h

rational\_number.h

rnd.h

polar\_coordinates.h

container.h

Всего: 6

2) Модули реализации

number.cpp

complex\_number.cpp

rational\_number.cpp

rnd.cpp

polar\_coordinates.cpp

container.cpp

Всего: 6

3) общий размер исходных текстов = 1 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 2 + 3 + 4 = 26КБ

1. время выполнения программы для различных тестовых наборов данных: (тесты рассчитаны на значения в диапазоне от 1 до 20 вводимых эл-тов (чисел))

1 тест:

0.008000 sec

|  |  |
| --- | --- |
| Тесты процедурной программы |  |
| 1 тест: |  |
| 0.002000 sec |  |
| 2 тест: |  |
| 0.003000 sec |  |
| 3 тест: |  |
| 0.003000 sec |  |
| 4 тест: |  |
| 0.004000 sec |  |
| 5 тест: |  |
| 0.004000 sec |  |
| 6 тест: |  |
| 0.003000 sec |  |
| 7 тест : |  |
| 0.001000 sec |  |
| 8 тест: |  |
| 0.00500 sec |  |

2 тест:

0.006400 sec

3 тест:

0.005300 sec

4 тест:

0.011000 sec

5 тест:

0.009200 sec

6 тест:

0.003000 sec

7 тест :

0.004700 sec

8 тест:

0.012000 sec

--2--.

Программа загружена в систему Github.

--3--.

Папка с тестами лежит рядом с exe файлом.

Вот путь:

HW1/cmake-build-debug/tests

--4--.

1. **Описание структуры ВС**

**ТАБЛИЦА КЛАССОВ**

**Базовые Классы**

|  |  |
| --- | --- |
| **1** | **Container [80004 байт]** |
| Поля | **Static:** Отсутствует  **Local:** int len [4 байта]  Number \*storage[80000] байт |
| Методы | **Static**: -  **Local:**  void Clear();  void In(FILE \*file);  void InRandom(int size);  void Out(FILE \*file);  void SortLargerElements() ; |

**Базовый абстрактный класс**

|  |  |
| --- | --- |
| **1** | **Number** |
| Поля | **Static: -**  **Local: -** |
| Методы | **Static: (ГП)**  Number \*StaticIn(FILE \*file);  Number \*StaticInRnd(FILE \*file)  Local: ~Number ();  Random rnd20();  Random rnd3() |

**Производные классы**

|  |  |
| --- | --- |
| **1** | **Complex\_number: Number[8 байт]** |
| Поля | **Static:** -  **Local:**  int real\_part 4[0]  int imaginary\_part 4[4] |
| Методы | **Static:** -  Local(override): ~Complex\_number();  void In(FILE \*fileIn);  void InRandom();  void Out(FILE \*fileOut);  double RealRepresentation(); |
| **2** | **Polar\_coordinates: Number[8 байт]** |
| Поля | **Static: -**  **Local:**  int length 4[0]  int radians 4[4] |
| Методы | Static: -  Local: ~ Polar\_coordinates();  void In(FILE \*file);  void InRandom();  void Out(FILE \*file);  double Fraction(); |
| **3** | **Rational\_numbers: Number[8 байт]** |
| Поля | **Static**: -  **Local:**  int numerator 4[0]  Int denominator 4 [4] |
| Методы | **Static:** -  **Local**: ~ Rational\_numbers();  void In(FILE \*file);  void InRandom();  void Out(FILE \*file);  double Fraction(); |

**ЛОКАЛЬНЫЕ МЕТОДЫ**

|  |  |
| --- | --- |
| Container |  |
| Void In(Container \*self, FILE \*file)  Void InRnd(Container \*self, int size)  void Out(Container \*self)  Double SortLargerElements(Container \*self) |  |

ТАБЛИЦА ВИРТУАЛЬНЫХ МЕТОДОВ БК

|  |  |
| --- | --- |
| Void In(Number \*self, FILE \*file)  Void InRnd(Number \*self, int size)  void Out(Number \*self)  Double SortLargerElements(Number \*self) |  |

ТАБЛИЦА ВИРТУАЛЬНЫХ МЕТОДОВ ПК

|  |  |
| --- | --- |
| Void In(Number \*self, FILE \*file)  Void InRnd(Number \*self, int size)  void Out(Number \*self)  Double SortLargerElements(Number \*self) |  |

**ПАМЯТЬ ПРОГРАММЫ**

|  |  |
| --- | --- |
| Main (40024 байт) | |
| Container c | 40008 |
| Int size | 4[40008] |
| FILE \*input | 4[40012] |
| FILE \*output1 | 4[40016] |
| FILE \*output2 | 4[40020] |

СРАВНИТЕЛЬНЫЙ АНАЛИЗ С ПРОЦЕДУРНЫМ ПОДХОДОМ

* Время работы программы при процедурном подоходе намного меньше, чем при ОО подходе. (Видно из данных выше) - преимущество процедурного подхода
* Процедурный подход в основном исп-ся в небольших программах, где нет сложной и массивной архитектуры, так как он быстрее по времени. Однако при более сложных структурах, для повышения читаемсоти и понятности кода прибегают к ОО подходу. - преимущество ООП
* ООП программа зачастую требует рисунка схемы, отражающую композицию программы (из каких сущностей она состоит, есть ли между ними какие-то отношения). Процедурный подход этого не требует - преимущество процедурного подхода
* Процедурный подход выстраивает взаимодействие между объектами через обобщенную структуру, и в ООП это выстроено намного лучше - преимущество ООП

**Формат ввода:**

Для ввода одного числа следует ввести следующие данные:

X1

f11 f12

X1 - тип вводимого числа (1 - для комплексного числа, 2 - для рационального числа, 3 - для полярных координат)

В следующей строке вводятся параметры для указанного числа (для каждого типа числа ровно 2 параметра)

Для комплексных - реальную и мнимую часть

Для рациональных - числитель и знаменатель

Для полярных координат - угол в радианах и длину до начала координат

Расположение входных и выходных данных

Входные данные лежат в:

HW-2/cmake-build-debug/tests/input

Выходные лежат в:

HW-2/cmake-build-debug/tests/output**Схема**

1. HEAP (Куча) - int\*
2. Стек (прилагаю картинку)

