*Преподаватель: Ольга Леонидовна Викентьева*

*Телеграмм: @oleovic*

Лабораторные работы на 1 семестр: №1, 3, 5, 9, 10, 11. Варианты по номеру в журнале

Парадигмы программирования:

1. Процедурно-ориентированное
   1. Входные данные (источники данных)
   2. Множество функции
   3. Выходные данные – результат (потребитель данных)

Main(){ f1() f2() … fn()}

1. Объектно-ориентированное
2. Функциональное
3. Логическое

**Жизненный цикл ПО** (ISO/IEC 12207) – совокупность процессов связанных с последовательным изменением состояния программного обеспечения от формирования исходных требований к нему до окончания его эксплуатации

***Команда разработки ПО:***

**Заказчик –** финансирует разработку**; пользователь**

**Аналитик** формирует понятные для разработчика требования

**Архитектор** формирует архитектуру программы

**Разработчик** пишет программу

**Тестировщик** тестирует программу

**Инженер по развертыванию** запускает программу для общего пользования

*Дизайнер*

*Менеджер*

Этапы разработки **Модель каскад (при простой разработке):**

1. Анализ
2. Проектирование
3. Разработка
4. Интеграция и тестирование
5. Внедрение

Тестирование – деление программы с целью нахождения в ней ошибок

Отладка - определение места возникновения ошибки в программе

Тест – набор исходных данных, для которых заранее известен результат

Ошибки:

1. Синтаксические
2. Семантические
3. Ошибки выполнения программы

Стратегии тестирования:

1. Черный ящик – тестируем функции не думая, что там внутри
   1. Тестирование функций
   2. Тестирование классов входных данных
   3. Тестирование классов выходных данных
   4. Тестирование области допустимых значений (ОДЗ)
   5. Тестирование длины набора входных данных
      1. Нормальный набор
      2. Максимальный набор
      3. Пустой набор
      4. Выход за пределы массива
      5. Единичный набор
   6. Тестирование упорядоченности
      1. Тест неупорядоченного массива
      2. Тест упорядоченного массива
2. Белый ящик –

Практические задания:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тест | Ожидаемый | Полученный |
| T1 | N = 1 m = 2 | R1 = 3 m = 3 n = 1  R2 = true m = 4 n = 1  R3 = true m = 4 n = 2 | + |
| T2 | N = 5 m = 1 | R = 6 m = 2 n = 5  R2 = false m = 3 n = 5  R3 = true m = 3 n = 6 | + |
| T3 | N = “ABC” | Error | + |
| T4 | N = 1,23 | Error | + |
| T5 | X = 0 | Y = 1 | + |
| T6 | X = 1 | Error | + |
| T7 | X = “ABC” | Error | + |

Черный ящик

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Критерии | Т1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
| Классы вх. Данных | + | + |  |  |  |  |  |  |  |  |
| M = целое число | + | + |  |  |  |  |  |  |  |  |
| M – не целое число |  |  | + |  |  |  |  |  |  |  |
| N -целое число |  |  |  | + |  |  |  |  |  |  |
| N – не целое число |  |  |  |  |  |  |  |  |  |  |
| X != 1, -1 |  |  |  |  |  |  |  |  |  |  |
| X = 1 |  |  |  |  |  |  |  |  |  |  |
| X = -1 |  |  |  |  |  |  |  |  |  |  |
| X – не число |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

**Тесты**

**Arrange – начальные условия**

**Act – выполняет действия**

**Assert – класс методов которые проверяют на валидность данные результат действия**

**Обобщенные коллекции**

**System.Collections.Generic**

**Преимущества:**

1. **Производительность**
2. **Безопасность**
3. **Повторное использование двоичного кода**

**-**

**Stack<T> -**

**-**

**-**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | **Комментарий** |
| **Comparer <T>** | **Comparer** |  |  |
| **LinkedList <T>** |  | **Связный список** |  |
| **List<T>** | **ArrayList** | **Массив** |  |
| **Queue<T>** | **Queue** | **Массив** |  |
| **Stack<T>** | **Stack** | **Массив** |  |
| **Dictionary<T, V>** | **HashTable** | **Хеш-таблица** | **Нужно переопределить** |
| **HashSet<T>** | **HashTable** | **Хеш-таблица** |
| **SortedDictionary<T, V>** | **SortedList** | **Дерево поиска** | **Нужно переопределять метод compareTo()** |
| **SortedSet<T>** | **SortedList** | **Дерево поиска** |

**LinkedList<T>**

**Узел – LinkedListNode<T> - содержит ссылки на предыдущий и следующий элемент, а так же значение.**

**.Contains(Значение) возвращает true or false**

**.Count – возвращает число значений в коллекции**

**Dictionary<T, V>. Пример: Dictionary<Person, Student> -**

**Dict.Keys – коллекция ключей**

**Stopwatch – класс для замерки времени.**

**SortedDictionary<T,W>**

**Лабораторная работа 11 делается по 10.**

**Достаем отдельный объект а не ссылку! Можно через .Clone()**

**IInit IComparable ICloneable**

**Equals – с GetHashCode**

**Написать свой GetHashCode()**

**ICollection <T> coll– обобщенный класс с коллекциями с одним элементом**

**ICo**