МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра прикладной математики, механики, управления и программного обеспечения**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

по дисциплине «Структуры и алгоритмы компьютерной обработки данных»

Направление «Математическое обеспечение и администрирования информационных систем»

|  |
| --- |
| Выполнил студент гр. Б8204  \_\_\_\_\_\_\_\_\_\_Слипенчук Дмитрий |
| Проверил доцент, к.т.н.  \_\_\_\_\_\_\_\_\_\_\_ С.Н. Остроухова  (зачтено/не зачтено) |

г. Владивосток

2016

**1. Неформальная постановка задачи**

Разработать класс строка, представленный в виде односвязного односимвольного списка. Реализовать минимальный набор методов, для работы со строкой.

Класс должен содержать в себе следующие методы:

* Конструктор
* Деструктор
* Оператор присваивания
* Конструктор копирования
* Оператор конкатенации
* Печать строки
* Подсчет длины строки

Члены класса:

* elem\* head; // указатель на начало списка
* struct elem { char info; // символ elem\* pointer; // указатель }

**2. Спецификация методов класса**

***strClass();***

Конструктор создает пустой односвязный список.

***strClass (string str);***

Конструктор создает односвязный односимвольный список, который посимвольно совпадает со строкой str. Если строка str пуста, то создается пустой список.

***strClass (char a);***

Конструктор создает односвязный односимвольный список, содержащий один символ – a.

***strClass (const strClass & another);***

Конструктор копирования, т. е. исходному списку при создании присваиваются значения всех элементов списка another.

**~*strClass* ();**

Деструктор освобождает память, выделенную под список, созданный конструктором.

***int length*();**

Данный метод возвращает количество элементов односвязного списка.

**friend ostream &operator<<(ostream &out, const strClass &obj);**

Перегрузка потока, т.е вывод в поток всех элементов односвязного односимвольного списка obj.

strClass &operator=(strClass another)

Перегрузка присваивания. (исходному списку присваиваются значения всех элементов списка another. Если размер исходного списка превосходит размер списка another, то оставшиеся после копирования элементы исходного списка удаляются)

strClass &operator=(char a)

Перегрузка присваивания. (исходному списку присваивается значение символа a. Если размер исходного списка превосходит размер )

***int* *ahead*(int num);**

Данный метод добавляет элемент num в начало дека. Метод возвращает 1, если добавление произошло удачно, и 0, если дек заполнен.

***int* *atail*(int num);**

Данный метод добавляет элемент num в конец дека. Метод возвращает 1, если добавление произошло удачно, и 0, если дек заполнен.

***int* d*head*();**

Данный метод удаляет элемент из начала дека и возвращает удалённый элемент. Либо NULL, если дек пуст.

***int* *dtail*();**

Данный метод удаляет элемент из конца дека и возвращает удалённый элемент. Либо NULL, если дек пуст.

***int* *phead*();**

Данный метод выводит в терминал все элементы дека c начала до конца. Если дек пустой, то возвращает NULL.

***int* *ptail*();**

Данный метод выводит в терминал все элементы дека c конца до начала. Если дек пустой, то возвращает NULL.

***dequeue operator = (dequeue &another)*;**

Перегрузка присваивания (исходному деку присваиваются значения всех элементов дека another, с сохранением указателя на начало и конец, если размер another при этом не превосходит размер исходного дека. Если размер another превосходит размер исходного дека, то исходному деку присваиваются n элементов из another, начиная с начала)

**3. Тестирование**

**Тестирование метода *cdequeue*(*int* *size* = 10);**

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Входные данные** | **Результат** |
| Создание дека с явным заданием размера | Size = 5 | n = 5.  pointerL = 1;  pointerR = 0; |
| Создание дека с заранее заданным размером |  | n = 10.  pointerL = 1;  pointerR = 0; |
| Создание дека недопустимого размера | Size = -5 | n = 10.  pointerL = 1;  pointerR = 0; |

**Тестирование метода *int* *ahead*(*int num*);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный дек | Входные данные | Результат |
| Добавление элемента в пустой дек | n = 5  pointerL = 1;  pointerR = 0;  arr= 1,2,3,4,5 | num=7; | n = 5  pointerL = 0;  pointerR = 0;  arr=7,2,3,4,5  return=1 |
| Добавление элемента в заполненный дек | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5 | num=7; | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5  return=0 |
| Добавление элемента в заполненный до начала дек, но со свободным местом в конце | n = 5  pointerL = 0;  pointerR = 3;  arr= 1,2,3,4,5 | num=7; | n = 5  pointerL = 0;  pointerR = 4;  arr= 7,1,2,3,4  return=1 |
| Добавление элемента в незаполненный дек | n = 5  pointerL = 2;  pointerR = 3;  arr= 1,2,3,4,5 | Num = 7; | n = 5  pointerL = 1;  pointerR = 3;  arr= 1,7,3,4,5  return=1 |

**Тестирование метода *bool* *atail*(*int num*);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный дек | Входные данные | Результат |
| Добавление элемента в пустой дек | n = 5  pointerL = 1;  pointerR = 0;  arr= 1,2,3,4,5 | num=7; | n = 5  pointerL = 1;  pointerR = 1;  arr=1,7,3,4,5  return=1 |
| Добавление элемента в заполненный дек | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5 | num=7; | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5  return=0 |
| Добавление элемента в заполненный до конца дек, но со свободным местом в начале | n = 5  pointerL = 1;  pointerR = 4;  arr= 1,2,3,4,5 | num=7; | n = 5  pointerL = 0;  pointerR = 4;  arr= 2,3,4,5,7  return=1 |
| Добавление элемента в незаполненный дек | n = 5  pointerL = 0;  pointerR = 2;  arr= 1,2,3,4,5 | Num = 7; | n = 5  pointerL = 0;  pointerR = 3;  arr= 1,2,3,7,5  return=1 |

**Тестирование метода *int* *dhead*();**

|  |  |  |
| --- | --- | --- |
| Описание теста | Исходный дек | Результат |
| Удаление элемента в пустом деке | Пустой дек | NULL |
| Удаление элемента в заполненном деке | n = 5  pointerL = 0;  pointerR = 2;  arr= 1,2,3,4,5 | n = 5  pointerL = 1;  pointerR = 2;  arr= 1,2,3,4,5  return 1 |

**Тестирование метода *int* *dtail*();**

|  |  |  |
| --- | --- | --- |
| Описание теста | Исходный дек | Результат |
| Удаление элемента в пустом деке | Пустой дек | NULL |
| Удаление элемента в заполненном деке | n = 5  pointerL = 0;  pointerR = 2;  arr= 1,2,3,4,5 | n = 5  pointerL = 0;  pointerR = 1;  arr= 1,2,3,4,5  return 3 |
| Удаление элемента в деке с одним жлементом | n = 5  pointerL = 2;  pointerR = 2;  arr= 1,2,3,4,5 | n = 5  pointerL = 2;  pointerR = 1;  arr= 1,2,3,4,5  return 3 |

**Тестирование метода *int* *phead*();**

|  |  |  |
| --- | --- | --- |
| Описание теста | Входные данные | Результат |
| Печать непустого дека | n = 7  pointerL = 3;  pointerR = 6;  arr= 0,1,2,229,230,231,232 | 3 : 229 head  4 : 230  5 : 231  6 : 232 tail |
| Печать пустого дека | Пустой дек | NULL |

**Тестирование метода *int*  *ptail*(*)***

|  |  |  |
| --- | --- | --- |
| Описание теста | Входные данные | Результат |
| Печать непустого дека | n = 7  pointerL = 3;  pointerR = 6;  arr= 0,1,2,229,230,231,232 | 6 : 232 tail  5 : 231  4 : 230  3 : 229 head |
| Печать пустого дека | Пустой дек | NULL |

**Тестирование метода *cdequeue* (*cdequeue* &*another*);**

|  |  |  |
| --- | --- | --- |
| Описание теста | another | Результат this |
| Копирование пустого дека | n = 5  pointerL = 1;  pointerR = 0;  arr= 1,2,3,4,5 | n = 5  pointerL = 1;  pointerR = 0;  arr= 1,2,3,4,5 |
| Копирование незаполненного дека | n = 5  pointerL = 0;  pointerR = 2;  arr= 1,2,3,4,5 | n = 5  pointerL = 0;  pointerR = 2;  arr= 1,2,3,4,5 |
| Копирование заполненного дека | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5 | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5 |

**Тестирование метода *dequeue operator = (dequeue &another)*;**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | this | another | Результат |
| Присваивание заполненного дека | n = 7  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5,6,7 | n = 7  pointerL = 0;  pointerR = 6;  arr= 1,2,3,4,5,6,7 | n = 7  pointerL = 0;  pointerR = 6;  arr= 1,2,3,4,5,6,7 |
| Присваивание меньшему деку большего | n = 5  pointerL = 1;  pointerR = 4;  arr= 1,2,3,4,5 | n = 10  pointerL = 0;  pointerR = 9;  arr= 1,2,3,4,5,6,7,8,9,10 | n = 5  pointerL = 0;  pointerR = 4;  arr= 1,2,3,4,5 |
| Присваивание большему меньшего | n = 10  pointerL = 1;  pointerR = 4;  arr= 1,2,3,4,5,6,7,8,9,10 | n = 5  pointerL = 1;  pointerR = 4;  arr= 6,7,8,9,10 | n = 10  pointerL = 1;  pointerR = 4;  arr= 6,7,8,9,10,6,7,8,9,10 |
| Самоприсваивание | n = 10  pointerL = 1;  pointerR = 4;  arr= 1,2,3,4,5,6,7,8,9,10 | n = 10  pointerL = 1;  pointerR = 4;  arr= 1,2,3,4,5,6,7,8,9,10 | n = 10  pointerL = 1;  pointerR = 4;  arr= 1,2,3,4,5,6,7,8,9,10 |

**Тестирование метода *int length*();**

|  |  |  |
| --- | --- | --- |
| Описание теста | Входные данные | Результат |
| Вывод длины заполненного дека | n = 7  pointerL = 0;  pointerR = 6;  arr= 565,567,67,229,230,231,232 | *7* |
| Вывод длины пустого дека | n = 7  pointerL = 3;  pointerR = 2;  arr= 565,567,67,229,230,231,232 | *0* |