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
* Оператор конкатенации
* Печать строки
* Подсчет длины строки
* Поиск подстроки
* Копирование подстроки с определенного символа

Члены класса:

* elem\* head; // указатель на первый элемент списка
* elem \*tail; // указатель на последний элемент списка
* struct elem { char info; // символ elem\* pointer; // указатель }

**2. Спецификация методов класса**

***strClass(); - Создание пустой строки.***

Конструктор создает пустой односвязный список.

***strClass (string str); Создание строки, состоящей из строки str.***

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

***strClass (char a); Создание строки, состоящей из символа a***

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

***strClass (const strClass & another); Создание строки, состоящей из строки another***

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

**~*strClass* (); Освобождение памяти из-под исходной строки**

Деструктор освобождает память, выделенную под список, созданный конструктором.

***int length*(); Возвращение длины исходной строки**

Данный метод возвращает количество элементов односвязного списка.

**friend ostream &operator<<(ostream &out, const strClass &obj); Вывод исходной строки в поток**

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

**strClass &operator=(strClass another) Присваивание исходной строке значения строки 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) Присваивание исходной строке значения символа a**

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

**strClass &operator=(string str) Присваивание исходной строке значения строки string**

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

**strClass operator+(char a) Конкатенация исходной строки и символа a**

Конкатенация.(в конец исходного списка добавляется символ a)

**strClass operator+(string str) Конкатенация исходной строки и строки str**

Конкатенация.(в конец исходного списка посимвольно добавляется строка str)

**friend strClass operator+(string str, strClass& obj) Конкатенация строки str и строки obj**

Конкатенация. (в конец строки str посимвольно добавляются все элементы списка obj)

**strClass operator+(strClass &another) Конкатенация исходной строки и строки another**

Конкатенация. (в конец исходного списка посимвольно добавляются все элементы списка another)

**strClass Copy(int a, int len) Копирование len символов исходной строки, начиная с a-го символа**

Данный метод возвращает часть исходного односвязного списка начиная с а-го элемента и заканчивая (a+len) элементом. Если (a+len) элемент находится за пределами списка, то по конец исходного списка.

**int search(string needle) Поиск первого вхождения строки needle в исходную строку**

Данный метод ищет первое вхождение строки needle в исходный список и возвращает индекс первого вхождения, либо -1, если строка не найдена

**int search(strClass needle) Поиск первого вхождения строки needle в исходную строку**

Данный метод ищет первое вхождение списка needle в исходный список и возвращает индекс первого вхождения, либо -1, если список не найден

**3. Тестирование**

**Тестирование метода *strClass*();**

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Входные данные** | **Результат** |
| Создание пустого списка |  | head = NULL;  tail = NULL; |

**Тестирование метода *strClass*(string str);**

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Входные данные** | **Результат** |
| Создание списка с явным присваиванием ему значения строки str. | Str = “AB” | head = “A”;  tail = “B”; |

**Тестирование метода *strClass*(char a);**

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Входные данные** | **Результат** |
| Создание списка с явным присваиванием ему значения символа a. | a = “h” | head = “h”;  tail = “h”; |

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

|  |  |  |
| --- | --- | --- |
| **Описание теста** | **Входные данные** | **Результат** |
| Конструктор копирования. | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “H”->”E”->”L”->”L” ->tail = “O”; |

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

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Добавление списка another в пустой исходный список | head = NULL  tail = NULL | head = “H”->”E”->”L” ->”L” ->tail = “O”; | head = “H”->”E”->”L” ->”L” ->tail = “O”; |
| Добавление списка another в исходный список большей длины. | head = “H”->”E”->”L”->”L” ->”O”->” ” -> ”W”->”O” ->”R”->”L”-> tail = “D”; | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “H”->”E”->”L” ->”L” ->tail = “O”; |
| Добавление списка another в исходный список меньшей длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “H”->”E”->”L”->”L” ->”O”->” ” -> ”W”->”O” ->”R”->”L”-> tail = “D”; | head = “H”->”E”->”L”->”L” ->”O”->” ” -> ”W”->”O” ->”R”->”L”-> tail = “D”; |
| Добавление списка another в исходный список такой же длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “H”->”E”->”L”->”L” ->tail = “O”; |

**Тестирование метода strClass &operator=(char a);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный дек | Входные данные | Результат |
| Добавление символа “a” в пустой исходный список | head = NULL  tail = NULL | Char = ‘H’ | head = “H”;  tail = “H”; |
| Добавление символа “a” в исходный список длины больше 1. | head = “H”->”E”->”L”->”L” ->tail = “O”; | Char = ‘H’ | head = “H”;  tail = “H”; |
| Добавление символа “a” в исходный список длины 1. | head = “W”;  tail = “W”; | Char = ‘H’ | head = “H”;  tail = “H”; |

**Тестирование метода strClass &operator=(string str);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный дек | Входные данные | Результат |
| Добавление строки str в пустой исходный список | head = NULL  tail = NULL | Str = “HELLO” | head = “H”->”E”->”L” ->”L” ->tail = “O”; |
| Добавление строки str в исходный список большей длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | Str = “W” | head = “W”;  tail = “W”; |
| Добавление строки str в исходный список меньшей длины. | head = “W”;  tail = “W”; | Str = “HELLO” | head = “H”->”E”->”L”->”L” ->tail = “O”; |
| Добавление строки str в исходный список такой же длины. | head = “W”->”O”->”R”->”L” ->tail = “D”; | Str = “HELLO” | head = “H”->”E”->”L”->”L” ->tail = “O”; |

**Тестирование метода strClass operator+(char a);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Конкатенация c пустым списком. | head = NULL  tail = NULL | Char = ‘W’ | head = “W”;  tail = “W”; |
| Конкатенация cо списком длины 1. | head = “W”;  tail = “W”; | Char = ‘O’ | head = “W”;  tail = “O”; |
| Конкатенация cо списком длины больше 1. | head = “H”->”E”->”L”-> tail = “L”; | Char = ‘O’ | head = “H”->”E”->”L”->”L” ->tail = “O”; |

**Тестирование метода strClass operator+(string a);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Конкатенация c пустым списком. | head = NULL  tail = NULL | Str = “HELLO” | head = “H”->”E”->”L”->”L” ->tail = “O”; |
| Конкатенация с пустой строкой | head = “H”->”E”->”L”->”L” ->tail = “O”; | Str = “” | head = “H”->”E”->”L”->”L” ->tail = “O”; |
| Конкатенация cо списком большей длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | Str = “!” | head = “H”->”E”->”L”->”L” -> “O”-> tail = “!”; |
| Конкатенация cо списком меньшей длины. | head = “W”;  tail = “W”; | Str = “ORLD” | head = “W”->”O”->”R”->”L” ->tail = “D”; |
| Конкатенация cо списком такой же длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | Str = “WORLD” | head = “H”->”E”->”L”->”L” ->”O” -> ”W”->”O” ->”R”->”L”-> tail = “D”; |

**Тестирование метода strClass operator+(strClass &another);**

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Конкатенация c пустым исходным списком. | head = NULL  tail = NULL | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “H”->”E”->”L”->”L” ->tail = “O”; |
| Конкатенация с пустым списком another | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = NULL  tail = NULL | head = “H”->”E”->”L”->”L” ->tail = “O”; |
| Конкатенация cо списком большей длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “!”;  tail = “!”; | head = “H”->”E”->”L”->”L” -> “O”-> tail = “!”; |
| Конкатенация cо списком меньшей длины. | head = “W”;  tail = “W”; | head = O”->”R”->”L” ->tail = “D”; | head = “W”->”O”->”R”->”L” ->tail = “D”; |
| Конкатенация cо списком такой же длины. | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “W”->”O”->”R”->”L” ->tail = “D”; | head = “H”->”E”->”L”->”L” ->”O” -> ”W”->”O” ->”R”->”L”-> tail = “D”; |

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

|  |  |  |
| --- | --- | --- |
| Описание теста | Исходный дек | Результат |
| Вывод пустого списка | head = NULL  tail = NULL | “” |
| Вывод непустого списка | head = “H”->”E”->”L”->”L” ->tail = “O”; | “HELLO” |

**Тестирование метода *strClass Copy(int a, int len);***

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Копирование из списка длины меньше **a**. | head = “H”->”E”->”L”->”L” ->tail = “O”; | a = 6  len = 3 | head = NULL  tail = NULL |
| Копирование из списка длины больше **a** и меньше **a + len.** | head = “H”->”E”->”L”->”L” ->tail = “O”; | a = 6  len = 3 | head = L”->”L” ->tail = “O”; |
| Копирование из списка длины больше **a + len.** | head = “H”->”E”->”L”->”L” ->tail = “O”; | a = 1  len = 3 | head = “H”->”E”-> tail = ”L” |

**Тестирование метода *int search(string needle);***

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Поиск в пустом списке | head = NULL  tail = NULL | needle = “HELLO” | -1 |
| Поиск пустой подстроки | head = “H”->”E”->”L”->”L” ->tail = “O”; | needle = “” | -1 |
| Поиск подстроки, не входящей в исходный список | head = “H”->”E”->”L”->”L” ->tail = “O”; | needle = “OLLEH” | -1 |
| Поиск подстроки, входящей в исходный список | head = “H”->”E”->”L”->”L” ->tail = “O”; | needle = “ELLO” | 1 |

**Тестирование метода *int search(strClass needle);***

|  |  |  |  |
| --- | --- | --- | --- |
| Описание теста | Исходный список | Входные данные | Результат |
| Поиск в пустом списке | head = NULL  tail = NULL | head = “H”->”E”->”L”->”L” ->tail = “O”; | -1 |
| Поиск пустой подстроки | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = NULL  tail = NULL | -1 |
| Поиск подстроки, не входящей в исходный список | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = “O”->”L”->”L”->”E” ->tail = “H”; | -1 |
| Поиск подстроки, входящей в исходный список | head = “H”->”E”->”L”->”L” ->tail = “O”; | head = ”E”->”L”->”L” ->tail = “O”; | 1 |

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

|  |  |  |
| --- | --- | --- |
| Описание теста | Исходный список | Результат |
| Длина пустого списка | head = NULL  tail = NULL | 0 |
| Длина непустого списка | head = “H”->”E”->”L”->”L” ->tail = “O”; | 5 |