

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК  
  
Кафедра прикладной математики, механики, управления и программного обеспечения**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7 ПО ДИСЦИПЛИНЕ  
«СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ»**

Направление 02.03.03 «Математическое обеспечение и администрирование  
информационных систем»

* Выполнил студент гр. Б8204  
  Проскурин Денис Александрович  
  Проверил:  
  Доцент, к.т.н С.Н.Остроухова

Владивосток 2019

**Неформальная постановка задачи**

Реализовать класс строки, который представлен многосимвольными звеньями(пробеллы сохраняются в свои собственные звенья) переменной длины, одно слово- одно звено , на основе двусвязного списка, имеющий следующие методы:

1. Конструктор
2. Деструктор
3. Конструктор копирования
4. Перегрузка присваивания
5. Перегрузка вывода
6. Перегрузка +
7. Длина(кол-во символов)
8. Позиция подстроки в строке
9. Выделение подстроки в строке
10. Удаление подстроки в строке

**Спецификация методов класса**

**Multi\_character\_links() : Head(nullptr), Tail(nullptr)** — конструктор.

**~Multi\_character\_links()** - деструктор.

**Multi\_character\_links(const Multi\_character\_links &obj)**- конструктор копирования. Создаёт новую строку на основе другой строки obj.

**Multi\_character\_links(const char \*const data)**-конструктор копирования. Создаёт новую строку на основе строковой константы const char\* const data.

**friend std::ostream &operator<<(std::ostream &out, const Multi\_character\_links &obj)-** выводит строку obj класса Multi\_character\_links в стандартный поток вывода(весь двухъсвязный список со звеньями переменной длины от HEAD до TAIL).

**Multi\_character\_links &operator=(Multi\_character\_links &data)-** строке this присваевается строка data класса.Допустимо множественное присваивание.

**Multi\_character\_links &operator=(const char \*const data)**-строке this присваевается строка, созданнная на основе строковой константы const char \*const data.Недопустимо множественное присваивание.

**Multi\_character\_links &operator+(Multi\_character\_links &data)-**возвращает строку, которая является результатом конкатенации строк this и data.

**Multi\_character\_links &operator+(const char \*const data)-**возвращает строку,которая является результатом конкатенации строк this и строки, созданнной на основе строковой константы const char \*const data.

**unsigned int length()**-возвращает длину исходной строки.

**friend Multi\_character\_links &operator+(const char \*const data, Multi\_character\_links &obj)-**возвращает строку,которая является результатом конкатенации строковой константы const char \*const data и строки obj.

**int pos(Multi\_character\_links &sub)**-возвращает позицию подстроки sub в строке this. Если данная подстрока не найдена, возвращает -1;

**Multi\_character\_links &substr(unsigned int k, unsigned int n)**-возвращает подстроку с k-го по n-ый символ данной сроки. Если k>n или k>length, то возвращает пустую строку.

**Multi\_character\_links &del(unsigned int k, unsigned int n)**-возвращает строку, которая получается в результате удаления из исходной строки подстроки с k-го по n-ый символ.Если k>n или k>length, возвращает исходную строку.

Void change(**Multi\_character\_links &obj1,Multi\_character\_links &obj2,Multi\_character\_links &****obj3)-**заменяет в строке obj1 все вхождения подстроки obj2 на строки obj3, при условии что obj2 не содержится в obj3. Если obj2 не найдено, данная строка остаётся нетронутой.

**Тесты**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **Тестирование конструктора класса** | |
| Multi\_character\_links example; | HEAD TAIL   |  | | --- | |  | |
| **Тестирование конструктора-копирования класса**  **Multi\_character\_links(const char \*const data)** | |
| Multi\_character\_links example("Hello my dear teacher"); | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Hello | \_ | my | \_ | dear | \_ | teacher | |
| Multi\_character\_links example("Hello my dear teacher"); | HEAD TAIL   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | \_\_\_\_\_ | Hello | \_ | my | \_ | dear | \_ | teacher | |
| Multi\_character\_links example("\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); | HEAD TAIL   |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| **Тестирование конструктора-копирования класса**  **Multi\_character\_links(const Multi\_character\_links &obj)** | |
| Multi\_character\_links example("Hello my dear teacher");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Hello | \_ | my | \_ | dear | \_ | teacher |   Multi\_character\_links example2(example); | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Hello | \_ | my | \_ | dear | \_ | teacher | |
| Multi\_character\_links example("\_\_\_\_Hello\_\_\_\_");  HEAD TAIL   |  |  |  | | --- | --- | --- | | \_\_\_\_ | Hello | \_\_\_\_ |   Multi\_character\_links example2(example); | HEAD TAIL   |  |  |  | | --- | --- | --- | | \_\_\_\_ | Hello | \_\_\_\_ | |
| Multi\_character\_links example("\_\_\_\_\_\_\_\_");  HEAD TAIL   |  | | --- | | \_\_\_\_\_\_\_\_ |   Multi\_character\_links example2(example); | HEAD TAIL   |  | | --- | | \_\_\_\_\_\_\_\_ | |
| **Тестирование метода**  **friend std::ostream &operator<<(std::ostream &out, const Multi\_character\_links &obj)** | |
| Multi\_character\_links example("Hello my dear teacher");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Hello | \_ | my | \_ | dear | \_ | teacher | | Hello my dear teacher |
| Multi\_character\_links example("\_\_\_\_\_\_\_\_");  HEAD TAIL   |  | | --- | | \_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_ |
| Multi\_character\_links example("\_\_\_\_Hello\_\_\_\_");  HEAD TAIL   |  |  |  | | --- | --- | --- | | \_\_\_\_ | Hello | \_\_\_\_ | | \_\_\_\_Hello\_\_\_\_ |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  | |  |
| **Тестирование метода**  **Multi\_character\_links &operator=(Multi\_character\_links &data)** | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2("I am second constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | constructor |   example = example2; | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | constructor | |
| Multi\_character\_links example("I am first constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | constructor |   Multi\_character\_links example2("I am second constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | Constructor |   example = example2; | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | Constructor | |
| Multi\_character\_links example("I am first constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | constructor |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   example = example2; | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   example = example2; | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2("\_\_\_\_Hello\_\_\_\_");  HEAD TAIL   |  |  |  | | --- | --- | --- | | \_\_\_\_ | Hello | \_\_\_\_ |   example = example2; | HEAD TAIL   |  |  |  | | --- | --- | --- | | \_\_\_\_ | Hello | \_\_\_\_ | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2("\_\_\_\_Hello\_\_\_\_");  HEAD TAIL   |  | | --- | | \_\_\_\_\_\_\_\_ |   example = example2; | HEAD TAIL   |  | | --- | | \_\_\_\_\_\_\_\_ |   example = example2; |
| Multi\_character\_links example("I am first constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | constructor |   Multi\_character\_links example2("I am second constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | Constructor |   Multi\_character\_links example3("I am fird constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | Constructor |   example = example2 = example3; | example2 имеет вид:  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | Constructor |   Example имеет вид:  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | Constructor | |
| Multi\_character\_links example("I am first constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | constructor |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example3("I am fird constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | Constructor |   example = example2 = example3; | example2 имеет вид:  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | Constructor |   Example имеет вид:  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | Constructor | |
| Multi\_character\_links example("I am first constructor");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | constructor |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example3("I am fird constructor");  HEAD TAIL   |  | | --- | |  |     example = example2 = example3; | example2 имеет вид:  HEAD TAIL   |  | | --- | |  |   Example имеет вид:  HEAD TAIL   |  | | --- | |  | |
| **Тестирование метода**  **Multi\_character\_links &operator=(const char \*const data)** | |
| Multi\_character\_links example="I am example const string ="; | HEAD TAIL   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | example | \_ | const | \_ | string | \_ | «=» | |
| Multi\_character\_links example=""; | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example="\_\_\_\_"; | HEAD TAIL   |  | | --- | | \_\_\_\_ | |
| Multi\_character\_links example="\_\_\_\_Hello\_\_\_\_"; | HEAD TAIL   |  |  |  | | --- | --- | --- | | \_\_\_\_ | Hello | \_\_\_\_ | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example3;  HEAD TAIL   |  | | --- | |  |   example = example2 = example3 = "hello"; | example имеет вид:  HEAD TAIL   |  | | --- | | Hello |   example2 имеет вид:  HEAD TAIL   |  | | --- | | Hello |   example3 имеет вид:  HEAD TAIL   |  | | --- | | Hello | |
| **Тестирование метода**  **Multi\_character\_links &operator+(Multi\_character\_links &data)** | |
| Multi\_character\_links example("I am first example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | example |   Multi\_character\_links example2("I am second example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | example |   Multi\_character\_links example3("I am fird example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | example |   example = example2 + example3; | HEAD   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | example |   TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | example | |
| Multi\_character\_links example("I am first example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | first | \_ | example |   Multi\_character\_links example2("I am second example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | example |   Multi\_character\_links example3("I am fird example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | example |   example = example3 + example2; | HEAD   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | example |   TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | second | \_ | example | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example3("I am fird example");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | example |   example = example2 + example3; | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | I | \_ | am | \_ | fird | \_ | example | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | |  |   Multi\_character\_links example3;  HEAD TAIL   |  | | --- | |  |   Multi\_chara  example = example2 + example3; | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | | \_\_\_\_ |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | | \_\_\_\_ |   Multi\_character\_links example3;  HEAD TAIL   |  | | --- | | \_\_\_\_ |   Multi\_chara  example = example2 + example3; | HEAD TAIL   |  |  | | --- | --- | | \_\_\_\_ | \_\_\_\_ | |
| Multi\_character\_links example;  HEAD TAIL   |  | | --- | | first |   Multi\_character\_links example2;  HEAD TAIL   |  | | --- | | second |   example = example + example2 + example; | HEAD TAIL   |  |  |  | | --- | --- | --- | | first | second | first | |
| **Тестирование метода**  **Multi\_character\_links &operator+(const char \*const data)** | |
| Multi\_character\_links example("first");  HEAD TAIL   |  | | --- | | first |   example = example + "second" + "third"; | HEAD TAIL   |  |  |  | | --- | --- | --- | | first | second | third | |
| Multi\_character\_links example("first");  HEAD TAIL   |  | | --- | | first |   example = example + "\_\_\_\_"; | HEAD TAIL   |  |  | | --- | --- | | first | \_\_\_\_ | |
| Multi\_character\_links example("first");  HEAD TAIL   |  | | --- | | first |   example = example + ""; | HEAD TAIL   |  | | --- | | first | |
| **Тестирование метода**  **friend Multi\_character\_links &operator+(const char \*const data, Multi\_character\_links &obj)** | |
| Multi\_character\_links example("first");  HEAD TAIL   |  | | --- | | first |   example = "third" + ("second" + example); | HEAD TAIL   |  |  |  | | --- | --- | --- | | third | second | firts | |
| Multi\_character\_links example("first");  HEAD TAIL   |  | | --- | | first |   example = "\_\_\_\_" + example; | HEAD TAIL   |  |  | | --- | --- | | \_\_\_\_ | first | |
| Multi\_character\_links example("first");  HEAD TAIL   |  | | --- | | first |   example = "" + example; | HEAD TAIL   |  | | --- | | first | |
| **Тестирование метода length()** | |
| Multi\_character\_links example("first example for you");  HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | first | \_ | example | \_ | for | \_ | you |   Multi\_character\_links example2("second");  HEAD TAIL   |  | | --- | | second |   Multi\_character\_links example3;  HEAD TAIL   |  | | --- | |  | | 21  6  0 |
| **Тестирование метода**  **int pos(Multi\_character\_links &sub)** | |
| Multi\_character\_links example("first example for you");  Multi\_character\_links example2("for");  example.pos(example2) | 14 |
| Multi\_character\_links example("first example for you");  example.pos(example) | 0 |
| Multi\_character\_links example("first example for you");  Multi\_character\_links example2("god");  example.pos(example) | -1 |
| Multi\_character\_links example("first example for you");  Multi\_character\_links example2("e");  example.pos(example2) | 6 |
| **Тестирование метода**  **Multi\_character\_links &substr(unsigned int k, unsigned int n)** | |
| Multi\_character\_links example("first example for you");  example.substr(0,5); | HEAD TAIL   |  |  | | --- | --- | | first | \_ | |
| Multi\_character\_links example("first example for you");  example.substr(0,20); | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | first | \_ | example | \_ | for | \_ | you | |
| Multi\_character\_links example("first example for you");  example.substr(0,40); | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | first | \_ | example | \_ | for | \_ | you | |
| Multi\_character\_links example("first example for you");  example.substr(10,20); | HEAD TAIL   |  |  |  |  |  | | --- | --- | --- | --- | --- | | ple | \_ | for | \_ | you | |
| Multi\_character\_links example("first example for you");  example.substr(10,40); | HEAD TAIL   |  |  |  |  |  | | --- | --- | --- | --- | --- | | ple | \_ | for | \_ | you | |
| Multi\_character\_links example("first example for you");  example.substr(10,10); | HEAD TAIL   |  | | --- | | p | |
| Multi\_character\_links example("first example for you");  example.substr(30,30); | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example("first example for you");  example.substr(20,10); | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example("first example for you");  example.substr(21,21).print(); | HEAD TAIL   |  | | --- | |  | |
| **Тестирование метода**  **Multi\_character\_links &substr(unsigned int k, unsigned int n)** | |
| Multi\_character\_links example("first example for you");  example.del(0,10).print(); | HEAD TAIL   |  |  |  |  |  | | --- | --- | --- | --- | --- | | le | \_ | for |  | you | |
| Multi\_character\_links example("first example for you");  example.del(10,10).print(); | HEAD TAIL   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | first | \_ | exam | le | \_ | for | \_ | you | |
| Multi\_character\_links example("first example for you");  example.del(0,20).print(); | HEAD TAIL   |  | | --- | |  | |
| Multi\_character\_links example("first example for you");  example.del(10,20).print(); | HEAD TAIL   |  |  |  | | --- | --- | --- | | first | \_ | exam | |
| Multi\_character\_links example("first example for you");  example.del(20,10).print(); | HEAD TAIL   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | first | \_ | example | \_ | for | \_ | you | |