Лабораторная работа по теме «класс-контейнер string»

Лабораторная работа по курсу «Инструментальные средства и технологии программирования» Тема: «Класс-контейнер STL string»

полная документация класса-контейнера string
http://www.cplusplus.com/reference/string/basic_string

Итераторы string:
begin() - итератор на первый символа;
end() - итератор на позицию за последним символом;
rbegin() - итератор на последний элемент (для обратных алгоритмов);
rend() - итератор на позицию перед первым элементом (для обратных алгоритмов);
Размер строки:
size(), **length**() - размер строки;
empty() – проверка на пустоту (возвращает true, если пуская строка)
Шаблонные функции для string:
copy(char* buf, size_type buf_size) const - копировать buf_size символов в буфер;
copy(char* buf, size_type buf_size, size_type pos) const - копировать buf_size символов с указанной

Доступ к элементу строки

- ***at***(size_type pos) const - доступ к символу в позиции pos;
- reference ***at***(size_type pos) - доступ к символу в позиции pos.

Если генерация исключения не нужна при некорректном аргументе pos, то можно воспользоваться операцией [].

Добавление в конец строки

- ***append***(const basic_string& s) - доб. stl строки;
- ***append***(const basic_string& s, size_type pos, size_type npos) - доб. части stl строки;
- ***append***(const charT* s) - доб. C строки;
- ***append***(const charT* s, size_type n) - доб. части C строки;
- ***append***(size_type n, charT c) - доб. n символов c;
- ***append***(InputIterator first, InputIterator last) - доб. строки заданной итераторами.

Методы возвращают ссылку на себя (*this). В качестве альтернативы методам с одним аргументом можно воспользоваться операцией +=.

Назначение данных строке

- ***assign***(const basic_string& s) - присв. stl строки;

- ***assign*** (const basic_string& s,size_type pos, size_type n) - присв. части stl строки;
- ***assign*** (const charT* s) - присв. С строки;
- ***assign*** (const charT* s, size_type n) - присв. части С строки;
- ***assign*** (size_type n, charT c) - присв. n символов c;
- ***assign*** (InputIterator first, InputIterator last) - присв. строки заданной итераторами.

Методы возвращают ссылку на себя (*this). В качестве альтернативы методам с одним аргументом можно воспользоваться операцией =.

Сравнение строк

- ***compare*** (const basic_string& str) - сравнение с stl строкой;
- ***compare*** (size_type pos1, size_type n1, const basic_string& str) const - срав. с частью stl строки;
- ***compare*** (size_type pos1,size_type n1,const basic_string& str,size_type pos2,size_type n2) const - срав. части stl строки с частью stl строки;
- ***compare*** (charT* s) const - срав. с С строкой;
- ***compare*** (size_type pos, size_type n1, charT* s) const - срав. с С строкой;
- ***compare*** (size_type pos, size_type n1, charT* s, size_type n2) const - срав. с С строкой.

Методы сравнения возвращают следующие значения:

- 0 - строки равны;
- <0 - строка лексиграфически меньше со сравниваемой строкой;
- >0 - строка лексиграфически больше со сравниваемой строкой.

В качестве альтернативы методам с одним аргументом можно воспользоваться операциями ==, !=, < >, <=, >=.

Вставка данных

- ***insert*** (size_type pos1, const basic_string& s) - вставка stl строки;
- ***insert*** (size_type pos,const basic_string& s,size_type pos2=0,size_type n=npos) - вст. части stl строки;
- ***insert*** (size_type pos, const charT* s) - вст. С строки;
- ***insert*** (size_type pos, const charT* s, size_type n) - вст. части С строки;
- ***insert*** (size_type pos, size_type n, charT c) - вст. нескольких одинаковых символов.

Вставка данных, позиция вставки указана итератором:

- ***insert*** (iterator p, charT) - вст. символа;
- ***insert*** (iterator p, size_type n, charT c) - вст. нескольких одинаковых символов;
- ***insert*** (iterator p,InputIterator f,InputIterator l) - вст. строки заданной итераторами.

Большинство методов возвращают ссылку на себя (*this).

замена части строки

Замена участка строки, указанного позицией и размером:

- ***replace*** (size_type pos,size_type n1, const basic_string& s) - замена stl строкой;

- ***replace*** (size_type pos1,size_type n1,const basic_string& str,size_type pos2,size_type n2) - замена частью stl строкой;
- ***replace*** (size_type pos, size_type n1, const charT* s) - замена C строкой;
- ***replace*** (size_type pos,size_type n1,const charT* s,size_type n2) - замена частью C строки;
- ***replace*** (size_type pos, size_type n1, size_type n2, charT c) - замена несколькими символами.

Замена участка, указанного итераторами:

- ***replace*** (iterator i1, iterator i2,const basic_string& str) - замена stl строкой;
- ***replace*** (iterator i1, iterator i2, const charT* s) - замена C строкой;
- ***replace*** (iterator i1, iterator i2, const charT* s,size_type n) - замена частью C строки;
- ***replace*** (iterator i1, iterator i2, size_type n,charT c) - замена несколькими символами;
- ***replace*** (iterator i1, iterator i2,InputIterator j1, InputIterator j2) - замена строкой заданной итераторами.

Методы возвращают ссылку на себя (*this).

Поиск указанного элемента

- ***find*** (const string& str) const - поиск stl строки;
- ***find*** (const string& str, size_type idx) const - поиск stl строки с указанной позиции;
- ***find*** (const charT* s, size_type pos = 0) const - поиск C строки с указанной позиции;
- ***find*** (const charT* s, size_type pos, size_type n) const - поиск подстроки с указанной позиции;
- ***find*** (charT c, size_type pos = 0) const - поиск символа с указанной позиции.

Find возвращает номер позиции, с которой начинается подстрока. Если подстрока не найдена, то возвращается число больше, чем длина строки

Также есть аналогичные методы rfind, делающие поиск справа налево. Методы возвращают позицию найденного элемента или npos (обычно равной -1).

Поиск символа не входящего в строку

- ***find_first_not_of*** (const basic_string& str, size_type pos = 0) const;
- ***find_first_not_of*** (const charT* s,size_type pos,size_type n) const;
- ***find_first_not_of*** (const charT* s,size_type pos=0) const;
- ***find_first_not_of*** (charT c, size_type pos = 0) const.

Методы возвращают позицию найденного символа или npos (-1).

Поиск символа входящего в указанную строку:

- ***find_first_of*** (const basic_string& str,size_type pos=0) const;
- ***find_first_of*** (const charT* s,size_type pos,size_type n) const;
- ***find_first_of*** (const charT* s, size_type pos=0) const;
- ***find_first_of*** (charT c, size_type pos=0) const;

Методы возвращают позицию найденного символа или npos (-1).

Удаление символов строки

- ***erase*** (size_type pos=0,size_type n=npos) - удаляет n символов с указанной позиции;
- ***erase*** (iterator p) - удаляет один символ в указанной позиции;
- ***erase*** (iterator f, iterator l) - удаляет символы с позиции f по l.

Конкатенация строк

str1 + str2, str3 += str2 – конкатенация (сцепление) строк

Задания

Вариант 1.

Дан массив слов и подстрока. Сформировать предложение из слов, содержащих заданную подстроку не более двух раз (каждая буква может входить только в одну подстроку), поменяв предварительно во всех словах с четной длиной символы попарно местами. Слова в предложении должны быть отсортированы по возрастанию длины слов.

Вариант 2.

Дан массив слов и две подстроки. Преобразовать массив слов, заменив во всех словах первое вхождение первой подстроки на вторую подстроку. Сформировать предложение из слов, в составе которых есть цифры, предварительно добавив к слову это же перевернутое слово (например, слово “ab9cd” должно войти в предложение в виде “ab9cddc9ba”).

Вариант 3.

Дан массив слов и две подстроки. Удалить из всех слов массива последнее вхождение второй подстроки. Сформировать два предложения из полученных слов массива. В первое предложение должны войти слова, длина которых четная, и которые имеют в своем составе перевернутую первую подстроку. Во второе предложение должны войти слова, длина которых нечетная, и которые имеют в своем составе первую подстроку.

Вариант 4.

Дан массив слов. Преобразовать массив слов, заменив во всех словах все группы символов “ab” на символы “ccc”. Получить подстроку, взяв от каждого слова массива длиной более 1 символа по две конечных буквы. Сформировать предложение из тех слов массива, которые не содержат в своем составе символов из полученной подстроки.

Вариант 5.

Дан массив слов и подстрока. Преобразовать массив слов, удалив во всех словах первое вхождение заданной подстроки. Если в массиве остались слова, которые содержат заданную подстроку, опять выполнить аналогичные преобразования. И т.д., пока в массиве не останется слов, содержащих заданную подстроку (например, для подстроки “abc” и слова “saabcbcdabc” надо выполнить последовательно преобразования: “saabcbcdabc” → sabcd**abc**” → “sd**abc**” → “sd”). Сформировать предложение из слов, которые не являются перевертышами и не состоят из одних цифр.

Вариант 6.

Дан массив слов и подстрока. Удалить из всех слов массива, длина которых больше удвоенной длины подстроки, заданную подстроку, если она стоит с первой позиции в слове (для подстроки “abc” и слов “abcfgfabc”, “abcabc” получим слова “fgfabc”, “abcabc”). Сформировать предложение из слов, которые содержат в своем составе после 4 символа перевернутую подстроку. Слова в предложении должны быть отсортированы по алфавиту.

Вариант 7.

Дан массив слов и подстрока. Удалить во всех словах массива все цифры, добавив в конец каждого слова столь символов ‘+’, сколько цифр удалено из слова. Сформировать предложение из двух начальных символов тех слов, в которых заданная подстрока встречается более одного раза после 3-го символа.

Вариант 8.

Дан массив слов из маленьких латинских букв и слово из маленьких латинских букв. Определить процент слов массива, в составе которых есть удвоенная гласная. Сформировать предложение из слов массива, входящих в заданное слово, добавив в начало каждого такого слова столько символов ‘*’, какова позиция слова в массиве.

Вариант 9.

Дан массив слов. Удалить из всех слов массива все цифры и заменить группы символов “++” и “***” на символ ‘?’. Сформировать предложение из слов, в которых первые k букв являются «перевертышем». Слова в предложении должны быть отсортированы по убыванию длины слов.

Вариант 10.

Дан массив слов и подстрока. Преобразовать массив слов, удалив в каждом слове, начиная с конца слова, не более 3-х символов, не входящих в

заданную подстроку. Во всех словах массива поменять местами два первых и два последних символа. Сформировать предложение из таких слов массива, которые имеют в своем составе только символы из заданного слова. Слова в предложении должны быть отсортированы по алфавиту.

Вариант 11.

Дан массив слов. Преобразовать все слова массива так, чтобы каждый символ в слове повторялся один раз, сохранив общий порядок следования символов (например, “abbcacda” → “abcd”). Получить подстроку, взяв от каждого слова массива, если это возможно, по n символов, начиная с k-ой позиции. Сформировать предложение из таких слов массива, которые не имеют в своем составе полученной подстроки.

Вариант 12.

Дан массив слов. Удалить в каждом слове массиве все символы после символа ‘*’. Сформировать подстроку, взяв от каждого слова массива, если это возможно, по n символов, начиная с k-ой позиции. Если полученная подстрока не пустая, то сформировать предложение из слов массива, длина которых больше n символов, циклически сместив в словах символы на n позиций влево.

Вариант 13.

Дан массив слов и слово. Если в слове массива есть какие-либо символы из заданного слова, то после каждого из таких символов добавить столько символов ‘*’, какова длина заданного слова. Сформировать предложение из слов, в составе каждого из которых не менее двух раз встречается или подстрока “**”, или подстрока “++” (причем встречающиеся подстроки не пересекаются, т.е. в слове “+++” подстрока “++” встречается 1 раз), предварительно удалив из слов последнее вхождение подстроки “++”.

Вариант 14.

Дан массив слов. Добавить в конец каждого слова по k символов, совпадающих с первым символом слова. Удалить из всех слов цифры. Сформировать предложение из слов, в которых 3 первых символа совпадают с 3-мя последними символами, взятыми в обратном порядке, упорядочив слова в предложении по убыванию количества символов в слове.

Вариант 15.

Дан массив слов. Отсортировать массив слов по возрастанию длин слов, причем слова одной длины должны быть отсортированы по алфавиту. Сформировать предложение из слов массива, преобразовав каждое слово длиной более 2 символов следующим образом: буквы слова, стоящие до

первой гласной (гласными считать латинские буквы ‘a’, ‘o’, ‘e’), перенести в конец слова.

Вариант 16.

Дан массив слов. Сформировать новое слово из символов слов массива, стоящих после последнего символа ‘*’ в слове. Сформировать предложение из слов, длина которых более 7 символов, оставив в предложении от каждого такого слова 6 начальных и 2 конечных символа, поставив между ними символ ‘‘. Добавить в начало и конец предложения полученное слово, если оно не пустое.

Вариант 17.

Дан массив слов. Преобразовать исходный массив, вставив в каждое слово длиной более 2 символов после второй буквы подстроку из двух начальных букв этого же слова. Удалить из всех слов массива все цифры. Сформировать предложение из тех слов полученного массива, которые являются «перевертышами». Слова в предложении должны быть упорядочены по алфавиту.

Вариант 18.

Дан массив слов. Для всех слов массива, если в слове после последней подстроки “**” не стоит символ ‘?’, то заменить эту подстроку на подстроку “+++”. Удалить из слов массива все символы, кроме цифр. Сформировать предложение из трехсимвольных слов, имеющих в своем составе повторяющиеся символы, вставив в предложении между всеми словами слова из одного символа ‘*’.

Вариант 19.

Дано предложение и слово. Удалить из предложения слова длиной 3 символа, а слова длиной 1 и 2 символа заменить на заданное слово. Сформировать массив слов из слов полученного предложения. Подсчитать количество таких слов массива, которые имеют в своем составе подстроку из латинской буквы ‘a’ длиной один или два символа, обрамленную цифрами (например, подходят слова: “**5aa7**р7”, “ва**7a9**ла”).

Вариант 20.

Дан массив слов, слово и подстрока. Сформировать новую подстроку, взяв в каждом слове те символы, которые не входят в заданную подстроку. В полученной подстроке все цифры заменить на подстроку “??”. Сформировать предложение из слов массива, удалив в каждом слове столько последних символов, сколько было выполнено замен в полученной подстроке. Добавить в конец предложения перевернутое заданной слово.

Вариант 21.

Дан массив слов и подстрока. Преобразовать исходный массив слов, укоротив слова с начала слова на количество символов в предыдущем слове, если длина предыдущего слова меньше. Сформировать подстроку из двух первых символов слов с четной длиной, в составе которых нет цифр. Сформировать предложение из слов, которые в своем составе содержат заданную подстроку, заменив первое вхождение заданной подстроки в таких словах на сформированную подстроку

Вариант 22.

Дан массив слов и подстрока. Сформировать предложение из слов, содержащих заданную подстроку менее трех раз (каждая буква может входить только в одну подстроку), поменяв предварительно во всех словах с нечетной длиной символы в обратном порядке. Слова в предложении должны быть отсортированы по возрастанию длины слов.

Вариант 23.

Дан массив слов и подстрока. Преобразовать массив слов, удалив в каждом слове, начиная с конца слова, не более 2-х символов, не входящих в заданную подстроку. Во всех словах массива поменять местами первый и последний символы. Сформировать предложение из таких слов массива, которые имеют в своем составе хотя бы один символ из заданного слова. Слова в предложении должны быть отсортированы по возрастанию длин строк.

Вариант 24.

Дан массив слов. Для всех слов массива, если в слове после последнего символа “?” не стоит подстрока ‘##’, то заменить эту подстроку на подстроку “$$$”. Удалить из слов массива все цифры. Сформировать предложение из четырехсимвольных слов, имеющих в своем составе повторяющиеся символы, вставив в предложении между всеми словами из двух символов символ ‘&’.

Вариант 25.

Дан массив слов, слово и подстрока. Сформировать новую подстроку, взяв в каждом слове те символы, которые входят в заданную подстроку. В полученной подстроке все цифры заменить на подстроку “DD”. Сформировать предложение из слов массива, удалив в каждом слове столько первых символов, сколько было выполнено замен в полученной подстроке. Добавить в конец предложения перевернутое заданное слово.

Вариант 26.

Дан массив слов и слово. Если в слове массива есть какие-либо символы из заданного слова, то после каждого из таких символов добавить столько символов ‘*’, какова длина этого слова. Сформировать предложение из слов, в составе каждого из которых не менее трех раз встречается или подстрока “**”, или символ “+” (причем встречающиеся подстроки не пересекаются, т.е. в слове “+++” подстрока “++” встречается 1 раз), предварительно удалив из слов первое вхождение подстроки “**”.

Вариант 27.

Дан массив слов. Преобразовать исходный массив, вставив в каждое слово длиной более двух и менее 6 символов после второй буквы подстроку из двух конечных букв этого же слова. Удалить из всех слов массива все латинские буквы. Сформировать предложение из тех слов полученного массива, которые являются «перевертышами». Слова в предложении должны быть упорядочены по длине слов.

Вариант 28.

Дан массив слов. Преобразовать все слова массива так, чтобы каждый символ в слове повторялся один раз, сохранив общий порядок следования символов (например, “abbcacda” → “abcd”). Получить подстроку, взяв от каждого слова массива, если это возможно, по 2 символа, начиная с k-ой позиции. Сформировать предложение из таких слов массива, которые не имеют в своем составе полученной подстроки.

Вариант 29.

Дан массив слов. Сформировать новое слово из символов слов массива, стоящих перед последним символом ‘*’ в слове. Сформировать предложение из слов, длина которых более 6 символов, оставив в предложении от каждого такого слова 4 начальных и 4 конечных символа, поставив между ними символ «#”. Добавить в начало и конец предложения полученное слово, если оно не пустое.

Вариант 30.

Дан массив слов и две подстроки. Преобразовать массив слов, заменив во всех словах последнее вхождение первой подстроки на вторую подстроку. Сформировать предложение из слов, в составе которых есть буква «a» или «b», предварительно добавив к слову это же перевернутое слово (например, слово “ab3cd” должно войти в предложение в виде “ab3cddc3ba”).