# Лабораторная работа №7. Разработка приложений с использованием типа string

**1 Цель работы**

Изучить теоретические принципы и получить практические навыки разработки программ в среде Visual Studio 2005 с использованием типа **string** из стандартной библиотеки шаблонов STL*.*

**2 Порядок выполнения работы**

Получить задание на выполнение лабораторной работы (раздел 6) согласно своему варианту. Разработать и отладить программу. Составить и защитить отчет о лабораторной работе у преподавателя.

**3 Содержание отчета**

* наименование и цель работы;
* задание на лабораторную работу согласно варианту;
* схема алгоритма, текст программы на алгоритмическом языке;
* результаты работы программы.

**4 Краткая теория**

STL (Standard Templates Library) - это библиотека стандартных шаблонов, которая является стандартом C++. В С++ поддерживаются два типа строк:

– встроенный тип,

– класс **string** из библиотеки STL.

Встроенный строковый тип перешел к С++ по наследству от С. Строка символов хранится в памяти как массив, и доступ к ней осуществляется при помощи указателя типа **char\***. Стандартная библиотека С предоставляет набор функций для манипулирования строками. Применение встроенного строкового типа не очень удобно из-за того, что он реализован на слишком низком уровне. Класс **string** удобней в применении, кроме того, предоставляет гораздо больше возможностей по сравнению со встроенным строковым типом.

Класс **string** определен в заголовочном файле **string**. (Нужно отметить, что заголовочные файлы **string.h** и **cstring** поддерживают библиотеку функций языка С для работы со стандартными строками, а не класс **string**.) Класс **string** содержит обширный набор методов, включая ненесколько конструкторов, перегруженные операторы, вывыполняющие присваивание, конкатенацию, сравнение строк и осуществляющие доступ к отдельным элементам, а также функции для поиска символов и подстрок в строке и многое другое.

Для того чтобы использовать объекты класса **string**, необходимо включить соответствующий заголовочный файл:

**#include <string>**

**4.1 Создание строки – конструкторы класса string**

Для создания объектов класса **string** имеется выбор из шести конструкторов. В таблице 7.1 кратко описаны все конструкторы и приведены примеры их использования.

*Таблица 7.1 – Конструкторы класса* ***string***

|  |  |  |
| --- | --- | --- |
| 1 | **Конструктор** | **string(const char \* s)** |
| Описание | Инициализирует объект класса string строкой с завершающим нулем (стандартная строка С) с указателем s. |
| **Пример** | **string one("Здравствуйте!");** |
| Результат примера | Здравствуйте! |
| 2 | **Конструктор** | **string(size\_type n, char c)** |
| Описание | Создает объект класса string размером n элементов, каждый из которых инициализируется символом с. (size\_type – это зависящий от реализации интегральный тип, определенный в строковом заголовочном файле.) |
| **Пример** | **string two(5, '$');** |
| Результат примера | $$$$$ |
| 3 | **Конструктор** | **string(const string \* str,**  **size\_type pos = 0,**  **size\_type n = npos)** |
| Описание | Инициализирует объект класса string объектом str, начиная с позиции pos в объекте str и завершая str, или копирует n символов, в зависимости от того, какое событие произойдет раньше. (npos – максимально возможная длина строки.) |
| **Пример** | **string three(one); string three1(one,10);** |
| Результат примера | Здравствуйте! те! |
| 4 | **Конструктор** | **string()** |
| Описание | Создает объект класса string нулевого размера. |
| **Пример** | **string four;** |
| Результат примера | – |
| 5 | **Конструктор** | **string(const string \* str, size\_type n)** |
| Описание | Инициализирует объект класса string строкой с завершающим нулем (стандартная строка С) с указателем s и длиной n символов, даже если это превышает размер строки с завершающим нулем. |
| **Пример** | **char alls[] = "Ура!!! Каникулы!";**  **string five(alls,4);** |
| Результат примера | Ура! |
| 6 | **Конструктор** | **template <class Iter>**  **string (Iter begin, Iter end)** |
| Описание | Инициализирует объект класса string значениями в диапазоне [begin, end), где begin и end действуют как указатели; диапазон включает в себя begin и простирается до end, но не включает его. |
| **Пример** | **string six(&five[7], &five[15]);** |
| Результат примера | Каникулы |

**4.2 Реализация ввода и вывода в классе string**

Существует два способа ввода строк:

1. через перегруженный оператор >>:

**string name;**

**cin >> name;**

1. с помощью эквивалента функции **getline()**:

**string fullName;**

**getline(cin, fullName);**

Если осуществляется ввод строки, содержащей пробелы, то необходимо использовать второй вариант ввода, так как первый может давать некорректный результат.

Вывод строк осуществляется через перегруженный оператор <<:

**cout << name << ‘\n’;**

**4.3 Общие принципы работы со строками**

***Длину строки*** возвращает функция **size()** или **length()** (длина не включает завершающий нулевой символ):

**string st(“Hello, World!”);**

**cout << "Длина "**

**<< st**

**<< " равна "**

**<< st.size()**

**<< "символов, включая символ новой строки\n";**

Функция **max\_size()**возвращает ***максимально допустимую длину строки***.

***Доступ к отдельному элементу строки*** осуществляется с использованием оператора [ ] и указанием индекса (номера) символа в строке:

**st2[1] = ‘E’;**

Метод **at()** обеспечивает такие же возможности доступа к отдельному символу строки, но номер используется как обозначение аргумента функции:

**string word("задача");**

**cout << word.at(4); // отображает ч**

Cпециальный метод **empty()** возвращает **true** для ***пустой строки*** и **false** для непустой:

**if (st.empty())**

**// правильно: пустая**

***Сравнение строк*** осуществляется с использованием операторов ==, >, < (сравнение идет по длине строк и по коду символов):

**if ( st == st1 ) cout << st;**

**4.4 Копирование и присваивание**

***Копирование*** одной строки в другую происходит с помощью обычной операции присваивания:

**string st1 = st;**

Или с помощью третьей формы конструктора (инициализация объекта типа string другим объектом того же типа):

**string st2 (st);**

Также класс строковых величин поддерживает методы **assign()**, позволяющие присвоить строковому объекту всю строку, ее часть или последовательность одинаковых символов.

Некоторые прототипы метода **assign():**

**basic\_string& assign(const basic\_string&);**

**basic\_string& assign(size\_type n, charT c);**

**// присваивает n копий символа с**

Примеры:

**string test;**

**string s\_matr("матрица – двумерный массив");**

**test.assign(s\_matr, 1, 5); //test = "атриц"**

**test.assign(6, '#"); // test равна "######"**

Функция **substr** возвращает ***новую строку, которая является подчиненной строкой для исходного объекта***. Синтаксис объявления функции:

**basic\_string substr(size\_type pos = О,**

**size\_type n = npos) const;**

Функция возвращает строку, являющуюся копией исходной строки, начиная с позиции pos и продолжая ее на n символов или до конца строки, в зависимости от того, как далеко находится конец строки. Например:

**string message("Оплатите счет.");**

**string s(message.substr(9, 4)); //s = «счет»**

**4.5 Конкатенация и дописывание строк**

***Конкатенация*** (объединение) строк:

**st2 += st;**

**string s3 = s1 + s2;**

Операция сложения может конкатенировать объекты класса string не только между собой, но и со строками встроенного типа. Например:

**string s4( "Ура" );**

**const char \*pc = "! ";**

**string s5( "Каникулы" );**

**string s6 = s4 + pc + s5 + pc + "\n";**

Подобные выражения работают потому, что компилятор знает, как автоматически преобразовывать объекты встроенного типа в объекты класса string. Возможно и простое присваивание встроенной строки объекту string:

**const char \*pc = "простая строка";**

**string s1 = pc;**

Обратное преобразование, однако, не работает. Попытка выполнить следующую инициализацию строки встроенного типа вызовет ошибку компиляции:

**char \*str = s1; // ошибка компиляции**

Чтобы осуществить такое преобразование, необходимо явно вызвать функцию-член **c\_str().** Правильный вариант инициализации выглядит так:

**const char \*str = s1.c\_str(); // правильно**

Одну строку также можно прибавить к другой с помощью метода **append().** Методы **append()** позволяют добавлять не только строковый объект, но и массив или отдельный символ к другой стростроке. Дополнительно они дают возможность присоединить часть строкового объекта, заданную с помощью начальной позиции и количества символов или посредством указания диапазона. Можно также добавить часть строки, указав количество требуемых символов строки. В варианте, позволяющем дописывать один символ, можно уточнить, сколько раз необходимо скопировать указанный символ.

Некоторые прототипы метода **append:**

**basic\_string& append(const basic\_string& str);**

**basic\_string& append(size\_type n, charT c) ;**

**// дописывает n копий символа с**

Примеры:

**string test("Тео");**

**test.append("рия"); // test = "Теория"**

**test.append(3, '!'); // test = "Теория!!!"**

**4.6 Методы вставки, удаления и замены**

Метод **insert()** позволяет ***вставлять*** строковый объект, массив, символ или несколько символов в строковый объект. Эти методы подобны методам append(), но они содержат дополнительный аргумент, показывающий, куда необходимо вставлять новые данные.

Новые данные вставляются в позицию, находящуюся перед точкой вставки. Некоторые из методов возвращают ссылку на результирующую строку.

Например, с помощью следующего фрагмента программы строка "форма" вставляется перед символом ‘.’ в строку str:

**string str("школьная .");**

**str.insert( 9, "форма");**

Следующий фрагмент программы вставляет строку "для детей" в конец строки "школьная форма." перед символом ‘.’:

**str.insert(str.size()-1, "для детей", 9);**

Метод **erase()** ***удаляет*** символы из строки.

Метод удаляет с позиций **pos** **n** следующих символов или все оставшиеся до конца строки символы, в зависимости от того, какой из вариантов осуществится первым. Пример:

**string st(“авария”)   
st.erase(0, 2); //st = “ария”**

***Удалить*** все символы из строки позволяет функция **clear()**:

**st.clear();**

Многочисленные методы **replace()** определяют часть строки, которую необходимо ***заменить***, и саму ***замену***. В качестве замены может выступать строковый объект, строковый массив или какой-нибудь символ, продублированный несколько раз.

Пример:

**string test("Раз два три");**

**test.replace(4,3,"2"); //заменить два на 2**

**4.7 Поиск строковых величин**

Класс **string** содержит целое семейство функций поиска. В таблице 7.2 приведено краткое описание.

*Таблица 7.2 – Семейство методов* ***find()***

|  |  |
| --- | --- |
| **Прототипы методов find()** | **Описание** |
| **size\_type find**  **(const basic\_string & str, size\_type pos = 0) const;** | Находит первое вхождение подстроки **str**, начиная с позиции **pos** в исходной строке. Возвращает индекс первого символа, если подстрока найдена, или **string::npos** |
| **size\_type find**  **(const charT\* s, size\_type**  **pos = 0) const;** | Находит первое вхождение подстроки **s**, начиная с позиции **pos** в исходной строке. Возвращает индекс первого символа, если подстрока найдена, или **string::npos** – в ином случае. |
| **size\_type find**  **(const charT\* s, size\_type**  **pos, size\_type n) const;** | Находит первое вхождение подстроки, состоящей из первых **n** символов строки **s**, начиная с позиции **pos** в исходной строке. Возвращает индекс первого символа, если подстрока найдена, или **string::npos** – в ином случае. |
| **size\_type find (charT c, size\_type pos = 0) const;** | Находит первое вхождение символа **c**, начиная с позиции **pos** в исходной строке. Возвращает индекс символа, если он найден, или **string::npos** – в ином случае. |

Например:

**string str("Это строка для поиска подстроки.");**

**string fstr("строк");**

**// присваивает ind1 значение 4**

**size\_type ind1 = str.find(fstr);**

**//присваивает ind2 значение 25**

**size\_type ind2 = str.find(fstr, ind1+1);**

При проверке значения на корректность используется величина **string::npos**:

**if (ind1 == string::npos) cout << "Подстрока не найдена\n";**

Второй метод выполняет то же самое, но использует в качестве подчиненной строки массив символов вместо строкового объекта:

**size\_type indЗ = str.find("для");**

**//присваивает indЗ значение 11**

В библиотеке также содержатся родственные методы

**rfind(),**

**find\_first\_of(),**

**find\_last\_of(),**

**find\_first\_not\_of(),**

**find\_last\_not\_of(),**

каждый из которых перегружен с таким же набором сигнатур функций, как и метод **find()**. Метод **rfind()** находит последнее вхождение подстроки или символа. Метод **find\_first\_of()** находит первое вхождение в исходной строке любого из символов, содержащихся в аргументе.

**4.8 Полезные советы**

1) Все прототипы и примеры применения рассмотренных в лабораторной работе функций и методов можно найти во встроенной справке среды программирования Visual Studio 2005.

2) Описание исключительных ситуаций, которые могут возникнуть при применении описанных методов и функций также можно найти во встроенной справке среды программирования Visual Studio 2005.

**5 Пример программы**

Дана строка, состоящая не менее чем из 5 слов. Найти количество слов, у которых первый и последний символы являются цифрами. Проверить, есть ли одинаковые слова среди найденных. Найти максимальную сумму из первой и последней цифры.

**#include <iostream>**

**#include <string>**

**using namespace std;**

***//функция для удаления лишних пробелов в строке***

**string trim(string str)**

**{**

***//удаление пробелов из начала***

**int i = 0;**

**while (str[i] == ' ')**

**i++;**

**str = str.erase(0,i);**

***//удаление концевых пробелов***

**i = str.length()-1;**

**while (str[i] == ' ')**

**i--;**

**str = str.erase(i+1, str.length()-i+1);**

***//удаление лишних пробелов в середине строки***

**i = 0;**

**while (i < str.length())**

**if (str[i] == ' ' && str[i+1] == ' ')**

**str = str.erase(i,1);**

**else**

**i++;**

**return str;**

**}**

**void main()**

**{**

**string str;**

**cout << "Input string - ";**

***//ввод исходной строки с клавиатуры***

**getline(cin, str);**

***//удаление лишних пробелов***

**str = trim(str);**

***//добавление одного пробела в начале и в конце***

***//исходной строки для того, чтобы каждое слово***

***//было ограничено пробелами***

**str = " " + str + " ";**

**int i = 0;**

***//счетчик слов, начинающихся и заканчивающихся //цифрой***

**int words\_count = 0;**

***//максимальная сумма первой и последней цифры***

***//сумма не может быть отрицательной, поэтому***

***//первое же сравнение даст результат***

**int max\_sum = -1;**

***//цикл поиска и подсчета слов***

**while (i < str.length()-1)**

**{**

***//получаем левую и правую границы слова***

***//т.е. индексы ограничивающих пробелов***

**int ind1 = str.find(" ") + i;**

**int ind2 = str.find(" ", ind1+1);**

***//получаем очередное слово***

**string word = str.substr(ind1+1,ind2-ind1-1);**

***//получаем первый и последний символ в слове***

**char first = word[0];**

**char last = word[word.length()-1];**

***//если первый и последний символы слова - //цифры,то...***

**if (isdigit(first) && isdigit(last))**

**{**

***//увеличиваем счетчик***

**words\_count++;**

***//переводим первый и последний символы в числовую***

***//форму***

**int f = atoi(word.substr(0, 1).c\_str());**

**int l = atoi(word.substr(word.length()-1, 1).c\_str());**

***//считаем сумму и проверяем, является ли***

***//она максимальной***

**if (l+f > max\_sum)**

**max\_sum = l+f;**

**}**

***//проверяем, является ли найденное слово***

***//единственным в строке***

**if (str.find(word,ind2) != string::npos)**

**cout << "The word \" " << word << "\" is not one is source string.\n";**

**i = ind2;**

**}**

***//вывод результатов***

**cout << "The number of words starting and ending from digits = "<<words\_count<<endl;**

**cout << "max summa of digits = "<<max\_sum<<endl;**

**}**

**6 Контрольные вопросы**

1. Какой класс используется для работы со строками в библиотеке STL? В каком заголовочном файле он определен?

2. Перечислите основные виды констркуторов класса строк. Чем они отличаются?

3. Чем отличается ввод/вывод стандартных строк С++ и строк библиотеки STL?

4. Назовите основные знаки операций работы со строками библиотеки STL. Какие действия они выполняют?

5. Перечислите основные функции работы со строками библиотеки STL. Для чего они предназначены?

**7 Варианты заданий для самостоятельного решения**

1. Дана последовательность слов. Проверить правильность написания сочетаний "жи", "ши", "ча", "ща", "чу", "щу". Если надо, то исправить ошибки их написания.

2. Дан список фамилий. Составить новый список, который будет содержать только женские фамилии. (Примечание: те фамилии, по которым пол трудно определить, не считать).

3. Найти произведение и минимальную четную цифру, составляющую некоторое число Х. Предусмотреть проверку на правильность ввода числа.

4. Заменить все вхождения строки Str1 на подстроку Str2, которые вводятся с клавиатуры. Подсчитать количество произведенных замен.

5. Перевести число из 10-ой системы в 16-тиричную и наоборот.

6. Дан текст, состоящий не менее чем из пяти слов. Сформировать строку, в которую попадут только те слова, где одинаковые буквы встречаются более двух раз. Например, молоко.

7. Из имеющегося набора слов выбрать наиболее длинное, в котором все буквы разные. Например, *лейкопластырь, неряшливость.*

8. Дан текст. Группы символов, разделенные пробелами и не содержащие пробелы внутри себя, будем считать словами. Напечатать эти слова в алфавитном порядке.

9. Дана последовательность, состоящая из цифр, букв и символов в произвольном порядке. Подсчитать чего больше и составить строки только из цифр, букв и символов. Вывести строки в порядке возрастания их длины.

10. Дана некоторая последовательность букв русского алфавита. Написать программу, которая запрашивает Ваше имя и определяет, можно ли из букв исходной строки составить его. Если нет, то выводит буквы, которых не хватает.

11. Дан текст, записанный в виде криптограммы (шифрограммы), в которой буквы истинного текста размещаются в позициях, кратных 3. Прочитать исходный текст.

12. Дана строка цифр. Составить из них 5-значные числа. Если на последнее число не хватит цифр, дополнить его первыми цифрами исходной строки.

13. Перевести число из 2-ой системы в 10-тичную и наоборот.

14. Дан текст. Найти самое короткое слово, длиной не более 3 букв.

15. Написать программу, которая переставит слова в строке по мере увеличения их длины.

16. Дан текст, состоящий не менее чем из пяти слов. Вывести на экран слова, в которых отсутствует буква "Е".

17. Дана последовательность чисел. Написать программу, которая запрашивает дату рождения человека (можно только число, месяц или год) и определяет, сколько раз эта дата встречается в строке.

18. Найти количество нечетных цифр и максимальную нечетную цифру, составляющую некоторое число Х.

19. Дан текст, состоящий не менее чем из пяти слов. Определить, есть ли в нем слова, начинающиеся и заканчивающиеся с буквы "А", а также количество таких слов.

20. Дан текст, состоящий не менее чем из пяти слов. Написать программу, которая выполняет следующие преобразования: в первом слове делает заглавной первую букву, во втором слове делает заглавной вторую букву и т.д. Примечание: слова вводятся строчными буквами!

21. Написать программу, которая проверяет, является ли введенное слово палиндромом. Палиндромом называется слово, которое читается одинаково слева направо и справа налево, например, "КАЗАК".

22. Дана строка слов. Подсчитать, какая из букв встречается наибольшее (наименьшее) количество раз, а также, общее количество разных букв в строке.

23. Дан текст, состоящий не менее чем из пяти слов. Вывести на экран слова, которые имеют одинаковые окончания "ИЯ","ИСТ","ИКА".

24. Найти минимальную и максимальную цифры среди четных и нечетных цифр, составляющих некоторое число Х.

25. Дан текст. Преобразовать его по следующему правилу: если нет символа ‘\*’, то оставить его без изменения, иначе заменить каждый символ, встречающийся после первого вхождения символа‘\*’, на символ ‘-’ .

26. Подсчитать сумму и количество всех цифр, входящих в некоторое предложение, вводимое с клавиатуры.

27. Дан текст, состоящий не менее чем из семи слов. Все слова из четырех букв записать наоборот.

28. Разбить предложение на отдельные слова и посчитать количество слов. Предложение вводится с клавиатуры. Слова сохранять в массиве.

29. Дан текст, оканчивающийся точкой. Найти количество слов, у которых первый и последний символы совпадают.

30. Найти среднее арифметическое значение между минимальной и максимальной цифрами некоторого числа Х.

31. Дан текст, состоящий не менее чем из семи слов. Определить, есть ли в нем слова, начинающиеся с буквы "Ф", а также количество таких слов.

32. Дан текст. Исключить из него символы, расположенные между скобками ‘(‘, ‘)’. Сами скобки тоже должны быть исключены. Предполагается, что внутри каждой пары скобок нет других скобок.