Федеральное государственное бюджетное образовательное учреждение высшего образования
Казанский государственный энергетический университет
Кафедра информационных технологий и интеллектуальных систем

Строки

Методические указания для выполнения лабораторных работ

Казань, 2024

Лабораторная работа. Строки

Цель работы: изучить принципы работы со строками и указателями на строки, алгоритмы решения задач на обработку строковых данных, научиться приемам решения задач на обработку строк, используя, функции для работы со строками и указателями на строки в языке С++.

План лабораторной работы

1. Функции обработки строк
2. Применение строк

1. Необходимые теоретические сведения к выполнению лабораторной работы

1.1. Текстовые данные

В языке С++ текстовая информация представляется двумя типами данных: с помощью символов и строк-массивов символов.

Символьный тип данных. Значением данных символьного типа является любой символ из набора всех символов компьютера или его код. Каждому символу соответствует порядковый номер (код) в диапазоне 0…255. Для кодировки символов первой половины диапазона используется код ASCII или более современные стандарты в последних версиях языка Си. При написании программ символьные данные могут быть представлены либо константами, либо переменными.

Символьная константа представляет собой одиночный символ, заключенный в апострофы, например: `Y` `!` `_' `Д`

Символьная переменная объявляется с помощью ключевого слова `char`, например: `char cr`;

Во внутренней памяти компьютера каждый символ занимаем 1 байт.

1.2 Ввод-вывод символьных данных Для ввода символьных данных используются функции: `scanf_s()` –форматированный ввод, `getchar()` или

getch() –специальные функции для ввода символа. Для форматного ввода и вывода символьных констант используется спецификатор (формат) %c.

Необходимо помнить, что нажатие любой небуквенной клавиши при вводе ([пробел], [Enter] и др.) будет значимым и восприниматься как символ.

Пример 1. Организовать ввод символьных переменных: a= 'i' b = 'j' c= 'k'.

```
main()
{  char a,b,c;
printf("Введите исходные данные");
scanf_s("%c%c%c",&a,&b,&sc);. . .}
```

При вводе символы набираются без апострофов и пробелов: **ijk** [Enter]
Символ клавиши [Enter] выходит за пределы списка ввода, поэтому он игнорируется.

При вводе символьной информации с помощью функции getchar() надо помнить, что функция переводит программу в состояние ожидания, но при нажатии клавиши символ выводится на экран. А, например, при выполнении следующего фрагмента программы

```
printf("Введите исходные данные");
a=getch();b=getch();c=getch();
```

переменные будут введены, но на экране их значения не отразятся.

Для вывода символьных данных используются операторы cout, функции printf() и putchar().

Пример 2. Организовать вывод указанных выше переменных на экран в одну строку. Запись оператора вывода будет следующей:

```
printf("%c%c%c\n",a,b,c);
```

На экране будет отображено: ijk

Если использовать для вывода функцию

```
putchar(); putchar(a); putchar(b); putchar(c);
```

на экране будет отображен тот же результат.

Правила ввода/вывода с помощью операторов cout/cin реализуется так же, как и с числовыми данными.

1.3 Обработка символьных данных. Поскольку символы в языке C++ упорядочены, к ним можно применять операции отношения (>, >=, <, <=, ==, !=). Это дает возможность записывать логические выражения с символьными данными в условных операторах.

Например, оператор if(ch=='!') ch='.'; сравнивает значение переменной ch с символом ‘!’ и в случае их равенства следующая команда заменяет в символьной переменной ch восклицательный знак точкой. Символьные

данные могут использоваться и в операторах цикла `for`. Так, при выполнении операторов:

```
for( ch='a'; ch>='d'; ch++)   printf("%c",ch);
```

в строку экрана выводится последовательность: **abcd**

Если значение символьной переменной вывести с помощью спецификатора для целых чисел `%d`, то на экране *отобразится код символа*. Например:

```
for(ch='a'; ch>='d'; ch++)   printf("%d ",ch);
```

на экран будет выведено: **97 98 99 100**

Над *символьными данными можно выполнять арифметические операции сложения и вычитания*. Так, например, операция `ch++;` из предыдущего примера увеличивает код символа, хранящегося в переменной `ch` на 1. Или, выполняя операцию `ch='a'-'A';` будет получена разница кода большой (A) и маленькой буквы (a) латинского алфавита.

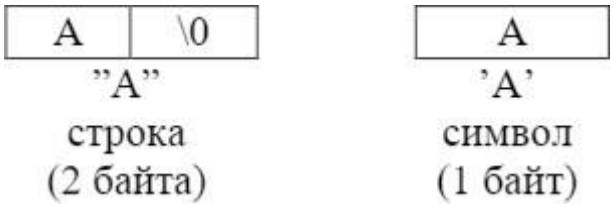
Так, например, если в символьной переменной `ch1` хранится маленькая буква алфавита, то, выполнив действия:

```
char ch,ch1,ch2;
ch='a'-'A';ch1='k';
ch2=ch1-ch;
printf("%c-%d %c-%d\n",ch1,ch1,ch2,ch2);
```

в переменную `ch2` запишется та же буква, только большая, а на экран будет выведено

```
k-107  K-75
```

1.4 Строки Строка в C++ –это массив символов, заканчивающийся нуль-символом –`'\0'` (нуль-терминатором). По положению нуль-терминатора определяется фактическая длина строки. Количество элементов в таком массиве на 1 больше, чем изображение строки:



Символы в кавычках будут записаны в начало массива s, а затем-признак окончания строки `'\0'`. При описании строки можно также написать так:

```
char s[80] = "Язык программирования Си++";
```

или `char s[] = "Язык программирования Си++";`

В этом случае компилятор подсчитает символы в кавычках, выделит памяти на 1 байт больше и занесет в эту область саму строку и завершающий ноль.

Присвоить значение строке с помощью оператора присваивания нельзя.

Поместить строку в массив можно либо при вводе, либо с помощью инициализации.

```
char s1[10]="string1";//инициализация
char s2[]="string2";//инициализация
char s3[10];
cin>>s3;//ввод
//выделение памяти под динамическую строку
char *s4=new char[strlen(s3)+1];
strcpy(s4,s3);//копирование строки s3 в строку s4
```

Ещё больше примеров по способам инициализации строк и способам работы со строками и символами можно найти в соответствующей лекции (слайд 5-16).

1.5 Стандартные функции обработки строк. Большинство действий над строками реализуется с помощью стандартных функций. Библиотека языка Си содержит большое количество таких функций, прототипы которых определяются в заголовочном файле `<cstring>`.

Рассмотрим некоторые из них.

Сравнение строк:

`strcmp(str1,str2)` – сравнивает две строки `str1` и `str2` и возвращает 0, если они одинаковы; результат отрицателен, если `str1<str2` и положителен, если `str1>str2`.

`strncmp(str1, str2, kol)` – сравниваются части строк `str1` и `str2` из `kol` символов. Результат равен 0, если они одинаковы. Сравнение двух строк выполняется последовательно слева неправо с учетом кодировки символов. Например, сравнивая стоки `st1` и `st2`

```
char st1[10]="Пример";
charst2[10]="ПРимер";
int a;
  if (strcmp(st1,st2)>0)
    a=1; else a=2;
```

переменной a будет присвоено значение 1, так как код символа 'р' больше кода символа 'Р'

Сцепление строк:

`strcat(str1,str2)` - сцепление строк в порядке их перечисления.

`strncat(str1,str2,kol)` – приписывает `kol` символов строки `str2` к строке `str1`.

Функция служит для объединения двух строк в одну. Например, в результате выполнения операторов:

```
charfam[] = "Андреева С.В.";
char pr[7]= ""; //7 пробелов
strcat(fam ,pr);
printf("|%20s|", fam);
```

на экран выведется строка:

|Андреева С.В. |

Заметим, что строка вывода занимает поле в 20 позиций, а переменная `fam` располагается в левой части поля.

Определение длины строки

`strlen(str)` – определяет длину строки `str`.

Пример. Определить длину строки

```
charfam[] = "Андреева С.В.";
printf("%d",strlen(fam));
```

функция `strlen()` вернёт значение равное 13 (символов).

Копирование строк

`strcpy(str1,str2)` –копирует строку `str2` в строку `str1`.

`strncpy(str1, str2, kol)` –копирует `kol` символов строки `str2` в строку `str1`.

Пример. Скопировать фамилию сотрудника в переменную `fam` и вывести на экран.

```
#include <iostream>
#include<cstring>
int main()
{ char fam[15];
char *str = "АндрееваС.В.";
strcpy(fam, str);
printf("|%s|\n", fam);return0; }
```

В результате выполнения данных операторов на экран будет выведена строка:

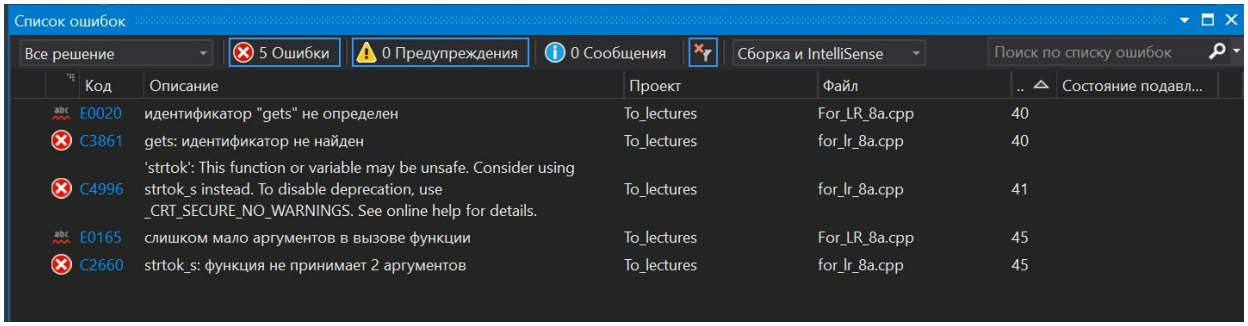
|Андреева С.В.|

1.6 Анализ кода программы со строковыми функциям. Анализ кода программы, как это уже было и ранее показано, является важнейшим умением программиста. Этот навык необходим и при проверке корректности работы своих программ, при изучении методов и алгоритмов, записанных в виде готовых кодов. При работе с программами на С++ могут возникать проблемы не только при записи, но и при анализе готовой программы. Проблема здесь в том, что , начиная с версии VisualStudio 2015, многие стандартные функции устарели и на замену им транслятором часто, но не всегда, рекомендуется использовать альтернативные функции. Однако, даже при наличии «подсказок» транслятора проблема не решается: альтернативные функции, имена которых может подсказать транслятор, меняют не только имена, но и список формальных параметров. Рассмотрим код программы на С++ и попытаемся определить, какую задачу она решает (все комментарии удалены в целях обучения)

```
int main() {
	setlocale(LC_ALL, "Russian");
```

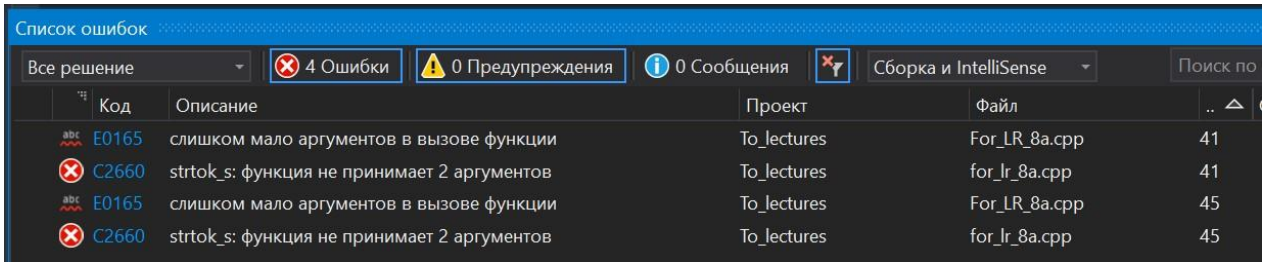
```
      char text[100], *p;
      const char *razd = " .,";
      int dlina;
puts("Введите текст ");
gets(text);
p = strtok(text, razd);
while (p) {  //
      dlina = strlen(p);
      cout << "\n слово " << p << " длина = " << dlina << "\n";
      p = strtok(NULL, razd);

}
...
```

Многие сразу попытаются прогнать программу через компьютер, «чтобы узнать ответ», однако задача анализа – это не только получить решение, но главное – понять *как* такой ответ получается. Но даже, если этот вариант запустить на выполнение, мы встретимся с ошибками, которые не позволят создать загрузочный файл. Вот какие ошибки будут найдены транслятором:



Как мы и предполагали, транслятор предлагает часть функций заменить на функции с дополнением «`_s`». Заменяем `gets` на `gets_s`, `strtok` на `strtok_s`.

Однако этого, оказалось, недостаточно. Вновь идут сообщения об ошибках.



Как видим, функция `strtok_s`, прототип которой `strtok(строка, строка_разделителей)`, оказывается, не принимает двух аргументов. Современ-

ные печатные учебники не успевают включать все те изменения, которые происходят в программном обеспечении, поэтому обращаемся за помощью на специализированные формы в Интернет.

Выясняется, что прототип функции изменился:

`strtok_s(char *строка, char* разделители, char **контекст);`

Суть изменения можно найти на http://msdn.microsoft.com/ru-ru/library/ftsafwz3.aspx

С учётом принятой информации, не вдаваясь на данном этапе в разбор сути третьего параметра, переписываем код программы, вставляя *фиктивный указатель* на `char *next_token`:

```
int main() {
setlocale(LC_ALL, "Russian");
char text[100], *p, *next_token;
const char *razd = " .,";
int dlina;
puts("Введите текст ");
gets_s(text);
p = strtok_s(text, razd, &next_token);
while (p) {
dlina = strlen(p);
cout << "\n слово " << p << " длина = " << dlina << "\n";
p = strtok_s(NULL, razd, &next_token);
}
```

…

Решение:

1. Выделяется строка `text[100]`, серия указателей на `char`, инициализируется указатель на строку разделителей `*razd = " ,"` - пробел, точка, запятая.
2. `gets_s`: Вводим текст (латинскими буквами!), например:
   `text` = «Yesterday, all my troubles seemed so far away»
3. `strtok_s(text, razd, &next_token)` находит в `text` первые из перечисленных разделителей после “Yesterday” – это запятая, а за ней - пробел. Функция возвращает указатель `p` на слово “Yesterday” , а перед следующим словом “all” ставится маркер, с которого начнётся поиск следующего разделителя.
4. Начинается цикл, который работает пока указатель `p` не встретил конец строки.
5. `dlina =` длине найденного слова, начинающегося с указателя `p` до нуль-символа, функция `strlen(p) = strlen(“Yesterday”)=8`.
6. На консоль выводится:

7. «слово Yesterday длина =8»
8. Затем снова выделяется очередное слово, указатель встаёт на начало слова **“all”.** Поскольку `p` не `NULL,` то цикл будет повторяться и, в результате, на консоли в каждой строке будут выведены слова введённой фразы и длина их соответствующая длина.
9. Цикл, прекратит работу после встречи конца строки
10. Консоль будет иметь следующий вид:

```
Введите текст
Ysterday, all my troubles seemd soo far away

 слово Ysterday длина = 8

 слово all длина = 3

 слово my длина = 2

 слово troubles длина = 8

 слово seemd длина = 5

 слово soo длина = 3

 слово far длина = 3

 слово away длина = 4
Для продолжения нажмите любую клавишу . . .
```

2. Методика выполнения самостоятельной работы

1. ***Внимательно изучить примеры***, приведенные в методической части лабораторной работы.
2. Из соответствующей лекции выписать прототипы стандартных функций по работе со строками.
3. Ознакомиться с условием задачи и примерами решений аналогичных задач из первой части лабораторной работы.
4. Составить контрольный пример.
5. Проверить полноту задачи: рассмотреть все возможные исходы решения в зависимости от исходных данных, предусмотреть случаи возможного зависания, зацикливания программы и запрограммировать корректную реакцию программы на эти ситуации.
6. Записать словесный алгоритм или составить блок-схему алгоритма.
7. Записать код программы на C^{++}.
8. Запустить программу, провести синтаксическую отладку.
9. Проверить работоспособность программы путём сравнения результатов с контрольным примером на все возможные случаи исходных данных.
10. Завершить работу составлением Отчёта, где будут описаны все этапы выполнения самостоятельного задания и приведены распечатки консольного вывода. Образец отчета приведен в Приложении.

3. Задания для самостоятельной работы

Номер варианта выбирается по формуле N%8+1, где N - порядковый номер студента в списке группы.

*Если Вам встретился вариант со *, и Вы не справляетесь с ним, можно увеличить Ваш N на 2. Возможна и замена лёгкого, на Ваш взгляд, варианта на вариант со *.*

Жирным шрифтом *в примерах записаны строки, которые вводятся с консоли при выполнении программы.*

Задание 1. Написать программу на языке C++, которая получает на входе одну или несколько строк символов (в зависимости от постановки задачи), выполняет обработку строк в соответствии с требованиями задания и выводит результат на экран. Ввод данных осуществляется с клавиатуры с учетом требований к входным данным, содержащихся в постановке задачи.

Вариант 1. Введите предложение, слова в котором разделены несколькими пробелами и в конце которого стоит точка. Удалите повторяющиеся пробелы между отдельными словами (оставляя по одному пробелу), выведите отредактированное предложение на экран.

Вариант 2. Дана строка. Выполните символьный анализ текста. Ниже представлен рекомендуемый вид диалога во время работы программы. Данные, вводимые пользователем, выделены жирным шрифтом.

```
Введите строку
Kazan was founded in 1005: I was burn in Kazan in 2003.
Во введённой строке :
   Строчных букв – 33
   Заглавных букв – 3
   цифровых символов – 8
   остальных символов, включая пробелы – 13
```

Вариант 3. Напишите программу, которая проверяет, является ли введенная с клавиатуры строка целым числом (знак числа не учитывать). Ниже представлен рекомендуемый вид диалога во время работы программы. Данные, вводимые пользователем, выделены жирным шрифтом.

```
Введите число: 24.5
Введенная строка не является целым числом.
```

Вариант 4. Введите строку и символ. Определите частоту появления данного символа в строке. Частота вычисляется как отношение количества данных символов в строке к длине всей строки (пробелы учитываются, а символ конца строки не учитывается). Ниже представлен рекомендуемый вид диалога

во время работы программы. Данные, вводимые пользователем, выделены жирным шрифтом.

```
Введите строку  >  To live a cat and dog life.
Введите символ  >  a
Частота появления символа 'a' в строке "To live a cat and dog life." равна 0.11
```

Вариант 5. Дана строка, в которой слова разделены одним пробелом. Замените первые буквы всех слов на заглавные (если слово начинается с заглавной буквы, оставьте без изменения).

Вариант 6. Дана строка, в которой слова разделены одним пробелом. Подсчитайте, сколько букв 'e' встречается в каждом слове. Например:

```
Введите строку  >  Better late than never
Символ 'e' встречается в словах:
Better - 2 раз(а)
late – 1 раз(а)
than – 0 раз(а)
never – 2 раз(а)
```

Вариант 7. Дана строка, в которой слова разделены или одним пробелом, или запятой и пробелом. Подсчитайте, сколько в каждом слове букв, совпадающих с его первой буквой.

Вариант 8. Дана строка. Преобразуйте ее так, чтобы сначала следовали цифровые символы, затем все остальные. Порядок следования символов между собой не изменять. Например:

```
Введите строку > About 2500 students study and 650 teachers work  at the university in 31 specialties
После преобразования получили > 2500 650 31 About students study and teachers work at the university in specialties
```

Задание 2. Написать программу на языке C++, разбивая алгоритм на модули: как минимум, одна функция, в дополнение к main, должна быть в Вашем коде.

Вариант 1. Дана строка, в которой слова разделены одним пробелом. Найдите и распечатайте все слова указанной длины n. Разработать функцию, которая сортирует найденные слова по алфавиту. Распечатать отсортированные слова в одну строку.

Вариант 2. Дана строка из символов латинского алфавита. Вставьте пробел перед каждой заглавной буквой, перед первой буквой пробел добавлять не надо. Распечатать преобразованную строку. Разработать функцию, которая будет находить в этой строке слова на заданную букву. Если таких слов будет найдено несколько, отсортировать найденные слова по их длине, разработав соответствующую функцию. Ниже представлен рекомендуемый вид диалога во время работы программы. Данные, вводимые пользователем, выделены жирным шрифтом.

```
Введите строку символов латинского алфавита:
AtTimesYouMayWantToReadDataFromTheKeyBoard
Полученная строка: At Times You May Want To Read Data
From The Key Board
```

Вариант 3. Написать программу, которая вычисляет значение выражения $N_0O_1N_1O_2...O_kN_k$, где N_i – целое число, O_i – один из двух знаков простейших арифметических действий: сложение (+) и вычитание (–). Отдельной функцией обеспечить контроль корректного ввода: в строке должны быть заданы только цифры и указанные знаки действий. Ниже представлен рекомендуемый вид диалога во время работы программы. Данные, вводимые пользователем, выделены жирным шрифтом.

```
Введите арифметическое выражение,
например, 45+5-3-125+2 (пробелы и другие знаки недопустимы)
354-457+74+2-37
Значение выражения 354-457+74+2-37 = -64
```

Вариант 4. Дана строка, представляющая собой арифметическое выражение. Разработать функцию, проверяющую правильность расстановки в ней круглых скобок: каждой открытой скобке должна соответствовать корректно закрытая скобка.

Вариант 5. Ввести фразу. Разработать функции, одна из которых разбивает данную строку на предложения, вторая - находит в ней слова, начинающиеся с заглавных букв. Результат желательно представить в виде:

```
Введите фразу: Our Father, which art in Heaven! Hallowed be Thy name.
Thy Kingdom come! Thy Will be done in Earth as it is in Heaven.
Получен стих:
Our Father, which art in Heaven!
Hallowed be Thy name.
Thy Kingdom come!
Thy Will be done in Earth as it is in Heaven.
```

Слова, которые необходимо писать с заглавной буквы:

Our, Father, Heaven, Hallowed, Thy, Thy, Kingdom,Thy, Will, Earth, Heaven.

Вариант 6. Ввести строку из файла. Вывести её на консоль. Составить функцию, которая вернёт наиболее часто встречающийся символ в строке и укажет число его повторений.

Вариант 7.* Ввести стих из файла. Разработать функцию, которая находит в нём слова с удвоенными буквами. Вывести на консоль исходный стих и найденные слова, показать, сколько и каких двойных букв в них встречается. Указание: обработке букв кириллицы использовать функции перекодировки – DOStoWIN, WINtoDOS – лекция 12, слайды 49-50.

Пример:

Ввести имя файла, где расположен стих: **stickh.txt**

В обработке находится стих

Аббревиатура,

Как клавиатура!

По нотке – аккорд или гамма,

По букве – сокращённое слово.

В нём слов с удвоенными буквами – 4:

Аббревиатура –‘б’, аккорд –`к’, гамма - ‘м’, сокращённое - ‘н’.

Вариант 8.* Составьте функцию шифрования текстового сообщения, считанного из файла. Предлагается такой способ шифрования: шифровальщик задает ключ шифровки – целое число, которое определяет величину смещения букв русского алфавита, например ключ = 3, тогда в тексте буква "а" заменяется на "г" и т.д. Составить контрольный пример. В обработке

Указание: обработке букв кириллицы использовать функции перекодировки – DOStoWIN, WINtoDOS – лекция 12, слайды 49-50.

* * *

4. Контрольные вопросы

1. По каким признакам в программе различаются прописные и строчные буквы, цифры?
2. Какие функции можно использовать для выделения отдельных слов в длинной фразе?
3. Как реализуется сортировка строковых данных?
4. Допустимы ли методы сортировок для символьных данных?
5. *Как задать динамический массив, состоящий из нескольких строк?

5. Домашнее задание

1. Представлен код программы. Проанализируйте его и скажите, какие результаты будут выведены на консоль, если с консоли будут введены слова: «слива», «apple», «яблоко», «груша».

```
void DosToWin(char d, char &w)
{
	if ((d >= -128) && (d <= -81))   // А...Я  а...п
		w = d + 64;
	if ((d >= -32) && (d <= -17))      // р...я
		w = d + 16;
	if (d == -16)   w = -88;    // Ё
	if (d == -15)  w = -72;     // ё
}
void convert(char * s, char *sw)
{
	for (int i = 0; i < strlen(s); i++) DosToWin(s[i], s[i]);
};int main()
{
	setlocale(LC_ALL, "Russian");
	char fruit[] = "яблоко";
	char answer[80];

	do
	{
		cout << "Угадайте мой любимый фрукт! >> ";
		cin >> answer;
		convert(answer, answer);
	} while (strcmp(fruit, answer) != 0);

	cout << "Правильный ответ!\n";
	system("pause");
	return 0;
}
```

2. Представлен код программы. Проанализируйте его, объясните, как и какие результаты Вы получаете после выполнения программы.

```
#include <string>
#include <iostream>
using namespace std;
```

```
char string1[] = "A string\tof ,,tokens\nand some more tokens";
char string2[] = "Another string\n\tparsed at the same time.";
char separators[] = " ,\t\n";
char *token1, *token2, *next_token1, *next_token2;


int main()
{
	cout << "Tokens:" << endl;
	token1 = strtok_s(string1, separators, &next_token1);
	token2 = strtok_s(string2, separators, &next_token2);


	while ((token1 != NULL) || (token2 != NULL))
	{
		if (token1 != NULL)
		{
			cout << token1 << endl;
			token1 = strtok_s(NULL, separators, &next_token1);
		}
		if (token2 != NULL)
		{
			cout << " " << token2 << endl;
			token2 = strtok_s(NULL, separators, &next_token2);
		}
	}
	system("pause");
	return 0;
	}
```