# Структура данных – строка

Тема. Стандартные типы данных языка программирования С++ для представления текстовых данных

Цель.

- Получение навыков в разработке алгоритмов обработки текста – извлечение отдельных элементов.

- Получение навыков использования средств языка Си и С++ для реализации алгоритмов обработки текстовых данных.

Оглавление

[Структура данных – строка 1](#_Toc95061610)

[1 Задания и требования к ним 1](#_Toc95061611)

[2 Варианты заданий 2](#_Toc95061612)

[3 Строки в стиле С (null терминальные строки) 7](#_Toc95061613)

[3.1 Строки, как статические массивы 7](#_Toc95061614)

[3.2 Строки как указатели 7](#_Toc95061615)

[3.3 Динамические строки 7](#_Toc95061616)

[3.4 Строковая константа 8](#_Toc95061617)

[3.5 Операции со строкой 8](#_Toc95061618)

[3.5.1 Присваивание 8](#_Toc95061619)

[3.5.2 Ввод-вывод строк в стиле С++ 9](#_Toc95061620)

[3.5.3 Ввод-вывод строк в стиле Си 10](#_Toc95061621)

[3.5.4 Примеры на выделение слов из текста 11](#_Toc95061622)

[3.5.5 Функции для строк <cstring.h> или <string.h> 15](#_Toc95061623)

[3.6 Примеры применения функции strtok для выделения слов в тексте 18](#_Toc95061624)

[4 Шаблонная строка класса string 25](#_Toc95061625)

[4.1 Класс string 25](#_Toc95061626)

[4.2 Конструкторы 25](#_Toc95061627)

[4.3 Ввод строки с клавиатуры и вывод в консоль 26](#_Toc95061628)

[4.4 Операции 26](#_Toc95061629)

[4.5 Методы 27](#_Toc95061630)

[4.5.1 Общие методы 27](#_Toc95061631)

[4.5.2 Методы модификации строки 29](#_Toc95061632)

[4.5.3 Методы поиска данных в строке 30](#_Toc95061633)

[4.6 Пример. Разбиение строки на лексемы (слова), отделяемые друг от друга символами разделителями из заданного набора. 31](#_Toc95061634)

# Задание 1. Разработать программу согласно задаче варианта, используя для представления в программе текста нуль терминальную строку и средства языка С для выполнения операций над этой строкой.

# Задание 2. Разработайте программу согласно задаче варианта, используя для представления обрабатываемого в программе текста строку string, стандартной библиотеки шаблонов. и возможности класса для выполнения действий со строкой.

# Задание 3. Оформить отчет по каждому из заданий.

Структура отчета

1. Разработка алгоритма задачи
   1. Постановка задачи
   2. Модель решения
2. Тестовый пример, приводящий к решению задачи
3. Описание подхода к решению

Описать все подзадачи, на которые будет декомпозирована основная задача. Возможно, что будет только одна задача.

Определить прототипы функций для задач декомпозиции.

1. Разработать алгоритмы задачи и подзадач, записать их на псевдокоде. Можно использовать в основном алгоритме и алгоритмах подзадач обращения к определенным в предыдущем пункте функциям.
2. Подготовить набор тестов для отладки задачи
3. Реализация задания 1
   1. Описать применяемые в реализации функции над строкой из библиотеки string.h и методы доступа к данным строки.
   2. Код реализации задания 1.
   3. Результаты тестирования
4. Код программы задания 2
   1. Описать применяемые в реализации функции над строкой из библиотеки <string> и методы доступа к данным строки.
   2. Код реализации задания 1.
   3. Результаты тестирования
5. Выводы по выполненной работе

# Варианты заданий

**Примечание.** Словом, считать последовательность символов, заключенную между символами разделителями. Разделителей может быть любое количество.

|  |  |
| --- | --- |
| 1 | Дано предложение, составленное из слов и групп цифр (считать тоже словами), определяющих целые числа. Слова разделены запятой или одним или несколькими пробелами. Удалить из него слова, которые встретились там более одного раза и сформировать массив из чисел, встретившихся в тексте. |
| 2 | Даны две строки, состоящие только из цифр. Считая, что в этих строках находятся очень длинные целые числа, сформировать третью строку - сумму этих чисел. |
| 3 | Дано предложение, в котором содержаться группы цифр (каждая не более 6 цифр) - это целое десятичное число представленное в строковом формате. Группы отделяются друг от друга пробелами. Найти эти группы, преобразовать в числовой формат своим, а не системным алгоримом. |
| 4 | Дано предложение, состоящее из слов, разделенных запятой или пробелами среди которых есть группы цифр, определяющих целые числа из диапазона 0..65535. Удалить из текста все числа, принадлежащие диапазону [-100..100] и на их место поставить символ \*. |
| 5 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Среди слов встречаются слова, представляющие код некоторого десятичного числа в 8- ой системе счисления (признака восьмеричного кода в тексте: число начинается с символа $: само число составлено из цифр ‘0’..’7’ ). Сформировать массив из чисел восьмеричного кода предварительно переведя их в десятичную систему счисления. |
| 6 | Дано предложение, состоящее из слов, разделенных запятой или пробелами, среди которых есть группы цифр, определяющих целые числа из диапазона int. Удалить из предложения те целые числа, в десятичной записи которых есть цифры 5,6,7 , а остальные числа увеличить на 2. |
| 7 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Вывести слова предложения предварительно преобразовав их следующим образом:  - перенести последнюю букву в начало слова;  - удалить из слова повторные вхождения каждой буквы. |
| 8 | Дано предложение, состоящее из слов, разделенных запятой или пробелами, среди которых есть слова, составленные только из цифр, определяющие целые числа из диапазона 0 .. 65535. Удалить из предложения все слова-числа, состоящие из одинаковых цифр. Сформировать массив из чисел предложения, преобразовав их в обратные (например: исходное число 123, а в массив записать 321). |
| 9 | Дано предложение и управляющий символ, значением которого может быть один из символов Y, N. Вывести этот текст без входящих в него цифр, если значение символа управления = Y ,а если этот символ = N, то перенести все цифры текста в конец предложения так, чтобы первая встреченная в исходном тексте цифра была последней цифрой, а последняя первой и был сохранен порядок следования остальных цифр. |
| 10 | Дано предложение, состоящее из слов, разделенных запятой или пробелами, среди которых есть группы цифр, определяющих целые числа из диапазона [0..19]. Отредактировать введенную строку, заменив каждое число, на последовательность, заключенную в круглые скобки, символа +, (если число четное) или на – (если число нечетное), длина которой равна найденному числу. |
| 11 | Дано предложение, состоящее из символов. Вывести это предложение, удалив из него все символы, которые находятся между символами ‘(‘ ‘)’. Сами скобки не удалять, а вместо удаленного текста вставить число, соответствующее количеству удаленных символов. Если хотя бы одной скобки нет, то сообщить об этом. |
| 12 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Распечатать те слова предложения, в которых буквы упорядочены по алфавиту, и удалить те слова, в которых каждая буква входит в слово не менее двух раз. |
| 13 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Найти самое длинное слово палиндром. Заменить слово в предложении числом, определяющим его номер среди палиндромов предложения. |
| 14 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Составить частотный словарь слов, указав, сколько раз данное слово встретилось в предложении. При этом изменить предложение: - удалив повторные вхождения слов и вставив перед первым вхождением слова подстроку вида:(Число), где - Число – определяет количество таких слов в предложении. |
| 15 | Дан произвольный текст, состоящий из слов. Отредактировать его, оставив между словами по одному пробелу, а между предложениями по два. Предложения завершаются символами точка, вопросительный или восклицательный знак. |
| 16 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Удалить из предложения все слова, начинающиеся с гласных букв русского алфавита, а слова, начинающиеся с согласных букв записать прописными буквами. Между словами, заканчивающимися и начинающимися одной буквой, вставить подстроку -\*-. |
| 17 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Среди слов этого предложения найти слова, составленные из тех же букв что и первое слово. Найденные слова переставлять в начало предложения. |
| 18 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. В данном предложении есть слова, представляющие запись вещественного числа в формате с плавающей точкой (т.е.+/-хххх.ххххЕ+/-рррр). Создать массив вещественных чисел. В тексте эти слова заменить словами, представляющими это число в формате с фиксированной точкой и с заданным (для всех чисел) количеством знаков после точки. |
| 19 | Дано предложение, состоящее из слов, разделенных запятой или пробелами. Слова, которые больше последнего слова, заменить на их перевертыши, а слова, которые меньше последнего, занести в массив слов и удалить из предложения. |
| 20 | Дана последовательность чисел (целых и вещественных). Преобразовать каждое число в строковое представление. Объединить полученные строки в текст разделяя их символом пробел. Объединение строки производить пока длина строки не превысит 255 символов. |
| 21 | Дано предложение, состоящее из слов, разделенных пробелами. Среди слов могут быть числа шестнадцатеричного кода, которые начинаются с символа $. Сформировать массив из десятичных значений шестнадцатеричных чисел, а сами шестнадцатеричные числа удалить из предложения. |
| 22 | Даны два предложения. Вывести слова, общие для этих двух предложений. Если таких слов нет, то вывести сообщение об этом. |
| 23 | Дан текст программы на языке С++. Вывести все ключевые слова, встретившиеся в тексте программы и указать их количество. Использовать словарь ключевых слов языка С++ (некоторый набор). |
| 24 | Дано предложение, слова в котором разделены пробелами и запятыми. Распечатать те пары слов, расстояние между которыми наименьшее. Расстояние – это количество позиций, в которых слова различаются. Например, МАМА и ПАПА, МЫШКА и КОШКА расстояние этих пар равно двум. |
| 25 | Дано предложение, слова в котором разделены любыми допустимыми знаками препинания. Сформировать массив из слов, в которых заданная подстрока размещается с первой позиции. |
| 26 | Дан текст, слова в котором разделены любыми допустимыми знаками препинания. Сформировать массив из слов, в которых заданная подстрока размещается в конце слова. |
| 27 | Дан текст, который содержит много подстрок (серий) состоящих из одинаковых символов, но в тексте могут быть последовательности в которых нет повторяющихся символов. Выполнить сжатие текста по алгоритму: серии заменить подстрокой вида (к) символ, где к – количество символов в серии, а последовательности символов не являющиеся серией заменить на строку вида (-к)последовательность символов. Например, строка до сжатия AAAABCDEFOOOOOOO строка после сжатия (4)A(-5)BCDEF(7)O. Определите коэффициент сжатия для большого по объему текста. Реализуйте алгоритм восстановления текста. |
| 28 | Дан текст, содержащий слова, которые разделены пробелами. Сформировать массив слов. Найти пары слов (анаграммы)- при прочтении каждого из которых в обратном порядке образуется другое слово этой пары, например, ТОК,КОТ; ПОЛК, КЛОП; БАР, РАБ. Сформировать двумерный массив этих пар. |
| 29 | Дан текст, содержащий слова, которые разделены пробелами. Сформировать массив слов. Найти самое длинное слово, в котором все буквы различны. Например, лейкопластырь, неряшливость, четырёхдюймовка. |
| 20 | Дан текст, содержащий слова, которые разделены пробелами. Найти слова в этом предложении, которые состоят из тех же букв, что и другое слово предложения. Например, ток, кот; лес, сел; гора, рога. |
| 31 | Расстояние между двумя словами равной длины – количество букв, в которых различаются эти слова. В заданном предложении найти пару наиболее далеко удаленных слов заданной длины. |

# Строки в стиле С (null терминальные строки)

Текст - это последовательность символов. Строка в Си – это массив символов кодировки ASCII расширенная, заканчивающийся символом \0 (null - символ конца строки).

## Строки, как статические массивы

Определение строковых переменных

char ps1[10] =”12345”, // памяти под 10 элементов из них 1 под \0.

При инициализации строковой переменной значение вписывается в выделенную память в направлении слева направо. Так в ps1 значение 1 займет первую ячейку строки с индексом 0, а далее по порядку остальные.

char ps2[]=”12345”; // память под ps2 объемом 6 байт: 5 под значение и один под \0

char ps3[10]=”123456”, // \0 в 6-ой позиции три байта

//инициализация длинными строками\

char ps4[]=”ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff\

aaaaaaaaaaaaaaaaaaaaa”; // символ \ - знак операции продолжения

Объем памяти для переменной строкового типа равен размеру массива, но количество символов в строке на 1 меньше.

В ps3 текст займет 6 первых байт строки и в 6 будет \0. Если определить для ps2 текст больше 10 символов, то компилятор выдаст сообщение об ошибке.

## Строки как указатели

char s[256]; Где s – указатель на первый байт выделенного участка памяти, так как s определена как массив.

char \*str; str – это указатель на строку или просто строка.

## Динамические строки

Создание динамической строки (переменной) из 100 символов:

char \*s1=(char \*) malloc (sizeof(char)\*100); //создание динамической строки под 100 символов

if (s1!=NULL) // проверка создания динамической переменной, если

gets(s1); // ввод значения в переменную

else

puts(“память не выделена”);

}

## Строковая константа

Это текст, заключенный в кавычки, например:˝Текст˝.

Пустая строка определятся константой вида: ˝˝.

## Операции со строкой

### Присваивание

Присваивать строковым переменным константных строк нельзя!!!!!

Можно только инициализировать переменную значением при определении (см. п.1.1.1).

Операции

* присваивать строкам нельзя

char \*s1,\*s2;

char str1[]=”12345”; //это можно

char str2[10];

str2=str1; // так нельзя

s1=”hhhhhhh”; // так нельзя

s2=”ffffffff”; // так нельзя

* инициализировать можно, но нежелательно.

int main()

{

char \*ps="1234"

cout<<ps; //выведет 1234

cin.get();

cin.get();

return 0;

}

* Вводить в переменные – указатели на строку, если не выделена память, нельзя

{

char \*s1,\*s2;

gets(s1); // ошибка времени выполнения, так как память для s1 не выделена.

}

* Присваивать однотипные указатели можно. В этом случае два указателя связаны с одной строкой.

nt \_tmain(int argc, \_TCHAR\* argv[])

{

char \*ps="1234",//и добавляет \0

s2[256];

int i;

puts("Введите строку");

scanf("%s",s2); //12345678

ps=s2;

puts(ps); //12345678

return 0;

}

### Ввод-вывод строк в стиле С++

1. Стандартный поток ввода с клавиатуры **cin**

*cin>>строковая переменная;* читает текст **до** первого разделителя: пробел, табуляция, \n.

1. Стандартный поток вывода на экран – **cout**

Выводит все значение, содержащееся в переменной или значение константы до символа \0. Выполняет символы esc-последовательности строки.

*сout<<строковая константа;*

*сout<<строковая переменная;*

Пример результата ввода текста стандартным потоком cin

char s[100];

cin>>s; .// желали ввести текст: *Введет только до пробела*

cout<<s; // а увидели только слово *Введет*

*cout.put(char)* //выводит символ

1. Методы стандартного потока cin для ввода всего текста

*istream \* cin.getline(char \*s1, int K,char ch=’\n’*);//читает строку и \n и записывает в s1 K-1 символов, в позиции с 0 по K-2, а в позицию K-1 записывает \0 значение К должно быть равно или больше количества вводимых символов

*istream \* cin.get(char \*s1, int K,char ch=’\n’*);//читает строку до \n (его не читает) и записывает в s1 K-1 символов, в позиции с 0 по K-2, а в позицию K-1 записывает \0 *значение К должно быть равно или больше количества вводимых символов*. Иначе создается блокировка ввода строки.

Пример 4. Варианты ввода строковых значений оператором cin.getline()

{

char ps1[256],ps2[256];

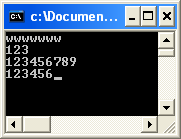
cin.getline(ps1,256,’\n’); //читает строку до \n, т.е. если ввести текст 123 то в ps1 будет строка 123

cin.getline(ps2,7); //Может быть заблокирован дальнейший ввод,

//если вводимых символов будет больше 7: //следующий оператор прочитает оставшиеся //символы

cin.getline(ps2,10,’#’); // читает всю строку до \n, но вводит в //переменную только до # Например, //aaaaaaa#aaa\n в ps1 будет текст aaaaaaa

}



### Ввод-вывод строк в стиле Си

Для корректного использования функций Си в программе с компилятором Microsoft, включите в приложение следующие директивы препроцессора

#if defined(\_WIN32) || defined(\_\_WIN32\_\_) || defined(WIN32)

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

#endif

//далее код программы

#include <iostream>

#include <stdlib.h>

#include "stdio.h"

……………………………..

Функции ввода строки c клавиатуры

1. *scanf (“%s”,строковая переменная);*

Читает текст, *вводимый* с клавиатуры до пробела.

Например. Текст вводимый с клавиатуры: *Это надо знать!*

char str[1024];

scanf("%s", &str);в переменную str будет введено только слово *Это*

1. gets\_s(строковая переменная); читает весь текст водимый с клавиатуры до \n.

Например. Текст вводимый с клавиатуры: *Это надо знать!*

char str[100]="";

gets\_s(str); в переменную str будет введен текст: *Это надо знать!*

Функции вывода строки на экран

1. printf(“%s”, строковая константа или переменная);
2. puts(строковая константа или переменная)

В стиле С++ (потоковый ввод)

*int cin.get(char* &); читает символ из потока в переменную, указанную в параметре и возвращает поток

*char cin.get();* читает символ из потока и возвращает его в качестве результата

*cin>>*символьная переменная;

Пример 5. Заполнение строк посимвольно: *вводить как массив поэлементно, но надо добавить в конце \0, чтобы далее работать как со строкой.*

int main()

{

char ps4[5]

for(int i=0;i<4;i++)

cin.get(ps4[i]);

*ps4[i]=’\0’;*//для корректной работы со строкой

// иначе будут обрабатываться все ячейки,

//следующие за последним введенным символом

cout<<ps4;

return 0;

}

### Примеры на выделение слов из текста

Пример 6. Дано предложение, состоящее из слов. Слова отделяются друг от друга одним пробелом. Красной строки нет. Вывести все слова текста. Текст рассматривать как строку максимальный размер 1024 символа, слово тоже строка размером не более 20 символов.

*Метод решения: полный перебор символов и формирование слова, как массива символов*

Тест, строка слов - чисел: 12 34 56 78

void slova(char \*s)

{

char slovo[20];

int i,j;i=0;

while(s[i]!='\0') //до конца текста

{

j=0;

while(s[i]!='\0' && (s[i])!=' ') //до конца слова

{

slovo[j]=s[i];i++;j++;

}

slovo[j]='\0';

puts(slovo);

if(s[i]!='\0')

{

i++;//уйти с разделителя слова, если оно не последнее

}

}

}

int main()

{

char str[1024];

cout<<"Vvedite tekst";

gets(str);

slova(str);

return 0;

}

Пример 7. Дано предложение, состоящее из слов. Слова отделяются друг от друга одним или несколькими пробелами. Есть красная строка. Вывести все слова текста. Текст рассматривать как строку максимальный размер 1024 символа, слово тоже строка размером не более 20 символов.

*Метод решения:*

*- полный перебор символов и формирование слова, как массива символов;*

*-пропуск пробелов красной строки;*

*- формирование слова;*

*- пропуск пробелов между словами*

Тест текста из слов чисел с несколькими пробелами между словами:

12 34 56 78

void slova2(char \*s)

{

char slovo[20];

int i,j;i=0;

while(s[i]!='\0')// обработка предложения

{

while(s[i]!='\0' && (s[i])==' ') //пропуск пробелов

{

i++;

}

if(s[i]!='\0') //формирование слова

{

j=0;

while(s[i]!='\0' && (s[i])!=' ')

{

slovo[j]=s[i];i++;j++;

}

slovo[j]='\0';puts(slovo);

if(s[i]!='\0') i++;

}

}}

int main()

{

char str[1024];

char \*\*mas=0;

int l;

char \*razd=" ,.!?";

cout<<"Vvedite tekst";

gets(str);

slova2(str);

return 0;

}

Пример 8. Дано предложение, состоящее из слов. Слова отделяются друг от друга одним или несколькими пробелами или другими символами разделителями, например: запятая, дефис, точка. Есть красная строка . Вывести все слова текста. Текст рассматривать как строку максимальный размер 1024 символа, слово тоже строка размером не более 20 символов.

*Метод решения:*

*- полный перебор символов и формирование слова, как массива символов;*

*-пропуск пробелов красной строки и других разделителей;*

*- формирование слова;*

*- пропуск пробелов и других разделителей между словами*

void slova3(char \*s)

{

char slovo[20];

int i,j;i=0;

while(s[i]!='\0')

{

while(s[i]!='\0' && (s[i]==' '|| (s[i]==','|| s[i]=='.'|| s[i]=='?'))) //пропуск разделителей

{

i++;

}

if(s[i]!='\0') //формирование слова

{

j=0;

while(s[i]!='\0' && (s[i])!=' '&& s[i]!=','&& s[i]!='.'&& s[i]!='?')

{

slovo[j]=s[i];i++;j++;

}

slovo[j]='\0';puts(slovo);

if(s[i]!='\0') i++;

}

}}

int main()

{

char str[1024];

char \*\*mas=0;

int l;

char \*razd=" ,.!?";

cout<<"Vvedite tekst";

gets(str);

slova3(str);

return 0;

}

Пример 9. Дано предложение, состоящее из слов. Слова отделяются друг от друга одним или несколькими пробелами или другими символами разделителями, например: запятая, дефис, точка. Есть красная строка. Вывести все слова текста. Текст рассматривать как строку максимальный размер 1024 символа, слово тоже строка размером не более 20 символов.

Удаление обработанных слов и разделителей из текста на основе указателя.

*Метод решения:*

*- полный перебор символов и формирование слова, как массива символов;*

*-пропуск пробелов красной строки и других разделителей;*

*-сместить указатель на начало слова (после разделителей);*

*- формирование слова, вывод;*

*- смещение указателя на длину слова, т.е. на начало разделителей*

void slova4(char \*s)

{

char slovo[20];

int i,j;i=0;

while(s[i]!='\0')

{

//пропуск разделителей

while(s[i]!='\0' && (s[i]==' '||(s[i]==','||s[i]=='.'||s[i]=='?')))

{

i++;

}

if(s[i]!='\0') //формирование слова

{s=s+i;i=0; //удаление разделителей, путем изменения

//указателя сдвинув его на i cимволов

// вправо и //корректирока индекса i в 0.

j=0;

while(s[i]!='\0' && (s[i])!=' '&& s[i]!=','&& s[i]!='.'&& s[i]!='?')

{

slovo[j]=s[i];i++;j++;

}

slovo[j]='\0';puts(slovo);

s=s+j;

if(s[i]!='\0') i++;

}

}}

int main()

{

char str[1024];

char \*\*mas=0;

int l;

char \*razd=" ,.!?";

cout<<"Vvedite tekst";

gets(str);

//create\_mass\_word(str,razd, mas, &l);

//out\_massiv(l,mas);

slova4(str);

return 0;

}

### Функции для строк <cstring.h> или <string.h>

1. Определение длины строки

*int strlen(char \*);* Результат- количество символов в строке

Пример использования

int main()

{

char s[56]=˝abcde”;

cout<<strlen(s); //вывод 5

}

1. Cцепление (конкатенация) строк

*char \*srtcat(char \*s1,char \*s2)* Добавляет значение s2 к строке s1 и результат в строка - s1. Возвращает строку s1.

Пример использования

int main()

{

…..

gets(s1); //123

gets(s2); //456

char \* s4=strcat(s1,s2);

puts(s1); //123456

puts(s4); //123456

}

1. Поиск символа в строке.

char \**strchar*(char \*s1,int ch) – Возвращает строку - указатель на первое вхождение символа ch в строку s1 или NULL если символа нет.

Пример использования

int main()

{

char s4[256]=”SDFGHJAKLU”;

char \*s4=strchr(s4,'A'); //результат - текст начинающийся с буквы А

if (s4= =NULL)

cout<<"Нет символа в строке";

else

puts(s4);

}

Результат

AKLU

1. Копирование строк (присваивание)

*char \* strcpy(char \*s1,char \*s2)* копирует строку s2 в s1 – результат s1 и возвращает указатель на строку s1.

Пример использования функции *strcpy*

int main()

{

char strr[100]="";

gets\_s(strr);// ввели 123 456

char masstr[100] = "";

strcpy(masstr, strr);

puts(masstr); // увидели 123 456

}

*char \*strncpy(char \*s1,char \*s2, int n)* –копирует первые n символов строки s2 в s1 – результат s1. Если в строке s1 количество символов больше или равно n, то \0 в конец строки не вставляется, иначе строка заполняется \0 до n – ого символа.

Пример использования функции *strncpy*

int main()

{

char s1[]="AAAAA";

char s2[]="BBB";

char s3[10];

puts(strncpy(s1,s2,2)); //результат BBAAA

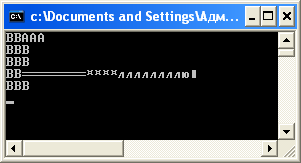
puts(strncpy(s1,s2,5)); //результат BBB

puts(strncpy(s1,s2,7)); //результат BBB

puts(strncpy(s3,s2,2)); //результат на скриншоте

puts(strncpy(s3,s2,10)); //результат на скриншоте

return 0;}



1. Сравнение строк.

Строки равны, если их длины равны и они посимвольно равны (“abc”==”abc”).

Если длины строк равны, то из двух сравниваемых строк больше та, первый не совпавший символ которой имеет больший код: (“abd”>”abc”).

Если длины не равны, то больше та, длина которой больше.

*int strcmp(char \*s1,char \*s2)* *сравнивает значения строк s1 и s2*

Результат:

Если s1<s2 то результат - отрицательное число

Если s1=s2 то результат равен 0

Если s1>s2 то результат – положительное число

*int strncmp(char \*s1,char \*s2,int n)* сравнивает первые n символов строки s2 со строкой s1, результат аналогичен функции *strcmp*.

Пример использования функции *strcmp*

char s1[]="AAB";

char s2[]="AAA";

int flag=strcmp(s1,s2); // flag=1 т.е. значение строки s1>s2

char s1[]="AA";

char s2[]="AA";

int flag=strcmp(s1,s2); // flag=0 т.е. значение строки s1=s2

char s1[]="AA";

char s2[]="AAA";

int flag=strcmp(s1,s2); // flag=-1 т.е. значение строки s1<s2

1. Поиск подстроки в строке

*char \*strstr(char \*s1,char \*s2)* ищет первое вхождение строки s2 в s1. Возвращает(результат) строку - указатель на первый символ строки s1, если s2 вошла целиком в s1, или NULL в противном случае.

Пример использования функции *strstr*

сhar s1=”AAAAAABBBGBBBBAAAAAA”;

сhar s2=”BB”;

сhar \*s=strstr(s1,s2); //результат- текст, начинающийся с ВВ

if (s==NULL)

puts("NOT");;

else

puts(s); //BBBGBBBBAAAAAA

1. Выделение слов (лексем) из текста.

*сhar \*strtok(char \*s1,char \*s2) где* s1 – строка с текстом; s2 – строка – символов *разделителей* слов в тексте

Функция возвращает указатель на слово в строке s1, после которого стоит один из символов строки s2 и ставит \0 на место этого символа разделителя в s1.

## Примеры применения функции strtok для выделения слов в тексте

Пример 8.1. Вывести слово, которое отделено от другого одним из символов, включенных в строку " ,.!"

char s1[]="Мы изучаем функции для строк в Си";

char s2[]=" ,.!"; //строка символов разделителей: пробел, запятая точка,!

char \*s=strtok(s1,s2); //результат – указатель на слово в строке s1

if (s==NULL)

cout<<”Нет разделителей”;

else

puts(s); //вывод Мы

Пример 8.2 Вывод всех слов текста.

Использование функции strtok для разбиения строки на слова, за которыми следует один из заданных разделителей.

char s1[]="Мы изучаем функции для строк в Си ";

char s2[]=" ,.!";

char \*s=strtok(s1,s2); //указатель на первое слово

while (s!=NULL)//пока не достигнем конца строки (\0 или NULL)

{

puts(s); вывод строки на которую указывает указатель s

s=strtok(NULL,s2); //указатель на слово между \0 и символом разделителем

}

После этих действий строка s1 будет содержать:

Мы**\0**изучаем**\0**функции**\0**для**\0**строк**\0**в**\0**Си**\0**

Пример 8.3. Сформировать массив слов заданного предложения. Считать, то в тексте не более 10 слов.

{

char s1[1024];

char \*s2[10]; массив из 10 указателей

char s3[]=" ,.!?";

int i=0;

gets(s1);

s2[0]=strtok(s1,s3);

while(s2[i]!=NULL)

{

puts(s2[i]);

s2[++i]=strtok(NULL,s3);

}

}

Пример 8.4. Формирование динамического массива слов – свободного массива. Свободный массив – это массив указателей на строки переменной длины.

Дано предложение. Сформировать массив из слов – чисел.

#include "string.h"

#include "stdlib.h"

#include "iostream"

using namespace std;

void create\_mass\_word(char \*s,char \*razd, char\*\*&mas, int &k)

{

char \*s1=strtok(s,razd);

int value;

char \*ss=0; //код завершения преобразованияв числ.формат

k=0;

mas=(char \*\*)malloc(4); //создание массива из одного элемента длиной 4 байта

mas[0]=ss;

while(s1!=NULL)

{

puts(s1);

value=(int)strtod(s1,&ss);//преобразование строкового значения в числовой формат

if(!(strcmp(ss,""))) //если выполнено преобразование ss пусто, т.е.найдено число

{ if(k)

{

mas=(char\*\*)realloc(mas, ((\*k)+1)\*4);

}

mas[k]=s1;(k)++;

}

s1=strtok(NULL,razd);

}

}

void out\_massiv(int n, char \*\*mas)

{

int i;

for(i=0;i<n;i++)

{

puts(mas[i]);

}

}

int main()

{

char str[1024];

char \*\*mas=0;

int l;

char \*razd=" ,.!?";

cout<<"Vvedite tekst";

gets(str);

strcat(str," ");

create\_mass\_word(str,razd, mas, &l);

out\_massiv(l,mas);

return 0;

}

Пример 10. Дан текст, в котором есть группы повторяющихся символов. Сжать текст, заменив группы повторяющихся символов на строку (к) символ для к>4. Например:ASDFFFFFFFFhJJJJJJJkkLLLL

#include "string.h"

int main()

{

char s1[100]="ASDFFFFFFFFhJJJJJJJkkLLLL";

unsigned int i=0,i1;

char str[5];

while(i<strlen(s1))

{

i1=i;

while(i<strlen(s1) && s1[i]==s1[i+1])

{

i++;

}

int k=i-i1+1; //количество повторяющихся символов в группе

if(k>1)

{

s1[i1]='(';

while(k!=0)

{

s1[++i1]=(k%10+'0');

k=k/10;

}

s1[++i1]=')';

k=i-i1-1; //количество оставшихся в группе символов с самим символом

// удаление оставшихся символов в повторяющейся группе

for(unsigned int i2=i1+1;i2<strlen(s1);i2++)

{

s1[i2]=s1[i2+k];

}

i=i1+1; //корректировка i

}

i++;

}

puts(s1);

}

1. Функции преобразования из строкового формата в числовой.

Функции преобразования: atoi, atol, atof

Строковое представление числа в числовой формат.

Преобразование числа из строкового формата в числовой формат выполниться верно, если строковое представление числа записано по правилам представления числа в языке программирования.

Если число начинается с цифр и в числе есть символы, отличные от цифр и знака, то функция преобразования выдаст неверный результат, число до символа, не допустимого в числе.

Применение функций в соответствии с типом числовой переменной:

int a=atoi("123"); //в формат int

long l=atol("123456"); // в формат long

double d=atof("123.45");//в формат вещественного

Пример применния функций для корректного и некорректного преобразования

int main()

{

int x;

char str[5]="124";//правильное число, резульат 124

x=atoi(str);

cout<<x<<'\n';

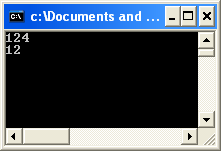
char str1[5]="12a4";//неверно представлено число, результат=12

x=atoi(str1);

cout<<x<<'\n';

}

Результат



1. Функции преобразования: strtod, strtol

*double strtod(const char \*s, char \*\*ch);*Преобразует число из строкового формата переменной s формат вещественного числа.

При успешном выполнении преобразования числа из строки s в, функция возвращает в качестве результата число в формате вещественного. Значение возвращаемой параметром ch=””.

При не успешном выполнении преобразования функция формирует код завершения и первый символ строки s, который не допустим при записи числа помещает в переменную ch.

Пример корректного и некорректного преобразования и применение при проверке ввода числовых данных.

#include "string.h"

#include "iostream"

using namespace std;

int main()

{

char str2[5]="12.4";

char \*ss;

double y=strtod(str2,&ss);

if(!(strcmp(ss,""))) //если выполнено преобразование,т.е.\*ss пусто

{

cout<<"y="<<y;

}else

{

cout<<"ss="<<ss<<' '<<"y="<<y;

}

cout<<'\n';

char str3[5]="12**a**4";

y=strtod(str3,&ss);

if(!(strcmp(ss,"")))

{

cout<<"y="<<y;

}else //если не выполнено преобразование, т.е.\*ss не пусто

{

cout<<"ss="<<\*ss<<' '<<"y="<<y;

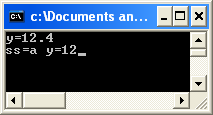
}

}

Результат выполнения

Первая строка на скриншоте это результат успешного преобразования

Вторая строка – результат не успешного преобразования и первый символ, на котором завершилось преобразование – это символ **а**, была преобразована только часть числа до символа **а**.



*double strtol(const char \*s, char \*\*ch, int R);*

Преобразует число из s в вещественный формат с учетом системы счисления R.

Функция возвращает вещественное число в системе счисления R, если код завершения в ch равен пусто, иначе в ch символ.

1. Преобразование из числового формата в строковый : sprint, itoa, ltoa, utoa, ultoa

*sprintf(char \*s, const char \*format, выражение числового формата);*

Выражение числового формата преобразуется по format в s.

Пример преобразования числа, заданного в формате *format,* в строковый формат

{

char buffer[12];

unsigned int value = 12345678ul;

sprintf(buffer, "%lu", value );

}

Примечание. После чего в массиве buffer лежит требуемая строка. Но, sprintf это функция форматированного вывода, которая много чего умеет и тянет за собой много других функций стандартной библиотеки. Размер машинного кода при ее использовании увеличивается значительно.

Функции *itoa, ltoa, utoa, ultoa*

В состав библиотек, поставляемых с компиляторами, часто включают функции преобразования числа в строку itoa, ltoa, utoa, ultoa.

Эти функции не стандартные, но часто имеются в наличии в stdlib.h, в отличии от sprintf, не делают ничего лишнего.

*char \*itoa( int value, char \* string, int radix ); // в int*

*char \*ltoa( int value, char \* string, int radix ); // в long*

*char \*utoa( int value, char \* string, int radix ); // в unsigned int*

*char \*ultoa( int value, char \* string, int radix ); // в unsigned long*

где: value преобразуется в string в формате системы счисления radix.

Пример применения функций на функции ultoa

char buffer[12];  
 unsigned int value = 12345678ul;  
 ultoa(value, buffer, 10); //в десятичную систему счисления

Чтобы отследить неправильное преобразование, используйте функцию \_itoa\_s, формат которой:

*char \*\_itoa\_s( int value, char \* string, int size, int radix );*

*где* size — размер строки string. Данная функция возвращает ноль, если преобразование проходит успешно.

*#include <cstdlib> /\* или #include <stdlib.h> \*/*

*char \*\_itoa(int val, char \*dest, int radix);*

*errno\_t \_itoa\_s(int val, char \*dest, size\_t size, int radix);*

В параметре val указывается число, в параметре dest — указатель на символьный массив, в который будет записан результат, в параметре size — максимальный размер символьного массива dest, а в параметре radix — система счисления (2, 8, 10 или 16). Пример использования функции \_itoa\_s():

char str[100] = {0};

int x = 255;

\_itoa\_s(x, str, 100, 10);

std::cout << str << std::endl; // 255

Пример вывода двоичного, восьмеричного и шестнадцатеричного значений:

char str[100] = {0};

// Двоичное значение

std::cout << \_itoa(100, str, 2) << std::endl; // 1100100

// Восьмеричное значение

std::cout << \_itoa(10, str, 8) << std::endl; // 12

// Шестнадцатеричное значение

std::cout << \_itoa(255, str, 16) << std::endl; // ff

# Шаблонная строка класса string

## Класс string

Класс *string* определен в стандартной библиотеке шаблонов STL. Для его использования необходимо подключить заголовочный файл <string>.

## Конструкторы

string s1;

string s2(const char \*х); записывает в строку значение параметра х – строка в стиле Си.

string s3(const char \*x,int n); записывает в строку n символов из значения параметра х - строка в стиле Си.

string s4(string &); .// инициализация сушествующей строкой.

Формат имени экземпляра класса string:

*[Имя пространства имен]::имя объекта*

Если указать в коде программы явно стандартное пространство имен, то в имени его можно опустить.

Подключение пространства имен выполняется оператором:

*using namespace std;*

Примечание. Явное указание пространства имен в имени объекта считается хорошим стилем.

## Ввод строки с клавиатуры и вывод в консоль

string s;

cin>>s; //до разделителя

s.puts();

getline(cin, s); //весь текст

cout<<s;

//Присваивание строкам значений

std::s1=’x’; //явно указано пространство имен

s2=”asdfg”; // в код должен быть включен оператор *using namespace std;*

s3=”1234567”;

s4=s3;

Пример Операции ввода, вывода, присваивания строкам значений

#include <iostream>

#include <string>

using namespace std;

int main()

{

string s1;

string s2("ABC");;

string s3("ABCDERTY",4);

string s4(s2);

cout<<s2; //ABC

cout<<s3; //ABCDERTY

cout<<s4; //ABC

s1="GFFF";

cout<<s1<<endl; //GFFF

s1='x';

s2="asdfg";

s4=s3; //копирование строк

cout<<s1<<endl; // x

cout<<s2<<endl; //asdfg

cout<<s3<<endl; //ABCDERTY

cout<<s4<<endl; //ABCDERTY

cin>>s1; //до пробела 123 456 78

cin.get(); //троки пробел

getline(cin,s2);//всю оставшуюся часть

cout<<s1<<endl; //123

cout<<s2<<endl; //456 78

cin.get();

return 0;

}

## Операции

В этих операциях могут смешиваться строки, завершающиеся нулем со строкой string

|  |  |  |  |
| --- | --- | --- | --- |
| = | присваивание | > | больше |
| + | конкатенация | >= |  |
| == | Сравнение на равенство | [] | Индексация(доступ к отдельному символу в строке) |
| != | Сравнение на не равенство | << | вывод |
| < | меньше | >> | ввод |
| <= | Меньше или равно | += | добавление |

Пример применения операций к строкам string

int main()

{

string ss;

string s;

getline(cin,ss); //ввод

cout<<ss<<endl; //вывод

cin>>s; //ввод

cout<<s<<endl;

string sss=ss+s; //конкатенация

cout<<sss;

cout<<">"<<(ss>s)<<endl; //сравнение строк

cout<<"=="<<(ss==s)<<endl;

cout<<"!="<<(ss!=s)<<endl;

cout<<ss.length()<<endl; //метод – длина строкаи

cout<<"c="<<sss.find(s,1)<<endl;//метод поиска в строке sss подстроки s,

//начиная с заданной позиции - 1

cout<<s.find('5',1); //символ в строке s

cin.get();

cin.get();

return 0;

}

## Методы

size\_type – ‘это тип для методов, представляет форму беззнакового целого, которая может быть автоматически конвертирована в целый тип.

Функции заголовочного файла string.

## Общие методы

1. Максимальный *объем* строки string

*size\_type max\_size() const;*

string S;

int i=S.max\_size(); //i=-2

cout<<S.max\_size(); //4294967294

1. Текущая *длина* строки string – количество символов в строке

Два метода:

*size\_type size() const;* //поддерживают все контейнера STL

*size\_type length() const;*

string S("123456");

int i=S.max\_size();

cout<<S.size()<<" "<<i; //6 -2

1. Определение текущей *емкости* строки string. Емкость строки – это начальный объем строки и ее увеличение на 16 байт при расширении строки.

*size\_type capacity() const;*

string S("123456");

int i=S.capacity();

cout<<"Начальная емкость="<<S.capacity()<<"\n"; //=15

S+="678567";

cout<<"Текущая емкость="<<S.capacity()<<"\n";//=15

S+="67856";

cout<<"Текущая емкость="<<S.capacity()<<"\n";//=31

1. Установить емкость строки заданного значения.

void reserve (int capacit);

Исключение перераспределения памяти. Если строка в переменной больше чем устанавливаемая емкость, то емкость уменьшается до указанного размера.

string S;

int i=S.capacity();

cout<<"Начальная емкость="<<S.capacity()<<" "<<S<<"\n";//=15

S.reserve(20);

cout<<"Текущая емкость="<<S.capacity()<<" "<<S<<"\n";//=31

S.reserve(128);

cout<<"Текущая емкость="<<S.capacity()<<" "<<S<<"\n";//=143

1. Копирование части строки

*string substr(size\_type indx=0, size\_type len=npos);*

Копирует подстроку длиной len, начиная с символа в позиции indx.

string S="AAADDDDDCCCCCC";

string SS=S.substr(3,5);

cout<<"Результат"<<SS<<"\n"; //DDDDD

1. Доступ к элементу строки.

Два варианта:

1. Перегруженная операция []
2. Метод ***at –*** возвращает символ в позиции indx

*char &at(size\_type indx)*

string S="AAADDDDDCCCCCC";

for(int i=0; i<S.length();i++)

cout<<S[i];

cout<<"\n";

for(int i=0; i<S.length();i++)

cout<<S.at(i);

1. Содержит ли строка значение (пуста ли строка)

*bool empty();*

Возвращает true ли строка пуста и false иначе.

string S="AAADDDDDCCCCCC";

cout<<S.empty(); //результат - false

1. Сравнение строк при применении методов.

*int compare(size\_type indx, size\_type len, const string &str) const;*

Сравнивает часть вызывающей строки начиная с элемента по индексу indx длиной len со строкой str.

Если подстрока меньше str, то результат <0.

Если подстрока больше str, то результат >0.

Если подстрока равна str, то результат =0.

string s(“Что?Где?Когда?”);

int rez=s.compare(4,4,”Где?”);

1. Преобразование строки в массив символов

char\* c\_str() const;

string S="AAADDDDDCCCCCC";

const char \*mas=S.c\_str();

cout<<mas;

## Методы модификации строки

1. Удаление символов из строки или очистка строки.

*string &erase(size\_type indx=0, size\_type len=npos);*

Удаляет len символjd, начиная с позиции indx.

string S="AAADDDDDCCCCCC";

S.erase(3,5);

cout<<"Результат"<<S<<"\n"; //AAACCCCCC

1. Добавляет символ в конец строки.

*void* ***push****\_back(const char ch);* Добавляет символ ch в конец строки.

//добавление символа С в конец строки

string S="AAADDDDD";

S.push\_back('C');

cout<<"Результат"<<S<<"\n"; // AAADDDDDС

S.push\_back('C');

1. Вставить подстроку в вызывающую строку с заданной позиции

*void* ***insert****(unsigned int indx, const string &s);*

string sA="fff hhh jjjjjkkk, 1234";

sA.insert(5,"dddddd");

cout<<sA<<endl;

1. Замена в вызывающей строке подстроки, начиная с элемента с индексом indx длиной len на строку str.

*string &replace (size\_type indx, size\_type len, const string &str);*

## Методы поиска данных в строке

1. Поиск первого вхождения строки в строку

*size\_type finde(const string &str, size\_type indx = 0) const;*

*size\_type finde(const char \*str, size\_type indx = 0) const;*

Возвращают индекс первого вхождения строки str в вызывающую строку после индекса indx.

string S="Что?Где?Когда?";

int i=S.find("Когда?");

cout<<i<<"\n"; // 8

i=S.find("?",8);

cout<<i<<"\n"; // 13

1. Поиск последнего вхождения строки в строку до символа с индексом indx.

*size\_type rfinde(const string &str, size\_type indx = npos) const;*

Если никакого соответствия не найдено, возвращает npos (соответствует NULL).

string S="Что?Где?Когда?";

int i=S.find("Когда?");

cout<<i<<"\n";

i=S.find("?",4); //результат 3

cout<<i<<"\n";

S="Что?ГдеКогда?";

i=S.find("?",8); //результат 3

1. Поиск первого вхождения любого символа из заданного набора в вызывающей строке.

*size\_type finde\_first\_of(const string &str, size\_type indx = 0) const;*

*size\_type finde\_first\_of(const char \*str, size\_type indx = 0) const;*

где str – набор символов поиска.

string S="ABCDEBBCDFG";

int i=S.find\_first\_of("DFG",1);

cout<<i<<"\n"; //результат 3по смволу D

1. Поиск первого вхождения любого символа из заданного набора не входящего в вызывающую строку.

*size\_type finde\_first\_not\_of(const string &str, size\_type indx = 0) const;*

*size\_type finde\_first\_not\_of(const char \*str, size\_type indx = 0) const;*

1. Поиск последнего вхождения любого символа из заданного набора входящего в вызывающую строку.

*size\_type finde\_last \_of(const string &str, size\_type indx = 0) const;*

1. Поиск последнего вхождения любого символа из заданного набора в не входящего в вызывающую строку.

*size\_type finde\_last\_not\_of(const string &str, size\_type indx = 0) const;*

## Пример. Разбиение строки на лексемы (слова), отделяемые друг от друга символами разделителями из заданного набора.

Создадим класс для строки и реализуем метод аналогичный функции strtok нуль терминальной строки.

class tokenizer{

string s;

string::size\_type startidx;// начальный индекс

string::size\_type endidx; //последний индекс

public:

tokenizer(const string &str){

s=str;

startidx=0;

}

// вернуть лексему

string get\_token(const string &delims)

{

if(startidx==string::npos) return string("");

//найти следуюий разделитель

endidx=s.find\_first\_of(delims,startidx);

//Скопировать строку, содержащую лексему

string tok(s.substr(startidx, endidx-startidx));

//найти начало следующей лексемы

startidx=s.find\_first\_not\_of(delims,endidx);

return tok;

}

};

int main()

{

string sA="fff hhh jjjjjkkk, 1234";

string sB="123 45 678, 1234";

string delimsiter(" ,!?.\n");

string token; //для лексемы

//объекты

tokenizer tokA(sA);

tokenizer tokB(sB);

//слова tokA

token=tokA.get\_token(delimsiter);

while(token!="")

{

cout<<token<<endl;

token=tokB.get\_token(delimsiter);

}

//слова tokB

token=tokB.get\_token(delimsiter);

while(token!="")

{

cout<<token<<endl;

token=tokA.get\_token(delimsiter);

}

cin.get();

return 0;

}