Строки

Строки

- Строка — последовательность (массив) символов.
- Строки в C/C++ представляются как массивы элементов типа char, заканчивающиеся **нуль-терминатором \0**
- Символьные строки состоят из набора символьных констант заключённых в двойные кавычки.
- При объявлении строкового массива необходимо учитывать наличие в конце строки нуль-терминатора, и отводить дополнительный байт под него.

Пример

```
char string[10];
```

- string – имя строковой переменной
- 10 – размер массива, в данной строке может поместиться 9 символов , последнее место отводится под нуль-терминатор.

Строки

- Строка может содержать символы, цифры и специальные знаки.
- Строки заключаются в двойные кавычки.
- Имя строки является константным указателем на первый символ.

Инициализация строки

```
char string[10] = "abcdefghf";
```

Посимвольная инициализация строки:

```
char string[10] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'f', '\0'};
```

Десятый символ - это нуль-терминатор.

Инициализация строки без указания размера:

```
char string[] = "abcdefghf";
```

Инициализация строки

Массив строк:

```
char s[3][25] = {"Пример", "использования", "строк"};
```

Массив из 3х строк по 25 байт каждая.

Возможности работы со строками

- функции стандартной библиотеки C;
- библиотечный класс C++ string;

Функции стандартной библиотеки C

- копирования строк (strcpy, strncpy);
- сравнения (strcmp);
- объединения строк (strcat, strncat);
- поиска подстроки (strstr);
- поиска вхождения символа (strchr, strrchr, strpbrk);
- определения длины строки (strlen);
- и др.

Заголовочные файлы

- <ctype.h>, <cctype> - содержат объявления функций для классификации и преобразования отдельных символов;
- <stdlib.h>, <cstdlib> - содержат в себе функции, занимающиеся выделением памяти, контроль процесса выполнения программы, преобразования типов и другие.

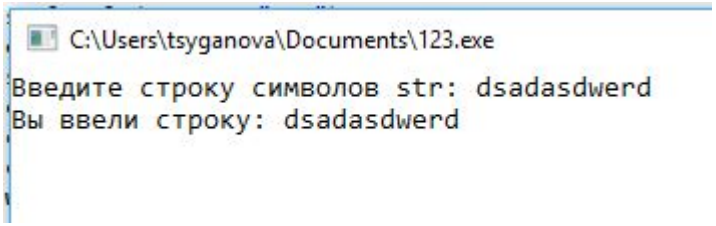
Ввод – вывод строк

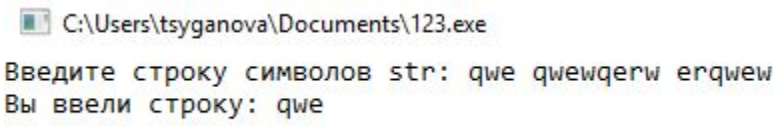
Использование объектов cout и cin

Программа 8.

Ввод-вывод строк с использованием объектов cout, cin

```
// Пример 1
// Ввод-вывод строк с использованием объектов cout, cin.
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
        setlocale(LC_ALL, "rus");
        char str[80];
        system ("cls");
        cout << "Введите строку символов str: ";
        cin >> str;
        cout << "Вы ввели строку: " << str << endl;
        while (!kbhit());
            return 0;
}
```





Использование методов getline или get класса iostream

Функции предназначены для ввода данных из потока, например, для ввода данных из консольного окна.

Формат вызова методов:

cin.getline (s,n)

cin.get(s,n)

Использование методов getline или get класса iostream

- Метод **getline** считывает из входного потока **(n-1)** символов или менее и записывает их в строковую пере менную **s**.
- Символ перевода строки также считывается из входного потока, но не записывается в строковую переменную, вместо него размещается завершающий \0.
- Символ перевода строки **‘\n’** появляется во входном потоке после нажатия клавиши Enter.

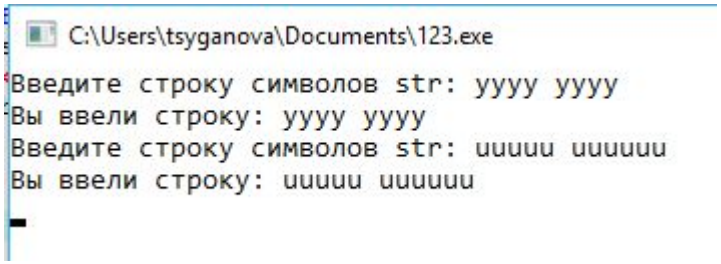
- Метод **get** работает аналогично, но не оставляет в потоке символ перевода строки. В строковую переменную добавляется завершающий \0.

Программа 9

Ввод-вывод строк с использованием методов getline и get

```
// Пример 2
// Ввод-вывод строк с использованием методов getline и get
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
        setlocale(LC_ALL, "rus");
        char str[80];
        const int n=80;
        cout << "Введите строку символов str: ";
        cin.getline (str,n);
        cout << "Вы ввели строку: " << str << endl;
        cout << "Введите строку символов str: ";
        cin.get (str,n);
        cout << "Вы ввели строку: " << str << endl;
        while (!kbhit());
          return 0;

}
```



Функции ввода-вывода библиотеки С

- scanf();
- printf ();
- gets ();
- puts ().

scanf()

- Является процедурой ввода общего назначения, считывающей данные из потока stdin.
- Может считывать данные всех базовых типов и автоматически конвертировать их в нужный внутренний формат.

Формат:

scanf(const char *format, arg-list)

scanf()

Код	Значение
%c	Считать один символ
%d	Считать десятичное число целого типа
%i	Считать целое число в любом формате
%e %f %g	Считать число с плавающей точкой
%o	Считать восьмеричное число
%s	**Считать строку**
%x	Считать шестнадцатеричное число
%p	Считать указатель
%n	Принять целое значение, равное кол-ву прочитанных символов
%u	Считать десятичное целое без знака
%%	Считать знак процента

Примеры scanf()

- scanf("%d", &co); - считать целое число и присвоить его переменной **co**

- scanf("%s", address); - считать строку и сохранить ее в массив **address**

printf()

- Записывает в stdout аргументы из списка arg-list под управлением строки, на которую указывает аргумент format.

- **Формат:**
- **printf(const char *format, arg-list)**

printf()

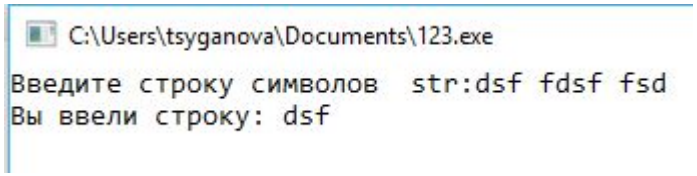
Код	Формат
%c	Символ типа char
%d %i	Десятичное число целого типа со знаком
%f	Десятичное число с плавающей точкой
%o	Восьмеричное целое число без знака
%s	**Строка символов**
%u	Десятичное число целого типа без знака
%x или %X	Шестнадцатеричное целое число без знака (буквы нижнего или верхнего регистра)
%p	Вывести на экран значение указателя
%%	Выводит символ %

Пример printf()

- printf ("Hello %c %d %s", ‘a’, 17, “world!");

Результат “Hello a 17 world”

```
// Пример
//  Ввод-вывод строк с использованием функций  printf, scanf
#include <stdio.h>
#include <conio.h>
#include <locale>
int main()
{
        setlocale(LC_ALL, "rus");
        char str[80];
        printf ("Введите строку символов  str:");
        scanf ("%s", str);
        printf ("Вы ввели строку: %s", str);
        while (!kbhit());
          return 0;
}
```

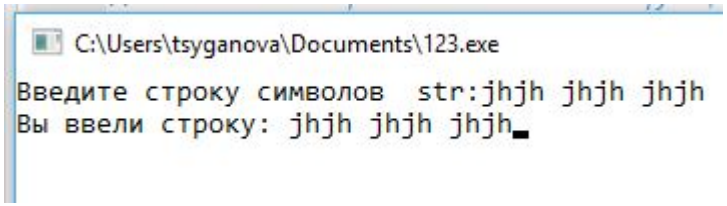


Примечание

- Если необходимо, чтобы функция считала за знак разделителя только конец строки, то рекомендуется использовать следующий формат:

```
scanf ("%[^\n]s", str);
```

```
// Пример
//  Ввод-вывод строк с использованием функций  printf, scanf
#include <stdio.h>
#include <conio.h>
#include <locale>
int main()
{
    setlocale(LC_ALL, "rus");
    char str[80];
    printf ("Введите строку символов  str:");
    scanf ("%[^\n]s", str);
    printf ("Вы ввели строку: %s", str);
    while (!kbhit());
        return 0;
}
```



gets()

- Считывает символы из стандартного потока ввода до символа новой строки \n или до тех пор, пока не будет достигнут конец файла EOF, после чего сохраняет считанные символы в строку типа char.
- Символ новой строки \n не копируется в строку.
- Нулевой символ \0 автоматически добавляется после последнего копируемого символа в string, чтобы сигнализировать о конце строки.

Формат:

gets (string);

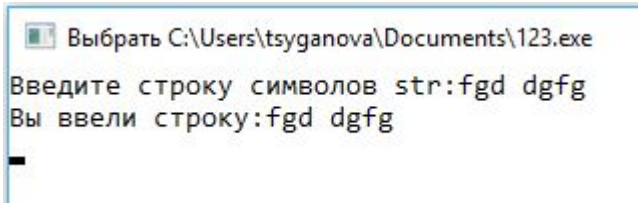
puts()

- Выводит строку типа char*, на которую указывает параметр string в стандартный поток вывод и добавляет символ новой строки ‘\n’.
- Заключительный, нулевой символ не копируется в стандартный поток вывода.

Формат:

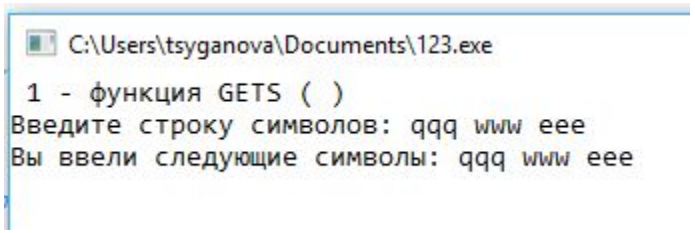
puts(string);

```
// Программа
// Ввод-вывод строк с использованием  функций gets() и puts()
#include <stdio.h>
#include <conio.h>
#include <locale>
int main()
{
        setlocale(LC_ALL, "rus");
        char str[80];
        printf ("Введите строку символов str:");
        gets (str);
        printf ("Вы ввели строку:");
        puts (str);
        while (!kbhit());
         return 0;
}
```

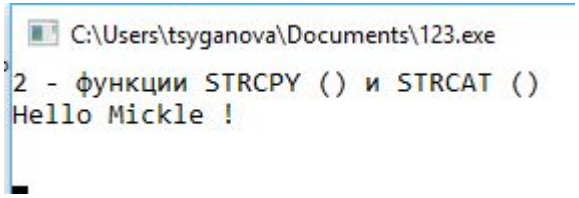


Операции со строками

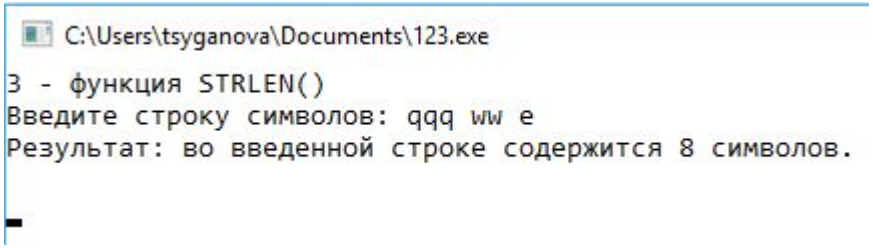
```
/* Программа - Использование библиотечных функций обработки строк */
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
#include <locale>
using namespace std;
int main ( )
{
setlocale(LC_ALL, "rus");
int n;
char str[80], s1[80], s2[80]; char first[20], second[10];
                                    // Функция считывания символов с клавиатуры - GETS()
printf (" 1 - функция GETS ( )\n");
        cout<<"Введите строку символов: ";
        gets(str);
        cout<<"Вы ввели следующие символы: ";
        printf ("%s", str);
printf ("\n\n");
```



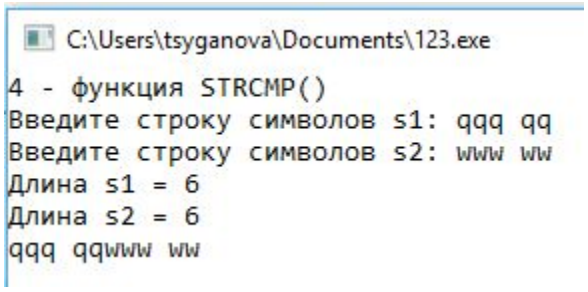
```
// Функция копирования строки - STRCPY()
// и функция объединения (сцепления) 2-х строк - STRCAT()
printf ("2 - функции STRCPY () и STRCAT ()\n");
strcpy (first,"Hello ");
strcpy (second,"Mickle !");
strcat (first,second);
printf ("%s",first);
printf ("\n\n");
```

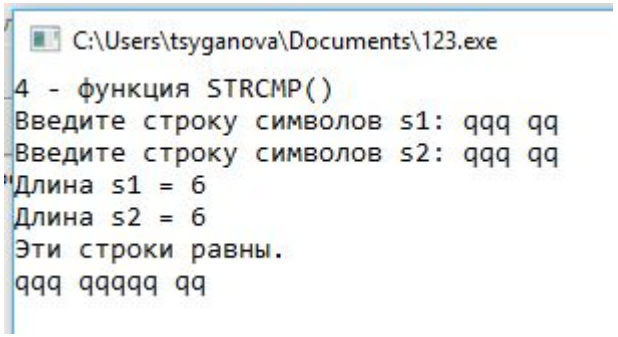


```
// Функция определения длины строки - STRLEN
()
   printf ("3 - функция STRLEN()\n");
   printf ("Введите строку символов: ");
   gets (str);
printf ("Результат: во введенной строке
содержится %d",strlen(str));
printf (" символов.");
printf ("\n\n");
```

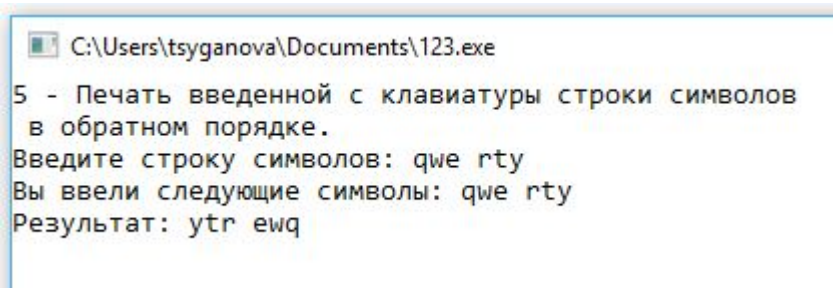


```
// Функция сравнения - STRCMP ()
   cout<<"4 - функция STRCMP()\n";
   cout<<"Введите строку символов s1: ";
   gets(s1);
printf ("Введите строку символов s2: ");
gets (s2);
printf ("Длина s1 = %d\n",strlen(s1));
printf ("Длина s2 = %d\n",strlen(s2));
if (!strcmp(s1,s2)) printf ("Эти строки равны.\n");
strcat (s1,s2);
printf ("%s\n",s1);
printf ("\n\n");
```





```
printf ("5 - Печать введенной с клавиатуры строки символов\n");
printf (" в обратном порядке.\n");
printf ("Введите строку символов: ");
gets (str);
printf ("Вы ввели следующие символы: ");
printf ("%s",str);
cout<<endl;
cout<<"Результат: ";
for (n=strlen(str)-1; n>=0; n--)
      printf ("%c",str[n]);
while (!kbhit ());
}
```



Примеры программ

Пример 10

Задан массив слов. Определить количество символов в словах.

Обозначения:

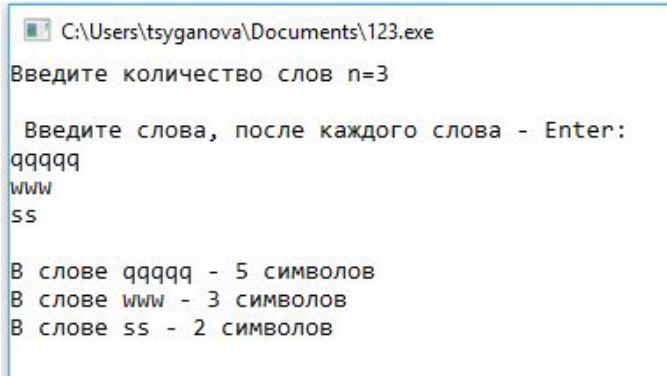
- a[n] – массив слов;
- n – количество слов;
- i – текущий номер слова;
- b[i] – массив, элементы которого - количество символов в словах.

Отладочный пример

n=3; a[n] = {abc; qq; xxxxx};

Результат:
b[n] = {3; 2; 5}

```
#include<iostream>
#include<conio.h>
#include<string.h>
#include <locale>
using namespace std;
int i,n,b[10];
char a[10][10];
int main()
{
	setlocale(LC_ALL, "rus");
	cout << « Введите количество слов n=";
	cin >> n;
	cout << "\n Введите слова, после каждого слова - Enter:\n";
	for (i=0; i<n; i++)
		cin >> a[i];
	for (i=0; i<n; i++)
		b[i]=strlen(a[i]);
	for (i=0; i<n; i++)
		cout << "\nВ слове " << a[i] << " - " << b[i] << " символов";
	cout << "\n\n";
	while (!kbhit());
	return 0;
}
```



Пример 11

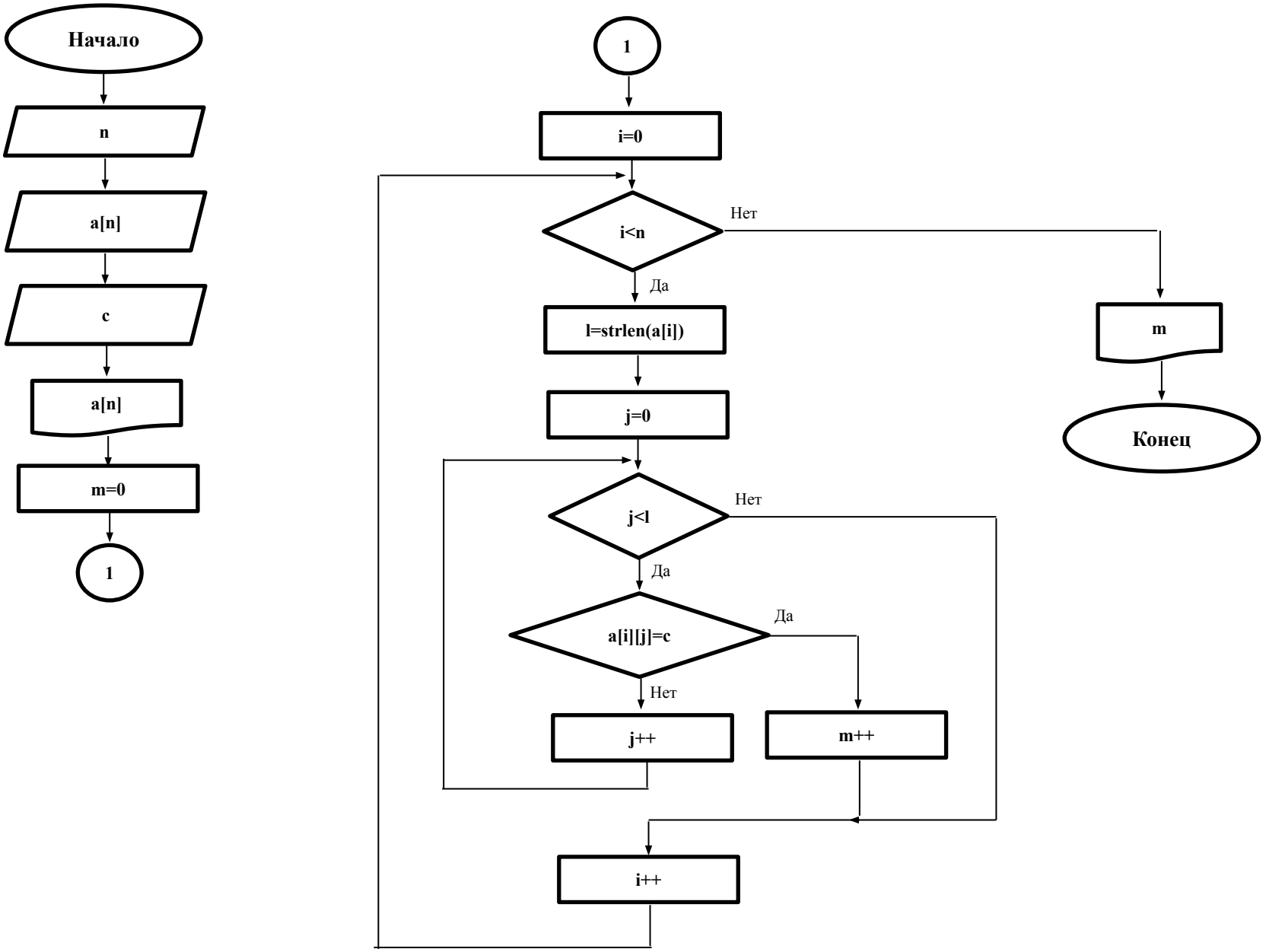
Задан массив слов. Определить количество слов, в которых встречается буква, вводимая с клавиатуры.

Обозначения:
n - количество слов;
a[n] - массив слов;
c - буква, вводимая с клавиатуры;
m - количество слов с буквой “c”;
i - текущий номер слова;
l - длина i-го слова;
j - текущий номер буквы в слове.

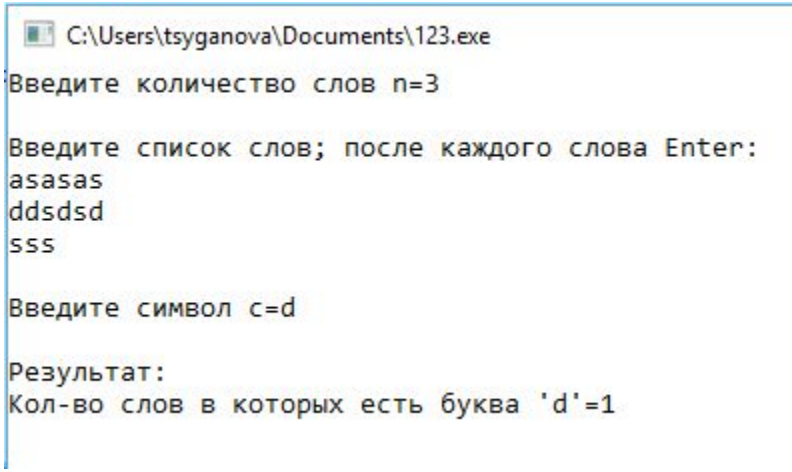
Отладочный пример

n=5;
a[n]= {asd; xqxx; cfx; klm; xxxxx};
c=x.

Результат:
m=3


Начало
n
a[n]
c
a[n]
m=0
1
1
i=0
i<n
Нет
Да
l=strlen(a[i])
j=0
j<l
Нет
Да
a[i][j]=c
Да
Нет
j++
m++
i++
m
Конец

```
#include<iostream>
#include<string.h>
#include <locale>
using namespace std;
int i,j,l,m,n;
char a[30][10],c;
int main()
{
        setlocale(LC_ALL, "rus");
  cout << "Введите количество слов n=";
  cin >> n;
  cout << "\nВведите список слов; после каждого слова Enter:\n";
        for (i=0; i<n; i++)
                cin >> a[i];
        cout << "\nВведите символ c=";
        cin >> c;
        m=0;
        for (i=0; i<n; i++)
        {
                l=strlen(a[i]);
                for (j=0; j<=l; j++)
                        if (a[i][j]==c)
                           {
                                m++;
                                break;              //Прервать цикл
                           }
```



Пример 12

Задан текст, слова разделены запятой, за последним словом точка. Определить количество слов, в которых встречается буква, вводимая с клавиатуры.

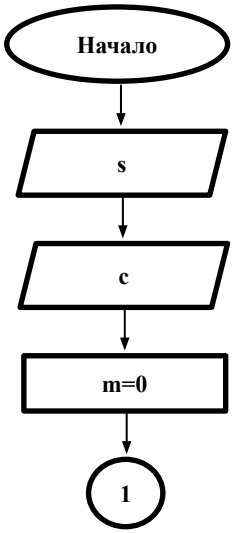
Обозначения

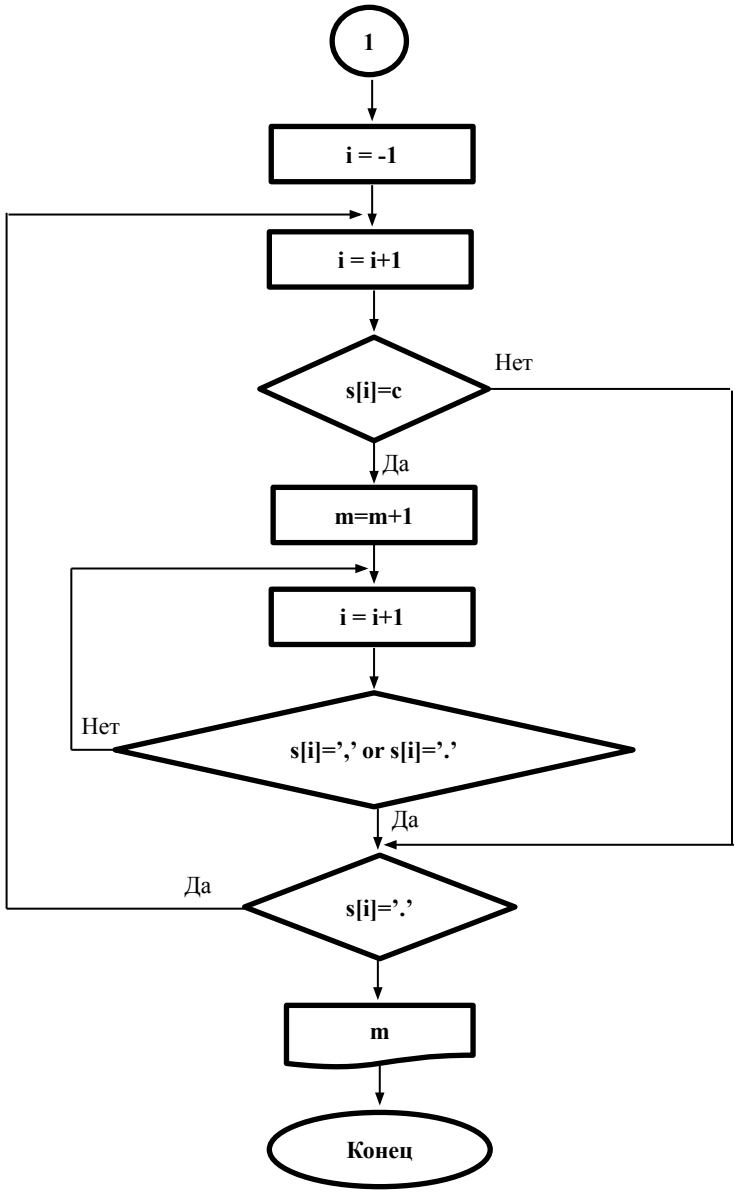
- s[n] – строка символов (заданный текст);
- c – буква, вводимая с клавиатуры;
- m – количество слов, в которых встречается буква, вводимая с клавиатуры ;
- i - текущий номер символа в строке s.

Отладочный пример

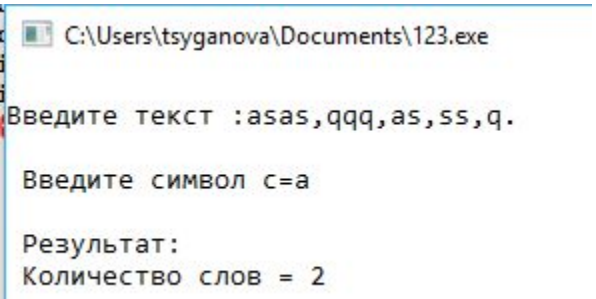
s[n]= {asd,xqxx,cfx,klm,xxxxx.}
c={x}

Результат:
m=3


Начало
s
c
m=0
1


1
i = -1
i = i+1
s[i]=c
Нет
Да
m=m+1
i = i+1
Нет
s[i]=',' or s[i]='.'
Да
Да
s[i]='.'
m
Конец

```
#include <iostream>
#include <string.h>
#include <conio.h>
#include <locale>
using namespace std;
char s[255];
char c;
int i,m;
int main()
{
    setlocale(LC_ALL, "rus");
    cout << "\nВведите текст :";
    cin >> s;
    cout << "\n Введите символ c=";
    cin >> c;
    m=0;
    i=-1;
```



Программа 13

Задан текст, между словами – пробел, за последним словом точка. Напечатать слова текста в обратном порядке.

Обозначения

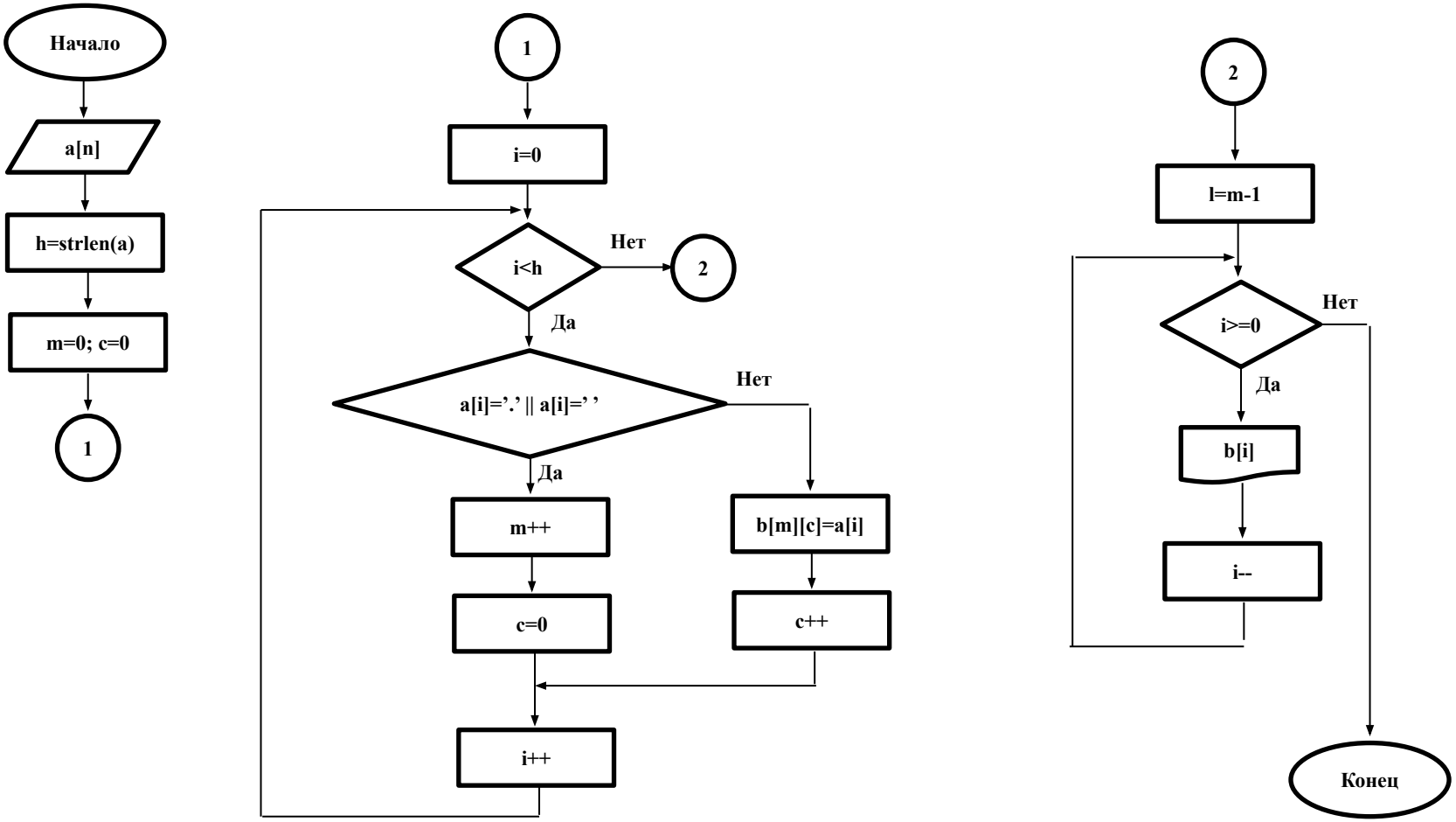
- a[n] – строка символов (заданный текст);
- h – количество символов в заданном тексте;
- b[m][c] – массив слов, который формируется из заданного текста;
- m – количество слов в заданном тексте;
- c – текущий номер символа в b[i] слове;
- i – текущий номер слова в массиве b[m];

Отладочный пример

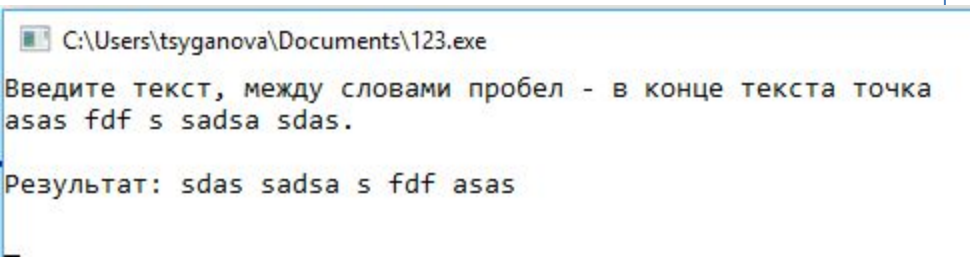
s[n]= {asd xqxx cfx.}

Результат:

b[m]={cfx; xqxx; asd}


Начало
a[n]
h=strlen(a)
m=0; c=0
1
1
i=0
i<h
Нет
2
Да
a[i]='.' || a[i]=' '
Нет
Да
m++
b[m][c]=a[i]
c=0
c++
i++
2
l=m-1
i>=0
Нет
Да
b[i]
i--
Конец

```
#include <iostream>
#include <string.h>
#include <conio.h>
#include <locale>
#include<stdio.h>
using namespace std;
char a[255],b[30][255];
int i,l,k,m,h,c;
int main()
{
	setlocale(LC_ALL, "rus");
	cout << "Введите текст, между словами пробел – в конце текста точка\n";
	gets (a);
	h=strlen(a);
	m=0;
```



Программа 14

Задан текст. Между словами – пробел, в конце – точка. Выполнить сортировку слов в алфавитном порядке.

Обозначения

text [i] - заданный текст (одномерный символьный массив);

word [k][j] – формируемый из текста text [i] массив слов;

r – ячейка обмена;

i – параметр цикла;

j - параметр цикла;

k – количество слов в массиве word [k][j]

Отладочный пример

text[i] – {klm, z, abc, aaaaa.}

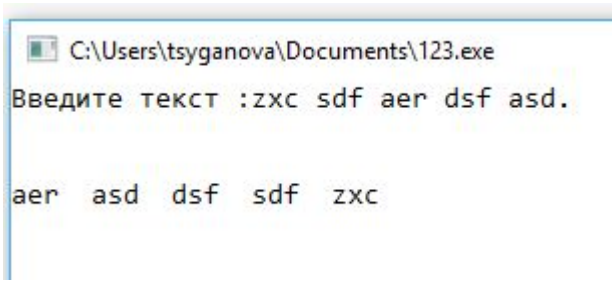
Результат:
word[k] - {aaaaa, abc, klm, z}

Пояснения к алгоритму сортировки

1. Преобразовать исходный текст text[i] в массив слов word[k]
2. Выполнить сортировку слов в массиве word[k] по алфавиту:

```
for (i=0; i<k; i++)
 for (j=i+1; j<=k; j++)
      if (strcmp(&word[i][0],&word[j][0]) > 0)
       {
         strcpy(&r[0],&word[j][0]);
         strcpy(&word[j][0],&word[i][0]);
         strcpy(&word[i][0],&r[0]);
       }
```

```
#include <string.h>
#include <conio.h>
#include <locale>
#include<stdio.h>
using namespace std;
char word[100][15],text[255],r[15];
int  i,j,k;
int main()
{
	setlocale(LC_ALL, "rus");
	printf("Введите текст :");
	scanf("%[^\n]s", &text);
	printf("\n\n");
	i=0;
	j=0;
	k=0;
```



Строки класса string

Строки класса string

- В современном стандарте C++ определен класс с функциями и свойствами (переменными) для организации работы со строками

```
#include <string>
```

- Для работы со строками также нужно подключить стандартный namespace:

```
using namespace std;
```

Основные возможности класса string:

- Инициализация массивом символов (строкой встроенного типа) или другим объектом типа string. Встроенный тип не обладает второй возможностью;
- Копирование одной строки в другую. Встроенный тип - функция strcpy();
- Доступ к отдельным символам строки для чтения и записи. Встроенный тип – используется операция взятия индекса или косвенная адресация с помощью указателя;
- Сравнение двух строк на равенство. Встроенный тип - функции семейства strcmp();

Основные возможности класса string:

- Конкатенация (сцепление) двух строк, дающая результат либо как третью строку, либо вместо одной из исходных. Для встроенного типа применяется функция strcat(), чтобы получить результат в новой строке, необходимо последовательно задействовать функции strcpy() и strcat(), а также выделять память;
- Встроенные средства определения длины строки (функции-члены класса size() и length()). Узнать длину строки встроенного типа можно только вычислением с помощью функции strlen();
- Возможность узнать, пуста ли строка.

Инициализация строк

Задание пустой строки:

```
string st2;
```

Задание инициализированной строки:

```
string st1( “Winter is coming\n” );
```

Определение длины строки

- Применяется к конкретной строке, для которой определяется размер:

st.size();

cout <<“Длина строки ”<<st1<<“-”<<st1.size()<< " символов.";

Проверка строки на пустоту

- Исходная строка:

```
string st2; // пустая строка
```

- Определение длины строки, если 0, значит строка пустая:

```
if  (!st2.size())  cout<<“Строка пустая”;
```

- Использование метода empty(): возвращает true, если строка пустая, и false в противном случае.

```
if (st2.empty()) cout<<“Строка пустая”;
```

Определение совпадения строк

```
string st1 (“Hello!”);
string st2 (“Hello!”);
…
if (st1==st2) cout<<“Строки совпадают”;
```

Копирование строк

Копирование строк осуществляется операцией присваивания:

```
string st1 (“Hello!”);
string st2;
st2=st1;
string st3=st2;
```

Конкатенация строк

Для конкатенации (объединения) строк используется оператор сложения + или оператор сложения с присваиванием +=

```
string st1(“Winter is ”);
string st2(“coming\n”);

string st3 = st1+st2; //получаем новую строку из двух предыдущих
st1+=st2; //добавляем содержимое второй строки в конец первой
```

Конкатенация строк

Допускается объединение между собой не только объектов класса string, но и строк встроенного типа:

```
char ch=“, ”;
string st1(“Hello”);
string st2(“Bro!”);
string st3 = st1+ch+st2+”\n”;
```

Результат: st3 = “Hello, Bro!\n”;

Преобразование

- Объекты встроенного типа возможно преобразовывать в объекты класса string:

```
string s1;
char ch = “Hear me roar\n”;
 s1=ch;
```

Преобразование

Функция c_str() - возвращает указатель на символьный массив, который содержит в себе строку типа string в том виде, в котором она размещалась бы во встроенном строковом типе.

```
string str1;
const char *str2 = str1.c_str();
```

Индексы

К отдельным символам объекта string можно обращаться при помощи индексов:

```
string str1(“Valar Morghulis”);
cout<<str1[0];
```

Метод at(). Обеспечивает проверку границ и создает исключение, если вы пытаетесь получить несуществующий элемент.

```
string str1(“Valar Morghulis”);
cout<<str1.at(0);
```

Пример

Заменить в строке все пробелы на символы подчеркивания.

```
string str (“Valar Morghulis and Valar Dohaeris”);
int size = str.size();
    for (int i=0; i<size; i++)
        if (str[i]==‘ ’) str[i]=‘_’;
```

Пример

Для решения задачи можно воспользоваться функцией replace();

```
#include <algorithm>
…
replace(str.begin(), str.end(), ‘ ’, ‘_’);
```

Указатели

Указатели

- *Указатель* — это переменная, значением которой является адрес памяти, по которому хранится объект определенного типа (другая переменная).

Пример:

если **ch** — это переменная типа *char,* а **p** — указатель на **ch**, значит в **p** находится адрес, по которому в памяти компьютера хранится значение переменной **ch**.

Объявление

тип *<имя переменной>

Пример:

```
int *p;     //по адресу, записанному в переменной p,
          //будет хранится переменная типа int
          //т.е. p указывает на тип данных int
```

Примеры объявления

```
char *p;

int *k, j,*l;

float *pf, f;
```

Операции над указателями

& - “взять адрес”

* - “значение, расположенное по данному адресу”

Операция &

Возвращает адрес своего операнда:

```
float a;          //объявлена вещественная переменная a
float *adr_a;     //объявлен указатель на тип float
adr_a = &a;       //оператор записывает в переменную adr_a
                  //адрес переменной a
```

Операция * - разадресация

Возвращает значение переменной, хранящееся в по заданному адресу, то есть выполняет действие, обратное операции &.

```
float a;          //объявлена вещественная переменная a
float *adr_a;  //объявлен указатель на тип float
a = *adr_a;        //оператор записывает в переменную a
                   //вещественное значение,
                   //хранящиеся по адресу adr_a
```

Примеры

Пусть переменная b размещается по адресу 2000.

```
b_addr = &b
```

Этот оператор присваивания помещает в переменную

b_addr адрес памяти переменной b, т.е. b_addr будет иметь значение 2000.

Примеры

Если b_addr содержит адрес ячейки памяти переменной b, тогда

```
y = *b_addr;
```

поместит значение переменной b в переменную y

Примеры программ

```
//Программа 1
#include <iostream>
using namespace std;
int main()
{
  float x=7.8, y;
  float *k;            //объявляется указатель на переменную типа float
  k=&x;               //в переменную k  помещается адрес переменной x
  y=*k;          //значение переменной x помещается в переменную y
  printf("%f",y);
  return 0;
}
```

```
//Программа 2
#include <stdio.h>
#include <conio.h>
#include <locale>
using namespace std;
int main()
{
  setlocale(LC_ALL, "rus");
        float x=10.0, y;
       float *pf;
       pf=&x;      //переменной pf присваивается адрес переменной x
       y=*pf;      //переменной y присваивается значение переменной x
       printf ("Результаты:\n");
       printf ("x=%f y=%f\n",x,y);
       while (!kbhit());
       return 0;
}
```

```
//Программа 3
#include <stdio.h>
#include <conio.h>
#include <locale>
using namespace std;
int main()
{
	setlocale(LC_ALL, "rus");
	int x=10;
	int *p;
	p=&x;
	printf ("Результаты:\n");
	printf ("%p\n",p);		//печать содержимого p
	printf ("%d,%d\n",x,p);	//печать x и величины по адресу p
	while (!kbhit());
	return 0;
}
```