

# Лекция 9. Строки и структуры в C++

## 9.1. Строки с C++

Строка – это последовательность символов. Если в выражении встречается одиночный символ, он должен быть заключен в **одинарные кавычки**. При использовании в выражениях строка заключается в **двойные кавычки**. Признаком конца строки является нулевой символ '\0'. В C\C++ в отличие от других языков программирования отсутствует тип данных строка, строки в Си можно описать с помощью массива символов (массив элементов типа char), в массиве следует предусмотреть место для хранения признака конца строки ('\0').

Например, описание строки из 25 символов должно выглядеть так:

```
char s[26];
```

Здесь элемент s[25] предназначен для хранения символа конца строки.

```
char s[7]="Привет";
```

Можно описать и массив строк

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

Определен массив из 3 строк по 25 байт в каждой.

Для работы с указателями можно использовать и указатели (char \*). Адрес первого символа и будет начальным значением указателя.

Рассмотрим несколько примеров объявления и ввода строк.

```
#include <iostream>
#include <stdio.h>
#include <malloc.h>
using namespace std;
int main(int argc, char* argv[])
{
    char *s1, s2[20], *s3, s4[30];
    int n;
    scanf("%s", s2);
    printf("\ns2=%s\n", s2);
    s3=s4;
    scanf("%s", s3);
    printf("\ns3=%s\n", s3);
    printf("n=");
    scanf("%d", &n);
    s1=(char *)calloc(n, 1);
    scanf("%s", s1);
    printf("\ns1=%s\n", s1);
    cin>>s1;
    cout<<endl<<"s1="<<s1<<endl;
    return 0;
}
```

Основные функции обработки строк приведены в таблицах 9.1, 9.2, 9.3. и 9.4.

Таблица 9.1. Некоторые функции из библиотеки обработки символов **ctype.h**

<i>Прототип функции</i>	<i>Описание функции</i>
int isdigit(int c)	Возвращает значение <b>true</b> , если <b>c</b> является цифрой, и <b>false</b> в других случаях
int isalpha(int c)	Возвращает значение <b>true</b> , если <b>c</b> является буквой, и <b>false</b> в других случаях
int isalnum(int c)	Возвращает значение <b>true</b> , если <b>c</b> является цифрой или буквой, и <b>false</b> в других случаях
int islower(int c)	Возвращает значение <b>true</b> , если <b>c</b> является буквой нижнего регистра, и <b>false</b> в других случаях
int isupper(int c)	Возвращает значение <b>true</b> , если <b>c</b> является буквой верхнего регистра, и <b>false</b> в других случаях
int tolower(int c)	Если <b>c</b> является буквой верхнего регистра, то результат – буква нижнего регистра, в других случаях возвращается аргумент без изменений
int toupper(int c)	Если <b>c</b> является буквой нижнего регистра, то результат – буква верхнего регистра, в других случаях возвращается аргумент без изменений

Таблица 9.2. Некоторые функции преобразования строк из библиотеки **stdlib.h**

<i>Прототип функции</i>	<i>Описание функции</i>
double atof(const char *s)	Преобразует строку <b>s</b> в тип <b>double</b>
int atoi(const char *s)	Преобразует строку <b>s</b> в тип <b>int</b>
long atol(const char *s)	Преобразует строку <b>s</b> в тип <b>long int</b>

Таблица 9.3. Некоторые функции из библиотеки **string.h**

<i>Прототип функции</i>	<i>Описание функции</i>
size_t strlen(const char *s)	Вычисляет длину строки <b>s</b> в байтах
char *strcat(char *s1, const char *s2)	Присоединяет строку <b>s1</b> в конец строки <b>s2</b>
char *strcpy(char *s1, const char *s2)	Копирует строку <b>s1</b> в место памяти, на которое указывает <b>s2</b>
char *strncat(char *s1, const char *s2, size_t maxlen)	Присоединяет строку <b>maxlen</b> символов строки <b>s2</b> в конец строки <b>s1</b>
char *strncpy(char *s1, const char *s2, size_t maxlen)	Копирует <b>maxlen</b> символов строки <b>s2</b> в место памяти, на которое указывает <b>s1</b>
char * strstr(char *s1, char *s2)	отыскивает позицию первого вхождения строки <b>s2</b> в строку <b>s1</b>
int strcmp(const char *s1, const char *s2)	сравнивает две строки в лексикографическом порядке с учетом различия прописных и

<i>Прототип функции</i>	<i>Описание функции</i>
	строчных букв, возвращает отрицательное число, если s1 располагается в упорядоченном по алфавиту порядке раньше, чем s2, и положительное в противном случае, функция возвращает 0, если строки совпадают.
int strcmp(const char *s1, const char *s2)	сравнивает две строки в лексикографическом порядке не различая прописные и строчные буквы, возвращает отрицательное число, если s1 располагается в упорядоченном по алфавиту порядке раньше, чем s2, и положительное в противном случае, функция возвращает 0, если строки совпадают.
char *strtok(char *s1, char *s2)	Последовательный вызов функции разбивает строку s1 на лексемы, разделенные символами, содержащимися в строке s2. При первом вызове функция получает в качестве аргумента строку s1, в последующих вызовах в качестве аргумента передается NULL. При каждом вызове возвращается указатель на текущую лексему строки s1, когда лексем не осталось, возвращается NULL.

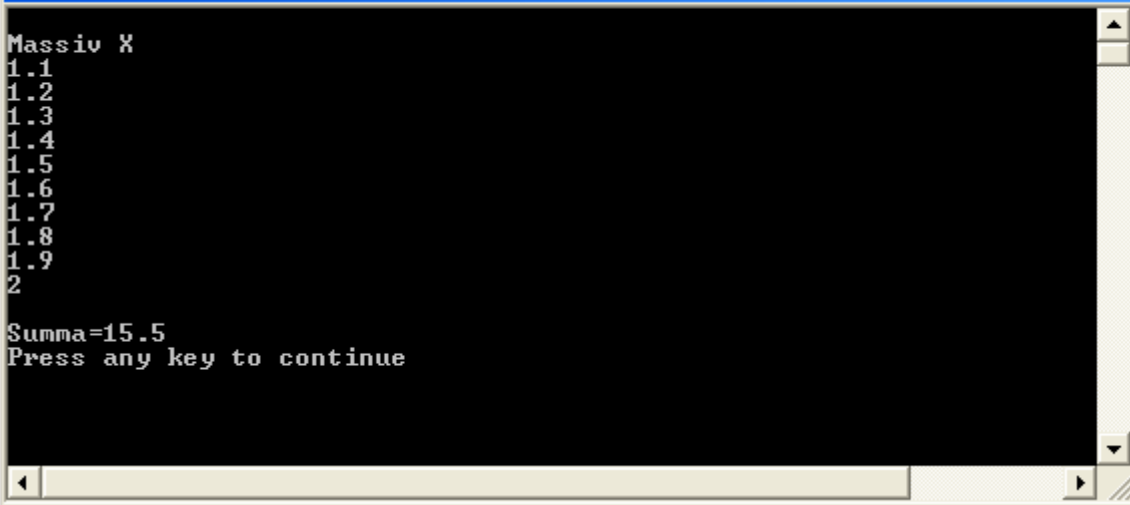
Таблица 9.4. Некоторые функции ввода/вывода из библиотеки **stdio.h**

<i>Прототип функции</i>	<i>Описание функции</i>
int getchar(void)	Вводит следующий символ со стандартного устройства ввода и возвращает его в формате целого
char *gets(char *s)	Вводит символы со стандартного устройства ввода в массив s до тех пор, пока не встретит символ новой строки или индикатор конца файла, после этого добавляется символ NULL
int putchar(int c)	Печать символа, хранящегося в c
int puts(const char *s)	Печать строки s с последующим символом новая строка
int sprintf(char *s,...)	Эквивалент функции printf, но результат вывода запоминается в массиве s, а не отображается на экране
int sscanf(char *s,...)	Эквивалент функции scanf, но ввод осуществляется из массива s, а не с клавиатуры

Далее приведены программы, демонстрирующие работу некоторых функций. В первом примере дана строка, в которой через пробел перечислены

вещественные числа. Из этой строки необходимо получить массив чисел и найти их сумму. Результаты работы программы показаны на рис. 9.1.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main ()
{
    char s1[]="1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0";
    char *S,*mas[100];
    double x[100], Summa=0;
    int i,j=0;
    //преобразование строки s1 в массив строк
    // разделитель - пробел
    S=strtok(s1," ");
    while(S!=NULL)
    {   mas[j]=S;   S=strtok(NULL," ");   j++;   }
    //преобразование массива строк в массив вещественных чисел
    for (i=0;i<j;Summa+=x[i],i++)      x[i]=atof(mas[i]);
    printf("\nMassiv X\n");
    for (i=0;i<j;i++)      printf("%g\n",x[i]);
    printf("\nSumma=%g\n",Summa);
}
```



```
Massiv X
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
Summa=15.5
Press any key to continue
```

Рис.9.1. Результаты работы программы

В следующем примере показано, как две строки объединить в одну, найти длину строки. Результаты работы программы приведены на рис.9.2.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main ()
{
    char s2[100]="Student";
    char s3[100]="Ivanov";
    int length;
    //определение длины строки s2
```

```

length=strlen(s2);
//вывод на экран строки s2
puts(s2);
printf("Dlina stroki=%d\n",length);
//объединение двух строк s2 и s3
strcat(s2," ");
strcat(s2,s3);
puts(s2);}

```



Рис.9.2. Результаты работы программы

## 9.2. Структуры с C++

**Структура** является собранием одного или более объектов (переменных, массивов, указателей, других объектов), которые для удобства работы с ними объединены под одним именем.

Определение структуры состоит из двух шагов:

- объявление структуры (задание нового типа данных определенного пользователем), структура состоит из полей;

```

struct student
{
char fio[30]; // определено поле fio
char group[8]; // определено поле group
int year;
int informatika, math, fizika, history;
}

```

- определение переменных типа структура;
- ```

student Vasya, ES[50];

```

Для обращения к полям структуры надо указать имя переменной и через точку имя поля

### Structura.Pole

Например,  
 Vasya.Year  
 ES[4].math

**ЗАДАЧА 9.1.** Задано  $n$  комплексных чисел, найти число наибольшего модуля.

```

#include <iostream>
#include <math.h>
using namespace std;
int main()
{
struct complex
{

```

```

float x;      //действительная часть
float y;      //мнимая часть
};
//Объявление массива комплексных чисел
complex p[100];
int i,n,nmax;
float max;
cout<<"n=";
cin>>n;
for(i=0;i<n;i++)
{
    cout<<"Vvedite complex chislo\n";
    //ввод действительной части i-го комплексного числа
    cin>>p[i].x;
    //ввод мнимой част i-го комплексного числа
    cin>>p[i].y;
    cout<<p[i].x<<"+"<<p[i].y<<"i"<<endl;
}
max=pow(p[0].x*p[0].x+p[0].y*p[0].y,0.5);nmax=0;
for(i=1;i<n;i++)
if (pow(p[i].x*p[i].x+p[i].y*p[i].y,0.5)>max)
{
max=pow(p[i].x*p[i].x+p[i].y*p[i].y,0.5);nmax=i;
}
cout<<"Nomer maximalnogo modulya "<<nmax<<
"\nEgo velichina "<<max<<endl;
return 0;}

```

### **Результаты**

**n=5**

**Vvedite complex chislo**

**1 3**

**1+3i**

**Vvedite complex chislo**

**5 0.2**

**5+0.2i**

**Vvedite complex chislo**

**-0.2 3**

**-0.2+3i**

**Vvedite complex chislo**

**4.2 0.01**

**4.2+0.01i**

**Vvedite complex chislo**

**-6.2 0.45**

**-6.2+0.45i**

**Nomer maximalnogo modulya 4**

**Ego velichina 6.21631**

**Press any key to continue**

## Динамические структуры

Решим предыдущую задачу с использованием динамических структур.

### Текст 1

```
#include <iostream>
#include <math.h>
#include <malloc.h>
using namespace std;
int main()
{
    struct complex
    {
        float x;
        float y;
    };

    complex *p;
    int i,n,nmax;
    float max;
    cout<<"n=";
    cin>>n;
    //выделяется память под массив p
    p=(complex *)calloc(n,sizeof(complex));
    for(i=0;i<n;i++)
    {
        cout<<"Vvedite complex chislo\n";
        cin>>p[i].x;
        cin>>p[i].y;
        cout<<p[i].x<<"+"<<p[i].y<<"i"<<endl;
    }
    max=pow(p[0].x*p[0].x+p[0].y*p[0].y,0.5);nmax=0;
    for(i=1;i<n;i++)
    if (pow(p[i].x*p[i].x+p[i].y*p[i].y,0.5)>max)
    {
        max=pow(p[i].x*p[i].x+p[i].y*p[i].y,0.5);nmax=i;
    }
    cout<<"Nomer maksimalnogo modulya "<<nmax<<"\nEgo velichina
"<<max<<endl;
    free(p);
    return 0;
}
```

### Текст 2

```
#include <iostream>
#include <math.h>
#include <malloc.h>
```

```

using namespace std;
int main()
{
struct complex
{
    float x;
    float y;
};

complex *p;
int i,n,nmax;
float max;
cout<<"n=";
cin>>n;
p=new complex [n];
for(i=0;i<n;i++)
{
    cout<<"Vvedite complex chislo\n";
//запись (p+i)->x аналогична записи p[i].x
    cin>>(p+i)->x;
    cin>>(p+i)->y;
    cout<<(p+i)->x<<"+"<<(p+i)->y<<"i"<<endl;
}
max=pow(p->x*p->x+p->y*p->y,0.5);
nmax=0;
for(i=1;i<n;i++)
if (pow((p+i)->x*(p+i)->x+(p+i)->y*(p+i)->y,0.5)>max)
{
max=pow((p+i)->x*(p+i)->x+(p+i)->y*(p+i)->y,0.5);nmax=i;
}
cout<<"Nomer maxsimalnogo modulya "<<nmax<<
"\nEgo velichina "<<max<<endl;
delete [] p;
    return 0;
}

```