**Строки в C++**

Строка — последовательность (массив) 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++ строки можно описать с помощью [массива](http://kvodo.ru/urok-7-1-massivyi-obshhie-svedeniya.html) символов (массив элементов типа **char**), в котором следует предусмотреть место для хранения признака конца строки.

Например, описание строки из 25 символов должно выглядеть так:

|  |  |
| --- | --- |
|  | char s[25]; |

Здесь элемент **s[24]** предназначен для хранения символа конца строки.

|  |  |
| --- | --- |
|  | char s[7] = "Привет"; |

Можно описать и массив строк:

|  |  |
| --- | --- |
| 1 | char s[3][25] = {"Пример", "использования", "строк"}; |

Определен массив из 3 строк по 25 байт в каждой.

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

Рассмотрим пример объявления и вывода строк.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | #include "stdafx.h" #include <iostream> using namespace std; int main() { setlocale(LC\_ALL,"Rus"); //описываем 3 строки, s3- указатель char s2[20], \*s3, s4[30]; cout<<"s2="; cin>>s2; //ввод строки s2 cout<<"s2="<<s2<<endl; //запись в s3 адреса строки, где хранится s4. Теперь в переменных //(указателях) s3 и s4 хранится значение одного и того же адреса s3=s4; cout<<"s3="; cin>>s3; //ввод строки s3 //вывод на экран строк s3 и s4, хотя в результате присваивния s3=s4; //теперь s3 и s4 - это одно и тоже cout<<"s3="<<s3<<endl; cout<<"s4="<<s4<<endl; system("pause"); return 0; } |

Результат работы программы:

Но следует отметить, что если пользователь введет в одну переменную слова разделенные пробелом, то программа будет работать иначе:

Все дело в том, что функция **cin** вводит строки до встретившегося пробела. Более универсальной функцией является **getline**.

**cin.getline(char \*s, int n);**

[Функция](http://kvodo.ru/urok-6-1-funktsii-obshhie-svedeniya.html) предназначена для ввода с клавиатуры строки **s** с пробелами, в строке не должно быть более **n** символов. Следовательно, для корректного ввода строк, содержащих пробел, необходимо в нашей программе заменить **cin>>s** на **cin.getline(s, 80)**.

**Операции над строками**

Строку можно обрабатывать как массив символов, используя алгоритмы обработки массивов или с помощью специальных функций обработки строк, некоторые из которых приведены ниже. Для работы с этими строками необходимо подключить библиотеку **cstring**.

Для преобразования числа в строку можно воспользоваться функцией **sprintf** из библиотеки **stdio.h**.

Некоторые функции работы со строками:

|  |  |
| --- | --- |
| **Прототип функции** | **Описание функции** |
| size\_t strlen(const char \*s) | вычисляет длину строки s в байтах. |
| char \*strcat(char \*dest, const char \*scr) | присоединяет строку src в конец строки dest, полученная срока возвращается в качестве результата |
| char \*strcpy(char \*dest, const char \*scr) | копирует строку scr в место памяти, на которое указывает dest |
| char strncat(char \*dest, const char \*dest, size\_t maxlen) | присоединяет строку maxlen символов строки src в конец строки dest |
| char \*strncpy(char \*dest, const char \*scr, size\_t maxlen) | копирует maxlen символов строки src в место памяти, на которое указывает dest |
| int ctrcmp(const char \*s1, const char \*s2) | сравнивает две строки в лексикографическом порядке с учетом различия прописных и строчных букв, функция возвращает 0, если строки совпадают, возвращает — 1, если s1 располагается в упорядоченном по алфавиту порядке раньше, чем s2, и 1 — в противоположном случае. |
| int strncmp(const char \*s1, const char \*s2, size\_t maxlen) | сравнивает maxlen символов двух строк в лексикографическом порядке, функция возвращает 0, если строки совпадают, возвращает — 1, если s1 располагается в упорядоченном по алфавиту порядке раньше, чем s2, и 1 — в противоположном случае. |
| double atof(const char \*s) | преобразует строку в вещественное число, в случае неудачного преобразования возвращается число 0 |
| long atol(const char \*s) | преобразует строку в длинное целое число, в случае неудачного преобразования возвращается 0 |
| char \*strchr(const char \*s, int c); | возвращает указатель на первое вхождение символа **c** в строку, на которую указывает **s**. Если символ **c** не найден, возвращается NULL |
| char \*strupr(char \*s) | преобразует символы строки, на которую указывает s, в символы верхнего регистра, после чего возвращает ее |

**Тип данных string**

Кроме работы со строками, как с массивом символов, в C++ существует специальный тип данных **string**. Для ввода переменных этого типа можно использовать **cin**, или специальную функцию **getline**.

**getline(cin, s);**

Здесь **s** — имя вводимой переменной типа **string**.

При описании переменной этого типа можно сразу присвоить значение этой переменной.

После считывания в буфере остается символ ‘/n’, поэтому для считывания следующей строки его необходимо очистить, например так

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

**string var(s);**

Здесь **var** — имя переменной, **s** — строковая константа. В результате этого оператора создается переменная **var** типа **string**, и в нее записывается значение строковой константы **s**. Например,

**string v(«Hello»);**

Создается строка **v**, в которую записывается значение **Hello**.

Доступ к *i-му* элементу строки **s** типа **string** осуществляется стандартным образом **s[i]**. Над строками типа **string** определенны следующие операции:

* присваивания, например s1=s2;
* объединения строк (s1+=s2 или s1=s1+s2) — добавляет к строке s1 строку s2, результат храниться в строке s1, пример объединения строк:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | #include "stdafx.h" #include <iostream> #include <string> using namespace std; int main() { setlocale(LC\_ALL,"Rus"); string a,b; cout<<"a="; getline(cin, a); cout<<"b="; getline(cin, b); a+=b; cout<<"a="<<a<<endl; system("pause"); return 0; } |

* сравнения строк на основе лексикографического порядка: s1=s2, s1!=s2, s1<s2, s1>s2, s1<=s2, s1>=s2 — результатом будет логическое значение;

При обработке строк типа **string** можно использовать следующие функции:

* **s.substr(pos, length)** — возвращает подстроку из строки **s**, начиная с номера **pos** длинной **length** символов;
* **s.empty()** — возвращает значение true, если строка **s** пуста, false — в противном случае;
* **s.insert(pos, s1)** — вставляет строку **s1** в строку **s**, начиная с позиции **pos**;
* **s.remove(pos, length)** — удаляет из строки **s** подстроку **length** длинной **pos** символов;
* **s.find(s1, pos)** — возвращает номер первого вхождения строки **s1** в строку **s**, поиск начинается с номера **pos**, параметр **pos** может отсутствовать , в этом случае поиск идет с начала строки;
* **s.findfirst(s1, pos)** — возвращает номер первого вхождения любого символа из строки **s1** в строку **s**, поиск начинается с номера **pos**, который может отсутствовать.

**Русский язык для строк**

Думаю вы уже заметили, что при выводе русских букв, в консоли появляются «левые» символы. Для того чтобы избежать этого недоразумения, необходимо воспользоваться сторонней функцией **CharToOemA**. Подключаем библиотеку **windows.h**, она нужна для того, чтобы наша функция могла преобразовать строки в другую кодировку. Также, нам понадобиться дополнительный символьный массив. Исходный код программы будет выглядеть вот так:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | #include "stdafx.h" #include <iostream> #include <windows.h> using namespace std; int main() {setlocale(LC\_ALL,"Rus"); char s[255]={" Меня надо преобразовать "}; char \*pre=new char[255]; CharToOemA(s, pre); //преобразовываем cout<<s; delete []pre; system("pause>>void"); return 0; } |

Способ только что описанный достаточно не удобен. Но существует более простой вариант решения «русской» проблемы. Как видите, в программе используется функция setlocale(), вместо этого удобнее вписать в главную функцию следующую конструкцию:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 | #include "stdafx.h" #include <iostream> #include <windows.h> using namespace std; int main() { SetConsoleCP(1251); SetConsoleOutputCP(1251); system("pause>>void"); return 0; } |

Теперь у нас появляется библиотека **windows.h**, а также две новые функции: **SetConsoleCP()** и **SetConsoleOutputCP()**.

get