

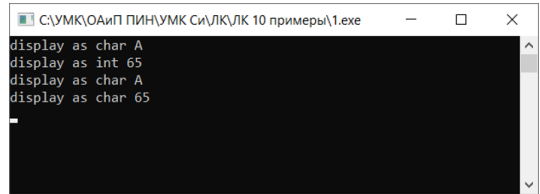
Тип char

Для хранения строк используются массивы типа **char**.

Тип char – **числовой**, он хранит один байт данных. Но в соответствии с таблицей кодировки каждое из этих чисел связано с символом. И в обратную сторону – каждый символ определяется своим порядковым номером в таблице кодировки.

Тип char

```
#include <stdio.h>
int main() {
    char c = 'A';
    int i = 65;
    printf("display as char %c\n", c);
    printf("display as int %d\n", c);
    printf("display as char %c\n", i);
    printf("display as char %d\n", i);
    getchar();
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\УМК\ОАиП ПИН\УМК Си\ЛК\ЛК 10 примеры\1.exe". The window contains the following text output from the program:

```
display as char A
display as int 65
display as char A
display as char 65
```

Спецсимволы тоже имеют код

```
#include <stdio.h>
```

```
int main() {
    printf("\\n - %d\\n", '\\n');
    printf("\\t - %d\\n", '\\t');
    printf("\\v - %d\\n", '\\v');
    printf("\\b - %d\\n", '\\b');
    printf("\\r - %d\\n", '\\r');
    getchar();
}
```

A screenshot of a Windows Command Prompt window. The title bar at the top shows the file path "D:\-хэшэ\... 10 яЁшыхЁ\1.exe". The command prompt displays several environment variables: \n - 10, \t - 9, \v - 11, \b - 8, and \r - 13. The background is black, and the text is white. A vertical scrollbar is visible on the right side of the window.

Строка в Си

Так как язык Си по своему происхождению является языком системного программирования, то **строковый тип данных** в Си как таковой **отсутствует**, а в качестве строк в Си используются **обычные массивы символов**.

Строка в Си – это массив типа **char**, последний элемент которого хранит терминальный символ **'\0'**. Числовое значение этого символа 0, поэтому можно говорить, что массив оканчивается нулём.

Исторически сложилось два представления формата строк:

- формат ANSI;
- строки с завершающим нулем (используется в Си).

Строка как массив

Для вывода использовался ключ `%s`. При этом строка выводится до первого терминального символа, потому что функция **`printf`** не знает размер массива `word`.

```
#include <conio.h>
#include <stdio.h>
void main() {
    char word[10];
    word[0] = 'A';
    word[1] = 'B';
    word[2] = 'C';
    word[3] = '\0';
    //word[3] = 0; эквивалентно
    printf("%s", word);
    getch();
}
```

Способы задания строк в Си

В программе строки могут определяться следующим образом:

- ❶ как строковые константы;
- ❷ как массивы символов;
- ❸ через указатель на символьный тип.

Кроме того, должно быть предусмотрено выделение памяти для хранения строки.

Строка как строковая константа

Любая последовательность символов, заключенная в двойные кавычки «», рассматривается как **строковая константа**.

Для корректного вывода любая строка должна заканчиваться нуль-символом '\0', целочисленное значение которого равно 0. При объявлении строковой константы нуль-символ добавляется к ней автоматически.

Строковые константы часто используются для осуществления диалога с пользователем в таких функциях, как printf().



Строка как массив символов

При определении **массива** символов необходимо сообщить компилятору требуемый размер памяти.

```
char m[82];
```

Компилятор также может самостоятельно определить размер массива символов, если инициализация массива задана при объявлении строковой константой:

```
char m2[]="Горные вершины спят во тьме ночной.";
char m3[]={ 'Т', 'и', 'х', 'и', 'е', ' ', 'д', 'о', 'л', 'и', 'н', 'ы',
            ' ', 'п', 'о', 'л', 'н', 'ы',
            ' ', 'с', 'в', 'е', 'ж', 'е', 'й', ' ', 'м', 'г', 'л', 'о', 'й', '\\0' };
```


Строка как указатель на символьный тип

Для задания строки можно использовать **указатель на символьный тип**

```
char *m4;
```

В этом случае объявление массива переменной m4 может быть присвоен адрес массива:

```
m4 = m3;
```

```
*m4;          // эквивалентно m3[0]='Т'
```

```
*(m4+1);      // эквивалентно m3[1]='и'
```

Работа со строками в С

Так как строки на языке Си являются **массивами символов**, то к любому символу строки можно обратиться по его индексу. Для этого используется синтаксис обращения к элементу массива, поэтому первый символ в строке имеет индекс ноль.

```
for(int i = 0; str[i] != 0; i++)  
{  
    if (str[i] == 'a')  
        str[i] = 'A';  
    else if (str[i] == 'A')  
        str[i] = 'a';  
}
```

Функции ввода строк

Для ввода строки может использоваться функция `scanf()`.

Однако функция `scanf()` предназначена скорее **для получения слова, а не строки**.

Если применять формат `"%s"` для ввода, строка вводится до (но не включая) следующего пустого символа, которым может быть пробел, табуляция или перевод строки.

```
char str[31] = "";  
printf("Введите строку: ");  
scanf("%30s", &str);  
printf("Вы ввели: %s", str);
```

Поэтому, используя данную функцию **невозможно ввести строку, содержащую несколько слов**, разделенных пробелами или табуляциями. Например, если в предыдущей программе пользователь введет строку: "Сообщение из нескольких слов" то на экране будет выведено только "Сообщение".

Функции ввода строк

Напишем простую программу, которая запрашивает у пользователя строку и возвращает её длину.

```
#include <conio.h>
#include <stdio.h>
void main() {
    char buffer[128];
    unsigned len = 0;
    scanf("%127s", buffer);
    while (buffer[len] != '\0') {
        len++;
    }
    printf("length(%s) == %d", buffer, len);
    getch();
}
```

Функции ввода строк

Напишем простую программу, которая запрашивает у пользователя строку и возвращает её длину.

```
#include <conio.h>
#include <stdio.h>
void main() {
    char buffer[128];
    unsigned len = 0;
    scanf("%127s", buffer);
    while (buffer[len] != 0) { // числовое значение символа '\0' - 0
        len++;
    }
    printf("length(%s) == %d", buffer, len);
    getch();
}
```

Функции ввода строк

Напишем простую программу, которая запрашивает у пользователя строку и возвращает её длину.

```
#include <conio.h>
#include <stdio.h>
void main() {
    char buffer[128];
    unsigned len = 0;
    scanf("%127s", buffer);
    while (buffer[len]) {// самая короткая запись
        len++;
    }
    printf("length(%s) == %d", buffer, len);
    getch();
}
```

Программа сравнения введенных слов

```
#include <conio.h>
#include <stdio.h>
void main() {
    char firstWord[128];    //Первое слово
    char secondWord[128];   //Второе слово
    unsigned i;             //Счётчик
    int cmpResult = 0;      //Результат сравнения

    scanf("%127s", firstWord);
    scanf("%127s", secondWord);

    for (i = 0; i < 128; i++) {
        if (firstWord[i] > secondWord[i]) {
            cmpResult = 1;
            break;
        } else if (firstWord[i] < secondWord[i]) {
            cmpResult = -1;
            break;
        }
    }

    printf("%d", cmpResult);
    getch();
}
```

Так как каждая буква имеет числовое значение, то их можно сравнивать между собой как числа. Кроме того, обычно (но не всегда!) буквы в таблицах кодировок расположены по алфавиту. Поэтому сортировка по числовому значению также будет и сортировкой по алфавиту.

Функции ввода строк

Для ввода и вывода строк в библиотеке **stdio.h** содержатся специализированные функции **gets** и **puts**.

Функция **gets** предназначена для ввода строк и имеет следующий заголовок:

```
char * gets(char *buffer);
```

Между тем использовать функцию **gets** категорически не рекомендуется, ввиду того, что она не контролирует выход за границу строки, что может произвести к ошибкам. Вместо нее используется функция **fgets** с тремя параметрами:

```
char * fgets(char * buffer, int size, FILE * stream);
```


Функции ввода строк

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    char myString[100];
```

```
    printf( "Enter a long string: " );
```

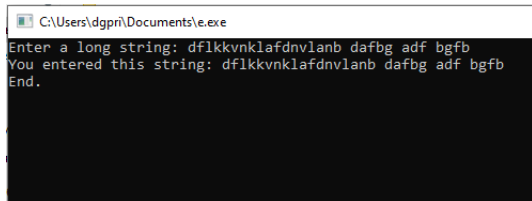
```
    fgets( myString, 100, stdin );
```

```
    printf( "You entered this string: %s", myString );
```

```
    printf("End.");
```

```
    getchar();
```

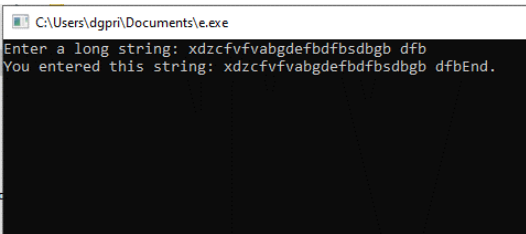
```
}
```



```
C:\Users\dgpri\Documents\l.exe
Enter a long string: dfllkvnklafdnvlnb dafbg adf bgfb
You entered this string: dfllkvnklafdnvlnb dafbg adf bgfb
End.
```

Ручное удаление '\n'

```
#include <stdio.h>
int main() {
    char myString[100];
    printf( "Enter a long string: " );
    fgets( myString, 100, stdin );
    for (int i = 0; i < 100; i++ ) {
        if ( myString[i] == '\n' ) {
            myString[i] = '\0';
            break;
        }
    }
    printf("You entered this string: %s", myString );
    printf("End.");  getchar();
}
```



```
C:\Users\dgpr\Documents\e.exe
Enter a long string: xdzcfvfva b g d e f b d f b s d b g b d f b
You entered this string: xdzcfvfva b g d e f b d f b s d b g b d f b End.
```

Функция ввода символов

Для ввода символов может использоваться функция

```
char getchar();
```

которая возвращает значение символа, введенного с клавиатуры. Указанная функция использовалась ранее для задержки окна консоли после выполнения программы до нажатия клавиши.

Функция вывода символов

Для вывода символов может использоваться функция


```
char putchar(char);
```

которая возвращает значение выводимого символа и выводит на экран символ, переданный в качестве аргумента.

Пример

```
#include <stdio.h>
int main() {
    char s[80], sym;
    int count, i;
    printf("Enter a string : ");
    gets(s);
    printf("Enter a char : ");
    sym = getchar();
    count = 0;
    for (i = 0; s[i] != '\0'; i++)
    {
        if (s[i] == sym)
            count++;
    }
```

```
    printf("In the string\n");
    puts(s);
    printf("the symbol ");
    putchar(sym);
    printf(" entered %d times", count);
    getchar(); getchar();
    return 0;
}
```

 C:\Users\dgpri\Documents\e.exe

```
Enter a string : Hello, world!
Enter a char : o
In the string
Hello, world!

the symbol o entered 2 times
```

Преобразование строк

В С для преобразования строк, содержащих числа, в численные значения в библиотеке **stdlib.h** предусмотрен следующий набор функций:

```
// преобразование в число типа double  
double atof(const char *string);
```

```
// преобразование в число типа int long  
int atoi(const char *string);
```

```
// преобразование в число типа long int  
int atol(const char *string);
```

```
// преобразование в число типа long long int  
long long int atoll(const char *string);
```

Пример

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    char s[80], sym;
    int i;
    printf("Enter a string : ");
    fgets(s,80,stdin);

    i = atoi(s);

    printf("%d", count);
    getchar(); getchar();
    return 0;
}
```