**Символы и строки**

**Символы**

Для работы с символами используется тип **char** – целочисленный тип, т.е. любой символ представляется каким – либо числом. Следовательно, помещая символ в переменную типа **char,** мы фактически помещаем туда соответствующее число.

* Если переменная типа **char** используется в программе там, где ожидается символьная информация, то её значение интерпретируется как символ.
* Если переменная типа **char** используется в программе там, где ожидается числовая информация, то её значение интерпретируется как число.

Можно осуществлять вывод значения символьной переменной как в числовом формате, так и в символьном.

Char c1='a';

printf("\ a character is %c", c1);

printf("\ a number is %d", c1);

в первом случае переменная c1 будет выведена как символ, во втором случае как целое число. Максимальное значение переменной типа **char** = 127 т.к.

0 – 127 – основной набор (буквы латинского алфавита, знаки, цифры).

128 – 255 – расширенный набор (буквы других алфавитов, символы псевдографики и т.д.).

Для работы с символами из основного набора используют тип **char**, для расширенного набора – **unsigned char.**

**Строки**

Специального типа строка в языке С нет, но строки можно хранить в массивах символов. Объявление осуществляется следующим образом: **char string[10]**

В данном массиве можно хранить 9 и меньшее количество элементов, т.к. в С строка определяется как последовательность символов, заканчивающаяся нулевым символом. Нулевой символ-это специальный символ, который обозначается **\0.** Компилятор воспринимает его как один символ. Например для хранения строки ALABAMA используются символы A, L, A, B, A, M, A, \0. Массив можно инициализировать при его описании:

**char string[10]={'A','L','A','B','A','M','A','\0'}**

но удобнее использовать строковый литерал, т.е. последовательность символов в двойных кавычках.

**char string[10]="ALABAMA";**

**char string[]="ALABAMA";**

Каждая строка должна заканчиваться знаком **\0**. Функции для работы со строками определяют длины строк по их завершающим символам. Функции EOLN не существует.

**Строки и указатели**

Т.к. конец строки четко обозначен, то для однозначного определения строки нужно лишь задать указатель на её начало. Имя массива представляет собой указатель типа **char** на начало строки. Массив нужен для выделения памяти для хранения строки. Выделение памяти при компиляции с использование указателя выглядит следующим образом:

**Char \*p = "Good day";**

Эта запись делает то же что и следующая:

**Char p[] = "Good day";**

**Ввод и вывод строк**

Для вывода строк используются функции **puts(), printf().**

**Puts()** – выводит строку текста на экран. После выведенной строки автоматически осуществляется переход на новую строку. Эту функцию нельзя использовать для вывода значений переменных.

**printf()** – используется как для вывода строк текста. Так и для вывода значений переменных.

# include <stdio.h>

char \*m1="We a the";

char \*m2="best";

char \*str="a mess to display";

int main()

{

printf("%s", str);

puts(m1);

puts(m2);

return 0;

}

Для ввода строк используются функции **gets(), scanf().**

**gets()** считывает все символы, набранные с клавиатуры, вплоть до первого символа конца строки (**enter**).

# include <stdio.h>

char input[257];

int main()

{

puts("Enter some text");

gets(input);

printf("you entered: %s \n", input);

return 0;

}

При вводе с помощью функции **scanf()** используется спецификация **%s** или **%ns**, где **n** – ширина ввода (т.е. количество считываемых символов). С помощью **scanf()** можно вводить несколько строк.

**scanf("%s %s %4s", s1, s2, s3);**

если ввести меньше строк, то функция будет ожидать ввода недостающих символов. Если ввести больше строк, чем требуется, то лишние строки останутся в буфере клавиатуры и будут считаны при последующих вызовах функции **scanf().**

**Функции для работы со строками**

Для работы со строками имеется большое число стандартных функций, прототипы которых находятся в заголовочном файле **string.h**. Указанные функции автоматически обрабатывают признак конца троки и обеспечивают его правильную установку. Если же программист определяет свои функции для работы со строками, то он сам должен заботится о правильности обработки указанного символа.

1. **unsigned strlen(char \*s)** – возвращает длину строки s в виде беззнакового целого числа.
2. **char \*strcat(char \*s1, const char \*s2)** – сцепление двух строк, где строка s2 помещается после строки s1.
3. **char \*strncat(char \*s1, const char \*s2, size\_t n)** – сцепление двух строк, где к строке s1 добавляется n символов строки s2.
4. **char \*strcpy(char \*s1, const char \*s2)** – копирует строку s2 в строку s1. В результате строка содержащаяся в s1 затирается, указатель возвращается на строку s1.
5. **char \*strncpy(char \*s1, const char \*s2, size\_t n)** – копирует n символов из строки s2 в строку s1.
6. **char \*strdup(char \*s)** – аналогична strcpy, выделение памяти под новую строку происходит автоматически.
7. **char \*strnset(char \*s, char c, size\_t n)** – устанавливает в первые n символов строки s значение c.
8. **char \*strlwr(char \*s)** – преобразовывает все буквы строки с верхнего регистра в нижний.
9. **char \*strup**r**(char \*s)** – преобразовывает все буквы строки с нижнего регистра в верхний.
10. **char \*strrev(char \*s) -** переставляет символы строки s в обратном порядке.

Функции поиска в строках

1. **char \*strchr(char \*s, char c) –** ищет первое вхождение символа c в строке s. В качестве результата возвращается указатель на найденное значение или 0, если вхождений нет.
2. **int strcsph(char \*s1, char \*s2) –** ищет в строке s1 первый сегмент не содержащий не одного символа из строки s2. В качестве результата возвращает длину найденного сегмента.
3. **char \*strpbrk(char \*s1, char \*s2) -** ищет в строке s1 первое вхождение любого из символов строки s2.
4. **char \*strrshk(char \*s, char c) –** возвращает последнее вхождение символа c в строку s, если вхождений нет то выдает значение 0.
5. **char \*strstr(char \*s1, char \*s2) –** возвращает указатель на первую позицию, с которой начинается строка s2 внутри строки s1.

Функции сравнения строк

1. **int strcmp(char \*s1, char \*s2) –** сравнивает строки s1 и s2 в соответствии с лексикографическим порядком. Если s1=s2 возвращает значение 0, если s1<s2 – отрицательное значение, если s1>s2 положительное значение.
2. **int stricmp(char \*s1, char \*s2) –** сравнивает строки s1 и s2 без учета регистра.
3. **int strncmp(char \*s1, char \*s2, unsigned n) –** сравнивает первые n символов строк s1 и s2.
4. **int strnicmp(char \*s1, char \*s2, unsigned n) –** сравнивает первые n символов строк s1 и s2 без учета регистра.
5. **char \*strerror(int n) –** возвращае указатель на строку сообщения для заданного кода ошибки n.

Функции преобразования строк в числа

1. **int atoi(char \*p)** – преобразование строки в значение типа int.
2. **long atol(char \*p)** – преобразование строки в значение типа long.
3. **long long atoll(char \*p)** – преобразование строки в значение типа long long.
4. **double atof(char \*p)** – преобразование строки в значение типа double.

**Примеры программ работающих со строками**

Сцепление строк (The strcat() function)

#include <stdio.h>

#include <string.h>

char str1[27] = "a";

char str2[2];

int main( void )

{

int n;

/\* Put a null character at the end of str2[]. \*/

str2[1] = '\0';

for (n = 98; n< 123; n++)

{

str2[0] = n;

strcat(str1, str2);

puts(str1);

}

return 0;

}

Определение длины введенной строки (Using the strlen() function)

#include <stdio.h>

#include <string.h>

int main( void )

{

size\_t length;

char buf[80];

while (1)

{

puts("\nEnter a line of text, a blank line to exit.");

gets(buf);

length = strlen(buf);

if (length != 0)

printf("\nThat line is %u characters long.", length);

else

break;

}

system("PAUSE");

return 0;

}

Сравнение n символов в 2-х строках(The strncmp() function.)

#include <stdio.h>

#include <string.h>

char str1[] = "The first string.";

char str2[] = "The second string.";

int main( void )

{

size\_t n, x;

puts(str1);

puts(str2);

while (1)

{

puts("\n\nEnter number of characters to compare, 0 to exit.");

scanf("%d", &n);

if (n <= 0)

break;

x = strncmp(str1, str2, n);

printf("\nComparing %d characters, strncmp() returns %d.", n, x);

}

return 0;

}

Копирование n символов из одной строки в другую(Using the strncpy() function)

#include <stdio.h>

#include <string.h>

char dest[] = "..........................";

char source[] = "abcdefghijklmnopqrstuvwxyz";

int main( void )

{

size\_t n;

while (1)

{

puts("Enter the number of characters to copy (1-26)");

scanf("%d", &n);

if (n > 0 && n< 27)

break;

}

printf("\nBefore strncpy destination = %s", dest);

strncpy(dest, source, n);

printf("\nAfter strncpy destination = %s\n", dest);

system("PAUSE");

return 0;

}

Поиск строки в подстроке (Searching with strspn())

#include <stdio.h>

#include <string.h>

int main( void )

{

char buf1[80], buf2[80];

size\_t loc;

/\* Input the strings. \*/

printf("Enter the string to be searched: ");

gets(buf1);

printf("Enter the string containing target characters: ");

gets(buf2);

/\* Perform the search. \*/

loc = strspn(buf1, buf2);

if ( loc == 0 )

printf("No match was found.\n");

else

printf("Characters match up to position %d.\n", loc-1);

system("PAUSE");

return 0;

}

Поиск строки внутри другой строки (Searching with strstr())

#include <stdio.h>

#include <string.h>

int main( void )

{

char \*loc, buf1[80], buf2[80];

/\* Input the strings. \*/

printf("Enter the string to be searched: ");

gets(buf1);

printf("Enter the target string: ");

gets(buf2);

/\* Perform the search. \*/

loc = strstr(buf1, buf2);

if ( loc == NULL )

printf("No match was found.\n");

else

printf("%s was found at position %d.\n", buf2, loc-buf1);

system("PAUSE");

return 0;

}