Лабораторная работа № 5

Работа со строками и текстами

Основные сведения

Строка в языке С – это одномерный массив символов, заканчивающийся нуль-символом (‘**0**’). Это единственный вид строки, определенный в языке С. Прототипы функций, константы и типы для работы с нуль-терминированными строками описаны в заголовочном файле <string.h>. Для работы со строками полезными могут оказаться и другие функции, описанные в заголовочных файлах <ctype.h>, <stdlib.h> и <stdio.h>. Список функций, рекомендуемых к использованию в лабораторной работе, приведен в Приложении А. В Приложении Б приводятся некоторые примеры применения функций С для работы со строками.

Задание

Составить программу в соответствии с предложенным вариантом из папки «Варианты».

Приложение А

Функции С для работы с символами и строками

А.1 Основные функции библиотеки С string.h

void * memcpy(void * *dst***, const void *** *src***, size_t** *count***);**
Копирует *count* байтов из области памяти, на которую указывает *src*, в область памяти, на которую указывает *dst*. Функция возвращает адрес назначения *dst*. Области памяти не должны перекрываться, иначе данные могут быть скопированы неправильно.

void * memmove(void * *dst***, const void *** *src***, size_t** *count***);**
Копирует *count* байтов из области памяти, на которую указывает *src*, в область памяти, на которую указывает *dst*. Функция возвращает адрес назначения *dst*. Области памяти источника и приемника могут перекрываться.

int memcmp(const void * *buf1***, const void *** *buf2***, size_t** *count***);**
Сравнивает первые *count* символов из *buf1* и *buf2* и возвращает значение, показывающее связь между ними. Знак ненулевого возвращаемого значения равен знаку разности между первой различающейся парой значений из буферов. Значения интерпретируются как **unsigned char**.

void * memchr(const void * *buf***, int** *c***, size_t** *count***);**
Ищет первое вхождение символа, код которого указывается аргументом *c*, в первых *count* байтах *buf*. Поиск останавливается при обнаружении *c* или после проверки первых *count* байт.

void * memset(void * *dst***, int** *c***, size_t** *count***);**
Заполняет первые *count* байтов области памяти, на которую указывает аргумент *dst*, символом, код которого указывается аргументом *c*.

size_t strlen(const char * *str***);**
Возвращает длину строки *str*.

char *strrev(char * *str***);**
Возвращает указатель на перевернутую строку *str***.**

char *strcpy(char * *strDestination***, const char *** *strSource***);**
Копирует символы из строки *strSource* в строку *strDestination* и возвращает указатель на строку strDestination.

char *strncpy(char ***strDestination*, const char ***strSource***, size_t** *count***);**
Копирует *count* символов из строки *strSource* в строку *strDestination* и возвращает указатель на строку *strDestination*.

char *strcat(char ***strDestination*, const char ***strSource***);**
Дописывает строку *strSource* в конец строки *strDestination*.

char *strncat(char ***strDestination*, const char ***strSource***, size_t** *count***);**
Дописывает не более *count* начальных символов строки *strSource* (или всю строку *strSource*, если ее длина меньше) в конец строки *strDestination*.

char *strchr(const char ***str*, int** *c***);**
Ищет символ *c* в строке *str*, начиная с головы, и возвращает его адрес, или **NULL,** если символ *c* не найден.

char *strrchr(const char ***str*, int** *c***);**
Ищет символ *c* в строке *str*, начиная с хвоста, и возвращает его адрес, или **NULL,** если символ *c* не найден.

int strcmp(const char ***string1*, const char ***string2***);**
Лексикографически сравнивает строки *string1* и *string2* с учетом регистра символов.

int strncmp(const char ***string1*, const char ***string2***, size_t** *count***);**
Лексикографически сравнивает первые *count* байтов строк *string1* и *string2* с учетом регистра символов.

int strcoll(const char ***string1*, const char ***string2***);**
Лексикографически сравнивает строки *string1* и *string2* с учетом параметров локализации.

char *strerror(int *errnum***);**
Возвращает строковое представление сообщения об ошибке *errnum*.

size_t strspn(const char ***str*, const char ***strCharSet***);**
Определяет максимальную длину начальной подстроки *str*, состоящей исключительно из байтов, перечисленных в *strCharSet*.

size_t strcspn(const char ***str*, const char ***strCharSet***);**
Определяет максимальную длину начальной подстроки строки *str*, состоящей исключительно из байтов, не перечисленных в *strCharSet*.

char *strpbrk(const char ***str*, const char ***strCharSet***);**
Находит первое вхождение любого символа, перечисленного в *strCharSet*, в строке *str*.

char *strstr(const char ***str*, const char ***strSearch***);**
Находит первое вхождение строки *strSearch* в строке *str*.

char *strtok(char ***strToken*, const char ***strDelimit***);**
Возвращает указатель на очередной токен, найденный в строке *strToken*. Множество символов строки *strDelimit* определяет возможные разделители между токенами. Если токенов в строке *strToken* больше нет, то возвращает **NULL.**

A.2 Основные функции библиотеки C ctype.h

```
int     isalnum(int);
int     isalpha(int);
int     iscntrl(int);
int     isdigit(int);
int     isgraph(int);
int     islower(int);
```

```
int     isprint(int);
int     ispunct(int);
int     isspace(int);
int     isupper(int);
int     isxdigit(int);
int     tolower(int);
int     toupper(int);
```

A.3 Функции библиотеки C stdlib.h для преобразования данных

```
double atof(const char *);
int atoi(const char *nptr);
long int atol(const char *nptr);
long long int atoll(const char *nptr);
double strtod(const char *, char **);
float strtof(const char *, char **);
long double strtold(const char *, char **);
long strtol(const char *, char **, int);
unsigned long int strtoul(const char *, char **, int);
long long int strtoll(const char *, char **, int);
unsigned long long int strtoull(const char *, char **, int);
```

A.4 Функции библиотеки C stdio.h для операций с символьными и строковыми данными

```
int     fgetc(FILE *);
int     fputc(int, FILE *);
int     getc(FILE *);
int     putc(int, FILE *);
char *  fgets(char *, int, FILE *);
int     fputs(const char *, FILE *);
char *  gets(char *);
int     puts(const char *);
int     fputchar(int);
int     getchar(void);
int     putchar(int);
int     sprintf(char *, const char *, ...);
int     sscanf(const char *, const char *, ...);
int     ungetc(int, FILE *);
```

Приложение Б

Примеры применения функций C для работы со строками

Пример 1. Функция strlen

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

int main(void)
{
char* str1 = "Count.";
// strlen gives the length of single-byte character string
printf("Length of '%s' : %d\n", str1, strlen(str1) );

return 0;
}
```

```
Length of 'Count.' : 6
```

Пример 2. Функции strcpy и strcat

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

int main(void)
{
char string[80];

// Note that if you change the previous line to char string[20];
// strcpy and strcat will happily overrun the string buffer.
// See the examples for strncpy and strncat for safer string handling.

strcpy( string, "Hello world from " ); // C4996
// Note: strcpy is deprecated; consider using strcpy_s instead
strcat( string, "strcpy " ); // C4996
// Note: strcat is deprecated; consider using strcat_s instead
strcat( string, "and " ); // C4996
strcat( string, "strcat!" ); // C4996
printf( "String = %s\n", string );

return 0;
}
```

```
String = Hello world from strcpy and strcat!
```

Пример 3. Функции strcpy и strncat

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

#define MAXSTRINGLEN 39

char string[MAXSTRINGLEN+1];
// or char *string = malloc(MAXSTRINGLEN+1);

void BadAppend( char suffix[], int n )
{
strncat( string, suffix, n );
}

void GoodAppend( char suffix[], size_t n )
{
strncat( string, suffix, min( n, MAXSTRINGLEN-strlen(string)) );
}

int main(void)
{
string[0] = '\0';
printf( "string can hold up to %d characters\n", MAXSTRINGLEN );

strcpy( string, "This is the initial string!" );
// concatenate up to 20 characters...
BadAppend( "Extra text to add to the string...", 20 );
printf( "After BadAppend : %s (%d chars)\n", string, strlen(string) );

strcpy( string, "This is the initial string!" );
// concatenate up to 20 characters...
GoodAppend( "Extra text to add to the string...", 20 );
printf( "After GoodAppend: %s (%d chars)\n", string, strlen(string) );

return 0;
}
```

```
string can hold up to 39 characters
After BadAppend : This is the initial string!Extra text to add to (47 chars)
After GoodAppend: This is the initial string!Extra text t (39 chars)
```

Пример 4. Функции strchr и strrchr

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

int ch = 'r';

char string[] = "The quick brown dog jumps over the lazy fox";
char fmt1[] = " 1 2 3 4 5";
char fmt2[] = "12345678901234567890123456789012345678901234567890";

int main(void)
{
char *pdest;
int result;

printf( "String to be searched:\n %s\n", string );
printf( " %s\n %s\n\n", fmt1, fmt2 );
printf( "Search char: %c\n", ch );

// Search forward.
pdest = strchr( string, ch );
result = (int)(pdest - string + 1);
if ( pdest != NULL )
printf( "Result: first %c found at position %d\n", ch, result );
else
printf( "Result: %c not found\n", ch );

// Search backward.
pdest = strrchr( string, ch );
result = (int)(pdest - string + 1);
if ( pdest != NULL )
printf( "Result: last %c found at position %d\n", ch, result );
else
printf( "Result:\t%c not found\n", ch );

return 0;
}
```

```
String to be searched:
The quick brown dog jumps over the lazy fox
1 2 3 4 5
12345678901234567890123456789012345678901234567890

Search char: r
Result: first r found at position 12
Result: last r found at position 30
```

Пример 5. Функция strspn

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

int main(void)
{
char string[] = "cabbage";
int result;
result = strspn( string, "abc" );
printf( "The portion of '%s' containing only a, b, or c "
"is %d bytes long\n", string, result );

return 0;
}
```

```
The portion of 'cabbage' containing only a, b, or c is 5 bytes long
```

Пример 6. Функция strcspn

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

```
void test( const char * str, const char * strCharSet )
{
int pos = strcspn( str, strCharSet );
printf( "strcspn( \"%s\", \"%s\" ) = %d\n", str, strCharSet, pos );
}

int main(void)
{
test( "xyzbxz", "abc" );
test( "xyzbxz", "xyz" );
test( "xyzbxz", "no match" );
test( "xyzbxz", "" );
test( "", "abc" );
test( "", "" );

return 0;
}
```

```
strcspn( "xyzbxz", "abc" ) = 3
strcspn( "xyzbxz", "xyz" ) = 0
strcspn( "xyzbxz", "no match" ) = 6
strcspn( "xyzbxz", "" ) = 6
strcspn( "", "abc" ) = 0
strcspn( "", "" ) = 0
```

Пример 7. Функция strstr

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

char str[] = "lazy";
char string[] = "The quick brown dog jumps over the lazy fox";
char fmt1[] = " 1 2 3 4 5";
char fmt2[] = "12345678901234567890123456789012345678901234567890";

int main(void)
{
char *pdest;
int result;
printf( "String to be searched:\n %s\n", string );
printf( " %s\n %s\n\n", fmt1, fmt2 );
pdest = strstr( string, str );
result = (int)(pdest - string + 1);
if ( pdest != NULL )
printf( "%s found at position %d\n", str, result );
else
printf( "%s not found\n", str );

return 0;
}
```

```
String to be searched:
The quick brown dog jumps over the lazy fox
1 2 3 4 5
12345678901234567890123456789012345678901234567890

lazy found at position 36
```

Пример 8. Функция strpbrk

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

int main(void)
{
char string[100] = "The 3 men and 2 boys ate 5 pigs\n";
char *result = NULL;

// Return pointer to first digit in "string".
printf( "1: %s\n", string );
result = strpbrk( string, "0123456789" );
printf( "2: %s\n", result++ );
```

```
result = strpbrk( result, "0123456789" );
printf( "3: %s\n", result++ );
result = strpbrk( result, "0123456789" );
printf( "4: %s\n", result );

return 0;
}
```

```
1: The 3 men and 2 boys ate 5 pigs

2: 3 men and 2 boys ate 5 pigs

3: 2 boys ate 5 pigs

4: 5 pigs
```

Пример 9. Функция strtok

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

char string[] = "A string\tof ,,tokens\nand some more tokens";
char seps[] = " ,\t\n";
char *token;

int main(void)
{
printf( "Tokens:\n" );
// Establish string and get the first token:
token = strtok( string, seps ); // C4996
// Note: strtok is deprecated; consider using strtok_s instead
while( token != NULL )
{
// While there are tokens in "string"
printf( " %s\n", token );

// Get next token:
token = strtok( NULL, seps ); // C4996
}

return 0;
}
```

```
Tokens:
A
string
of
tokens
and
some
more
tokens
```

Пример 10. Функция sscanf из библиотеки C stdio.h

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

int main(void)
{
char tokenstring[] = "15 12 14...";
char s[81];
char c;
int i;
float fp;

// Input various data from tokenstring:
// max 80 character string:
sscanf( tokenstring, "%80s", s ); // C4996
sscanf( tokenstring, "%c", &c ); // C4996
sscanf( tokenstring, "%d", &i ); // C4996
sscanf( tokenstring, "%f", &fp ); // C4996
```

```
// Output the data read
printf( "String = %s\n", s );
printf( "Character = %c\n", c );
printf( "Integer: = %d\n", i );
printf( "Real: = %f\n", fp );
}

String = 15
Character = 1
Integer: = 15
Real: = 15.000000
```

Пример 11. Функция sprintf из библиотеки C stdio.h

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

int main(void)
{
char buffer[200], s[] = "computer", c = 'l';
int i = 35, j;
float fp = 1.7320534f;

// Format and print various data:
j = sprintf( buffer, " String: %s\n", s ); // C4996
j += sprintf( buffer + j, " Character: %c\n", c ); // C4996
j += sprintf( buffer + j, " Integer: %d\n", i ); // C4996
j += sprintf( buffer + j, " Real: %f\n", fp );// C4996
// Note: sprintf is deprecated; consider using sprintf_s instead

printf( "Output:\n%s\ncharacter count = %d\n", buffer, j );

return 0;
}

Output:
String: computer
Character: l
Integer: 35
Real: 1.732053

character count = 79
```