

Функции с переменным числом параметров.

```
int sum(int n, ...)
{
    int result = 0;
    va_list params;          //указатель va_list
    va_start(params, n);    // устанавливаем указатель
    for(int i=0; i<n; i++)
    {
        result += va_arg(params, int); // получаем значение текущего параметра типа
    }
    va_end(params); // завершаем обработку параметров
    return result;
}
```

Стандартная библиотека CLib

Можно изобретать свои велосипеды, но иногда быстрее, надежнее и проще поехать на фабричном.

CLib — стандартная библиотека языка C. Включает множество функций, сгруппированных по разделам, описываемым разными заголовочными файлами.

<assert.h>	void assert(int) --- проверка условия / завершение при false, отключение #define NDEBUG
<ctype.h>	проверка категорий ASCII символов, преобразование tolower toupper
<errno.h>	extern int errno;
<float.h>	макро (константы), касающиеся представлений float и double
<limits.h>	макро (константы), касающиеся диапазонов разных целых
<locale.h>	структура, описывающая "национальные" представления (деньги, даты)
<math.h>	элементарные функции
<setjmp.h>	сохранение/восстановление окружения программы
<signal.h>	обработка сигналов (SIGFPE, SIGSEGV, SIGINT и др.)
<stdarg.h>	поддержка переменного числа параметров у функций (va_list, va_start(), va_arg(), va_end())
<stddef.h>	контроль позиционирования полей в структурах
<stdio.h>	ввод/вывод
<stdlib.h>	нечто "стандартное"
<string.h>	работа с ASCII строками
<time.h>	время (в основном календарное)

Для C++ есть аналоги с заголовочными файлами cstdio, cmath, cstdlib и т.п.

Файл math.h

```
-----
double acos(double x)
double asin(double x)
double atan(double x)
double atan2(double y, double x)
double cos(double x)
```

```

double cosh(double x)
double sin(double x)
double sinh(double x)
double tanh(double x)
double exp(double x)
double frexp(double x, int *exponent)    x = mantissa * 2 ^ exponent.
double ldexp(double x, int exponent)    x * 2 ^ exponent.
double log(double x)
double log10(double x)
double modf(double x, double *integer)  {x} x [x]
double pow(double x, double y)          exp(y*log(x))
double sqrt(double x)
double ceil(double x)                   целое сверху
double fabs(double x)
double floor(double x)                  целое снизу
double fmod(double x, double y)         x % y

```

Файл ctype.h

```

int isalnum(int c)
int isalpha(int c)
int iscntrl(int c)
int isdigit(int c)
int isgraph(int c)
int islower(int c)
int isprint(int c)
int ispunct(int c)
int isspace(int c)
int isupper(int c)
int isxdigit(int c)

```

```

int tolower(int c)
int toupper(int c)

```

Файл stdlib.h

```

NULL EXIT_FAILURE EXIT_SUCCESS RAND_MAX

```

```

double atof(const char *str)
int atoi(const char *str)
long int atol(const char *str)
double strtod(const char *str, char **endptr)
long int strtol(const char *str, char **endptr, int base)
unsigned long int strtoul(const char *str, char **endptr, int base)

```

```

void *calloc(size_t nitems, size_t size)
void free(void *ptr)
void *malloc(size_t size)
void *realloc(void *ptr, size_t size)

```

```

void abort(void)
int atexit(void (*func)(void))
void exit(int status)
char *getenv(const char *name)
int system(const char *string)

void *bsearch(const void *key, const void *base, size_t nitems, size_t size,
              int (*compar)(const void *, const void *))
void qsort(void *base, size_t nitems, size_t size,
           int (*compar)(const void *, const void*))
int abs(int x)
div_t div(int numer, int denom)
int rand(void)
void srand(unsigned int seed)

```

Файл time.h

CLOCKS_PER_SEC

clock_t

time_t секунды с 00:00:00 1.01.1970

```

struct tm {
int tm_sec;               /* seconds,  range 0 to 59               */
int tm_min;               /* minutes,  range 0 to 59               */
int tm_hour;              /* hours,    range 0 to 23               */
int tm_mday;              /* day of the month, range 1 to 31       */
int tm_mon;               /* month,    range 0 to 11               */
int tm_year;              /* The number of years since 1900       */
int tm_wday;              /* day of the week, range 0 to 6        */
int tm_yday;              /* day in the year, range 0 to 365       */
int tm_isdst;             /* daylight saving time                 */
};

```

char *asctime(const struct tm *timeptr)

clock_t clock(void)

char *ctime(const time_t *timer)

double difftime(time_t time1, time_t time2) секунды

struct tm *gmtime(const time_t *timer)

struct tm *localtime(const time_t *timer)

time_t mktime(struct tm *timeptr)

size_t strftime(char *str, size_t maxsize, const char *format,
 const struct tm *timeptr)

time_t time(time_t *timer)

Файл stdio.h

stdin stdout stderr

```

FILE
EOF
SEEK_SET  SEEC_CUR  SEEK_END

int feof(FILE *stream)
int ferror(FILE *stream)

int fflush(FILE *stream)
int fgetpos(FILE *stream, fpos_t *pos)
int fsetpos(FILE *stream, const fpos_t *pos)
FILE *fopen(const char *filename, const char *mode)
FILE *freopen(const char *filename, const char *mode, FILE *stream)
int fclose(FILE *stream)
    режимы открытия  "r" "w" "a"  "rb" "wb" "ab",  текстовый: "\n" <-> "\r\n"
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)

int fseek(FILE *stream, long int offset, int whence)
long int ftell(FILE *stream)
void rewind(FILE *stream)          fseek(stream, 0 , SEEK_SET)

int remove(const char *filename)
int rename(const char *old_filename, const char *new_filename)
FILE *tmpfile(void)
char *tmpnam(char *str)

int fprintf(FILE *stream, const char *format, ...)
int printf(const char *format, ...)
int sprintf(char *str, const char *format, ...)
int vfprintf(FILE *stream, const char *format, va_list arg)
int vprintf(const char *format, va_list arg)
int vsprintf(char *str, const char *format, va_list arg)

int fscanf(FILE *stream, const char *format, ...)
int scanf(const char *format, ...)
int sscanf(const char *str, const char *format, ...)

int fgetc(FILE *stream)
char *fgets(char *str, int n, FILE *stream)
int fputc(int char, FILE *stream)
int fputs(const char *str, FILE *stream)

int getc(FILE *stream)
int getchar(void)
char *gets(char *str)
int putc(int char, FILE *stream)
int putchar(int char)
int puts(const char *str)

```

```
int ungetc(int char, FILE *stream)
```

```
void perror(const char *str)
```

Работа со строками

Строка в языке С — это `char` массив, в котором обязательно присутствует байт, имеющий нулевое значение. Этот нулевой байт обозначает конец строки. Например, пустая строка занимает 1 байт, в котором как раз и записано нулевое значение, а строка, содержащая, например, текст “это слово” занимает 10 байтов, из которых 9 относится к печатным символам (включая и пробел), а один последний байт — это опять тот самый завершающий ноль.

Если ваша программа работает со строками, то вы должны предусматривать необходимое место в используемых `char` массивах для размещения всех символов строки, в том числе и завершающего нуля.

Для работы со строками в стандартной библиотеке предусмотрено некоторое количество функций, которые используют соглашение о завершающем нуле, т.е. ставят его, если они формируют строку для результата работы и рассчитывают на его наличие, если принимают строки в качестве входных параметров. Как правило, стандартные функции не проверяют наличие необходимого места в используемых `char` массивах, за этим должен следить сам программист. Ошибки в оценке необходимого размера массивов или отсутствие завершающего нуля могут привести к выходу за границы массивов и соответственно к аварийному завершению программы.

Стандартные функции работы со строками в С ориентированы на однобайтовую кодировку символов. Такими кодировками являются, например, `cp1251` (Windows) или `ko8-r` (Unix). Однако в настоящее время часто используется Unicode кодировка UTF-8, которая использует переменное количество байт на символ. Для латинского алфавита, цифр, знаков препинания — это 1 байт, а для других символов — от 2 до 4 байтов (для букв русского языка — 2 байта). Тем не менее, с этими функциями вполне можно работать и с русским текстом, если помнить о том, что русские буквы требуют в два раза больше памяти для размещения в строке.

Стандартные функции работы со строками включают

```
void *memcpy(char *dest, const char *src, size_t n);
    копирует n байт из области памяти src в dest, которые не
    должны пересекаться, иначе результат не определён
int a[100], b[100];
for (i=0; i<100; i++) a[i] = b[i];
или
memcpy((char *)a, (const char *)b, 100*sizeof(int));

void *memmove(char *dest, const char *src, size_t n);
    копирует n байт из области памяти src в dest,
    которые в отличие от memcpy могут перекрываться

void *memchr(const char *s, char c, size_t n);
    возвращает указатель на первое вхождение значения c
    среди первых n байтов s или NULL, если не найдено

int memcmp(const char *s1, const char *s2, size_t n);
    сравнивает первые n байтов в областях памяти
```

```
void *memset(char *, int z, size_t);
    заполняет область памяти одним байтом z
for (i=0; i<100; i++) a[i] = 0;
или
memset((char *)a, 0, 100*sizeof(int));

char *strcat(char *dest, const char *src);
    дописывает строку src в конец dest
char *strncat(char *dest, const char *src, size_t n);
    дописывает не более n начальных символов строки src
    (или всю src, если ее длина меньше) в конец dest
char *strchr(const char *s, int c);
    возвращает адрес символа c в строке s, начиная с головы,
    или NULL, если строка s не содержит символ c
char *strrchr(const char *s, int c);
    возвращает адрес символа c в строке s, начиная с хвоста,
    или NULL, если строка s не содержит символ c
int strcmp(const char *, const char *);
    лексикографическое сравнение строк
int strncmp(const char *, const char *, size_t);
    лексикографическое сравнение первых n байтов строк
int strcoll(const char *, const char *);
    лексикографическое сравнение строк с учетом локали collating order
char *strcpy(char *toHere, const char *fromHere);
    копирует строку из одного места в другое
char *strncpy(char *toHere, const char *fromHere, size_t n);
    копирует до n байт строки из одного места в другое
char *strerror(int);
    возвращает строковое представление сообщения об ошибке errno
size_t strlen(const char *);
    возвращает длину строки
size_t strspn(const char *s, const char *accept);
    определяет максимальную длину начальной подстроки, состоящей
    исключительно из байтов, перечисленных в accept
size_t strcspn(const char *s, const char *reject);
    определяет максимальную длину начальной подстроки, состоящей
    исключительно из байтов, не перечисленных в reject
char *strpbrk(const char *s, const char *accept);
    находит первое вхождение любого символа, перечисленного в accept
char *strstr(const char *haystack, const char *needle);
    находит первое вхождение строки needle в haystack
char *strtok(char *, const char *);
    преобразует строку в последовательность токенов.
```