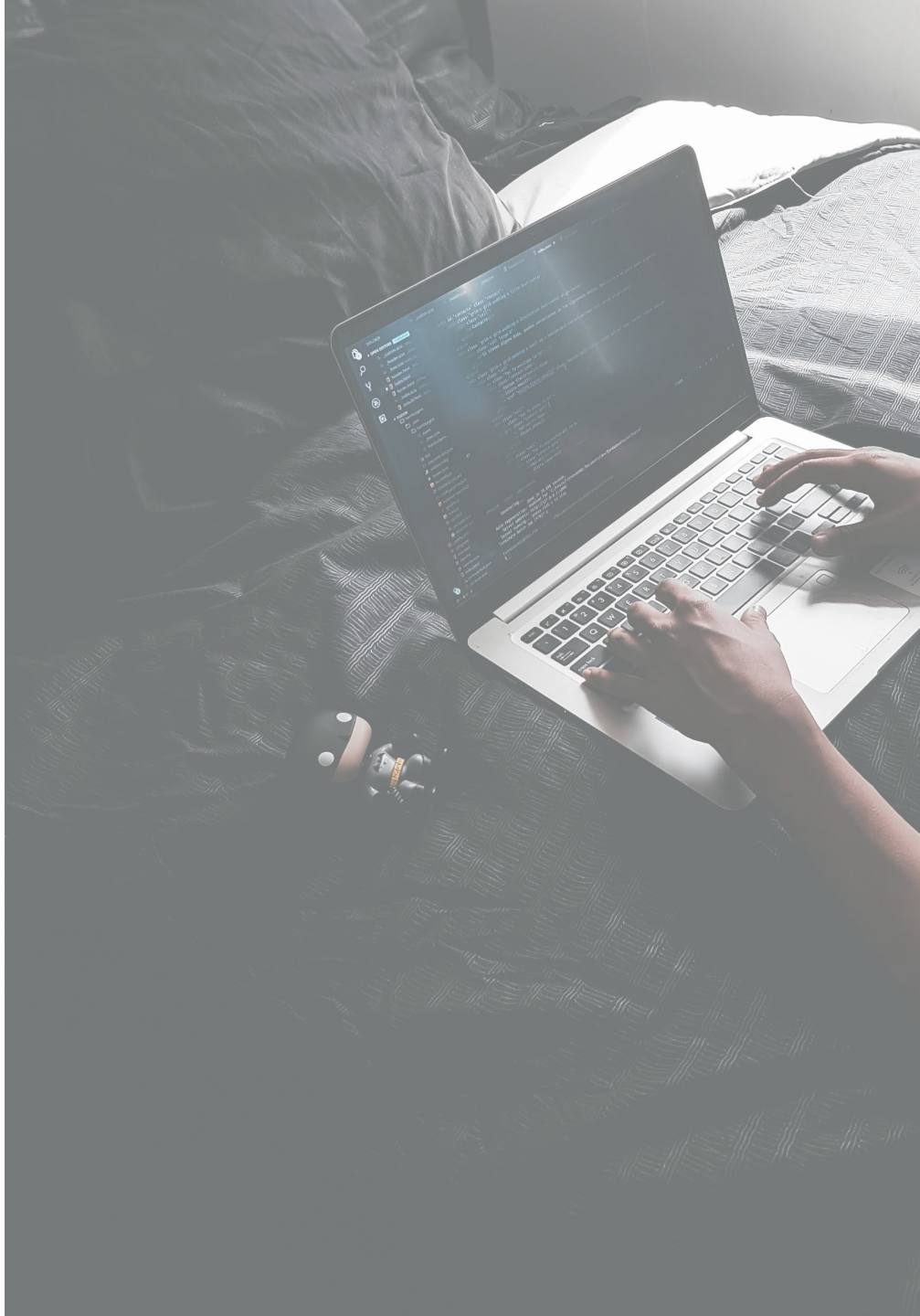
GeekBrains

Урок 9. Адресная арифметика. Массивы, строки



Урок 9

На этом уроке

- Узнаем про адресную арифметику в массивах
- Как работают указатели
- Как массивы передаются в функции
- Разберём, что такое строки
- Как работает scanf

Массивы и указатели

В языке Си имя массива — это указатель на самый первый элемент массива

```
int a[5];
 int b[5];
 a = b; // так делать нельзя
```

Можно создать обычный указатель, установить его на нулевой элемент массива, а после изменять уже его

```
int a[5];
int *pa;
pa = a; // так можно
pa = &a[0]; // и так
```

Операция sizeof

`int a[5];`
`int *pa;`
`pa = a;`
`printf("sizeof(a) = %lu\n", sizeof(a));`
`printf("sizeof(pa) = %lu\n", sizeof(pa));`
	sizeof(a) = 20
	sizeof(pa) = 8

Размер массива a = <количество элементов> * <размер типа> = 5 * sizeof(int) = 20.
Размер указателя pa = 8 для 64-х разрядной системы или 4 для 32-х.

Операции над указателями

- Операции сравнения: ==, !=, >=, <=, <, >
- Вычитание и добавление целочисленной константы
- Вычитание одного указателя из другого
- Унарные операции: ++, --
- Операции: -=, +=

Пример

```
    int a[5];
    printf("&a[0] = %p\n",
&a[0]);
    printf("&a[1] = %p\n",
&a[1]);
```

```
&a[0] = ?
&a[1] =?
```

Пример

```
    int a[5];
    printf("&a[0] = %p\n",
&a[0]);
    printf("&a[1] = %p\n",
&a[1]);
```

```
&a[0] = 0x7ffee1a20ab0
&a[1] = 0x7ffee1a20ab4
```

Пример

```
    int a[5] =
{10,20,30,40,50};
    int *pa, n;
    pa = a;
    n = *(pa+2);
    printf("n = %d\n", n);
```

```
n = ?
```

Пример

```
    int a[5] =
{10,20,30,40,50};
    int *pa, n;
    pa = a;
    n = *(pa+2); // a[2]
    printf("n = %d\n", n);
```

```
n = 30
```

Пример

При вычислении операции pa+2 к адресу, хранящемуся в переменной pa, присваивается значение 2 * sizeof(int), это как раз соответстветствует элементу с индексом 2, массива a

```
    int a[5] =
{10,20,30,40,50};
    int *pa, n;
    pa = a;
    n = *pa+2;
    printf("n = %d\n", n);
```

```
n = ?
```

Пример

При вычислении операции pa+2 к адресу, хранящемуся в переменной pa, присваивается значение 2 * sizeof(int), это как раз соответстветствует элементу с индексом 2, массива a

```
    int a[5] =
{10,20,30,40,50};
    int *pa, n;
    pa = a;
    n = *pa+2; // a[0] + 2
    printf("n = %d\n", n);
```

```
n = 12
```

Пример

При вычитании одного указателя из другого, вычисляется разность адресов, которая затем делится на размер (в байтах) элемента массива. Результатом является количество элементов массива, расположенных между данными указателями

```
int a[5] = {10,20,30,40,50};
int *pa, *qa;
pa = &a[1];
qa = &a[3];
printf("%ld\n", qa-pa);
```

```
2
```

Что будет напечатано?

```
int a[5] = {10,20,30,40,50};
int *pa, n=10;
pa = a+2;
*pa++ = n+3;
printf("%d\n",*pa);
printf("%d %d %d %d %d\n",
   a[0],a[1],a[2],a[3],a[4]);
```

?

Что будет напечатано?

```
int a[5] = {10,20,30,40,50};
int *pa, n=10;
pa = a+2;
*pa++ = n+3;//*++pa= n+3;?
printf("%d\n",*pa);
printf("%d %d %d %d %d\n",
  a[0],a[1],a[2],a[3],a[4]);
```

```
40
10 20 13 40 50
```

Пример

Рассмотрим порядок выполнения строки *pa++ = n+3, в данном случае необходимо правильно расставить приоритете операций: *(pa++) = n+3; Операция инкремента меняет адрес (прибавляет 4 байта)

```
*pa = n+3;
 pa++;
```

Что будет напечатано?

```
    int a[5] =
{10,20,30,40,50};
    int *pa, n=10;
    pa = a+2;
    n = ++*pa; // ++(*pa)
    printf("%d\n",n);
```

?

Пример

```
    int a[5] =
{10,20,30,40,50};
    int *pa, n=10;
    pa = a+2;
    n = ++*pa; // ++(*pa)
    printf("%d\n",n);
```

```
31
```

Пример

Имя массива является неизменяемым указателем на его нулевой элемент: при указании в качестве параметра, имени массива в функцию передаётся копия адреса начала той области памяти, в которой находятся элементы массива

Также известно, сколько места занимает каждый элемент массива, что позволяет обратиться к любому элементу массива

Однако информация о количестве элементов массива теряется, поэтому размер массива следует передавать в функцию через дополнительный параметр

```
int sum(int a[], int size) {
    int i, s=0;
    for(i=0;i<size;i++)
        s+=a[i];
    return s;
}
```

```
int sum(int *a, int size) {
    int i, s=0;
    for(i=0;i<size;i++)
        s+=a[i];
    return s;
}
```

Что будет напечатано?

`int a[5] = {10,20,30,40,50};`
`printf("%d\n",sum(a,5));`
`printf("%d\n",sum(a,2));`
 	
`printf("%d\n",sum(a+2,3));`
	?

Что будет напечатано?

```
    int a[5] =
{10,20,30,40,50};
    printf("%d\n",sum(a,5));
    printf("%d\n",sum(a,2));

printf("%d\n",sum(a+2,3));
```

```
150
30
120
```

Пример

Можно в 0-м элементе массива хранить длину

```c
int sum0(int* a0){
    int i, s=0;
    for(i=1;i<a0[0]+1;i++)
        s+=a0[i];
    return s;
}
```

```c
#define SIZE 5
int main(int argc, char **argv)
{
    int a0[SIZE+1] = {SIZE,10,20,30,40,50};
    printf("%d\n",sum0(a0));
    return 0;
}
```

Задачи

1. Написать функцию вычисления скалярного произведения двух вещественных массивов
2. Написать функцию сдвига массива на 1 элемент влево
3. Написать функцию сдвига массива на N элементов вправо
4. Написать функцию реверса массива. Первый элемент становится последним, второй элемент — предпоследним и т.д.
5. Составить функцию, которая определяет в массиве, состоящем из положительных и отрицательных чисел, сколько элементов превосходят по модулю максимальный элемент. Показать пример её работы на массиве из 10 элементов

Задачи

Написать функцию вычисления скалярного произведения двух вещественных массивов

```
#include <stdio.h>
float scalar(float*arrA,float*arrB,int len)
{
    float result = 0;
    for(int i=0;i<len;i++)
    {
//        result += arrA[i]*arrB[i];
//        result += (float*)(arrA+i) * (float*)(arrB+i);
        result += *(arrA+i) * *(arrB+i);
//        printf("%f,%f,%f\n",result,*(arrA+i),*(arrB+i));
    }
    return result;
}
```

Задачи

Написать функцию вычисления скалярного произведения двух вещественных массивов

```
#define SIZE 3
int main(int argc, char **argv)
{
    float arrA[SIZE]={1,2,3};
    float arrB[SIZE]={1,2,3};
    //float arrC[SIZE]={1,2,3};
    printf("%f",scalar(arrA,arrB,SIZE));
    //Print(arrC,SIZE);
    return 0;
}
```

Многомерные массивы

Можно определять многомерные массивы, при этом массив размерности n рассматривается как одномерный массив, каждый элемент которого представляет собой массив размерности n - 1. По сути, двумерный массив является фактически одномерным массивом, каждый элемент которого в свою очередь также является одномерным массивом.

```
int matr[3][5]; // 3 строки и 5 столбцов
```

matr[0][0]	matr[0][1]	matr[0][2]	matr[0][3]	matr[0][4]
matr[1][0]	matr[1][1]	matr[1][2]	matr[1][3]	matr[1][4]
matr[2][0]	matr[2][1]	matr[2][2]	matr[2][3]	matr[2][4]

Многомерные массивы

Можно объявлять массивы размерностью больше 2-ух

```
int box[10][20][30]; // 10 матриц по 20 строк и 30 столбцов
```

Пример

Можно объявить указатель на двумерный массив, однако при этом теряется информация о размере строки.

```
int matr[3][2];
int *pm;
pm = matr;
pm[1][1] = 123;  //ТАК НЕЛЬЗЯ
```

```
int matr[3][2];
int *pm;
pm = matr;
*(pm + 3) = 123; // matr[1][1]
pm[3] = 123; // или так
printf("%d\n",matr[1][1]);
```

Пример

А можно объявить указатель на строку матрицы

```
int matr[3][2];
int (*pm)[2]; //указатель на строку из 2-ух int
pm = matr;
pm[1][1] = 123; //теперь все ок
printf("%d\n",matr[1][1]); // 123
```

Что будет напечатано?

```
int m[3][3] =
{
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
int *p1;
int (*p2)[3];
p1 = m[1];
p2 = m + 1;
p1++;
p2++;
printf ("%d %d\n", *p1, **p2);
```

	m[i]		m [i] [j]	
m	m[0]	1	2	3
	m[1]	4	5	6
	m[2]	7	8	9

Что будет напечатано?

```
int m[3][3] =
{
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
int *p1;
int (*p2)[3];
p1 = m[1];
p2 = m + 1;
p1++;
p2++;
printf ("%d %d\n", *p1, **p2);
```

5 7

VLA массивы

В C99 можно описывать локальные массивы неконстантного размера (Variable-length array) и выделять для них динамическую память. Без использования указателей VLA массивы могут быть использованы как локальные переменные либо параметры функций.

```
   uint32_t n;
    scanf("%u",&n); //вводим количество
элементов
    int32_t ar[n]; //создаем VLA массив
```

VLA массивы

Внимание! Если массив используется в качестве параметра функции и размерности массива передаются в эту функцию, то указатель должен следовать после описания размеров.

// Делайте так	// ТАК НЕЛЬЗЯ - ОШИБКА
`void func(int m, int n, int arr[m][n])`	`void func(int arr[m][n], int m, int n)`

Массивы указателей

В таком массиве каждый элемент это ссылка на объект одного типа. Обычно массивы указателей ссылаются на строки. Каждый элемент содержит адрес нулевого символа строки. Очевидное преимущество такого способа хранения - это разная длина хранящихся строк.

```
char *ps[] = {"one", "two", "three", NULL}; // NULL признак конца
for(uint32_t i=0; ps[i] ;i++)
    printf("%s\n", ps[i]);
```

Задачи

Написать функцию, которая выводит на печать матрицу

```c
void print_matrix_1 (int m, int n, int *a)
{
    int i, j;
    for (i = 0; i < m; i++){
        for (j = 0; j < n; j++){
            printf ("%d ", a[i * n + j]);
        }
        printf ("\n");
    }
}
```

```c
void print_matrix_2 (int m, int n, int a[m][n])
{
    int i, j;
    for (i = 0; i < m; i++){
        for (j = 0; j < n; j++){
            printf ("%d ", a[i][j]);
        }
        printf ("\n");
    }
}
```

Генератор случайных чисел

Для организации тестирований программ с использованием массивов удобно заполнить массив случайными числами. Используем библиотеку stdlib.h

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

int main() {
    int x;
    printf("MAX RANDOM = %d\n",RAND_MAX); //2147483647
// Задать начальное значение последовательности
    srand(123); // одни и те же числа, seed задан константно
    srand( time(NULL) ); // числа меняются при каждом запуске программы
//Случайное целое число в интервале [0, RAND_MAX]
    x = rand();
    printf("x = %d\n",x);// первое случайное число
    x = rand();
    printf("x = %d\n",x);// другое случайное число



    return 0;
}
```

Генератор случайных чисел

```c
#include <stdio.h>
#include <stdlib.h>
enum {SIZE = 10, SEED = 123};

int random_number(int n) {
    return rand() % n;
}

int main(void)
{
    int a[SIZE], i;
    srand(SEED);
    printf("Array:\n");
    for (i = 0; i < SIZE; i++ ) {
        a[i] = random_number(100) + 50;
        printf("%4d", a[i]);
    }
    return 0;
}
```

Задачи

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

enum {SIZE = 10, SEED = 123};

int random_number(int n) {
    return rand() % n;
}
void init_array(int size, int a[], int max_random) {
    for (size_t i = 0; i < size; i++ ) {
        a[i] = random_number(max_random);
    }
    return;
}
```

```c
void print_array(int size, int a[]) {
    for (size_t i = 0; i < size; i++ ) {
        printf("%d ",a[i]);
    }
    printf("\n");
    return;
}
int main(void)
{
    int a[SIZE];
    srand(SEED);
    init_array(SIZE, a, 100);
    print_array(SIZE,a);
    return 0;
}
```

Манипуляции с массивами

```
// циклический сдвиг массива влево на 1 элемент
void shift_array_left(int size, int a[]) {
    int tmp=a[0];
    for (size_t i=0; i < size-1; i++ ) {
        a[i] = a[i+1];
    }
    a[size-1] = tmp;
}
```

Манипуляции с массивами

```
void swap(int*a,int*b)
{
	int t = *a;
	*a = *b;
	*b = t;
}
/*
 Реверс массива
 */
void revers_array(int size, int a[]) {
    for(size_t i=0; i<size/2; i++) {
        swap(&a[i], &a[size-1-i]);
    }
    return;
}
```

Манипуляции с массивами

```
/*
 Сортировка пузырьком
 */
void buble_sort_array(int size, int a[]) {
    int i = 0;
    _Bool flag;
    do {
        flag = 0; // сбросить флаг
        for (int j = size-2; j >= i ; j-- )
        if ( a[j] > a[j+1] ) {
            swap(&a[j], &a[j+1]);
            flag = 1;  // поднять флаг если есть обмен
        }
        i++;
    }
    while ( flag );  // выход при flag = 0
}
```

Манипуляции с массивами

GeekBrains

Сортировка выбором — Википедия

```
/*
 Сортировка выбором
 */
void choose_sort_array(int size, int a[]) {
    int nMin;
    for(int i = 0; i <  size-1 ; i ++ ) {
        for (int j =  i+1; j < size; j ++)
        if( a[j] < a[nMin] ) {
            nMin = j;
        }
        if( nMin != i ) {
            swap(&a[i], &a[nMin]);
        }
    }
}
```

Сортировка qsort

- Функция qsort() сортирует массив, на который указывает параметр base, используя quicksort — алгоритм сортировки широкого назначения, разработанный Т. Хоаром. Параметр num задает число элементов массива, параметр size задает размер в байтах каждого элемента.
- Функция, на которую указывает параметр compare, сравнивает элементы массива с ключом. Формат функции compare следующий:
- int func_name(const void *arg1, const void *arg2)
- Она должна возвращать следующие значения:
- Если arg1 меньше, чем arg2, то возвращается отрицательное целое.
- Если arg1 равно arg2, то возвращается 0.
- Если arg1 больше, чем arg2, то возвращается положительное целое.
- Массив сортируется по возрастанию таким образом, что наименьший адрес соответствует наименьшему элементу.

Пример

```
/* сравнение двух целых V1 */
int comparator (const int *a, const int *b) {
    return *a - *b;
}
int main(void)
{
....
    qsort(a, SIZE, sizeof (int), (int(*) (const void *, const void *)) comparator);
....
}
```

Пример

```
/* сравнение двух целых V2 */
int comparator (const void *a, const void *b) {
    return *(int *)a - *(int *)b;
}
int main(void)
{
....
    qsort(a, SIZE, sizeof (int), comparator);
....
}
```

Читаем декларатоpы

- Сайт для декодирования деклораторов https://cdecl.org/
- Декларатор содержит имя определяемого объекта, но в некоторых местах может не содержать имя определяемого объекта. Анонимные декларатоpы допускаются в операции приведения типа и при описании формальных параметров в прототипах функций. Например декларатор может иметь вид:
- char (*(*x[3])())[10]; - на самом деле это значит объявить x как массив 3 указателей на функцию, возвращающую указатель на массив из 10 char.

Пример

int a[3][4]; - массив из 3 элементов типа массива из 4 элементов int

char **b; - указатель на указатель на char

char *c[]; - массив из неопределённого количества элементов типа указатель на тип char

int *d[10]; - массив из 10 элементов типа указатель на тип int

int (*a)[10]; - указатель на массив из 10 элементов типа int

int (*a)[5]; - указатель на массив из 5 элементов типа int

int *f(); - функция, возвращающая указатель на int

int (*f)(); - f это указатель на функцию, возвращающую значение int

int *(*f)(); - указатель на функцию, возвращающую указатель на int

Пример

```
typedef  int(*comparator_type) (const void *, const void *);

/* сравнение двух целых V1 */
int comparator (const int *a, const int *b) {
    return *a - *b;
}
int main(void)
{
….
    qsort(a, SIZE, sizeof (int), (comparator_type) comparator);
….
}
```

Пример

```
typedef  int(*comparator_type) (const void *, const void *);

/* сравнение двух целых V1 */
int comparator (const int *a, const int *b) {
    return *a - *b;
}
int main(void)
{
….
    qsort(a, SIZE, sizeof (int), (comparator_type) comparator);
….
}
```

VLA массивы

В C99 можно описывать локальные массивы неконстантного размера (Variable-length array) и выделять для них динамическую память. Без использования указателей VLA массивы могут быть использованы как локальные переменные либо параметры функций.

```
   uint32_t n;
    scanf("%u",&n); //вводим количество
элементов
    int32_t ar[n]; //создаем VLA массив
```

Строки

Строка в языке Си — это массив из символов типа char, в конце символ с кодом 0(\0). Все программы и библиотечные функции, работающие со строками, основаны на том, что в конце строки обязательно есть символ 0

Особенности строк:

- Все строки состоят из символов с кодом отличным от 0
- Длина строки не содержится в самой строке
- Нет никаких ограничений на длину строки

Примеры объявления строк

```
    char s[10];// Строка из 9 значимых
символов
    char s[N];// Строка из N-1 значимых
символов, в строке всегда есть символ \0
```

Строки, также, как и массивы, можно объявлять с инициализацией

```
   char st[] = "hello";
// st[0]='h' st[1]='e' st[2]='l'
st[3]='l' st[4]='o' st[5]='\0'
   printf("%lu", sizeof(st)); // 6
```

В этой строке 5 значащих символа и 6-ой символ с кодом 0. Размер занимаемой памяти — 6 байт

```
   char st[10] = "hello"; // st[0]='h'
st[1]='e' st[2]='l' st[3]='l' st[4]='o'
st[5]='\0'
   printf("%lu", sizeof(st)); // 10
```

В этой строке 5 значащих символа и 6-ой символ с кодом 0. Размер занимаемой памяти — 10 байт

Ввод и вывод строк

```
char s[10];
scanf("%s",s); // считать строку до первого пробельного символа или \n
printf("%s",s); // напечатать строку
```

Для ввода и вывод строк можно также использовать функции printf и scanf, указав соответствующие спецификаторы в параметре форматной строки

Последовательно считывает в строку s символы вводимой строки и добавляет в конец строки символ '\0'. Ввод выполняется до первого символа пробел, символа перевода на следующую строку и т.п. Необходимо позаботиться о размере массива, куда считывается строка, сама функция никак эт не контролирует. Если строка больше, чем выделенное под неё место, то произойдёт аварийное завершение программы. Размер считываемой строки можно ограничить

Внимание! При выводе длины строки учитывается символ ‘\0’

```
char s[10];
scanf("%9s",s); // считать строку не более 9 символов
```

Ввод и вывод строк

Рассмотрим пример посимвольного ввода строки с помощью функции getchar()

```
char s[10], c;
 int i=0;
 while( (c=getchar())!='\n' )
     s[i++]=c;
 s[i]='\0';
```

Обратите внимание на скобки внутри while. Считываем строку до первого символа «перенос строки» и заносим всё в массив. Необходимо поставить признак конца строки после считывания. Напечатать строку можно также с помощью функции посимвольного вывода putchar()

```
while( s[i] ) // s[i] != 0
    putchar(s[i++]);
```

Множество сканирования scanf

```
char s[100];
scanf("%[a-z]",s);// считать стр буквы
printf("%s\n",s);
```

```
helloWORLD

?
```

```
scanf("%[0-9]",s);// считать цифры

printf("%s\n",s);
```

```
123WORLD

?
```

```
scanf("%[^\n]",s);// все кроме \n

printf("%s\n",s);
```

```
Hello world

?
```

```
char s1[100], s2[100];
scanf("%s%s",s1,s2);
printf("s1 = %s s2 = %s\n",s1,s2);
```

```
Hello world

s1 = ? s2 = ?
```

```
char s1[100], s2[100];
scanf("%[0-9]=%[a-z]", s1, s2);
printf("s1 = %s s2 = %s\n", s1, s2);
```

```
123=hello

s1 = ? s2 = ?
```

Множество сканирования scanf

`char s[100];`
`scanf("%[a-z]",s);// считать стр буквы`
`printf("%s\n",s);`
	`helloWORLD`
	
	`hello`

`scanf("%[0-9]",s);// считать цифры`
 	
`printf("%s\n",s);`
	`123WORLD`
	
	`123`

`scanf("%[^\n]",s);// все кроме \n`
 	
`printf("%s\n",s);`
	`Hello world`
	
	`Hello world`

`char s1[100], s2[100];`
`scanf("%s%s",s1,s2);`
`printf("s1 = %s s2 = %s\n",s1,s2);`
	`Hello world`
	
	`s1 = Hello s2 = world`

`char s1[100], s2[100];`
`scanf("%[0-9]=%[a-z]", s1, s2);`
`printf("s1 = %s s2 = %s\n", s1, s2);`
	`123=hello`
	
	`s1 = 123 s2 = hello`

Множество сканирования scanf

`char s1[100], s2[100];`
`int r;`
`r=scanf("%[0-9]=%[a-z]",s1,s2);`
`printf("r = %d\n",r);`
	123=hello
	r = ?

`r=scanf("%[0-9]=%[a-z]",s1,s2);` `printf("r = %d\n",r);`	123=123 r = ?

`r=scanf("%[0-9]=%[a-z]",s1,s2);` `printf("r = %d\n",r);`	hello=123 r = ?

`r=scanf("%[0-9]=%[a-z]",s1,s2);` `printf("r = %d\n",r);`	\EOF r = ?

Множество сканирования scanf

```
char s1[100], s2[100];
int r;
r=scanf("%[0-9]=%[a-z]",s1,s2);
printf("r = %d\n",r);
```

```
123=hello
r = 2
```

```
r=scanf("%[0-9]=%[a-z]",s1,s2);
printf("r = %d\n",r);
```

```
123=123
r = 1
```

```
r=scanf("%[0-9]=%[a-z]",s1,s2);
printf("r = %d\n",r);
```

```
hello=123
r = 0
```

```
r=scanf("%[0-9]=%[a-z]",s1,s2);
printf("r = %d\n",r);
```

```
\EOF
r = -1
```

Задачи

Для работы со строками существует стандартная библиотека string.h. Рассмотрим пример функции strlen (const char *cs), которая возвращает длину строки

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

int main(void) {
    char st[10] = "hello";
    printf("Sizeof = %lu\n",
sizeof(st));
    printf("Strlen = %lu\n",
strlen(st));
    return 0;
}
```

```
Sizeof = 10
Strlen = 6
```

Задачи

Для работы со строками существует стандартная библиотека string.h. Рассмотрим пример функции strlen (const char *cs), которая возвращает длину строки

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

int main(void) {
    char st[10] = "hello";
    printf("Sizeof = %lu\n",
sizeof(st));
    printf("Strlen = %lu\n",
strlen(st));
    return 0;
}
```

```
Sizeof = ?
Strlen = ?
```

Задачи

Для сравнения строк нельзя использовать операции ==, != и т.п., поскольку при этом происходит сравнение указателей на начало соответствующих строк, а не самих строк

`char st1[10] = "hello";`
`char st2[10] = "hello";`
`if(st1 == st2)`
`  printf("Yes");`
`else`
`  printf("No");`
	No

Задачи

Реализовать строковую функцию strlen(const char *cs)

```c
#include <stdio.h>
int strlen(const char *src)
{
    int len=0;
    while (*src++) len++;
    return len;
}
int main(int argc, char **argv)
{
char* str={"Hello!"};
    printf("%d\n",strlen(str));
    return 0;
}
```

Задачи

Реализовать строковую функцию strcpy(char *dst, const char *src)
Копирования строки src (включая '\0') в строку dst . Функция возвращает указатель на первый символ строки dst

```
#include <stdio.h>
char *strcpy (char *dst, char *src)
{
char *ptr = dst;
    while(*dst++=*src++);
    return ptr;
}
int main(int argc, char **argv)
{
char str1[]={"Hello!"};//char* str1 = {"Hello!"};
char str2[]={"World!"};//char* str2={"World!"}
    printf("%s\n",strcpy(str2,str1));
    printf("%s\n",str2);
    return 0;
}
```

Задачи

Реализовать строковую функцию strcmp(const char *cs, const char *ct)
Функция сравнивает в лексикографическом порядке строку cs со строкой ct. Если строка cs меньше строки ct, возвращается значение < 0, если строка cs больше строки ct, возвращается значение > 0, в случае равенства строк возвращается значение 0

```
#include <stdio.h>
int strcmp(const char *a, const char *b){
    while ( *a && *b && *a == *b )
        ++a, ++b;
    return *a - *b;
}
```

Задачи

Реализовать строковую функцию strcmp(const char *cs, const char *ct)

Функция сравнивает в лексикографическом порядке строку cs со строкой ct. Если строка cs меньше строки ct, возвращается значение < 0, если строка cs больше строки ct, возвращается значение > 0, в случае равенства строк возвращается значение 0

```
int main(void){
    char *a = "abcde";
    char *b = "xyz";
    char *c = "abcd";
    char *d = "xyz";
    int res;
    printf("A = %s\nB = %s\nC = %s\nD = %s\n\n", a, b, c, d);
    printf("A is %s B\n", ( ( res = strcmp(a, b) ) == 0 ) ? "equal to" : ( res < 0 ) ? "less" : "greater than");
    printf("A is %s C\n", ( ( res = strcmp(a, c) ) == 0 ) ? "equal to" : ( res < 0 ) ? "less" : "greater than");
    printf("A is %s D\n", ( ( res = strcmp(a, d) ) == 0 ) ? "equal to" : ( res < 0 ) ? "less" : "greater than");
    printf("B is %s C\n", ( ( res = strcmp(b, c) ) == 0 ) ? "equal to" : ( res < 0 ) ? "less" : "greater than");
    printf("B is %s D\n", ( ( res = strcmp(b, d) ) == 0 ) ? "equal to" : ( res < 0 ) ? "less" : "greater than");
    printf("C is %s D\n", ( ( res = strcmp(c, d) ) == 0 ) ? "equal to" : ( res < 0 ) ? "less" : "greater than");
    return 0;
}
```

Задачи

Посчитать количество слов в тексте, слова разделены одним или несколькими пробелами

Попробуйте решить данную задачу с использованием функции getchar()

```c
int main(void) {
    char s[100];
    int count=0;
    while(scanf("%s",s)==1)
        count++;
    printf("In this text %d
words\n",count);
    return 0;
}
```

```
hello   world.     I love
peace

In this text 5 words
```

Задачи

Реализовать функцию, которая преобразует переданную строку в массив байт, возвращает количество байт

int StrToHex(const char *str,char* Hex)

```
#include <stdio.h>
#include <stdint.h>
int strlen(const char *src)
{
    int len=0;
    while (*src++) len++;
    return len;
}
int CharToHex(char c)
{
  int result=-1;
  if      (c>='0' && c<='9') result=c-'0';
  else if(c>='A' && c<='F') result=c-'A'+10;
  else if(c>='a' && c<='f') result=c-'a'+10;
  return result;
}
```

```
int StrToHexMas(char* Str,uint8_t* mas);
int main(int argc, char **argv)
{
    uint8 t arr[10];
    int len = StrToHexMas("AAa a 1 15",arr);
   printf("%s\n","AAa a 1 15");
   printf("%d\n",len);
    for(int i=0;i<len;i++)
        printf("%02x,",arr[i]);
    return 0;
}
```

Задачи

Реализовать функцию, которая преобразует переданную строку в массив байт, возвращает количество байт
int StrToHex(const char *str,char* Hex)

```
//данные идут последовательно, не более двух сиволов
int StrToHexMas(char* Str,uint8_t* mas)
{
   int Result = 0; //порлученное число
   int data   = 0; //временная переменная
   int i      = 0; //счетчик сиволов по строке
   int index  = 0; //счетчик данных в массиве
   int StrLenght = strlen(Str);
   printf("%d\n",StrLenght);
   while(i<StrLenght)//выполняем цикл, пока есть символы в строке
   {
      Result=0;                    //обнуляем число
      data = CharToHex(Str[i++]); //анализируем очередной символ
      if(data>=0)                  //если это значащий символ
      {
         Result = data;
         if(i<StrLenght)     //проверка на выход за границы массива
         {
            data =
CharToHex(Str[i++]);//анализируем
очередной символ
            if(data>=0)
//если это данные
            {
               Result *= 16;
               Result += data;
            }
         }
         mas[index++]=Result;
//кладем число в массив
      }
   }
   return index;
}
```

Практическое задание

F1 - F20

Для новичков достаточно 9-10 задач на выбор!

* - сделать все!

ВАШИ ВОПРОСЫ