运算符、函数、数组

---

1 运算符

1.1 位操作

位域

```
#include <stdio.h>
typedef struct{
    int a:2;
    int b:3;
    int c:1;
}test;

int main()
{
    test t = {1, 3, 1};
    printf("%d %d %d %d", t.a, t.b, t.c, sizeof(test));
    //输出结果为：1 -1 -1 4
}
```

利用位运算的快速算法

```
a << 1;  //a*2
a >> 1;  //a/2
a & 7;   //a%8
(a << 3) - a;  //a*7
```

补码：正整数的补码是其二进制表示，与原码相同。负整数的补码，将其原码除符号位外的所有位取反后加1。

求-5的补码：

-5对应正数5（00000101）→所有位取反（11111010）→加1(11111011)。所以-5的补码是11111011。

```
/* 已知机器数采用补码形式，若寄存器内容为9CH，求对应的十进制数。*/
9CH 的二进制形式为 10011100
减1，得 10011011
取反码，得 01100100
故此负整数的绝对值为 64+32+4 = 100
此负整数为 -100
```

1.2 自增/自减运算符

```
int i=5,k;
k=(++i)+(++i)+(i++);
//++号在i前面的有两个，所以在计算k之前，i要先增两次，即i变为7
printf("%d,%d",k,i);
//结果为：21,8
```

1.3 类型转换

2 函数

2.1 通过地址调用函数

示例程序：

```
void fun(){
    printf("fun\n");
}
void callback(int i, void (*fun)()){
    printf("fun%d\n", i);
}
int main()
{
    /* 定义函数指针 */
    void (*ptr)();
    /* 打印函数fun的地址 */
    printf("0x%x\n", fun);
    /* 函数指针赋值 */
    ptr = (void (*)())0x4016fc;
    /* 通过函数指针调用函数fun */
    ptr();
    /* 通过地址调用函数fun */
    ((void (*)())0x4016fc)();
    /* 调用回调函数fun */
    callback(1, fun);
    return 0;
}
```

程序执行结果为：

```
0x4016fc
fun
fun
fun1
```

函数只根据函数名来获取入口地址，与参数和返回值无关。回调函数就是一个通过函数指针调用的函数，当一个函数指针作为参数传递给另一个函数时，就称为回调函数。

2.2 可变参数函数

C语言中支持函数调用的参数为变参形式。例如，printf函数的函数原型是`int printf( const char* format, ...)`，它除了有一个参数format固定以外，后面跟的参数的个数和类型都是可变的。

```c
void fun(int num, ...)
{
    int i;
    int *p = &num+1;
    for(i = 0; i < num; i++){
        printf("%d ", *p++);
    }
}
int main()
{
    int i = 1, j = 2, k = 3;
    fun(3, i, j, k);
    return 0;
}
//程序执行结果为：1 2 3
```

3 数组

3.1 数组下标

```c
int a[2][2] = {{1}, {2,3}};
//按行初始化，未赋值的元素自动赋值为0，故a[0][1]
```

3.2 数组指针

a为数组，(int *)(&a+1)表示什么意思?

```c
int main()
{
    int a[5] = {1,2,3,4,5};
    int *p = (int *)(&a+1);
    printf("%d", *(p-1));
    return 0;
}
//地址：a=635628; &a=6356728; p=6356748
//程序输出结果：5
```

注意：数组名 a 代表数组首元素的地址，但 &a 是数组指针，指向含有5个整型元素的数组的指针，故 &a+1 的地址是&a的地址再加上5*4=20字节的偏移量，即p指向了a[4]的下一个元素。

a 和 &a 的地址是一样的，但含义不一样，a是数组首元素的地址，也就是a[0]的地址；&a是 对象（数组）首地址；a+1是数组下一元素的地址，即a[1]；而&a+1是下一个对象的地址， 即 a[5]。

3.3 二维数组指针

```
#include <stdio.h>
void f(char**p){
      *p +=2;  //*p即二维数组a的首行首元素的地址
}
int main()
{
    char *a[] = {"123","abc","456"}, **p;
    p = a;
    f(p);
    printf("%s\r\n",*p);  //输出结果: 3
}
```

4 字符串

4.1 strcat函数

连接两个字符串，把src连接到dest之后，返回dest的地址。

```
char *strcat(char *dest, char *src){
    char *str = dest;  //保存字符串首地址
    if(!dest || !src) return dest;
    while(*dest)
        dest++;
    while(*dest++ = *src++)
        ;
    return str;
}
```

4.2 strcmp函数

比较两个字符串：

- 当str1指向的字符串大于str2指向的字符串时，返回正数。
- 当str1指向的字符串等于str2指向的字符串时，返回0。
- 当str1指向的字符串小于str2指向的字符串时，返回负数。

```
int strcmp(char *str1, char *str2){
    if(!str1 && !str2) return 0;
    //注意不可用 while(*str1++==*str2++)
    while(*str1 == *str2){
        if(*str1 == '\0') return 0;
        str1++;
        str2++;
    }
    return *str1 - *str2;
}
```

4.3 strcpy函数

拷贝字符串，把src所指向的内容拷贝到dest。

```c
char *strcpy(char *dest, char *src){
    char *str = dest;
    if(!dest || !src) return dest;
    //会把src结尾的'\0'也拷贝到dest中，等价于while((*dest++ = *src) != '\0')
    while(*dest++ = *src++)
        ;
    return str;
}
```

4.4 strlen函数

返回字符串str的长度（不包括'\0'）。

```c
int strlen(char *str){
    int count = 0;
    if(!str) return 0;
    while(*str){
        str++;
        count++;
    }
    return count;
}
```

4.5 strchr函数

查找str中首次出现c的位置(指针)，如有有，则返回出现位置，否则返回NULL。

```c
char *strchr(char *str, char c){
    if(!str) return NULL;
    while(*str != c && *str)
        str++;
    return ((*str == c) ? str : NULL);
}
```

4.6 strstr函数

查找字符串str2在str1中出现的位置，找到则返回位置，否则返回NULL。

```c
char *strstr(char *str1, char *str2){

    if(!str1 || !str2) return NULL;
    while(*str1){
        char *s1 = str1, *s2 = str2;
        if(*s1 == *s2){
            while(*s1 == *s2 && *s1 && *s2){
                s1++;
                s2++;
            }
            if(*s2 == '\0') return str1;
        }
        str1++;
    }
    return NULL;
}
```