20160306学习笔记

1.while（先判断） ， do while（后判断）

> 循环体中的语句进来时都要做一次时用do while，且while后加分号。
> 二者处理同一问题时，若循环体部分一样，结果一样。但如果while后面一开始为零，结果则不同。

```
void main(){
    int sum,i;
    sum = 0;
    i = 1;
    do{
    sum = sum + i;
    i++;
    }while(i<=100);
    printf("sum is %d\n",sum);//5050
}
```

2.for语句

> for语句可以省略表达式1，但分号不能省。

```
void main(){
    int sum,i;
    for(sum=0,i =1;i<=100;i++){
            sum += i;
    }
    printf("sum is %d\n",sum);//5050
}
```

3.continue和break

> continue语句结束本次循环。
> break语句提前结束循环。

```
void main(){
    int sum,i;
    for(sum=0,i =1;i<=100;i++){
           if (i%2==1) {
                 sum += i;
            }else{
                   continue;
            }
    }
    printf("sum is %d\n",sum);//2500
}
```

```
void main(){
    int sum = 0,i = 1;
    for(;;;){
           if(i<=100){
                if (i%2==1) {
                     sum += i;
                     i ++;
                }else{
                        i++;
                      continue;
                }
         }else{
             break;
         }
    }
    printf("sum is %d\n",sum);//2500
}
```

数组

1.一维数组

> 不能用变量说明数组的大小

```
void main(){
    int a[10];
    int i;
    for(i=0;i<10;i++){
        a[i] = i+1;
    }
    //数组在内存中连续存放，相差四个字节
    //监视时a+1可偏移到a[1]，类型为int *，*(a+1)取其的值
}
```

数组初始化
可以全部输入 或初始化为零
int a[10] = {0}
char a[10] = {0}(存储的是\0)

2.一维字符数组

> 可用%s直接打印数组，或者用循环逐个打印
> 注意区分'\0'(ASCII 0)和'0'(ASCII 48)

stdin 输入缓冲区
stdout 输出缓冲区

```c
void main(){
    char a[10] = "hello";
    char b[] = "hello";
    printf("c is %s\n",a);
    //a[5]为'\0'
    //sizeof(c) = 6
}
```

201c60307 下午

scanf(“%c”,c) 会匹配\n
scanf(“%s”,c) 值传递

```c
void main(){
    char c[10];
    scanf("%s",c);
    printf("c is %s\n",c);
    //输入hello, c[5]为'\0'
}
```

scanf(%c%c) 不使用这种形式

字符串处理函数

- puts(字符数组) 将一个字符转输出到终端
- gets(字符数组) 从终端讲一个字符串到字符数组

```c
void main(){
    char c[10];
    gets(c);
    //printf("c is %s\n",c);//和scanf作用一样
    puts(c);
}
```

- strlen(字符数组) 测试字符串长度

```c
#include <string.h>
void main(){
    char c[10];
    gets(c);
    printf("%d\n",strlen(c));//输入hello打印5
}
```

- strcpy(字符数组1，字符串2，数字（可省略）) 将字符串2复制到字符数组1中

```c
#include <string.h>
void main(){
    char c[10];
    char c1[10];
    gets(c);
    strcpy(c1,c);//复制
    printf("%s\n",c1);
}
```

一维数组长度传不进来

```c
#include <string.h>
void main(){
    char c[10];
    gets(c);
    printf("c size is %d\n",sizeof(c));//10
    cal_array(c);
}
void cal_array(char c[]){
    printf("c size is %d\n",sizeof(c));//4,c显示的是char *, 一个指针4个字节
}
```

- strcmp(字符串，字符串2) 比较字符串是否相等

```c
#include <string.h>
void main(){
    char c1[10] = “hello”;
    char c2[10] = “hi”;
    printf("%d\n",strcmp(c1,c2));//e<i 所以返回-1    =返0     >返1
}
```

- strcat(字符数组1，字符数组2) 连接两个字符数组中的字符串

```c
#include <string.h>
void main(){
    char c1[10] = "hello";
    char c2[10] = " hi";
    strcat(c1,c2);//拼接
    printf("%s\n",c1);//hello hi
}
```

内存操作函数

- memcpy 内存拷贝 主要用于非字符的情况
  void *memcpy(void* to, const void *from, size_t count);

```c
#include <string.h>
void main(){
    int c1[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
    int c2[12];
    memcpy(c2,c1,sizeof(c1));//将c1的内容拷贝到c2中
}
```

- memset 将一段内存初始值化为某值
  void *memset(void* buffer, int ch, size_t count);

```c
#include <string.h>
void main(){
    int c1[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
    int c2[12];
    char c[20] = {0};
    while(gets(c)!=NULL){//用gets时不用EOF用NULL
        puts(c);
        memset(c,'\0',sizeof(c));//输入hello，打印hello，之后清零
    }
}
```

- memcmp 比较buffer12的前count个字符 int memcmp(const void *buffer1, const void* buffer2, size_t count);
- memmove 与memcpy相同，但to和from重叠仍能正常工作 void *memmove(void* to, const void *from, size_t count);

3.二维数组

> 在内存中的存放顺序是按行存放

```c
void main(){
    int a[3][4] = {{11,12,13,14},{21,22,23,24},{31,32,33,34}};
    //在监视窗口中    输入a    类型int[3][4]    点开有三个一维数组  类型int[4]
    //输入&a    类型int[3][4]*    地址连续按行存放
    char b[5][10] = {0};
    char d[10] = {0};
    int i = 0;
    while(gets(d)!=NULL){
           strcpy(c[i],d);
            i++;
            memset(d,0,sizeof(d));
    }
    for(i=0;i<5;i++){
        printf("%s\n",c[i]);
    }
}
```

int a[3][4]

*(a+1) -> a[1]

*(a[1]+1) -> a[1][1]