2020九月暑期复习题

笔记本: 我的第一个笔记本
创建时间: 2020/9/9 20:09 **更新时间:** 2020/9/11 21:08
作者: 1272209351@qq.com

练习题:
1、32位平台下 strlen ------ 遇到 \0 结束
sizeof ------ 大小

(1)

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

int func( int a[] )
{
    //如果函数func参数传递数组Array,那么sizeof(a)呢?
    printf("%d\n", sizeof(a));
}

void main()
{
    char str[] = "Welcome to Bit";
    int Array[] = {1,2,3,4,5,6,7,8,9};
    char *p = str;
    int  n;
    printf("len = %d\n",strlen(str));
    printf("size = %d\n",sizeof(str));
    printf("%d\n",sizeof(Array));
    printf("%d", sizeof(p));
    printf("%d\n",sizeof(n));

    func(Array);

}
```

结果:
14 (遇到 \0 结束)
15 4*9=36 4 4 4

(2)

```
char str1[] = "HelloBit";
printf("len = %d\n", strlen(str1));
printf("size = %d\n", sizeof(str1));

char str2[10] = "HelloBit";
printf("len = %d\n",  strlen(str2)); //
printf("size = %d\n", sizeof(str2)); //
```

结果:
8 9 8 10

(3)

```
char str3[10] ={'H','e', 'l', 'l', 'o', 'B','i','t'};
printf("len = %d\n",  strlen(str3)); //
printf("size = %d\n", sizeof(str3)); //
```

结果:

8 10

(4)

```
char str4[] ={'H','e', 'l', 'l', 'o', 'B','i','t'};
printf("len = %d\n",  strlen(str4)); //
printf("size = %d\n", sizeof(str4)); //8
```

```
char str5[10];
for(int i=0; i<5; ++i)
    str5[i] = 'a' + i;
printf("len = %d\n",  strlen(str5)); //
printf("size = %d\n", sizeof(str5)); //
```

结果:

~~24 (随机，未完全初始化后边赋值 \0)~~ 8

~~44 (随机，因为未进行初始化找不到 \0)~~ 10

全局数据 && 局部数据 :

未进行初始化时，系统给出的初始化不一样

全局数据初始化为 \0

局部数据初始化为 随机值

(5)

```
char str7[10];
for(int i=0; i<5; ++i)
    str7[i] = '0';
printf("len = %d\n",  strlen(str7)); //随机
printf("size = %d\n", sizeof(str7)); //10
```

字符 '0' 不等同于 '\0'

'\0' === '0' ；反斜杠起到转义的作用

char ch='\5' =====存放的是 5

char ch='5' ======存放的是53 （ '0' 字符ASII 码值为 0，5---ASII 值为53)

(6)

```
short *par[10][10]; //数组
printf("size = %d\n", sizeof(par));  //400
```

数组类型，10*10=100个元素，short* ==4 字节

2、运行是否出问题？指出问题 / 循环次数?

2、请观察下列程序运行时是否会出现问题，若有，请指出问题，若不存在问题，请指出程序循环的次数

```
#define MAX_SIZE 255
void main()
{
    unsigned char buff[MAX_SIZE +1];
    unsigned char i;
    for(i=0; i <= MAX_SIZE; i++)
    {
        buff[i] = i;
    }
}
```

有问题 --- 死循环
原因：变量 i 为无符号char 类型 [0 ,255] 总小于 MAX_SIZE+1

3、添加下面粗斜体部分代码的初衷是为了给 gui show image 这句代码加上限制条件，请问这样修改有什么隐患？该如何修改？

```
        .....................
        gui_push_clip();
    #ifdef AAA
        if (show_status == MMI_TRUE)
    #endif
    #ifdef BBB
        gui_show_image(x , y , image_id);
    #endif

        gui_pop_clip();
        update_dt_display();
        .....................
```

修改方案：

```
#ifdef BBB
    #ifdef AAA
        if (show_status == MMI_TRUE)
    #endif
    gui_show_image(x , y , image_id);
#endif

    gui_pop_clip();
    update_dt_display();
    .....................
```

```
+     ..................
      gui_push_clip();
  #ifdef AAA
      if (show_status == MMI_TRUE)
  #endif
  #ifdef BBB
      gui_show_image(x , y , image_id);
  #endif
      ;
      gui_pop_clip();
      update_dt_display();
      ..................
```

或者加个；号　（不建议）

4、请问代码运行问题?

（1）

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

void GetMemory(char *p)
{
    p = (char*)malloc(57);
}
void main()
{
    char *str = NULL;
    GetMemory(str);
    strcpy(str, "HelloBit");
    printf(str);
}
```

形参改变不能改变实参，开辟的空间无法使用 ---- 用二级指针

（2）

```
char* GetMemory(void)
{
        char p[] = "HelloBit";
        return p;
}
void main()
{
        char *str = NULL;
        str = GetMemory();
        printf(str);
}
```

p 是个临时数组，使用完之后会被释放

（3）

```
void GetMemory(char **p)
{
        *p = (char *)malloc(57);
}
void main()
{
        char *str = NULL;
        GetMemory(&str);
        strcpy(str,"HelloBIt");
        printf(str);
}
```

申请成功判断，内存释放
改进：

```
void GetMemory(char **p)
{
        *p = (char *)malloc(57);
        assert(*p != NULL);
}
void main()
{
        char *str = NULL;
        GetMemory(&str);
        strcpy(str,"HelloBIt");
        printf(str);

        free(str);
}
```

(4)

```
void main()
{
    char *str = (char *)malloc(57);
    strcpy(str, "Hello");
    free(str);
    if (str != NULL)
    {
        strcpy(str, "C++");
        printf(str);
    }
}
```

str 被释放，成为野指针，空间不能被正常使用因此不可以将‘C++’写入
改进：

```
void main()
{
    char *str = (char *)malloc(57);
    strcpy(str, "Hello");
    free(str);
    str = NULL; //预防野指针
    if (str != NULL)
    {
        strcpy(str, "C++");
        printf(str);
    }
}
```

5、编程题

5、请编码实现以下功能的函数

功能：实现对一个 8 bit 数据（unsigned char 类型）的指定位（例如第 n 位）的置 0 或者置 1 操作，并保持其他位不变。

函数原型：

void bit_set(unsigned char *p_data, unsigned char position, bool flag)

函数参数说明：

P_data 是指定的源数据，position 是指定位（取值范围 1~8）；flag 表示是置 0 还是置 1 操作，true：置 1 flase:置 0

解答：

```
void bit_set(unsigned char *p_data, unsigned char position, bool flag)
{
    if(flag)  //1
    {
        *p_data |= (0x01<<(position-1));
    }
    else   //0
    {
        *p_data &= ~(0x01<<(position-1));
    }
}
void main()
{
    unsigned char data = 123; //0 ~255  //0111 1011   double
    unsigned char pos = 3;
    bool flag = true;

    bit_set(&data, pos, flag);
```

如果未告知取值要考虑边界条件：

```
void bit_set(unsigned char *p_data, unsigned char position, bool flag)
{
    assert(p_data != NULL || (position>=1 && position<=8));

    if(flag)  //1
    {
        *p_data |= (0x01<<(position-1));
    }
    else      //0
    {
        *p_data &= ~(0x01<<(position-1));
    }
```

6、

6、请实现字符串右循环移位函数，比如：“abcdefghi”循环右移 2 位就是“hiabcdefg”。

函数原型：void RightLoopMove(char *pStr, unsigned short steps)

函数参数说明：

pStr: Point to a '\0' terminated string

steps: The rotate shift numbers

解答：

法一：将字符串划分为两部分，前边部分拼接到后边字符之后

```
void RightLoopMove(char *pStr, unsigned short steps)
{
    assert(pStr != NULL && *pStr!='\0');
    int len = strlen(pStr);
    char *tmp = (char*)malloc(sizeof(char) * (len+1)); //空间复杂度0(n)
    assert(tmp != NULL);

    steps %= len;

    strcpy(tmp, pStr+(len-steps));
    strncat(tmp, pStr, len-steps);

    strncpy(pStr, tmp, len);
    free(tmp);
    tmp = NULL;
}
void main()
{
    char str[] = "abcdefghi";
    printf("str = %s\n", str);
    RightLoopMove(str, 20);
    printf("str = %s\n", str);
}
```

法二：将最后一个字符提前保存在临时空间变量，其余字符整体后移

```
void RightLoopMove(char *pStr, unsigned short steps)
{
    assert(pStr != NULL && *pStr!='\0');
    int len = strlen(pStr);
    steps %= len;
    for(int i=0; i<steps; ++i)
    {
        char tmp = pStr[len-1];              //空间复杂度0(1)
        for(int end=len-1; end>0; --end)     //时间复杂度0(n^2)
            pStr[end] = pStr[end-1];
        pStr[0] = tmp;
    }
}
void main()
{
    char str[] = "abcdefghi";
    printf("str = %s\n", str);
    RightLoopMove(str, 40);
    printf("str = %s\n", str);
}
```

华为：

1、

1、给出以下定义：

```
char acX[] = "abcdefg";
char acY[] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g' };
```

则正确的叙述为（ ）

A）数组 acX 和数组 acY 等价　　B）数组 acX 和数组 acY 的长度相同

C）数组 acX 的长度大于数组 acY 的长度　　D）数组 acX 的长度小于数组 acY 的长度

strlen (acX) --- 包含 \0
strlen (acY) ---- 没有 \0

4、设有如下定义：

```
unsigned long pulArray[] = {6, 7, 8, 9, 10};
unsigned long *pulPtr;
```

则下列程序段的输出结果为（ C ）

```
pulPtr = pulArray;
*(pulPtr + 3) += 3;
printf("%d ,%d\n",*pulPtr, *(pulPtr + 3));
```

6　　12

4、

exten 引用外部变量

（32 位小字节序处理器）

8、

14、

```
int main()
{
    char c;                 //-128 ~ 127
    unsigned char uc;       //0 ~ 255
    unsigned short us;      //0 ~ 65535

    c = 128;
    uc = 128;
    us = c + uc;
    printf("0x%x\n", us);   //?0x0

    us = (unsigned char)c + uc; //128 + 128
    printf("0x%x\n", us);   //?0x100

    us = c + (char)uc;      //-128 + -128 = -256
    printf("0x%x\n", us);   //?0xff00
    return 0;
}
```

```
#pragma pack(4)

unsigned short *pucCharArray[10][10];
typedef union unRec
{
    unsigned long ullndex;      //4
    unsigned short usLevel[7]; //14 + 2
    unsigned char ucPos;        //1
}REC_S;
REC_S stMax, *pstMax;

void main()
{
    printf("%d\n", sizeof(pucCharArray));  //400
    printf("%d\n", sizeof(stMax));         //16
    printf("%d\n", sizeof(pstMax));        //4
    printf("%d\n", sizeof(*pstMax));       //16
}
```

填空

5、

```
#pragma pack(1)

struct tagAAA
{
    unsigned char ucId : 1;
    unsigned char ucPara0 : 2;  //1
    unsigned char ucState : 6;  //1
    unsigned char ucTail : 4;   //1
    unsigned char ucAvail;      //1
    unsigned char ucTail2 : 4;  //1
    unsigned char ucData;       //1
}AAA_S;

void main()
{
    //问：AAA_S在分别为1字节对齐和四字节对齐的情况下，占用的空间大小是：_____
    printf("%d\n",sizeof(AAA_S));
}
```

7、

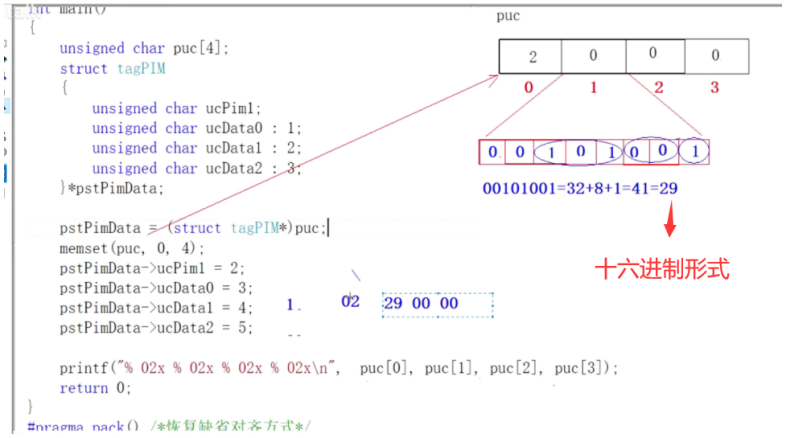
```
int fun(int x, int y)
{
    static int m = 0;
    static int i = 2;
    i += m + 1;
    m = i + x + y;
    return m;
}
void main()
{
    int j = 4;
    int m = 1;
    int k;
    k = fun(j, m);
    printf("%d\n", k);  // 8

    k = fun(j, m);
    printf("%d\n", k);  // 17
    return;
}
```

i= i+m+1=2+0+1=3　　i=3+8+1=12

m=3+4+1=8　　m=12+4+1=17

静态变量 记忆性

10、注意位域


int main()
{
unsigned char puc[4];
struct tagPIM
{
unsigned char ucPim1;
unsigned char ucData0 : 1;
unsigned char ucData1 : 2;
unsigned char ucData2 : 3;
}*pstPimData;
pstPimData = (struct tagPIM*)puc;
memset(puc, 0, 4);
pstPimData->ucPim1 = 2;
pstPimData->ucData0 = 3;
pstPimData->ucData1 = 4;
pstPimData->ucData2 = 5;
printf("% 02x % 02x % 02x % 02x\n", puc[0], puc[1], puc[2], puc[3]);
return 0;
}
#pragma pack() /*恢复缺省对齐方式*/
puc
2 0 0 0
0 1 2 3
0 0 1 0 1 0 0 1
00101001=32+8+1=41=29
十六进制形式
02
29 00 00