

### 知识点1【函数指针 作为 函数的形参】

### 知识点2【malloc函数 和 free函数】

malloc函数

free函数

案例1：从堆区申请 一个int空间

案例2：从堆区申请一个数组 数组的大小 由用户决定

案例3：从堆区申请一个数组 数组的大小 由用户决定（函数版本）

### 知识点3【calloc函数】

案例：

### 知识点4【realloc动态追加或减少空间】

### 知识点5【堆区空间使用的注意事项】

### 知识点6【防止多次释放】

### 知识点7【字符串处理函数】

1、strlen测量字符串长度

作业：自定义一个my\_strlen函数测量字符串长度（不能在函数里面使用strlen）

2、strcpy strncpy字符串拷贝(重要)

strcpy

实现strcpy

strncpy

3、strcat字符串的拼接

strncat拼接前n个

作业：自定义my\_strcat函数 不许使用 strcat

4、strcmp字符串的比较（整个字符串的比较）

strncmp 字符串局部比较

5、strchr字符查找函数

6、strstr字符串查找

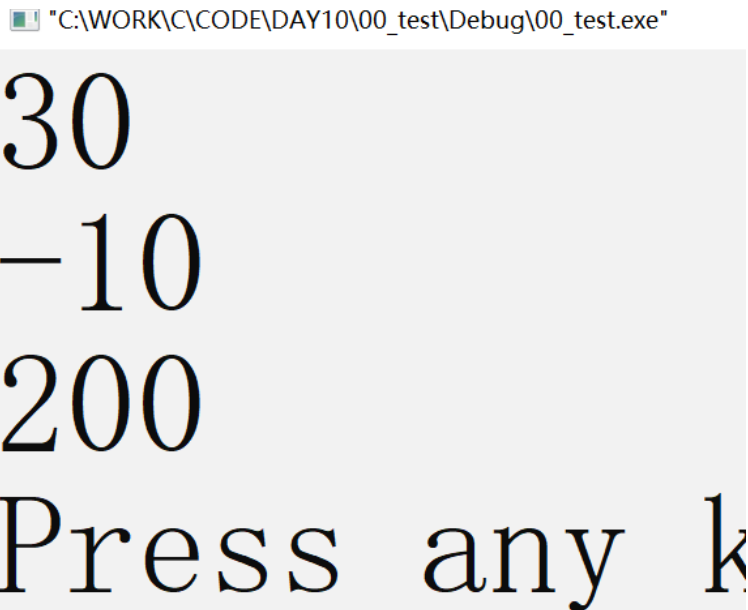
## 知识点1 【函数指针 作为 函数的形参】

案例1：

```
1  #include<stdio.h>
2  int my_add(int a,int b)
3  {
4      return a+b;
5  }
6  int my_sub(int a,int b)
7  {
8      return a-b;
9  }
10 int my_mul(int a,int b)
11 {
12     return a*b;
13 }
14
15 //定义一个函数 实现上述函数的功能
16
17 int my_calc(int a,int b, int (*fun_pointer)(int,int) )
18 {
19     return fun_pointer(a,b);
20 }
21 void test01()
22 {
23     printf("%d\n",my_calc(10,20,my_add));
24     printf("%d\n",my_calc(10,20,my_sub));
25     printf("%d\n",my_calc(10,20,my_mul));
26
27 }
28 int main(int argc,char *argv[])
29 {
30     test01();
```

```
31 return 0;
32 }
```

运行结果：



```
"C:\WORK\C\CODE\DAY10\00_test\Debug\00_test.exe"
30
-10
200
Press any k
```

## 知识点2【malloc函数 和 free函数】

### malloc函数

```
1 #include<stdlib.h>
2 void *malloc(unsigned int num_size);
3 形参：num_size需要申请空间大小的字节数。
4 返回值：
5 成功：返回空间的起始地址
6 失败：NULL
7 特点：
8 1、对于malloc的返回值 一般要强制类型转换
9 2、malloc申请的空间 内容不确定 一般使用memset进行清空
10 3、多次调用malloc 第1次malloc 和 第2次malloc的地址不一定连续
```

### free函数

```
1 void free(void *addr);
2 释放堆区空间
```

### 案例1：从堆区申请 一个int空间

```
1 void test02()
```

```

2 {
3     int *addr = NULL;
4
5     addr = (int *)malloc(sizeof(int));
6     if(addr == NULL)//申请失败了
7     {
8         printf("malloc err\n");
9         return;
10    }
11
12    printf("*addr=%d\n", *addr);//不确定的值
13
14    //对堆区空间 清0
15    memset(addr, 0, sizeof(int));
16    printf("*addr=%d\n", *addr);//0
17
18    //对addr的空间 就行写 或 读
19    *addr = 1000;//写
20
21    printf("*addr=%d\n", *addr);//1000 读
22
23
24    //释放堆区空间 空间使用权限的回收 是否对空间内容清0 这是不确定的
25    free(addr);
26 }
27 int main(int argc, char *argv[])
28 {
29     test02();
30     return 0;
31 }

```

运行结果：

```

*addr=-842150451
*addr=0
*addr=1000
Press any key to continue

```

## 案例2：从堆区申请一个数组 数组的大小 由用户决定

1 1、从键盘获取 用户要申请的数组大小

- 2 2、根据大小 从堆区申请空间
- 3 3、对空间的读写操作
- 4 4、释放该空间

```
1 void test03()
2 {
3     int n = 0;
4     int i=0;
5     int *arr=NULL;
6
7     //1、获取用户大小
8     printf("请输入元素的个数:");
9     scanf("%d", &n);
10
11     //2、根据大小从堆区申请空间
12     arr = (int *)malloc(n*sizeof(int));
13     if(NULL == arr)
14     {
15         //perror 错误输出
16         perror("mallac");
17         return;
18     }
19
20     //对arr的读写操作
21     printf("请输入%d个int数据\n",n);
22     for(i=0;i<n; i++)
23     {
24         scanf("%d", arr+i);
25     }
26
27     //遍历数组
28     for(i=0;i<n;i++)
29     {
30         printf("%d ", arr[i]);
31     }
32     //释放空间
33     free(arr);
34 }
35 int main(int argc,char *argv[])
36 {
```

```
37 test03();
38 return 0;
39 }
```

### 案例3：从堆区申请一个数组 数组的大小 由用户决定（函数版本）

```
1 int get_n(void)
2 {
3     int n = 0;
4     printf("请输入元素的个数:");
5     scanf("%d", &n);
6     return n;
7 }
8 int* get_addr(int n)
9 {
10    return (int *)malloc(n*sizeof(int));
11 }
12 void my_input_array(int *arr, int n)
13 {
14     int i=0;
15     //记得将arr指向的空间清0
16     memset(arr,0,n*sizeof(int));
17
18     //获取键盘输入
19     printf("请输入%d个int数据\n",n);
20
21     for(i=0;i<n; i++)
22     {
23         scanf("%d", arr+i);
24     }
25 }
26 void my_print_array(int *arr, int n)
27 {
28     int i=0;
29     for(i=0;i<n;i++)
30     {
31         printf("%d ", arr[i]);
32     }
33     printf("\n");
34 }
```

```

35 void test04()
36 {
37     int *arr=NULL;
38     int n = 0;
39
40     //得到用户输入的元素个数
41     //1、获取用户大小
42     n = get_n();
43
44     //定义函数 给arr申请堆区空间
45     arr = get_addr(n);
46     if(arr == NULL)
47     {
48         perror("get_addr");
49         return;
50     }
51
52     //对空间读写操作
53     my_input_array(arr, n);
54
55
56     //对空间数组遍历
57     my_print_array(arr, n);
58
59     //释放空间
60     free(arr);
61
62 }
63 int main(int argc,char *argv[])
64 {
65     test04();
66     return 0;
67 }

```

运行结果：

```
请输入元素的个数:5
请输入5个int数据
1 2 3 4 5
1 2 3 4 5
Press any key to continue_
```

### 知识点3 【calloc函数】

```
1 #include<stdlib.h>
2 void * calloc(size_t nmemb,size_t size);
3 参数:
4 1、nmemb 申请的数据块数
5 2、size 每一块大小
6 3、所以申请总大小=nmemb * size
7 返回值:
8 成功: 返回空间的起始地址
9 失败: 返回NULL
10 特点: 申请的空间自动清零
```

#### 案例:

```
1 void test04()
2 {
3     int n = 0;
4     int i=0;
5     int *arr=NULL;
6
7     //1、获取用户大小
8     printf("请输入元素的个数:");
9     scanf("%d", &n);
10
11     //2、根据大小从堆区申请空间
12     #if 0
13     arr = (int *)malloc(n*sizeof(int));
14     if(NULL == arr)
15     {
16         //perror 错误输出
```



```

17  perror("malloc");
18  return;
19  }
20  memset(arr,0,n*sizeof(int));//清零
21  #endif
22
23  #if 1
24  arr=(int *)calloc(n, sizeof(int));//自动清零 不需要使用memset
25  if(NULL == arr)
26  {
27  //perror 错误输出
28  perror("calloc");
29  return;
30  }
31  #endif
32  //对arr的读写操作
33  printf("请输入%d个int数据\n",n);
34  for(i=0;i<n; i++)
35  {
36  scanf("%d", arr+i);
37  }
38
39  //遍历数组
40  for(i=0;i<n;i++)
41  {
42  printf("%d ", arr[i]);
43  }
44
45  //释放空间
46  free(arr);
47
48  }

```

## 知识点4 【realloc动态追加或减少空间】

```

1  #include<stdlib.h>
2  void* realloc(void *s, unsigned int newsize);

```

功能：

在原先s指向的内存基础上重新申请内存，新的内存的大小为 new\_size 个 字节，如果原先内存后面有足够大的空间，就追加，如果后边的内存不够用，则realloc函数会在堆区找一个newsize个字节大小的内存申请，将原先内存中的内容拷贝过来，然后释放原先的内存，最后返回新内存的地址。

参数：s：原先开辟内存的首地址    newsize：新申请的空间的大小

返回值：新申请的内存的首地址

**注意：一定要保存 realloc的返回值**

```
1 void test06()
2 {
3     int *arr = NULL;
4     int n = 0;
5     int i=0;
6     int n_new = 0;
7
8     //1、获取用户大小
9     printf("请输入元素的个数:");
10    scanf("%d", &n);
11
12    arr=(int *)calloc(n, sizeof(int)); //自动清零 不需要使用memset
13    if(NULL == arr)
14    {
15        //perror 错误输出
16        perror("calloc");
17        return;
18    }
19
20    printf("请输入%d个int数据\n",n);
21    for(i=0;i<n; i++)
22    {
23        scanf("%d", arr+i);
24    }
25
26    //遍历数组
27    for(i=0;i<n;i++)
28    {
29        printf("%d ", arr[i]);
30    }
31
32    //再追加5个元素
```

```

33  printf("请输入新增的元素个数:");
34  scanf("%d", &n_new);
35  arr = (int *)realloc(arr, (n+n_new)*sizeof(int));
36
37  printf("请输入新增的%d个int数据\n", n_new);
38  for(i=n; i<(n+n_new); i++)
39  {
40      scanf("%d", arr+i);
41  }
42
43  for(i=0; i<(n+n_new); i++)
44  {
45      printf("%d ", arr[i]);
46  }
47  printf("\n");
48
49  free(arr);
50  }
51  int main(int argc, char *argv[])
52  {
53      test06();
54      return 0;
55  }

```

运行结果:

```

请输入元素的个数:5
请输入5个int数据
10 20 30 40 50
10 20 30 40 50 请输入新增的元素个数:3
请输入新增的3个int数据
60 70 80
10 20 30 40 50 60 70 80
Press any key to continue_

```

## 知识点5 【堆区空间使用的注意事项】

```

1  void test08()
2  {
3      int *p2 = NULL;

```

```

4  int *p3 = NULL;
5  //1、 指向堆区空间的指针变量 不要随意的更改指向
6  int *p=(int *)calloc(1,sizeof(int));
7  int num = 10;
8  p = &num;//p指向num 导致calloc申请的空间泄露
9
10 //2、 不要操作已经释放的空间
11 p2 = (int *)calloc(1,sizeof(int));
12 *p2 = 1000;
13 //释放该空间
14 free(p2);
15 printf("*p2 = %d\n", *p2);//不确定
16
17 //3、 不要对堆区空间重复释放
18 p3 = (int *)calloc(1,sizeof(int));
19 free(p3);
20 free(p3);//多次释放
21 }

```

## 知识点6 【防止多次释放】

```

1  void test09()
2  {
3  int *p = (int *)calloc(1,sizeof(int));
4
5  if(p != NULL)//防止多次释放
6  {
7  free(p);
8  p=NULL;
9  }
10
11
12 if(p != NULL)
13 {
14 free(p);
15 p=NULL;
16 }
17 }

```

## 知识点7 【字符串处理函数】

只要是以str开头的函数 都是遇到'\0'结束

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

## 1、strlen测量字符串长度

```
1  size_t  strlen(const char *s)
2  //s:被测量的字符串首元素地址
3  //返回值为 字符串的长度 不包含'\0'
4  //const char *s strlen函数 不会通过s修改s指向的空间内容
```

案例:

```
1  void test01()
2  {
3      char buf1[128]="hehehe";
4      char buf2[]="hehehe";
5      char buf3[]="hehe\0hehe";
6
7      // \123 代表一个字符 \hhh 八进制转义h: 0~7
8      //\\表示'\ '
9      char buf4[]="hehe\123\\he";
10
11     //\\x2f表示一个字符 \xdd 十六进制转义 d:0~9 a~f
12     char buf5[]="hehe\x2fhe";
13
14
15     printf("%d\n",sizeof(buf1));//126
16     printf("%d\n",strlen(buf1));//6
17
18     printf("%d\n",sizeof(buf2));//7
19     printf("%d\n",strlen(buf2));//6
20
21     printf("%d\n",sizeof(buf3));//8
22     printf("%d\n",strlen(buf3));//4
23
24     printf("%d\n",sizeof(buf4));//9
25     printf("%d\n",strlen(buf4));//8
26     printf("%s\n",buf4);
27
28     printf("%d\n",sizeof(buf5));//8
29     printf("%d\n",strlen(buf5));//7
30
31     char buf6[]="\0hehe\0hehe";
32     printf("%d\n",strlen(buf6));//0
33     return;
```

**作业：自定义一个my\_strlen函数测量字符串长度（不能在函数里面使用strlen）**

```
1 int my_strlen(const char *s);
```

## 2、strcpy strncpy字符串拷贝(重要)

### strcpy

原型:char \*strcpy( char \*dest, const char \*src )

功能：把src所指向的字符串复制到dest所指向的空间中

返回值：返回dest字符串的首地址

注意：遇到'\0'会结束，只是'\0'也会拷贝过去

```
1 void test02()
2 {
3     char src[]="hello string";
4     //保证dst足够大
5     char dst[128]="";
6
7     strcpy(dst,src);
8
9     printf("dst=%s\n",dst); //"hello string"
10 }
```

```
1 void test02()
2 {
3     char src[]="hello\0string";
4     //保证dst足够大
5     char dst[128]="";
6
7
8     strcpy(dst,src);
9
10    printf("dst=%s\n",dst); //"hello"
11 }
```

### 实现strcpy

```

1 char *my_strcpy(char *dst,const char *src)
2 {
3     #if 0
4     do
5     {
6         *dst = *src;
7         dst++;
8         src++;
9     }while(*src != '\0');
10    *dst='\0';
11    #endif
12    while(*dst++ = *src++);
13 }
14 void test02()
15 {
16     char src[]="hello\0string";
17     //保证dst足够大
18     char dst[128]="";
19     my_strcpy(dst,src);
20
21     printf("dst=%s\n",dst); //"hello"
22 }

```

## strncpy

原型:char \*strncpy( char \*dest, const char \*src,int num)

功能：把src指向字符串的前num个复制到dest所指向的空间中

返回值：返回dest字符串的首地址

注意：'\0'不拷贝

```

1 void test03()
2 {
3     char src[]="hello string";
4     //保证dst足够大
5     char dst[128]="";
6

```

```

7  strncpy(dst,src,3);
8  printf("dst=%s\n",dst);//"hel"
9  }

```

```

1  void test03()
2  {
3  char src[]="hello";
4  //保证dst足够大
5  char dst[128]="";
6
7  strncpy(dst,src,30);
8  printf("dst=%s\n",dst);//"hello"
9  }

```

### 3、strcat字符串的拼接

```

1  char *strcat(char *dest, const char *src);
2  将src的字符串 拼接到 dst的末尾（dst第一个'\0'的位置）

```

```

1  void test04()
2  {
3  char src[]="world";
4  char dst[128]="hello";
5
6  strcat(dst,src);
7
8  printf("%s\n",dst);//"helloworld"
9  }

```

```

1  void test04()
2  {
3  char src[]="wor\0ld";
4  char dst[128]="hello";
5
6  strcat(dst,src);
7
8  printf("%s\n",dst);//"hellowor"
9  }

```

### strncat拼接前n个

```

1  void test04()
2  {

```



```

3 char src[]="world";
4 char dst[128]="hello";
5
6 strncat(dst,src,2);
7
8 printf("%s\n",dst); //"hellowo"
9 }

```

## 作业：自定义my\_strcat函数 不许使用 strcat

```

1 char *my_strcat(char *dest, const char *src);

```

## 4、strcmp字符串的比较（整个字符串的比较）

```

1 int strcmp(const char *s1, const char *s2);
2 功能：将s1和s2指向的字符串 逐个字符比较
3 返回值：
4 >0: 表示s1 > s2
5 <0: 表示s1 < s2
6 ==0: 表示s1 == s2

```

```

1 void test05()
2 {
3 char s1[]="hehe haha";
4 char s2[]="hehe xixi";
5 char s3[]="hehe haha";
6
7 if(strcmp(s1,s2) >0 )
8 {
9 printf("s1 > s2\n");
10 }
11 else if(strcmp(s1,s2) <0 )
12 {
13 printf("s1 < s2\n");
14 }
15 else if(strcmp(s1,s2) == 0)
16 {
17 printf("s1 == s2\n");
18 }
19
20 if(strcmp(s1,s3) >0 )
21 {
22 printf("s1 > s3\n");

```

```

23  }
24  else if(strcmp(s1,s3) <0 )
25  {
26  printf("s1 < s3\n");
27  }
28  else if(strcmp(s1,s3) == 0)
29  {
30  printf("s1 == s3\n");
31  }
32  }

```

运行结果:

```

s1 < s2
s1 == s3
Press any key to continue

```

## strncmp 字符串局部比较

```

1  void test05()
2  {
3  char s1[]="hehe haha";
4  char s2[]="hehe xixi";
5
6
7  if(strncmp(s1,s2,4) >0 )
8  {
9  printf("s1 > s2\n");
10 }
11 else if(strncmp(s1,s2,4) <0 )
12 {
13 printf("s1 < s2\n");
14 }
15 else if(strncmp(s1,s2,4) == 0)
16 {
17 printf("s1 == s2\n");
18 }
19
20 }

```

运行结果:

```
s1 == s2
Press any key to continue
```

## 5、strchr字符查找函数

```
1 char *strchr(const char *s, int c);
2 //从字符串s中查找第一次c字符出现的地址
3 //没有找到 返回NULL
```

```
1 void test06()
2 {
3     char str[]="www.1000phone.com";
4     char *ret = NULL;
5
6     ret = strchr(str,'o');
7     if(ret != NULL)
8     {
9         *ret = '#';
10    }
11    printf("str=%s\n",str); //"www.1000ph#ne.com";
12 }
```

(了解)

```
1 void test06()
2 {
3     char str[]="www.1000phone.com";
4     char *ret = NULL;
5
6     while((ret = strchr(str,'o')) && (*ret = '#') );
7
8     printf("str=%s\n",str); //"www.1000ph#ne.c#m";
9 }
```

重要 啥都写 了解

## 6、strstr字符串查找

```
1 char *strstr(const char *s1, const char *s2);
2 //从s1中查找字符串s2 返回第一次s2出现的地址
3 //查找失败 返回NULL
```

```
1 void test07()
2 {
3     char s1[]="www.sex.777.sex.com";
4     char s2[]="sex";
5
6     char *ret = NULL;
7     ret = strstr(s1,s2);
8     if(ret == NULL)
9     {
10        return;
11    }
12    printf("%s\n",ret); //"sex.777.sex.com"
13 }
```

```
1 void test07()
2 {
3     char s1[]="www.sex.777.sex.com";
4     char s2[]="sex";
5
6     char *ret = NULL;
7
8     while(1)
9     {
10        ret = strstr(s1,s2);
11        if(ret == NULL)
12        {
13            break;
14        }
15        memset(ret,'#', 3);
16    }
17
18    printf("%s\n",s1); //"www.###.777.###.com"
19
20 }
```

