知识点1【sscanf高级用法】

1、%[a-z] 提取a-z的字符串

2、%[aBc] 提取 a B c

3、%[^abc] 只要不是a b c任何一个 都要

知识点2【const关键字】

1、const修饰变量 为只读

2、const int *p;

3、int * const p;

4、const int * const p;(*p 只读 p只读)

知识点3【结构体类型的定义形式】

知识点4【结构体变量的定义】

知识点5【结构体变量的初始化】（下一）

知识点6【结构体变量 获取 键盘输入】

知识点7【结构体变量 之间的赋值】

知识点8【结构体数组】

案例：定义一个结构体数组 获取键盘输入 求平均age

知识点9【冒泡排序】下二

知识点10【结构体数组排序】

知识点1【sscanf高级用法】

1、%[a-z] 提取a-z的字符串

```
1 void test01()
2 {
3  char buf[128]="";
```

```
 //%[]都是 按 字符串 提取
 sscanf("abcDefABC","%[a-z]",buf);
 printf("buf=%s\n", buf);//"abc"

 return;
}
```

2、%[aBc] 提取 a B c

```
void test02()
{
 char buf[128]="";

 sscanf("aaBBcEdef","%[aBc]",buf);
 printf("buf=%s\n", buf);//aaBBc
}
```

3、%[^abc] 只要不是a b c任何一个 都要

```
void test03()
{
 char buf[128]="";
 sscanf("ABCcABC","%[^abc]",buf);
 printf("buf=%s\n", buf);//ABC
}
```

案例:

```
void test04()
{
 char name[32]="";
 char addr[32]="";
 //sscanf("lianghe@1000phone.com","%[^@]%*1s%[^.]",name,addr );
 //sscanf("lianghe@1000phone.com","%[^@]%*c%[^.]",name,addr );
 sscanf("lianghe@1000phone.com","%[^@]@%[^.]",name,addr );
 printf("name=%s\n",name);//"lianghe"
 printf("addr=%s\n",addr);//"1000phone"
 }
```

案例:

```
void test05()
{
 int m1=0,s1=0;
 int m2=0,s2=0;
```

```
5   char song[128]="";
6
7   char msg[128]="[12:13.46][8:23.45]给我一个威武雄壮的汉子";
8
9   sscanf(msg,"[%d:%d%.%*d][%d:%d.%*d]%s",&m1,&s1,&m2,&s2,song);
10   printf("%d-%d\n",m1,s1);
11   printf("%d-%d\n",m2,s2);
12   printf("%s\n", song);
13  }
```

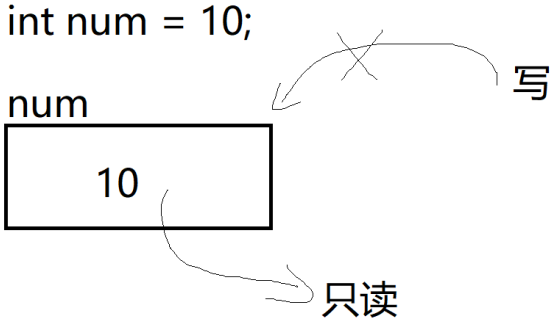
运行结果：

```
12-13
8-23
给我一个威武雄壮的汉子
Press any key to continue
```

知识点2【const关键字】

1、const修饰变量 为只读

const int num = 10;




```
1   void test06()
2   {
3    //const 修饰num为只读变量 num只能取值 num不能被赋值
4    //const 修饰的变量 尽量 初始化
5    const int num=10;
6    printf("num = %d\n",num);//ok
7    //num = 100;//err num不能被赋值
8
9    printf("&num = %p\n", &num);
10
11    //但是如果知道 num的地址 也可以间接的修改num的值
12    *(int *)(&num) = 1000;//少干
```

```
 printf("num = %d\n",num);

}
```

运行结果：

```
num = 10
&num = 0019FED8
num = 1000
Press any key to continue
```

2、const int *p;

const在*的左边 表示 const 修饰的是* 而不是 p.

效果：用户不能借助*p更改空间的内容 但是 p可以指向其他空间（*p 只读 p可读可写）

```
void test07()
{
 int num1 = 10;
 int num2 = 20;

 //const在*的左边 *p只读 p可读可写
 const int *p = &num1;
 printf("*p = %d\n", *p);//10
 //*p = 1000;//err *p只读

 p=&num2;//ok p可读可写
 printf("*p = %d\n", *p);//20
}
```

3、int * const p;

const 在*的右边 const修饰的是p 而不是*。

用户可以通过*p 修改p所指向空间的内容 但是 不能再更改p的指向（*p可读可写 p只读）

```
void test08()
{
 int num1 = 10;
 int num2 = 20;

 //const在*的右边 *p可读可写 p只读
 int * const p = &num1;//p一旦初始化 就不能更改指向

 printf("*p = %d\n", *p);//10
 *p = 1000;//ok *p可读可写
```

```
 printf("*p = %d\n", *p);//1000

 //p=&num2;//err p只读
}
```

4、const int * const p;(*p 只读 p只读)

```
void test09()
{
 int num1 = 10;
 int num2 = 20;

 const int * const p = &num1;//*p 和 p都是只读
 //*p = 1000;//err *p只读
 //p = &num2;//err p只读
}
```

知识点3【结构体类型的定义形式】

```
//先定义类型 再定义变量（推荐）
struct stu
{
    int num;
    char name[32];
    int age;
};

struct stu  lucy;
```

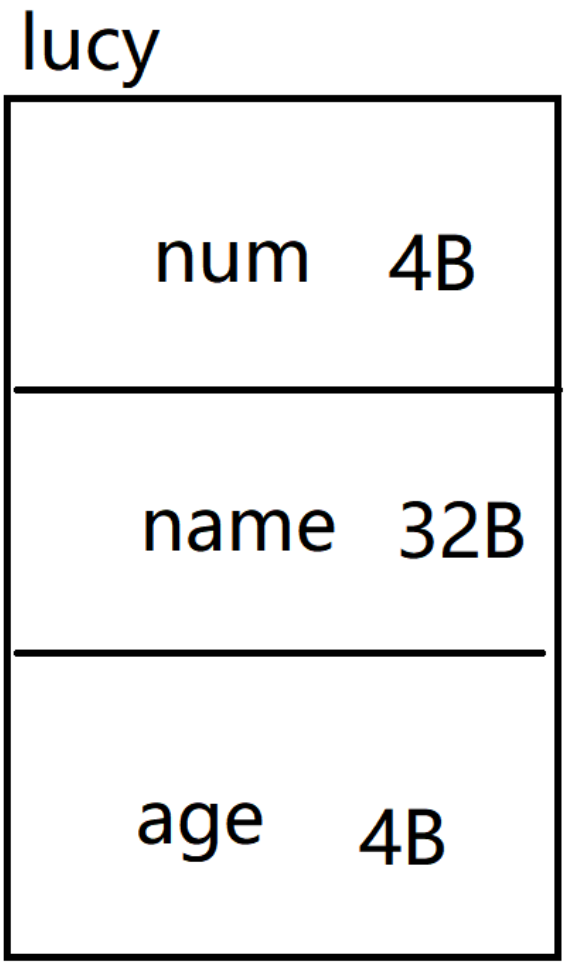
```
//定义类型的同时定义变量
struct stu
{
    int num;
    char name[32];
    int age;
}lucy;

struct stu  bob;
```

```
//定义一次性结构体
struct
{
    int num;
    char name[32];
    int age;
}lucy;

不能定义其他变量
```

```
//struct 是结构体关键字 stu是结构体类型名
//使用结构体类型 必须是 struct stu
//num name age 叫做结构体中的成员
//定义结构体类型的时候 不要给成员 赋值（重要）
//定义结构体类型的时候 并没有分配空间 所以 不能给成员赋值
struct stu
{
 int num=10;//4B
 char name[32];//32B
  int age; //4B
};//一定要记得有;号

```

```
13 //结构体中的成员 拥有独立的空间（重要）
```

struct stu lucy;




```
1 #include<stdio.h>
2 struct stu
3 {
4  int num;//4B
5  char name[32];//32B
6  int age;//4B
7 };
8
9 void test01()
10 {
```

```
    printf("sizeof(struct stu) =%d\n",sizeof(struct stu));//40B
    return;
}
int main(int argc,char *argv[])
{
    test01();
    return 0;
}
```

知识点4【结构体变量的定义】

```
void test01()
{
    //就用结构体类型 定义一个变量
    struct stu lucy;//lucy是局部变量 lucy的成员内容不确定

    //通过结构体变量 访问 结构体中的成员 （一定要遵循成员自身的类型）
    printf("num = %d\n", lucy.num);
    printf("name = %s\n", lucy.name);
    printf("age = %d\n", lucy.age);

    //一定要遵循成员自身的类型
    lucy.num = 100;
    strcpy(lucy.name, "德玛西亚");
    lucy.age = 18;

    printf("num = %d\n", lucy.num);
    printf("name = %s\n", lucy.name);
    printf("age = %d\n", lucy.age);
    return;
}
```

运行结果:

```
num = -858993460
name = 烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫烫0
age = -858993460
num = 100
name = 德玛西亚
age = 18
Press any key to continue
```

知识点5【结构体变量的初始化】（下一）

```c
//结构体类型的定义
struct stu
{

 int num;//4B
 char name[32];//32B
 int age;//4B
};

void test03()
{
 //初始化的顺序 必须和 结构体成员的 顺序一致
 struct stu lucy={100,"德玛西亚",18 };

 printf("num=%d, name=%s, age=%d\n",lucy.num,lucy.name,lucy.age);

 //清空结构体变量
 memset(&lucy,0,sizeof(lucy));
}
```

知识点6【结构体变量 获取 键盘输入】

```c
void test04()
{
 struct stu lucy;
 memset(&lucy, 0,sizeof(lucy));

 printf("请输入num name age:");
 //&lucy.num 取的是num成员地址
 scanf("%d %s %d",&lucy.num, lucy.name , &lucy.age);

 printf("num=%d, name=%s, age=%d\n",lucy.num,lucy.name,lucy.age);

}
```

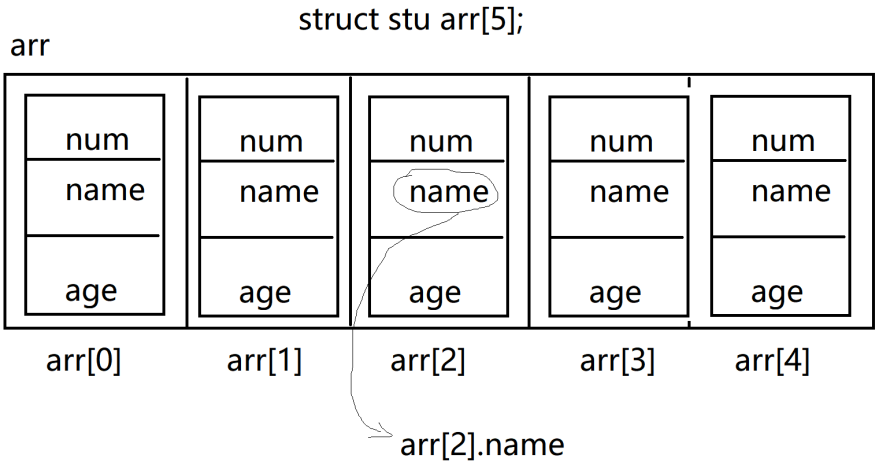
知识点7【结构体变量 之间的赋值】

```c
void test05()
{
 struct stu lucy={100,"小法", 18};
```

```
    struct stu bob;

    //需求 将lucy的值 赋值 bob
    //方式一:逐个成员赋值
    //bob.num = lucy.num;
    //strcpy(bob.name,lucy.name);
     //bob.age = lucy.age;

     //方法二：相同类型的结构体变量 可以直接赋值(推荐)
     //bob = lucy;

     //方法三：方法二的底层实现
     memcpy(&bob,&lucy,sizeof(struct stu));

     printf("num=%d, name=%s, age=%d\n",bob.num,bob.name,bob.age);
}
```

知识点8【结构体数组】

结构体数组




```
void test06()
{
 struct stu arr[5]={
 {100,"小法",18},
 {101,"德玛西亚",19},
 {102,"盲僧",20},
 {103,"快乐风男",30},
 {104,"提莫",8}
 };
```

```
  int n = sizeof(arr)/sizeof(arr[0]);
  int i=0;

  for(i=0;i<n;i++)
  {
  printf("num=%d, name=%s, age=%d\n",arr[i].num,arr[i].name,arr[i].age);
  }
}
```

运行结果：

```
num=100, name=小法, age=18
num=101, name=德玛西亚, age=19
num=102, name=盲僧, age=20
num=103, name=快乐风男, age=30
num=104, name=提莫, age=8
Press any key to continue
```

案例：定义一个结构体数组 获取键盘输入 求平均age

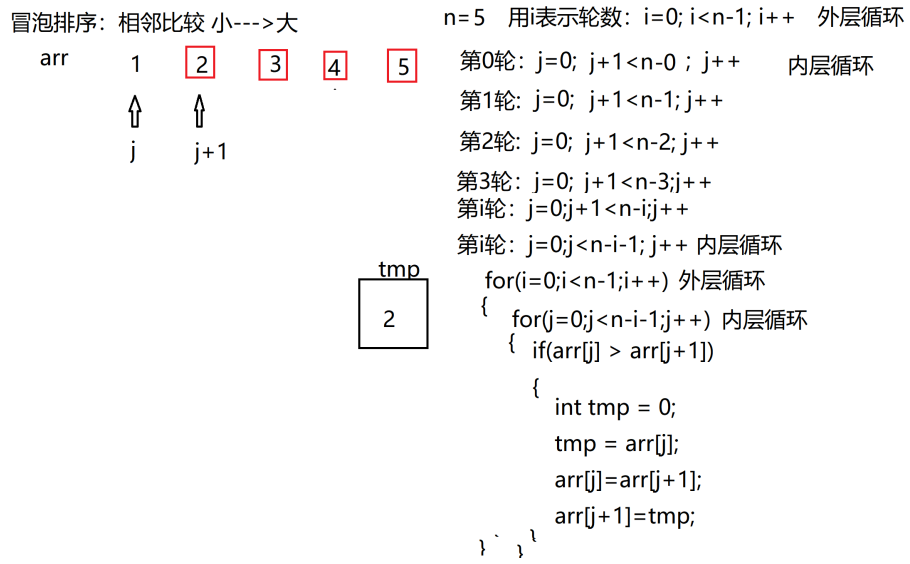
```
void test07()
{
 struct stu arr[5];
 int n = sizeof(arr)/sizeof(arr[0]);
 int i=0;
 int sum = 0;

 memset(arr,0,sizeof(arr));//清空整个数组

 printf("请输入%d个学生的信息\n", n);
 for(i=0;i<n;i++)
 {
 scanf("%d %s %d", &arr[i].num, arr[i].name, &arr[i].age );
 }

 for(i=0;i<n;i++)
 {
 sum += arr[i].age;
 }
 printf("平均年龄为%d\n",sum/n);
```

```
}
```

运行结果：

```
请输入5个学生的信息
100 name1 18
200 name2 19
300 name3 20
400 name4 21
500 name5 22
平均年龄为20
Press any key to continue
```

知识点9【冒泡排序】下二




```c
#include<stdlib.h>
void test08()
{
 int n = 0;
 int i=0,j=0;
 int *arr = NULL;
 printf("请输入数据的个数:");
 scanf("%d", &n);

 //根据元素的个数申请空间
 arr = (int *)calloc(n,sizeof(int));
 if(NULL == arr)
```

```
{
perror("calloc");
return;
}

//键盘给动态数组 获取输入
printf("请输入%d个int数据\n", n);
for(i=0;i<n;i++)
{
scanf("%d", arr+i);//arr+i 第i个元素的地址
}

//排序
for(i=0;i<n-1;i++)
{
for(j=0;j<n-i-1;j++)
{
if(arr[j]>arr[j+1])//从小--->大
{
int tmp = 0;
tmp = arr[j];
arr[j]=arr[j+1];
arr[j+1]=tmp;
}
}
}

//数组的遍历
for(i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
printf("\n");

//释放空间
if(arr != NULL)
{
free(arr);
arr=NULL;
}
```

```
53 }
```

运行结果：

```
请输入数据的个数:10
请输入10个int数据
4 3 1 5 2 7 10 8 9 6
1 2 3 4 5 6 7 8 9 10
Press any key to continue
```

冒泡排序升级版：

```
1  #include<stdlib.h>
2  void test08()
3  {
4   int n = 0;
5   int i=0,j=0;
6   int *arr = NULL;
7   printf("请输入数据的个数:");
8   scanf("%d", &n);
9
10  //根据元素的个数申请空间
11  arr = (int *)calloc(n,sizeof(int));
12  if(NULL == arr)
13  {
14  perror("calloc");
15  return;
16  }
17
18  //键盘给动态数组 获取输入
19  printf("请输入%d个int数据\n", n);
20  for(i=0;i<n;i++)
21  {
22  scanf("%d", arr+i);//arr+i 第i个元素的地址
23  }
24
25  //排序
26  for(i=0;i<n-1;i++)
27  {
28  int flag = 0;
```

```
29  for(j=0;j<n-i-1;j++)
30  {
31  if(arr[j]>arr[j+1])//从小--->大 交换
32  {
33  int tmp = 0;
34  tmp = arr[j];
35  arr[j]=arr[j+1];
36  arr[j+1]=tmp;
37  flag = 1;
38  }
39  }
40
41  if(flag == 0)//数组已经有序
42  break;
43  }
44  printf("i=%d\n",i);
45
46  //数组的遍历
47  for(i=0;i<n;i++)
48  {
49  printf("%d ",arr[i]);
50  }
51  printf("\n");
52
53  //释放空间
54  if(arr != NULL)
55  {
56  free(arr);
57  arr=NULL;
58  }
59 }
```

运行结果：

```
请输入数据的个数:10
请输入10个int数据
1 2 5 3 4 6 7 8 9 10
i=1
1 2 3 4 5 6 7 8 9 10
Press any key to continue
```

知识点10【结构体数组排序】

```c
void test09()
{
 struct stu arr[5];
 int i=0,j=0;
 int n = sizeof(arr)/sizeof(arr[0]);
 memset(arr,0,sizeof(arr));//整个数组 清0

 //获取键盘输入
 printf("请输入%d个学生信息\n",n);
  for(i=0;i<n;i++)
  {
  scanf("%d %s %d",&arr[i].num, arr[i].name, &arr[i].age);
  }

  //根据学号 排序
  for(i=0;i<n-1;i++)
  {
  for(j=0;j<n-i-1;j++)
  {
  if(arr[j].num < arr[j+1].num)//按照num的大小排序
  {
  struct stu tmp;
  tmp = arr[j];
  arr[j]=arr[j+1];
  arr[j+1]=tmp;
  }
  }
  }
```

```
29
30  //遍历
31  printf("---------------\n");
32  for(i=0;i<n;i++)
33  {
34  printf("%d %s %d\n",arr[i].num, arr[i].name,arr[i].age);
35  }
36 }
```

运行结果：