实训六　数据类型构造与模块化程序设计
——数组的构造与应用

一、实训目的及要求

数组是一种构造类型，在解决问题中使用非常普遍，通过本次实训内容，使学生对数组类型有一个系统的认识，并能灵活的在具体问题中进行应用。

程序 1：一个班有 30 个同学，通过键盘输入成绩，并打印输出，每行输出 10 个同学的成绩。

算法分析：
(1)定义一个数组用来存放 30 个成绩数据，
```
int    score[30];
```
(2)用循环结构实现成绩输入；
```
for(i=0;i<30;i++)
    scanf("%d",&score[i]);
```
(3)用循环结构实现成绩输出,并控制换行；
```
for(i=0;i<30;i++)
   {printf("%5d",score[i]);
    if((i+1)%10==0) printf("\n");}
```

```
#include "stdio.h"
main()
{
   int i;
   int    score[30];                  /*成绩数组的定义*/
   for(i=0;i<30;i++)             /*输入成绩*/
         scanf("%d",&score[i]);
     for(i=0;i<30;i++)           /*输出成绩*/
       {printf("%5d",score[i]);
         if((i+1)%10==0) printf("\n");}/*输出 10 个数据换行*/
}
```

程序 2：一个班有 n 个同学，通过键盘输入成绩，并进行以下处理：

(1)求平均成绩；（数组求和）

算法分析：
1.输入 n 的值及 n 个成绩；
2.对成绩进行汇总求和，存入变量 s 中；
3.求平均数：average=s/n;
4.输出平均分 average。

```
#include “stdio.h”
main()
{
   int n,i,s=0;
int    score[30];
float average;
printf(“请输入学生的人数：”);
scanf(“%d”,&n);
printf(“请输入%d 学生的成绩：\n”,n);
for(i=0;i<n;i++)
          {scanf(“%d”,&score[i]);
           s=s+score[i];}
   average=(float)s/n;
    printf(“%.1f”,average);
}
```

(2)添加 m 个同学的成绩；（数组添加）

算法分析：
1.当前成绩个数设为 n 个；
2.输入 m 的值；
3.从第 n 个元素开始输入 m 个成绩；
4.更新数组元素的个数:　　n=m+n;
5.输出添加完后的成绩。

(3)把不及格同学的成绩更新为 60 分；（数组更新）

算法分析：
1.当前成绩个数设为 n 个；
2.从第 0 个元素开始逐个元素进行测试：
　　if(score[i]<60)　score[i]=60;
直到最后一个元素；
3.输出修改完后的成绩元素。

(4)求成绩的最高分和最低分，并记住对应元素的下标；（数组求极值）

算法分析：

1.当前成绩个数设为 n 个，定义变量 max 和 min 分别用来存放最大数和最小数；

2.为 max 和 min 赋初始值:max=min=score[0];

3.从第 1 个元素开始逐个元素进行测试：

```
if(score[i]>max)    max=score[i];
  if(score[i]<min)    min=score[i];
```

直到最后一个元素；

4.输出 max 和 min。

(5)对成绩进行排序。（数组排序）

两种基本算法:

1. 起泡法：将相邻两个数比较，小的调到前面。

2. 选择法：将前面的数和后面的所有数依次进行比较，记住小数的下标，当比较完一遍，用前面的数和该小数进行交换。

起泡法排序：

```
#include “stdio.h”
main()
{int score[10], i,j,k,n=10;
   printf(“ 输入成绩：\n”);
   for(i=0;i<n;i++)
   scanf(“%d”,&score[i]);
   for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++)
             if(score[j] > score[j+1] )
                {k= score[j];
                 score[j] = score[j+1];
                 score[j+1] =k;       }
   for(i=0;i<n;i++)
        printf("%5d", score[i] );
   printf("\n");                }
```

选择法排序：

```
#include “stdio.h”
main()
{
   int i,j,k,m,score[10],n=10;
   for(i=0;i<n;i++)
   scanf(“%d”,&score[i]);
   for(i=0;i<n-1;i++)
      { k=i;
        for(j=i+1;j<n;j++)
```

```
            if (score[k]> score[j])    k=j;
        m=score[i]; score[i]=score[k]; score[k]=m;
        }
    for(i=0;i<n;i++)
         printf("%5d", score[i]);
    printf("\n");
    }
```

程序 3：对已经排好序的成绩数组进行以下操作：把一个新成绩按照顺序插入到数组的合适位置。（提高）。

算法分析：

1. 从键盘接收一个数据，存入变量 temp；
2. 根据变量 m 的大小进行定位，其对应下标为 k;
3. 把 a[9]到 a[i]的元素依次后移，为新数据腾出空间；
4. 把 temp 存入下标为 i 的空间中：score[i]=temp;
5. 输出处理完后的新数组。

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

int main()
{
    int a[11]={1,4,6,9,13,16,19,28,40,100};
    int temp,i=0,j=0;
    scanf("%d",&temp);
    for(i=9;i>=0;i--)
    {
        if(temp<a[i]){
                a[i+1] = a[i];
        }else{
                a[i+1] = temp;
                break;
        }
    }
    for(j=0;j<11;j++)
    {
        printf("%d\t",a[i]);
     }
    //system("pause");
    return 0;
}
```

程序 4：编程实现求一个 3 行 4 列整型数组的平均数。

算法分析：
1.定义一个二维数组 a[3][4];
2.为数组赋值；
3.累加元素的和，存入变量 s 中；
4.求平均数 ave=s/12;
5.输出平均数 ave。

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

int main()
{
    int a[3][4],i,j,s=0;
    float ave;
    for(i=0;i<3;i++)
        {
            for(j=0;j<4;j++)
            scanf("%d",&a[i][j]);
        }
    for(i=0;i<3;i++)
        {
            for(j=0;j<4;j++)
            s+=a[i][j];
        }
    ave=s/12;
    printf("average=%5f\n",ave);
    return 0;
}
```

程序 5：编程实现把一个三行三列的二维数组转置输出。

算法分析：
1.定义一个二维数组 a[3][3];
2.为数组赋值；
3.交换 a[i][j]与 a[j][i]的值；
4.输出交换后的数组 a。

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

```
int main()
{
	int a[3][3],i,j;
	int t;
	for(i=0;i<3;i++)
		{
			for(j=0;j<3;j++)
				{
					scanf("%d",&a[i][j]);
				}
		}
	for(i=0;i<3;i++)
		{
			for(j=0;j<i;j++)
			{
				t=a[i][j];
				a[i][j]=a[j][i];
				a[j][i]=t;
			}
		}
	for(i=0;i<3;i++)
		{
			for(j=0;j<3;j++)
				{
					printf("%d ",a[i][j]);
				}
			printf("\n");
		}
	return 0;
}
```

程序 6：某学习小组有 4 名同学，学习了 5 门课程，求每个同学的平均分和每门课程的平均分。

算法分析：

1.定义一个二维数组 a[4][5];
2.为数组赋值；
3.求行平均数，把平均数存入 SAvg 中；
4.求列平均数，把平均数存入 CAvg 中；
5.输出行和列的平均数。

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

```
#include <stdlib.h>

int main()
{
	int a[4][5];
	int i,j;
	float SAvg,CAvg;
	for(i=0;i<4;i++)
		{
			for(j=0;j<5;j++)
			{
				scanf("%d",&a[i][j]);
			}
		}
	for(i=0;i<4;i++)
		{
			SAvg=0;
			for(j=0;j<5;j++)
			{
				SAvg=SAvg+a[i][j];
				printf("%f",SAvg);
			}
			printf("%d 课程平均成绩:%f\n",i,SAvg/5);
		}
	for(i=0;i<5;i++)
		{
			CAvg=0;
			for(j=0;j<4;j++)
			{
				CAvg=CAvg+a[j][i];
				printf("%f",CAvg);
			}
			printf("%d 学生平均成绩:%f\n",i,CAvg/4);
		}
	return 0;
}
```

程序 7：从键盘输入一行字符，要求删除某个字符（要删除的字符也由键盘输入）。

算法分析：

1.定义存放字符串的字符数组 str 和存放单个字符的字符变量 ch；

2.输入字符串 str 和要删除的字符 ch；
3.对要删除的 ch 定位；
4.从该位置开始，开始把后续字符依次前移；
5.检查字符串的结束标记。

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


char s[10000];
char c;

int main()
{
printf("Input the original string:\n");
gets(s);
printf("The original string is: %s\n",s);
printf("Input the char should be deleted:\n");
scanf("%c",&c);
printf("The char should be deleted is %c\n",c);
for(int i=0;s[i];i++)
if(s[i]==c)
{
memmove(s+i,s+i+1,strlen(s+i+1)+1);
break;
}
printf("The string after delete operation is: %s\n",s);
return 0;
}
```

程序 8：编一程序，将两个字符串连接起来。

算法分析：
1.定位：第一个字符串的‘\0’的位置；
2.从‘\0’开始把第二个字符串的字符依次放入第一个字符串的后端，直至第二个字符串的‘\0’；
3.检验第一个字符串的末端是否有结束符’\0’，若没有，修正所得的字符串，在它的末端加上‘\0’。

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

```
int main()
{
    char s1[80],s2[80];
    int i=0,j=0;
    gets(s1);
    gets(s2);
    while(s1[i]!='\0')
        {
            i++;
        }
    while(s2[j]!='\0')
        {
            s1[i]=s2[j];
            i++;
            j++;
        }
    s1[i]='\0';
    puts(s1);
    return 0;
}
```

程序 9：有三个字符串(长度不超过 20)，要求找出其中最大者。

算法分析:
1.输入三个字符串,存入二维字符数组中;
2.先取前两个字符串比较,找出大的存入 string 数组中;
3.用 string 和后续的依次进行比较,当出现比 string 的时更新 string 的值;
4.输出 string 中的字符串。

```
#include "stdio.h"
int main()
{
    char string[20],str[3][20];
    int i;
    for(i=0;i<3;i++)
        {
            gets(str[i]);
        }
    if(strcmp(str[0],str[1])>0)
        {
            strcpy(string,str[0]);
        }
    else
```

```
        {
            strcpy(string,str[1]);
        }
    if(strcmp(str[2],string)>0)
        {
            strcpy(string,str[2]);
        }
    printf("\nthe largest string is:\n%s\n",string);
    return 0;
}
```

字符串的大小并不是指长度的大小。实际上,字符串的比较是比较字符串中各对字符的 ASCII 码。

首先比较两个串的第一个字符，若不相等，则停止比较并得出大于或小于的结果；

如果相等就接着比较第二个字符然后第三个字符等等。

如果两上字符串前面的字符一直相等,像"disk"和"disks"那样,

前四个字符都一样, 然后比较第五个字符,前一个字符串"disk"只剩下结束符'/0',后一个字符串"disks"剩下's','/0'的 ASCII 码小于's'的 ASCII 码,所以得出了结果。因此无论两个字符串是什么样，strcmp 函数最多比较到其中一个字符串遇到结束符'/0'为止，就能得出结果。

注意:字符串是数组类型而非简单类型，不能用关系运算进行大小比较。

strcmp 函数是比较两个字符串的大小,返回比较的结果。一般形式是：

strcmp(字符串 1，字符串 2);

其中，字符串 1 、字符串 2 均可为字符串常量或变量；

①字符串 1 小于字符串 2,strcmp 函数返回一个负值;

②字符串 1 等于字符串 2,strcmp 函数返回零;

③字符串 1 大于字符串 2,strcmp 函数返回一个正值;

实训题目

1、从键盘输入 20 个整型数据，统计其中正数的个数，并计算它们的求和。

2、把 1000 之内的素数存放在数组中，并输出素数的个数和各个素数。

3、在第一题的基础上找出最大数和最小数并输出对应的下标。

4、任意输入 10 个数据，对其进行排序（用选择法小到大）。

5、在第 4 题的基础上，从键盘上接收一个数据，如果该数不存在，把该数按照顺序放在数组中，若存在则把和该数相等的元素删除。

6、某学习小组有 4 名同学，学习了 5 门课程，编程求出最高分和最低分及其对应的行号和列号。