第6章函数

C程序是由一个个函数组成的，最简单的C程序也有一个main()函数 。

函数分库函数和用户自定义函数。

本章所要介绍的是C语言的用户自定义函数。我们将分析学生信息管理系统中的用户自定义函数，使读者掌握用户自定义函数的编程方法。

【学习目标】

- 掌握函数的定义、函数的声明及函数的调用
- 掌握函数的实参与形参的概念及参数结合方式
- 了解函数的嵌套与递归调用
- 掌握全局变量、局部变量、静态变量的概念及使用方法
- 了解变量的存储属性
- 了解多文件的编译与连接
- 能够熟练使用函数编写程序

6.1案例中的自定义函数

6.1.1 案例中的自定义函数简介

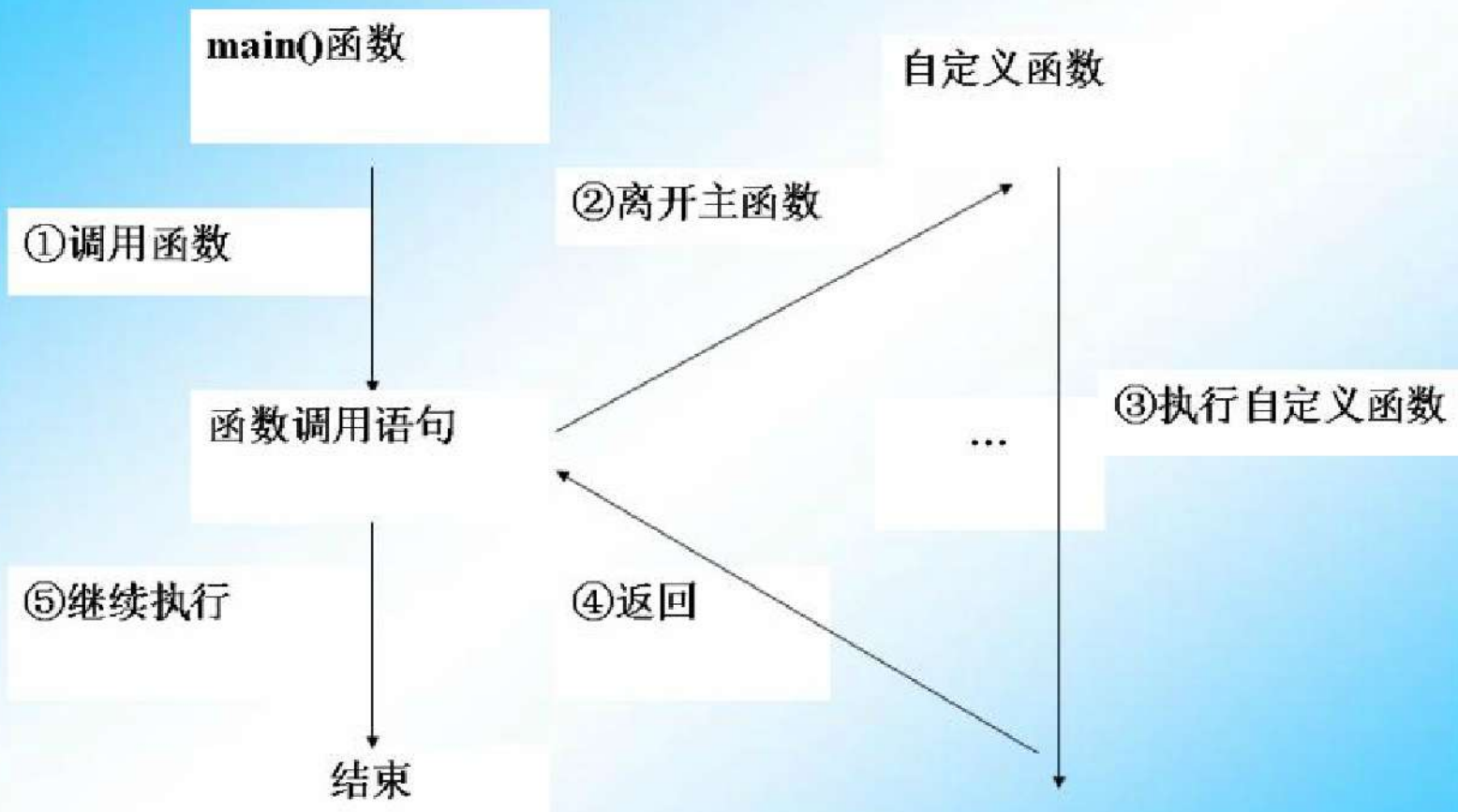
菜单选项	函数名	函数功能
1	add()	添加学生信息
2	display()	显示学生信息
3	modify()	修改学生信息
4	del()	删除学生信息
5	find()	查找学生信息
6	compute()	统计平均分等
7	sort()	按总分排序
8	save_txt()	保存至文件(文本方式)
9	load_txt()	从文件读入(文本方式)
10	save_bin()	保存至文件(二进制方式)
11	load_bin()	从文件读入(二进制方式)
无	menu_choose()	程序主菜单

【例6-1】案例中main()函数与用户自定义函数之间的调用关系。

```
#include "student.h"
void main()
{
    int menu_choose(); //菜单函数声明
    while(1)
    {     int choice=menu_choose(); //调用菜单函数，得到选择结果
          switch(choice)          //根据用户选择的菜单调用自定义函数
          {         case 1: add();     break; //添加学生信息
                    case 2: display(); break; //显示学生信息
                    case 3: modify();  break; //修改学生信息
                    case 4: del();     break; //删除学生信息
                    case 5: find();    break; //查找学生信息
                    case 6: compute(); break; //统计不及格人数、最高分和平均分
                    case 7: sort();    break; //按总分排序
                    case 8: save_txt(); break; //数据保存至文件(文本方式)
                    case 9: load_txt(); break; //从文件读取数据(文本方式)
                    case 10: save_bin(); break; //数据保存至文件(二进制方式)
                    case 11: load_bin(); break; //从文件读入数据(二进制方式)
                    case 12: exit(0);    break; //退出程序
          }
    }
}
```

```
int menu_choose()
{
    int choice;
    printf("\n                欢迎使用学生信息管理系统\n");
    printf("————————————————————————————————————————————————\n");
    printf(" 1.添加学生信息          2.显示学生信息\n");
    printf(" 3.修改学生信息          4.删除学生信息\n");
    printf(" 5.查找学生信息          6.统计不及格人数、最高分和平均分\n");
    printf(" 7.按总分排序            8.保存至文件(文本方式)\n");
    printf(" 9.从文件读入(文本方式)  10.保存至文件(二进制方式)\n");
    printf("11.从文件读入(二进制方式) 12 .退出程序\n");
    printf("————————————————————————————————————————————————\n");
    printf(" 请选择功能模块，输入数字1-12: ");
    while(1)
    {     scanf("%d",&choice);
          if(choice>=1 && choice<=12)
                break;
          else
                printf("\n\t 输入数字不正确，请重输1-12: ");
    }
    return choice;
}
```

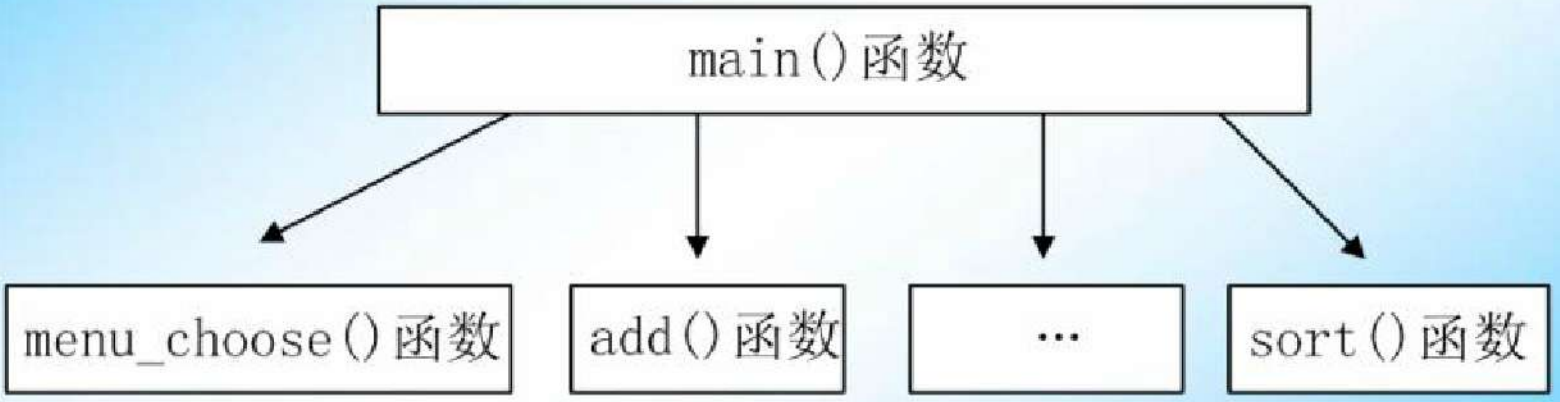
6.1.2 函数调用过程



例如，在例6-1中，main()函数体内有以下调用自定义函数的语句：

```
int choice=menu_choose();
```

6.1.3 案例中函数之间的调用关系



6.2 函数的分类、定义与声明

6.2.1 函数的分类

1. 从函数的来源分，可分为库函数和用户自定义函数
2. 从函数有无返回值分，可分为有返回值函数和无返回值函数

- 有返回值函数：执行完后，会向主调函数返回一个执行结果。
  如例6-1中的menu_choose()函数就属于有返回值函数，通过语句choice=menu_choose();调用后，会将函数的返回值赋给变量choice。
- 无返回值函数：此类函数执行完后，不向主调函数返回任何值。如例6-1中的add()函数，调用函数后没有返回值。

3. 从参数传递的角度分，可分为无参函数和有参函数

6.2.2 函数的定义

定义函数的语法格式如下：

```
返回值类型  函数名(形式参数列表)
{
    函数体
}
```

例如，定义menu_choose()函数的程序代码如下：

```
int menu_choose()
{
  ...
  return choice;
}
```

定义add()函数的程序代码如下：

```
void add()
{
    ...
}
```

6.2.3 函数的声明

函数声明的目的是：说明函数的类型和参数情况，以便调用函数时，编译系统能够正确识别函数，并检查函数调用是否合法。

函数声明有以下两种语法格式：

返回值类型　函数名(类型1 形参1，...，类型n 形参n);

返回值类型　函数名(类型1，...，类型n);

例如，在例6-1中，在main()函数的开头有一条语句：

```
int menu_choose();
```

如果自定义函数放在主调函数之前，则可以不进行函数声明，反之则必须声明。

6.3 函数的调用

6.3.1 函数调用的一般格式

一般格式如下：

函数名(实际参数列表);

(1) 实际参数(简称实参)可以是常量、变量或表达式

例如：

```
int x=6,y=4,z;
z=pow(x,y);  //求x^y
```

上述语句中，x、y为实参，是两个变量。

(2) 无参函数调用时不需要给出实际参数。

例如：choice=menu_choose();

函数调用的方式有三种：

①赋值表达式

choice=menu_choose();

通常用于调用有返回值的函数。

②函数语句

例如：add();

通常用于调用没有返回值的函数或不需要使用函数的返回值。

③函数参数：将函数调用作为一个函数的实参。

例如：printf("%d ", menu_choose());

6.3.2 参数传递

函数调用时将实参的值传递给形参，称为参数传递，常常也称为“虚实结合”。

【例6-2】 编写一个自定义函数swap()，在main()函数中输入两个数，调用swap()函数交换两个形参变量的值。

```
#include<stdio.h>
void main()
{   int m,n;
    void swap(int a,int b);                          //函数声明
    printf("请输入两个整数:");
    scanf("%d%d",&m,&n);
    printf("main()中交换前:m=%d,n=%d\n",m,n);
    swap(m,n);                                       //调用函数
    printf("main()中交换后:m=%d,n=%d\n",m,n);
}
void swap(int a,int b)
{   int temp;
    printf("swap()中交换前:a=%d,b=%d\n",a,b);
    temp=a;                          //交换两参数的值
    a=b;
    b=temp;
    printf("swap()中交换后:a=%d,b=%d\n",a,b);
}
```

6.3.3 函数结果的返回

return语句的使用格式如下：

return (表达式);

或者

return;

其中，圆括号可以省略。第一种格式有返回值，第二种格式无返回值。

当函数无返回值时，函数中可以没有return语句。

例如：例6-1中的add()函数。

函数也允许有一个或多个return语句，但每次调用时只执行一个return语句，因此，只能返回一个值。

【例6-3】编写一个求圆的面积的函数，要求在main()函数中输入圆的半径，调用该函数后输出结果。

```
#include<stdio.h>
#define PI 3.14
void main()
{   float r;
    float ymj(float r);
    printf("请输入圆的半径:");
    scanf("%f",&r);
    printf("圆面积=%.2f\n",ymj(r));//输出结果
}
float ymj(float r)                                  //函数定义
{
    float s;
    s=PI*r*r;
    return s;                                       //函数返回
}
```

6.4 函数应用举例

【例6-4】编写函数判断一个数是否是素数，如果是，则显示“是素数”，否则，显示“不是素数”。要求在main()函数中进行数据的输入和输出。

编程思路：

将判断一个数是否为素数编写成一个函数，函数需要一个参数，以说明是判断哪一个数；如果是素数，返回值为1，否则，返回值为0，因此，函数的返回值类型为整型。

```
#include <stdio.h>
#include <math.h>
int prime(int k)                          //定义函数
{   int i,n;
    n=(int)sqrt(k);
    for(i=2;i<=n;i++)
    {     if(k%i==0)
                  return(0);              //返回结果
    }
    return(1);                            //返回结果
}
void main()
{   int a,flag;
    printf("请输入一个数:");
    scanf("%d",&a);
    flag=prime(a);                  //调用函数prime
    if(flag==1)           printf("%d是一个素数!\n",a);
    else                  printf("%d不是一个素数!\n",a);
}
```

【例6-5】编写求n!的函数，调用该函数求

$$C_m^n = \frac{m!}{n!*(m-n)!}$$

要求在main()函数中输入自然数m和n的值，并输出结果。

编程思路：

将求n!编写成一个函数，函数需要一个参数，以说明是求哪个数的阶乘，函数的返回值类型为长整型(或整型)。

```
#include <stdio.h>
void main()
{   long fac(int k);                          //函数声明
    int m,n;
    long c;
    printf("请输入m与n的值(m>n):");
    scanf("%d%d",&m,&n);
    c=fac(m)/(fac(n)*fac(m-n));//三次调用函数fac
    printf("c=%ld\n",c);
}
long fac(int k)                                    //定义函数
{   int i;
    long f=1;
    for(i=1;i<=k;i++)          f=f*i;
    return f;                              //返回结果
}
```

【例6-6】编写函数判断一个数是否为回文，在main()函数中调用该函数输出1000到9000之间所有的回文数字，每行输出10个数。所谓回文数字是指正读和倒读都一样的数字。例如：98789。

编程思路：

- 将判断一个数是否为回文编写成函数，函数需要一个参数，以说明是判断哪一个数；如果是回文数字，返回值为1，否则，返回值为0，因此，函数的返回值类型为整型。
- 判断回文时可以用除10取余的方法，从最低位开始，依次取出该数的各位数字，然后，将最低位放到最高位，按反序重新构建一个新数，并比较新数与原数是否相等，若相等，则是回文数字。

```
#include <stdio.h>
int huiwen(int n);                          //在所有函数外声明函数
void main()
{   int i,j=0;
    for(i=1000;i<=9000;i++)
    {     if(huiwen(i))                     //调用函数
          {       printf("%5d",i);
                  j++;                      //输出计数
                  if(j%10==0)               printf("\n");
          }
    }
    printf("\n");
}
int huiwen(int n)                           //定义函数
{   int k,m=0;   //m为重新构建的新数
    k=n;
    while(k)
    {     m=m*10+k%10; //每次将旧数的最低位加入新数中
          k=k/10;      //每次舍掉旧数的最低位
    }
    return(m==n);
}
```

6.5 函数的嵌套调用与递归调用

6.5.1函数的嵌套调用

在函数中又调用其它函数，称为函数的嵌套调用。

【例6-7】使用函数嵌套调用，编程计算

$$C_m^n = \frac{m!}{n!*(m-n)!}$$

```
#include <stdio.h>
long fac(int k)                                    //定义函数fac,求n!
{   int i;
    long f=1;
    for(i=1;i<=k;i++)        f=f*i;
    return f;                                      //返回结果
}
long comb(int m,int n)                             //定义函数comb,求
{   long c;
    c=fac(m)/(fac(n)*fac(m-n)); //三次调用函数fac
    return c;                                                    //返回结果
}
void main()
{
    int m,n;
    long c;
    printf("请输入m与n的值(m>n):");
    scanf("%d%d",&m,&n);
    c=comb(m,n);                                                 //调用函数
    comb
    printf("c=%ld\n",c);
}
```

6.5.2函数的递归调用

- 一个函数除了可以调用其它函数外，还可以直接或间接地调用该函数本身，这种函数自己调用自己的形式称为函数的递归调用。

函数直接递归调用	函数间接递归调用	
int hs1(int x) **{** **int y;** **…** **y=hs1(x-1);** **…** **return y;** **}**	**int hs2(int x)** **{** **int y;** **…** **y=hs3(x);** **…** **return y;** **}**	**int hs3(int x)** **{** **int z;** **…** **z=hs2(x);** **…** **return z;** **}**

编写递归程序的有两个关键点：

(1)构造递归表达式；

(2)找出终止条件。

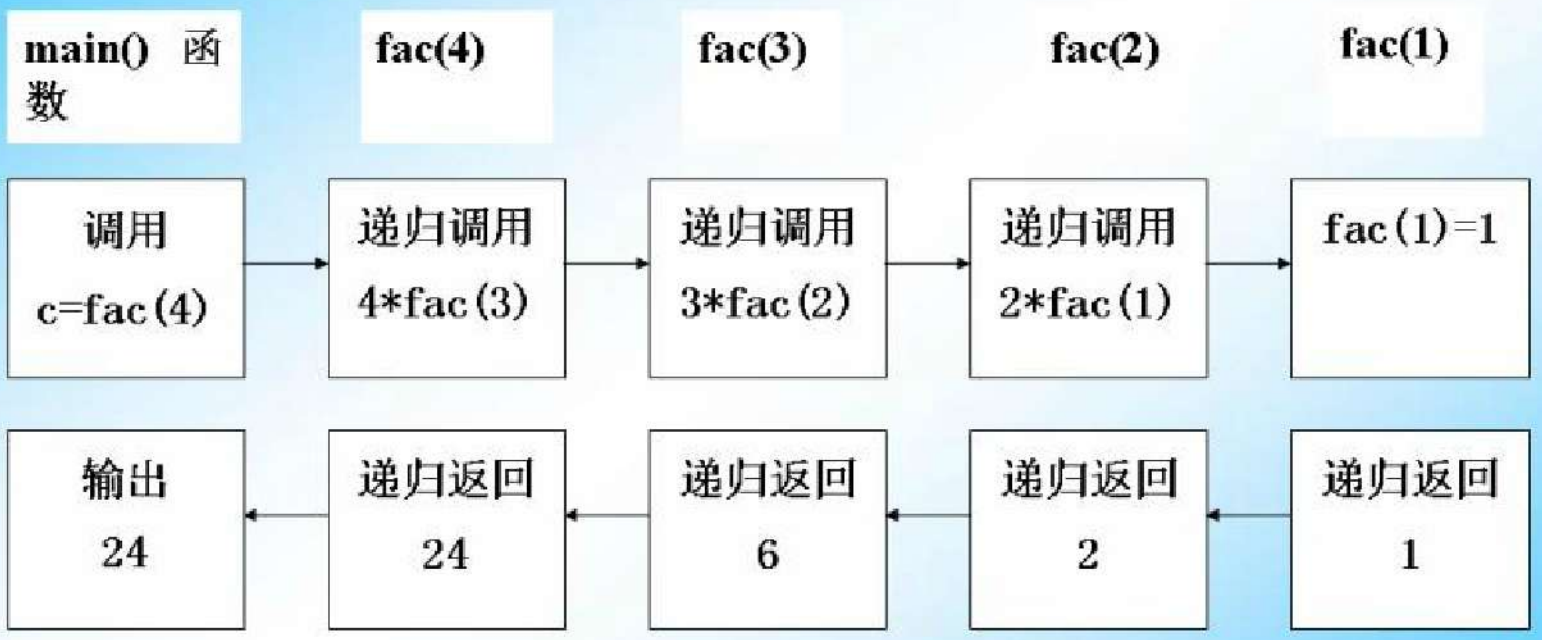
【例6-8】用递归法求n!。

编程思路：构造递归公式：

$$n!=\begin{cases}1 & (n=1) \quad \text{——终止条件}\\ n(n-1)! & (n>1) \quad \text{——递归表达式}\end{cases}$$

```
#include <stdio.h>
long fac(int k)                                    //定义函数fac
{   long f;
    if(k==1)
        f=1;
    else
        f=k*fac(k-1);                              //递归调用函数fac
    return f;                                      //返回结果
}
void main()
{   int n;
    long c;
    printf("请输入n的值:");
    scanf("%d",&n);
    c=fac(n);                                      //调用函数fac/
    printf("%d!=%ld\n",n,c);
}
```

函数递归调用的执行过程



【例6-9】汉诺(Hanoi)塔问题：古代有一个梵塔，塔内有A、B、C三个座，座A上有64个大小不等的盘子，大的在下，小的在上，如图6-7所示。有一个和尚想把这64个盘子从座A全部移到座C，在移动过程中可以借助座A、座B或座C，但每次只允许移动一个盘子，并且不允许大盘放在小盘的上面。要求打印出移动的步骤。

编程思路：

这是一个使用递归方法解决问题的典型例子，编程时首先要找出递归的两个关键点，即：

递归终止条件：只有一个盘子时，可以直接移动。

递归表达式：要找递归表达式，可按如下设想进行：

将n个盘子从座A移到座C可以描述为：

(1)将n-1个盘子从座A借助座C移到座B；

(2)将剩下的一个盘子从座A移到座C；

(3)将n-1个盘子从座B借助座A移到座C。

上述(1)、(3)两步操作都是将n-1个盘子从一个座上移到另一个座上，只是座的名字不同，因此，可以将它们的操作编写成一个函数，函数需要四个参数，一个是盘子的个数，另外三个分别为：起点座、借助座、终点座。

```
#include <stdio.h>
void main()
{   void hanoi(int n,char a,char b,char c); //声明递归函数
    int m;
    printf("请输入盘子的个数：");
    scanf("%d",&m);
    printf("%d个盘子移动的步骤如下:\n",m);
    hanoi(m,'A','B','C');                          //调用hanoi函数
 }
//定义递归函数，将n个盘子从a座借助b座,移到c座
void hanoi(int n,char a,char b,char c)
{   if(n==1)                                    //1个盘子，直接移动
        printf("from %c to %c\n",a,c);
    else
    {    hanoi(n-1,a,c,b);                      //递归调用hanoi函数
         printf("from %c to %c\n",a,c);
         hanoi(n-1,b,a,c);                      //递归调用hanoi函数
    }
}
```

6.6数组作为函数参数

数组作为函数参数有两种情况：(1) 数组元素作为函数实参；(2) 数组名作为函数实参。

6.6.1 数组元素作为函数实参

- 数组元素相当于一个普通变量，用数组元素作为函数实参时，函数的形参只能是变量，函数调用时将实参的值传给形参。

【例6-10】编写一个将小写字母转换成大写字母的函数，在main()函数中调用该函数，将一个字符串中所有的小写字母转换成大写字母。要求采用数组元素作为函数实参。

```
#include<stdio.h>
char change(char x)                              //定义函数
{   x=x-32;
    return x;
}
void main()
{   char a[10]="hello",b[10];
    int i;
    printf("调用函数前数组a的值:");
    puts(a);
    for(i=0;a[i]!='\0';i++)
          b[i]=change(a[i]);                     //调用函数,数组元素作实参
    b[i]='\0';
    printf("调用函数结果:");
    puts(b);
    printf("调用函数后数组a的值:");
    puts(a);
}
```

6.6.2数组名作为函数实参

- 数组名作为函数实参时，形参应当用数组或指针(指针将在第7章中介绍)。
- 由于数组名表示数组的首地址，因此，实参向形参传递的不是数组的值，而是实参数组的首地址，这样，形参数组和实参数组共占相同的内存单元。

【例6-11】编写一个将小写字母转换成大写字母的函数，在main()函数中调用该函数，将一个字符串中所有的小写字母转换成大写字母。要求采用数组名作为函数实参。

```
#include<stdio.h>
void change(char b[10])                                    //定义函数
{
    int i;
    for(i=0;b[i]!='\0';i++)
        b[i]=b[i]-32;                    //函数形参值改变了
    printf("调用函数结果:");
    puts(b);
}
void main()
{
    char a[10]="hello";
    printf("调用函数前数组a的值:");
    puts(a);
    change(a);//调用函数
    printf("调用函数后数组a的值:");
    puts(a);
}
```

【例6-12】有一个N×N的矩阵，编写函数求两条对角线上的所有元素之和。要求在main()函数中输入矩阵数据，并输出结果。

```
#define N 3
#include <stdio.h>
void main()
{   int i,j,a[N][N],s;
    int sum(int array[][N]);                    //函数声明
    printf("请输入矩阵:");
    for(i=0;i<N;i++)
          for(j=0;j<N;j++)
                scanf("%d",&a[i][j]);
    s=sum(a);                                   //调用函数
    printf("两条对角线上的元素之和为:%d\n",s);
}
int sum(int array[][N])          //定义函数，可以省略第一维大小说明
{   int i,s=0;
    for(i=0;i<N;i++)             s=s+array[i][i];
    for(i=0;i<N;i++)             s=s+array[i][N-1-i];
    return (s);                                 //返回函数值
}
```

6.7变量的作用域和存储属性

6.7.1 变量的作用域

【例6-13】在学生信息管理系统中，各种类型变量的使用。

```
struct stu_type stu[100];  //定义学生结构体数组
void main()
{
   int n;
   ...
   while(1)
   {
      int choice;
      choice=menu_choose();
      ...
   }
}
int menu_choose()
{
   int choice;
   ...
}
```

choice 的有效范围

choice 的有效范围

stu[]的有效范围

- C语言中，所有的变量都有自己的使用范围，即作用域，每个变量只在自己的作用域范围内有效、是可见的。
- 变量的作用域由变量在程序中定义的位置决定。
- C语言中，根据作用域的不同，可以将变量分为局部变量和全局变量两种。

1. 局部变量

- 局部变量是在函数内定义的变量 。
- 在例6-13中，menu_choose()函数内的choice变量是局部变量，在本函数体内有效；main()函数复合语句内的choice变量也是局部变量，该变量只在本复合语句内有效。
- 允许在不同函数中使用相同的变量名，它们代表不同的对象，分配不同的存储单元，互不干扰。

【例6-14】局部变量的使用。

```
#include<stdio.h>
void main()
{
    int i=2;
    int k=8;                                    //main函数中的变量k
    {
        int k=1;                                //复合语句中的变量k
        k=k+i;
        printf("复合语句中的k=%d\n",k);//输出复合语句中变量k的值
    }
    k=k+i;
    printf("复合语句外的k=%d\n",k);//输出复合语句外变量k的值
}
```

2. 全局变量

- 全局变量(也称为外部变量)是在函数外定义的变量。它不属于哪一个函数，其作用域从定义变量的位置开始到本程序结束。
- 例如，在例6-13程序开头定义的结构体数组stu[]，在main()函数和menu_choose()函数中都能使用。
- 全局变量在不同函数间建立了一条直接的数据传递通道。
- 例如，“已输入的学生人数”和“每个学生的信息”，在add.c文件中定义方式如下：

```
struct stu_type stu[100];
int stu_num=0;     //已输入的学生数量
```

- 对于全局变量，特别说明如下：

(1) 一般情况下，尽量不要使用全局变量。

(2) 允许全局变量和局部变量同名。此时，在局部变量的作用域内，全局变量不起作用。

【例6-15】写出下面程序的执行结果。

```
#include <stdio.h>
int a=66,b=11;                                  // a,b为全局变量
int max(int x,int y)
{
int c;
c=x>y?x:y;
   return c;
}
void main()
{
int a=88;                                       //a为局部变量
printf ("max=%d\n", max(a,b));
 }
```

【例6-16】输入一行字符，统计其中字母、数字、空格及其它字符的个数。

```
#include<stdio.h>
int letter,digit,space,others;//全局变量
void main()
{   char str[1000];
    void count(char ch[]);                  //声明函数
    printf("请输入一个字符串(<1000个字符):");
    gets(str);                                          //获取字符串
    letter=0;
    digit=0;
    space=0;
    others=0;
    count(str);                                 //调用函数
    printf("字母:%d\n数字:%d\n空格:%d\n
其它:%d\n",letter,digit,space,others);
}
```

```
void count(char ch[])                                    //定义函数
{
    int i;
    for(i=0;ch[i]!='\0';i++)
    {
        if((ch[i]>='a' && ch[i]<='z')||(ch[i]>='A' && ch[i]<='Z'))
                letter++;
        else if(ch[i]>='0' && ch[i]<='9')
                digit++;
        else if(ch[i]==32)
                space++;
        else
                others++;
   }
}
```

6.7.2 变量的存储属性

- 供用户程序使用的内存空间分为四部分。
- 存放在静态存储区中的变量有：

(1) 全局变量；

(2) 用static说明的局部变量。

- 存放在动态存储区中的变量有：没用static说明的局部变量。
- 自由存储区用于动态内存分配，它的使用方法将在第7章指针中进行介绍。

- 定义变量的完整格式是：

  [存储类别] 数据类型 变量名[=初值];
- 变量的存储类别有四种，即auto(自动)、static(静态)、register(寄存器)和extern(外部)。
- 存储类别可以省略，省略时自动采用默认值，局部变量的默认值是auto。

1. 自动局部变量(auto变量)

- 其定义格式如下：

  [auto] 数据类型 变量名[=初值];

- 由于这种变量极为常用，C语言将其作为缺省的存储类型，我们以前用到的多数变量都属于这种类型。
- 自动局部变量存储在内存的动态存储区中，属于动态存储方式，其所占内存空间在程序运行过程中动态建立和撤消。

【例6-17】分析下面程序中变量的生存期。

```
#include<stdio.h>
void main()
{
    void prt(void);//函数声明
    int x=1;
    prt();
    printf("%d\n",x);
}
void prt(void)
{
    auto int x=5;
    printf("%d\n",x);
}
```

2. 静态局部变量(static变量)

- 定义格式如下：

  static 数据类型 变量名[=初值];

- 静态局部变量存储在内存的静态存储区，属于静态存储方式。静态局部变量在程序开始运行时就分配了存储空间，并在程序运行期间始终占用，直到整个程序运行结束。

【例6-18】利用静态变量求1到6之间各个数的阶乘。

```
#include <stdio.h>
void main()
{
    int fac(int n);
    int i;
    printf("1-6各数的阶乘为:\n");
    for(i=1;i<=6;i++)
        printf("%d!=%d\n",i,fac(i));  //调用函数fac
}
int fac(int n)
fac
{
    static int f=1;
    f=f*n;
    return f;
}
```

3. 寄存器变量(register变量)

- 所谓寄存器变量就是为变量分配的存储单元不是在内存中，而是在CPU的某个寄存器中。寄存器变量的定义格式如下：

  register数据类型 变量名[=初值];

- 例如，register int x=9;
- 将一些使用频率很高的变量放在CPU的寄存器中，使用时直接从寄存器中取数参与运算，这样可以避免对内存的频繁访问，提高程序的执行效率。

4. 外部变量

- 全局变量在函数的外部定义，存放在静态存储区中，其生存周期为程序的整个运行期。
- 全局变量如果没有指定其存储类别，默认它就是一个外部变量。
- 外部变量的作用域是从定义点开始到本文件的末尾。如果要在定义点之前或在其它文件中使用它，就必须采用关键字extern对该外部变量进行声明，声明的语法格式为：

  extern 数据类型 外部变量名;
- **extern**不是用来定义外部变量的，而是用来声明外部变量的。

例如，在学生信息管理系统中

文件**add.c**

```
int stu_num=0;     //已输入的学生数量
struct stu_type stu[100]; //学生数组
void add()
{  ...}
```

文件**display.c**

```
extern int stu_num;      //已输入的学生数量
extern struct stu_type stu[]; //学生数组
void display()
{  ...}
```

将这两条声明语句放到了头文件**student.h**中，其它文件只需要包含该头文件即可。

6.8 外部函数与内部函数

1 外部函数

- 外部函数能被其它文件调用，其定义的语法格式如下：

  [extern] 类型标识符 函数名(形参表)

- 在定义函数时如果省略extern，则默认为外部函数。以前我们介绍过的所有函数都是外部函数。

2 内部函数

- 内部函数只能在本文件中调用，其定义的语法格式如下：

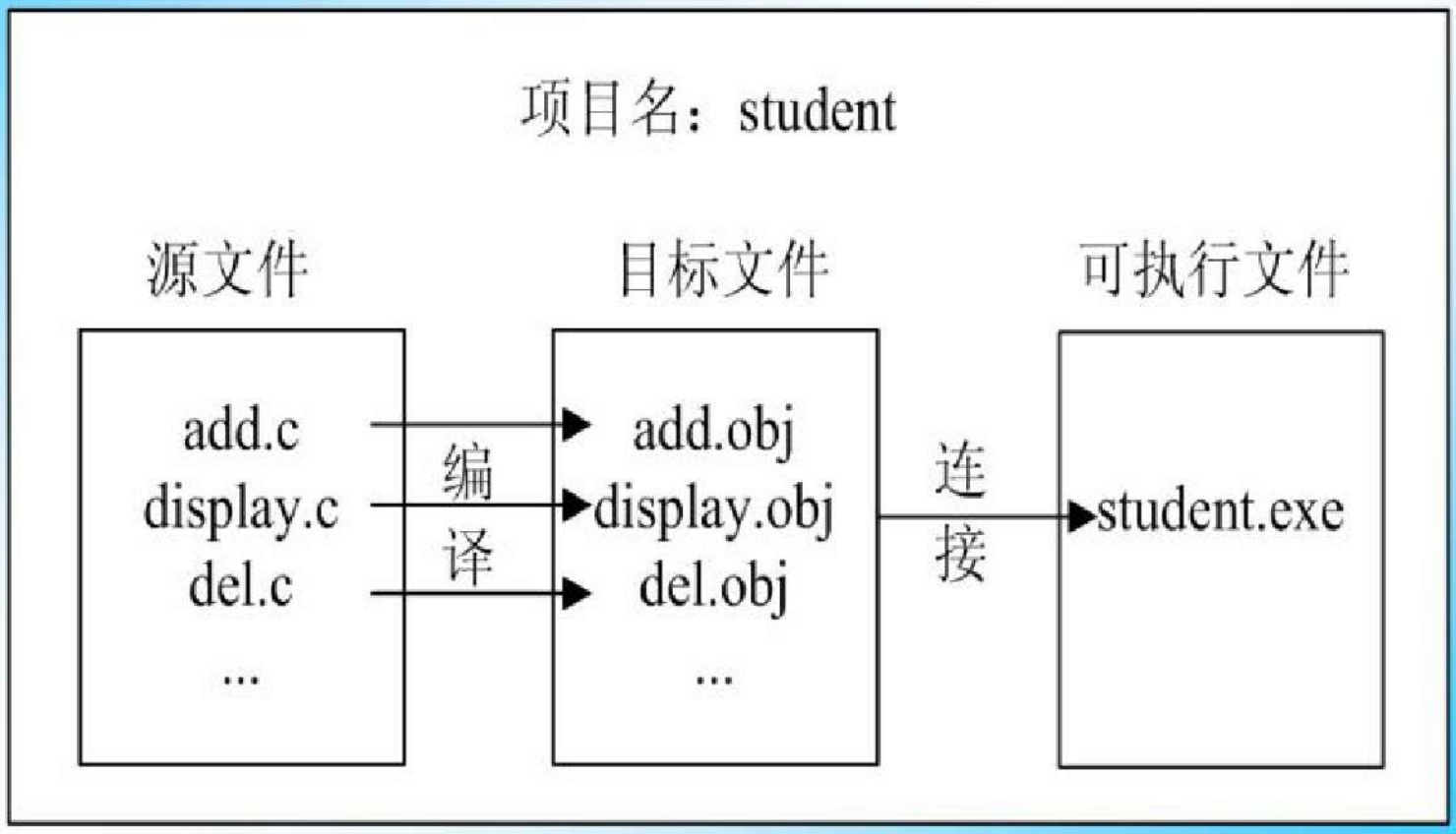
  static 类型标识符 函数名(形参表)

- 内部函数又称静态函数，如果在不同文件中有同名的内部函数，将互不干扰。

6.9 多文件的编译与连接

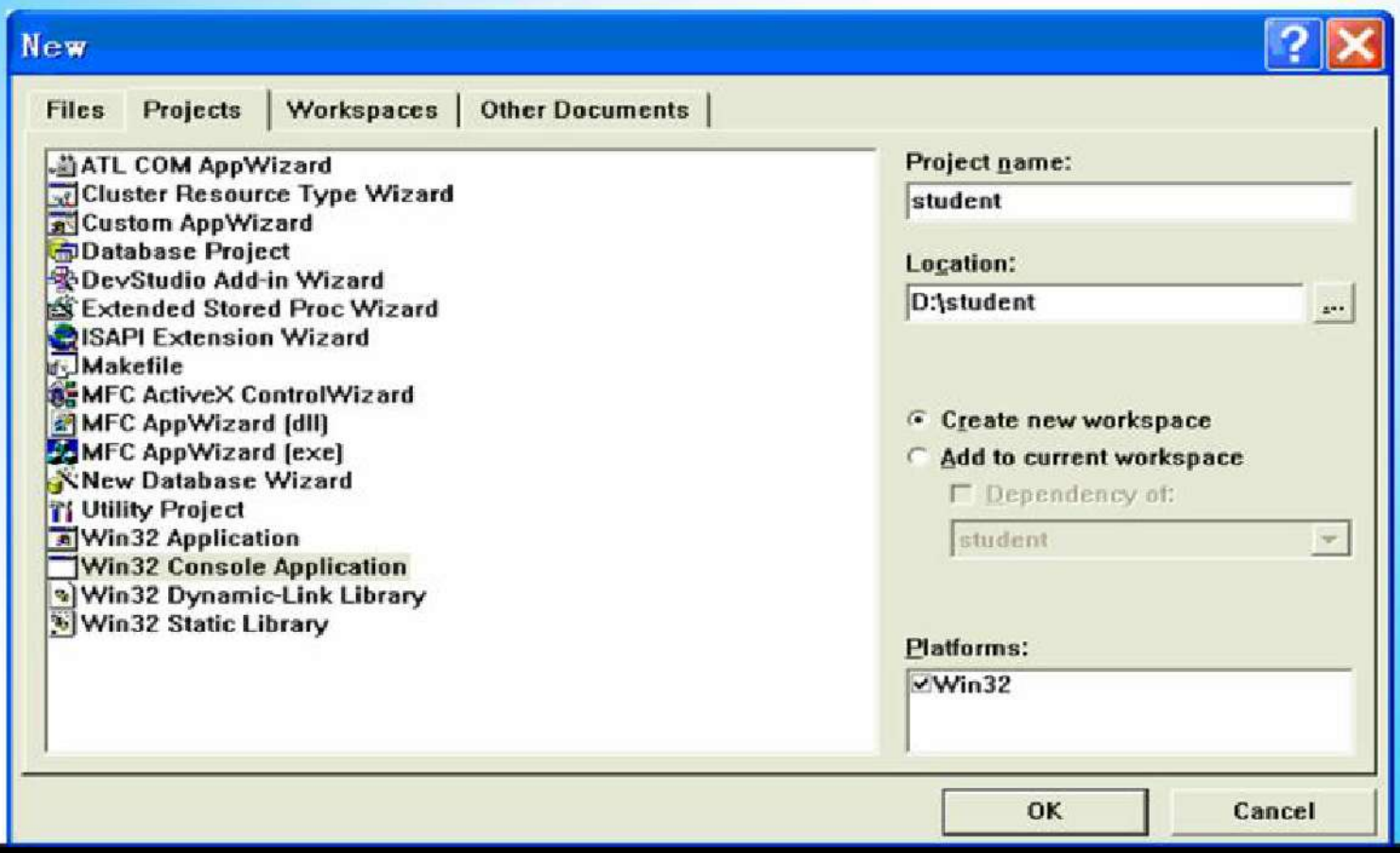
- C语言是一种支持模块化程序设计的语言，它允许将一个大型程序分成多个文件进行编写，通常每个文件对应系统的一个功能模块，学生信息管理系统就是采用这种方式。
- 在项目开发时，通常将数据结构的定义、全局变量声明、自定义函数声明等内容统一放在一个头文件中，其它文件只需要包含该头文件即可。
- 采用模块化程序设计的好处是：程序结构清晰，便于编程、调试和维护。

VC 6.0采用项目来管理多文件的编译和连接，我们以学生信息系统为例说明整个项目的编译、连接过程，如图6-9所示。

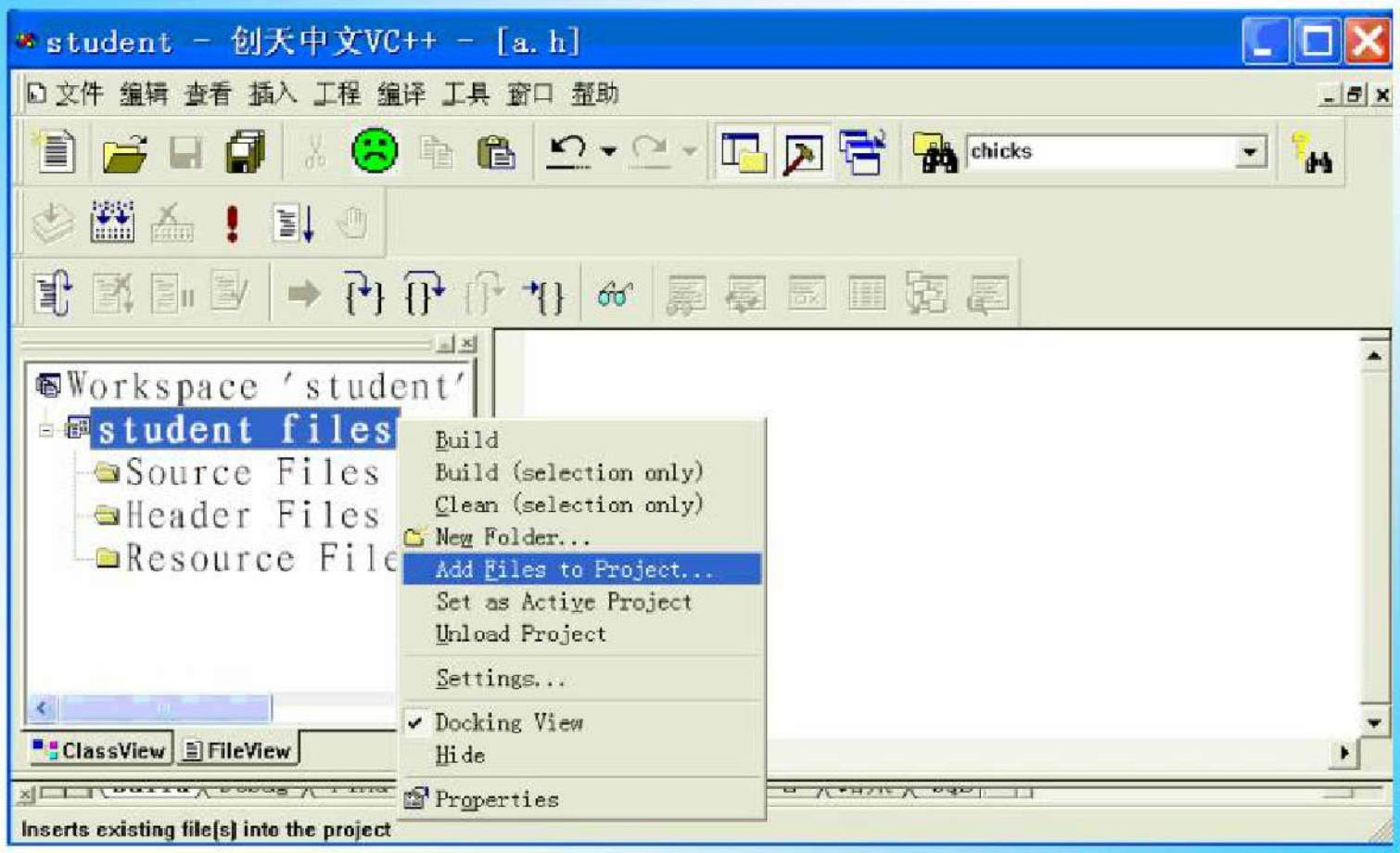


建立项目及编译、连接的具体步骤如下：

(1) 建立一个项目



(2) 添加已有的.c或.h文件到项目中



(3)在项目中添加新文件

- 如果需要在项目中添加新文件，方法和第1章1.4.2小节中建立源文件的方法相同。

(4)编译、连接和执行

本章小结

- 本章主要介绍了C程序中用户自定义函数的使用方法，主要内容包括：

(1) 函数的定义与声明。

(2) 函数的调用，包括：函数调用格式、调用过程、参数传递方式(值传递与地址传递)、函数的返回值。

(3) 数组元素与数组名都可以作为函数参数。数组元素作为参数时，是按值传递，数组名作为参数时，是按地址传递。

(4) 函数可以嵌套调用及递归调用。

(5) 变量的作用域与存储属性总结如下：

①按变量的作用域分
- 局部变量：在函数内定义的变量
- 全局变量：在函数外定义的变量

②按变量的生存期分
- 自动变量(auto)：定义时分配内存，作用域结束时销毁
- 静态变量(static)：在程序运行期间始终存在
- 寄存器变量(register)：保存在 CPU 的寄存器中，作用域结束时销毁
- 外部变量：在程序运行期间始终存在，在变量的作用域外使用时需用 extern 进行声明

(6) 函数分内部函数与外部函数。内部函数用**static**关键词修饰，只能在本文件中调用，而外部函数可以被其它文件调用。

(7) 采用项目来管理多文件的编译与连接。

15. 编写函数计算$z=x^2+y^3$，在main()函数中输入x和y，并在main()函数中输出计算结果。

```
#include<stdio.h>
void main()
{
    int x,y,z;
    int comput(int x,int y);                    //函数声明
    scanf("%d%d",&x,&y);
    z=comput(x,y);
    printf("%d ",z);
}
int comput(int x,int y)
{
    return x*x+y*y*y;
}
```

练习题

1. 编写函数：计算以下表达式的值。要求在主函数中输入x的值，并输出结果。

$$y=\begin{cases}x^2-2x+1 & (x<0)\\ x^3+x+3 & (x\geq 0)\end{cases}$$

2. 编写函数输出所有在正整数m~n之间能被6和9整除的数。在主函数中输入m和n的值，调用该函数输出结果。
3. 编写函数判断一个数是否是素数，如果是，则显示“是素数”，否则，显示“不是素数”。要求在main()函数中进行数据的输入和输出。
4. 编写函数power()，用来求n^k的值。在主函数中输入两个正整数n和k，调用函数power()，求$1^k+2^k+3^k+\cdots+n^k$的值，并输出结果。