第5章 函 数

CHAPTER

5.1 内容概述

本章主要介绍函数的定义、调用、参数传递规则、嵌套调用和递归调用及其与带参数的宏的区别，以及主函数与命令行参数、变量的作用域和存储类别等内容。本章的知识结构如图 5.1 所示。

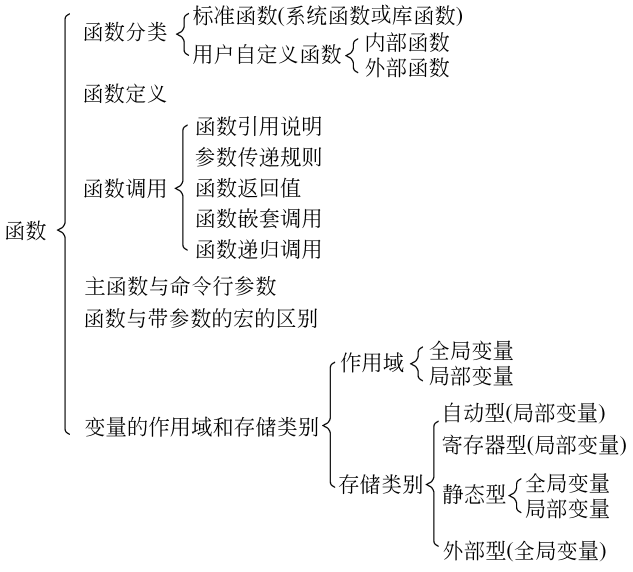

图 5.1 第 5 章知识结构

考核要求：熟练掌握函数的定义、调用形式、参数传递规则、返回值类型和递归调用；熟练掌握变量的作用域与存储类别；了解主函数与命令行参数；了解函数与带参数的宏的区别。

重点难点：本章的重点是函数的定义和调用方法，调用函数时的数据传递方法，变量的作用域和存储类别。本章的难点是函数的参数值传递和地址传递的区别以及递归函数的设计。

核心考点：函数的定义和调用，变量的作用域和存储类别。

5.2 典型题解析

【例5.1】 以下函数首部中正确的是(　　)。

A. int play(var :integer,var b:integer)

B. float play(int a,b)

C. double play(int a,int b)

D. void play(a as integer,b as integer)

解析：函数定义的一般形式为

```
[函数存储类别][函数返回值类型] 函数名([函数形式参数表])
{ 函数体说明部分
  函数功能语句序列
  [return 表达式;]
}
```

若省略存储类别,则系统默认为 extern,即外部函数。若省略返回值类型,则系统默认为 int。形式参数表的说明格式为

```
类型 1 形参 1,类型 2 形参 2,…,类型 n 形参 n
```

本题中,选项 A、选项 B 和选项 D 的形式参数表的说明格式都是错误的。

答案：C

【例 5.2】 若已定义的函数有返回值,则以下关于该函数调用的叙述中错误的是(　　)。

A. 函数调用可以作为独立的语句存在

B. 函数调用可以作为一个函数的实参

C. 函数调用可以出现在表达式中

D. 函数调用可以作为一个函数的形参

解析：函数调用有三种方式。一是把函数调用作为一个语句,二是函数调用出现在一个表达式中,三是把函数调用作为一个函数的实际参数。

答案：D

【例 5.3】 有以下函数定义：

```
void fun(int n, double x) { … }
```

若以下选项中的变量都已正确定义并赋值,则正确调用函数 fun 的语句是(　　)。

A. fun(int y,double m);　　B. k=fun(10,12.5);

C. fun(x,n);　　D. void fun(n,x);

解析：函数调用的一般形式为

```
函数名([实际参数表])
```

函数调用时不能写实际参数的类型和函数返回值的类型,故选项 A 和选项 D 是错误的。若函数返回值的类型为 void(空类型),则禁止在函数调用中使用被调用函数的返回

值，故选项 B 是错误的。

答案：C

【例 5.4】 程序中对函数 fun()有如下引用说明：

```
void * fun();
```

此说明的含义是(　　)。

A. 函数 fun()无返回值

B. 函数 fun()的返回值可以是任意数据类型

C. 函数 fun()的返回值是无值型的指针类型

D. 指针 fun 指向一个函数，该函数无返回值

解析：函数引用说明的主要作用是利用它在程序的编译阶段对被调用函数的合法性进行全面检查，包括函数名、函数返回值的类型、形式参数的个数、形式参数的类型和顺序。函数引用说明的形式为

```
函数返回值类型 函数名(类型 1 形参 1,类型 2 形参 2,…);
```

其中，形参名可以省略，写成

```
函数返回值类型 函数名(类型 1,类型 2,…);
```

当参数类型都为 int 或 char 时，形参类型名也可以省略，写成

```
函数返回值类型 函数名();
```

本题中，函数引用说明的含义是被调用函数的函数名为 fun，函数的返回值类型为空类型指针。

答案：C

【例 5.5】 有以下程序：

```
#include<stdio.h>
int f(int n);
void main()
{ int s;
  s=f(4); printf("%d\n",s);
}
int f(int n)
{ int s;
  if(n>0) s=n+f(n-1);else  s=0;
  return s;
}
```

该程序的输出结果是(　　)。

A. 4　　B. 10　　C. 14　　D. 6

解析：本题主要考查对函数递归调用的理解。该程序中，递归结束条件是 n≤0，若满足递归结束条件，则不再递归，否则一直执行 s＝n＋f(n－1)操作，展开此求和公式得

s=4+f(3)=4+3+f(2)=4+3+2+f(1)=4+3+2+1+f(0)=4+3+2+1+0=10。

答案：B

【例5.6】 有以下程序：

```
#include<stdio.h>
void fun(char *c,int d)
{ *c=*c+1;d=d+1;
  printf("%c,%c,",*c,d);
}
void main()
{ char b='a',a='A';
  fun(&b,a);  printf("%c,%c\n",b,a);
}
```

该程序的输出结果是(　　)。

A. b,B,b,A　　B. b,B,B,A　　C. a,B,B,a　　D. a,B,a,B

解析：C语言中，函数参数的传递是单向值传递。确切地说，函数被调用时，系统会为每个形参分配存储单元，然后把相应的实参值传送到这些存储单元作为形参的初值，最后执行规定的操作。在函数中对形参的操作不会影响调用函数中的实参，即形参的值不能传回给实参。当指针作为函数的参数时，形参和实参指向同一存储单元，修改形参指向存储单元的值就等于修改实参所指向的存储单元的值。

本题中，变量a的值('A')传递给形参变量d，a和d在内存中占用不同的存储单元。形参变量d的值在函数fun中被修改为'B'，但实参变量a的值不变。变量b的地址传递给形参指针变量c，此时，&b和c都指向变量b。因此，*c就是b，对*c进行操作就是对b进行操作，即b的值在函数fun中被修改为'b'。

答案：A

【例5.7】 有以下程序：

```
#include<stdio.h>
void sum(int a[])
{ a[0]=a[-1]+a[1];}
void main()
{ int a[10]={1,2,3,4,5,6,7,8,9,10};
  sum(a+2);
  printf("%d\n",a[2]);
}
```

该程序的输出结果是(　　)。

A. 6　　B. 7　　C. 5　　D. 8

解析：一维数组名可以作为函数的参数，调用函数时的参数传递方式是址传递。此时，实参数组和形参数组共占用同一段内存，函数中对形参数组的操作实质上就是对实参数组的操作。

本题中，实参传递给形参的是数组元素a[2]的地址a+2。因此，函数sum()中的

a[0]就是主函数 main()中的 a[2]。函数调用结束后,实参数组元素 a[2]的值为实参数组元素 a[1]和 a[3]的和,其值为 6。

答案：A

【例 5.8】 有以下程序：

```
#include<stdio.h>
float f1(float n)
{ return n * n;}
float f2(float n)
{ return 2 * n;}
void main()
{ float (*p1)(float),(*p2)(float),(*t)(float),y1,y2;
  p1=f1; p2=f2;
  y1=p2(p1(2.0));
  t=p1;p1=p2;p2=t;
  y2=p2(p1(2.0));
  printf("%3.0f,%3.0f\n",y1,y2);
}
```

该程序的输出结果是(　　)。

A. 8，16　　B. 8，8　　C. 16，16　　D. 4，8

解析：C 语言中,可以通过指向函数的指针变量调用函数,函数调用的一般形式为

(*指针变量名)(实参表列)

或

指针变量名(实参表列)

本题中,先使指针变量 p1 指向函数 f1(),p2 指向函数 f2(),通过 p1 调用函数 f1()(返回值为 4),通过 p2 调用函数 f2()(返回值为 8,即 y1＝8),再交换 p1 和 p2,使指针变量 p1 指向函数 f2(),p2 指向函数 f1(),通过 p1 调用函数 f2()(返回值为 4),通过 p2 调用函数 f1()(返回值为 16,即 y2＝16)。

答案：A

【例 5.9】 以下叙述中正确的是(　　)。

A. 局部变量说明为 static 存储类别,其生存期将得到延长

B. 全局变量说明为 static 存储类别,其作用域将被扩大

C. 任何存储类别的变量在未赋初值时,其值都是不确定的

D. 形参可以使用的存储类别说明符与局部变量完全相同

解析：若局部变量的存储类别说明为 static,则该变量在静态存储区分配存储空间,所占用的空间一直持续到程序执行结束,其生存期将得到延长,故选项 A 正确。若全局变量的存储类别说明为 static,则该变量只能在定义它的文件内引用,同一程序的其他文件不能使用,其作用域将被缩小,故选项 B 错误。static 型和 extern 型变量的初始化在程序编译时处理,程序执行时不再处理,系统为没有初始化的变量赋 0 值,故选项 C 错误。

局部变量的存储类别可以说明为static,但形参不可以说明为static,故选项D错误。

答案：A

【例5.10】 有以下程序：

```
#include<stdio.h>
int fun()
{ static int x=1;
  x*=2;
  return x;
}
void main()
{ int i,s=1;
  for(i=1;i<=3;i++) s*=fun();
  printf("%d\n",s);
}
```

该程序的输出结果是(　　)。

A. 0　　B. 10　　C. 30　　D. 64

解析：static型局部变量在静态存储区分配存储空间,其占用的存储单元在函数调用结束后不会释放。下一次函数调用时,该变量的值就是上一次函数调用结束时的值。另外,static型变量的初始化在程序编译时处理,程序执行时不再处理。

本题中,在函数fun()内定义了static型局部变量x,其初值为1。函数fun()被调用了3次,每次调用前后x值的变化情况如下。

第1次调用：调用前,x=1;调用后,x=2。

第2次调用：调用前,x=2;调用后,x=4。

第3次调用：调用前,x=4;调用后,x=8。

由此可得,s=1×2×4×8=64。

答案：D

【例5.11】 有以下程序：

```
#include<stdio.h>
int a=2;
int f(int n)
{ static int a=3;
  int t=0;
  if(n%2){ static int a=4; t+=a++;}
  else { static int a=5; t+=a++; }
  return t+a++;
}
void main()
{ int s=a, i;
  for(i=0;i<3;i++) s+=f(i);
```

```
  printf("%d\n", s);
}
```

该程序的输出结果是(　　)。

A. 26　　B. 28　　C. 29　　D. 24

解析：C语言中，不同范围内允许使用同名的变量，在引用时，如果局部范围内定义了变量，则局部引用，否则向外扩展引用。另外，static型局部变量在编译时赋初值，在程序运行时已有初值，以后每次调用函数时不再为其重新赋初值，只保留上次调用结束时的值。auto型局部变量在函数调用时赋初值，每调用一次函数都会重新分配存储单元并赋初值。

本题中，在程序首部定义了全局变量a，其作用域是整个程序；在函数f()的说明部分定义了static型局部变量a，其作用域是函数f()的内部；在if和else后的复合语句内分别定义了static型局部变量a，其作用域分别是定义它的复合语句。为了便于说明，函数f()的说明部分定义的变量a用f_a表示，if后面的复合语句中定义的变量a用if_a表示，else后面的复合语句中定义的变量a用else_a表示。程序的执行过程如下。

主函数中使用全局变量a，s=2。

第1次调用：n=0，f_a=3，t=0。由于n%2的值为0，执行else后的复合语句，else_a=5，t=0+else_a=5，else_a=6。返回t+f_a的值(8)，f_a=4。

第2次调用：n=1，f_a=4，t=0。由于n%2的值为1，执行if后的复合语句，if_a=4，t=0+if_a=4，if_a=5。返回t+f_a的值(8)，f_a=5。

第3次调用：n=2，f_a=5，t=0。由于n%2的值为0，执行else后的复合语句，else_a=6，t=0+else_a=6，else_a=7。返回t+f_a的值(11)，f_a=6。

由此可得，s=2+8+8+11=29。

答案：C

【例5.12】 函数fun()的功能是对形参s所指向的字符串中下标为奇数的字符按ASCII码值进行递增排序，并将排序后下标为奇数的字符取出，存入形参p所指向的字符数组中，形成一个新串。请填空。

```
#include<stdio.h>
void fun(char *s, char *p)
{ int i,j,n=0,x,t;
  for(i=0; s[i]!='\0'; i++)  n++;
  for(i=1;i<n-2;i=i+2)
  { ____①____;
    for(j=____②____;j<n;j=j+2)
      if(s[t]>s[j]) t=j;
    if(t!=i)
    { x=s[i]; s[i]=s[t]; s[t]=x;}
  }
  for(i=1,j=0;i<n;i=i+2,j++)  p[j]=s[i];
  p[j]=____③____;
}
```

解析：用字符数组名或指向字符串的指针变量作为函数参数，可以在被调用函数中修改主调函数中的字符串内容，其原因是实参和形参指向同一存储单元。

本题使用的排序算法是简单选择排序。根据简单选择排序的思想及函数结构，t是存放最小元素下标的变量，初始值应为i，故①处应填写t=i。因为只对下标为奇数的字符进行排序，所以第i趟排序的第一次比较应是s[i]与s[i+2]的比较，故②处应填写i+2或t+2。把排序后下标为奇数的字符存入p所指向的字符数组后，应在尾部加上字符串结束标志'\0'，故③处应填写'\0'。

答案：① t=i　　② i+2或t+2　　③ '\0'

【例5.13】 下列程序中，函数select()的功能是从N行M列的二维数组中找出最大值。将最大值的地址作为函数返回值，并通过形参传回此最大值所在行的下标和列的下标。请填空。

```
#include<stdio.h>
#define  N 3
#define  M 3
int *select(int a[N][M],int *n,int *m)
{ int i,j,row=0,col=0;
  for(i=0;i<N;i++)
    for(j=0;j<M;j++)
      if(a[i][j]>a[row][col]) {row=i;col=j;}
  *n=____①____;
  *m=col;
  return ____②____;
}
void main()
{ int a[N][M]={9,11,23,6,1,15,9,17,20},*max,n,m;
  max=select(____③____);
  printf("max=%d,row=%d,col=%d\n",*max,n,m);
}
```

解析：查找函数select()的基本思路如下。用row和col分别存放最大值的行标和列标，初始时，把a[0][0]看作临时最大值，即row=0，col=0。然后用数组中的元素逐个与临时最大值a[row][col]进行比较，若大于临时最大值，则用当前数组元素的下标(i,j)更新临时最大值下标(row,col)。最终得到的最大值是a[row][col]。由于函数的返回值是最大值地址，因此②处应填写&a[row][col]、*(a+row)+col或a[row]+col，其中的row和col也可以分别用*n和*m替代。由于最大值的列标已存放到指针m指向的单元，因此最大值行标应存放到指针n指向的单元，即①处应填写row。由于函数调用时函数实参的类型和个数一定要与形参的类型和个数保持一致，因此③处应填写a，&n，&m。

答案：① row　　② &a[row][col]或*(a+row)+col或a[row]+col(其中的row和col也可以分别用*n和*m替代)　　③ a，&n，&m

【例 5.14】 函数 fun()的功能是先找出 N 行 N 列矩阵各行中的最大数，再求出这 N 个最大数中的最小数。请填空。

```
#include<stdio.h>
#define  N 10
int fun(int (*a)[N])
{ int row,col,max,min;
  for(row=0;row<N;row++)
  { for(max=a[row][0],col=1;col<N;col++)
    if(    ①    ) max=a[row][col];
      if(row==0) min=max;
      else if(    ②    ) min=max;
  }
  return min;
}
```

解析：查找函数 fun()的基本思路如下。用 max 存放一行中的最大数，用 min 存放 N 个最大数中的最小数。对于矩阵的每一行，初始时把该行的第一个元素看作临时最大数，即 max=a[row][0]，然后用 max 与该行后面的 N－1 个元素依次进行比较，若 max 较小，则用当前数组元素更新 max。如果当前行是第 1 行，则 min 的值就是 max 的值，否则用当前行最大数 max 与 min 进行比较，若 max 较小，则用 max 更新 min。由此可知，①处应填写 max<a[row][col](或 max<*(*(a+row)+col)，或 max<*(a[row]+col))，②处应填写 max<min(或 min>max)。

答案：① max<a[row][col] 或 max<*(*(a+row)+col)或 max<*(a[row]+col)

② max<min 或 min>max

【例 5.15】 函数 fun()的功能是从形参 s 所指向的字符串中寻找与参数 c 相同的字符，并在其后插入一个与之相同的字符，若找不到相同的字符，则函数不做任何处理。

```
void fun(char *s, char c)
{ int i,j,n;
  for(i=0;s[i]!=    ①    ; i++)
    if(s[i]==c)
    { n=    ②    ;
      while(s[i+1+n]!='\0')  n++;
      for(j=i+n+1; j>i; j--) s[j+1]=s[j];
      s[j+1]=s[    ③    ];
      i=i+1;
    }
}
```

解析：函数 fun()的基本思路如下。从 s 所指向的字符串的第一个字符(s[0])开始顺序查找与 c 相同的字符，若找到与 c 相同的字符(s[i])，则统计 s[i]后面的字符的个数

(n),并将其后面的字符(包括'\0')依次后移一位,然后将找到的字符s[i]插入空出的位置,再从插入字符的下一个字符开始,重复上述操作,直到查找到串尾('\0')为止。由此可知,①处应填写'\0',②处应填写0,③处应填写i(或j)。

答案:① '\0'　　② 0　　③ i或j

【例5.16】 编写函数fun(int n),其功能是用筛选法统计所有小于或等于n(n<10000)的素数的个数。

解析:用筛选法得到小于或等于n的所有素数的方法如下。从数2开始,将所有2的倍数从数表中删除(把数表中相应位置的值置0),然后从数表中查找下一个非0数,并从数表中删除该数的所有倍数,以此类推,直到搜索完整个数表为止。

```
int fun(int n)
{ int a[10000],i,j,count=0;
  for(i=2;i<=n;i++) a[i]=i;
  i=2;
  while(i<n)
  { for(j=a[i]*2;j<=n;j+=a[i]) a[j]=0;
    i++;
    while(i<n&&a[i]==0) i++;
  }
  for(i=2;i<=n;i++)
  if(a[i]!=0) count++;
  return count;
}
```

【例5.17】 假设字符串中只包含字母和“*”号。编写程序,通过函数调用方式删除字符串中除前导“*”号之外的全部“*”号。要求:不得使用C语言提供的字符串处理函数。

解析:设字符串的首地址为a。首先统计前导“*”号个数(i),然后从第一个非“*”号字符(指针变量p指向)开始扫描,如果p指向的字符不为“*”号,则执行a[i++]=*p,重复此操作,直到p指向的字符为'\0'为止。

```
#include<stdio.h>
void fun(char *a)
{ int i=0;char *p=a;
  while(*p&&*p=='*')
  { i++;p++;}
  while(*p)
  { if(*p!='*') { a[i]=*p;i++;}
    p++;
  }
  a[i]='\0';
}
void main()
```