第5章 函 数

CHAPTER

5.1 内容概述

本章主要介绍了函数的定义、调用、函数参数传递规则、函数的嵌套调用和递归调用、函数与带参数的宏的区别、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。

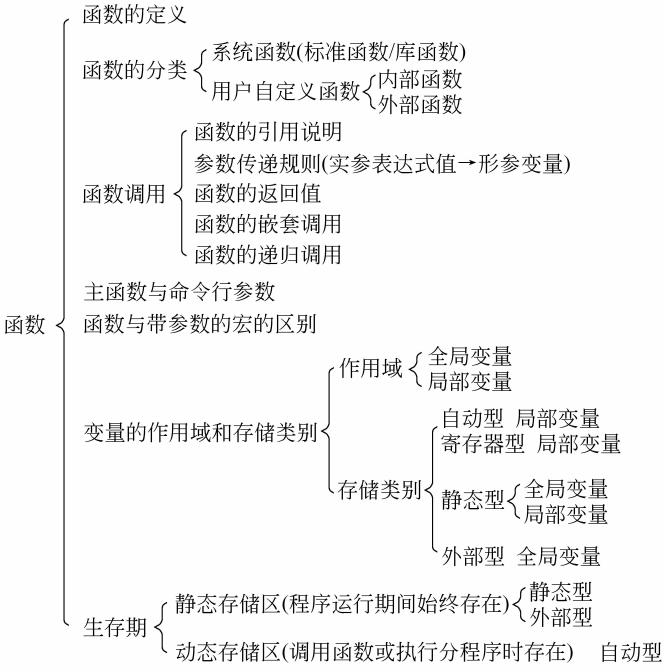

图 5.1 第 5 章知识结构

考核要求：熟练掌握函数的定义。熟练掌握函数的调用形式、参数传递规则、返回值类型和函数的递归调用。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、调用函数时数据的传递方法、变量的作用域和存储类别。本章的难点是函数参数值传递和地址传递的区别、递归函数设计。

核心考点：函数定义和调用、变量的作用域和存储类别。

5.2 典型题解析

【例 5.1】 以下所列的函数首部中，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：

```
存储类别　返回值类型　函数名(形参表列)
{ 函数体说明部分
  函数功能语句序列
}
```

若存储类别省略，则系统默认为 extern，即外部函数。若返回值类型省略，则系统默认为 int。形参表列的说明格式为：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 的形参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，则对函数 fun 正确调用的语句是(　　)。

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类型，故选项 A 和选项 D 是错误的。若函数返回值类型为“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 fun 函数有如下引用说明：

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

此说明的含义是（　　）。

A. fun 函数无返回值

B. fun 函数的返回值可以是任意的数据类型

C. fun 函数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：

```
存储类别 返回值类型 函数名(类型 1 形参名 1,类型 2 形参名 2,…,类型 n 形参名 n);
```

其中，形参名可以省略，写成

```
存储类别 返回值类型 函数名(类型 1,类型 2,…,类型 n);
```

若参数类型为 int 或 char 时，则参数类型也可以省略，写成

```
存储类别 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);
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;
}
```

程序运行后的输出结果是（　　）。

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。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)的操作，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);
}
main()
{ char b='a',a='A';
   fun(&b,a);  printf("%c,%c\n",b,a);
}
```

程序运行后的输出结果是(　　)。

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，再执行规定的操作，在函数中对形参的操作，不会影响到调用函数中的实参，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存储单元的值。

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

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

程序运行后的输出结果是(　　)。

A. 6　　B. 7　　C. 5　　D. 8

解析：一维数组名可以作为函数的参数，调用函数时把数组的首地址传递给形参，这

样实参数组和形参数组共占同一段内存，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，形参数组中元素 a[i]是实参数组中的元素 a[i+2]，因此，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

```
float f1(float n)
{ return  n*n;}
float f2(float n)
{ return  2*n;}
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);
}
```

程序运行后的输出结果是(　　)。

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，指向函数的指针变量的定义形式为：

```
函数返回值类型 (*指针变量名)(类型,…,类型);
```

若类型为 int，则类型可以省略，定义形式变为：

```
函数返回值类型 (*指针变量名)();
```

通过指向函数的指针调用函数的形式为：

```
(*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，其生存期将得到延长

B. 全局变量说明为 static 存储类，其作用域将被扩大

C. 任何存储类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说明符与局部变量完全相同

解析：若局部变量说明为 static 存储类，则该变量在静态存储区分配存储空间，所占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。若全局变量说明为 static 存储类，则该变量只能在定义它的文件内引用，连接在一起的其它文件不能使用，其作用域将被缩小，故选项 B 错误。static 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，系统为未初始化的变量赋以 0 值，故选项 C 错误。局部变量可以说明为 static 存储类，但形参不能，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;
}
main()
{ int i,s=1;
  for(i=1;i<=3;i++) s*=fun();
  printf("%d\n",s);
}
```

程序运行后的输出结果是(　　)。

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 被调用三次，每次调用前后 x 值的变化情况如下：

第一次调用：调用前，x＝1，调用后，x＝2；

第二次调用：调用前，x＝2，调用后，x＝4；

第三次调用：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

```
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++;
```

```
}
main()
{ int s=a, i;
  for(i=0;i<3;i++) s+=f(i);
  printf("%d\n", s);
}
```

程序运行后的输出结果是(　　)。

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重新赋初值，只是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 说明部分定义的变量 a 用 f_a 表示，if 后面复合语句中定义的变量 a 用 if_a 表示，else 后面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；

第一次调用：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。

第二次调用：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。

第三次调用：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 所指字符串中下标为奇数的字符按 ASCII 码值递增排序，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排序，所以第 i 趟排序的第一次比较应是 s[i]与 s[i＋2]比较，故②处应填写 i＋2。把排序后下标为奇数的字符存入 p 所指字符数组后，应在尾部加上字符串结束标志'\0'，故③应填写'\0'。

答案：① t＝i　　② i＋2　　③ '\0'

【例 5.13】 下列程序中，select 函数的功能是在 N 行 M 列的二维数组中找出最大值，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的行下标和列下标。请填空。

```
#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____;
}
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 的基本思路是：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](或 (* (a＋row)＋col)，或(a[row]＋col))。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)
③ '\0'

【例 5.14】 fun 函数的功能是：首先对 a 所指的 N 行 N 列的矩阵，找出各行中最大的数，再求这 N 个最大数中最小的那个数作为函数值返回。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 的基本思路是：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是临时最大数，即 max=a[row][0]；然后，用 max 与该行后面的 N－1 个元素依次比较，若 max 小，则用当前数组元素更新 max。如果当前行是第一行，则 min 的值就是 max 的值，否则，用当前行最大数 max 与 min 比较，若 max 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 的功能是在形参 s 所指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。

```
#include <stdio.h>
#include <string.h>
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 的基本思路是：从 s 所指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)，并将其后面字符(包括'\0')依次后移一位，然后将找到的字符 s[i]插入到空出的位置。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。

答案：① '\0'　② 0　③ i

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

解析：用筛选法得到小于等于 n 的所有素数的方法是：首先从数 2 开始，将所有 2 的倍数的数从数表中删去(把数表中相应位置的值置成 0)；接着从数表中找下一个非 0 数，并从数表中删去该数的所有倍数；依次类推，直到所找的下一个数等于 n 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(a[i]==0) i++;
  }
  for(i=2;i<=n; i++)
  if(a[i]!=0) count++;
  return  count;
}
```

【例 5.17】 假定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，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';
}
```