

实验四 函数实验

1.实验目的

- 1) 掌握 C 语言程序结构开发的基本要点，即自顶向下，逐步求精和模块化设计；
- 2) 掌握函数的定义方法和调用规则；
- 3) 掌握在 C 语言程序中主调函数和被调函数之间进行数据传递的规则。
- 4) 初步掌握多文件 C 语言程序的编译流程；
- 5) 初识文件操作。

2.实验相关知识

在 C 语言的程序设计中，无论多么复杂、规模多么大的程序，最终都落实到一个小而简单的函数编写工作上，因此，C 语言程序设计的基础工作是函数的设计和编制。一个 C 语言程序清单是由一个或多个函数定义组成的，包括系统提供的标准库函数和用户自定义的函数。

1) 函数的定义

函数定义的一般格式为：

```
<存储类型> <数据类型> 函数名 (<形式参数及说明>)
{
    说明语句;
    执行语句;
}
```

2) 函数的调用

a) 函数的原型（也称函数声明）

函数声明是指在主调函数中，对在本函数中将要被调用的函数提前做必要的声明。函数原型的一般格式为：

```
函数数据类型 函数名（参数类型 1，参数类型 2，.....）
（或 函数数据类型 函数名（参数类型 1 参数名 1，参数类型 2 参数名 2，.....））
```

b) 函数的调用一般格式：

```
函数名（实参表）；
```

3) 函数间的数据传递

C 语言在函数间传递数据有 4 中方式，值传递方式、地址传递方式、返回值方式、全局变量传递方式。其中：

- a) 值传递方式所传递的是参数值，其特点是“参数值的单向传递”。
- b) 地址传递方式所传递的是地址，其特点是“参数值的双向传递”。
- c) 返回值方式不是在形式参数和实际参数之间传递数据，而是通过函数调用后直接返回一个值到主调函数中。该函数的数据类型不能是 `void` 类型，且函数体中应有“`return<表达式>`”语句。
- d) 全局变量传递方式不是在形式参数和实际参数之间传递数据，而是利用在主调函数和被调函数中均有有效的全局变量，在主调函数和被调函数之间任意传递数据。

3.实验范例

1) 编写两个函数，分别求最大公约数 `gcd` (greatest common divisor) 和最小公倍数 `lcm` (least common multiple)。

① 程序分析：

最大公约数函数： `int gcd(int a,int b);`

最小公倍数函数： `int lcm(int a,int b,int g)`，其中 `int g` 为两个数的最大公约数。

② 参考源程序：

```
#include<stdio.h>
int gcd(int a, int b)
{
    int t, r;
    if (a < b)
    {
        t = a;
        a = b;
        b = t;
    }
    while (b != 0)
    {
        r = a%b;
        a = b;
        b = r;
    }
    return(a);
}

int lcm(int a, int b, int g)
```

```

{
    return(a*b / g);
}

void main()
{
    int m, n, g, l;
    scanf("%d,%d", &m, &n);
    g = gcd(m, n);
    l = lcm(m, n, g);
    printf("greatest common divisor:%d\n", g);
    printf("least common multiple:%d\n", l);
}

```

③ 运行结果：



```

10,56
greatest common divisor:2
least common multiple:280
请按任意键继续. . .

```

2) 编写一个判断素数 (prime number) 的函数。

① 程序分析：

用一个子函数来说明素数函数：int prime (int n)，有返回值，当返回值为 1 则为素数，当返回值为 0 则不是素数。

② 参考源程序：

```

#include<stdio.h>
#include<math.h>
int prime(int n)
{
    int i, flag = 1;
    for (i = 2; i <= (int)sqrt(n/1.0); i++)
        if (n%i == 0)
            flag = 0;
    return(flag);
}

void main()
{
    int n;
    printf("please input an integer:\n");
    scanf("%d", &n);
    if (prime(n))

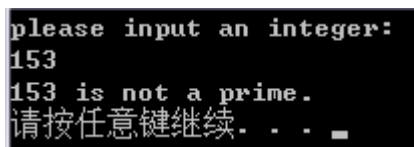
```

```

        printf("%d is a prime.\n", n);
    else
        printf("%d is not a prime.\n", n);
}

```

③ 运行结果：



```

please input an integer:
153
153 is not a prime.
请按任意键继续. . .

```

4.实验内容

1) 在图 1 所示的示意性菜单处理程序的基础上，编制完成一个菜单处理程序，每个菜单分别处理本实验内容 2) -5)。要求菜单处理代码在文件 menu.c 中，实验内容 2)、3) 的函数代码在 func1.c 中，实验内容 4)、5) 的函数代码在文件 func2.c 中。不允许在 menu 文件中通过#include 将文件 func1.c 和 func2.c 直接包含进来。在主函数中完成相关函数参数的输入及函数返回结果的输出。

2) 输入两个整数，调用函数 squSum()求两数平方和，返回主函数显示结果。

3) 求方程 $ax^2 + \sin x = 0$ 在 b 附近的一个实根，a 和 b 由键盘输入并作为函数的实参，函数返回计算得到的实根，采用牛顿迭代法求解该方程的根，牛顿迭代公式如下式所示：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

迭代过程结束的条件为根的变化小于 1e-6，上式中 $f'(x)$ 为 $f(x)$ 的导数。

4) 编制函数，完成以下定积分的计算，该函数的积分区间和分段数作为函数的输入参数，被积函数也要求用一函数实现：

$$\int_0^1 (x^2 + e^x \sin x) dx$$

在主函数中输入参数、调用该函数完成并输出计算结果。

5) 求方程 $ax^2+bx+c=0$ 的根，用三个函数分别求 b^2-4ac 大于零、等于零和小于零时的根，在主函数中输入参数，并输出结果，不得使用全局变量。

6) 编制程序，读入一图像文件，绘制该图像数据的灰度直方图。图像文件为群文件中的 image.dat。

7) 编写一个程序，从键盘连续读取字符，直到输入的字符为 '0'，对每个读入的字符，检查并报告是否是一个字母，如果是，还应报告该字母在字母表中的位置，例如，c 和 C 的字母位置都是 3，字符的判断及位置值返回均由一个函数完成。

8) 编写一个函数, 返回一个 int 数组中的最大值及其对应的下标值, 并在一个简单程序中测试这个函数。

9) 编写判断一个自然数是否为素数的函数, 以及判断该数是否含有数字 7 的函数, 在主函数中输出 1~200 之间含有数字 7 的素数。

例3.12

编一示意性的菜单处理程序, 要求按下一功能键, 执行响应的功能处理, 重复执行直到按ESC键退出。

将例3.5进行简单修改得到

```
#define ESC 0x1b;
#define F1 0x3b00
//F1键的键值为0x3b00
#define F2 0x3c00
#define F3 0x3d00
#define F4 0x3e00
#define F5 0x3f00
#define F6 0x4000
.....
#include <stdio.h>
#include <bios.h>
void main( )
{
    unsigned int key_value;
    while(1)
    {
        key_value = bioskey(0);
        if(key_value == ESC)
            break;
    }
}
```

跳出while循环

```
switch (key_value)
{ case F1: F1功能处理程序;
    break;
  case F2: F2功能处理程序;
    break;
  case F3: F3功能处理程序;
    break;
  case F4: F4功能处理程序;
    break;
  case F5: F5功能处理程序;
    break;
  case F6: F6功能处理程序;
    break;
  .....
  default: 相应处理程序;
    break;
}
printf(" ");
```

跳出switch结构

华中科技大学信息学院平台课—C语言程序设计

48

图 1 示意性菜单处理程序

/* 获取键值的函数, 在BC编译环境下, 该函数无需定义 */

```
unsigned short bioskey(int a)
{
    unsigned short ch;

    fflush(stdin);

    while (!_kbhit());

    ch = _getch();
    if (0 == ch || 224 == ch)
    {
        ch = _getch();
        ch = ch << 8;
    }
}
```

```
}  
  
return ch;  
}
```

附:

1、多文件 c 语言程序代码的编译

由多个文件构成的 C 语言程序的编译需要建立工程文件，将多个文件加入到工程文件，对工程文件进行编译处理即可。Codeblock 下工程文件的建立过程可观看 C 慕课第一周第 2 讲内容。

BC 编译器下工程文件的建立请阅读群文件 BC31.pdf 中第 26 页“建立工程文件”

2、图像文件的读入:

文件的读写是通过文件指针进行的，定义说明文件指针的一般形式为:

FILE * 指针变量标识符;

其中 FILE 应为大写，如下例:

```
FILE *fp;
```

fp 就是所定义的文件指针变量，也可用其它变量名。

文件在进行读写操作之前要先打开，使用完毕要关闭。所谓打开文件，实际上是使文件指针指向该文件，以便进行其它操作。关闭文件则断开指针与文件之间的联系，也就禁止再对该文件进行操作。

文件打开通过调用 fopen 函数实现，其调用的一般形式为:

if((文件指针名= fopen(文件名, 使用文件方式))==NULL)
--

其中，“文件指针名”必须是被说明为 FILE 类型的指针变量，“文件名”是被打开文件的文件名，“使用文件方式”是指文件的类型和操作要求，“文件名”是字符串常量或字符串数组。

例如: FILE *fp;

```

if((fp=fopen("d:\\image.dat","rb"))==NULL)
{
    printf("File Open Error!\n");
    return;
}

```

其意义是检查文件指针 `fp` 所指向的文件存在否，如果存在，则打开 `d` 盘根目录下的 `image.dat` 文件，只允许进行"读"操作，并使 `fp` 指向该文件。两个反斜线 `\\` 为转义字符，表示字符 `\`。如果文件打开出错的化，就返回空指针，需要进行文件出错处理，通常退出程序。

使用文件的方式共有 12 种，具体含义见表 1。

表 1 C 语言中使用文件的方式

文件使用方式	意义
"rt"	只读打开一个文本文件，只允许读数据
"wt"	只写打开或建立一个文本文件，只允许写数据
"at"	追加打开一个文本文件，并在文件末尾写数据
"rb"	只读打开一个二进制文件，只允许读数据
"wb"	只写打开或建立一个二进制文件，只允许写数据
"ab"	追加打开一个二进制文件，并在文件末尾写数据
"rt+"	读写打开一个文本文件，允许读和写
"wt+"	读写打开或建立一个文本文件，允许读写
"at+"	读写打开一个文本文件，允许读，或在文件末追加数据
"rb+"	读写打开一个二进制文件，允许读和写
"wb+"	读写打开或建立一个二进制文件，允许读和写
"ab+"	读写打开一个二进制文件，允许读，或在文件末追加数据

对于文件使用方式有以下几点说明：

(1) 文件使用方式由 `r,w,a,t,b,+` 六个字符拼成，各字符的含义是：

`r(read)`: 读

`w(write)`: 写

`a(append)`: 追加

`t(text)`: 文本文件，可省略不写

`b(binary)`: 二进制文件

`+`: 读和写

(2) 凡用"r"打开一个文件时，该文件必须已经存在，且只能从该文件读出。

(3) 用"w"打开的文件只能向该文件写入。若打开的文件不存在，则以指定的文件名建立该文件，若打开的文件已经存在，则将该文件删去，重建一个新文件。

(4) 若要向一个已存在的文件追加新的信息，只能用"a"方式打开文件。但此时该文件必须是存在的，否则将会出错。

(5) 在打开一个文件时，如果出错，fopen 将返回一个空指针值 NULL。在程序中可以用这一信息来判别是否完成打开文件的工作，并作相应的处理。因此常用以下程序段打开文件：

```
if((fp = fopen("c:\\hzk16.dat","rb"))==NULL)
    //检查能否打开 hzk16.dat 文件
{
    printf("\n error on open c:\\hzk16.dat file!\n");
    exit (1);                //退出
}
```

这段程序的意义是，如果返回的指针为空，表示不能打开 C 盘根目录下的 hzk16 文件，则给出提示信息"error on open c:\\hzk16.dat file!"，下一行 exit(1)退出程序。

(6) 把一个文本文件读入内存时，要将 ASCII 码转换成二进制码，而把文件以文本方式写入磁盘时，也要把二进制码转换成 ASCII 码，因此文本文件的读写要花费较多的转换时间。对二进制文件的读写不存在这种转换。

由于图像文件是已存在的二进制文件，只需要对其进行读操作，所以采用"rb"方式打开。

文件正确打开之后，就可用读入图像数据，所给的图像文件中，最前面两个数据为短整型量，分别为图像的行和列数，读入二进制文件采用 fread 函数，

函数 fread() 的原型如下所示：

<pre>size_t fread(void *ptr, size_t size, size_t n, FILE *fp);</pre>
--

对于函数 fread() 而言，形参变量 ptr 是存放所读入数据的内存地址，形参变量 size 为每个数的长度，类型为 size_t，一般代表无符号整数，可用通过 sizeof

(数据类型名) 获得单个数据的长度, n 为读入数据的个数。

在图像文件 image.dat 中, 头四个字节记录了图像的行数和列数, 各占 2 各字节, 读入行列值可采用以下语句:

```
fread(&row, sizeof(short), 1, fp);
```

```
fread(&col, sizeof(short), 1, fp);
```

row 和 col 为存储行数和列数的短整型变量。

文件中, 紧接行列数后面的是图像上每个像素位置上的灰度值, 像素位置由行值 i 和列值 j 来确定。在所给定的图像文件中, 每个像素灰度值为 unsigned char 类型, 灰度值的取值范围为 0~255, 通过遍历所有像素位置, 顺序读取每个像素的灰度值, 遍历可通过以下的二重循环完成, 读入的像素值存入到变量 g 中:

```
for(i=0; i<row; i++)
{
    for(j=0; j<col; j++)
    {
        fread(&g, sizeof(unsigned char), 1, fp);    //单个像素值的读入、
        统计、
        每个区间的计数等处理
    }
}
```

每个像元值可用 fread 读取, 注意每个数据只占 1 个字节长度, 需要定义 unsigned char 类型变量存储。

文件读入并处理完之后, 需要关闭文件, 文件的关闭通过调用 fclose 函数完成, fclose 函数调用的一般形式是:

```
fclose(文件指针);
```

例如: fclose(fp);

其中 fp 是已定义过的文件指针。该函数在关闭前清除与文件有关的所有缓冲区, 正常完成关闭文件操作时, fclose 函数返回值为 0。如返回非零值则表示有错误发生。

在循环读取图像像素的过程中, 可采用下节的描述进行直方图的统计, 然后

根据统计数据绘制对应的灰度直方图。

3、灰度直方图的绘制

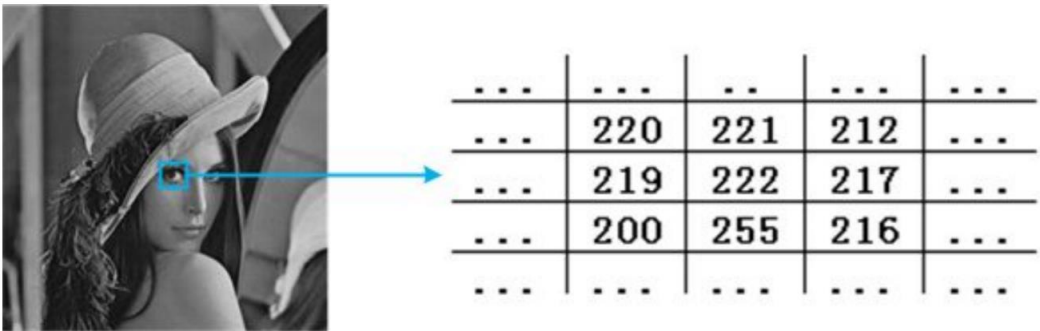
灰度直方图（histogram）描述的是图像中落在每一灰度区间中像素个数，它可以在一定程度上反映图像的特性。直方图中，横坐标是划分的灰度区间，纵坐标是灰度值落在对应灰度区间里的像素个数。实验中，我们将 0~255 的灰度范围划分为 10 个灰度区间，统计灰度值落在每个区间的像素个数，对于每个灰度区间，用横向的*的个数表示不同的像素个数，绘制如下的直方图形，图形中，左侧数字表示统计区间的灰度范围，1 个*表示 200 个像素数，*号的个数最多不超过 30，超出部分则不绘制。

```
0 - 24 | *
25 - 49 | **
50 - 74 | ***
75 - 99 | *****
100 - 124 | *****
125 - 149 | *****
150 - 174 | *****
175 - 199 | *****
200 - 224 | *****
225 - 255 | *****
```

4、有关图像的一些基本概念

1) 数字图像

数字图像是二维图像用像素值的表示。在数学的领域，图像以矩阵的形式进行定义，以下就是数字图像的矩阵表示示意图：



2) 像素

像素又叫像元，是数字图像的基本元素。像素在图像中有两个基本属性，像素值和坐标。像素值对应着像素的灰度值或者颜色值，坐标对应着像素在数字图像中的位置。

3) 图像坐标

在解析几何当中，通常我使用左下角为原点建立二维笛卡尔坐标系，但是，在数字图像中，我们通常以图像的左上角为原点建立图像坐标系，对像素位置进行描述。如下图所示，x 轴表示列方向，y 轴表示行方向。



4) 位宽

在图像处理中我们经常遇到几位几位图像这一说，它的意思是说，一个像素我们使用多少个比特位进行描述。比如，灰度图像经常使用 8 位进行存储，那么它的每一个像素在内存中对应着 8 个比特位，其位宽为 8。

5) 通道

又叫颜色通道，表明一个像素对应着多少个像素值。比如，8 位灰度图像，就是一（单）通道图像，它的每一个像素对应着一个 0-255 的灰度值；24 位真彩图像，就是 3 通道图像，它的每一个像素需要 (R,G,B) 三个颜色值进行描述。我们实验用的数据为 8 位灰度图像。

5、随机数的产生

首先需要调用函数 `randomize()` 对随机数发生器进行初始化，然后调用 `random` 函数产生随机数。如下语句：

```
randomize();  
a = random(num);
```

在上面的语句中，首先初始化随机发生器，然后产生一个 0~num-1 范围里的随机数赋给 a。

使用以上两个函数时，需要将 time.h 和 stdlib.h 文件包含进来。

上述方法只能在 BC 编译器下使用，更一般的随机数产生方法是使用 srand() 和 rand()函数。首先采用如下语句调用 srand 对随机数发生器进行初始化：

```
srand((unsigned)time(NULL));
```

然后调用以下语句产生一个 0~num-1 范围里的随机数给 a：

```
a=rand()%num;
```

使用上面两个函数同样需要在源代码中将 time.h 和 stdlib.h 文件包含进来。

6、定积分的定义与近似计算

定积分定义：设函数 $f(x)$ 在区间 $[a,b]$ 上连续，将区间 $[a,b]$ 分成 n 个子区间 $[x_0,x_1], (x_1,x_2], (x_2,x_3], \dots, (x_{n-1},x_n]$ ，其中 $x_0=a$ ， $x_n=b$ 。可知各区间的长度依次是： $\Delta x_1=x_1-x_0$ 。在每个子区间 $(x_{i-1},x_i]$ 中任取一点 $\xi_i (1,2,\dots,n)$ ，作和式

$$\sum_{i=1}^n f(\xi_i) \Delta x_i$$

该和式叫做积分和，设 $\lambda=\max\{\Delta x_1, \Delta x_2, \dots, \Delta x_n\}$ （即 λ 是最大的区间长度），如果当 $\lambda \rightarrow 0$ 时，积分和的极限存在，则这个极限叫做函数 $f(x)$ 在区间 $[a,b]$ 的定积分，记为

$$\int_a^b f(x) dx$$

并称函数 $f(x)$ 在区间 $[a,b]$ 上可积。其中： a 叫做积分下限， b 叫做积分上限，区间 $[a, b]$ 叫做积分区间，函数 $f(x)$ 叫做被积函数， x 叫做积分变量， $f(x)dx$ 叫做被积表达式， \int 叫做积分号。之所以称其为定积分，是因为它积分后得出的值是确定的，是一个常数，而不是一个函数。

根据上述定义，若函数 $f(x)$ 在区间 $[a,b]$ 上可积分，则可以采用 n 等分的特殊分法进行近似计算，如下图所示：

$$\int_a^b f(x)dx \approx \sum_{i=1}^n f(a + (i-1) * h + \frac{h}{2}) \times h$$

其中, $h = \frac{b-a}{n}$ 。

