《数字图像处理概论》大作业

作 业 名 称： 图像处理系统软件

专　　业 软件工程

组　序　号 09

组 长 姓 名 王嘉林

组 长 学 号 2021211968

联 系 方 式 17639376836

成绩		评阅人	夏勇

1	2	3	4	5	6

学号	姓名	贡献权重	成绩
		1	
		0.95	

哈尔滨工业大学计算机学院

报告正文

0. 小组分工介绍

组长王嘉林：负责实验 1、2、5、6、8、9、10 的完成和验收，批处理程序的实现以及报告的撰写；

组员陈硕：负责实验 3、4、7 的完成和验收。

1. 系统设计 （5 分）

1.1 总体设计描述

本次程序设计参考《数字图像处理概论》的课堂内容，结合 PPT 文件知识以及网络上的讲解，完成了对 BMP 文件的简单处理、生成直方图、处理空间域滤波、图像变换、阈值分割、基于区域的分割、边缘检测、霍夫变换、区域标记和轮廓提取等数字图像处理任务，并设计完成了菜单和批处理功能，同时处理多个图片、输出多个图片结果。

1.2 编程平台与语言

本程序设计使用 Clion 作为集成开发环境，使用 C 语言作为编程语言，并用 Cmake 作为编译工具进行开发。

1.3 交互方式介绍及主菜单界面截图

如图 1，运行程序后，通过输入需要进行的程序标号来执行对应的图像处理任务，输入异常时提示错误并且等待用户重新输入。

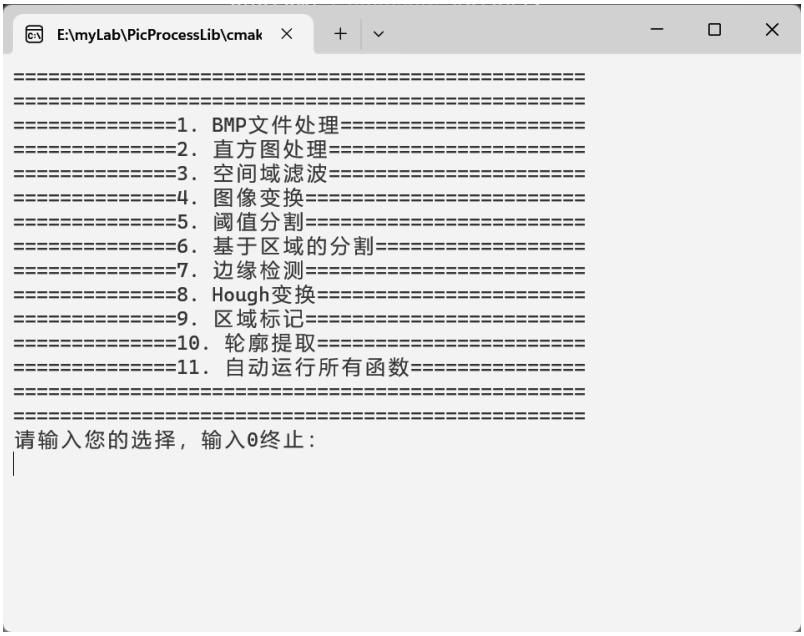

图 1 主菜单界面截图

输入某一标号后，进入目标程序的二级页面（以 1. BMP 图像处理为例）。接下来继续输入编号，即可完成对应的图像处理任务。

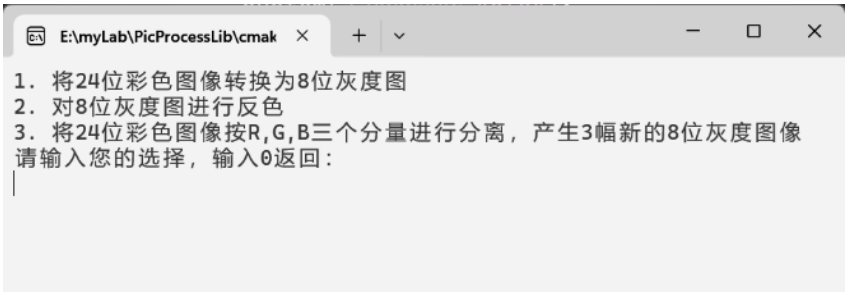

图 2 菜单二级页面实例

1.4 批处理参数设置方式介绍及运行界面截图

批处理时无需输入参数（如图 3），当无参数输入时，函数调用默认参数。由于 C 不支持自动默认参数，这里在每个函数的开始加入无参情况的考虑（如图 4，以 1. BMP 图像处理为例）。运行界面截图见图 5。

```
system( Command: "cls");
printf( format: "下面以默认参数运行所有图像处理函数：\n");
grayProcess( srcPath: NULL, dstPath: NULL);
printf( format: "1.1success!\n");
reverseProcess( srcPath: NULL, dstPath: NULL);
printf( format: "1.2success!\n");
depatureRGB( srcPath: NULL, dstPathB: NULL, dstPathG: NULL, dstPathR: NULL);
printf( format: "1.3success!\n");
makeHistogram( srcPath: NULL, dstPath: NULL);
printf( format: "2.1success!\n");
histogramEqualization( srcPath: NULL, dstPath: NULL);
printf( format: "2.2success!\n");
```

图 3 批处理程序

```
void grayProcess(char* srcPath, char* dstPath) {
    // 默认处理
    if(srcPath == NULL){
        srcPath = "../img/rgb.bmp";
    }
    if(dstPath == NULL){
        dstPath = "../lab1/img/GrayRGB.bmp";
    }
```

图 4 无参情况时的函数处理

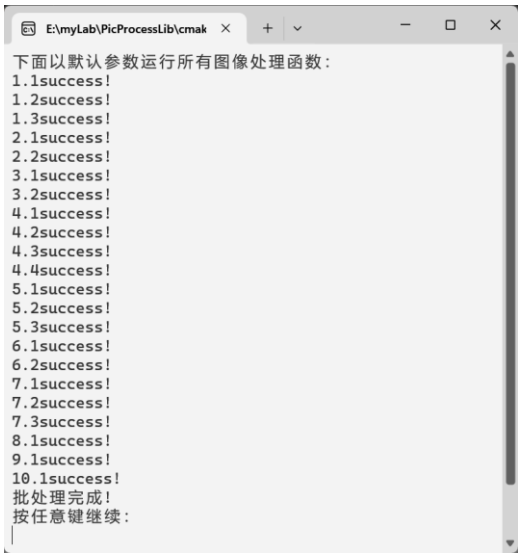

图 5 批处理程序运行界面

2. 功能模块设计与实现 （40 分）

2.1 文件处理

1) 将 24 位彩色图像转换为 8 位灰度图

- 基本原理
  按照 BMP 的图像格式读取图片，按照 RGB 色彩图像变为灰度图的公式对图像数据进行计算，修改图片的文件头、信息头，编写调色板，最后将文件头、信息头、调色板、修改后的图像数据按顺序写入 BMP 图片中，即获得了转换后的 8 位灰度图。
- 输入图像

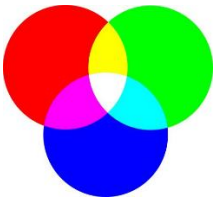

图 6 rgb 三通道图像

- 结果图像

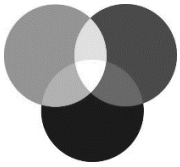

图 7 灰度处理后的 rgb 图像

- 结果分析
  成功将 24 位真彩色图像转换为了 8 位灰度图，较好的完成了实验要求。

2) 对 8 位灰度图进行反色

- 基本原理
  对每个像素 p，将 255-p 的值保存在原像素的位置中。
- 输入图像
  见图 7。
- 结果图像

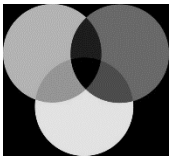

图 8 将图 7 反色后的结果

- 结果分析
  成功地对 8 位灰度图进行反色，较好的完成了实验要求。

3) 将 24 位彩色图像按 R、G、B 三个分量进行分离，产生三幅新的 8 位灰度图像

- 基本原理
  读取 rgb 真彩色图片，对于每个像素，将用于表示每个颜色的三个字节分别取出，存在相应的位置中，即分离了 R、G、B 三个通道的颜色。

- 输入图像
  见图 6。
- 结果图像

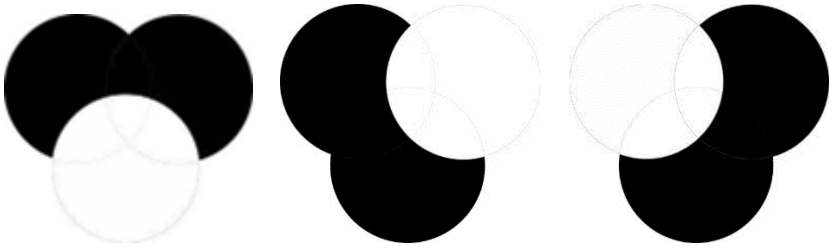

图 9 B、G、R 三通道地分离结果

- 结果分析
  成功将 24 位彩色图像按 R、G、B 三个分量进行分离，产生三幅新的 8 位灰度图像，较好的完成了实验要求。

2.2 直方图处理

1) 直方图统计

- 基本原理
  遍历所有的像素点，统计每个灰度值出现的次数，根据出现次数的最大值进行归一化处理，之后新建文件头、信息头、调色板，并将其与灰度值统计信息一起保存在 BMP 图片中，得到图像的直方图统计结果。
- 输入图像

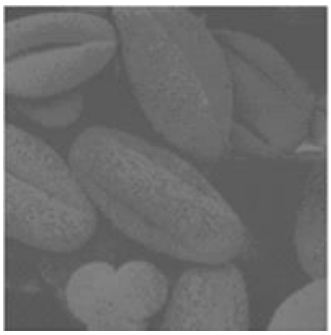

图 10 输入图像 dim

- 结果图像

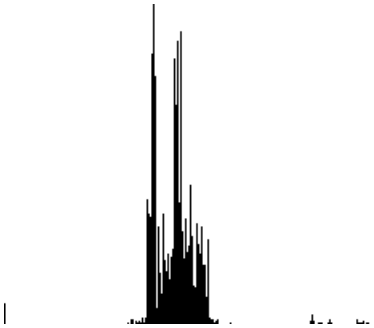

图 11 图 10 的直方图

- 结果分析
  输出了正确的直方图结果，较好的完成了实验要求。

2) 直方图均衡化

- 基本原理

  基于公式：

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{n}$$

  对直方图进行均衡化，并且调用直方图统计中的函数输出均衡化后的直方图。

- 输入图像

  见图 10。

- 结果图像

图 12 均衡化后的图像 dim

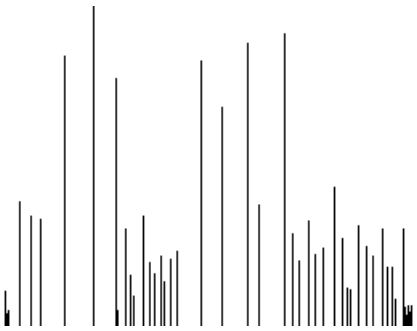

图 13 均衡化后的直方图

- 结果分析

  完成了直方图的均衡化，较好的完成了实验要求。

2.3 空间域滤波

1) 平均处理

- 基本原理

  使用卷积核

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

  对图像进行卷积处理，得到的结果即为平均处理结果。

- 输入图像

图 14 lena

- 结果图像

图 15 平均滤波后的结果

- 结果分析

  可以看出，图 15 进行了模糊化处理，较好的完成了实验要求。

2) 中值滤波

- 基本原理

  取 3×3 作为模板大小，取 9 个数中的中位数作为居中值的替代，即为中值滤波的过程。

- 输入图像

图 16 有椒盐噪声的 lena

- 结果图像

图 17 中值滤波处理后的结果

- 结果分析

  从图 17 中可以看出，中值滤波消除了较多的椒盐噪声，较好的完成了实验要求。

2.4 图像变换

1) 图像缩放

- 基本原理
  设置缩小倍率 T 后，设置新图像大小为之前的 $1/T^2$，在新图像的[i,j]位置赋值为原图像[T×i,T×j]的像素值，即实现了图像缩小；对于填充，则将空白像素填充为附近的像素值。
- 输入图像
  见图 14。
- 结果图像

图 18 缩小后的 lena

- 结果分析
  以缩小为例，实现了图像的缩放，完成了实验要求。

2) 图像平移

- 基本原理
  设置平移的 x、y 后，将新图像的[i,j]的像素值赋值为原图像[i-x,j-y]的像素值，未处理到的部分设置为白色。
- 输入图像
  见图 14。
- 结果图像

图 19 平移后的 lena

- 结果分析
  实现了图像的平移，较好的完成了实验要求。

3) 图像镜像

- 基本原理
  按照反方向顺序读取原图像并且赋值给新图像，即可得到图像的镜像结果。
- 输入图像
  见图 14。

- 结果图像

图 20 **左右镜像的** lena

图 21 **上下镜像的** lena

- 结果分析

  完成了图像的镜像，较好的完成了实验要求。

4) 图像旋转

- 基本原理

  以图像中心作为旋转中心，设旋转角度为 θ，旋转前的点坐标为(x_0,y_0)，则旋转后的点的坐标(x,y)满足：

$$\begin{cases} x = r\cos(\alpha-\theta) = r\cos\alpha\cos\theta + r\sin\alpha\sin\theta = x_0\cos\theta + y_0\sin\theta \\ y = r\sin(\alpha-\theta) = r\sin\alpha\cos\theta - r\cos\alpha\sin\theta = -x_0\sin\theta + y_0\cos\theta \end{cases}$$

- 输入图像

  见图 14。

- 结果图像

图 22 **旋转后的** lena

- 结果分析

  实现了任意角度的图像的旋转，较好的完成了实验要求。

2.5 阈值分割

1) 给定阈值 T

- 基本原理

  对于每个像素，若像素值大于 T 则置为 255，若像素值小于 T 则置为 0。

- 输入图像

  见图 14。

- 结果图像

图 23 分割后的 lena

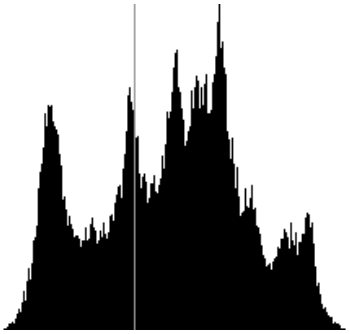

图 24 T 值在直方图的显示

- 结果分析

可以看到给定阈值 T 的分割效果较差。

2) 迭代阈值法

- 基本原理

将灰度中值最为初始阈值，将图像分为两部分，分别计算其灰度均值， μ_1、 μ_2，再利用公式

$$T_{i+1} = \frac{1}{2}(\mu_1 + \mu_2)$$

获得新的阈值，直至两次阈值之差小于某个固定值时迭代结束。

- 输入图像

见图 14。

- 结果图像

图 25 迭代阈值法分割后的 lena

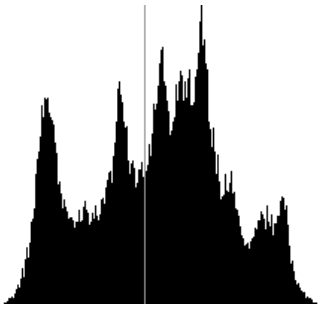

图 26 迭代阈值法分割的 T 值

- 结果分析

迭代阈值法的 T 分割效果更好。

3) Otsu

- 基本原理

遍历 0-255 的每个值，使其作为阈值分割 T，找到某个 T 使得类间方差

$$\sigma_B^2 = \omega_1\omega_2(\mu_1 - \mu_2)^2$$

最小，则该 T 为较好的分割值。

- 输入图像

见图 14。

- 结果图像

图 27 Otsu 法分割后的 lena

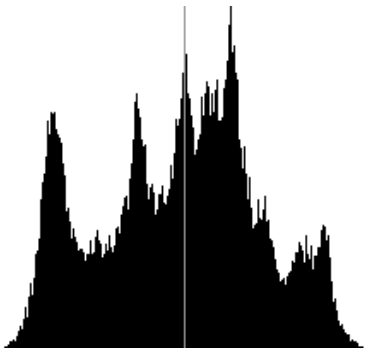

图 28 Otsu 法分割的 T 值

- 结果分析
  对比可知，Otsu 的分割效果最好。

2.6 基于区域的分割

1) 基于种子点进行区域增长

- 基本原理
  选定种子点，将种子点入队列，使用深度优先搜索的迭代形式进行求解，对队列首部的元素出队，判断其是否访问过，以及其与上、下、左、右四个点的像素差值是否小于阈值 T，若小于则将对应点入队列，队列为空时迭代结束。
- 输入图像

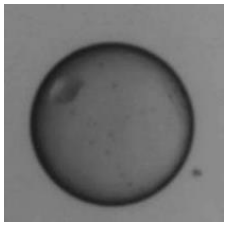

图 29 用于区域增长的图像

- 结果图像

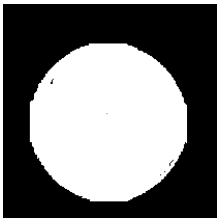

图 30 区域增长后的结果（种子点在最中间处）

- 结果分析
  实现了区域增长的功能，输出了正确的结果。

2) 区域分裂

- 基本原理
  记录当前区域左上角、右下角的两个点，统计当前区域的极差，若极差大于某个阈值 T，

则对图像进行四等分，并递归调用程序。

- 输入图像

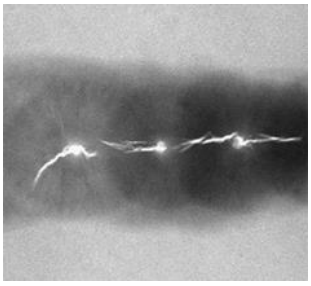

图 31 用于区域分割的图像

- 结果图像

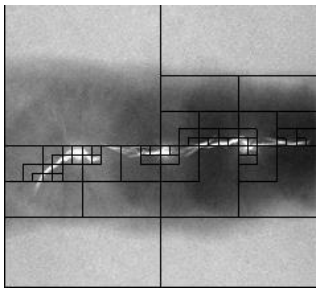

图 32 区域分割的结果

- 结果分析
  输出了正确的结果，较好的完成了实验要求。

2.7 边缘检测

1) Prewitt 法

- 基本原理
  使用卷积核

$$\begin{matrix}-1 & -1 & -1\\ 0 & 0 & 0\\ 1 & 1 & 1\end{matrix} \text{ 以及 } \begin{matrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{matrix}$$

对原图像进行卷积处理，之后对结果进行阈值化处理，得到图像边缘。

- 输入图像
  见图 14。
- 结果图像

图 33 Prewitt 边缘检测结果

- 结果分析
  由图可知，Prewitt 法效果较差。

2) Sobel 法

- 基本原理

  使用卷积核

$$\begin{matrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{matrix} \text{ 以及 } \begin{matrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{matrix}$$

  对原图像进行卷积处理，之后对结果进行阈值化处理，得到图像边缘。

- 输入图像

  见图 14。

- 结果图像

图 34 Sobel **边缘检测结果**

- 结果分析

  由图可知，结果相较于 Prewitt 略好一些，但效果仍然一般。

3) Log 法

- 基本原理

  使用 5×5 的卷积核

$$\begin{bmatrix}0 & 0 & -1 & 0 & 0\\ 0 & -1 & 2 & -1 & 0\\ -1 & -2 & 16 & -2 & -1\\ 0 & -1 & -2 & -1 & 0\\ 0 & 0 & -1 & 0 & 0\end{bmatrix}$$

  对原图像进行卷积处理，之后对结果进行阈值化处理，得到图像边缘。

- 输入图像

  见图 14。

- 结果图像

图 35 LOG **边缘检测结果**

- 结果分析

  对比之下，使用 LOG 方法边缘检测的效果最好。

2.8 霍夫变换

1) 直线检测

- 基本原理

  遍历整个图像，对每个需要检测的像素值记录它所确定的直线参数，并记录在霍夫空间中，每遇到一个直线参数则在对应的空间位置加 1。若最终霍夫空间的某个值大于阈值 T，则绘制出该直线。

- 输入图像

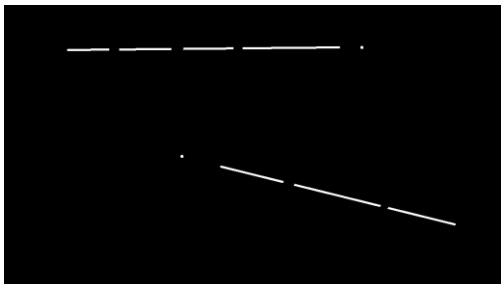

图 36 需要进行霍夫变换的图像

- 结果图像

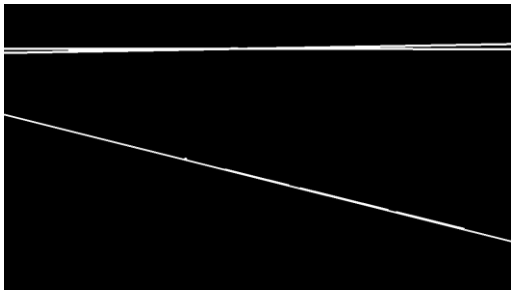

图 37 霍夫变换后的结果

- 结果分析

  输出了正确的图像结果，较好的完成了实验要求。

2.9 区域标记

1) 将每个联通区域用一个不同的颜色进行表示，以便于展示结果。

- 基本原理

  遍历所有的像素点，对不是背景的点入队，使用深度优先搜索的迭代形式进行求解：对队列首部的元素出队，判断其是否访问过，以及其与上、下、左、右四个点的像素值是否相等，若相等则将对应点入队列，并且标记为相同区域，当队列为空时迭代结束。

- 输入图像

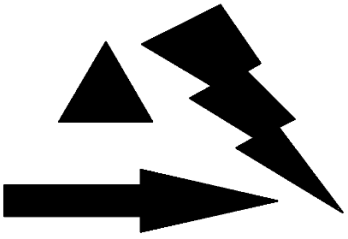

图 38 用于区域标记的图像

- 结果图像

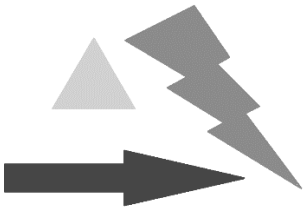

图 39 区域标记后的图像

- 结果分析
  输出了区域标记后的图像，较好的完成了实验要求。

2.10 轮廓提取

1) 基于边界点的定义提取轮廓

- 基本原理
  遍历所有的像素点，当某一点不为背景，且四周 8 个点均为内容时，本点也为内容，否则该点即为轮廓边缘。
- 输入图像

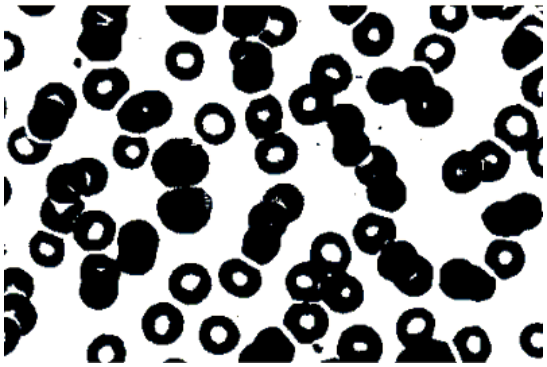

图 40 用于轮廓提取的图像

- 结果图像

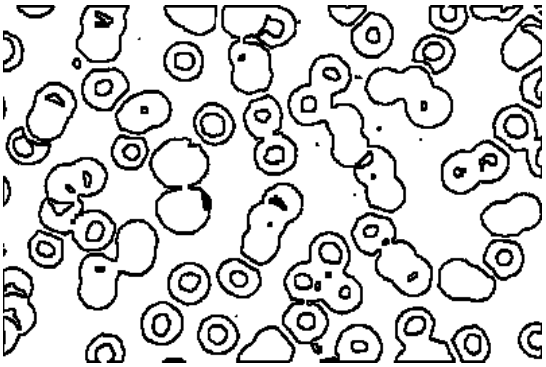

图 41 轮廓提取后的图像

- 结果分析
  由图，轮廓提取的效果相对很好。

3. 总结 （5 分）

通过本次《数字图像处理概论》课程的学习以及 10 个实验的完成，我对图像的本质、尤其是 BMP 图像的处理有了更深的认识，学习了 BMP 格式图像的存储形式，并且掌握了一些常用的图像处理方法；与此同时，在 10 次编程作业中，通过大量的动手实践，我对 C 语言的使用有了更深的理解，还巩固了所学习过的算法知识，大大提高了编程能力。

4. 格式评价（无需填写内容，教师专用） （5 分）

5. 程序评价（无需填写内容，教师专用） （45 分）