

**程 序 设 计 专 题**

**大 程 序 报 告**



1. **姓名 ： 王佳宁 学号： 电话：\_\_\_\_**
2. **姓名 ： 蒋昀昊 学号： 电话：\_\_\_\_**
3. **姓名 ： 秦子昂 学号： 电话：\_\_\_\_**

**指导老师： 肖少拥老师**

**2019~2020春夏学期 2020 年 6 月 20 日**

**目 录**

[1 大程序简介 1](#_Toc41775283)

[1. 选题背景及意义 1](#_Toc41775284)

[2. 目标要求 1](#_Toc41775285)

[3. 术语说明 1](#_Toc41775286)

[2 功能需求分析 1](#_Toc41775287)

[3 程序开发设计 2](#_Toc41775288)

[1. 总体架构设计 2](#_Toc41775289)

[2. 功能模块设计 3](#_Toc41775290)

[3. 数据结构设计 3](#_Toc41775291)

[4. 源代码文件组织设计 3](#_Toc41775292)

[5. 函数设计描述 5](#_Toc41775293)

[4 部署运行和使用说明 10](#_Toc41775294)

[1. 编译安装 10](#_Toc41775295)

[2. 运行测试 10](#_Toc41775296)

[3. 使用操作 10](#_Toc41775297)

[5 团队合作 11](#_Toc41775298)

[1. 任务分工 11](#_Toc41775299)

[2. 开发计划 11](#_Toc41775300)

[3. 编码规范 11](#_Toc41775301)

[4. 合作总结 12](#_Toc41775302)

[5. 收获感言 12](#_Toc41775303)

[6 参考文献资料 13](#_Toc41775304)

**小型算法流程图绘制工具大程序设计**

# 大程序简介

## 选题背景及意义

随着学习的不断深入，我们所设计的程序架构越来越大，而这种情况下，算法流程图的绘制就显得越来越重要。尤其是大程序的设计，算法流程图更是详细设计文档所必须包含的内容。通过流程图来了解模块的设计，对于理解程序有很大的帮助；在开始代码实现之前使用流程图，也有助于理清思路，便于确定实现的步骤与方法。

选择这个课题，一来可以锻炼我们的团队协作的能力、提高程序设计的水平，二来也有助于我们更深层次地理解流程图与算法。

## 目标要求

主要目标：实现简单流程图的绘制、编辑与模拟运行，并提供保存和读取功能。

## 术语说明

1. 图元：本篇报告中用于指代流程图中代表动作的各图形以及内部文本，e.g.菱形、平行四边形及其内部语句。
2. 工作流：指用户完成一个基础功能操作所对应的整个操作流程与程序的内部运算

# 功能需求分析

功能需求：

|  |  |  |
| --- | --- | --- |
| 菜单系统 | 图标工具栏 | 文件：新建、打开、保存、关闭、退出等 |
| 编辑：选择、复制、粘贴、删除等 |
| 绘图：矩形、圆角矩形、菱形等 |
| 帮助：关于、使用说明等 |
| 快捷键 | 工具栏各功能对应的快捷键 |
| 显示系统 | 状态信息栏 | 窗口底部即时显示操作 |
| 主功能系统 | 绘图 | 图形绘制，文本框输入编辑 |
| 编辑 | 拖拽，复制，粘贴 |
| 模拟执行过程 | 通过流程图模拟执行程序 |
| 文件系统 | 文件读取 | 将图源信息保存至文本文件；从文件中读取图形 |

业务逻辑需求：

1. 绘图：选择绘图位置-选择语句类型-选择图形-执行绘图操作产生图元-相邻图元连接
2. 编辑：选择图元-确定操作类型-执行操作-结束编辑
3. 模拟执行：启动模拟执行-从start开始扫描图元-识别语句类型-进行语句处理-识别到end-清空缓存-结束模拟执行
4. 文件保存：从start开始扫描图元-检查是否有对应txt文件-有则打开，无则新建-将图元信息以一定格式输出至对应txt文件-识别到end-结束保存
5. 文件读取：识别开始读取关键字-依格式读入文本-转换为图元信息进行存储-识别到end并输入-结束读取
6. 状态信息显示：检测到用户操作-识别操作事件-输出对应文本

数据结构需求：

1. 图元信息存储：使用多分支的链表来存储流程图的结构信息，使用链表存储图元的信息。
2. 模拟执行过程中，流程图内变量存储：变量使用链表存储。

# 程序开发设计

## 总体架构设计

## 功能模块设计

1. 图元绘制模块：包括位置选择与图形绘制，用于开始绘制工作流以及完成图元最终的建立与插入
2. 工具栏模块：浏览工具栏，开始文件存储和读取、模拟执行的工作流以及对图元的整体编辑，完成绘制工作流中图形和语句类型的选择过程
3. 模拟执行模块：用于完成模拟执行工作流的内部运算
4. 信息栏模块：用于显示用户操作日志
5. 文件操作终端：用户用于输入文件名
6. 弹窗功能：使用MFC编写了新的图形从而实现了弹窗。

## 数据结构设计

1. 图元信息的存储：为了便于遍历绘图，选择链式存储。同时由于存在选择结构，图元的后继可能存在两个，所以图元信息最终以每个节点至多两个分支的类链式结构存储每个图元信息。
2. 模拟执行过程中变量的存储：考虑查找、赋值与修改的需要，以链表形式存储。

## 源代码文件组织设计

* + 1. 文件函数结构

本程序除了使用libgraphic与simpleGUI包外，还包含下列文件：

display.h：包含display函数声明

display.c：包含display函数的实现

draw\_button.h：包含draw\_button函数声明

draw\_button.c：包含draw\_button函数的实现

draw\_figure\_extemsion.h：包含drawTriangle，drawParallelogram，drawDiamond三个函数的声明

draw\_figure\_extension.c：包含drawTriangle，drawParallelogram，drawDiamond三个函数的实现

draw\_log.h：包含add\_log和draw\_log函数的声明

draw\_log.c：包含add\_log和draw\_log函数的实现

draw\_menu.h：包含draw\_menu函数的声明

draw\_menu.c：包含draw\_menu函数的实现

draw\_terminal.h：包含is\_terminal\_input，terminal\_row与terminal\_ data的外部变量声明和函数draw\_terminal，add\_terminal与get\_last\_ terminal的声明

draw\_terminal.c：包含is\_terminal\_input，terminal\_row与terminal\_ data的变量定义和函数draw\_terminal，add\_terminal与get\_last\_ terminal的实现

enum.h：包含类figure与sentence的声明和type\_choose的外部变量声明

enum.c：包含变量type\_choose的定义

file\_operate.h：包含is\_file\_open，fp，to\_open\_file，to\_new\_file的外部变量声明和函数open\_a\_file，new\_a\_file与close\_a\_file的声明

file\_operate.c：包含is\_file\_open，fp，to\_open\_file，to\_new\_file的变量定义和函数open\_a\_file，new\_a\_file与close\_a\_file的实现

generate\_id.h：包含genid的外部变量声明和函数generate\_id的声明

generate\_id.c： 包含genid的变量定义和函数generate\_id的实现

inputbox.h：包含函数inputbox的声明

inputbox.h：包含函数inputbox的实现

link.h：包含类node的定义，变量insert\_flag的定义，list，figure\_to\_move，insert\_place，insert\_prev，tra\_i，tra\_j，figure\_num的外部变量声明，函数init\_list，insert，insert\_in，delete，traverse，traverse\_loop，free\_r，write\_insert，init\_graph，write\_delete，search\_serial的声明

link.c：包含变量insert\_flag与search\_serial的定义，list，figure\_to\_move，insert\_place，insert\_prev，tra\_i，tra\_j，figure\_num的定义，函数init\_list，insert，insert\_in，delete，traverse，traverse\_loop，free\_r，write\_insert，init\_graph，write\_delete，的实现

main.h：包含window\_width, window\_height，menu\_height，icon\_width，icon\_height的外部变量声明

main.c：包含window\_width, window\_height，menu\_height，icon\_width，icon\_height的变量定义与程序的Main函数

mousecallback.h：包含is\_exec，current\_exec，is\_create\_circle，is\_ draw\_circle，is\_create\_rectangle，is\_draw\_rectangle is\_create\_ triangle，is\_draw\_triangle，is\_create\_parallelogram，is\_draw\_parallelogram，is\_create\_diamond，is\_draw\_diamond，is\_move\_figure，is\_add\_figure，circle1x, circle1y, circle2x, circle2y，rectangle1x, rectangle1y, rectangle2x, rectangle2y，triangle1x, triangle1y, triangle2x, triangle2y，parallelogram1x, parallelogram1y, parallelogram2x, parallelogram2y，diamond1x, diamond1y, diamond2x, diamond2y，variable\_num，variable input\_data的外部变量声明（其中绝大部分是状态标志和图形的坐标），结构体variable的定义，函数read\_sentence，is\_in\_a\_figure，in\_box，MouseEventProcess，CharEventProcess，KeyboardEventProcess，TimerEventProcess的声明

mousecallback.c：包含绝大部分状态标志和图形的坐标变量定义，函数read\_sentence，is\_in\_a\_figure，in\_box，MouseEventProcess，CharEventProcess，KeyboardEventProcess，TimerEventProcess的实现

tip.exe：用于程序启动时的弹窗说明

* + 1. 多文件构成机制

所有头文件均使用预编译指令#pragma once进行保护，确保所有头文件不会被包含多次。各个.c文件均包含对应名称的头文件。由draw\_figure\_extemsion.c提供基础图形的绘制功能，由enum.h提供程序所需的类定义，由display.c提供程序画面的刷新功能，由draw\_button.c，draw\_menu.c，draw\_terminal.c，draw\_log.c提供程序的整体GUI框架，由mousecallback.c提供对用户的鼠标、键盘与字符输入的检测与响应功能，由link.c提供流程图绘制的程序内部数据操作，由file\_operate.c提供文件操作功能，最后通过main.c进行初始化从而组成完整的程序。

## 函数设计描述

函数原型：void init\_list(Link \*head);

功能：创建图元存储结构的头结点

参数：Link \*head:图元的存储链

函数原型：void insert(Link \*head, Link p);

功能：向图元存储的头部插入节点

参数：Link \*head:图元的存储链

Link p:要插入的新节点

函数原型：void insert\_in(Link \*head, Link figure\_to\_insert, Link prev ,Link p);

功能：将新图元figure\_to\_insert的存储节点插入至prev和p之间

参数：Link \*head:图元的存储链

Link figure\_to\_insert:要插入的节点

Link prev:插入后节点figure\_to\_insert的前驱节点

Link p:插入后节点figure\_to\_insert的后继节点

函数原型：void delete(Link \*head, Link p);

功能：删除图元p

参数：Link \*head:图元的存储链

Link p:要删除的节点

函数原型：void traverse(Link perv, Link tmp);

功能：遍历链，绘制图元，连接线以及连接线上的插入按钮，同时检测用户是否启动了绘制工作流并进行响应

参数：Link perv:tmp的前继节点

Link tmp:存储链中的任意一个节点

算法：判断tmp->type，绘制文本框与标签；判断tmp->kind，绘制图元形状。当prev不为空时，判断prev->type，若为condition，则判断tmp所在的prev后继，然后绘制连接线；若为loop，则调用traverse\_loop进行循环的绘制；若为其他语句，则直接绘制prev与tmp间的连接线。绘制连接线的同时绘制插入按钮，同时检测用户是否触发按钮事件，若是，则将is\_add\_figure设为TRUE，启动绘制工作流开始绘制新图元。最后递归调用本函数检查tmp的后继节点以遍历整个链。

函数原型：void traverse\_loop(Link p);

功能：绘制循环语句循环体内的图元

参数：Link p:存储链内语句类型为loop的某个节点

算法：绘制 loop\_start和loop\_end并与节点p进行连接，中间使用traverse函数进行遍历绘制

函数原型：void init\_graph();

功能：初始化存储链并绘制start与end节点

函数原型：bool inBox(double x0, double y0, double x1, double y1, double x2, double y2);

功能：判断当前鼠标是否在某个图形内

参数：double x0:鼠标所在x坐标

double y0:鼠标所在y坐标

double x1:图形左端坐标

double y1:图形底端坐标

double x2:图形右端坐标

double y2:图形顶端坐标

返回值：布尔量（0或1）

函数原型：bool is\_in\_a\_figure(Link tmp; double x; double y);

功能：判断鼠标是否在某个图元内

参数：Link tmp:图元节点

double x:鼠标所在x坐标

double y：鼠标所在y坐标

返回值：布尔量（0或1）

算法：若tmp为空则返回FALSE，否则使用inBox函数判断鼠标是否在图元图形内，若是，拖移工作流标记figure\_to\_move设为tmp同时返回TRUE。最后递归调用本函数检查tmp的后继节点以遍历整个树。

函数原型：void drawTriangle(double x, double y, double w, double h);

功能：绘制等腰三角形

参数：double x:三角形左下角点x坐标

double y:三角形左下角点y坐标

double w:三角形底边宽度

double h:三角形高

函数原型：void drawParallelogram(double x1, double y1, double x2, double y2, int fillflag);

功能：绘制平行四边形

参数：double x1:左下角点x坐标

double y1:左下角点y坐标

double x2:右上角点x坐标

double y2:右上角点y坐标

int fillflag:填充判断标志

函数原型：void drawDiamond(double x1, double y1, double x2, double y2, int fillflag);

功能：绘制菱形

参数：double x1:左顶点x坐标

double y1:上顶点y坐标

double x2:右顶点x坐标

double y2:下顶点y坐标

int fillflag:填充判断标志

函数原型：void draw\_menu();

功能：绘制菜单，响应菜单操作

部分重要局部变量：int selection:判断用户是否点击按钮以及点击了哪个按钮

算法：使用drawMenuBar函数绘制菜单区域，使用menuList函数绘制菜单并获取返回值赋值给selection。通过selection来判断用户是否点击按钮，若是，则进一步判断用户点击了哪个按钮，并做出对应响应。

函数原型：void add\_log(char \*s);

功能：向状态信息栏中写入操作日志

参数：char \*s:要写入的字符串

函数原型：void draw\_log();

功能：绘制状态信息栏

函数原型：void CharEventProcess(char ch);

功能：字符回调函数，用于获取用户的文本输入

参数：char ch:按键的ASCII码

函数原型：void KeyboardEventProcess(int key, int event);

功能：键盘回调函数，用于获取用户的键盘操作

参数：int key:代表哪个按键

int event:代表触发何种事件

函数原型：void MouseEventProcess(int x, int y, int button, int event);

功能：鼠标回调函数，用于获取用户的鼠标操作，并做出对应响应

参数：int x:触发事件时鼠标的x坐标

int y:触发事件时鼠标的y坐标

int button:触发事件的鼠标按键

int event:鼠标按键触发的事件种类

部分重要局部变量：double mx:以inch为单位的鼠标x坐标

double my:以inch为单位的鼠标y坐标

double omx:上一次触发事件时鼠标x坐标

double omy:上一次触发事件时鼠标y坐标

算法：先判断event值，若为按下，则判断button值，若为左键，则判断当前type\_choose的值，若为非空，则依次检查is\_create\_circle/ rectangle/triangle/parallelogram/diamond标志，将对应图形的绘制标志设为TRUE，进入画图模式；同时调用is\_in\_a\_figure函数判断鼠标光标是否在任意一个图元内，若是，则将拖移标志设为TRUE，启动拖移工作流，并使用omx与omy记录当前光标坐标。

若event值为松开，判断button值，若为左键，则检查标志is\_add\_figure，若为TRUE，则调用inBox函数判断光标是否在任意一个图形选择按钮中，若是，则将对应图形的is\_create标志设为TRUE，同时重置type\_choose变量，进入语句种类选择；检测拖移工作流标志is\_move\_figure，若为TRUE，则将其设为FALSE，结束工作流；依次检测is\_draw标志，判断是否在画图模式，若是，则将is\_add，is\_create，is\_draw三个标志都设为FALSE，结束画图，写入图元内的文本，然后将图形坐标和形状种类写入节点信息，最后将节点插入链中。若button值为右键，则调用is\_in\_a\_figure函数判断光标是否在任意一个图形中，若是，则调用write\_delete函数写入删除操作。

若event为移动，则检查is\_draw标志，判断是否在画图模式，若是，则将对应图形的2x与2y坐标设为光标坐标；检查is\_move\_figure，判断是否在拖移模式，若是，则更新目标图元的坐标信息，同时将omx与omy更新为光标坐标。最后使用display函数刷新画面完成绘制显示。

函数原型：void Main();

功能：初始化界面

函数原型：void Display();

功能：完成图形的绘制显示

算法：清除窗口的所有图形，然后重新绘制菜单栏、按钮和状态信息栏，之后检查标志is\_draw\_rectangle/triangle/parallelogram/diamond，判断是否在画图模式，若是，则根据对应的图形坐标全局变量绘制图形，达成画图模式的预览效果，然后绘制语句种类选择按钮，并判断用户是否进行了语句选择，若是，检查标志is\_add\_figure，判断用户是否确定插入位置启动了绘制工作流，若是，则将用户选择的语句种类赋给type\_choose并生成操作日志，最后使用traverse函数遍历树重新绘制其他图元。

函数原型：void draw\_terminal();

功能：绘制左下角文件操作终端

算法：首先利用绘制函数绘制出框架与按钮。使用button函数判断用户是否按下“ok”按钮，若是，则检测文本框内文本长度，若大于0（说明用户有输入），则将终端输入标志设为true，并检查标志to\_open\_file与to\_new\_file，并执行对应函数进行文件操作同时重置标志。最后清空终端文本框

函数原型：void add\_terminal();

功能：向终端信息栏中写入用户输入日志

函数原型：char \*get\_last\_terminal();

功能：寻找终端最后的一条文本

返回值：字符指针（字符串）

函数原型：void open\_a\_file(char \*str);

功能：打开一个之前所保存的流程图文件，若没有对应名称，则新建一个

参数：char \*str:文件名

算法：首先打开对应文件，若返回指针为NULL，则新建文件，并写入操作日志。使用init\_graph进行流程图初始化，然后开始读入文件，根据operate\_way特征值进行操作，若为2则直接进入下一个循环，若为1则依次读入图元信息，再检查图元的语句种类并对其文本信息进行赋值，若为循环语句则额外绘制一个循环体，赋予图元id，使用insert\_in函数将其插入存储链。

函数原型：void new\_a\_file(char \*str);

功能：新建一个流程图文件

函数原型：void close\_a\_file();

功能：关闭文件，并初始化图元存储链

函数原型：delete (Link \*head, Link p);

功能：删除图元

参数：Link \*head:存储链的头结点

Link p:将要删除的节点

算法：先检测图元p的语句类型，若为空，循环开始与结束，则不作操作（因为这些图元是不可以删除的），然后从头结点开始递归调用本函数遍历存储链，直到搜寻到节点p的前继节点，然后将p的next后继与前继节点对应的分支连接，即实现删除

函数原型：void write\_insert();

功能：向文件中写入新节点的数据，进行保存

算法：使用itoa函数将节点中的数字数据转换为字符串进行输出，将字符数据进行直接输出

函数原型：void write\_delete();

功能：向文件中写入删除节点操作

算法，直接向文件中写入operate\_way为2，在open时跳过该节点

# 部署运行和使用说明

## 编译安装

启动dev工程，直接编译即可。若编译失败，请在编译选项中加入-std:c99，重新尝试编译。

## 运行测试

案例：在测试绘制图形的过程中，发现平行四边形的连接线没有连接到其边框上，于是重新检查traverse函数中有关连接线绘制的代码，发现了常数设置错误，进行修改

## 使用操作

* + 1. 文件的创建：点击File菜单栏下的New按键或使用快捷键ctrl + N，然后在左下角文本框内输入文件名，点击OK，即可创建一个新的流程图文件
    2. 文件的保存：本程序在绘制流程图的同时进行实时保存，无需用户手动保存
    3. 文件的打开：点击File菜单栏下的Open按键或使用快捷键ctrl+O，然后在左下角文本框内输入想要打开的流程图文件名，点击OK，即可打开对应流程图
    4. 文件的关闭：点击File菜单栏下的Close按键或使用快捷键ctrl+C即可关闭当前文件
    5. 流程图绘制

#注意#本程序需要先进行文件的创建或打开，否则绘制流程将无法开始。

* + - 1. 添加新元素：点击两元素连线间的“+”号，随后在左上角选择元素图形与语句种类，最后鼠标拖动绘制，即可在连线中间插入一个新元素
      2. 各语句的内容编辑

1. 输入语句：在文本框内输入变量名=常数值
2. 输出语句：在中央文本框内输入变量名
3. 条件语句：在中央文本框内输入表达式
4. 循环语句：在WHILE下的文本框内输入表达式

#注意#用户可以不遵循上述规则进行编辑，但此时将无法正常进行模拟执行

#警告#请勿随意修改start,end,loop\_start和loop\_end，否则可能会出现无法预知的错误

* + 1. 模拟执行：在流程图已经完成绘制的情况下，点击start\_exec即可

#注意#在模拟执行的过程中，流程图将锁定，用户将无法对其进行任何操作

# 团队合作

## 任务分工

王佳宁：除模拟执行外的代码实现。

蒋昀昊：报告撰写与文档编辑。

秦子昂：模拟执行的代码实现。

## 开发计划

程序框图大程序开发包括三大板块：图形与交互板块、操作与配合板块、逻辑与存储板块。

图形与交互部分负责图形界面的所有问题，包括建立菜单系统、程序框图图形部分、图形与用户交互部分，还包括图形界面的控制台制作等。具体而言，菜单系统需要实现对于用户的每一个点击事件的响应，完成调用相应的回调函数，并处理相应的菜单显示情况。程序框图的图形部分是整个图形界面显示的核心，需要处理框图的插入、删除时的线路连接，模拟执行时框图高亮等过程的显示，以及框图内容的编辑和保存。其中最重要的是增删改时因图元信息的变更而造成的图形界面改变问题，这里创新性地计划采用支持多分支的扩展型链表数据结构。图形界面交互部分，需要处理好拖动、提示信息等问题。图形控制台需要显示用户与图形交互的结果，以及在模拟执行时的输出结果。

逻辑与存储部分负责框图的内部逻辑，包括了图元操作中最基本的函数，如新建、删除、查找等函数。还包括了图形界面信息的存储、模拟执行时变量信息的动态维护、框图信息的存储与读取，以及一些细节问题。这部分的开发思路就是，尽可能将函数功能单一化，增强其可移植性；函数单独调试，确保每一个细小功能执行顺利；功能相似的函数名称要具有相似性，并在书写时布置一起，以便调试时能快速查阅、修改。另外，数据存储需要规定格式，保证能够运行。

## 编码规范

1.文件命名：文件名全部使用英文小写字母，单词之间使用下划线分隔。

2.预处理：头文件的重复包含保护全部使用#pragma once。

3.函数名：由程序编写者定义的函数使用小写字母，单词之间使用下划线分隔。对libgraphics里函数的扩充使用和libgraphics相同的命名规范，即首字母大写且不使用下划线分隔。

4.语句中大括号的使用：while，if，for语句后如果是单个语句那么写在同一行，除非语句过长。如果是多个语句那么使用大括号，大括号的缩进使用visual studio自动缩进。

## 合作总结

开发亮点：1.将绘制流程图和文本编辑与模拟执行过程分开设计。

挑战点：

1. 普通的链表难以完美的存储流程图，故使用了类似树的混合链表结构。

2、一开始发现libgraphics难以实现弹窗功能，后来使用vs2017编写mfc实现了inputbox和弹窗功能。

应用知识点：c语言编程基础、链表操作、树操作、深度优先搜索、win32api、mfc、windows shell命令。

## 收获感言

王佳宁：我们组本次大程设计基于给出的libgraphics图形库，一开始的时候我们组三位成员对文件中的源码一头雾水，后来经过三个人细致认真的阅读，终于对图形界面的编写有了一定的概念和想法。  
在经过我们组内的讨论之后，决定选择算法流程图绘制工具作为我们的大程题目，在合作初期，三个人在讨论组内对整个程序的框架和具体实现都进行过细致的讨论，大家基本上都对程序的架构有了基本的认识。但是，在后来对大程具体分工的讨论上由于组内成员交流不够密切，导致我们三个人实现的功能有一部分的重复，这在一定程序上拖慢了我们组的完成进度，但是我们及时发现了这个问题，并且对三个人所实现的代码进行了重组，最后完成了一基本合格的算法流程图绘制工具。  
在整个大程的收尾阶段，通过三个人的讨论分析，我们认识到某些功能对于算法流程图来说是不合理或不必要的，我们在经过缜密的思考之后决定放弃实现这些功能进而把时间花在实现更有用的模拟执行功能上。通过这一次分组合作，让我认识到合作开发的必要性，也让我终于明白了合理分工对整个项目完成的影响是非常巨大的。

蒋昀昊：本次大程序设计中，对于libgraphic图形库的学习让我对图形界面的运行机制认知有了极大的加深。同时，本人负责程序的文档编辑以及主要的报告撰写，对该工作也有了不少的心得。（偷偷膜一下大佬，直接把代码任务全揽了下来，我自己写的一些代码全没用上）。不过这也反映出我们任务分工的不完备，导致任务量偏斜极大，是一个教训。由于大程设计工程繁杂，我们在讨论组中进行了深入的探讨，并一点一点地解决了程序设计中的难题，队友的想法也对我的设计思路产生了极大的启发，这让我意识到沟通与交流的重要性，同时也明白了对大型任务进行先决分析对于完成任务的巨大作用。总体来说，这次大程设计不仅提高了我的编程能力，也提高了我的合作沟通积极性、交流技巧以及文档撰写的水平。

秦子昂：本次大程序设计基于libgraphics图形库。开始时，我们对此都不熟悉，而其中涉及到的“回调函数”等名词，我们都是一头雾水，所以我们各自先熟悉了这个图形库的基本操作和简要的运行机制。这个过程中，图形不断刷新的操作让我大开眼界，对图形界面有了新的理解。在接下来的分工合作当中，由于我们对于整体情况的掌握并不能很好地把握，所以我们各自写的代码存在一定的功能重合。这一定程度上浪费了时间，降低了效率。这让我明白，完成任务时要对任务不断明确。另外，我认为对任务提前的分析与思考，是完成任务的先决条件。我们在群里深入讨论了关于程序框图“复制”的问题。我们发现，程序框图需要保持合法性，但复制粘贴未必能保证其合法性。如果我们只考虑画图，复制粘贴是容易的；但若考虑程序的执行，就不那么容易了。所以我们决定取消这个并不合理的功能。可见，沟通可以让我们的项目目标更为准确。总的来说，本次项目不仅提高了我的编程能力，还提高了我的合作意识。

# 参考文献资料

* 1. Raptor(the Rapid Algorithmic Prototyping Tool for Ordered Reasoning)-下载网址：<https://raptor.martincarlisle.com/>