浙江大学

程 序 设 计 专 题

大 程 序 报 告

1. 姓名 ： 朱理真 学号： 3190101094 电话：19817862976

2. 姓名 ： 包德政 学号： 3190105240 电话：19817866048

3. 姓名 ： 颜天明 学号： 3190105027 电话：19817863672

指导老师： 徐静春

2019~2020 春夏学期 2020 年 6 月 6 日

报告撰写注意事项

1） 图文并茂。文字通顺，语言流畅，无错别字。

2） 书写格式规范，排版良好，内容完整。

3） 存在拼凑、剽窃等现象一律认定为抄袭；0 分

4） 蓝色文字为说明，在最后提交的终稿版本，请删除这些文字。

目 录

小型算法流程图绘制工具大程序设计

1 大程序简介

1.1 选题背景及意义

流程图是一种用于辅助文字对算法、程序结构进行说明的有效语言。它通过使用矩形、菱形、连接线和文字等基本元素，对于所要描述的对象进行清晰的结构展示与顺序说明。是日常教学与学习过程、论文写作以及工业工程中必不可缺的一员。

在目前大家常用的流程图绘制工具中，主要有以 Microsoft Visio 和 OmniGraffle 为代表的电脑端软件以及 ProcessOn 为代表的在线网页等，他们功能丰富、元素详尽，为各行各业的人们带来了许多便利。

因此相较于本次候选题目中的其他三个题目（疫情数据分析与可视化工具、图书管理系统与本科生信息管理系统），小型算法流程图绘制工具拥有更接近我们的学习生活、实用性强与可拓展性高的突出优点。

而在本学期《程序设计专题》的课程中，本小组成员学习了有关链表、回调函数以及 libGraphics 与 simpleGUI 两个图形库等应用性强的相关知识。

因此基于以上背景，我们尝试在吸取前人经验的前提下，基于 libGraphics 与 simpleGUI 两个给定的图形库，用 C 语言（具有 C99 特性）编写一个小型的算法流程图绘制工具。

通过编写这一绘制工具，我们期望可以为老师同学们提供一个轻便、可用且基本功能齐全的工具，应用到实际的教学与学习生活中。另外，也希望可以积累一些多文件结构下编写 C 语言大程序的经验，与大家共同进步。最后，希望可以在大家的建议与意见中，增长关于需求分析、技术应用与优化用户交互体验等基本学科素养的知识与能力。

1.2 目标要求

针对这一项目，经小组协商，我们提出了以下目标与要求：

- **目标**：
  - 通过合理分工、通力合作，编写出一个符合要求、功能完善、切实可用的小型算法流程图绘制工具；
  - 充分考虑用户交互需求，以服务用户体验为前提思考功能设计；
  - 程序运行各个按钮清晰明了，帮助详尽没有遗漏，运行尽可能流畅，耐压能力较强；
  - 对于各项功能，应当具有良好的可扩展性，为进一步更新做准备；
  - 程序外观设计美观时尚，避免陈旧的时间感，符合基本的审美标准。
- **要求**：
  - 程序编写采用 C 语言，仅基于 `libGraphics` 与 `simpleGUI` 两个给定的图形库，不得使用其他程序语言与其他开源库；
  - 文件架构、接口、全局变量事先约定，如有冲突，应尽早进行协商沟通；
  - 代码质量应符合基本标准，具有可读性、可维护性；
  - 代码风格与注释风格组内统一；
  - 注释要求言简意赅，既不简陋也不冗余。

1.3 术语说明

下面是对于本程序所有相关术语的说明：

- **`.dat` 文件**：

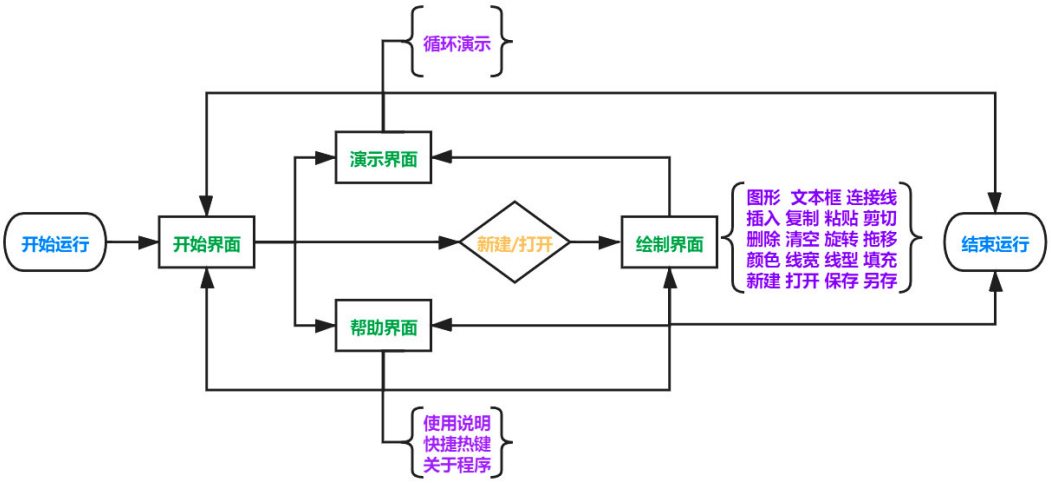
  这是本程序规定的存储与可以读取的二进制文件后缀，保存/另存为/欲读取的文件都需采用此后缀。更改将导致不能正常读取。
- **快捷热键**：

  即快捷键，均采用 `Ctrl + 某个字母键` 的形式。具体对应关系与可使用的界面，详见程序内帮助页面->快捷热键。
- **元素**：

指的是构成整个流程图的基本元素，即图形（包括矩形、菱形与圆角矩形，以下简称为图形）、文本框与连接线。

2 功能需求分析

- **业务逻辑：**
  - **架构**：
    - **表示层**：开始界面、绘制界面、演示界面与帮助界面等 4 个界面，实现与用户的表层交互；
    - **业务逻辑层**：鼠标、键盘回调函数，用于接收用户的请求，并进而调用相应的函数响应请求；
    - **数据访问层**：采用链表进行软件使用过程中的数据储存与读取，并最终以二进制文件的形式保存在用户的磁盘中，等待下一次读取；



  - **业务逻辑图**：

    图片说明：程序开始运行后进入开始界面，由开始界面、绘制界面可以进入其他三个界面；在演示界面、帮助界面可以返回上一级界面（即开始界面或绘制界面）；在除帮助界面以外任意一个界面，用户都可以选择结束运行。

- **功能需求：**

基于候选题目中提出的基本要求、上述叙述的目标与用户交互体验的考虑，

梳理本程序的功能需求如下：

- **页面方面**：具备开始页面、绘制页面、帮助页面与演示页面等四个基本页面，分别对应实现转接各个页面、绘制流程图（核心）、给予用户使用程序等方面帮助以及流程图教学样例演示四个功能；
- **开始界面**：需要实现新建文件 / 打开已有文件，切换至演示与帮助页面，退出程序等功能；
- **绘制界面（核心）**：需要实现如下功能：
  - 图形、文本框与连接线（含箭头）的添加、复制、粘贴、剪切与删除；
  - 图形、文本框与连接线的选定以及后续的编辑；
  - 图形的缩放、旋转、移动，文本框的缩放、移动，连接线整体的移动、转折点的移动；
  - 图形样式（包括边框颜色、边框粗细、边框虚实、是否填充与是否闪烁）、文本框样式（包括边框颜色、边框粗细、边框虚实、文字大小与文字颜色）与连接线样式（包括颜色、粗细）的变化；
  - 绘制页面的清空、保存、另存为，新建文件、打开现有文件，退出程序；
  - 底部有关于当前模式（分为自由模式、文本模式、图形模式与连线模式四种）、文件是否有所更改以及当前选定元素的基本信息的状态信息栏；
- **演示界面**：需要实现流程图显示、模拟执行、显示代码等功能；
- **帮助界面**：需要实现包括使用帮助、快捷热键与关于程序（互评版程序中这一内容将被屏蔽）的三个界面展示，以及返回上一级界面的功能；
- **音乐功能**：在开始、绘制与演示的三个界面中实现音乐播放（预置三首音乐：浙江大学校歌，The Congress Reel 以及 Yellow）与停止播放的功能。
- **菜单系统**：
  - 在开始界面中，上述功能需集成为文件、演示、音乐、退出与帮助等 5 个按钮组成的菜单系统中；

- 在绘制界面中，上述功能需集成为文件、编辑、样式、文本、图形、连线、音乐与帮助等 8 个按钮组成的菜单系统中；
- 在演示界面中，上述功能需集成为返回菜单、代码、音乐与步骤等 4 个按钮组成的菜单系统中；

- **图标工具栏**：在绘制界面中，需要对于常用功能（插入矩形、菱形、圆角矩形，清空页面与保存文件等 5 个功能）在页面右侧另设图标工具栏按钮以方便使用。
- **快捷键**：在开始界面、绘制界面与演示界面，结合菜单系统，实现若干个快捷键功能。

数据结构需求分析：

- **元素的数据结构**：

  由于每一个元素（图形、文本框与连接线）都具有多个属性信息，如图形具有：种类、位置、大小、颜色等等属性信息，因此应采用一个可以囊括多个信息的数据结构。

- **整体的数据结构**：

  元素构成的整体相当于是一个个小包裹构成的大包裹，为了能实现编辑功能，应当具有快速查找、单独删除不影响整体的功能、可复制性等性质。

- **文件的数据结构**：

  应当能够将整体的数据结构完整的储存，并可以再次被读取，如果能被快速读取将更佳。

性能要求：

- **适用平台**：

  目前仅考虑 Windows 平台即可。

- **编译器要求**：

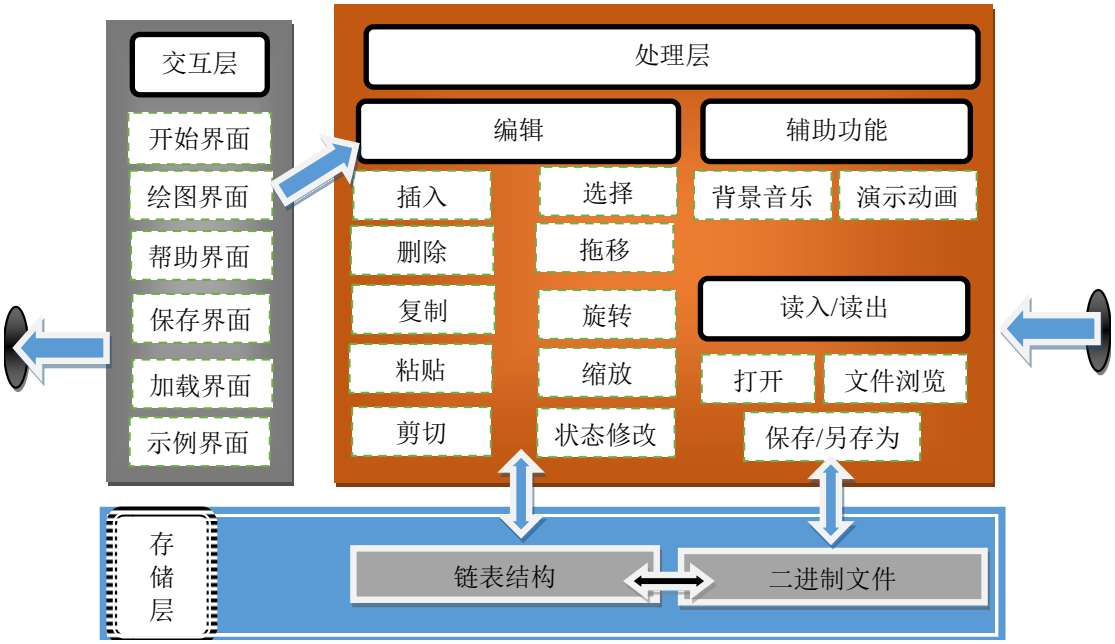
  应在 Visual Code 与 Dev C++ 下按给定步骤进行编译均可正常生成文件。

- **运行要求**：

  应达到基本流畅、正常情况下不闪退、正常保存与读取文件等基本要求。

3 程序开发设计

3.1 总体架构设计



3.2 功能模块设计

- **交互模块设计**

我们定义了枚举类型 enum __page { StartPage_, SavePage_, DrawPage_, HelpPage_, ExamplePage_, LoadPage_ }，并设定全局变量 Page，以表示不同的页面状态，分别是开始页面，保存页面，绘画页面，帮助页面，示例页面和读取页面。

不同页面的显示函数和不同页面的各类（鼠标、键盘）回调函数都有不同的名称和定义，但都返回 int 类型的值，表示下一次显示的页面。这样分别设立显示函数，保证了页面的独立性，也有利于小组的分工合作。

具体来说，对于显示函数，将各页面的显示函数都通过 switch 语句统括在总显示函数 Display()中，swtich 的关键词为全局变量 Page，case 则对应不同的页面的枚举值。

在不同页面 case 下，总是会执行一个赋值语句，这个赋值语句先调用相

应的显示函数，然后将这个函数的返回值赋给全局变量 Page。如果用户没有退出该页面，那么函数将返回自己的页面枚举值，Page 因此不会改变；如果用户通过交互退出页面，函数将返回不同值给 Page 以切换页面。

在每个页面下，为与程序进行交互，我们使用了 SimpleGUI 中的按钮和菜单控件。对于绘画界面，还要编写对应的鼠标回调函数、字符回调函数、键盘回调函数以控制和修改图形对象和文本框内容。

```
switch (Page) {
case DrawPage_:
    Page = Display_Draw(prehead, p, pretext);
    break;
case StartPage_:
    Page = Display_Start(winwidth, winheight);
    break;
```

图为显示函数 Display()定义的一部分截图

- **处理模块设计**

a)图形编辑

图形编辑采用模式控制，主要分为图形模式、文本模式和连线模式，不同模式下只能进行针对不同模式的操作，模式又可通过菜单对应选项切换。

图形模式下，可以插入矩形、圆角矩形和菱形三种图形，具体方式为以默认状态出现在屏幕默认位置。

在多个图形的状态下，需要选择当前待操作的图形对象。每一个对象都按照自己的形状大小形成一个选择范围，用户如果左键单击某一个对象，该对象就会被选中，并显示出四边的拉伸、旋转图标：其中红色的旋转图标默认在图形的正上方，其他三个拉伸图标默认在图形的正左、正右和正下方。图形的大小形状的修改，正是通过这些图标完成的。至于图形选择的内部实现，我们根据鼠标点击坐标搜索链表，找到第一个符合条件的图形然后返回该图形指针。

图形（特指矩形、圆角矩形和菱形）其他的编辑功能包括拖动、旋转、缩放，删除、复制、粘贴、颜色切换、边框粗细切换、边框线型切换、闪烁状态切换等。拖动通过对准矩形内部左键按下移动实现；旋转通过拖动旋转图标实现；缩放通

过拖动拉伸图标实现；其他功能则需要在菜单选项中操作实现，但并不是定量修改，而是预置几个颜色值、粗细值和实虚两种线型等供选择切换。

连线模式下，可以进行对连接线的操作。要产生一条连接线，首先要选中一个图形对象，然后从它的拉伸图标出发，引出一条连接线。其中根据实际需要，连接线只有水平和垂直两个方向，在拉伸过程中根据鼠标的移动确定。

一条连接线在生成完成后，可以进行后续的操作，包括线条的拉伸和移动，以及颜色、粗细等状态的改变。移动的具体操作是拖动连接线中任一一个线段的中部，线段就会相应地移动，而与该线段邻接而且必然与之垂直的另外两个线段，就会沿着线段移动的方向拉伸。

拉伸的具体操作则是要拖动连接线的末端，延其线段方向任意缩短或伸长。其他类型的操作，需要在菜单中选择。同图形模式，颜色等状态的修改为不连续修改。

文本模式，相应地，对文本框进行操作。在文本模式下，可进行插入文本框、编辑文本、选择文本框和文本颜色、字体大小等修改。具体地说，文本框由用户在文本模式下鼠标拖动生成，并闪烁光标提示输入。

输入完毕后，按下回车键结束输入。在一个文本框被选中时，显示线条边框；未被选中时，边框隐藏；初次编辑完成过后，选择菜单的“编辑文本”，并单击任意一个文本框，可再次进行编辑。选择“选择文本”，可以自由移动文本，且拖动下边框和右边框进行拉伸。文本颜色、大小等属性的修改直接通过菜单来选择切换。

b)读入/读出

对于另存为和打开功能，都设计各自的交互界面。

对于文件的保存，首先通过当前打开文件的文件名判断磁盘中是否已存在同名同地址的文件，若是，则保存不会跳转至另存为页面，而是直接调用保存函数处理；反之，若文件名为默认的“new.dat”，或者找不到该文件，就会自动进入另存为界面，该界面可以输入要保存的绝对地址或相对地址。

如果直接选择对文件另存为，则会直接跳入另存为界面。

为用户考虑，打开（读取）界面不仅可以手动输入文件地址，也可以浏览磁盘文件夹进行选择。

c)菜单控制

对于个别页面，特别是绘画页面，专门设置菜单以供实现多种功能。菜单模块首先利用 SimpleGUI 库中的 DrawMenu 函数绘画出菜单栏，然后根据其返回值传递相应值给内部全局变量 FlagSelection，而外部文件需要调用函数 GetSelection 来获取该值，同时保证了变量间的独立性。外部文件则根据不同的值构造选择结果产生不同的结果。

d)背景音乐

应课题要求，本程序还具备背景音乐播放功能。预置了三种背景音乐，在开始页面，绘制页面均有菜单控制音乐的播放与停止。准备在程序开始时就播放背景音乐，以营造良好的氛围，放松用户的心情。

e)流程图示例演示

该功能是本程序额外功能。主要包括流程图显示，代码显示和动画演示，即根据代码（几个简单的 C 语言）程序绘制流程图，且可供模拟执行其过程。

3.3 数据结构设计

根据不同的图元类型，我们定义了不同的结构体，以储存相应的参数信息。以下定义包含在 draw.h 中。

- **结构体介绍**

基本图形（矩形、菱形、圆角矩形）：

```
typedef struct graphics {
    int n;                  //类型信息，不同的数字代表不同类型
    double width, height;     //宽，高
    double angle;            //距正东方向的角度
    double cx, cy;           //图形几何中心坐标
    int color;               //预置颜色数组的下标，代表填充和边框的颜色
    int pensize;             //边框粗细
    int frame;               //表示边框的虚实
    int filled;              //是否填充
    struct graphics *next;    //链表后继,如无后继则为NULL
```

```
    cline *cl;                //附有的连接线指针，如无连接线附着则为NULL
    struct graphics *connect; //通过连接线关联着的另一个图形指针，无则为NULL
}node;
```

有向元线段结构：

```
typedef struct Line {
    double bx, by;            //始端点坐标
    double ex, ey;            //末端点坐标
    int dir;                  //表示水平还是竖直，水平为xaxs,竖直为yaxs
    struct Line *next;        //链表后继,如无后继则为NULL
    struct Line *before;      //链表前驱,如无后继则为NULL

}line;
```

连接线总括结构：

```
typedef struct ConnectLine {
    line *begin;              //始端元线段指针
    line *end;                //末端元线段指针
    int num;                  //线段数量
    int color, size, solid, erase; //分别表示颜色数组下标（表示颜色）、线段粗细、虚实、是否擦除
    int blineangle, elineangle;    //始端元线段指针的方向、末端元线段指针的方向
}cline;
```

文本框结构：

```
typedef struct Textbox {
    double cx, cy;          //左下角坐标
    double width, height;   //宽高
    char s[100];            //储存的字符
    int show;               //显示光标 showcuror
    int blink;              //是否闪烁
    int position;           //光标位置
    int linenum;            //每行字符数
```

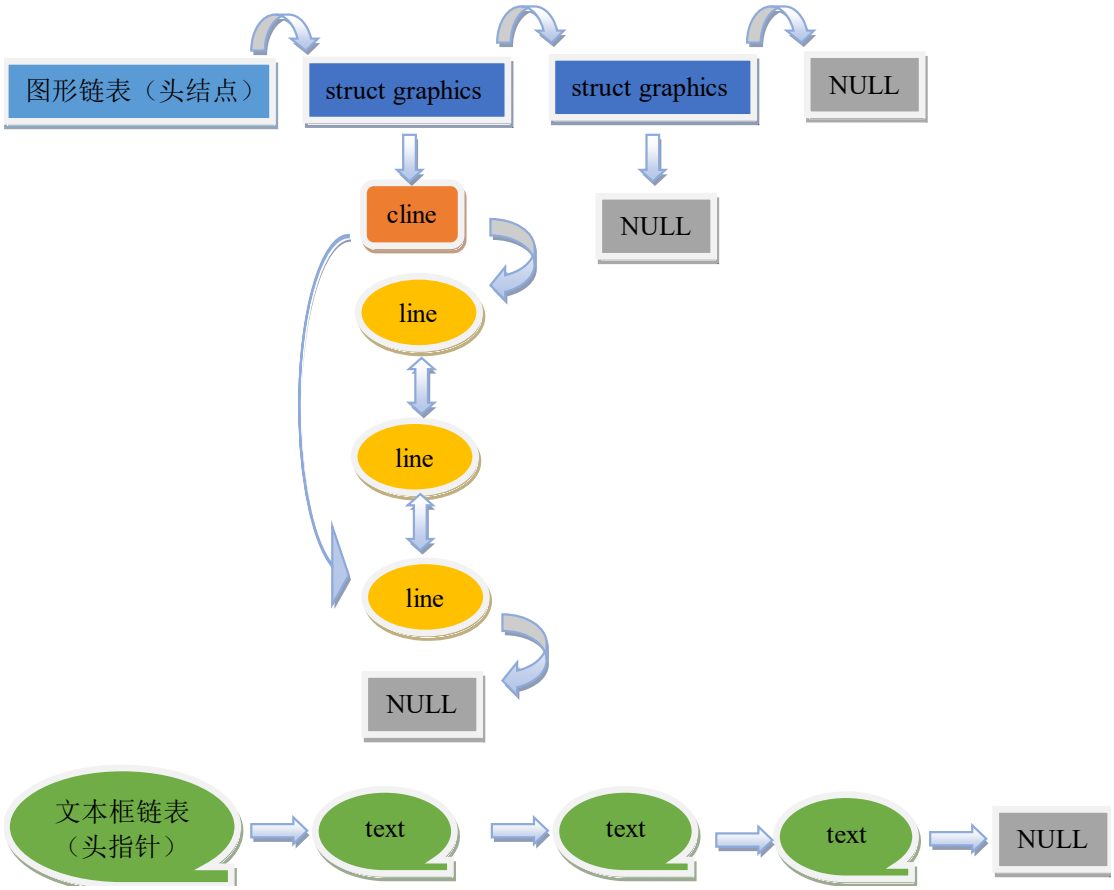
```
    int precolor;           //上一次的颜色数组下标
    int color;              //颜色数组下标
    int fontstyle;          //字体风格
    struct Textbox* next;   //链表后继,如无后继则为NULL
}text;
```

- **储存方式介绍**

内存表示：

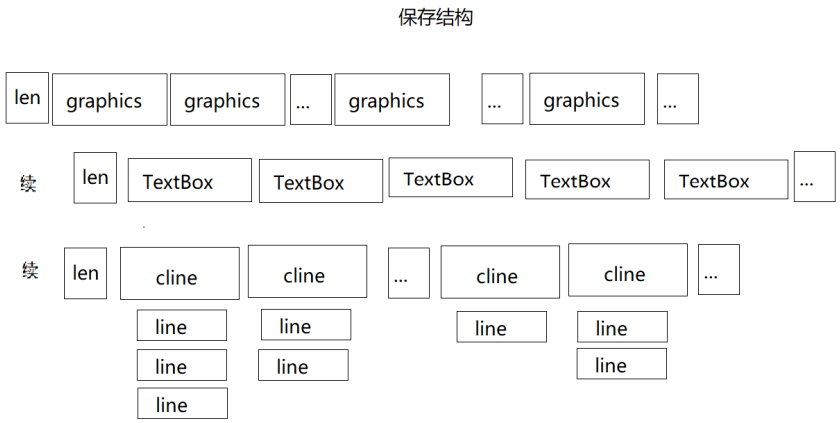
如下图，内存存储主要有两支链表，一支为图形链表，给定头指针，存储 struct graphics 结构体（代表矩形、菱形或圆角矩形），其中每个结构都指向一个连接线结构体指针 cline，若指针为 NULL 则不附带连接线；若不为空，则又指向一个有向元线段结构体 line 的双向链表。

文本框链表则从文本框头指针开始，以顺序单向的方式进行独立的存储。



图为内存示意图

文件存储：



本程序使用二进制文件进行存储，考虑到文件的线性储存结构，设置了如图的储存方式，先存入图形链表的长度（不计入头结点和 NULL 结点），然后依次延链表顺序将结构体数据写入文件。

第一支主链表写入完毕后，类似地，先写入文本框链表的长度，然后依次写入文本框结构体。然后写入完毕，保存函数返回。接着是连接线的存储。

读取时也是类似的，根据开始读入的链表的长度，以及结构体有无连接线附着，进行相反的操作。

3.4源代码文件组织设计

<文件目录结构>

1）文件函数结构

- **程序包含文件与功能简介**

文件名	功能简介
file.c	处理数据的保存、加载、链表的破坏。
file.h	用于声明 file.c 的函数接口，和已知结构体的类型定义
edit.c	图形、连接线、文本框的选择、复制、粘贴、删除、拖移、旋转、缩放等功能，以及文本框的输入编辑。
edit.h	用于声明 edit.c 的函数接口。
menu.c	绘制主界面的整体框架，菜单，状态信息栏，工具栏等

	对象，有快捷键，并关联到对应的处理函数。
menu.h	用于声明 menu.c,start.c,help.c 的函数接口。
draw.c	实现绘画，包括矩形、圆角矩形、菱形、箭头线条等多种形状的绘制， 包括颜色、线宽、线型、填充等属性。
draw.h	用于声明声明 menu.c,start.c,help.c 的函数接口，并给出基本图形结构体 node,连接线总括结构体 cline,有向元线段结构体 line,文本框结构体 text 的定义，并给出一些
help.c	撰写帮助文档，使用方法等,有回退按钮至主界面
save.c	读取文件和保存界面， 支持输入文件名在本地目录存取， 也支持输入绝对路径+文件名存取。 可以列出本地目录所有文本文件（加了我们规定的后缀名的）进行存取
main.c	主文件，含 Main 函数。 初始化链表头指针； 回调函数和显示函数定义； 控制并处理各个不同的页面，根据不同的页面采用不同的显示函数和回调函数。
start.c	绘制开始界面，有打开，新建，退出等按钮，要求界面美观简洁。
example.c	示例代码的动态执行过程，包括代码展示、流程图绘制和动画演示。
ExampleMenu.c	为示例界面编写菜单函数，实现功能交互。
music.c	支持背景音乐功能。

- **各文件内容一览**

文件名	**函数定义 / 文件内容**
main.c	void Main()

start.c	void Display()
	void MouseEventProcess(int x, int y, int button, int event)
	void TimerEventProcess(int timerID)
	void CharEventProcess(char key)
	void KeyboardEventProcess(int key, int event)
	void Default_Color_Define()
	int Display_Draw(node* prehead, node* p, text* t,text* tt)
	int DrawMode_Mouse_Process(int x, int y, int button, int event)
	int DrawMode_Time_Process(int timerID)
	void Start(double winwidth, double winheight)
	int DrawStartMenu(double winwidth, double winheight)
	void DrawTitle(double winwidth, double winheight)
	void DrawAuthor()
	int Display_Start(double winwidth, double winheight)
draw.c	void ChangeLine()
	void forward(double distance)
	void move(double distance)
	void imLine(double dis)
	void Drawline(int flag,double dis)
	void Draw(node *g, int isDraw)
	void DrawFrame(node *g, int color, int erase)
	void DrawArrow(double bx, double by, double angle, double length)
	void Drawedge(node *g)

draw.h

```
double GetAngle(double ax, double ay, double bx, double by)
void Liner(cline *l)
int Judge(int dir, line *pre, double x, double y)
line * AddLine(int dir,line *pre, double x, double y)
void Arrow(int dir, line *pre, double prex, double prey, double x, double y, int erase)
void Fill(node *g)
void DrawButton(double x, double y, int n, int isErase)
void SelectButton(double x, double y, int n, int isDraw)
void DrawCADButton(int button)
void DrawTextEdge(text *t)
void DrawCursor(text *t)
```

常量定义：

```
Pi, dot
```

颜色宏定义：

```
white, Red ,Green ,Blue,black ,yellow , , pink ,orange , cyan,deepgreen,gray
```

模式宏定义：

```
TEXT, LINE, GRAPH, FREE
```

方向相关宏定义：

```
left, right, down, up, xaxs, yaxs, ROTATE;
```

文本相关宏定义：

```
Textzoom, Textmove
```

字体宏定义：

```
宋体, source code pro, consolas
```

	静态全局变量： static double Angle; static int clinecolor; static char scolor[20][10]; static char fontstyle[3][20]
	结构定义： typedef struct Line{}line; typedef struct ConnectLine{}cline; typedef struct graphics{}node; typedef struct Textbox{}text;
	对 draw.c 中的函数声明。
edit.c	void Zoom(node *p, int dir, double x, double y)
	void Move(node *p, double prex, double prey, double x, double y)
	void Rotate(node *p, double x, double y)
	node * Copy(node *p)
	void Create(node *prehead, node *t)
	node * CreateNew(node * prehead, int n)
	node * Delete(node *prehead, node *p)
	void updateline(node *p)
	void DrawList(node *prehead)
	void Converse(cline *t, cline *p)
	void TextBox(double x, double y, double width, double height, char string[])
	line *Trveal(cline *cl, double x, double y)
	node * TrvealLine(node *prehead, double x, double y)
	void MoveLine(line *l, double x, double y)
	void ZoomLine(line *l, double x, double y)

	int JudgeEdge(node *p, double x, double y)
	void JudgeConnect(node *prehead, node *p)
	node *Search(node *lhead, double x, double y)
	node * SearchLine(node *prehead, double x, double y)
	double Commute(double x, double y, double x0, double y0, double k)
	int InRect(double cx, double cy, double w, double h, double x, double y)
	int InRegion(double cx, double cy, double w, double h, double x, double y, double angle)
	int InArc(double cx, double cy, double x, double y, double r, double angle, double d)
	text * AddText(text *pretext, double cx, double cy)
	void DisplayText(text* pretext, text* tt)
	text* DeleteText(text* pretext, text* tt)
	text* SearchText(text* pretext, double x, double y)
	int JudgeText(text* t, double x, double y)
	void MoveText(text* t, double prex, double prey, double x, double y)
	int ZoomText(text* t, double x, double y, int dir)
	void DrawString(text* t)
	void Cursor(text* t)
	void AddChar(text* t, char key)
	void DeleteChar(text* t)
edit.h	对 edit.c 的函数声明。
music.c	void Music(int Choice)
menu.c	void DrawMode(double winwidth,double winheight)
	int GetSelection()

	void DrawMenu(double winwidth, double winheight)
	void DrawStatusBar(LinkStatusInfo PresentLinkStatusInfo, double winwidth, double winheight)
	void DrawIconButton(double winwidth, double winheight)
menu.h	结构定义： typedef struct tag_StatusInfo {}StatusInfo, * LinkStatusInfo;
	静态变量： static int ReturnFlag, FlagSelection; static int CreateMode;
	void Example_Time_Process(int timerID);
	int Display_Example();
	double TurnAngle(double bx, double by, double ex, double ey);
	double distance(double bx, double by, double ex, double ey);
	void ProcessInitial_1();
	void Example_Time_Process_1(int timerID);
example.c	int ExampleDisplay_1();
	void startTimer(int id, int timeinterval);
	void cancelTimer(int timerID);
	static void ShowCode(double x, double y, char* code);
	static void Object_ArrayToChain(Link head, int n, double x, double y);
	static void TextBox_ArrayToChain(text* head, int n, double x, double y);

	static void _showmytext(text* Mytext);
	static void _drawlines_(MyLine* lines, int cur1);
	static void Line_Adjust(MyLine* head, int cur, double dx, double dy);
example.h	对 example.c 中函数的声明。
ExampleMenu.c	void DrawExampleMenu(double winwidth, double winheight)
	int GetExampleSelection()
ExampleMenu.h	对 ExampleMenu.c 中的函数的声明。
file.c	status Save(char* filename, Link head, text* thead, int len, int SaveCopyAs)
	status Load(char* filename, Link ptrh, text* ptrth)
	int GetListLen(Link head)
	void Close(Link head, text* texthead)
	void _Destroy(Link head, text* texthead)
	static void TextBox_Destroy(text* head)
	static void CLine_Destroy(cline* cl)
	static void MainChain_Destroy(Link head)
	static void SaveTBox(FILE* fp, text* thead)
	static void SaveMainChain(FILE* fp, Link head)
	static void SaveLine(FILE* fp, struct ConnectLine* cline)
	static text* LoadTBox(FILE* fp)
	static struct ConnectLine* LoadLine(fp)
	static Link LoadMainChain(FILE* fp, int ListLen)
file.h	结构指针命名： typedef node* Link

	对 file.c 中的函数的声明。
save.c	int Display_Save(double wholeW, double wholeH,char* filename,Link head, text* thead)
	int Display_Load(double wholeW, double wholeH, Link ptrh, text* ptrth, char* Name)
	static int SetFileNames_(int page,char* relafname,char* baseAddress)
	void AddressCatch(char* base, char* source)
save.h	对 save.c 中的函数的声明。
help.c	int drawButtons(double winwidth, double winheight)
	void ProgramHelp(double winwidth, double winheight)
	void ShortcutHelp(double winwidth, double winheight)
	void AboutHelp(double winwidth, double winheight)
	int Display_Help(double winwidth, double winheight)
	void SetHelpMode(int mode)

2）多文件构成机制

为方便，记下列文件包含为**标准包含**：

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stddef.h>
#include <assert.h>
#include "graphics.h"
#include "genlib.h"
#include "conio.h"
```

```
#include <windows.h>
#include <olectl.h>
#include <stdio.h>
#include <mmsystem.h>
#include <wingdi.h>
#include <ole2.h>
#include <ocidl.h>
#include <winuser.h>
#include "imgui.h"
```

文件	包含文件
file.c	标准包含 #include "edit.h" #include "draw.h" 无外部变量或函数（非库函数）
draw.c	标准包含 #include "draw.h" 无外部变量或函数（非库函数）

edit.h	标准包含
	#include "draw.h"
	#include "menu.h"
	(draw.c)
	void Draw(node* g, int isDraw);
	void Fill(node* g);
	void Liner(cline *l);
	void DrawTextEdge(text* t);

example.c	标准包含 #include "draw.h" #include "edit.h"

	#include "file.h"
	#include "ExampleMenu.h"
	(draw.c)
	void DrawArrow(double bx, double by, double angle, double length);
	(example.c)
	void DrawMenu_(double winwidth, double winheight)
	(edit.c)
	void DrawList(node* head);

ExampleMenu.c	标准包含 #include "menu.h" 无外部变量或函数（非库函数）
menu.c	标准包含 #include "menu.h" #include "draw.h" 无外部变量或函数（非库函数）
music.c	#include<stdio.h> #include<windows.h> #include<mmsystem.h> #pragma comment(lib, "WINMM.LIB") 无外部变量或函数（非库函数）
save.c	标准包含 #include <io.h> #include "file.h" #include "save.h" (file.c)

	status Save(char* filename, Link head, text* thead, int len, int SaveCopyAs); status Load(char* filename, Link ptrh, text* ptrth);
help.c	标准包含 #include "menu.h" 无外部变量或函数（非库函数）
start.c	标准包含 #include "draw.h" #include "edit.h" 无外部变量或函数（非库函数）

main.c	标准包含
	#include "draw.h"
	#include "edit.h"
	#include "save.h"
	(start.c)
	int Display_Start(double winwidth, double winheight);
	(help.c)
	int Display_Help(double winwidth, double winheight);
	(example.c)
	int Display_Example();
	void Example_Time_Process(int timerID);
	(save.c)
	Display_Save(winwidth, winheight, __filename, prehead,

```
pretext);
Display_Load(winwidth,
winheight,prehead,    pretext,
__filename);
(edit.c)
void AddChar(text* t, char key);
void DeleteChar(text* t);
int JudgeChinese(text* t);
void DrawList(node* head);
void DisplayText(text* pretext,
text* tt);
node* Copy(node* p);
void Create(node* head, node*
t);
node* Delete(node* head, node*
p);
text* DeleteText(text* pretext,
text* tt);
node* CreateNew(node* prehead,
int n);
text*  AddText(text*  pretext,
double cx, double cy);
text* SearchText(text* pretext,
double x, double y);
int JudgeText(text* t, double x,
double y);
node* SearchLine(node* prehead,
double x, double y);
node* TrvealLine(node* prehead,
```

```
double x, double y);
line* Trveal(cline* cl, double x, double y);
int JudgeEdge(node* p, double x, double y);
node* Search(node* lhead, double x, double y);
void MoveText(text* t, double prex, double prey, double x, double y);
int ZoomText(text* t, double x, double y, int dir);
void Zoom(node* p, int dir, double x, double y);
void Move(node* p, double prex, double prey, double x, double y);
void Rotate(node* p, double x, double y);
node* TrvealLine(node* prehead, double x, double y);
void MoveLine(line* l, double x, double y);
void ZoomLine(line* l, double x, double y);
node* SearchLine(node* prehead, double x, double y);
void JudgeConnect(node *prehead, node *p);
line* AddLine(int dir, line*
```

```
pre, double x, double y);
void DrawFrame(node *g, int color);
(menu.c)
int GetSelection();
void ChangeMode(int a);
void DrawMenu(double winwidth, double winheight);
void DrawStatusBar(LinkStatusInfo PresentLinkStatusInfo, double winwidth, double winheight);
void DrawIconButton(double winwidth, double winheight);
void DrawMode(double winwidth,double winheight);
(draw.c)
void Drawedge(node* g);
(file.c)
void _Destroy(Link head, text* texthead);
status Save(char* filename, Link head, text* thead, int len,int SaveCopyAs);
void Close( Link head, text* texthead);
void CLine_Destroy(cline* cl);
(help.c)
void SetHelpMode(int mode);
```

draw.h	宏保护： #ifndef _draw_h #define _draw_h 标准包含
edit.h	edit.h 宏保护： #ifndef _edit_h #define _edit_h 标准包含 #include "draw.h" #include "menu.h"
save.h	宏保护:#pragma once #include "file.h"
file.h	宏保护:#pragma once #include "draw.h"
example.h	宏保护:#pragma once
menu.h	宏保护： #ifndef _menu_h #define _menu_h 标准包含

3.5 函数设计描述

- **main.c**

序号	具体内容
1	函数原型：**void Main()** 功能描述：程序入口函数，主要进行了图形界面的初始化、预置颜色的定义、窗口大小的获取全局变量 prehead 和 pretext（即图形链表和文本框链表的头指针）的空间分配、回调函数的注册以及信息数据结构的初始化 参数描述：无

	返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：无
2	函数原型：**void Display()** 功能描述：总的显示函数，根据全局变量 Page 决定显示何种页面并将当前显示函数的返回值赋给 Page,决定下次是否更换页面。 参数描述：无 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过 switch 结构分支来选择执行页面函数，同时记录 Page 值的变动，从而更新 prePage
3	函数原型：**void MouseEventProcess(int x, int y, int button, int event)** 功能描述：总的鼠标回调函数，根据全局变量 Page 决定显示何种页面并将当前回调函数的返回值赋给 Page,决定下次是否更换页面 参数描述： x,y:　鼠标坐标 button：鼠标按键 event:　鼠标事件信息 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过 switch 结构分支来选择执行页面函数，同时记录 Page 值的变动，从而更新 prePage
4	函数原型：**void TimerEventProcess(int timerID)** 功能描述：总的计时器回调函数，根据全局变量 Page 决定显示何种页面并将当前回调函数的返回值赋给 Page,决定下次是否更换页面 参数描述： timerID：计时器事件标识值 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过 switch 结构分支来选择执行页面函数，同时记录 Page 值的变动，从而更新 prePage。
5	函数原型：**void Default_Color_Define()** 功能描述：定义预置的颜色

	参数描述：无 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：无
6	函数原型：**int Display_Draw(node* head, node* p, text* t,text* tt)** 功能描述：绘制页面的显示函数，绘制包括菜单栏、状态栏、图标工具栏、状态球以及图形、文本、连接线的显示 参数描述： head：图形链表头指针 p：　当前的图形对象指针 t：　文本框链表的头指针 tt：　当前的文本框对象指针 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：顺序执行各显示函数
7	函数原型：**int DrawMode_Mouse_Process(int x, int y, int button, int event)** 功能描述：绘制页面的鼠标回调函数，主要有两个功能：一是接收绘制页面的菜单选项，并做出相应的反应；而是相应绘制界面的鼠标操作，比如图形的拖动、旋转、选择、拉伸等 参数描述： x，y，鼠标位置，button，按键，event，事件 返回值描述：返回页面枚举值，页面不动返回 DrawPage_ 重要局部变量定义：cx，cy，prex，prey，premx，premy;textdir，newflag1，newflag2，switchtext,newtext，textmove，textzoom,dir，flag，linemove，aline，isline，isdraw，linezoom,zoom，zoomdir，ismove，isrotate 重要局部变量用途描述：鼠标位置有关变量：当前位置，前一个位置；文本有关变量：方向，选择模式，创建模式，移动模式，缩放模式；连接线变量：方向，标志，移动，对齐，开启，绘制，缩放；图形相关变量：缩放，缩放方向，移动，旋转 函数算法描述：处理菜单选项分支；顺序检验各鼠标事件。
8	函数原型：**int DrawMode_Time_Process(int timerID)** 功能描述：绘制页面的计时器回调函数，主要有两个功能：一是进行图形对象的闪烁，二是进行文本框图标的闪烁

	参数描述：timerID：时间注册器代号 返回值描述：返回页面枚举值，页面不动返回 DrawPage_。 重要局部变量定义：fcolor 重要局部变量用途描述：图形边框闪烁的颜色 函数算法描述：根据计时器标志号处理分支
9	函数原型：**void KeyboardEventProcess(int key, int event)** 功能描述：键盘回调 参数描述：key，按键字符；event，事件 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：详见代码
10	函数原型：**void CharEventProcess(char key)** 功能描述：字符回调 参数描述：key，按键字符；event，事件 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：详见代码

- **start.c**

序号	具体内容
1	函数原型：**void Start(double winwidth, double winheight)** 功能描述：这个函数用于初始化窗口标题与调用整体的界面绘图函数。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：利用 libGraphics 内函数命名窗口标题，并通过调用 Display_Start 函数进行整体界面的绘画。
2	函数原型：**int DrawStartMenu(double winwidth, double winheight)** 功能描述：这个函数用于绘制开始界面的五个菜单按钮，并实时监测，以 result 记录点击结果并返回。实现页面转换。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：枚举类型中 xxxPage 对应的 int 型的值，说明应转换至哪个界面以响应用户请求。 重要局部变量定义：

	static char* menuListFile[], menuListExample[], menuListHelp[], menuListExit[], menuListMusic[] int FileSelection, ExampleSelection, HelpSelection, ExitSelection, MusicSelection, Result=StartPage_; 重要局部变量用途描述：五个 static char*数组定义了按钮的内容；对应的 xxxSelection 记录点击返回值；Result 根据点击结果被赋值为枚举中的不同值代表应该转接到的不同页面。 函数算法描述：利用 simpleGUI 中的 menulist 函数进行绘制、监测与保存点击后的返回值，继而根据返回值做不同的操作，如，更改 result 的值、直接调用 Music 函数等
3	函数原型：**void DrawTitle(double winwidth, double winheight)** 功能描述：这个函数用于绘制开始界面中央的标题。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： double TitleX, TitleY; 重要局部变量用途描述：标题的位置坐标。 函数算法描述：借助窗口的宽度与高度，在中央位置绘制软件的名称。
4	函数原型：**void DrawAuthor()** 功能描述：这个函数用于绘制开始界面左下角的作者信息（互评阶段不显示身份信息）。 参数描述：无 返回值描述：无 重要局部变量定义： double AuthorX, AuthorY; 重要局部变量用途描述：作者的位置坐标。 函数算法描述：借助窗口的宽度与高度，在左下角位置绘制作者信息。
5	函数原型：**int Display_Start(double winwidth, double winheight)** 功能描述：这个函数调用上述各个绘制函数，按顺序绘制开始界面的各个元素。并返回菜单按钮点击后对应的转换目标页面信息。 参数描述：当前窗口的宽度与高度 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：调用 DrawTitle，DrawAuthor 与 DrawStartMenu 三个函数，依次绘制标题、作者信息与菜单按钮。并返回 DrawStartMenu 的返回值。

- **draw.c**

序号	具体内容
1	函数原型：**void ChangeLine()** 功能描述：改变连接线颜色 参数描述：无 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过函数改变局部变量
2	函数原型：**void forward(double distance)** 功能描述：指定方向画指定长度线段 参数描述：distance：距离 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过局部变量 Angle 画线
3	函数原型：**void move(double distance)** 功能描述：移动画笔 参数描述：distance：距离 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过局部变量 Angle 移动画笔一定距离。
4	函数原型：**void imLine(double dis)** 功能描述：画虚线。 参数描述：dis：距离 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：通过局部变量 Angle 画出虚线。
5	函数原型：**void Drawline(int flag,double dis)** 功能描述：自定义画线函数 参数描述：dis：距离，flag，实线还是虚线 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：flag 真画实线，flag 假画虚线

6	函数原型：**void Draw(node *g, int isDraw)** 功能描述：画图形 参数描述：g 需要画出的图形；isDraw，是否擦除 返回值描述：无 重要局部变量定义：r，a 重要局部变量用途描述：距离中心的距离，与中心连线的方向 函数算法描述：通过不同的 n，switch 画出不同的图形，通过 angle 旋转画出图形
7	函数原型：**void DrawArrow(double bx, double by, double angle, double length)** 功能描述：画箭头函数 参数描述：bx，by 起始位置，angle，方向，length，长度 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：在最后的节点分叉 39 度形成两个箭头
8	函数原型：**void DrawFrame(node *g, int color, int erase)** 功能描述：画图形边框 参数描述：g，需要画边框的图形，color，边框颜色，erase，是否擦除 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：在图形的外部画出一个矩形的边框
9	函数原型：**void Drawedge(node *g)** 功能描述：画图标 参数描述：g，需要画图标的图形 返回值描述：无 重要局部变量定义：w，h 重要局部变量用途描述：图形的宽和高 函数算法描述：在图形的四周旋转画出四个图标
10	函数原型：**double GetAngle(double ax, double ay, double bx, double by)** 功能描述：获取角度 参数描述：ax，ay，起始位置；bx，by，结束位置 返回值描述：返回角度（弧度） 重要局部变量定义：无 重要局部变量用途描述：无

	函数算法描述：通过反正切函数算出角度
11	函数原型：**void Liner(cline *l)** 功能描述：画出连接线 参数描述：l，需要画的连接线 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据不同的方向依次画出连接线上的各个线段
12	函数原型：**int Judge(int dir, line *pre, double x, double y)** 功能描述：判断连接线方向 参数描述：pre，需要判断的连接线 ，dir，方向，x，y，鼠标的位置 返回值描述：返回连接线的方向 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据起始与末尾 x 和 y 的相等关系判断方向
13	函数原型：**line * AddLine(int dir,line *pre, double x, double y)** 功能描述：添加连接线 参数描述：pre，需要画的连接线 ，dir，方向，x，y，鼠标的位置 返回值描述：返回添加的连接线 重要局部变量定义：l 重要局部变量用途描述：临时节点 函数算法描述：根据鼠标的停止的位置添加当前连接线
14	函数原型：**void Arrow(int dir, line *pre, double prex, double prey, double x, double y, int erase)** 功能描述：动态画出连接线 参数描述：pre，需要画的连接线 ，dir，方向，x，y，鼠标的当前位置，prex，prey，鼠标上一个位置 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据鼠标的移动即时更新屏幕中的连接线画面
15	函数原型：**void Fill(node *g)** 功能描述：填充图形 参数描述：g，需要填充的图形 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无

	函数算法描述：填充当前图形（需要封闭，故与虚线不兼容）
16	函数原型：**void DrawTextEdge(text *t)** 功能描述：画出文本框 参数描述：t，需要画的文本框 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：画出当前的文本框

- **edit.c**

序号	具体内容
1	函数原型：**void Zoom(node *p, int dir, double x, double y)** 功能描述：缩放图形 参数描述：p，需要缩放的图形，dir，缩放方向，x,y,鼠标位置 返回值描述：无 重要局部变量定义： angle，r 重要局部变量用途描述：鼠标与中心线连线的夹角 ，鼠标与中心距离 函数算法描述：根据鼠标在对应方向上的移动距离确定缩放比例
2	函数原型：**void Move(node *p, double prex, double prey, double x, double y)** 功能描述：移动图形。 参数描述：p，需要移动的图形，prex,prey,前一个鼠标位置，x,y,鼠标位置 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据鼠标与前一个位置的相对距离确定图形的移动方向与距离
3	函数原型：**void Rotate(node *p, double x, double y)** 功能描述：旋转图形。 参数描述：p，需要移动的图形，x,y,鼠标位置。 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无。 函数算法描述：获取当前位置方向进行旋转。
4	函数原型：**node * Copy(node *p)** **void Create(node *prehead, node *t)**

	node * CreateNew(node * prehead, int n) 功能描述：复制图形。 参数描述：p，需要复制的图形；prehead,图形链表头结点，t，临时节点；n，需要创建的图形的类型 返回值描述：返回创建的图形在链表中的位置 重要局部变量定义：pt 重要局部变量用途描述：临时贮存复制的图形 函数算法描述：创建一个当前图形的副本，并插入链表中。
5	函数原型：**line *Trveal(cline *cl, double x, double y)** 功能描述：搜寻点击当前连接线的哪一部分 参数描述：cl，当前连接线，x,y,鼠标位置 返回值描述：点击的那条线 重要局部变量定义：temp 重要局部变量用途描述：临时指针 函数算法描述：遍历链表，搜寻结构体
6	函数原型：**node *Search(node *prehead, double x, double y)** 功能描述：寻找图形 参数描述：prehead，图形链表头结点 x,y,鼠标位置 返回值描述：选中图形 重要局部变量定义：pt 重要局部变量用途描述：临时指针 函数算法描述：遍历链表，搜寻结构体。
7	函数原型：**node * Delete(node *prehead, node *p)** 功能描述：删除图形。 参数描述：prehead，图形链表头结点，p，当前图形 返回值描述：链表中下一个图形 重要局部变量定义：pt 重要局部变量用途描述：临时指针 函数算法描述：遍历链表，删除结构体。
8	函数原型：**void updateline(node *p)** 功能描述：更新连接线。 参数描述：p，当前图形 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：先判断是连接线起始还是结束，然后根据图形的状态更新连接线
9	函数原型：**void DrawList(node *prehead)**

	功能描述：画图形链表。 参数描述：prehead，链表头结点 返回值描述：无 重要局部变量定义：p 重要局部变量用途描述：临时指针 函数算法描述：遍历链表画图形
10	函数原型：**void Clear(node * prehead, text * pretext,double w,double h)** 功能描述：清屏。 参数描述：prehead，图形链表头结点，pretext，文本链表头结点，w，h，屏幕大小 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：清屏
11	函数原型：**node * TrvealLine(node *prehead, double x, double y)** 功能描述：判断是否选中连接线 参数描述：prehead，图形链表头结点，x，y 鼠标位置 返回值描述：连接线相连图形 重要局部变量定义：temp 重要局部变量用途描述：临时指针 函数算法描述：利用库函数清屏后重画
12	函数原型：**void MoveLine(line *l, double x, double y)** 功能描述：移动选中移动连接线 参数描述：l，当前线段，x，y 鼠标位置 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据 x 还是 y 方向相对移动。
13	函数原型：**void ZoomLine(line *l, double x, double y)** 功能描述：缩放选中移动连接线 参数描述：l，当前线段，x，y 鼠标位置 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：只能在连接线的末端拖拉缩放
14	函数原型：**node * SearchLine(node *prehead, double x, double**

	y) 功能描述：寻找需要操作的连接线 参数描述：prehead，图形链表头节点，x，y，鼠标位置 返回值描述：相连的图形 重要局部变量定义：temp 重要局部变量用途描述：临时指针。 函数算法描述：根据鼠标位置遍历。
15	函数原型：**double Commute(double x, double y, double x0, double y0, double k)** 功能描述：计算距离。 参数描述：x0，y0，中心位置，x，y，鼠标位置，k，斜率 返回值描述：距离 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：y = kx + b 计算点到直线距离
16	函数原型：**int InRect(double cx, double cy, double w, double h, double x, double y)** 功能描述：检测矩形区域。 参数描述：cx，cy，中心位置，x，y，鼠标位置，w，h，大小 返回值描述：是否位于矩形内部 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据两边距离检测。
17	函数原型：**int InArc(double cx, double cy, double x, double y, double r, double angle, double d)** 功能描述：检测圆形区域。 参数描述：cx，cy，中心位置，x，y，鼠标位置，w，h，大小，angle，旋转角度，d 旋转方向上移动距离 返回值描述：是否位于圆形内部 重要局部变量定义：r 重要局部变量用途描述：距离圆心的距离 函数算法描述：根据到圆心的距离是否大于半径判断
18	函数原型：**text * AddText(text *pretext, double cx, double cy)** 功能描述：添加新文本框 参数描述：pretext，文本链表头结点，cx，cy，文本框左上角 返回值描述：新创建的文本框在链表中的位置 重要局部变量定义：t 重要局部变量用途描述：临时指针。

	函数算法描述：根据现有的初始化样例创建一个副本添加到链表
19	函数原型：**text * DeleteText(text *pretext, text *tt)** 功能描述：删除当前文本框 参数描述：pretext，文本链表头结点，tt，当前文本框 返回值描述：链表中当前文本框下一个节点 重要局部变量定义：t 重要局部变量用途描述：临时指针。 函数算法描述：遍历链表，然后删除节点，释放内存。
20	函数原型：**text * SearchText(text *pretext, double x, double y)** 功能描述：搜寻选中的文本框 参数描述：pretext，文本链表头结点，tt，当前文本框 返回值描述：选中的文本框 重要局部变量定义：t 重要局部变量用途描述：临时指针。 函数算法描述：根据鼠标位置，遍历链表，寻找符合的区域
21	函数原型：**int JudgeText(text *t,double x, double y)** 功能描述：检测文本框移动还是缩放 参数描述：t，当前文本框，x，y，鼠标位置 返回值描述：进行的操作 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据鼠标位置，判断应该进行的操作
22	函数原型：**void MoveText(text *t, double prex, double prey, double x, double y)** 功能描述：移动文本框 参数描述：t，当前文本框，x，y，当前鼠标位置 ，prex，prey，上一个鼠标位置 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：根据当前鼠标位置与上一个鼠标位置的相对距离，移动当前文本框
23	函数原型：**int ZoomText(text *t, double x, double y, int dir)** 功能描述：缩放文本框 参数描述：t，当前文本框，x，y，当前鼠标位置，dir，缩放的方向 返回值描述：返回能否进行缩放 重要局部变量定义：无

	重要局部变量用途描述：无 函数算法描述：根据缩放方向对文本框大小进行操作
24	函数原型：**void DisplayText(text *pretext, text *tt)** 功能描述：更新绘制文本框 参数描述：pretext，文本框链表头结点，tt，当前文本框 返回值描述：无 重要局部变量定义：t 重要局部变量用途描述：临时指针。 函数算法描述：遍历链表更新，并对当前文本框绘制边框
25	函数原型：**void DrawString(text *t)** 功能描述：绘制字符串 参数描述：t，当前文本框 返回值描述：无 重要局部变量定义： k，j，str，l，temp 重要局部变量用途描述：k:行数递增；j，temp 的下标；temp：临时字符串；l：一个字符的宽度；l：一行能够容纳的字符数 函数算法描述：通过另一个字符数组 temp 对当前字符串进行分割，逐行绘制字符串，实现自动换行功能。
26	函数原型：**void Cursor(text *t)** 功能描述：绘制光标。 参数描述：t，当前文本框 返回值描述：无 重要局部变量定义： kline，begin，s 重要局部变量用途描述：kline：光标所在行数；begin：光标所在列数；s：光标前面的字符串 函数算法描述：定位光标的位置，在根据前面的字符串宽度画出光。
27	函数原型：**void AddChar(text *t, char key)** 功能描述：添加字符 参数描述：t，当前文本框；key，需要添加的字符 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：从后向前移位，并将需要添加的字符加入光标位置前
28	函数原型：**void DeleteChar(text *t)** 功能描述：删除字符 参数描述：t，当前文本框

	返回值描述：无 重要局部变量定义：flag 重要局部变量用途描述：判断是否遇到汉字字符。 函数算法描述：从前向后移位，将光标前的字符删除；如遇到汉字字符（char 值小于零），则再进行一次删除操作。

- **music.c**

序号	具体内容
1	函数原型：**void Music(int Choice)** 功能描述：这个函数应用 Windows 系统的库，进行 wav 格式音乐的播放与停止播放。wav 音乐应放在 exe 文件同目录下。 参数描述：对音乐功能的选择。1,2,3 分别对应浙江大学校歌、The Congress Teel 与 Yellow；4 对应停止播放。 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：利用 Windows 系统的库文件，实现音乐的播放与停止。通过对 Choice 的 switch-case 进行音乐选择与停止播放。

- **menu.c**

序号	具体内容
1	函数原型：**void DrawMode(double winwidth,double winheight)** 功能描述：这个函数用于绘制窗口左下角的状态球，用于提示用户当前所处的状态。分为蓝色底色的自由模式、黄色底色的文本模式、绿色底色的图形模式与红色底色的连线模式总计四种模式。通过改变 CreateMode 的数值，改变状态球样式。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： double ModeCircleX, ModeCircleY, r; char* ModeInformation, * ModeColor; 重要局部变量用途描述：ModeCircleX, ModeCircleY 为状态球的位置坐标，r 为状态球的半径，ModeInformation 为状态球内文字内容，ModeColor 为状态球底色。 函数算法描述：利用对 CreateMode 进行 switch-case 选择，确定状态球内文字内容与底色，最后进行绘画。
2	函数原型：**int GetSelection()** 功能描述：用于将局部全局变量 FlagSelection 传递给其他文件内的函数。

	参数描述：无 返回值描述：用户点击的按钮对应的参数值 重要局部变量定义： Int PresentSelection 重要局部变量用途描述：用于暂存 FlagSelection 的值并返回，以方便对 FlagSelection 做置-1 处理。 函数算法描述：无
3	函数原型：**void DrawMenu(double winwidth, double winheight)** 功能描述：利用 simpleGUI 绘制界面上方的菜单栏，并对用户的点击进行监测，实时记录按下按键对应的参数值，储存在 FlagSelection 中。同时改变 CreateMode 的值表示变更模式，对于比较简单的函数（如 Music）直接进行调用。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： static char* menuListFile[],menuListEdit[],menuListMode[],menuListText[], menuListGraphic[],menuListLine[],menuListMusic[],menuListHelp[]; int ReturnFlag, FileSelection, EditSelection, ModeSelection, TextSelection, GraphicSelection,LineSelection, MusicSelection, HelpSelection; 重要局部变量用途描述：8 个 static char*设定了 8 个菜单按钮的内容，对应/个 xxxSelection 用于记录对应的菜单按钮返回值，ReturnFlag 则用于判别应该将哪个 Slection 保存在 FlagSelection 中。 函数算法描述：利用 simpleGUI 中的 menulist 函数进行绘制、监测与保存点击后的返回值；根据不同返回值进行适当的 CreateMode、ReturnFlag 值的修改，最后通过对 ReturnFlag 的 switch-case 操作，将适当的 Slection 保存在 FlagSelection 中，等待其他文件中的函数调用。
4	函数原型：**void DrawStatusBar(LinkStatusInfo PresentLinkStatusInfo, double winwidth, double winheight)** 功能描述：通过传递进来的含有信息的结构体指针，了解文件名、当前用户是否选中了元素，若选中，则进一步了解选中元素的相关属性。将以上信息全部展现在底部的信息栏，以便用户操作。对于文件是否处于

	保存状态，采用在文件名后加上*标志表示未保存，没有则表示已被保存。 参数描述：含有当前选中信息的结构指针、当前窗口的宽度与高度。 返回值描述：无 重要局部变量定义： static char PositionX[100], PositionY[100],MouseX[100], MouseY[100],ItemWidth[100], ItemHeight[100],LineBeginX[100], LineBeginY[100],LineEndX[100], LineEndY[100],PenSize[100]; static char* SelectMsg[] 重要局部变量用途描述：11 个字符串用于储存对应的元素属性，SelectMsg 定义时即储存好所有情况的提示信息。 函数算法描述：根据 PresentLinkStatusInfo 指向的结构中的信息判别当前是否处于选中状态/选中元素类型，进而调用相关信息，用 sprintf 函数进行 double，int 向字符串的转化。最后用 DrawTextString 函数输出在界面底部信息栏中。
5	函数原型：**void DrawIconButton(double winwidth, double winheight)** 功能描述：利用 simpleGUI 绘制界面右侧的图标工具栏，实现常用功能的快捷使用。实现原理采取与 DrawMenu 相同的方法，都将参数传入 FlagSelection，方便统一管理，统一对接。 参数描述：当前窗口的宽度与高度 返回值描述：无 重要局部变量定义： static double IconX, IconY, IconH, IconW; static char* ButtonName[] 重要局部变量用途描述：前四个用于规定图标按钮的位置与高度、宽度。后面的数组定义了图标按钮的名字。 函数算法描述：利用 simpleGUI 的 Button 功能，做出图标工具栏。同样采用与 DrawMenu 相同的参数传递机制，即将参数与 DrawMenu 中统一，传入 FlagSelection 中。

- **example.c**

序号	具体内容
1	函数原型：**void ProcessInitial_1()** 功能描述：该函数为示例 1 的数据初始化函数，主要用于将图形数组和文本框数组转化为链表，并调整三种结构的坐标值 参数描述：无 返回值描述：无

	重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：先调整图形，再调整文本，再调整连接线
2	函数原型：**static void Line_Adjust(MyLine* head, int cur, double dx, double dy)** 功能描述：该函数为连接线线数组的坐标调整函数，可以为每一个线段元素进行坐标平移。 参数描述： head：线段数组 cur：　线段数组长度 dx：　横坐标变动的相对值 dy：　纵坐标变动的相对值 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：遍历数组，调整线段元素的坐标
3	函数原型：**static void Object_ArrayToChain(Link head, int n,double x,double y)** 功能描述：该函数为基本图形结构的坐标调整函数，可以为每一个元素进行坐标平移，并将结构数组相继连接为链表 参数描述： head：图形数组头指针 cur：　图形数组长度 x：　横坐标变动的相对值 y：　纵坐标变动的相对值 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：遍历数组，调整数组元素的坐标，并赋予其与下一个数组元素的后继关系。
4	函数原型：**static void TextBox_ArrayToChain(text* head, int n,double x, double y)** 功能描述：该函数为文本结构的坐标调整函数，可以为每一个元素进行坐标平移，并将结构数组相继连接为链表 参数描述： head：文本框数组头指针 cur：　文本框数组长度 x：　横坐标变动的相对值

	y:　纵坐标变动的相对值 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：遍历数组，调整数组元素的坐标，并赋予其与下一个数组元素的后继关系。
5	函数原型：**static void _showmytext(text* Mytext)** 功能描述：该函数为文本显示函数,遍历文本链表简单地输出文字 参数描述： Mytext：文本框链表头指针 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述： 遍历链表，根据坐标值和字符串内容显示出相应的文本
6	函数原型：**void Example_Time_Process_1(int timerID)** 功能描述：该函数为示例 1 的动画演示函数（用于回调），根据给定的过程让特定的图形改变颜色 参数描述： timerID：计时器标志值，检验该计时器标志值是否是给定计时器的计时器标志值 返回值描述：无 重要局部变量定义： static int times, order, i = 0 重要局部变量用途描述： timers：计时器执行次数 order：　当前对应的流程图图形数组的下标 i：　　模拟变量“i” 函数算法描述：不断执行，根据当前的对应流程框图和模拟变量 i 决定下一步跳转的流程框图，直至模拟程序结束。
7	函数原型：**static void _drawlines_(MyLine* lines, int cur)** 功能描述：该函数为连接线绘制函数，根据连接线属性判断是否绘制箭头 参数描述： lines：线段数组 cur：　线段数组长度 返回值描述：无 重要局部变量定义：无

	重要局部变量用途描述：无 函数算法描述：遍历链表，根据起始点和终止点坐标值和是否是末端线段绘制出相应的线条。
8	函数原型：**double distance(double bx, double by, double ex, double ey)** 功能描述：计算给定端点的线段的长度 参数描述： bx：起始横坐标 by：起始纵坐标 ex：结束横坐标 ey：结束纵坐标 返回值描述：返回相应距离。 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：使用两点间距离公式

9	函数原型：**int ExampleDisplay_1()**
	功能描述：示例 1 的显示函数，伴有初始化功能和代码显示、流程图显示功能。
	参数描述：无
	返回值描述：返回 1 代表页面跳转，0 则页面不变
	重要局部变量定义：
	`static int Back=1,New=1,CODE_;`
	`double winx = GetWindowWidth();`
	`double winy = GetWindowHeight();`
	`int flag;`
	重要局部变量用途描述：
	Back：代表是否改变页面
	New： 决定是否初始化
	CODE_:决定是否显示代码
	winx，winy：当前窗口大小
	flag：代表菜单选项
	函数算法描述：绘制菜单，并通过分支结构(switch 语句)根据菜单选项进行相应的操作；绘制代码；绘制图形、连接线、文本。

10	函数原型：**static void ShowCode(double x, double y, char* code)** 功能描述：代码显示函数，根据给定的坐标和代码字串相应地显示出代码。 参数描述： x,y： 代码显示的坐标

	code：代码内容 返回值描述：无 重要局部变量定义：double Y = y; 重要局部变量用途描述： Y ：代表当前行应显示的代码的坐标 函数算法描述：遍历字符，遇到代码中的换行符就进行相应的换行：具体来说，先将字符’\n’替换为’\0’,然后显示代码，接着恢复’\n’，并下降当前 Y 坐标准备进行下一次显示。
11	函数原型：**static void ShowCode(double x, double y, char* code)** 功能描述：代码显示函数，根据给定的坐标和代码字串相应地显示出代码。 参数描述： x,y： 代码显示的坐标 code：代码内容 返回值描述：无 重要局部变量定义：double Y = y; 重要局部变量用途描述： Y ：代表当前行应显示的代码的坐标 函数算法描述：遍历字符，遇到代码中的换行符就进行相应的换行：具体来说，先将字符’\n’替换为’\0’,然后显示代码，接着恢复’\n’，并下降当前 Y 坐标准备进行下一次显示。
12	函数原型：**void Example_Time_Process(int timerID)** 功能描述：该函数为总的示例动画演示函数（用于回调），根据给定的过程让特定的图形改变颜色。 参数描述：timerID：计时器标志值 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：无
13	函数原型：**int Display_Example()** 功能描述：总的显示函数，伴有初始化功能和代码显示、流程图显示功能。根据子函数返回值决定是否返回继续自身页面枚举值或返回其他值进行跳转。 参数描述：无 返回值描述：返回页面枚举值，如果页面不动返回 ExamplePage_,否则返回 LastPage_ 重要局部变量定义：无 重要局部变量用途描述：无

	函数算法描述：无

- **ExampleMenu.c**

序号	具体内容
1	函数原型：**void DrawExampleMenu(double winwidth, double winheight)** 功能描述：利用 simpleGUI 绘制演示界面上方的菜单栏，并对用户的点击进行监测，实时记录按下按键对应的参数值，储存在 ExampleFlagSelection 中。对于比较简单的函数（如 Music）直接进行调用。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： static char* menuListFile[], menuListShow[], menuListCode[], menuListMusic[], menuListProcessure []; int FileSelection, ShowSelection, CodeSelection, MusicSelection, ProcessureSelection, ExampleFlagSelection; 重要局部变量用途描述：5 个 static char*设定了 5 个菜单按钮的内容，对应/个 xxxSelection 用于记录对应的菜单按钮返回值，ExampleReturnFlag 则用于判别应该将哪个 Slection 保存在 ExampleFlagSelection 中。 函数算法描述：利用 simpleGUI 中的 menulist 函数进行绘制、监测与保存点击后的返回值；根据不同返回值对 ExampleReturnFlag 的 switch-case 操作，将适当的 Slection 保存在 ExampleFlagSelection 中，等待其他文件中的函数调用。

- **file.c**

序号	具体内容
1	函数原型：**status Save(char* filename, Link head, text* thead, int len, int SaveCopyAs)** 功能描述：将图形链表和文本框链表写入指定文件中 参数描述： filename：　要保存的文件地址或文件名 head：　　图形链表头指针 thead：　　文本框链表头指针

	len:　　图形链表长度 SaveCopyAs 是否另存为，1 代表是，0 代表不是 返回值描述：typedef _Bool status; 状态 status 分为 false 和 true 两种，表示函数是否执行成功。 重要局部变量定义： FILE* fp 重要局部变量用途描述：fp:临时文件指针，用于以二进制写模式打开要保存的文件并进行后续的读写 函数算法描述：检查传入参数若传入参数不正常，如指针为空，长度为 0，则执行失败，立即返回。 若 SaveCopyAs 为真，尝试以读模式打开指定文件，若存在，则执行失败；反之继续执行； 若 SaveCopyAs 非真，尝试以读模式打开指定文件，若不存在，则执行失败；反之继续执行； 尝试以写模式打开指定文件，若不存在，则执行失败；反之继续执行； 先写入链表长度，再写入图形函数链表，和文本框链表，最后关闭文件指针。
2	函数原型：**static void SaveTBox(FILE* fp, text* thead)** 功能描述：将文本框链表写入指定文件中。 参数描述： fp:　　写文件指针 thead:　文本框链表头指针 返回值描述：无 重要局部变量定义： text* p; int len = 0; long pos,endpos; 重要局部变量用途描述： p：当前要写入的文本框结构指针 len：记录链表长度 pos:记录文件写指针开始位置 endpos:记录文件写指针写入完毕后位置 函数算法描述： 先占用文件长度储存空间，记录此时位置为 pos; 循环写入链表数据，直至 NULL,同时用 len 记录链表长度 记录此时写指针位置为 endpos,跳到 pos 处写入长度 跳回 endpos

3	函数原型：**static void SaveLine(FILE* fp, struct ConnectLine* cline)** 功能描述：将连接线链表写入指定文件中。 参数描述： fp：　　写文件指针 cline：　连接线总结构指针 返回值描述：无 重要局部变量定义：struct Line* p 重要局部变量用途描述：p：当前要写入的有向线段结构指针 函数算法描述：将 cline 的开始指针赋给 p,进行写入循环，直至 NULL。
4	函数原型：**static void SaveMainChain(FILE* fp, Link head)**功能描述：将图形链表写入指定文件中 参数描述： fp：　　写文件指针 head：　图形结构指针 返回值描述：无 重要局部变量定义：struct Line* p 重要局部变量用途描述：p：当前要写入的图形结构指针 函数算法描述：若 head 非空，将 head 的后继指针赋给 p,进行写入循环，直至 NULL。
5	函数原型：**status Load(char* filename, Link ptrh, text* ptrth)** 功能描述：从指定文件中读取图形和文本框数据，并组织为链表 参数描述： filename：要读取的文件名或地址 ptrh：　图形结构头指针，链表将附于其后继 ptrth：　文本框结构头指针，链表将附于其后继 返回值描述：typedef _Bool status; 状态 status 分为 false 和 true 两种，表示函数是否执行成功。 重要局部变量定义： int ListLen; FILE* fp; 重要局部变量用途描述： ListLen：储存图形链表长度，方便后续读取 fp：　　临时文件指针，用以读模式打开文件并读取 函数算法描述： 尝试以二进制读模式打开文件，若打开失败，则返回 false;否则继续函数继续执行； 读取长度数据储存至 ListLen,若长度太大或小于等于 0 则返回 false;

	否则继续函数继续执行； 依次读取图形数据和文本框数据，将其头部指针分别赋给 prth 和 ptrth,然后关闭 fp 指针。
6	函数原型：**static Link LoadMainChain(FILE* fp, int ListLen)** 功能描述：从指定文件指针中读取图形数据，并组织为链表 参数描述： fp:　　读文件指针 ListLen:　应读取的数据长度 返回值描述：typedef node* Link; 返回图形链表的第一个结构的指针。 重要局部变量定义： Link last = NULL, head = NULL,result=NULL; 重要局部变量用途描述： last: 记录上一个结构的指针，无则为 NULL head: 记录当前结构的指针 result: 链表的第一个结构指针，用于返回。 函数算法描述：循环读取数据，并与上一个结构的指针连接形成链表，直至为 NULL.若某一结构还拥有连接线结构，则先读取其连接线链表。
7	函数原型：**static struct ConnectLine* LoadLine(FILE* fp)** 功能描述：从指定文件指针中读取连接线数据，并组织为链表。 参数描述：fp:　　读文件指针 返回值描述：struct ConnectLine* 类型，代表读取的连接线结构指针。 重要局部变量定义： struct ConnectLine* cline; struct Line* last=NULL,*p=NULL; 重要局部变量用途描述： cline：指向用来存放连接线总括结构的指针 last: 记录上一个结构的指针，无则为 NULL p: 记录当前结构的指针 函数算法描述：循环读取数据，并与上一个结构的指针连接形成链表。
8	函数原型：**static text* LoadTBox(FILE* fp)** 功能描述：从指定文件指针中读取文本框数据，并组织为链表 参数描述：fp:　　读文件指针 返回值描述：text* 类型，代表读取的文本框链表的第一个结点的指针。 重要局部变量定义： text* last = NULL, *head = NULL,*p=NULL; 重要局部变量用途描述：

	last：记录上一个结构的指针，无则为 NULL head：链表的第一个结构指针，用于返回。 p：　记录当前结构的指针 函数算法描述：首先读取链表长度，然后循环读取数据，并与上一个结构的指针连接形成链表
9	函数原型：**void _Destroy(Link head, text* texthead)** 功能描述：该函数通过传入的图形元素链表头指针和文本框链表头指针逐步释放链表空间，其中头指针不会被释放。 参数描述： head：图形链表头指针 texthead:文本框链表头指针 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：先破坏图形链表，再破坏文本框链表。
10	函数原型：**static void MainChain_Destroy(Link head)** 功能描述：该函数通过传入的图形元素链表头指针逐步释放链表空间，其中头指针不会被释放。 参数描述：head：图形链表头指针,同时也是工作变量 返回值描述：无 重要局部变量定义：Link tmp = NULL, h = head; 重要局部变量用途描述： tmp:在 head 指针指向其后继前先保存 head 指针，然后释放其空间 h：　保存链表头指针。 函数算法描述：从头指针的后继结点开始逐步释放空间，且待释放指针在被释放前先被储存，然后指向后继结点，最后释放。如果该结构体的连接线结构指针非空，则先释放连接线结构及其附带的线段链表。
11	函数原型：**static void CLine_Destroy(cline* cl)** 功能描述：该函数通过传入的连接线链表总括结构指针逐步释放链表空间。 参数描述：cl：连接线结构指针 返回值描述：无 重要局部变量定义：line* tmp = NULL, * head = cl->begin; 重要局部变量用途描述： tmp:在 head 指针指向其后继前先保存 head 指针，然后释放其空间 head:工作变量，指向当前的线段结构指针。 函数算法描述：从头指针的后继结点开始逐步释放空间，且待释放指针在被释放前先被储存，然后指向后继结点，最后释放。

12	函数原型：**static void TextBox_Destroy(text* head)** 功能描述：该函数通过传入的连接线链表总括结构指针逐步释放链表空间。 参数描述：head：文本框链表头指针 返回值描述：无 重要局部变量定义：text* tmp = NULL,*h=head; 重要局部变量用途描述： tmp:在 head 指针指向其后继前先保存 head 指针，然后释放其空间 head： 工作变量，指向当前的线段结构指针。 函数算法描述：从头指针的后继结点开始逐步释放空间，且待释放指针在被释放前先被储存，然后指向后继结点，最后释放。
13	函数原型：**void Close(Link head, text* texthead)** 功能描述：该函数通过传入的图形元素链表头指针和文本框链表头指针逐步释放链表空间，然后退出程序。 参数描述： head：图形链表头指针 texthead:文本框链表头指针 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：先破坏图形链表，再破坏文本框链表。然后退出程序
14	函数原型：**int GetListLen(Link head)** 功能描述：该函数通过传入的图形元素链表头指针返回链表的总长度。 参数描述：head：图形链表头指针 返回值描述：返回链表长度 重要局部变量定义： Link p; int i; 重要局部变量用途描述： p:工作变量，指向当前的链表 i:记录到当前为止的链表长度 函数算法描述：遍历链表记录链表长度，直至 NULL

- **save.c**

序号	具体内容
1	函数原型：**int Display_Save(double wholeW, double wholeH,char* filename,Link head, text* thead)** 功能描述：绘制保存交互界面

	参数描述： wholeW：窗口宽度 wholeH：窗口高度 filname：默认文件名 head：　要保存的图形链表的头指针 thead：　要保存的文本框链表的头指针 返回值描述：返回页面枚举值，如果页面不动返回 SavePage_,否则返回 LastPage_ 重要局部变量定义： static char textbuf[FileLen]; static int labelmode, backflag = 1; 重要局部变量用途描述： textbuf：　文本框上的文本储存区 labelmode：为现今 labels[]字符串数组中，应该选择显示的字符串的对应的数组下标； backflag：　表示是否返回，backflag 为 1 代表退出保存界面或进入保存界面，为 0 则代表继续保持该界面 函数算法描述：文本框：输入文件名（默认为 filename）/绝对路径 　　按钮 1：返回 　　按钮 2：保存 　　标签栏 1：另存为（文件后缀.prc） 　　标签栏 2：文件将保存在本地目录//输入示例 "C:\\abc\\xx.prc"//文件名重复！
2	函数原型：**int Display_Load(double wholeW, double wholeH, Link ptrh, text* ptrth, char* Name)** 功能描述：绘制打开交互界面,可以进行文件夹的浏览。。 参数描述： wholeW：窗口宽度 wholeH：窗口高度 ptrh：　要读取的图形链表所依附的头指针 ptrth：　要保存的文本框链表所依附的头指针 Name：　默认文件名 返回值描述：返回页面枚举值，如果页面不动返回 LoadPage_,否则返回 LastPage_ 重要局部变量定义： static char textbuf[AddressLen]; static int labelmode, backflag = 1,Isfull,page=1; 重要局部变量用途描述：

	textbuf:文本框上的文本储存区 labelmode:为现今 labels[]字符串数组中，应该选择显示的字符串的对应的数组下标； backflag:表示是否返回，backflag 为 1 代表退出保存界面或进入保存界面，为 0 则代表继续保持该界面 isfull:若非零，代表当今显示的文件是该地址的最后一页；反之代表还未到最后一页 函数算法描述： 文本框：输入文件名（默认为 new1 或原文件名）/绝对路径 按钮 1：打开 按钮 2：返回 按钮 3：上一页 按钮 4：下一页 按钮 5,6,..13:（各类给定目录下的文件名），若为文件夹可点击打开，若为 dat 文件可加载读取。 标签栏 1：保存（文件后缀.prc） 标签栏 2：文件将保存在本地目录//输入示例 "C:\\abc\\xx.prc"//文件名重复。

3	函数原型：**static int SetFileNames_(int page,char* relafname,char* baseAddress)**
	功能描述：给定页数和地址，将一定数量的文件名字符串数组传递到全局变量 filenames 中
	参数描述：
	page:　　要指定打开的页数
	relafname:　要新增在基础地址上的文件名,NULL 则不增加。
	baseAddress: 基础地址字符串返回值描述：无
	重要局部变量定义：
	`int count = 0;`
	`struct _finddata_t fileinfo;`
	`static char tmp[200];`
	`long handle = _findfirst(tmp, &fileinfo);`
	重要局部变量用途描述：
	count:　　表示当前数到的文件数目
	fileinfo:　定义文件信息结构
	tmp:　　保存当前要打开的带通配符的地址
	handle:　　文件信息结构句柄
	函数算法描述：在内部地址的基础上，根据相对地址（relafname)显示出文件夹下的所有文件夹和.dat 文件，page 代表页数。

	下一次调用改函数时，将会把上一次调用的相对地址增加到其基础地址上。如果 relafname="RESTART",将会清空基础地址.
4	函数原型：**void AddressCatch(char* base, char* source)**功能描述：给定页数和地址，将一定数量的文件名字符串数组传递到全局变量 filenames 中 参数描述： base:　基础地址 source:　新增地址 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：若新增地址为“..”表示文件的上一级，则通过字符数组变量找到最后一个“\\”字符并将其设为“\0”以缩短地址长度；否则将新增地址用分隔符“\\”加在基础地址上。

- **help.c**

序号	具体内容
1	函数原型：**int drawButtons(double winwidth, double winheight)** 功能描述：这个函数用于绘制窗口右下角的返回按钮，用于返回上一级界面。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：页面对应的枚举_enum 中的值，用于确定是否返回上一级页面。 重要局部变量定义： Double h, w, x, y; 重要局部变量用途描述：按钮的高度宽度与位置坐标。 函数算法描述：利用 simpleGUI 内函数绘制右下角的返回按钮。
2	函数原型：**void ProgramHelp(double winwidth, double winheight)** 功能描述：这个函数用于绘制帮助界面-使用帮助界面，给予用户关于程序的使用帮助。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： char*　ProgramHelpTitle[], ProgarmHelpText[]; double　Interval[2]; 重要局部变量用途描述：char*数组中分别记录了使用帮助的标题与正文内容；double 数组中记录了标题与正文、正文与正文间的两个 y 方向

	间隔。 函数算法描述：利用 libGraphics 中的 DrawTextString 函数依次进行标题与正文的绘制。
3	函数原型：**void ShortcutHelp(double winwidth, double winheight)** 功能描述：这个函数用于绘制帮助界面-快捷热键界面，给予用户关于各个界面的快捷键汇总提示。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： char*　ShortcutTitle[],ShortcutText[]; double　Interval[2]; 重要局部变量用途描述：char*数组中分别记录了快捷热键的标题与正文内容；double 数组中记录了标题与正文、正文与正文间的两个 y 方向间隔。 函数算法描述：利用 libGraphics 中的 DrawTextString 函数依次进行标题与正文的绘制。
4	函数原型：**void AboutHelp(double winwidth, double winheight)** 功能描述：这个函数用于绘制帮助界面-关于程序界面，给予用户关于各个界面的快捷键汇总提示。 参数描述：依次为当前窗口的宽度、高度。 返回值描述：无 重要局部变量定义： char*　AboutHelpTitle[],AboutHelpText[]; double　Interval[2]; 重要局部变量用途描述：char*数组中分别记录了关于程序的标题与正文内容；double 数组中记录了标题与正文、正文与正文间的两个 y 方向间隔。 函数算法描述：利用 libGraphics 中的 DrawTextString 函数依次进行标题与正文的绘制。
5	函数原型：**int Display_Help(double winwidth, double winheight)** 功能描述：这个函数用于根据当前的 HelpMode 进行对应帮助页面的函数调用。 参数描述：当前窗口的宽度与高度 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无

	函数算法描述：先绘制按钮，然后根据 HelpMode 的 switch-case 进行选择性绘制。
6	函数原型：**void SetHelpMode(int mode)** 功能描述：这个函数用于根据传入的值，对 HelpMode 值进行修改。起到对帮助界面展示的选择作用。 参数描述：修改 HelpMode 的目标值。 返回值描述：无 重要局部变量定义：无 重要局部变量用途描述：无 函数算法描述：利用传入的 mode 值对 HelpMode 进行赋值。

4 部署运行和使用说明

4.1 编译安装

- **Visual Studio 2019 下的编译安装**
  - **预先准备**

    1）新建一个文件夹，这里以 D:\ProjectSourse 为例；

    2）将以下三个文件夹：

    * 压缩包中的 codes 文件夹；

    * libgraphics 文件夹；

    * simpleGUI 文件夹。

    放入 1）中新建的文件夹中（后两个文件来自于老师给定压缩包 LibGraphics2019Hanoi）。

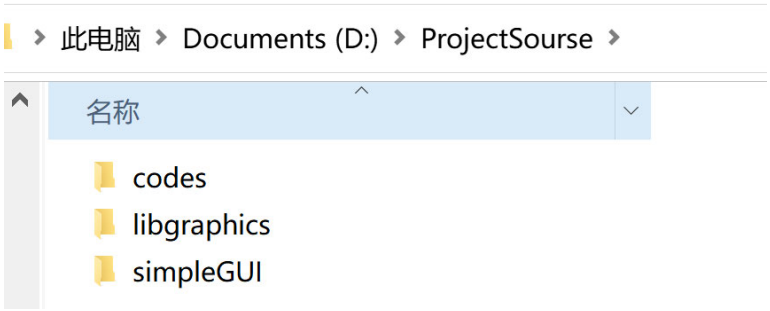


图 4.1-1 预先准备工作-文件夹

  - **新建项目**

1）启动 VS2019，选择“创建新项目”，接下来选择“Windows 桌面应用程序”（C++类型）；

2）为新项目取名字并选择储存路径：名字可以任意取，这里以 Project 为例，路径选择【预先准备】1）中的路径，这里就是 D:\ProjectSourse；

3）勾选“将解决方案和项目放在同一目录中”；

4）点击创建。

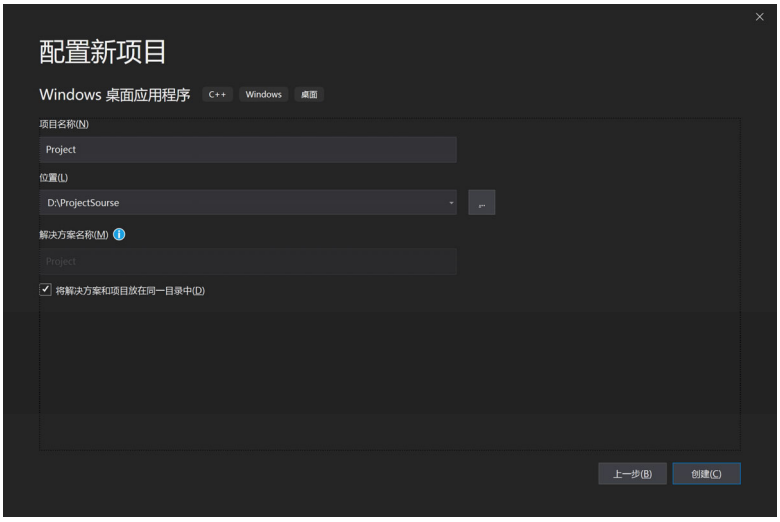


图 4.1-2 创建项目

- **环境配置**

1）移除源文件中的默认文件：Project.cpp：在侧栏中展开“源文件”文件夹，右键单击 Project.cpp，选择“移除”；

2）添加源文件：右键单击“源文件”，依次选择“添加”->“现有项”，浏览目录，打开 codes 文件夹（这里对应的路径为 D:\ProjectSourse\codes），选择所有文件，进行添加（可以拉大浏览框，以用鼠标全部选中后添加，更加便捷）；

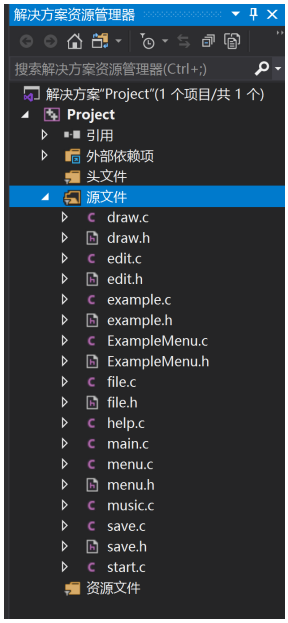


图 4.1-3 环境配置-添加源文件

3）右键单击侧栏中 Project，选择“属性”;

图 4.1-4 环境配置-属性选项

4）进一步依次选择“C/C++”选项，右侧有“附加包含目录”选项：

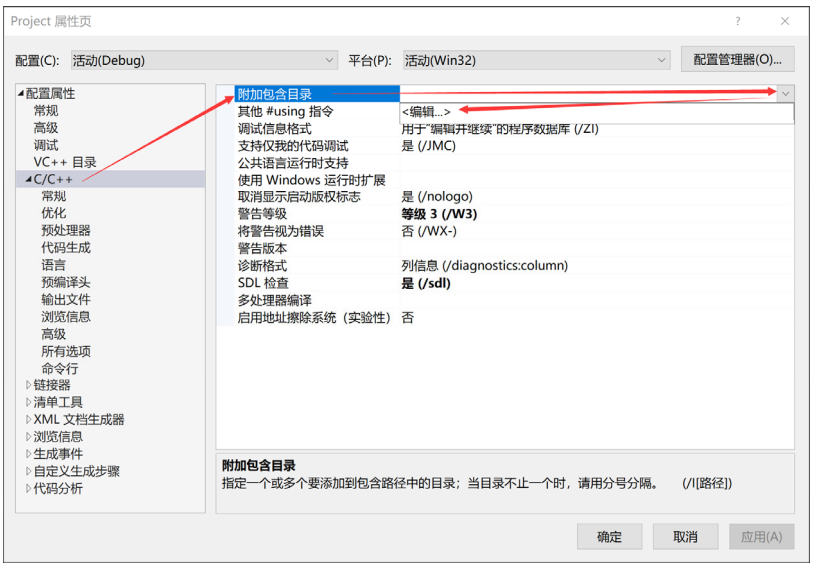


图 4.1-5 环境配置-属性选项-进入附加包含目录

5) 点击编辑，选择文件夹图标的“新行”，随后点击出现的“…”进行文件浏览，依次包含 codes，libgraphics 与 simpleGUI 文件夹(这里目录分别为：D:\ProjectSourse\codes，D:\ProjectSourse\libgraphics，D:\ProjectSourse\simpleGUI)，然后点击“确定”：

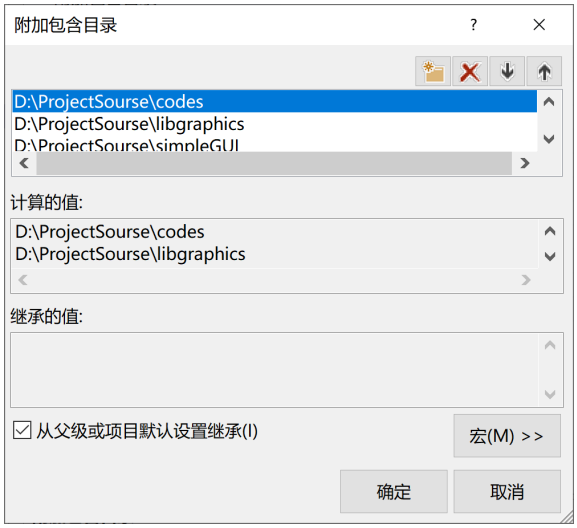


图 4.1-6 环境配置-属性选项-编辑附加包含目录

6) 选择“C/C++”中的子选项“预处理器”，在“预处理器定义”->“编辑”中添加如下 C 语言需要的命令行语句：_CRT_SECURE_NO_WARNINGS，然后点击“确定”(不要删除原有的命令行语句)：

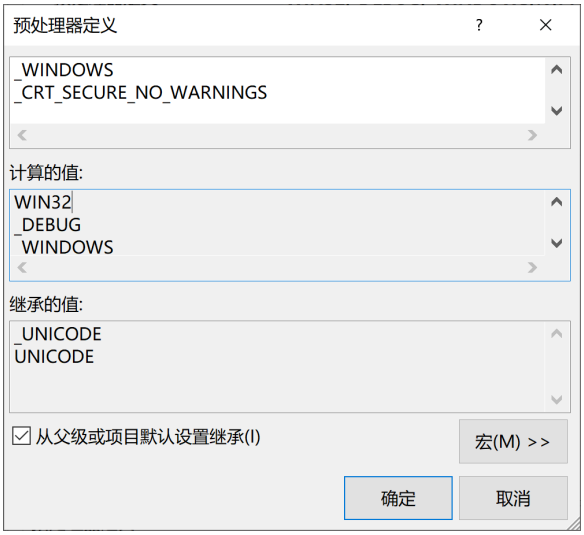


图 4.1-7 环境配置-属性选项-编辑预处理器定义

7）字符集调整：在“配置属性”->“高级”中，将字符集选项置为“未设置”即可。点击“确定”完成所有的属性配置。

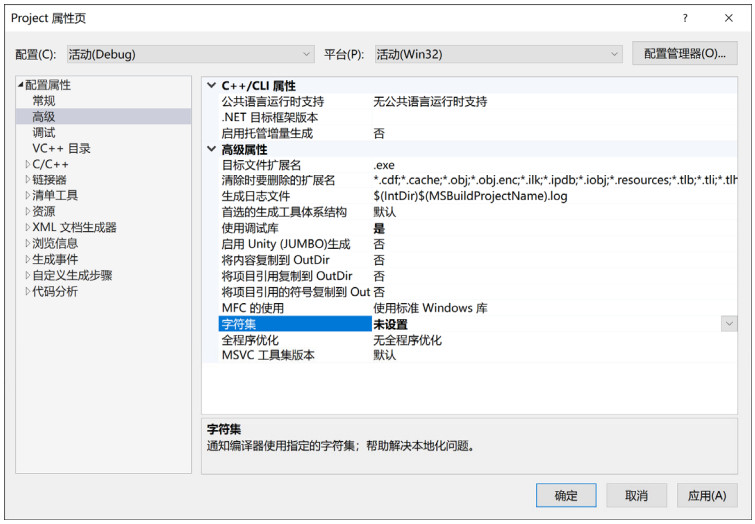


图 4.1-8 环境配置-属性选项-字符集调整

8）输出目录调整：在“配置属性”->“常规”中，设置输出目录，点击“浏览”，在 Project 文件夹下新建一个文件夹作为输出目录（这里命名为 ProjectEXE，对应路径：D:\ProjectSourse\Project\ProjectEXE）。至此完成所有的属性配置，点击“确定”退出选项卡。

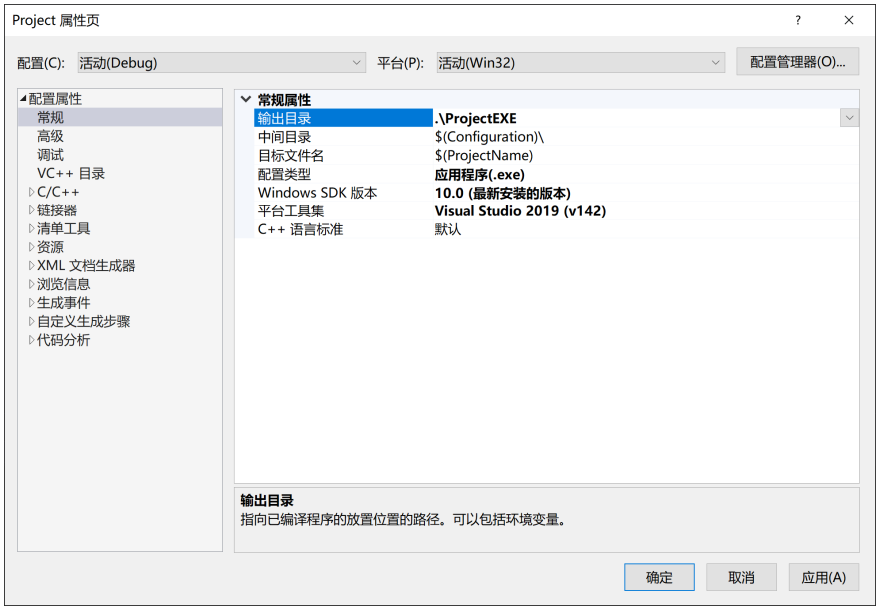


图 4.1-9 环境配置-属性选项-输出路径调整

9）右键单击“资源文件”，如 2）中添加源文件般添加 libgraphics 与 simpleGUI 两个文件夹内所有的文件：

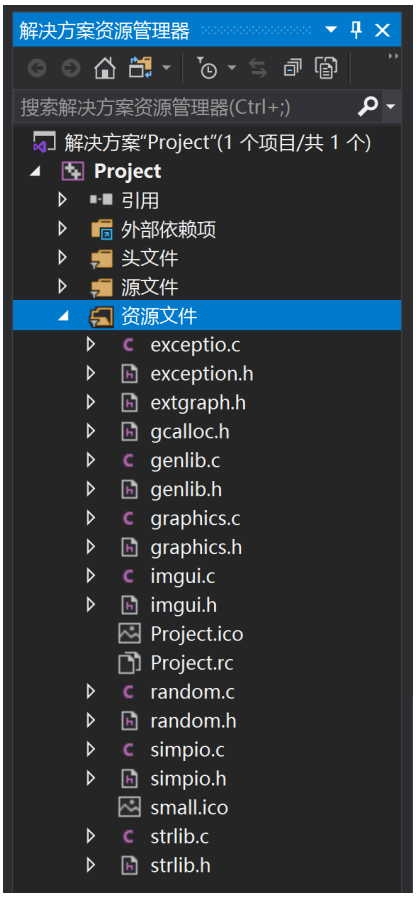


图 4.1-10 环境配置-添加资源文件

10）音乐文件配置：将大程.zip->documents 文件夹下的三个音乐文件

（.wav 后缀）复制粘贴到刚刚配置的输出目录下（对应路径：D:\ProjectSourse\Project\ProjectEXE）

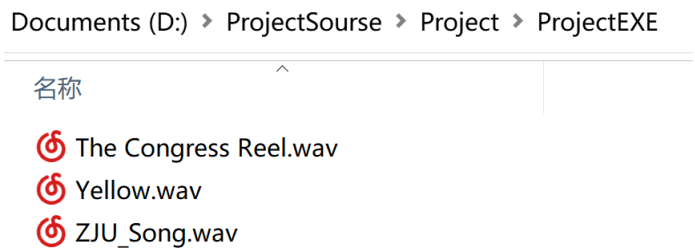


图 4.1-11 环境配置-添加音乐文件

- **编译运行**

1）点击上方的“生成”->“生成解决方案”，进行编译；

2）编译成功后，点击上方的“调试”->“开始调试”，开始运行程序。

● DEV C++下的编译运行

- **预先准备**

解压大程.zip 文件，获得 codes 与 documents 两个文件夹，在编译运行的过程中需要用到 codes 里面的源代码与 documents 的三个.wav 后缀的音乐文件。

- **环境配置**

1）运行 DEV C++，选择 文件 -> 新建 -> 项目（这里命名为 test），选择窗体应用程序（Windows Application），并勾选 C 项目。

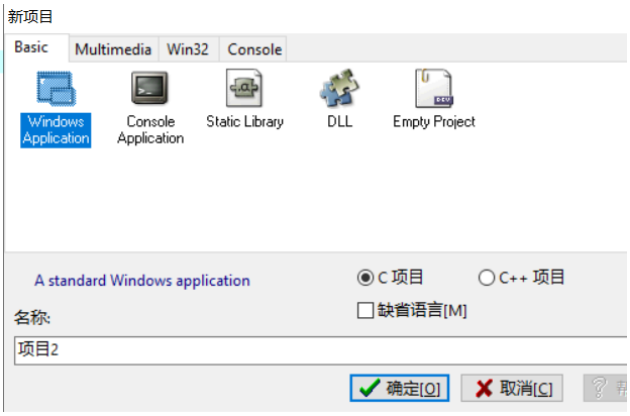


图 4.1-12 DEV 环境配置-新建项目

2）移除预置文件 main.c。

3）添加库文件和源代码：右键单击项目，选择“添加文件夹”，然后添加“库文件”与“源文件”两个文件夹。

随后，右键单击“库文件”，选择添加，将 libgraphics 与 simpleGUI（老师给定）内所有的文件都添加到“库文件”下；相似地，将 codes 中所有的代码添加到 “源代码” 下。

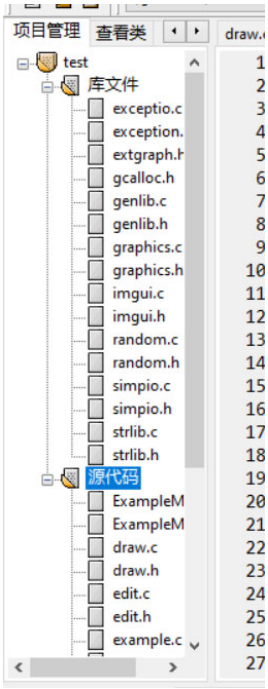


图 4.1-13 DEV 环境配置-添加文件

4）右键单击项目名，选择 项目属性 -> 文件/目录 -> 包含文件目录，依次添加 libgraphics 文件夹、simpleGUI 文件夹和 codes 文件夹的路径（通过点击右下角的橙色文件图标选择，并点击添加）。

图 4.1-14 DEV 环境配置-包含文件目录

5）选择 项目属性 -> 参数 -> 链接 ，添加 DEV 软件 lib 文件夹中的 libwinmm.a 文件，具体的路径通常为：Dev-Cpp\MinGW64\x86_64-w64-mingw32\lib。

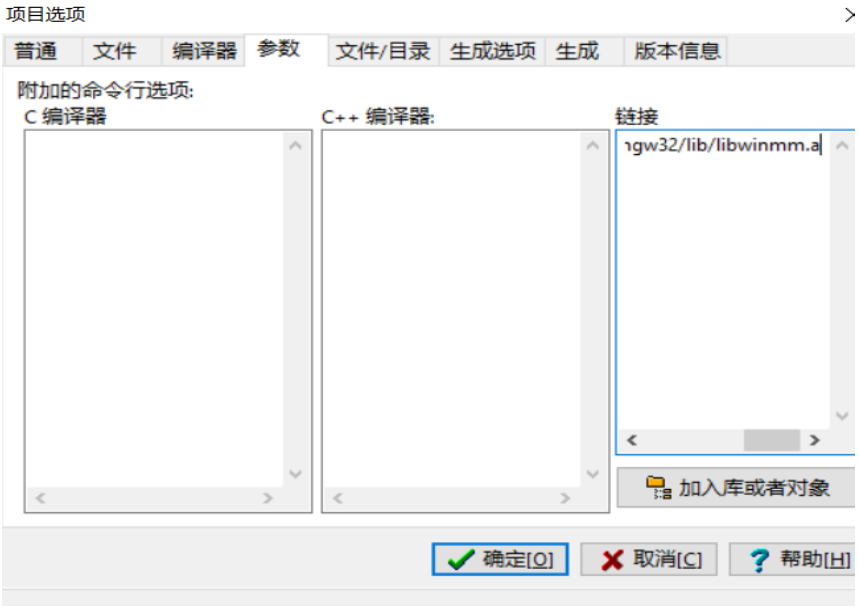


图 4.1-15 DEV 环境配置-参数配置

接着，在菜单栏的工具-编译选项-编译器中加入命令-std=c99

6）编译、运行即可。

4.2 运行测试

- **测试案例一**

测试文件：menu.c 与 draw.c

测试问题：menu.c 中的按键响应后，传递参数到 draw.c 的过程中出现未知问题，导致 draw.c 不能正常响应。

测试方法：在 menu.c 传递参数后与 draw.c 接收参数后，分别添加如下代码（这里仅展示 menu.c 的测试代码，draw.c 中代码类似）：

```
// 通过对ReturnFlag进行switch-case，记录恰当的返回值，并在一些情况下置SaveMode为0
switch (ReturnFlag) {
case 0: FlagSelection = 0; break;
case 1: FlagSelection = FileSelection; break;
case 2: FlagSelection = EditSelection; SaveMode = 0; break;
case 3: FlagSelection = ModeSelection; SaveMode = 0; break;
case 4: FlagSelection = TextSelection; SaveMode = 0; break;
case 5: FlagSelection = GraphicSelection; break;
case 6: FlagSelection = LineSelection; break;
case 7: FlagSelection = MusicSelection; break;
case 8: FlagSelection = HelpSelection; break;
default: FlagSelection = -1; break;
}
// 重置ReturnFlag
ReturnFlag = 0;
InitConsole();
printf("FlagSelection in menu.c is %d\n", FlagSelection);
FreeConsole();
```

图 4.2. 1 测试案例一-menu.c 中测试代码

即通过调用控制台来查看参数。

测试结果：在控制台查看得知，参数已正确传递。但由于鼠标回调函数中的重复设置，造成鼠标移动就会使 FlagSelection 归零，从而覆盖了关键的参数。

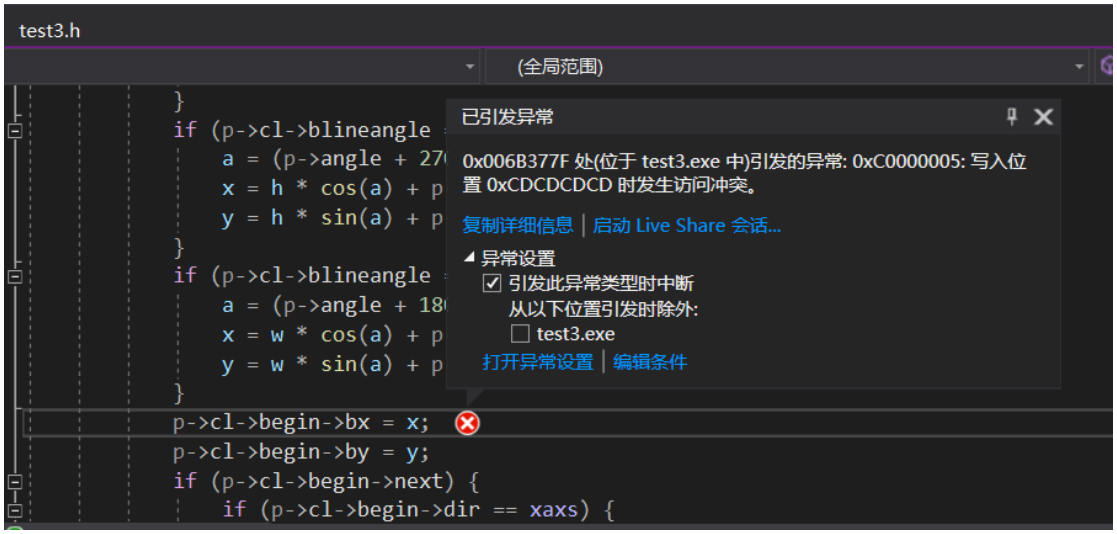
解决方案：重新调整了鼠标回调函数，从而避免了鼠标移动造成的参数覆盖，解决了问题。

- **测试案例二**

测试文件：main.c

测试问题：连接线问题——尝试将绘图功能（edid.c，draw.c）与主文件(main.c)和菜单功能(menu.c)进行合流后，发现连接线无法正常插入，具体表现为按在图形框圆点上后，试图“拉”出一条连接线时，程序直接崩溃。

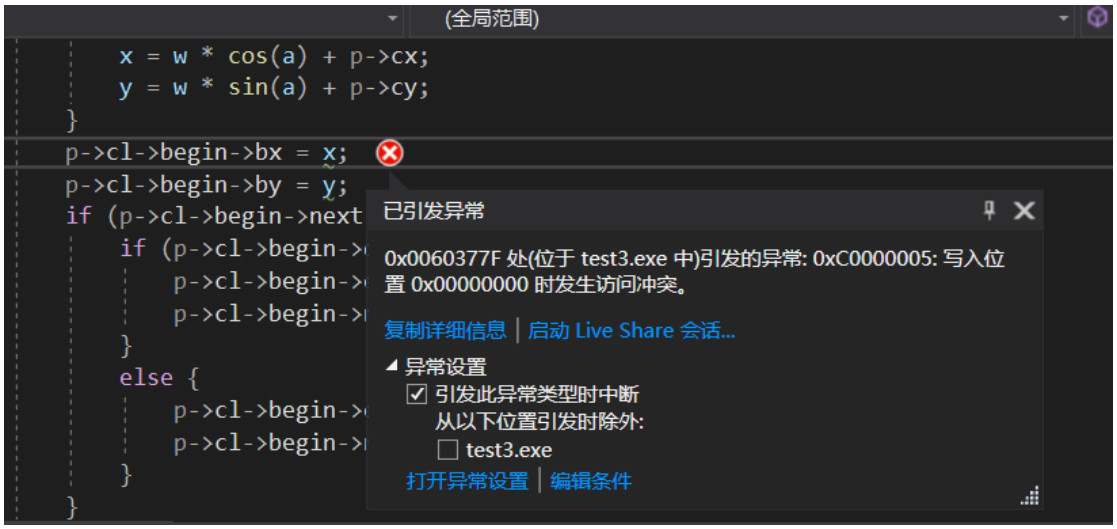
调试时重复此操作，引发中断，原因为 edit.c 的 updateline 函数中，局部变量 p 指向成员所指向的成员变量 p->cl->begin 为野指针（0xCDCDCDCD），即未赋值的指针（如下图）。



测试方法：鼠标回调函数中的显示函数（Display_Draw）包含了调用 updateline 的函数 Liner，因此错误主要是鼠标回调函数变量分配失误造成的。而 updateline 函数中成为野指针的变量 p->cl->begin 的 p 变量是由 main.c 文件中全局变量 p 传入的，谨慎起见，在 draw 页面鼠标绘图函数中的 case BUTTON_DOMN（此时鼠标左击在图形框的拉伸圆点上）的对应代码块内，将内存 p->cl->begin 初始化为 NULL(0x0)。再次调试（如图 1-2），发现在同一处中断，p->cl->begin 为空指针。这意味着 p->cl->begin 初始化后再未使用过，于是寻找本当改变该变量的代码，最终在 BUTTON_UP（鼠标放开）的对应位置，找到了赋值语句 p->cl->begin = head->next;然而，错误是在鼠标按下后移动产生的，即 case MOUSEMOVE 情况下。

解决方案：因而，鼠标按下移动过程中 p->cl->begin 还未被赋为有效值。实际上，在鼠标左击开始生成连接线到移动的过程中，新连接线的数据都储存在以变量 head 为头指针的链表中，通过增加结点以代表新增一段连接线，最终在 BUTTON_UP 中即鼠标放开时，通过赋值语句 p->cl->begin = head->next;连接到全局变量 p 的连接线统括结构 cl 的变量 begin 上。为了修正这个错误，需要在 case MOUSEMOVE 时就修改全局变量 p 指向的连接线的数据。因此，我们在 case BUTTON_DOMN 时就要将 head 指向的链表与

成员变量 p->cl->begin 联系起来。经过一番努力，连接线问题终于被修复。



4.3 使用操作

使用操作分为五个部分，依次从开始界面、绘制界面、演示界面、帮助界面以及其他功能（主要为快捷键、音乐播放等）五个角度对本程序的使用进行介绍。

- 开始界面

下图为开始界面的截图。由上至下为窗口标题、程序标题、开始界面的按钮栏以及左下角的作者信息（互评阶段不完全显示）。

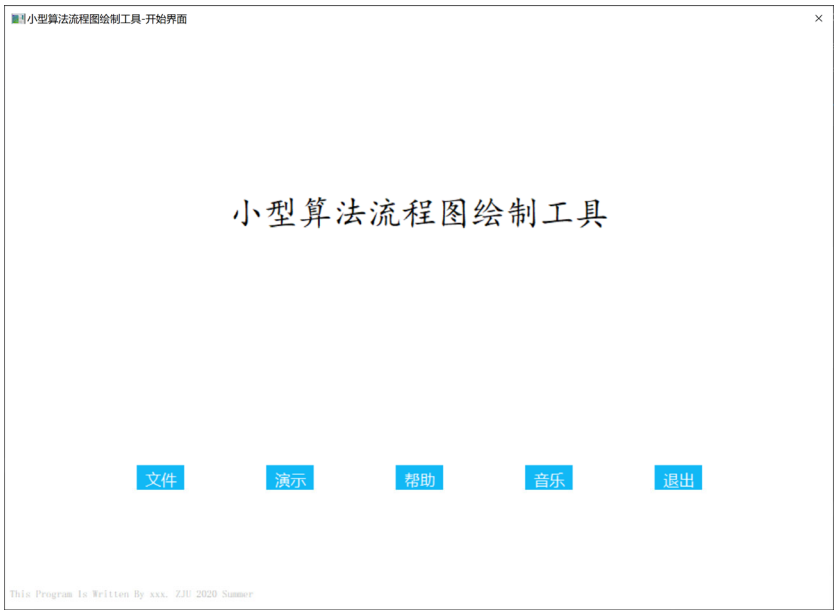


图 4.3 - 1 开始界面

➢ **按钮栏介绍**

文件：分为“新建”与“打开”两个二级按钮。点击“新建”以创建一个新的.DAT 文件（命名为 new.dat），并进入绘制界面开始进行流程图的绘制；点击“打开”以读取一个预先储存的.DAT 文件，对已有的绘制内容进行修改。

演示：仅有“进入演示”一个二级按钮。点击以进入演示模式（详见使用操作-演示界面）。

帮助：分为“使用帮助”，“快捷热键”与“关于程序”三个二级按钮。点击以进入各自对应的帮助界面（详见使用操作-帮助界面）。

音乐：分为“浙江大学校歌”，“The Congress Reel”，“Yellow”与“停止播放”四个二级按钮。对应启动相应的音乐功能（详见使用操作-其他功能-音乐功能）。

退出：仅有“确认退出”一个二级按钮。点击以结束程序运行。

● **绘制界面**

下图为绘制界面的截图。由上至下为窗口标题、菜单栏、绘制界面，右侧的图标工具栏，左下角的模式球以及最下方的信息栏。

图 4.3 - 2 绘制界面

➢ **总体使用逻辑**

1）绘制界面分为四种模式：自由模式、文本模式、图形模式与连线模式。默认为自由模式。

- ✧ 进行各个对象的操作时，需先进入对应的模式 (例如进行文本框的插入、选择、插入、剪切、样式变换等操作时，需先进入文本模式)。
- ✧ 各个模式之间通过菜单栏的“模式”按钮直接进行切换。
- ✧ 处于某个模式时，菜单栏其他模式的功能将处于灰色状态，表示禁用。
- ✧ 对某一元素进行操作前，需在对应模式下，先通过鼠标点击该元素实现“选中”的操作。选中成功的标志为信息栏显示该元素的信息。

2）在选中状态下，可以进行简单的伸缩、拖移与旋转操作。长按移动以进行拖移，长按三个小的黑色圆形并移动以伸缩，长按红色的圆形并移动以旋转。

- ➢ **菜单栏介绍**

文件：分为“新建”，“打开”，“保存”，“另存为”，“返回”，“清空”与“退出”七个二级按钮。

- ✧ 点击“新建”会丢弃当前的文件，重新创建一个新的文件；
- ✧ 点击“打开”会丢弃当前的文件，通过浏览目录打开一个预先保存过的.dat 文件；
- ✧ 点击“保存”与“另存为”都会对当前正在编辑的文件进行保存，区别在于点击“另存为”可以浏览选择保存的路径；
- ✧ 点击“返回”会返回到开始界面，如果没有保存当前文件，则会丢弃；
- ✧ 点击“清空”会清空当前文件的所有内容；
- ✧ 点击“退出”会结束运行程序，如果没有保存当前文件，则会丢弃。

编辑：分为“复制”，“粘贴”，“剪切”，“删除”与“选择”五个二级按钮。

- ✧ 点击“复制”，“粘贴”，“剪切”与“删除”四个按钮，会对已经选中的元素进行对应的操作。需要注意的是，粘贴操作只能进行一次。
- ✧ 点击“选择”，则会退回到自由模式，可以自由地对任意元素进行选

择，并且进行简单的伸缩与拖移。

模式：分为“文本模式”，“图形模式”与“连线模式”三个二级按钮，点击以进行模式切换。

文本：分为“插入文本/退出文本”，“文本选择”，“文本编辑”，“切换颜色”，“字体大小”与“字体风格”六个二级按钮。

- ✧ 点击“插入文本/退出文本”以插入文本与退出文本模式，选项内容与功能自动切换。点击“插入文本”后，通过拉拽以创建一个新的文本框，并开始输入文本内容；点击“退出文本”后，退出文本模式。
- ✧ 点击“文本选择”后，再通过鼠标在文本附近点击以选中一个文本框，并进行样式的变换。当出现文本框的外框时，可以通过拖拽右侧边框与下侧边框调整大小。
- ✧ 点击“文本编辑”以继续编辑当前选中的文本框。
- ✧ 点击“切换颜色”，“字体大小”与“字体风格”以切换对应的样式。

图形：分为“插入矩形”，“插入菱形”，“插入圆角”，“切换颜色”，“切换虚实”，“切换粗细”与“切换填充”七个二级按钮。

- ✧ 点击“插入矩形”，“插入菱形”与“插入圆角”以在某预先设定的位置插入一个对应的图形并选中；
- ✧ 点击“切换颜色”，“切换虚实”，“切换粗细”与“切换填充”以对当前选中的图形做对应的样式（颜色、线型、线号与填充与否）切换。

连线：分为“切换颜色”，“切换虚实”与“切换粗细”三个二级按钮。

- ✧ 预先选中图形后进入连线模式，可以由三个黑色圆圈开始引出连接线，并随鼠标移动增长、转弯。单击以确定终点。
- ✧ 点击“切换颜色”，“切换虚实”与“切换粗细”以对当前的连接线进行样式的切换。

音乐：同开始界面的介绍。

帮助：同开始界面的介绍。

- ➢ 图标工具栏

设置了“插入矩形”，“插入菱形”，“插入圆角”，“清空页面”与“保存文件”五个常用的按钮，用于简化用户的操作步数。

注意，前三个图形的插入操作，仍需在图形模式下才能进行。

- **模式球与信息栏**

模式球：左下角的模式球用于提示用户当前所处模式，分别为蓝色底色的自由模式，黄色底色的文本模式，绿色底色的图形模式与红色底色的连线模式。

信息栏：信息栏分为左右两个部分，左侧显示当前文件的文件名，右侧显示当前选中的元素的信息，两者以“|”作为分隔。

- ✧ 文件名尾部以“*”号来提示用户当前文件的保存状态，不带则处于未保存状态，否则已保存。
- ✧ 选中元素的属性依元素种类不同而现实不同信息，如未选中，则显示当前鼠标的位置坐标。

- 演示界面

下图为演示界面的截图。由上至下为窗口标题、菜单栏、流程图以及右侧的代码区。

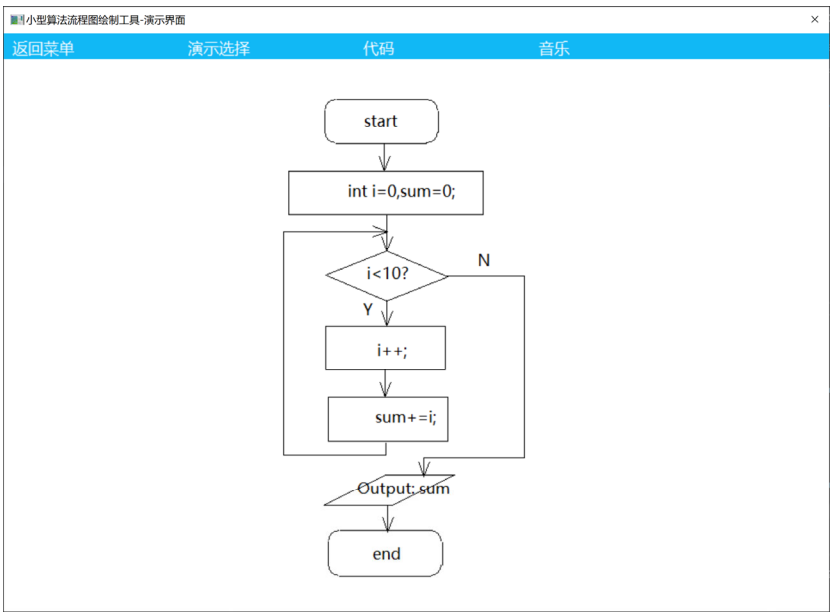


图 4.3 - 3 演示界面

- **菜单栏介绍**

返回菜单：仅含“确认返回”一个二级按钮。点击以返回开始界面。

演示选择：仅含“Program 1”一个二级按钮。点击以开始模拟执行样

例程序，通过区域高亮以显示到达步骤，执行到 end 后自动结束。

代码：仅含“展示代码/隐藏代码”一个二级按钮，自动切换内容与功能。点击以在右上方展示/隐藏代码。

音乐：同开始界面的介绍。

- 帮助界面

下图为帮助界面-使用帮助的截图。帮助界面一共分为使用帮助、快捷热键与关于程序三个子界面。每个子界面的右下角都有一个“返回”按钮，用于返回上一级界面。

使用帮助：分为程序说明、绘制模块、演示模块、特色功能与感谢五个部分，提供基础的使用帮助。

快捷热键：综合了开始界面、绘制界面与演示界面所有的快捷键操作。

关于程序：分为程序作者与联系方式两个部分（互评阶段不予显示），用于声明和获得意见反馈。

图 4.3 - 4 帮助界面-使用帮助

- 其他功能
  - 快捷键

在开始界面与绘制界面都提供了方便的快捷键，统一为 `Ctrl + 字母键` 的形式，具体对应关系在对应界面的菜单栏与帮助界面的快捷热键均可查阅。

  - 音乐功能

程序预置了三首音乐，点击即可实现单曲循环播放，点击其他两首音乐即可进行切换。点击停止播放即可中断音乐的播放。

注意，音乐文件需按 4.1 要求与可执行文件处于相同路径下，否则不能播放。

5 团队合作

5.1 任务分工

朱理真：file.c example.c save.c,main.c(部分)及相应的头文件，负责文件保存、存储和演示功能，同时负责程序整合、优化与部分 bug 修复，负责大程报告 3.1，3.2，3.3，3.41，3.5（部分），4.1（部分），4.2（部分），5.1（部分），5.2（部分）,5.4（部分）,5.5（部分）, 6。

包德政：menu.c start.c help.c music.c 及其头文件，负责各个交互界面的绘制，以及菜单、按钮的优化和修改。负责大程报告 1，2，3.41，3.5（部分及其整合），4.1（部分），4.2（部分），4.3，5.1（部分），5.2（部分）,5.4（部分）,5.5（部分）.

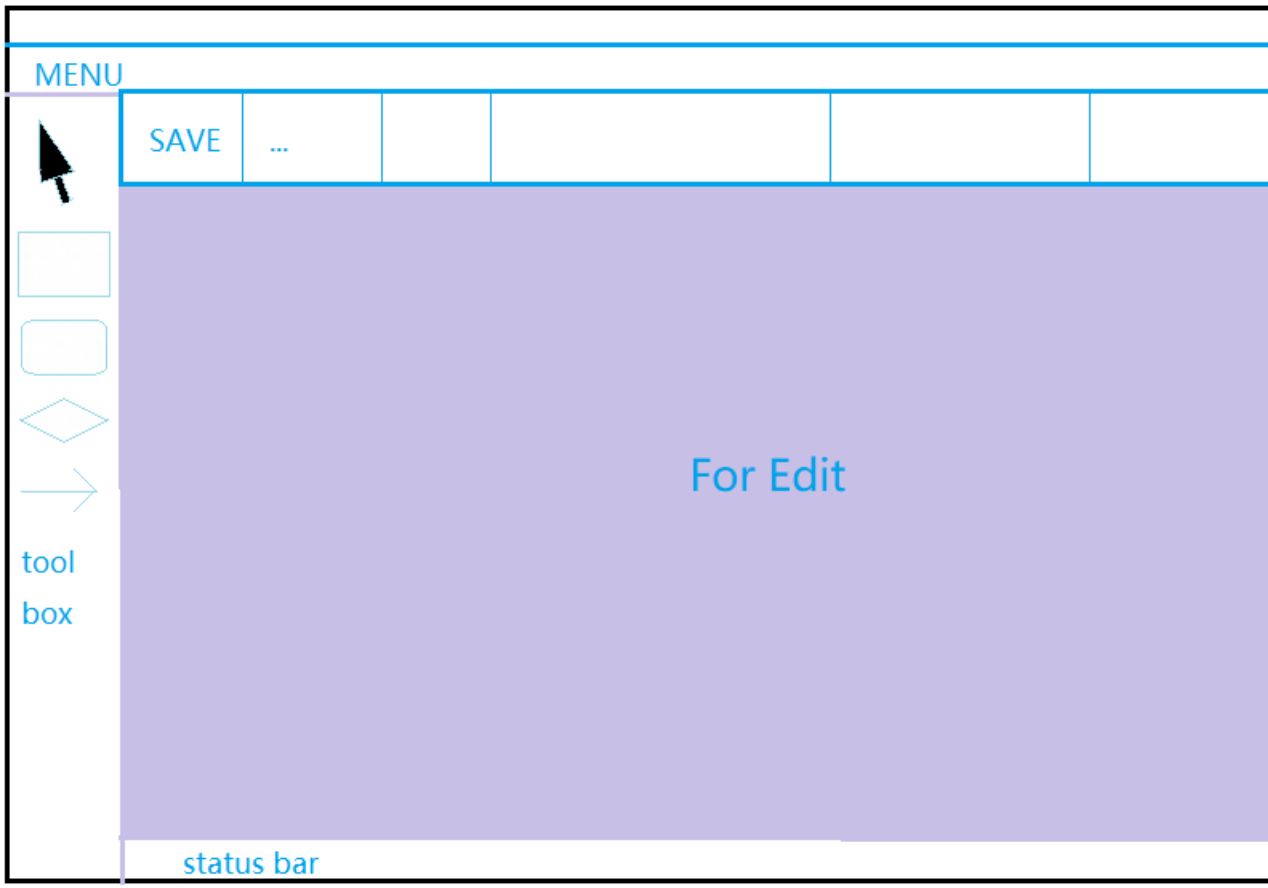
颜天明：draw.c，edit.c，main.c（部分）及其头文件，负责数据结构的设计，以及对象（图形，连线、文本框）的绘制函数、编辑函数，以及绘制图形时各个回调函数的设计。负责大程报告 3.5（部分），5.5（部分）

5.2 开发计划

- **初步构想**

大程作业筹备阶段，小组成员们倾向于选择富有挑战性且具有可行的课题，于是在候选题目中选择了第二个课题，即流程图绘制程序设计。

应对多文件机制，初步设想了几个源代码组成文件，用以实现不同的功能（详情请浏览初期大纲文件 Outline0.7，以下为最初设想，最终版详见 3.4）。



文件名	主要功能
FILE.C	* 处理数据文件（用链表存储图形对象的 * 类型，状态（大小，颜色，线条粗细， * 旋转角度，位置等）数量），包括 * 打开（读取至缓冲区），插入、删除，保存、新建、关闭、退出等
EDIT.C	选择、复制、粘贴、删除、拖移、旋转、缩放，等等
INTERACT.C	实现鼠标、键盘交互以及文本处理
MENU.C	绘制主界面的整体框架，菜单，状态信息栏，工具栏等对象，有快捷键，并关联到对应的处理函数
DRAWOBJ.C	实现绘画，包括矩形、圆角矩形、菱形、箭头线条等多种形状的绘制，包括颜色、线宽、线型、填充等属性，图形框支持文字输入编辑
HELP.C	撰写帮助文档，使用方法等,有回退按钮至主界面

SAVE.C	* 读取文件和保存界面， * 支持输入文件名在本地目录存取， * 也支持输入绝对路径+文件名存取。 * 有能力可以列出本地目录所有文本文件（加了我们规定的后缀名的）进行存取
MAIN.C	主文件，含 Main 函数
START.C	绘制开始界面，有打开，新建，退出等按钮，要求界面美观简洁
EXAMPLE.C	示例一些代码的动态执行过程
MUSIC.C	支持背景音乐

- **工期、数据结构与具体内容安排**

由于初稿规定为 4 月 21 日至 5 月 20 日，故将完成期限提前 10 天，设置在 5 月 10 日，且每 10 天进行一次总结与讨论。根据组长设定的大纲，组员组员三人经过充分讨论后，计划分别完成文件读入读出功能与示例功能、绘制与编辑功能、交互界面设计和模块整合即音乐等辅助功能设计。

对于课题中可选的代码分析并生成流程图并模拟执行要求，考虑到需要编译原理等方面的知识，将此要求暂且搁置，如有余力，可以研究。

数据结构方面，预期准备将文本框、线段、带箭头线段、其他图形置于一个结构体中，为节省空间，使用联合类型使多种图形类型某些数据共用相同的空间，并在结构体中标记类型。所有结构体都连接成一条双向链表，便于搜索、删除、增加。

图形编辑方面，准备为每一个选中的图形对象外嵌一个虚线选中框，为矩形，四角有圆点，可以拉伸；拖动中间部分，可以移动图形；右键拖动，可以旋转；单击某个图形，通过链表从上层到下层搜索，选中某个对象，并将其置于最上层。

程序交互界面方面，拟采用一个窗口多个页面形式，具体如下：

1. 封装不同界面的回调函数的处理过程(记为 Pi(argu),i=1,2,3,...)，统一放在相应的回调函数里(P())；

2. 封装不同界面的 display(绘画)过程(记为 Di(argu),i=1,2,3,...)，统一放在一个函数 (display())里；

3. 设置全局变量 mode，使得 mode=Pi(argu)，且 mode=Di(argu);

4. P() 和 display() 通过过程 switch(mode)选择调用某个 Pi 或 Di;

5.3 编码规范

- **缩进、空格与空行**
  - 缩进统一采用 Tab 键进行缩进，而不使用空格；
  - 在同一类型的变量定义中，每个变量间的逗号后使用一个空格；
  - 关键字后使用空格；
  - 调用函数时，函数名后不加空格，紧跟括号；
  - 二元、三元操作符前后留有空格；
  - 变量定义与函数定义后留有空行；
  - // 形式的注释如需单独占一行，则前空一行。
- **函数、变量命名与定义**
  - 不得采用无意义、易混淆的名字进行变量、函数命名(如 a,b,c,fun 等)；
  - 统一采用双驼峰命名法，如 Music，DrawMenu，DrawStatusBar 等。
  - 函数的左右两个括号{ }，都单独占一行；
  - 整型前缀为 i，指针前缀为 ptr，坐标变量后缀为 X，Y，宽高变量后缀为 W，H；
  - 宏变量大写。
- **if、switch 与循环语句规范**
  - if 语句后必须要有{ }，即使只有一条语句；
  - switch 语句中始终包含 default 语句；
  - 空循环体中使用{ }或 continue，而非一个分号。
- **注释规范**
  - 除特殊情况，在函数外部（如函数前、全局变量定义前等）使用/**/形式的注释，函数内部采用// 形式注释；
  - 特殊地，在全局变量定义的同一行后，采用// 形式注释对该行变量进行注释；
  - 注释比例不得低于 30%

➢ 函数前的注释，需按照以下格式：

```
/*
* 函数名：DrawMode
* 接口：DrawMode(double winwidth,double winheight)
* ————————————————————————————————
* 这个函数用于绘制窗口左下角的状态球，用于提示用户
* 当前所处的状态。分为蓝色底色的自由模式、黄色底色
* 的文本模式、绿色底色的图形模式与红色底色的连线模
* 式总计四种模式。通过改变CreateMode的数值，改变状
* 态球样式。
*/
```

即需要具有函数名、接口与功能说明三个主要部分；分割线与接口等长，功能说明每行与分割线等长。

5.4 合作总结（注意匿名）

- **2020 年 4 月 12 日：第一次视频会议–审题、初步规划界面、外观与分工**

通过视频会议与屏幕共享，小组成员一同审题，手绘了基本的界面样例，根据功能基本确定了文件框架，后续进行了进一步的分工。下面是一些截图记录：

功能	绘图功能	15	基本流程图的图形绘制，矩形、圆角矩形、菱形、箭头线条等多种形状的绘制，包括颜色、线宽、线型、填充等属性，图形框支持文字输入编辑
	编辑功能	20	选择、复制、粘贴、删除、拖移、旋转、缩放，等等
	模拟执行过程	加分功能	实现根据语法分析树生成流程图并模拟执行变化过程功能

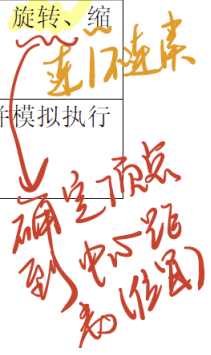


图 5.4．1 第一次会议记录-审题

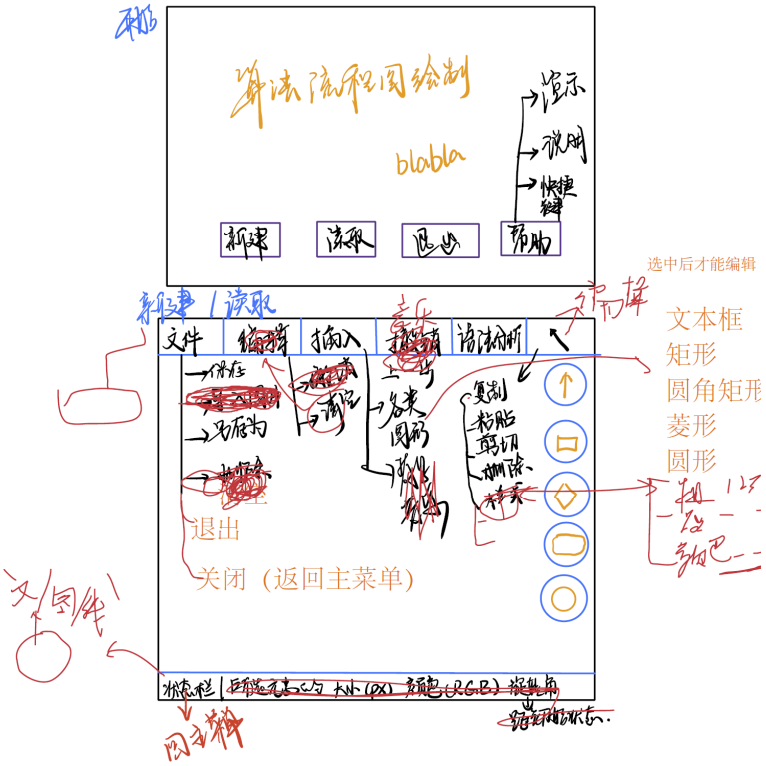


图 5.4. 2 第一次会议记录-界面规划

- **2020 年 4 月 15 日：第一版 OutLine 文件出炉**

首个全局变量、接口约定出炉（后续逐步更新，最终更新至 0.7），方便了组员之间的合作与最后的组装。使得写作具有了规范性。

```
//一些常量和规范：（待补充）
#define true 1
#define false 0
#define MaxStrLen 64
#define string char*
#define ColorNum


const char Colors[ColorNum][MaxStrLen];          //预置颜色，推荐使用函数 DefineColor(color,r,g,b) 来定义颜色;

typedef _Bool status;                            //函数状态（true or false）


/*
使用驼峰命名法
整型前缀 i
指针前缀 ptr
坐标变量后缀 X,Y
宽高变量后缀 W,H
...
不合规范的，需注释说明意义
*/

/*
大家写函数时，遇到子函数不是自己负责的，要看看别的文件的函数是否实现该功能，避免重复造轮子。
*/
```

图 5.4. 3 第一版 OutLine 内容 1

```
//drawobj.c
/*
实现绘画，包括矩形、圆角矩形、菱形、箭头线条等多种形状的绘制，
包括颜色、线宽、线型、填充等属性，图形框支持文字输入编辑。
*/

//常量

//缩放小正方型大小
#define ZoomSqrSize


    //接口：
    void DrawObjects(Link object);    //绘制对象，内部可以包括DrawRect(Link object),DrawRdRect(Link object)等，以及文字和光标输出（对箭头，文字在上方；其他文字在中间）
    void DrawValidZone(Link object);  //显示对象的有效矩形，虚线线条，四角有用来缩放的正方型（其大小与对象大小无关）
```

图 5.4．4 第一版 OutLine 文件内容 2

- **2020 年 4 月 21 日：正式开工后的第一次讨论**

正式开工，小组成员们进行讨论，讨论整体布局，与某些重要细节，其中就有一些挑战点，是我们以前没有经验的问题：

cad 的旋转、拖动、缩放和文本编辑的具体形式； 选中需要链表搜索，探讨最好的搜索方式和链表结构。

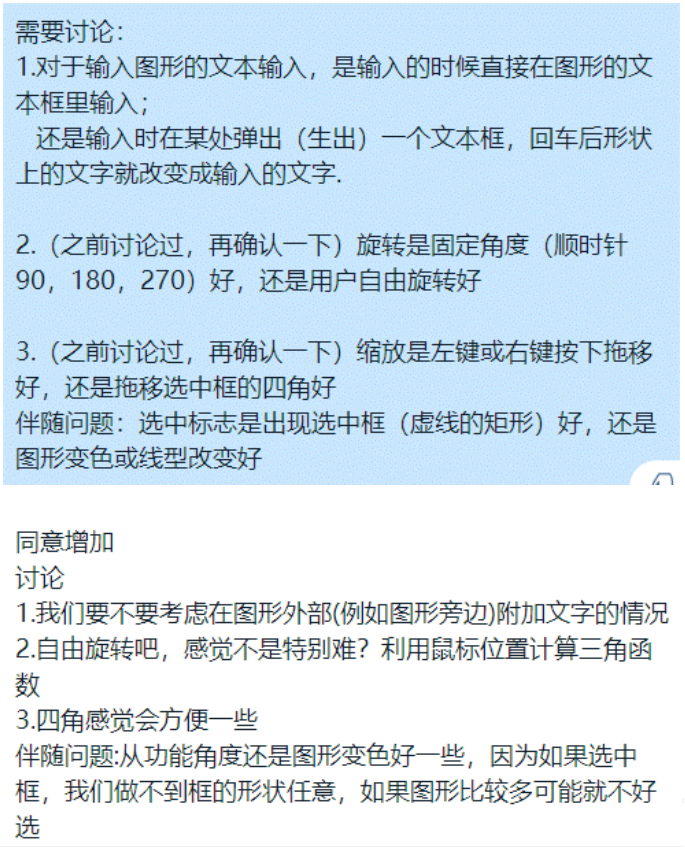


图 5.4．5 小组内的初期讨论（钉钉 app）

- **2020 年 5 月 1 日：正式开工后的某次交流**

经过了 10 天，各个部分都小有成果，包括背景音乐的支持，图形对象模仿一些商业流程图软件的操作形式；开始界面使软件对用户更加友好，示例程序模拟代码执行过程也增加了该程序独有的亮点。

当然，虽然做好了规划，但我们初来乍到，还需学习分工意识和管理意识，一些任务冲突不可避免的，因此需要良好的讨论与交流：

图 5.4．6 小组内的讨论（钉钉 app）

经过一些沟通交流和妥协让步，大家就能更好的分工合作了。

5.5收获感言（注意匿名）

- **团队体会、心得、经验与教训：**

对于小组成员中的每一位，这都是第一次接触以库为基础，近似白手起家地从头开始构建一个可用的程序。在这个过程中，我们在课业之外的时间全身心地投入到程序的源代码编写、小组内沟通协商、互帮互助以及互相提出建议之中。

作为以后学习科研生涯中的第一步，我们都感受到了团队合作的魅力与挑战。

1+1+1>3 可以精准概括团队合作的独特魅力，不但是一份任务分担在三个人的身上带来的异步并行的高效率，更是每一处都汇集了三个人的想法。随着开发过程的不断推进，组员之间不断提出改进意见与更方便用户使用的功能需求，这都一步一步地使得程序走向更好。

但在合作过程中，我们也发现了因为自身经验欠缺造成的困扰，这都是以后团队合作时值得借鉴的宝贵经验。首先是对于编码规范与接口协定要进一步明确与落实，只有打好这个基础才能避免日后无谓的大规模整改。其次是避免水桶效应，要督促每一位组员按时按期完成份内任务，否则可能会拖沓整体进度。最后是对于每一次讨论，都应留下可供回顾的记录（如文字、视频等），以免珍贵的灵光一闪被遗忘。

最后，身为小组中的一员，我们都深深感谢同组伙伴的辛勤付出、热情伸出的援助之手、不厌其烦的沟通交流以及积极分担任务的精神。总而言之，当以后，我们再次身处工程开发人员的位置时，对外响应需求，为用户更好的体验而服务；对内积极沟通，为团队更好的协作而努力。

- **成员 A 自我评价：**

这次的大程作业，对于我来说，既是一次新的体验，也是一次新的挑战。第一次以团队合作的形式完成这样一个大（相对于自己以前写的文件）工程，在工程的初期，难免会出现一些手忙脚乱的情况。面对庞大的工程量，竟不

知该从何处着手。

在具体完成任务的时候，经常会出现一些在写代码时难以察觉的错误，这需要我不断的调试，不断改进，在找出这个错误的同时，也避免之后犯相同的错误。在一些具体的细节实现上，有时会遇到一些瓶颈，常常是规划之前设想的步骤十分容易，具体完成却十分困难，所以在写代码的过程中，我也是产生了很多很好的想法，设计了一些可能较为巧妙的步骤。在后期的整合过程中，其实麻烦不比完成代码少，可能主要的原因还是前期写代码的时没有做好相关的注释，代码的风格有些不友好。

不过，既然是小组作业，大家的合作必不可少。在后期，大家经常熬夜到一两点，相互之间的讨论和交流也是十分频繁。

- **成员 B 自我评价：**

这次大程作业，使我收获良多。多人合作编写程序不同于一个人单打独斗，还需要互相协作的能力。作为小组的组长，除了自己负责的文件存储/读入和示例程序外，还需要监督和引导工程的进行，最后实际由我来进行整合修复工作。因此，我的收获也是多方面的。

在之前 C 语言的学习中，我实际上较少接触过二进制文件的处理和编译原理相关的知识，这鼓励了我去迎接新的挑战，自学获得新技能，同时收获习得新技能的喜悦。

大程的编写，比平时的作业需要更多的工作量，这让我的代码编写能力更上一层楼；同时，正因程序庞大，还需要自顶向下、统筹全局的规划能力；承担整合的任务，更需要我耐心读别人的代码，理解结构，发现错误。

面对这次课题，我自认为是不遗余力地去做的，而且做得还不错。当然，一些教训需要吸取，就是要和组员常沟通，防止剑走偏锋，闭门造车。另一方面，我的沟通协调能力获得了极大的提升，我相信，在以后的项目中，我一定能与他人更好地合作。

- **成员 C 自我评价：**

在人生中第一次的大作业写作中，我感受到了自己的一些可靠的地方以及值得改进的地方。

可靠的地方主要有以下几个方面：1）对任务的细分、规划；2）对规定好的任务能够按时完成；3）积极参与小组讨论，贡献自己的想法与思路；4）善于对讨论结果进行总结梳理；5）小组荣誉感强。

上述可靠的地方具体体现如下:1)在小组的讨论中承担会议纪要的职责，对讨论结果进行整理；2）按时完成任务，时常提醒合作的小伙伴按时完成进度；3）帮助小组成员对他的源代码 Debug 等

值得改进的地方主要有：1）对于 C 语言的一些基础知识掌握还不够熟练；2）代码的可维护性意识还需加强；3）尝试的勇气还需要增加。

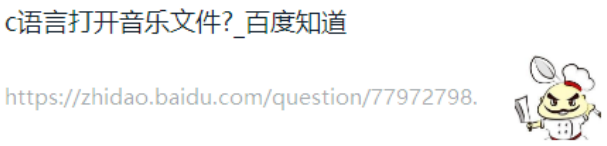
上述值得改进的地方具体体现如下：1）由于同组小伙伴提出了对我自己负责部分的功能新的需求，由于代码可维护性不强，花费了比较长的时间；2）对于小伙伴提出的新功能，一开始不敢尝试；3）对于 C 语言一些基本知识（如 static 变量等）掌握还不够熟练等。

6 参考文献资料

1．C 语言如何打开音乐文件—百度知道

链接：

https://zhidao.baidu.com/question/77972798.html



2．C 语言播放音乐—CSDN 博客

链接：

https://blog.csdn.net/chenaifeiwu99/article/details/80303936

C语言播放音乐_C/C++_chenaifeiwu99的博客-CSDN博客

https://blog.csdn.net/chenaifeiwu99/article/de

3．C 语言读取指定文件夹下的所有文件（各种信息）—CSDN 博客

链接：

https://blog.csdn.net/baidu_30000217/article/details/53098788?utm_source=blogxgwz9

C语言读取指定文件夹下的所有文件（各种信息）_c/c++_LSGOZJ的博客-CSDN博客

绑定脉脉第三方账户获得 授予每个自然月内发布4篇或4篇以上原创或翻译IT博文的用户。不积跬步无以至千里，不积小流...