浙江大学

程 序 设 计 专 题

大 程 序 报 告

2019~2020 春夏学期　2020 年 6 月 3 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

流程图作为算法的一种简单直观的实现方式，贯穿了小学到高中的数学教学。同样，学习编程，也需要先学习程序流程。通过小型算法流程图绘制工具，用图形化的方式，可以让使用者更加简易地理解流程图的实现过程，并且不需要限制于某种特定的编程语言。在使用算法流程图绘制工具的过程中，可以让学习者简单地理解并实现流程图及效果，更加直观和易用。

1.2 目标要求

基于 libgraphics，实现简单流程图的输入、编辑功能，提供文件保存和读取功能。

Extra：通过语法分析实现流程图的模拟运行和实时变化过程观测。

1.3 术语说明

解释器 解释器与编译器不同，它并不通过编译的方式生成目标程序，而是直接利用用户的输入来执行源程序中指定的操作。

2 功能需求分析

本程序尝试实现以下功能：

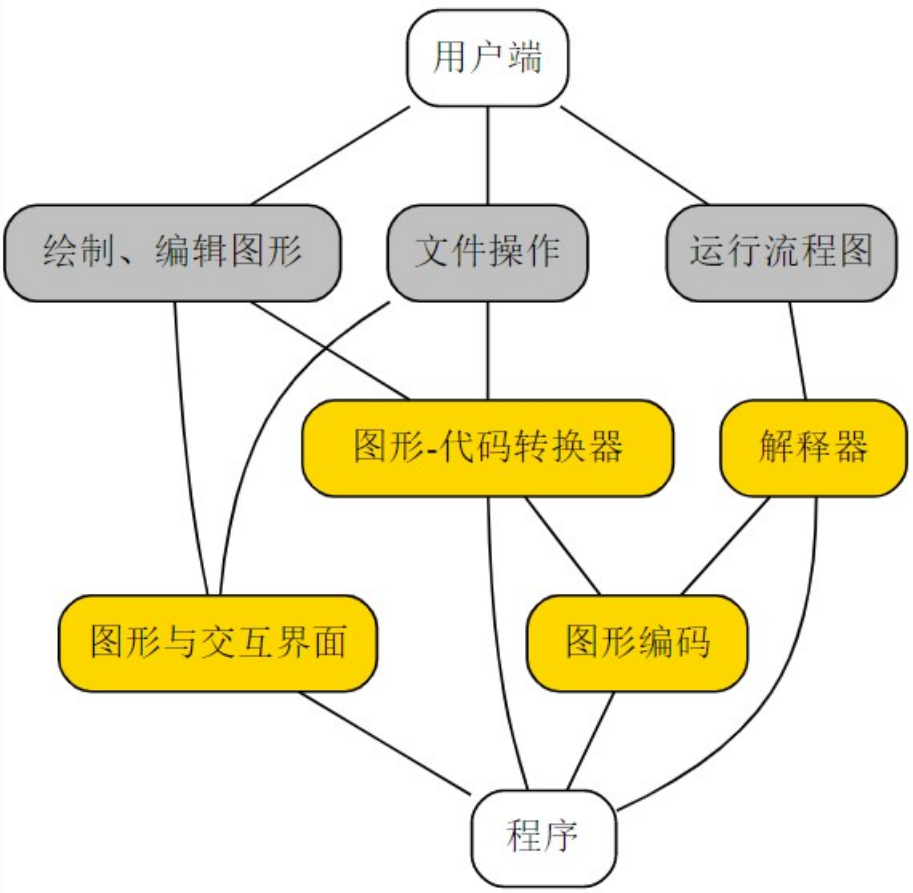
- 用户可以在文件中绘制流程图、编辑图形框中的文字；
- 用户可以选中、复制、粘贴、移动、删除、旋转、缩放流程图中的元素；
- 用户可以运行所绘制的流程图，实时观察每一步的运行状态和结果；
- 用户可以新建、打开、保存文件。

功能的内部要求包括：

- 在允许用户编辑流程图的前提下识别流程图；
- 在运算等过程之前防止出现错误使程序崩溃；
- 用链表存储图形数据及其关系，从而节省空间。

3 程序开发设计

3.1 总体架构设计



用户需要的主要功能包括绘制、编辑图形，文件操作和运行流程图。因此，程序总体上应该包括图形、交互和运行三个部分。

3.2 功能模块设计

为实现前述功能，我们需要：

- 客户端图形界面和交互系统；
  - 图形绘制（各种图形、箭头）
  - 按钮系统
  - 图形和交互界面
  - 交互系统（鼠标、键盘）

- 一套自创的、适合本程序的图形编码；
- 一套图形编码到图形的转换器、一套图形到图形编码的转换器；
- 一套运行图形编码的解释器；
- 一套简单 C 语言代码到图形编码的编译器。

3.3 数据结构设计

程序中涉及到需要生成和保存的数据主要是绘制出的形状和运行产生的变量信息。为了使得这些信息安全地存储，并提高分配空间的利用率，我们采用链表来存储这些信息。具体信息如下：

（一） Arrow：箭头类型

```
/*
 * Type: Arrow
 * ------------
 * This structure is the form of all Arrows.
 *
 * Meanwhile, this structure is also a linked list, which is designed
 * with a "next" and "last" pointer to the next/last node of Arrow.
 * -------------
 * Variables:
 *
 * The "isFilled" variable shows whether the shape is filled.
 *
 * The "width" variable shows the width of lines in pixels.
 *
 * The "posX/Y" variable shows the position(in inches) of the arrow
by
 * a coordinate (x, y) of the beginning point, in the unit of inch.
 *
 * The "sizeX/Y" variable shows the offset of the arrow
 * by the length (in inches) in the x and y direction.
 *
 * The "color" variable shows the color of the arrow by a string, which
can
 * be one of: "Black", "Dark Gray","Gray", "Light Gray", "White",
"Brown",
 * "Red", "Orange", "Yellow", "Green", "Blue", "Violet", "Magenta",
"Cyan".
 *
```

```
 * The "contents" variable is designed to use in if-else structures.
 * We write "Y" or "N" on the arrow to show which way to go.
 *
 * The "arrBegin" & "arrEnd" variable shows where the arrow is
 * from and to which shape the arrow point to.
 */
typedef struct myArrow
{
   bool isDotted;
   int width;
   double posX, posY, sizeX, sizeY;
   string color, contents;
   struct myArrow *next, *last;
   Shape *arrBegin, *arrEnd;
}Arrow;

/* Global variable: arrowHead, arrowTail
 * The head and tail of the linked list
 */
Arrow *arrowHead = NULL, *arrowTail = NULL;
```

（二） Shape：图形类型

```
/*
 * Type: Shape
 * ------------
 * This structure is the basic form of all required shapes:
 * rectangle, rounded rectangle, lozenge and parallelogram.
 *
 * Meanwhile, this structure is also a linked list, which is designed
 * with a "next" and "last" pointer to the next/last node of Shape.
 * -------------
 * Variables:
 *
 * The "isFilled" variable shows whether the shape is filled.
 *
 * The "shapeType" variable shows the actual shape with an integer 1~4,
 * which corresponds with the 4 shapes mentioned above in order.
 *
 * The "width" variable shows the width of lines in pixels.
 *
 * The "posX/Y" variable shows the position(in inches) of the shape
by
 * a coordinate (x, y) of the center point, in the unit of inch.
```

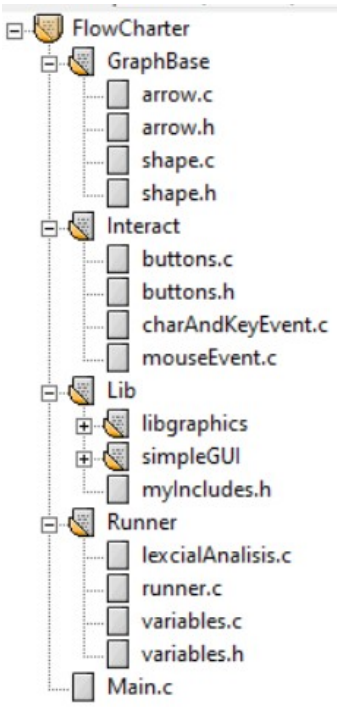
```
 *
 * The "sizeX/Y" variable shows the size of the shape by the length
(in inches)
 * in the x and y direction of its minimum enclosing rectangle.
 *
 * The "color" variable shows the color of the lines (and filler) by
a string,
 * which can be one of: "Black", "Dark Gray","Gray", "Light Gray",
"White",
 * "Brown", "Red", "Orange", "Yellow", "Green", "Blue", "Violet",
"Magenta", "Cyan".
 *
 * The "contents" variable shows the text inside the shape.
 */
typedef struct myShape
{
   bool isFilled;
   int shapeType, width;
   double posX, posY, sizeX, sizeY;
   char contents[101];
   string color;
   struct myShape *next, *last;
}Shape;

/* Global variable: shapeHead, shapeTail
 * The head and tail of the linked list
 */
extern Shape *shapeHead, *shapeTail;
```

（三） Var：变量类型

```
/*
 * Type: Var
 * ------------
 * This structure is the basic form of all variables.
 *
 * Meanwhile, this structure is also a linked list, which
 * is designed with pointers to the next/last node of Var.
 */
typedef struct myVariable{
   string name;
   double value;
   struct myVariable *next, *last;
}Var;
```

```
/* Global variable: varHead
 * The head of the linked list.
 */
Var *varHead = NULL, *varTail = NULL;
```

3.4 源代码文件组织设计



源代码分为 5 个部分：**GraphBase**（图形基础，包括 Shape 和 Arrow 类型的定义、链表组织、新建、查找和绘制等函数），**Interact**（交互，包括按钮的定义、事件的处理，以及鼠标、键盘、字符回调函数的定义），**Lib**（库，包含 Libgraphics 库以及共用头文件 myIncludes.h），**Runner**（运行解释器，包括语义分析器、程序运行器以及 Var 类的定义、链表组织、查找和值的处理等），**Main.c**（主程序）。各个部分代码的简介如下：

（一） GraphBase

arrow.c 和 arrow.h

引用了 shape.h 和 myIncludes.h

定义了 Arrow 类型：

```
typedef struct myArrow
{
   bool isDotted;
   int width;
   double posX, posY, sizeX, sizeY;
   string color, contents;
   struct myArrow *next, *last;
   Shape *arrBegin, *arrEnd;
```

```
}Arrow;
```

定义了 ArrowPair 类型：

```
typedef struct doubleArrow
{
   Arrow *arrowOne, *arrowTwo;
}ArrowPair;
```

定义（.h 中声明）了一系列函数：

```
void NormalizeArrow(Arrow *A);
void DrawArrow(Arrow *A);
void DrawAllArrow();
void SaveArrow(FILE *fp, Arrow *A);
void SaveAllArrow(FILE *fp);
bool InArrow(Arrow *A, double x, double y);
ArrowPair SearchArrowPair(Shape *shape);
Arrow *SearchArrow(double x, double y);
Arrow *CreateArrow(bool isDotted, int width, double posX,
double posY,  double sizeX, double sizeY, string color, string
contents);
bool ReadArrow(FILE *fp);
```

shape.c 和 shape.h

引用了 myIncludes.h

定义了 Shape 类型：

```
typedef struct myShape
{
   bool isFilled;
   int shapeType, width;
   double posX, posY, sizeX, sizeY;
   char contents[101];
   string color;
   struct myShape *next, *last;
}Shape;
```

定义（.h 中声明）了一系列函数：

```
void DrawAllShape();
void SaveShape(FILE *fp, Shape *A);
void SaveAllShape(FILE *fp);
bool InShape(Shape *A, double x, double y);
Shape *SearchShape(double x, double y);
Shape *CreateShape(bool isFilled, int shapeType, int width,
double posX, double posY, double sizeX, double sizeY, string
color, string contents);
bool ReadShape(FILE *fp);
```

（二） Interact

button.c 和 button.h

引用了 arrow.h, shape.h 和 myIncludes.h

定义（声明）了一些重要的全局变量：

```
/* Global Variable: buttons[]
 * -----------
 * This array contains all the buttons needed
 * in the project.
 *
 * buttonStatus: 0 - invisiable, 1 - normal,
 * 2 - overed, 3 - chosen
 */
struct myButtons
{
   bool isFilled;
   int buttonIndex, buttonStatus;
   double posX, posY, sizeX, sizeY;
   string contents, color;
};
struct myButtons buttons[BUTTON_NUM];

/* Global Variables:
 * ---------------
 * isFilledStatus: TRUE or FALSE.
 *
 * DrawStatus: 0~6. Each for: select, rectangle,
 * rounded rectangle, lozenge, parallelogram,
 * arrow, dotted arrow.
```

```
 *
 * SizeStatus: 0~2. Each for: thin, middle, thick.
 *
 * ColorStatus: one in the color table.
 */
bool isFilledStatus;
int DrawStatus, SizeStatus;
string ColorStatus;
```

定义（.h 中声明）了一系列函数：

```
void setButton(bool isFilled, int buttonIndex, int
buttonStatus, double posX, double posY, double sizeX, double
sizeY, string contents, string color);
void initButtons();
void DrawButtons();
void drawSpecialButtons(int index, double posX, double posY);
int SearchButton(double x, double y);
void OverButton(double mx, double my);
void ClickButton(double mx, double my);
void RelatedVisiability(int buttonIndex, int changedStatus);
void ButtonEvent(int index);
```

charAndKeyEvent.c

引用了 buttons.h, shape.h 和 myIncludes.h

定义（声明）了一些重要的全局变量：

```
bool isGettingStr = FALSE, isEditingText = FALSE,
    keyboardCtrlPlus = FALSE, isEditingTextBox = FALSE,
    isEditingInput = FALSE;
int textTag = 0;
Shape *changeTextShape;
```

引用了外部变量：

```
extern Shape textBox, inputBox;
```

定义了字符和键盘回调函数：

```
void CharEventProcess(char c);
void KeyboardEventProcess(int key,int event);
```

mouseEvent.c

引用了 arrow.h, buttons.h, shape.h 和 myIncludes.h

定义（声明）了一些重要的全局变量：

```
Shape *nowShape;
Arrow *nowArrow;
```

引用了外部变量和外部函数：

```
extern bool isFilledStatus, isGettingStr;
extern int DrawStatus, SizeStatus;
extern string ColorStatus, ArrowWords;
extern Shape *changeTextShape;
void CharEventProcess(char c);
```

定义了鼠标回调函数：

```
void MouseEventProcess(int x, int y, int button, int event)
```

（三） Lib

包含了程序使用的图形库（libgraphics）和各代码共同引用的头文件。此略。

（四） Runner

lexcialAnalisis.c

语义分析。具体信息参见 3.5 函数设计描述。

runner.c

运行解释器。具体信息参见 3.5 函数设计描述。

variable.c 和 variable.h

引用了 myIncludes.h

定义了 Var 类型：

```
typedef struct myVariable{
   string name;
   double value;
   struct myVariable *next, *last;
}Var;
```

定义（.h 中声明）了一系列函数：

```
Var *CreateVar(string name, double value);
Var *searchVar(string variable);
double getVarValue(string variable);
void DrawVar();
void initVar();
```

（五） Main.c

引用了 arrow.h, buttons.h, shape.h, variable.h, buttons.h 和 myIncludes.h

定义（声明）了一些重要的全局变量：

```
Shape textBox, inputBox;
const string statusMessage[];
char thisBgmName[80];
```

引用了外部变量和外部函数：

```
void MouseEventProcess(int x, int y, int button, int event);
void CharEventProcess(char c);
void KeyboardEventProcess(int key,int event);
```

定义了一系列函数：

```
void PlayBGM();
void initProject();
void DrawAll();
```

3.5 函数设计描述

由于程序中使用的函数较多，此处仅描述各个功能的核心函数，对过程中的工具函数以及功能简单、浅显的函数不作介绍。

arrow.c 和 arrow.h

```
/* Function: NormalizeArrow
 * Usage: NormalizeArrow(A)
 * -----------
 * This function makes sure that the Arrow is large enough
 * to be operated on.
 */
void NormalizeArrow(Arrow *A);

/* Function: InArrow
 * -------------
 * This function judges whether point (x, y) is
 * near the Arrow enough.
 */
bool InArrow(Arrow *A, double x, double y);

/* Function: SearchArrowPair
 * Usage: nowArrow = SearchArrowPair(nowShape).arrowOne
 * ----------
 * This function search for the latest added 2 Arrows
 * which begin from the given shape.
 * If there is less than 2 Arrow begins from the given
 * shape, the second or both pointer will be NULL.
 */
ArrowPair SearchArrowPair(Shape *shape);

/* Function: SearchArrow
 * -------------------
 * This function search for the latest added Arrow
 * whose area contains point (x, y)
```

```
 */
Arrow *SearchArrow(double x, double y);

/* Function: CreateArrow
 * ------------
 * This function creates a new shape with the given
arguments
 * and adds it to the tail of the linked list, finally draw
it.
 */
Arrow *CreateArrow(bool isDotted, int width, double posX,
double posY, double sizeX, double sizeY, string color,
string contents);
```

shape.c 和 shape.h

```
/* Function: InShape
 * Usage: if(InShape(A, x, y)) ...
 * ----------
 * This function judges whether point (x, y) is
 * in the minimum enclosing rectangle of theShape.
 */
bool InShape(Shape *A, double x, double y);

/* Function: SearchShape
 * Usage: nowShape = SearchShape(x, y)
 * ----------
 * This function search for the latest added Shape
 * whose area contains point (x, y)
 */
Shape *SearchShape(double x, double y);

/* Function: CreateShape
 * ------------
 * This function creates a new shape with the given
arguments
 * and adds it to the tail of the linked list, finally draw
it.
 */
Shape *CreateShape(bool isFilled, int shapeType, int
width, double posX, double posY, double sizeX, double
sizeY, string color, string contents);
```

button.c 和 button.h

```
/* Function: setButton
 * -------------------
 * Set buttons[buttonIndex] with given arguments.
 */
void setButton(bool isFilled, int buttonIndex, int
buttonStatus, double posX, double posY, double sizeX,
double sizeY, string contents, string color);

/* Function: drawSpecialButtons
 * ---------------------------
 * There are several buttons that has
 * special shapes in it. This function
 * is used to draw these shapes.
 */
void drawSpecialButtons(int index, double posX, double
posY);

/* Function: SearchButton
 * ----------------------
 * Given (x, y), search a visiable button
 * that contains this point.
 * If there is not, return -1.
 */
int SearchButton(double x, double y);

/* Function: OverButton
 * --------------------
 * Change the status of the button overed
 * by mouse, meanwhile reset the original
 * overed button to normal status.
 */
void OverButton(double mx, double my);

/* Function: ClickButton
 * ---------------------
 * When a click happens, check whether there is
 * a button selected. If there is, change the
 * status and the global variables related.
 */
void ClickButton(double mx, double my);
```

```
/* Function: RelatedVisiability
 * ----------------------------
 * Given a change on the status of a button,
 * automatically change the status of its
 * related buttons.
 */
void RelatedVisiability(int buttonIndex, int
changedStatus);

/* Function: ButtonEvent
 * ---------------------
 * All the event of drop-down menus
 * and short cut keys.
 */
void ButtonEvent(int index);
```

lexcialAnalisis.c

```
/* Function: LexcialAnalisis
 * -------------------------
 * Analisis the expression "ex" and
 * save the result to the file.
 */
bool LexcialAnalisis(string tex);

/* Function: JudgeAnalisis
 * -----------------------
 * Given an expression in string, return
 * its truth-value (1 or 0)
 * Return -1 if the expression is invalid
 */
int JudgeAnalisis(string jex);
```

runner.c

```
/* Function: initRunner()
 * --------------------------
 * Preparation for interpreter and running.
 * --------------------------
 * Return Value:
 * TRUE - successful
 * FALSE - failed: no "START" node.
 */
bool initRunner();
```

```
/* Function: nextStep()
 * --------------------
 * Run current shape and go to next step.
 * --------------------
 * Return value:
 * -3: invalid: multi "START" node
 * -2: running shape is NULL
 * -1: invalid expression in next shape
 *  0: success
 *  1: no next shape
 *  2: wait for input
 */
int nextStep()
```

variable.c 和 variable.h

```
/* Function: CreateVar
 * ------------
 * This function creates a new variable with the given
 * arguments and adds it to the tail of the linked list.
 */
Var *CreateVar(string name, double value);

/* Function: SearchVar
 * --------------------
 * Return the Var searched by the given name.
 */
Var *searchVar(string variable);

/* Function: getVarValue
 * ----------------------
 * Return the value of the given Var.
 * If there is not, create a new Var in the list
 * with default value 0.0.
 */
double getVarValue(string variable);

/* Function: drawVar()
 * -------------------
 * Print all the variables on the window.
 */
void DrawVar();
```

4 部署运行和使用说明

4.1 编译安装

使用者可以直接通过 FlowCharter.bat 运行程序。如果需要查看工程文件可以在 FlowCharter/demoprj-devc/ 路径下找到工程文件 demo.dev。编译后可以在 FlowCharter/demoprj-devc/output/ 路径下找到可执行文件 demo.exe。

请特别注意：如果您在尝试编译时出现报错，请首先检查您使用的 GCC 是否为 32 位。如果是，且仍然出错，请尝试将 FlowCharter/demoprj-devc/output/路径下后缀名为 .o 的文件一并删除。本程序的编写和测试使用的编译器均为 MinGW GCC 4.7.2 32-bit Release，使用其他编译器可能会导致意料之外的链接错误。

4.2 运行测试

典型测试包括：

1 单部分测试：绘画功能测试

（测试了图形的大小、位置更改，及绘制、复制、粘贴、删除等功能）

2 单部分测试：语义分析功能测试

（测试了多种语句，验证了运行结果）

3 联合测试：运行测试

（测试了多种流程图，验证了运行结果）

纠正了的错误包括：

1 空指针导致的程序崩溃

2 未加 static 导致局部变量指针随机值导致的程序崩溃

3 语义分析功能遇空格崩溃

纠错的一般方法是利用文件输出首先定位出现崩溃的位置，然后输出这里参与运算的变量，发现问题后进行修改，

典型测试和纠错的截图展示：

	字符串	描述
1	a = b	用 = 连接的赋值语句
2	a=b a= b a =b	空格的位置和数目不定
3	abc = b + cd	变量名的长度不定； 运算后赋值：+, -, *, /, %
4	a = b + c * d	操作数的数目不定； 存在优先级问题
5	a += cd	存在赋值运算符 +=, -=, *=, /=, %=
6	a = -a	存在负号
7	a = a + 1.2	存在字面量（常量）
8	a = (b + c) * 3	存在括号

```
else if(isChangeSize)                    用于测试的输出
{
    fp = fopen("test.txt", "a");
    fprintf(fp, "(1 <%p>", nowArrow);
    fclose(fp);
1   if(nowArrow >= 0x10000)
    {
        nowArrow->sizeX += mx - omx;
        nowArrow->sizeY += my - omy;
        omx = mx;
        omy = my;
        break;
    }
```

```
<00000000>(1 <00000000>(1 <00000000>(1 <00000000>(1
<00000000>(1 <00000000>(1 <00000000>(1 <00000000>(1
<00000000>(1 <00000000>(1 <00B0BFB8>(1 <00B0BFB8>(1
<00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1
<00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1
<00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1
<00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1
<00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1
<00B0BFB8>(1 <00B0BFB8>(1 <00B0BFB8>(1 <00000001>(1
<00000001>(1 <00000001>(1 <00000001>(1 <00000001>(1
<00000001>(1 <00000001>(1 <00000001>(1 <00000001>(1
<00000001>(1 <00000001>(1 <00000001>
```

4.3 使用操作

参见文末附件:《小型算法流程图绘制工具 FlowCharter 用户使用手册》

5 团队合作

5.1 任务分工（互评版略）

5.2 开发计划

其中 橙色表示小组讨论，蓝色表示关键文档 DDL，绿色表示项目开发进程，红色表示代码 DDL

- 4.30：提供《选题分析》。
- 4.30：**开题讨论**。主要讨论选题、时间规划、基本要求、任务分配等内容。确定题目和基本任务分配。
- 5.2 之前：完成《报告》中第 2 部分（功能需求分析）和 5.1 ~ 5.3 部分（团队分工、开发计划、编码规范）的内容。

- 5.3 - 5.13：完成 **程序架构和大多数功能** 的实现。此过程中，随时完成《报告》中 3.5 部分（函数设计描述）、5.4 部分（工作记录）、6 部分（参考文献资料）的个人填写。
- 5.7：**中期讨论**。主要讨论前期开发中的问题，如功能实现问题、文件填写问题、代码规范问题、程序架构问题等。**尤其是程序架构中接口的要求和提供**。
- 5.10 之前：完成《报告》中第 3.1 - 3.3 部分（总体架构设计、功能模块分析、数据结构分析）的内容。
- 5.13：**提交所有 *.h 文件**。此后尽量不要对 .h 文件中提供的接口作更改。如有必要，即时进行告知。
- 5.14：**架构解释**。各位对各自的程序架构（.h 文件）作解释。对接口有要求的成员如有需要，可提出接口修改或增删的建议。
- 5.14 - 5.20：完成 **全部功能** 的实现。与 iv 一样，此过程中，随时完成《报告》中 3.5 部分（函数设计描述）、5.4 部分（工作记录）、6 部分（参考文献资料）的个人填写。
- 5.20：**提交所有 *.h 和 *.c 文件**。
- 5.21：**整合讨论**。各位对目前其他成员需要修改的部分提出意见和建议。讨论整合、测试以及后期文件编写的具体事项。
- 5.21 - 5.25：完成程序整合，需要 **形成功能完整的可执行文件**。
- 5.25：**提交工程文件和 .exe 文件**。
- 5.26 - 5.27：试用、测试程序。整理存在的问题和可以改进的地方。
- 5.28：**测试讨论**。交流测试情况，分配 3.4 部分（源代码文件组织设计）、4 部分（使用说明）编写任务。
- 5.28 - 6.3：继续测试、改进程序。
- 5.30 - 6.2：交付 3.4 部分（源代码文件组织设计）、4 部分（使用说明）、5.4 - 5.5 部分（总结、感言）。
- 6.4：**最终讨论**。解决最后问题。
- 6.5：**交付作业**。

5.3 编码规范

本规范参考课程提供的《C 编码规范》一文，同时根据作业相关要求有所修改。本规范对仅一些基本规则进行规定，未提及的规则以个人习惯为准。

1．缩进和对齐

a．同一代码块中采用同一缩进。

2．空格

a．逗号、分号后留空格，除非其后即为换行；

b．二元操作符前后留空格，除非其位于 for, while, switch, if 语句中且表达式较长。

3．空行

a．各个结构体、枚举、函数定义之后都应留有空行；

b．在一个函数体中，应用空行将逻辑不连续或可分块的代码段分开。

4．大括号

a．代码中所有表示代码块的大括号独占一行；

b．if, while, for 等表达式后面的语句必须加大括号，即使只有 1 个语句甚至没有语句；

c．若循环体为空，应使用 { continue; } 。

5．命名

a．所有命名均使用英文单词，除非其在小作用域内作为循环指示；

b．尽量避免命名中出现数字编号，除非逻辑上的确需要编号。

c．对简单的英文单词，应使用 my 等前缀来防止与保留字冲突，如 MySort();

d．所有变量名、文件名第一个单词小写，第二个单词开始首字母大写，如 maxLength；

e．所有函数、结构体、枚举每个单词首字母大写，如 GetLength()；

f．所有常量（包括宏定义的常量、const 常量和枚举中的枚举元素）全字母大写，用下划线分隔各个单词，如 MAX_LENGTH。

6．注释

a．所有注释必须使用 /* */ 而不是 //（防止续行符等导致的错误）；

b．所有注释均应使用英文，以防文件转移时发生的编码问题；

c．所有文件和函数头部必须进行注释，格式可参见 3.5 函数设计描述 一节；

d．所有注释均应保证不会透露个人和小组信息；

e．对较长或多重的大括号，应在结束处注释说明其对应的前括号以便阅读；

f．在有理解难度的语句、变量等处应加注释，注释加在代码的右边或上方。

7．其他要求

a．**头文件保护**。所有头文件都应使用 #define 防止其被多重包含。

b．**标识符唯一性**。声明在某内部作用域的一个标识符不应屏蔽了外部作用域的标识符。

5.4 合作总结

2020 年 4 月 30 日：《选题分析》成型，组员通过语音会议完成选题。

2020 年 4 月 30 日：组员详细讨论了基本要求、规划和任务分配。

2020 年 5 月 1 日：面对组员出现的临时问题，调整了规划和任务分配。

2020 年 5 月 7 日：组长完成了架构的初步设计，并给组员分配了具体的代码接口要求，并给出了实现方式的简要提示。

2020 年 5 月 13 日：.h 文件接口的交流如期进行。小组成员们交流了遇到的问题和基本的编程情况。

2020 年 5 月 23 日：完整的 .c 文件略延期收集整合。小组成员对当前的程序进行了单部分测试，发现了一些问题，并交流、解决。

2020 年 6 月 1 日：小组成员给程序增加了 BGM 功能，提高用户体验。

2020 年 6 月 4 日：程序确认完成，进行最后测试。

附 - 一些截图：

2 小型算法流程图绘制工具

H4 **2.1 描述**

实现简单流程图的输入、编辑功能，提供文件保存和读取功能。可选：参考结合编译原理抽象语法树有关知识，对简单C语言代码能够绘制程序流程。

2.2 重难点

- 形状绘制及其属性
- 编辑功能
- [模拟执行过程]（需要学习语法分析树相关）

3 图书管理系统

3.1 描述

设计一个图形用户界面的图书管理系统，具有图书信息增加、查询、修改、删除；图书借书、还书；用户登陆、注册、审核；系统参数管理等功能，且功能不限于下表，可扩展。

3.2 重难点

- 编辑功能
- 统计功能
- 检索功能（尤其是模糊检索，需要学习正则表达式相关）
- 排序功能

具体需要完成的任务：

请在文件 **lexicalAnalisis.c** 中定义函数：

```
int LexicalAnalisis(char *str)
```

对给定的字符串 str 进行识别。成功识别返回 TRUE，失败返回 FALSE。

其中 TRUE 和 FALSE 分别用宏定义：

```
#define TRUE 1
#define FALSE 0
```

然后编写 main() 函数，用适当的样例对这个函数进行测试。

首先，请学习中缀表达式写成后缀表达式的方法（百度有很多）。

注意到百度上的相关代码都是对数字进行运算，这里我们需要对其进行改进。

我们需要对表示表达式的字符串 `str` 进行逐字遍历，将其按不同内容（变量、运算符、数字）区分开来，并随时注意是否出现错误，出现错误直接 **`return FALSE`**。

> 这三部分内容的字符构成有如下特点：
>
> 本程序中，我们要求变量名称里只出现大小写字母以及下划线。
>
> 本程序中，表达式允许出现的运算符只有 `+ - * / % ( )`。注意 `-` 有可能表示负号。
>
> 数字可能是小数，因此可能存在小数点，另外只出现阿拉伯数字。本程序中，我们不允许使用科学计数法表示数字。
>
> 由此分析，在识别过程中可能发现的错误包括（可能需要补充）：
>
> - 出现不在上述内容中的符号，如等号；
> - `1.02.2` 这类存在多个小数点的数字；
> - 变量和数字的直接接触、两个运算符的直接接触等。

为啥

```
402     switch(b=*(ex+assignment-1))
403     {
404         case '+':way=1;break;
405         case '-':way=2;break;
406         case '*':way=3;break;
407         case '%':way=4;break;
408         default:way=0;break;
409     }
```

你这里way=4是%

```
457     if(trans()==1){
458         if(compvalue()==1){
459             switch(way){
460                 case '1':answer=answer+getVarValue(vari);break;
461                 case '2':answer=answer-getVarValue(vari);break;
462                 case '3':answer=answer*getVarValue(vari);break;
463                 case '4':answer=answer/getVarValue(vari);break;
464             }
465             fprintf(fp, "%f", answer);
```

但是这里是除

QAQ

你可不可以试试加一个函数 就是做判断的 给一个判断语句 比如 a+d>=b+c 这种 然后返回1或者0

已读

差不多实现就是

已读

因为之前已经写好了计算 a+b 这种东西的函数

已读

所以这里应该就是读取一下判断运算符然后比较一下值QAQ

已读

你在我给你发的文件上改吧

已读

你把bgm.wav放在demo.c一个路径下面

6月1日 11:32

然后main.c playsound的那个函数换成这个

```
PlaySound("bgm.wav",NULL,SND_FILENAME | SND_ASYNC| SND_LOOP);
```

2020/6/4 3:48:54

测试稿.rar (32.33MB)

成功发送离线文件，文件助手暂存7天

打开 打开文件夹 转发

5.5 收获感言

团队心得

要做出这样一个流程图绘制工具，对我们来说是一次挑战，也一次对于我们小组中每个人组织能力，学习能力，策划能力等更方面能力的考察，从开始的规划，到任务分配后每个人抓紧时间学习自己还未掌握的知识，到中途大家努力做好自己的部分，到最后程序做出时大家的调试，这个过程中我们需要相互配合，相互指导，相互学习，在面对这样的挑战，我们学到了许多 C 语言的新知识，看到了其中的神奇之处，也明白了小组之间只有相互配合才能做出令自己满意的成果。

在本次项目过程中，我们也遇到了许多问题，让我们积累了许多经验。我们最重要的收获是学习了合作完成工程的方法。首先完成各个部分架构的设计是十分重要的，这有助于组员们互相交流需要其他同学提供的接口，从而避免之后再进行更改。每位组员自己的代码部分的测试也需要十分严格，只有这样才能让最后成型的大程序出现更少的问题。

自我评价 – 组长

在项目开始之前，我与组员分析了各个选题，并选择项了相对有挑战性的流程图。在做程序的过程中，我安排了前期的任务分配，对每个人需要做的部分做了详细的规划。在这个过程需要我提前对程序的实现有完整的构想，并对细节进行分配。在后期做程序的过程中我担任了主要的图形化制作以及流程图的实现功能，做出这样的功能需要我去额外学习许多知识，对于学习能力是一次较大的考验，在付出巨大的时间和精力的同时，对自己的能力也有了进一步的认识和提升，在以后我也一定会更加努力。作为组长需要协调和督促组员任务进度，也让我提升了自己的组织能力和领导能力。总之，这次的大作业在各方面都对我有巨大的提升。

自我评价 - 组员（注：本组仅 2 人）

在大作业的过程中，我学习到了很多新知识，认识到了自己很多的不足，在大程序的过程中，我所承担的任务虽然不多，但由于是没有学过的语法分析相关的知识，因此对我来说依旧很难。组长对我及时的激励和帮助，让我坚持了下去。在不断地学习中，看到了自己和别人的差距，但同时又令我充满信心继续学习，在和大家的配合中，我们相互配合，相互指导，相互学习，我明白了团队合作的力量，也学会了在与他人合作的同时不断学习来提升自我，这样的经历对以后的学习生活也会有莫大的帮助。

6 参考文献资料

- 《编译原理（第 2 版）》
- 百度百科：逆波兰式

https://baike.baidu.com/item/%E9%80%86%E6%B3%A2%E5%85%B0%E5%BC%8F/128437?fr=aladdin

小型算法流程图绘制工具

FlowCharter

用户使用手册

请您务必首先阅读本手册后再进行使用

环境说明

FlowCharter 的一切开发、测试均在 Windows 10 环境下使用 Dev-C++ 进行。一些功能可能在其他环境下不能正常运行。请您务必在上述环境下进行使用。

功能概述

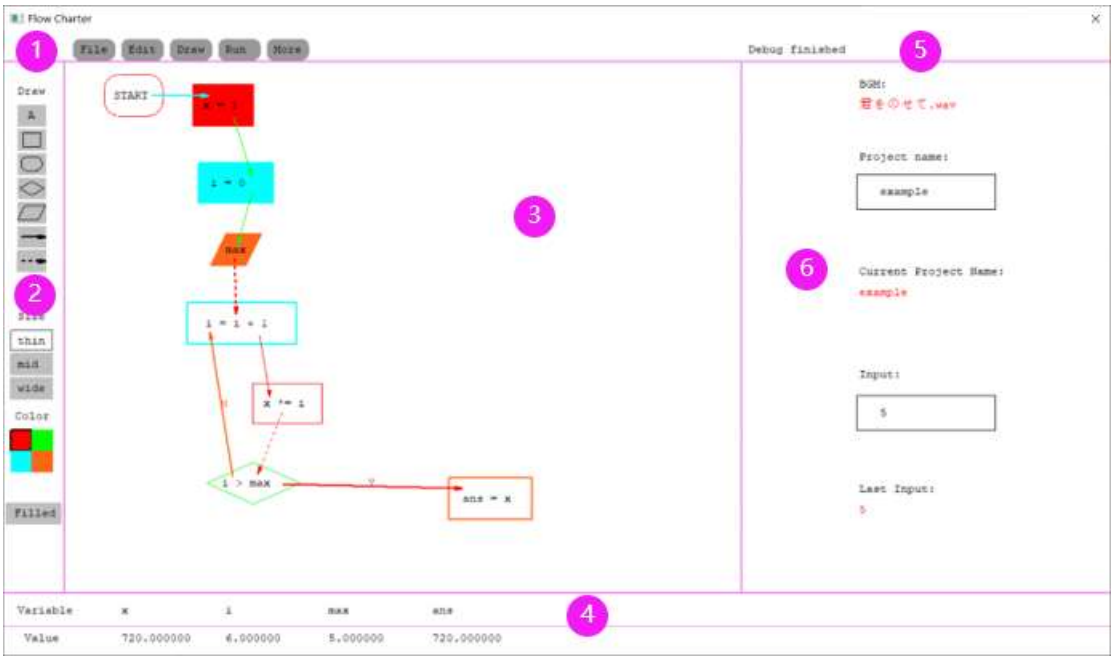
FlowCharter 提供以下主要功能：

1. **文件操作**，包括流程图文件的新建、打开、保存、关闭。
2. **图形的编辑**，包括大小、位置的改变以及图形的复制、粘贴和删除。
3. **绘图**，包括长方形（流程图中的赋值节点）、圆角矩形（流程图中的开始节点）、菱形（流程图中的判断节点）、平行四边形（流程图中的输入节点），以及实线箭头、虚线箭头（表示节点之间的联系关系，这两种箭头没有功能上的差异）。
4. **图形格式设计**，包括线宽（粗、中、细），颜色和是否填充，同时可以在形状中填入文字。
5. **模拟执行过程**，包括流程图的单步调试和瞬间出结果的那种运行。
6. **BGM 支持**，包括随机更换曲库中的 BGM 的功能。
7. **上述功能的按钮和快捷键实现**。

这些功能将在后文依次介绍。

界面简介

当您打开程序时，向您展示的会是求阶乘的流程图。



1 菜单栏，对应文件操作、图形编辑、图形绘制、流程图运行和更多五个功能。点按其中任一按钮可以展开二级下拉菜单。

2 侧边栏，提供绘图、线宽选择、颜色和填充的快捷按钮。

3 绘图区，用户可以在绘图区域中进行绘图。

4 变量显示区，在调试或运行过程中，此区域实时显示当前用到的变量名和值。

5 状态信息栏，此区域实时显示程序的运行状态。

6 运行状态区，此区域展示正在播放的曲目，提供输入功能和显示当前文件名与上次输入的功能。

文件功能

按下菜单栏的“File”按钮可以进行文件功能的操作。这里展示了支持的功能和对应的快捷键。

“New”功能允许用户新建一个流程图文件。用户使用此功能时，当前的流程图会被**保存并关闭**，用户需要在右侧输入框中输入新建的流程图工程名称，按 OK 按钮后开始绘图。

“Open”功能允许用户打开一个已有的流程图工程文件。实际上，程序运行之初就会打开 example 工程。用户使用此功能时，当前流程图会被**保存并关闭**。同样的，用户需要在右侧输入框输入流程图工程名称，按 OK 按钮后程序打开对应工程。如果不存在相应工程，状态信息栏提示打开失败。

“Save”功能允许用户将当前流程图保存入当前工程的文件中。当前工程的名称可以在运行状态区看到。如果工程名为空，那么工程文件会在本次程序运行中暂时保存，但在下一次运行 FlowCharter 时会被清空。

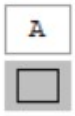
“Close”功能允许用户将当前流程图工程**不保存**并关闭。

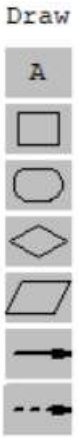
“Exit”功能允许用户**不保存**当前工程并退出程序。

编辑功能

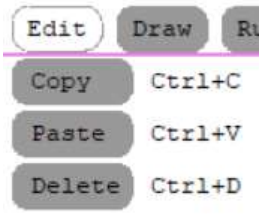
编辑功能分为两个部分：大小、位置的调整 和 图形的复制、粘贴和删除。

大小、位置的调整在未绘图状态下进行，即“Draw”侧边栏的按钮均处于未选中状态。此时，用户用鼠标左键点击并拖动实现图形位置的调整，鼠标右键点击并拖动实现图形大小的调整。





选中（上）与未选中（下）。当 Draw 的 7 个按钮都未选中时是位置调整状态。



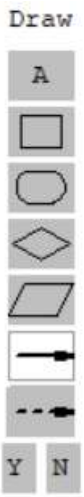
按下菜单栏的“Edit”按钮可以进行图形的复制、粘贴与删除。这里也展示了对应的快捷键按钮。

当用户选中一个长方形、圆角矩形、菱形或平行四边形时，运行状态区会出现一个 Now 箭头来为用户指示当前图形。

绘图功能

绘图功能可以在菜单栏的“Draw”实现，也可以在侧边栏实现。

侧边栏分为 Draw，Size，Color，Filled 四个功能部分。



Draw 部分允许用户选择需要绘制状态。用户选中需要绘制的图形后，可以在绘图区按下鼠标左键并拖动实现图形的绘制。

“A”按钮允许用户在长方形、圆角矩形、菱形和平行四边形中输入文字，即流程图的内容。

特别地，当用户选中的是两种箭头之一时，下方会出现 Y 和 N 按钮，来为流程图的判断节点提供分支的运行方向，Y 和 N 分别指向判断结果为真和假时运行的下一个节点。当已经选中 Y 和 N 之一时，再次点击对应按钮会取消选中，之后建立的箭头上面没有文字标志。

当 Draw 部分选中了某种绘制状态时，再次点击对应按钮将取消选择，回到大小、位置调整的功能状态。

请注意：不要在一个图形中引出多于需要的箭头，否则程序可能不会如您的期望那样运行。另外，箭头的结束点必修严格包含在对应的图形中，否则程序将无法正确识别，可能会提前退出运行。

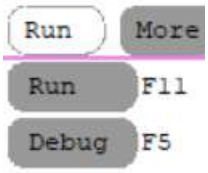
Size 部分允许用户选择绘图线条的粗细，分为 thin，mid，wide 三档。

Color 部分允许用户选择绘图线条的颜色。

Filled 按钮允许用户选择绘制的长方形、圆角矩形、菱形和平行四边形是否会被填充。当按钮被选中时，绘制的图形是填充的；再次点击此按钮将取消选中，此时绘制的图形是不填充的。

运行功能

运行功能可以在菜单栏的“Run”实现。这里同样展示了对应的快捷键。



运行（run）和调试（debug）的区别是，调试将一步一步进行，每一步需要用户按下按钮，直到程序结束。而运行则是直接运行，直到需要用户输入或程序结束。在调试过程中，用户可以随时选择 Run 从而直接运行。

执行运行或调试时，程序会首先寻找表示“开始”的圆角矩形。如果没有找到，状态信息栏会报错："Error: no 'Start' rounded rectangle"。圆角矩形的内容不会给寻找圆角矩形带来影响。

开始运行后，程序按照箭头顺序依次运行，遇到错误、没有箭头指出或箭头没有指向图形时程序结束。

程序运行过程中，如果运行到表示“运算”的长方形时，程序会识别、分析其中的语句并对变量进行值的更改。相关语法要求将在后文介绍。如果语句不合法，状态信息栏会报错："Error: Invalid expression"。

如果运行到表示“开始”的圆角矩形，这说明程序不止一次运行到“开始”，这是不合法的。因此状态信息栏会报错："Error: multi 'Start' nodes"。

如果运行到表示“判断”的菱形，程序会识别、分析其中的语句并给出返回值。程序按照返回值的真假选择 N、Y 箭头进行运行。没有对应的箭头会导致程序认为运行已经正常结束。如果语句不合法，状态信息栏会报错："Error: Invalid expression"。

如果运行到表示“输入”的平行四边形，程序会暂停运行，并提示："Waiting for input"。用户此时需要在右下方的文本框中输入平行四边形中变量的值，按回车或 OK 按钮结束输入。输入完成后，如果合法，程序继续运行；如果不合法，状态信息栏会报错："Error: Invalid input"。

程序运行的每一个步骤中，最下方的变量显示区将实时展示各个变量的值。程序中所有变量都用 double 类型存储，但是可以使用取模运算，程序会将其强制转为整型后进行运算（尽管如此，我们不推荐您使用这样的操作）。之前没有用到的变量将被初始化为 0。

如果您在使用过程中遇到变量状态与期望不同时，请尝试对照语法要求进行检查。如果程序在您期望之前的地方提前正常退出（Debug finished 或 Run finished），请检查您的箭头是否指到了对应的形状之内。

BGM 与帮助

在“More”菜单中有两项功能：帮助和更改 BGM。更改 BGM 的快捷键是 F1。帮助会打开“help.pdf”（即本文件），更改 BGM 会从曲库（7 首）中随机挑选一首进行循环播放。

语法要求

在流程图中，像编程语言一样，我们对语法采取了一些规定：

1．我们基本沿用了 C 语言的赋值和判断语句格式。

2．变量名只允许包含大小写字母，且不超过 8 个字符。

3．我们支持的计算运算符包括 + - * / % ( )，支持的赋值运算符包括 = += -= *= /= %=，支持的比较运算符包括 == >= <= > < !=。

请注意：自增、自减运算符 ++ -- 以及逻辑与、或、非 && || ! 并不被支持。

4．请注意语句应用的形状。在判断节点中的赋值语句、运算节点中的比较语句等都会被判定为语法错误。

5．表示"输入"的平行四边形节点中只允许填入一个变量名。

6．运算和判断节点中只允许填入一个语句。