**高级语言程序设计**

**课设报告**

**题 目\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**学 号\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **报告内容** | **报告结构** | **报告图表** | **界面** | **最终成绩** |
| **□丰富正确**  **□正确**  **□基本符合要求**  **□过于简单** | **□完全符合要求**  **□基本符合要求**  **□有欠缺** | **□符合规范**  **□基本符合规范**  **□有一些错误** | **□丰富、炫酷**  **□有背景图片**  **□有背景色**  **□简单**  **□开始界面 □游戏界面** |  |
| **程序功能实现** | **程序执行情况** | **问题回答情况** | **总体评价** | |
| **□多个扩展功能**  **□少量扩展功能**  **□完成基本功能**  **□未完成基本功能** | **□流畅**  **□界面有闪动**  **□操作不灵活**  **□有错误** | **□清楚、正确**  **□基本正确**  **□回答有部分错误**  **□不能回答问题** |  | |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 1 需求分析

这次课设我们需要制作一个大鱼吃小鱼的游戏。主要要实现的功能有要实现小鱼在碰到高级的大鱼的时候要扣分，而在碰到比自己级数略低的小鱼时，要做出相应的奖励。在这个过程中，要对玩家的游戏成绩进行记录。这个程序应该有自己的菜单，以及游戏内容等等。

## 1.1 功能

游戏功能主要分为以下内容：

1. **游戏主菜单：**
   1. **开始游戏**

这里是玩家开始游戏的地方。

* 1. **继续游戏**

这里玩家可以恢复保存的游戏

* 1. **游戏选项**

这里玩家可以设定游戏的设定

* 1. **关于**

这里玩家可以看到关于作者与开发过程的内容

* 1. **退出**

这里玩家可以退出游戏

1. **游戏内容**
   1. **中断游戏**

玩家中断游戏，返回至主菜单

* 1. **渲染**

渲染游戏内容，我们采取了帧为单位的渲染方式

* 1. **判定**

判定玩家位置信息，并作出逻辑判断

* 1. **存储操作**

存储游戏相关数据结构（例如鱼的链表，以及鱼素材的纹理）

1. **游戏资料存储**
   1. **游戏内容存储**

将中断游戏时的游戏内容存储至文件，并可以从文件中恢复游戏内容。方便玩家进行游戏。

* 1. **游戏设置存储**

将游戏设置存至文件，包括分辨率设置（需重启游戏）以及难度设置。

### 1.1.1 基本功能

1、可以以链表形式存储小鱼

2、可以判定玩家鱼的位置与每只在链表中的小鱼的关系

3、可以记录玩家的得分情况

4、小鱼可以以多种运动方式在游戏中移动

### 1.1.2 扩展功能

1、加入了更加美观的主菜单以及主菜单背景动画

2、加入了游戏设置与游戏内容的存储功能

3、加入了继续游戏的功能

5、加入了小鱼游动会避开玩家，大鱼游动会朝向玩家的设定

## 1.2数据

typedef struct game

{

// mode 0 for rand position based

// mode 1 for rand speed based

// other is for rand speed without y

int mode = 2;

int speed\_ratio = 1;

wchar\_t \*player; // Player's name

int time\_begin, time\_sec; // Timer for count Game time

int score = 0; // Score

bool god; // GOD Flag

float level; // Player's level

MOUSEMSG mouse; // Mouse Message

IMAGE npc\_fishes[RES\_FISHES]; // NPC Texture Array

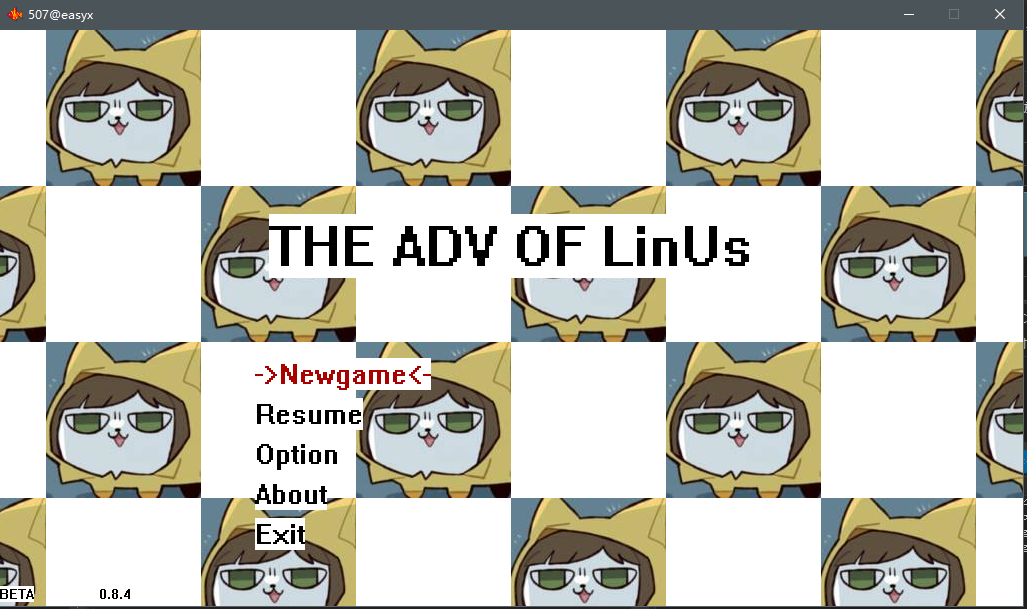
IMAGE player\_fish; // Player's fish texture

IMAGE background; // Background Texture

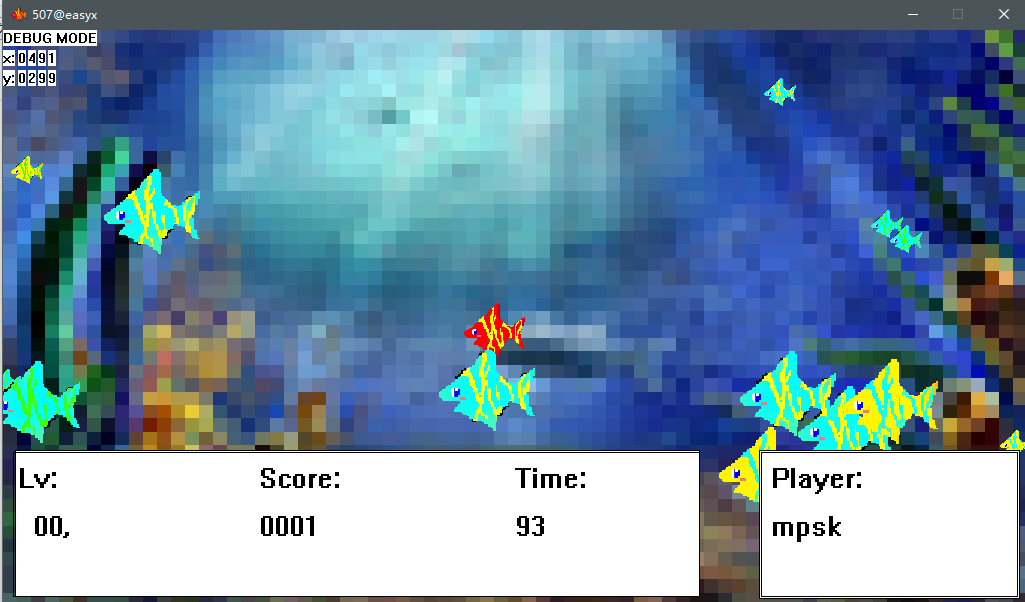
FISH \*fish = NULL; // Head of NPC chain list

}GAME;

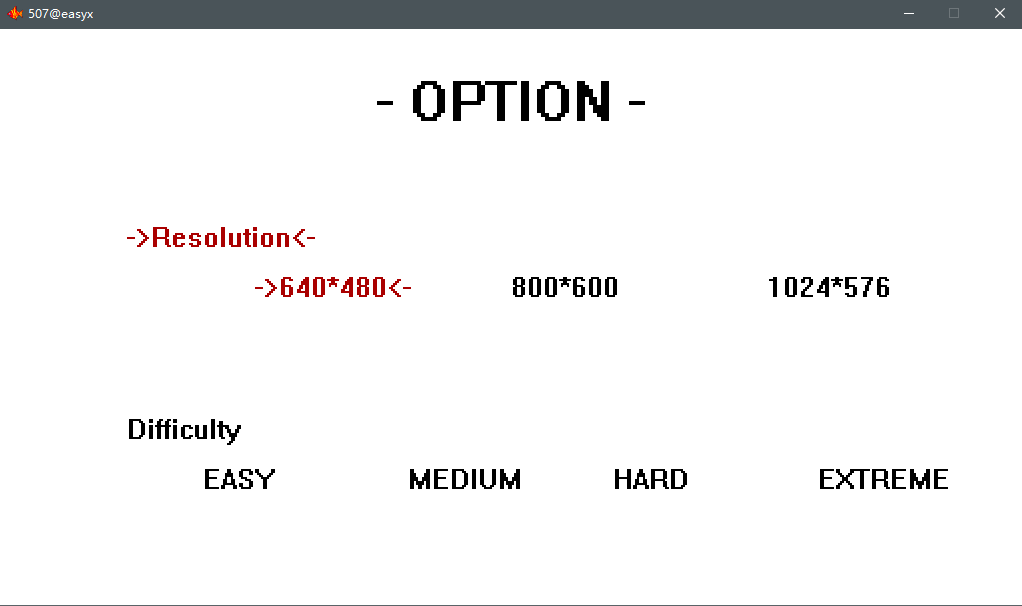
## 1.3界面



游戏主菜单



游戏内容



游戏设置界面

## 1.4 开发与运行环境需求

Visual Studio 2015 with easyX

# 2 总体设计

在需求分析的基础上，采用自顶向下的方法进行模块化的程序设计，合理划分模块。在此部分应给出程序总体的设计方案，划分了几个模块。

## 2.1 主要数据结构

游戏结构体GAME，包含游戏内的所有参数，以及进行内容操作的资源。

typedef struct game

{

    //  mode 0 for rand position based

    //  mode 1 for rand speed based

    //  other is for rand speed without y

    int         mode = 2;

    int         speed\_ratio = 1;

    wchar\_t     \*player;                    //  Player's name

    int         time\_begin, time\_sec;       //  Timer for count Game time

    int         score = 0;                  //  Score

    bool        god;                        //  GOD Flag

    float       level;                      //  Player's level

    MOUSEMSG    mouse;                      //  Mouse Message

    IMAGE       npc\_fishes[RES\_FISHES];     //  NPC Texture Array

    IMAGE       player\_fish;                //  Player's fish texture

    IMAGE       background;                 //  Background Texture

    FISH        \*fish = NULL;               //  Head of NPC chain list

}GAME;

鱼节点结构：用来存储鱼类特征的结构体，鱼的存储方式是线性链表

typedef struct fish

{

    float x;                //  Node fish's position in X

    float y;                //  Node fish's position in Y

    float s\_x;              //  Node fish's speed in X

    float s\_y;              //  Node fish's speed in Y

    float level;            //  Node fish's level(judge score or hurt)

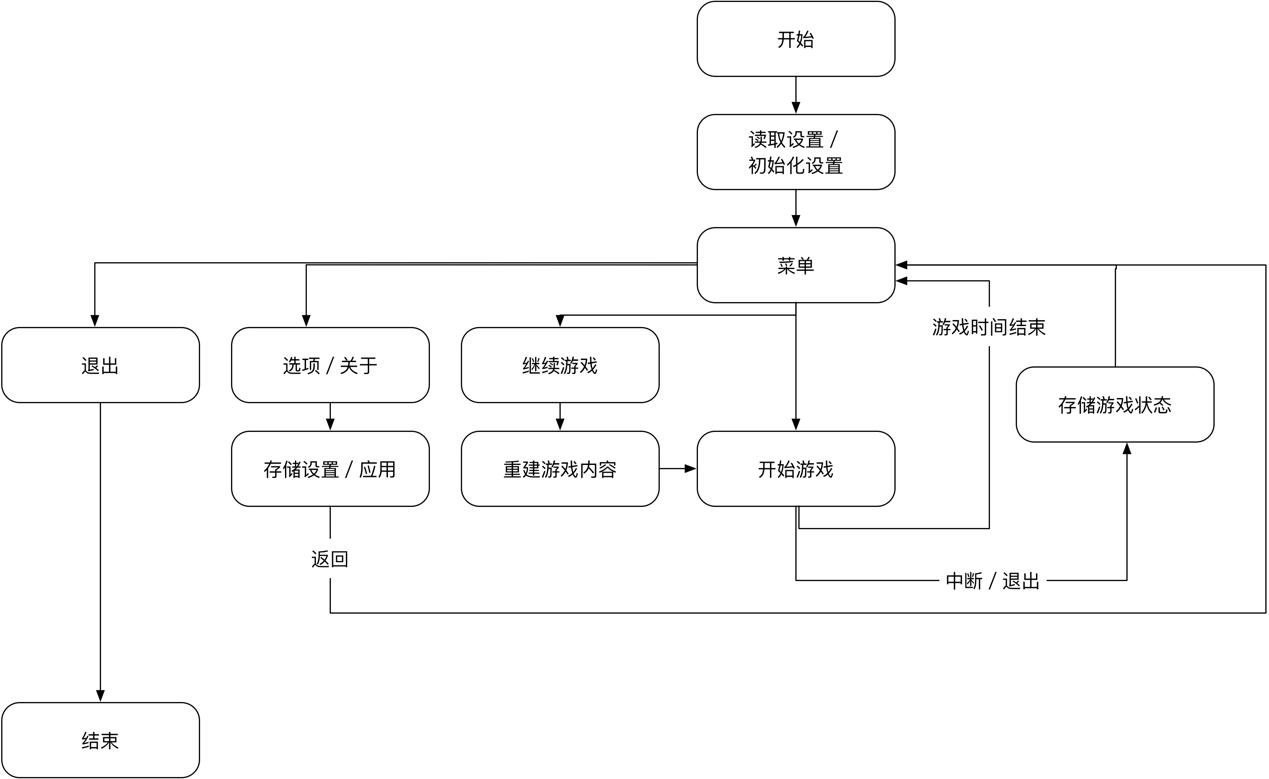
    int res\_num;            //  Alternate texture amount

    int flag;               //  From Left or Right

    fish \*next;             //  Pointer to next

}FISH;

## 2.2 程序模块设计



### 2.2.1 子模块设计

我接下来的所有程序设计都是围绕着操作游戏结构体来展开的。实际上这些模块都是逻辑上的，仅仅体现在函数命名上。

1. 开始游戏

这个模块主要是创建一个新的游戏结构体并维护。使其完成对游戏内容的更新与渲染。此模块中主要的内容就是维护一个游戏结构体。

1. 继续游戏

这个模块主要是从文件中恢复之前存储的游戏内容，相当于从文件读取游戏内容并重新组织一个游戏结构体。

1. 选项

这个模块用于维护从文件中读取或向文件中写入设置，并将设置传入主程序。

1. 关于
2. 退出

1、鱼的管理

本模块用于和鱼相关的操作。

相关函数：

int fish\_add(GAME &game, int num, int SCREEN\_W, int SCREEN\_H);

功能：向鱼链表中添加一条鱼

参数：游戏结构体，添加数量，屏幕宽，屏幕高

返回值 整型，表示添加状态（成功/失败）

int fish\_clear(GAME &game)

功能：清空鱼链表

参数：游戏结构体

返回值：整型，表示删除状态

int fish\_init(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：初始鱼链表，创建链表的头指针

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示删除状态

int fish\_judge(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：判定鱼的位置是否与玩家相交

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示判断状态（成功/失败）

FISH \*fish\_rm(GAME &game, FISH \*fish)

功能：删除特定的鱼

参数：游戏结构体，屏幕宽，屏幕高

返回值：指向鱼类型的指针（被删除的鱼的下一条鱼）

int fish\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：鱼类显示的单周期函数，包括鱼类运动方式

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

2、游戏内容

int game\_background\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：显示游戏世界背景

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_loop(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：游戏世界渲染循环

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_main(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：游戏内容主函数

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

返回值决定了游戏走向（返回值代表退出游戏类型）

int game\_player\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：显示玩家对象

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_score(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：显示最终游戏结果

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_status\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：游戏状态栏显示

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

3、接待画面

int name\_loop(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：接待画面渲染循环

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

4、选项画面

int option\_loop(OPTION option, int SCREEN\_W, int SCREEN\_H)

功能：选项画面渲染循环

参数：选项结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

5、菜单画面

int menu\_loop(MENU &menu, int SCREEN\_W, int SCREEN\_H)

功能：菜单画面渲染循环

参数：菜单结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

void menu\_show\_version(int SCREEN\_W, int SCREEN\_H)

功能：菜单页面显示版本

参数：屏幕宽，屏幕高

返回值：无

6、文件操作

int load\_setting(SETTING &setting)

功能：从文件载入设置

参数：设置结构体

返回值：整型，表示单步执行状态（成功\失败）

int write\_setting(SETTING setting)

功能：向文件写入设置

参数：设置结构体

返回值：整型，表示单步执行状态（成功\失败）

int init\_setting(SETTING &setting)

功能：生成初始设置

参数：设置结构体

返回值：整型，表示单步执行状态（成功\失败）

7、关于画面

void about\_loop(ABOUT about, int SCREEN\_W, int SCREEN\_H);

功能：关于画面渲染循环

参数：关于结构体

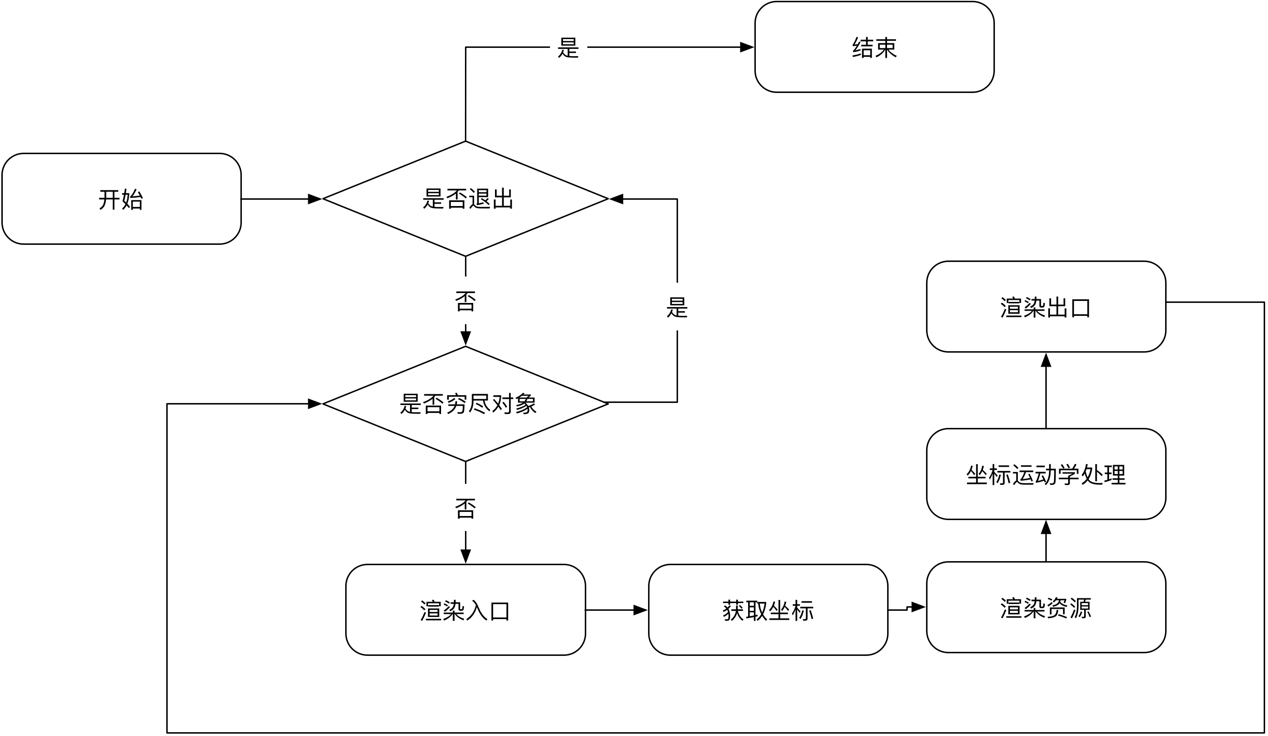
返回值：无

# 3 关键技术的实现

在此部分给出核心模块和关键模块的设计方案和实现细节。用流程图和局部的数据结构说明。注意函数的命名应与概要设计部分对应。写出自己有特色的地方

必须细分小节，说明每个关键设计。例如动画的实现等。

## 3.1 动画的实现



## 3.2 鱼的运动方式

我们采用了抽象鱼的速度与鱼的位置，方便我们的坐标的运算。

根据公式△*r* =*r*（*t*+△t）－*r*（*t*），我们就可以用一个常量来表示一个固定的抽象加速度，然后根据随机生成的x，y速度，计算出对应的鱼的位置。

根据公式v = ∆S／∆t，我们可以将∆t简化为1（tik tok），那么得到的了公式∆S = v，这样我们简易的运动学模拟就完成了。

鱼的运动系数主要是通过速度描述的，而这个速度还会乘上一个时间系数SPEED\_RATIO（宏），相当于∆S = v \*（∆t \* speed\_ratio）。因为不直接操作位置，我们就能获得更佳平滑的NPC运动效果，也会显得更佳自然。

## 3.3 鱼的

# 4 测试

在此部分报告应给出你所设计的用于整个程序的测试用例和测试结果。测试用例的设计应包括：正确运行程序的用例、导致程序运行错误的用例、边界数据的用例等。

如果需要可以再细分小节。

# 5 总结提高

## 5.1 课程设计总结

同学依据自己的经历写出个性化的总结。内容可以包括：程序开发中的体会与收获，开发中遇到的问题与解决情况，自己对自己完成课设情况的评价等等。

## 5.2 对本课程意见与建议

我们共同完成了一个学期的学习，其中有辛酸，也有收获，有感动，也有遗憾，请同学们针对《高级语言程序设计Ⅲ》与《高级语言程序设计课程设计》这两门课程谈谈你的体会，总结优点，给出不足，为我们提高教学质量做出你的贡献。谢谢同学们！

## 5.3 附件：程序源代码