

# C++程序设计课程设计

## 实习报告

班 级: 10041612

姓 名: 白云鹏

学 号: 1004161221

指导教师: 夏军宝

提交日期: 2018/01/18

# 目 录

1	系统概述.....	1
1.1	系统概述.....	1
1.2	需求分析.....	1
2	系统设计与关键技术.....	3
2.1	系统总体结构设计.....	3
2.2	关键技术概述.....	4
3	系统实现.....	6
3.1	系统 1 实现.....	6
3.2	系统 2 的实现.....	8
4	系统测试.....	9
4.1	系统 1 测试.....	12
4.2	系统 2 测试.....	13
5	总结与感想.....	16

# 1 系统概述

## 1.1 系统概述

概述所完成实践项目的基本情况。控制台程序和图形界面程序可分别描述。

### 1. 控制台程序

所做的控制台程序是实现了一个简单的课程管理系统。由于时间有限，只将课程分为了选修课以及必修课两类。其两类都由一个基类所派生而得到，在完成情况上，控制台程序所列出的一系列操作都得到了实现，也经过了测试。

### 2. 图形界面程序

所选的图形界面程序实现了一个简单的迷宫小游戏，从生成地图到寻找到出口的路径，在普通的迷宫上，增加了小怪物的实现，小怪物可以随机的向一个可行的方向移动，如果操作角色与其相遇，将导致游戏结束。除此之外，还实现了从任意一点寻找并展示到出口的路径的功能。

## 1.2 需求分析

概述所开发项目的需求，介绍系统的主要功能及逻辑关系。

### 1. 控制台程序

```
----- Welcome to the course manager system -----
----- 1)Add course      2)Delete course   -----
----- 3)Show one course  4)Show all course -----
----- 5)update course    6)date show       -----
----- 7)Build show       8)Exit           -----
```

在课程管理系统中，首当其冲的功能必然是向其中增加课程。所添加的课程分为了两类：必修课以及选修课。如果连课程都无法添加，那么这个管理系统也是没有任何作用的。添加的课程内容根据课程类别也有所不同。如果是必修课，一般必修课的上课学生都属于一个院系，所以在必修课所需的内容有级别，院系，参与学生，由于必修课有可能是公共课，那么参加的院系也就有可能不只是一个

院系，至于学生而言则可能性更多，因此，在储存它们的时候，采用了无重复的集合 Set 来存放，就避免了重复元素。在有了相应的元素之后，可能就需要查看对应的信息，为此，就需要提供有查询的接口，在实现中提供了单门课程查询以及显示所有课程信息的接口。除此之外，如果某一门课由于某种原因而停课，这时就需要有删除的操作。由于删除的操作并不多见，所以只提供了单条删除的接口。其次，不同的课程，时间和所占的地点也应该有所不同，由此提供了显示时间，地点的功能，用来剔除某些时间地点都冲突的课程。最后，即使是添加进入系统的课程，也会有所变化，由此又提供了更改课程信息的接口。课程的信息在程序运行和结束的时候读取以及保存到文件之中。

## 2. 图形界面程序

迷宫程序，思路比较简单，在生成一副比较好的地图之后选中起始位置和终止位置。用户经过一系列的操作可以从起始位置到达终止位置，但是这样游戏可能会比较枯燥，为了增加难度，在地图上增加了小怪物机制，遇到小怪物后，游戏就终止。同时，如果每次完成一关以后，下一关的阶数将比上一关增加 10 级，最高不超过 120 级，在级数过大后，元素的分辨率就不会很高，由此导致后期游戏体验感不如前期优秀，所以要限制最高级数。同时，由于后期难度的增加，用户可能不能解出谜题，所以增加了自动寻路功能，用算法来解决问题，最后再将路径反馈给用户。由此，迷宫的关键问题就在于如何生成一副比较满意的地图。

## 2 系统设计与关键技术

### 2.1 系统总体结构设计

介绍所开发系统的总体设计，包括

- 功能结构与主要处理流程；
- 核心数据结构及磁盘存储结构的设计；
- 类设计及类之间关系。

#### 1. 控制台程序

类似于课上所学的购物车管理系统，首先是向管理系统中增加课程或从本地文件读取所需内容，在读取之后才能进行操作，如果恢复成功，那么此时程序中就有了文件中所存储的信息，否则程序的数据为空，此时只能手动添加。添加完成之后，即可对所需信息进行操作。

##### a. 增加课程

首先选择增加课程的类别，也就是必修课或者选修课，在选择类别完成之后，新创建一个相应的对象并调用其 `input()` 函数进行对自身信息的完善。

##### b. 删除课程

由于人们对数字没有像文字一样敏感，所以课程的序号也并不是经常会用到，而课程用不应有相同名字的课程，因此在删除课程中就有了两种方式，通过课程序号或者课程的名字来删除。选择删除方式之后，会调用相对应的 `findcourse` 函数来得到要删除的迭代器，接着就是从课程列表中剔除和该迭代器。

##### c. 查询单个课程信息

同样的，根据要查询的方式（课程序号或者课程名）调用相应的查找函数得到迭代器，接下来调用基类中定义的虚函数 `showmess()` 来显示课程信息。

##### d. 显示所有课程信息

循环遍历列表，调用 `showmess()` 函数。

##### e. 更新课程信息

查找对应的课程之后，由于不同的课程类有着不同的数据成员，因此并没有直接在管理类中进行更新而是调用各自的 update() 虚函数来更新。

#### f. 根据时间地点查询

循环遍历列表查找相应的时间地点，然后进行对应输出。

在这个程序中一共有四个类，`basecourse` 作为基类，派生出 `opinioncourse` 类和 `compulsorycourse` 类，由于两种课程的相似度比较高，因此基类中的公有函数也比较多，两种课程不同的地方只有参加院系，因此两类之间的辨识度并不是很高，函数的不同地方多在于信息的恢复及储存，读取和输出，更新课程信息上。而最后的课程管理类是最重要的部分，它包含有一个基类指针的 `vector` 用来储存课程类的全部信息并对其进行管理。

### 2. 图形界面程序

首先在开始界面讲解规则，游戏开始后生成地图以及各点到终点的路径，根据玩家的操作让游戏角色移动并且使地图上已存在的小怪物随机移动，如果玩家到达终点，释放地图并重新生成新地图。类似于课上的程序，这个程序包含有 `mainwindow`, `QWidget` 类，除此之外还有怪物类，角色类，这些共同作用在地图类上，地图类可谓是这个程序最重要的成员了，一切的移动，判断都要与地图类进行交互。

## 2.2 关键技术概述

介绍实现系统的关键技术。

### 1. 控制台程序

控制台程序中，考虑到课程管理系统的函数有很多相同的函数类型，所以在 `main` 函数中建立了一个 `route` 表来进行转发，每次从 `menu` 菜单函数中得到用户要选择的功能后，直接匹配并调用对应的函数，由此省略了 `switch` 或者 `if` 的条件判断。在类的读入输出中，借鉴了重载流运算符的思想，产生了 `recover()`, `save()` 等函数，即可用来将类的信息保存到文件中，又能进行控制台的输出。两类公有的部分被定义在基类之中

### 2. 图形界面程序

在迷宫类之中，最重要也是最难的就是迷宫的生成以及绘画。在迷宫的

生成上，采用了随机化的 prim 算法，首先随机挑选一个点作为起点，遍历与其相邻的顶点并将符合条件的点加入至一个临时数组中，从临时数组中随机挑选一个点将其加入到堆栈之中，并且在此时将这个点与起点之间的墙壁消除，接下去从堆栈中随机挑选一点作为起点，不断重复直到所有的点都被访问。在绘画墙壁时，由于一个点的周围至多有四面墙，所以遍历所有的四面墙判断这面墙是否被消除过，如果没有被消除，就需要调用 Qt 的 drawLine 函数进行绘画。同时，在移动角色或者怪物的时候也需要对当前格子四周的墙壁进行判断，从而判断是否可以移动。在迷宫寻路的过程中，则是采用了广度优先搜索，将终点作为起点不断进行搜索，并将其前一个顶点加入到 pathTo 数组之中，这样就得到了任意一点到终点的路径。还有比较难实现的是怪物累的平滑移动，因为整个地图是采用的二维数组进行存储而 Qt 在更新时则是按照像素点进行绘画，由此就需要将二维数组中的坐标转换为相应的像素及对应位置。还有通过设置刷新时间，实现了一些简单的动画效果，比如在迷宫中的小怪物的走动，欢迎界面的操作说明等。

### 3 系统实现

#### 3.1 系统 1 实现

介绍程序核心类的实现，包括代码与说明。

1. main 函数中函数表的实现：

```
void (coursemanager::*type)(void) = nullptr;
using pfunc = decltype(&coursemanager::addcourse);
map<char, pfunc> route;
long long basecourse::nextId = 0;
void initroute() {
    route['1'] = &coursemanager::addcourse;
    route['2'] = &coursemanager::deletecourse;
    route['3'] = &coursemanager::show;
    route['4'] = &coursemanager::showall;
    route['5'] = &coursemanager::updatecourse;
    route['6'] = &coursemanager::dateshowmess;
    route['7'] = &coursemanager::locationshowmess;
}
```

首先自定义 pfunc 类型作为 coursemanager 中 addcourse 函数的类型，接下来在建立 char 类型到 pfunc 类型的关联数组，主键与函数之间的对应关系相同于 menu 函数中的对应关系。

2. 文件输出输入

```
ostream &compulsorycourse::save(ostream &out) {
    out << getInfo() << endl;
    return out;
}
```

(文件输出函数)

```

string compulsorycourse::getInfo() {
    stringstream out;
    out << getType() << "\n";
    out << getId() << " " << getRoom() << " " << getDate() << " "
        << getBeginTime() << " " << getEndTime() << " " << getGrade() << "\n";
    out << getName() << "\n";
    out << getBuild() << "\n";
    out << getMajor().size() << "\n";
    for (auto c : major) {
        out << c << "\n";
    }
    out << getMembers().size() << endl;
    for (auto c : members) {
        out << c << " ";
    }
    return out.str();
}

```

getInfo()函数表示将类的信息按照一定的格式进行编码，然后返回编码完成的字符串。

```

istream &compulsorycourse::recover(istream &in) {
    string line;
    getline(in, line);
    stringstream iin(line);
    cout << line << endl;
    long long id;
    int roo;
    string na, da, bt, et, bu;
    int gra;
    int si;
    iin >> id >> roo >> da >> bt >> et >> gra;
    setGrade(grat);
    setId(id);
    setRoom(roo);
    setDate(da);
    setBeginTime(bt);
    setEndTime(et);
    getline(in, na);
    setName(na);
    getline(in, bu);
    setBuild(bu);
    getline(in, line);
    stringstream iin2(line);
    iin2 >> si;
    while (major.size() < si) {
        string ma;
        getline(in, ma);
        addMajor(ma);
    }
    getline(in, line);
    stringstream iin3(line);
    iin3 >> si;
    while (members.size() < si) {
        getline(in, line);
        stringstream iin4(line);
        long long tem;
        iin >> tem;
        addMember(tem);
    }
    return in;
}

```

recover()函数表示从文件中读取，其格式按照 getInfo 中的编码格式进行读取。

```
void coursemanager::readfromfile(const std::string &filename) {
    ifstream in(filename);
    string line;
    if (!getline(in, line)) {
        return;
    }
    stringstream in1(line);
    int n;
    in1 >> n;
    if (!getline(in, line)) {
        return;
    }
    stringstream in2(line);
    long long cuid;
    in2 >> cuid;
    basecourse::setNextId(cuid);
    while (courses.size() < n) {
        getline(in, line);
        stringstream in3(line);
        int t;
        in3 >> t;
        if (t == 1) {
            auto c = new compulsorycourse;
            c->recover(in);
            courses.push_back(c);
        } else {
            auto c = new opinioncourse;
            c->recover(in);
            courses.push_back(c);
        }
    }
}
```

在读取文件时，首先恢复当前课程的序号，接下来读取文件中保存的课程的数量，随后首先读取课程的类别，调用对应的函数进行读取恢复。

## 3.2 系统 2 的实现

介绍程序核心类的实现，包括代码与说明。

### 1. 迷宫生成

```
void basicmap::loop(pii a) {
    // using pii=pair<int,int>;
    // a是一个随机生成的起始点
    memset(road, 0, sizeof road);
    // road数组为每一块四周的墙壁 bool road[200][200][4];
    memset(mark, 0, sizeof mark);
    // mark 数组表示是否被访问过
    vector<pii> q;
    // 起始点的候选数组
    srand(time(0));
    q.push_back(a);
```

```

while (q.size() != 0) {
    int curid = rand() % q.size();
    mark[q[curid].first][q[curid].second] = 1;
    // 首先从q数组中随机选出一个点作为这一步的起始点
    vector<pii> nexts;
    // 判断这一个点是否越界
    int xx = q[curid].first, yy = q[curid].second;
    if (xx >= size || xx < 0) {
        q.erase(q.begin() + curid);
        continue;
    }
    if (yy >= size || yy < 0) {
        q.erase(q.begin() + curid);
        continue;
    }
    // 获得与起点相邻的四个点(上,下,左,右)
    pii up1(q[curid].first, q[curid].second - 1);
    pii down1(q[curid].first, q[curid].second + 1);
    pii left1(q[curid].first - 1, q[curid].second);
    pii right1(q[curid].first + 1, q[curid].second);
    // 判断这四个点是否已经在候选数组中并且没有越界
    if (up1.second >= 0 && !mark[up1.first][up1.second]) {
        nexts.push_back(up1);
    }
    if (down1.second < size && !mark[down1.first][down1.second]) {
        nexts.push_back(down1);
    }
    if (left1.first >= 0 && !mark[left1.first][left1.second]) {
        nexts.push_back(left1);
    }
    if (right1.first < size && !mark[right1.first][right1.second]) {
        nexts.push_back(right1);
    }
    // 如果nexts数组为0, 说明这个起始点周围的所有点都被访问过了, 删掉这个点
    if (nexts.size() == 0) {
        q.erase(q.begin() + curid);
        continue;
    }
    // 随机从这个数组中挑出一个点
    int nextid = rand() % nexts.size();
    auto c = nexts[nextid];
    // 消除起始点与所选择点之间的墙壁
    if (c == up1) {
        road[q[curid].first][q[curid].second][0] = 1;
        road[c.first][c.second][1] = 1;
    } else if (c == down1) {
        road[q[curid].first][q[curid].second][1] = 1;
        road[c.first][c.second][0] = 1;
    } else if (c == left1) {
        road[q[curid].first][q[curid].second][2] = 1;
        road[c.first][c.second][3] = 1;
    } else if (c == right1) {
        road[q[curid].first][q[curid].second][3] = 1;
        road[c.first][c.second][2] = 1;
    }
    // 标记所选的点, 表示这个点已经进入了候选数组
    mark[c.first][c.second] = 1;
    // 加入候选数组
    q.push_back(c);
}
}

```

```

void basicmap::draw(QPainter *p) {
    //得到窗体的左上角的位置
    int len = min(r.width(), r.height());
    len /= size;
    QPen pen(lineColor, 3, Qt::SolidLine);
    int xmid = r.width() / 2 + r.x();
    int ymid = r.height() / 2 + r.y();
    xmid -= len * (size / 2);
    ymid -= len * (size / 2);
    p->setPen(pen);
    QBrush brush(Color(11, 132, 87));
    p->setBrush(brush);
    p->drawRect(QRect(xmid, ymid, size * len, size * len));
    QFont font("宋体", 18, QFont::Bold, true);
    p->setFont(font);
    //每一个点绘制墙壁
    //由于数组中的x, y与像素位置是相反的, 所以在绘画过程中也要进行相反操作
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            if (road[j][i][0] == 0) {
                p->drawLine(QPoint(leftUp.first + len * j, leftUp.second + len * i), QPoint(leftUp.first + len * j + len, leftUp.second + len * i));
            }
            if (road[j][i][1] == 0) {
                p->drawLine(QPoint(leftUp.first + len * j, leftUp.second + len * i + len), QPoint(leftUp.first + len * j + len, leftUp.second + len * i + len));
            }
            if (road[j][i][2] == 0) {
                p->drawLine(QPoint(leftUp.first + len * j, leftUp.second + len * i), QPoint(leftUp.first + len * j, leftUp.second + len * i + len));
            }
            if (road[j][i][3] == 0) {
                p->drawLine(QPoint(leftUp.first + len * j + len, leftUp.second + len * i), QPoint(leftUp.first + len * j + len, leftUp.second + len * i + len));
            }
        }
    }
}

```

## 2. 绘画墙壁

```

void basicmap::showDistance() {
    //定义一个局部结构体用来保存相应的信息
    struct ppi {
        int x, y, len;
        ppi(int a, int b, int c)
        : x(a)
        , y(b)
        , len(c) {}
    };
    //广度优先搜索标记数组
    bool mmark[200][200];
    //清空
    memset(mmark, 0, sizeof mmark);
    //搜索队列
    deque<ppi> q;
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            distance[i][j] = 32765;
        }
    }
    //终点进队
    ppi endnode(getSize() - 1, getSize() - 1, 0);
    q.push_back(endnode);
    distance[endnode.x][endnode.y] = 0;
    pathTo[endnode.x][endnode.y] = pii(endnode.x, endnode.y);
    //开始循环
}

```

## 2. 自动寻路

```
//开始循环
while (!q.empty()) {
    auto c = q.front();
    q.pop_front();
    if (mmark[c.x][c.y] == 1) continue;
    mmark[c.x][c.y] = 1;
    //如果该点的上方没有墙壁,上方点进队,并且设置path数组
    if (road[c.x][c.y][0] == 1) {
        if (mmark[c.x][c.y - 1] == 0) {
            q.push_back(ppi(c.x, c.y - 1, c.len + 1));
            distance[c.x][c.y - 1] = min(c.len + 1, distance[c.x][c.y - 1]);
            patho[c.x][c.y - 1] = pii(c.x, c.y);
        }
    }
    //如果该点下方没有墙壁,下方点进队,并且设置path数组
    if (road[c.x][c.y][1] == 1) {
        if (mmark[c.x][c.y + 1] == 0) {
            q.push_back(ppi(c.x, c.y + 1, c.len + 1));
            distance[c.x][c.y + 1] = min(c.len + 1, distance[c.x][c.y + 1]);
            patho[c.x][c.y + 1] = pii(c.x, c.y);
        }
    }
    //如果左方没有墙壁,左方点进队,并且设置path数组
    if (road[c.x][c.y][2] == 1) {
        if (mmark[c.x - 1][c.y] == 0) {
            q.push_back(ppi(c.x - 1, c.y, c.len + 1));
            distance[c.x - 1][c.y] = min(c.len + 1, distance[c.x - 1][c.y]);
            patho[c.x - 1][c.y] = pii(c.x, c.y);
        }
    }
    //如果右方点没有墙壁,右方点进队,并且设置path数组
    if (road[c.x][c.y][3] == 1) {
        if (mmark[c.x + 1][c.y] == 0) {
            q.push_back(ppi(c.x + 1, c.y, c.len + 1));
            distance[c.x + 1][c.y] = min(c.len + 1, distance[c.x + 1][c.y]);
            patho[c.x + 1][c.y] = pii(c.x, c.y);
        }
    }
}
```

## 4 系统测试

### 4.1 系统 1 测试

系统主要功能的运行结果说明，包括：

- 测试环境说明
- 测试数据
- 典型功能的测试结果说明（截图）

测试环境：

系统：Mac OS Sierra 10.12.6

控制台：zsh

Qt 版本：4.5.0

测试样例：

课程名称: 1 课程地点: zong he lou 903 课程日期:2017/8/27 课程类别:1

上课年级:1016 开始结束时间 16:00-18:00 参与专业数量 1 (123)

参与成员数量: 12 成员学号: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

```
----- Welcome to the course manager system -----
----- 1)Add course      2)Delete course   -----
----- 3)Show one course  4)Show all course -----
----- 5)update course    6)date show       -----
----- 7)Build show       8)Exit           -----
Your choice:
1
What type do you want to add?
1)compulsorycourse
2)opinioncourse
1
Course name:1
Course building:zong he lou
Room 903
Course date:2017/8/27
Grade:1016
Course begin time:16:00
Course end time:18:00
Major size:1
Input the majors 123
Members size:12
Input the members:
0 1 2 3 4 5 6 7 8 9 10 11
----- Welcome to the course manager system -----
----- 1)Add course      2)Delete course   -----
----- 3)Show one course  4)Show all course -----
----- 5)update course    6)date show       -----
----- 7)Build show       8)Exit           -----
Your choice:
3
Which function do you prefer?1)By name
2)By id
2
The id:0
Course name: 1
Course id: 0
Course date: 2017/8/27
Course location: zong he lou  903
Course time: 16:00--18:00
Join major numbers: 1
Join member numbers: 12
Do you want to show the majors?(yes/no) Do you want to show the members?(yes/no)
4
Update success
----- Welcome to the course manager system -----
----- 1)Add course      2)Delete course   -----
----- 3)Show one course  4)Show all course -----
----- 5)update course    6)date show       -----
----- 7)Build show       8)Exit           -----
Your choice:
```

文件输出：

A screenshot of a Mac OS X terminal window titled "baiyun — mvim -v test.txt — vim — Vim -g -v test.txt — 80x24". The terminal displays the following text:

```
1
2 1
3 2
4 0 903 2017/8/27 16:00 18:00 1016
5 1
6 zong he lou
7 1
8 123
9 12
10 0 1 2 3 4 5 6 7 8 9 10 11
~
~
~
~
~
~
~
~
~
~
~
~
~
NORMAL ./test.txt
"test.txt" [readonly] 10L, 91C
```

The status bar at the bottom shows "unix utf-8 text 10% 1:1".

## 4.2 系统 2 测试

测试环境：

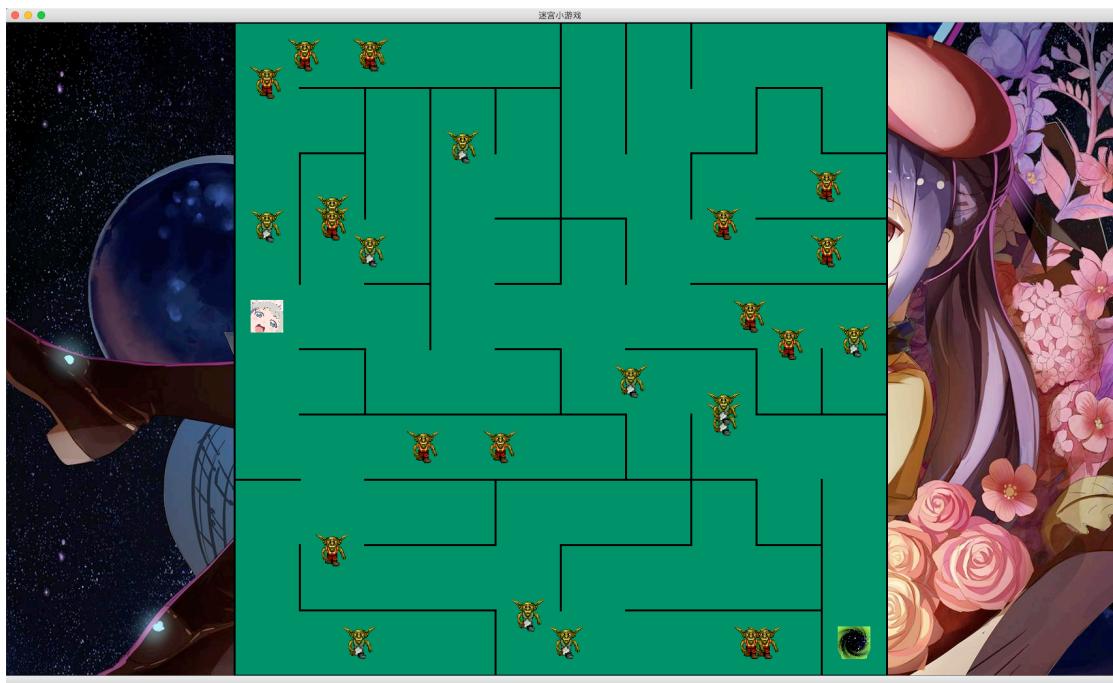
系统：Mac OS Sierra 10.12.6

Qt 版本：4.5.0

运行结果：



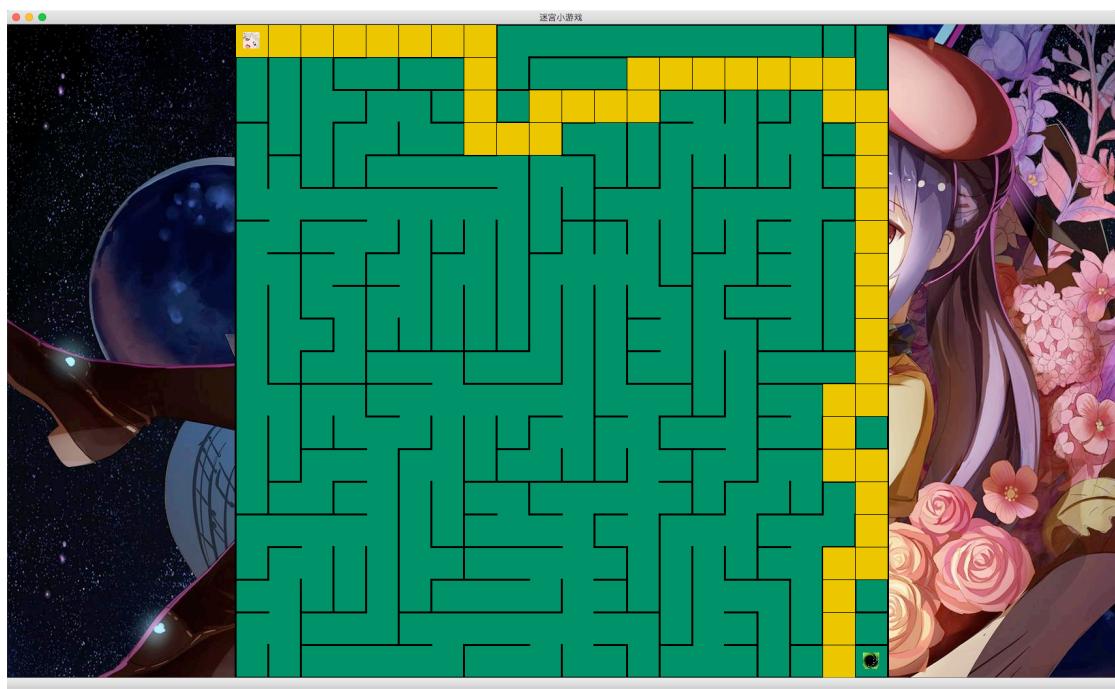
(欢迎界面)



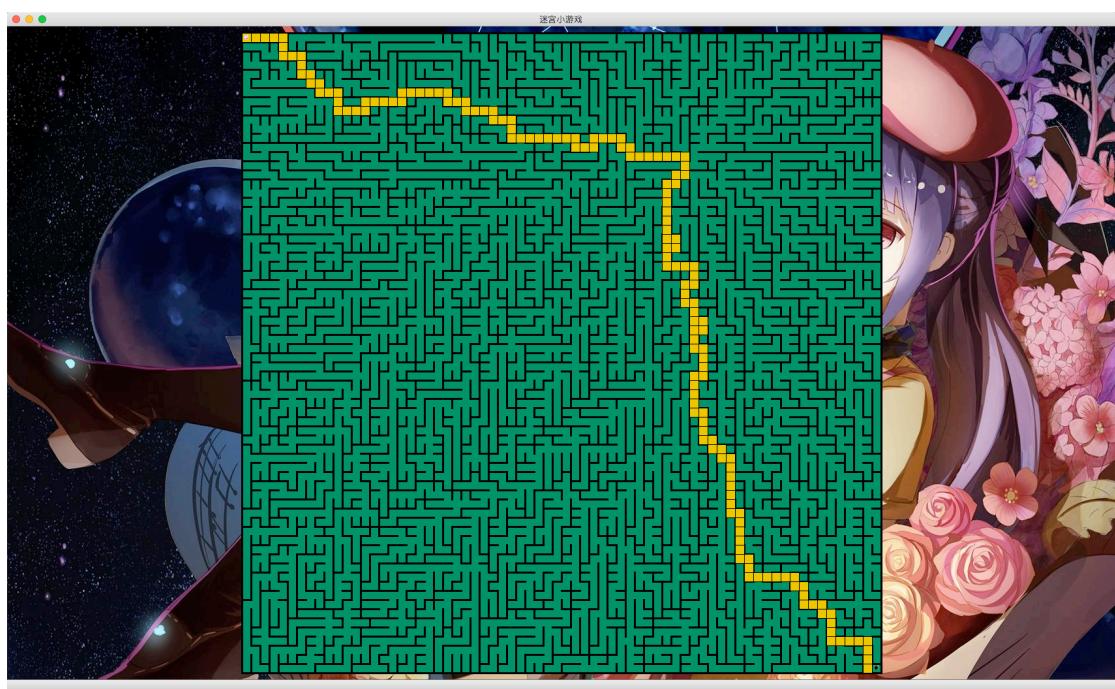
(游戏界面)



(游戏结束界面) (图片宽高比不适当造成如图效果)



(寻路界面)



(级别更高的迷宫)

无测试样例，测试样例均为程序随机生成。

## 5 总结与感想

对所开发系统的总结与展望，对 C++ 编程、课程的认识、感想与建议等。

经过一个学期的学习，自己所制作的控制台程序也确实有所不足，比如在增加课程的时候没有检验输入的日期是否合理，其次由于时间的原因没有把其他的课程类写出来，其次是由于自己对异常处理机制的不了解，导致在系统中没有能够使用到异常，系统的某些地方的安全性不是十分可靠。但是，如果自己的管理系统能够得到应用的话，我希望的是能够与学生管理系统，教务管理系统等实现对接，通过学生的学号贯穿整个系统，能够简化后端的很多操作。在编写的过程中，自己也发现了很多存在的问题，尤其是对于 C++ 的多态性，继承，模版有了更为深刻的理解。在多态方面，我理解了编译时多态与运行时的多态，通过对模版的学习才知道了标准模版库强大的理由，区别于 C 语言的面向过程，C++ 中的类的借口为我们的编程提供了更多的便利，同时也在认识事物的方面有了更加独特的见解。可以从事物的共同特征出发，总结它们相同的特征特点。在 C++ 编程方面，也正是应了老师的话：“程序员要对自己的行为负责”。确实，C++ 虽然强大，可是却也要求程序员要有一颗睿智清醒的头脑，才能保证思路的清晰，不同于 python, golang 等语言，虽然没有很多强大的第三方现成的包供程序员使用，但是所有的一切都可以由程序员自己制造出来。C++ 虽然强大，但却难以驾驭，由此也导致 C++ 课程学习有写吃力，还有在学习课程的时候，不能只是依赖于上课所讲的内容，下课也需要勤学苦练才能有所进步，而且有些事情不能只是停留在口头方面，更重要的是要亲手去做，亲手去实现，这样才能够加深对它的理解与印象。希望在以后的课程学习中能够将课程稍加分散，这样能给予更多的时间来消化所学的知识，完成相应的练习。总而言之，C++ 作为一门老牌编程语言，涉及身边各个领域，我相信在未来，它也一定会像 JAVA, python 等语言一样，不再是造轮子的语言而是用轮子的语言，希望更多的新特性能加入其中，更有意义的第三方库能够支持它，这样，它的生命力才能更加多姿多彩。