第一次课程设计

张涵之-191220154-计科

课程设计主要内容：贪吃蛇；

目标：实现经典小游戏贪吃蛇的控制台版本；

设计思路：以经典贪吃蛇为基础参考；

初始地图四面围墙，后续进入新关卡不断加入障碍物；

玩家用键盘方向键控制蛇的移动；

每吃掉1个食物，蛇的长度加一，并在地图上随机产生一个新食物；

每吃掉5个食物，附加产生一个限时食物，移动30步之后自动消失；

计分规则为每局满10分进入下一关，否则重复该局，每轮游戏结束显示分数；

每次进入下一关时，蛇的移动速度加快，障碍物数量增加；

游戏共十局，第十局满10分则整个游戏通关；

游戏地图右侧状态栏显示当前得分、本次游戏最高分和关卡信息；

枚举类型和全局变量：

两个枚举类型：property和direction，表示：

棋盘每格的属性（墙/蛇/普通食物/限时食物/空）和蛇的运动方向（上下左右）；

四个int型全局变量：通关分数PASS\_SCORE，最终获胜轮数MAX\_ROUND，

棋盘格高度HEIGHT和棋盘格宽度WIDTH；

辅助函数：

getHandle(int x, int y) 获得控制台句柄并隐藏光标；

主要类的设计：

Snake (SnakeNode)：蛇

Snake类内部的数据结构为SnakeNode链表；

链表每个节点储存一对横纵坐标（数组）和一对pre/next指针；

Snake含有蛇头蛇尾两个节点，表示是否存活的布尔量和运动方向；

后面出现的Board类是Snake的友元；

成员函数是构造（初始化蛇）函数和消亡函数；

Food：食物

数组储存食物的横纵坐标，Board类是Food的友元；

TempFood：限时食物

比Food增加了表示是否投放的布尔量和表示存在时间的int型变量；

本来想写成Food的subclass，不太会搞，怕弄错就没用；

Board：棋盘格

棋盘格是一个二维数组；

含有一个蛇，一个食物，一个限时食物，

表示蛇运动速度、分数、第几局和本次游戏最高分的四个int型变量；

构造函数：初始化棋盘格，用墙划分出游戏区和记分栏；

析构函数：归还创建二维数组时申请的空间；

void dropFood() 随机生成食物，如果落在蛇/障碍物上则重新生成；

void dropTempFood() 随机生成限时食物，同上；

void moveSnake() 移动蛇，具体功能如下：

从键盘获得方向键，生成新头，根据运动方向确定新头坐标；

如果新头在棋盘格所在位置是墙或者蛇身，蛇死了；

如果新头所在位置是食物，蛇长度+1，该位置属性由食物变为蛇，

判断限时食物投放条件并进行相关操作；

如果新头所在位置空白，该位置属性由空白变为蛇，

蛇尾所在位置变成空白，并删除尾节点；

给限时食物倒计时（如果有的话）并移除达到时间限制的食物；

void printBoard() 游戏开始时打印棋盘格（游戏区和记分栏）；

void playGame() 玩游戏，判断获胜和进入下一关的条件；

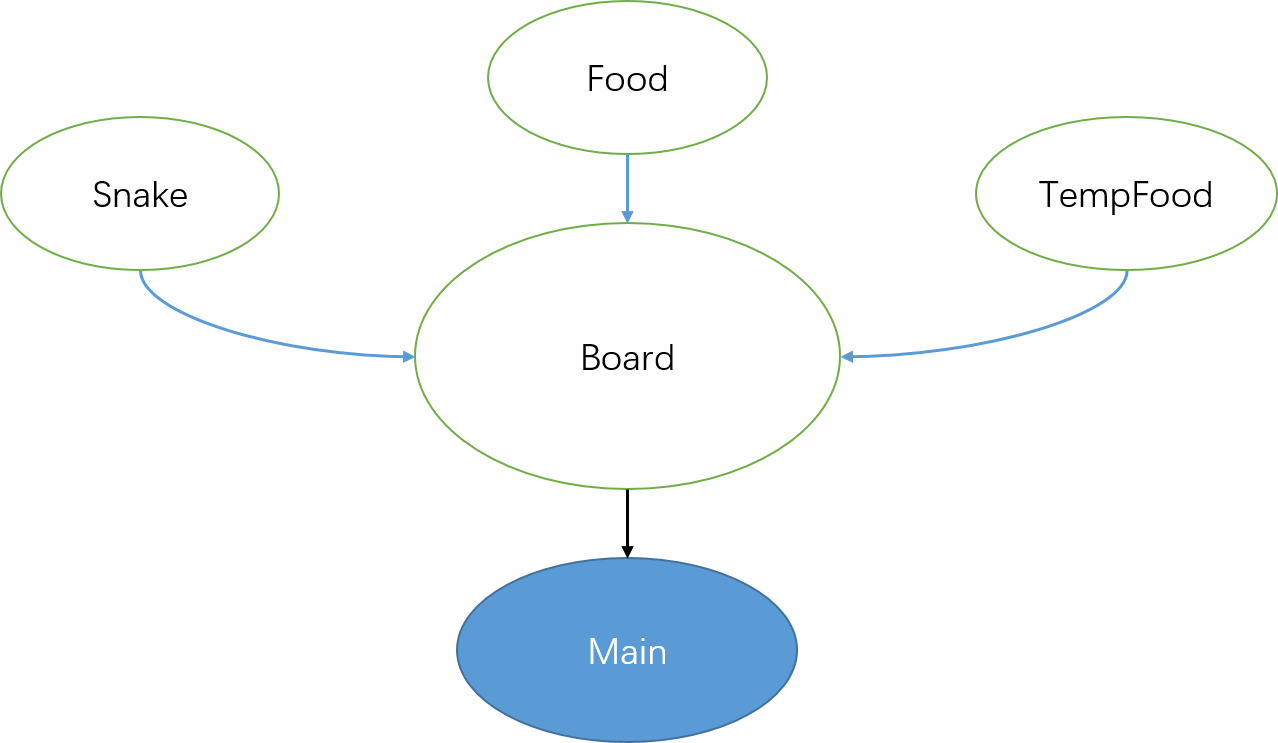
void updateGame() 每轮游戏结束后清空棋盘，删除蛇并初始化新蛇

根据本轮分数判断是否更新最高分；

达到通关分数则更新游戏设置，加快蛇运动速度并随机生成障碍物；

复活蛇并重置计分器；

int getRound() 返回轮数以作为判断最终胜利的依据；



总而言之，Board类是游戏的主体，游戏实际操作都在Board的成员函数中进行。

Snake，Food，TempFood都是Board的数据成员，在Board中创建和调用。

这样Main函数只需要创建一个Board类并调用其中的游戏函数就够了。

Main函数：

新建并初始化一个board变量b；

当轮数小于通关轮数时，轮流调用b.playGame()和b.updateGame()；

当轮数等于通关轮数MAX\_ROUND时，打印"BIG WINNER!"并退出程序；

程序的功能特点和运行操作方法：

墙的表示为#，蛇为\*，普通食物为@，限时食物为#；

通关失败则自动返回上一局，成功则自动进入下一局；

用户通过键盘上的方向键控制蛇的运行方向，按任意键进入下一局；

遇到的问题和解决方案：

不知道如何从键盘获取方向键和在WINDOWS控制台中移动光标：

百度了一些库函数的用法和参考代码；

写一两个小程序验证它们的功能，熟悉用法后再写入贪吃蛇的程序；

发现两局之间不能衔接而报错退出：

断点调试发现删除蛇并初始化新蛇时会陷入Runtime Error；

原因是for循环逐个遍历删除节点时对已经删除的节点取next操作；

多增加一个临时指针来储存待删除的节点；

发现随机生成食物时偶尔陷入死循环并把整个屏幕打满@或%：

原来是判断落在蛇和墙上重新生成时一不小心写成递归；

这样会造成永远没有循环中止条件，改成一般循环了。

最终实现的效果：



代码完成日期：2020/3/17 设计报告完成日期：2020/3/22