程序设计实践

19061529 李品成
19061536 王一凡

main

```
#include "snake.c"
//chcp 65001
int main() {
    srand((unsigned int)time(0));             //生成随机数种子
    int end = 1, result;
    while (end) {
        result = Menu();              //显示主菜单,并根据用户选择菜单选项决定游戏的执行
        switch (result) {
        case 1:                    //选择1表示,开始贪吃蛇游戏
            InitMap();             //初始化地图、蛇和食物
            while (MoveSnake());  //如果返回0,则蛇停止移动;返回1,继续移动
            break;
        case 2:                     //选择2表示,显示帮助信息
            Help();
            break;
        case 3:                     //选择3表示,显示关于信息
            About();
            break;
        case 0:                     //选择0表示,表示结束程序
            end = 0;
            break;
        }
    }
    return 0;
}
```

snake.h

```
/*引入必要的头文件*/
#include <stdio.h>
#include <Windows.h>
#include <conio.h>
#include <time.h>
#include<stdlib.h>
#include<string.h>
/*宏定义*/
#define MAP_HEIGHT 20   //定义地图高度
#define MAP_WIDTH 40    //定义地图宽度
#define UP 'w'          //定义上移键
#define DOWN 's'        //定义下移键
#define LEFT 'a'        //定义左移键
#define RIGHT 'd'       //定义右移键
#define MAX_FOOD_LEVEL 4
#define MAX_NODE 1000
```

```
#define MAX_MOVE 15
//enum food_level {poisonous, small, mid, big};

/*结构体定义*/
typedef struct      //定义食物和蛇节点位置的结构体
{
    int x;          //x坐标位置
    int y;          //y坐标位置
}Snakenode;

typedef struct {
    int x;
    int y;
    int level;
}Food;

typedef struct      //定义蛇的结构体
{
    Snakenode snakeNode[MAX_NODE];   //蛇长最多包含1000个节点
    int length;                      //蛇长度
    int speed;                       //蛇移动速度
}Snake;

/*函数定义*/
void GotoXY(int, int);  //光标定位函数
void Hide();            //隐藏光标函数
int Menu();             //主菜单函数
void Help();            //帮助信息
void About();           //关于信息
void InitMap();         //地图初始化
void PrintFood();       //生成食物
int MoveSnake();        //蛇移动
int IsCorrect();        //自撞或撞墙检测
void SpeedControl();    //速度控制
void End();             //打印结束信息
void Score(char *c,int num);//榜单排名
```

snake.c

```
#include "snake.h"

/*全局变量定义*/
#define MAX_FOOD_COUNT 7
Snake snake;    //定义蛇结构体变量
Food food[MAX_FOOD_COUNT];      //定义食物结构体变量
int end_count = 0; //需要增加几个蛇尾
Snakenode temp; //记录蛇尾
int food_count = 0; //记录食物数量
char now_Dir = RIGHT;   //当前蛇头方向
char direction = RIGHT; //预期蛇头方向
int moved = 0;
char c[100];
typedef struct{  // 得分结构体
    char name[100];
    int data;
}score;
```

```c
/*主菜单实现*/
int Menu() {
    GotoXY(40, 12);                    //定位光标位置
    printf("欢迎来到贪吃蛇小游戏");
    GotoXY(43, 14);
    printf("1.开始游戏");
    GotoXY(43, 16);
    printf("2.帮助");
    GotoXY(43, 18);
    printf("3.关于");
    GotoXY(43, 20);
    printf("其他任意键退出游戏");
    Hide();                           //隐藏光标
    char ch;
    int result = 0;
    ch = _getch();                    //接收用户输入的菜单选项
    switch (ch) {                  //根据选项设置返回结果值
        case '1': result = 1; break;
        case '2': result = 2; break;
        case '3': result = 3; break;
    }
    system("cls");                    //调用系统命令cls完成清屏操作
    return result;
}

//光标定位函数,将光标定位到(x,y)坐标位置
void GotoXY(int x, int y) {
    HANDLE hout;
    COORD cor;
    hout = GetStdHandle(STD_OUTPUT_HANDLE);
    cor.X = x;
    cor.Y = y;
    SetConsoleCursorPosition(hout, cor);
}

/*隐藏光标*/
void Hide() {
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_CURSOR_INFO cor_info = { 1, 0 };
    SetConsoleCursorInfo(hout, &cor_info);
}


/*关于菜单实现*/
void About() {
    GotoXY(30, 12);
    printf("杭州电子科技大学--程序设计综合实践案例");
    GotoXY(43, 14);
    printf("贪吃蛇-控制台游戏");
    GotoXY(43, 16);
    printf("按任意键返回上级菜单");
    Hide();     //隐藏光标
    char ch = _getch();
    system("cls");
}

/*帮助菜单实现*/
void Help() {
```

```
    GotoXY(40, 12 );
    printf("w 上");
    GotoXY(40, 14);
    printf("s 下");
    GotoXY(40, 16 );
    printf("a 左");
    GotoXY(40, 18);
    printf("d 右");
    GotoXY(40, 20);
    printf("当蛇撞到自身或撞墙时游戏结束");
    GotoXY(45, 22);
    printf("按任意键返回上级菜单");
    Hide();     //隐藏光标
    char ch = _getch();
    system("cls");
}

/*初始化地图函数*/
void InitMap() {
    printf("请输入昵称:");
    scanf("%s",c);
    system("cls");//输入昵称之后清屏
    Hide();                  //隐藏光标
    //设置蛇头位置在地图中心
    snake.snakeNode[0].x = MAP_WIDTH / 2 - 1;
    snake.snakeNode[0].y = MAP_HEIGHT / 2 - 1;
    GotoXY(snake.snakeNode[0].x, snake.snakeNode[0].y);       //将光标移动到蛇头位置
    printf("@");                //打印蛇头
    moved = 0;                  //设置初始步数
    snake.length = 3;           //设置蛇长初始值为3节
    snake.speed = 250;          //设置蛇初始移动速度为250
    now_Dir = RIGHT;            //当前蛇头方向
    //显示蛇身
    for (int i = 1; i < snake.length; i++) {
        //设置蛇身的纵坐标位置和蛇头位置相同
        snake.snakeNode[i].y = snake.snakeNode[i - 1].y;
        //设置蛇身的横坐标位置,蛇身在蛇头的左边,所以横坐标依次减1
        snake.snakeNode[i].x = snake.snakeNode[i - 1].x - 1;
        GotoXY(snake.snakeNode[i].x, snake.snakeNode[i].y);     //移动光标到蛇身位置
        printf("o");        //打印蛇身
    }
    //生成地图上下边界
    for (int i = 0; i < MAP_WIDTH; i++) {
        GotoXY(i, 0);
        printf("-");
        GotoXY(i, MAP_HEIGHT - 1);
        printf("-");
    }
    //生成地图左右边界
    for (int i = 1; i < MAP_HEIGHT - 1; i++) {
        GotoXY(0, i);
        printf("|");
        GotoXY(MAP_WIDTH - 1,i);
        printf("|");
    }
    //生成食物
    PrintFood();
    //得分说明
```

```
    GotoXY(50, 5);
    printf("当前得分:0");
}

/*生成食物函数*/
void PrintFood() {
    int flag = 1;
    food_count = rand() % MAX_FOOD_COUNT + 1;
    int n = 0;

    while (n < food_count) {
        while (flag) {
            flag = 0;
            //设置随机的食物坐标位置
            food[n].x = rand() % (MAP_WIDTH - 2) + 1;
            food[n].y = rand() % (MAP_HEIGHT - 2) + 1;
            //循环判断食物位置是否和蛇的位置重叠,如果重叠则需要重新设置食物位置
            for (int k = 0; k <= snake.length - 1; k++) {
                if (snake.snakeNode[k].x == food[n].x && snake.snakeNode[k].y ==
food[n].y) {
                    flag = 1;               //位置有重叠,需要继续循环
                    break;
                }
            }
            for (int k = 0; k < n; k++) {
                if (food[k].x == food[n].x && food[k].y == food[n].y) {
                    flag = 1;               //位置有重叠,需要继续循环
                    break;
                }
            }
        }
        flag = 1;
        food[n].level = rand() % MAX_FOOD_LEVEL;
        GotoXY(food[n].x, food[n].y);
        printf("%d", food[n].level);
        n++;
    }
}


/*蛇移动函数实现,返回值为1表示继续移动,为0表示停止移动*/
int MoveSnake() {
    int flag = 0;
    moved++;
    temp = snake.snakeNode[snake.length - 1];           //记录蛇尾
    for (int i = snake.length - 1; i >= 1; i--)
        snake.snakeNode[i] = snake.snakeNode[i - 1];    //将所有蛇身向前移动一个位置
    GotoXY(snake.snakeNode[1].x, snake.snakeNode[1].y);
    printf("o");            //前进方向打印一节蛇身,其他蛇身不需要打印
    //响应键盘修改
    if (_kbhit()) {         //键盘输入返回1,非键盘输入返回0
        direction = _getch();
        switch (direction) {
        case UP:  //按下w键
            if (now_Dir != DOWN)        //如果蛇头向下,按向上移动的键w时不起作用
                now_Dir = direction;
            break;
        case DOWN:  //按下s键
```

```
            if (now_Dir != UP)           //如果蛇头向上,按向下移动的键s时不起作用
                now_Dir = direction;
            break;
        case LEFT:  //按下a键
            if (now_Dir != RIGHT)        //如果蛇头向右,按向左移动的键a时不起作用
                now_Dir = direction;
            break;
        case RIGHT:  //按下d键
            if (now_Dir != LEFT)         //如果蛇头向左,按向右移动的键d时不起作用
                now_Dir = direction;
            break;
        }
    }
    switch (now_Dir) {                       //根据现在的方向修改蛇头的位置
    case UP:snake.snakeNode[0].y--; break;       //向上移动
    case DOWN:snake.snakeNode[0].y++; break;         //向下移动
    case LEFT:snake.snakeNode[0].x--; break;         //向左移动
    case RIGHT:snake.snakeNode[0].x++; break;    //向右移动
    }
    //打印蛇头
    GotoXY(snake.snakeNode[0].x, snake.snakeNode[0].y);
    printf("@");
    //判断是否吃到食物,如果蛇头的位置和食物的位置相同表示吃到食物
    for (int i = 0; i < food_count; i++)
    if (snake.snakeNode[0].x == food[i].x && snake.snakeNode[0].y == food[i].y)
{
        if (!food[i].level) { //遇到障碍物
            End();
            return 0;
        }
        moved = 0;
        end_count += food[i].level;
        //清空食物
        for (int j = 0; j < food_count; j++) {
            if (j != i) {
                GotoXY(food[j].x, food[j].y);
                printf(" ");
            }
        }
        food_count = 0;
        //snake.length += food[i].level;         //吃到食物,蛇长加1
        //flag = 1;              //flag为1表示吃到食物,为0表示没有吃到食物
        PrintFood();
        break;
    }
    //输出蛇此时状态
    //没吃到食物时,在原来的蛇尾打印一个空格,去掉原来的蛇尾
    if (moved == MAX_MOVE) {
        for (int j = 0; j < food_count; j++) {
            GotoXY(food[j].x, food[j].y);
            printf(" ");
        }
        moved = 0;
        PrintFood();
    }
    if (end_count) {
        if (snake.length == MAX_NODE) End();
        snake.snakeNode[snake.length] = temp;    //吃到食物,蛇尾加一节
```

```
        snake.length++;
        end_count--;
    } else {
        GotoXY(temp.x, temp.y);
        printf(" ");
        GotoXY(50, 5);
        printf("当前得分:%d", snake.length - 3);    //打印得分,得分为蛇长减原始长度3
    }
    //判断是否死亡
    if (!IsCorrect()) {          //如果自撞或撞墙,则清楚屏幕,打印最终得分,游戏结束
        End();
        return 0;
    }
    //调整速度
    SpeedControl();
    Sleep(snake.speed);     //把进程挂起一段时间,用于控制蛇移动的速度
    return 1;
}

/*判断是否自撞或撞墙,返回值为0表示自撞或撞墙,否则为1*/
int IsCorrect() {
    if (snake.snakeNode[0].x == 0 || snake.snakeNode[0].y == 0 ||
snake.snakeNode[0].x == MAP_WIDTH - 1 || snake.snakeNode[0].y == MAP_HEIGHT - 1)
//判断蛇头是否撞墙
        return 0;
    for (int i = 1; i < snake.length; i++) {   //判断蛇头是否和蛇身重叠,重叠表示自撞
        if (snake.snakeNode[0].x == snake.snakeNode[i].x && snake.snakeNode[0].y
== snake.snakeNode[i].y) {
            GotoXY(snake.snakeNode[snake.length-
1].x,snake.snakeNode[snake.length-1].y);
            printf(" ");
            snake.length--;   //蛇的长度减一
            GotoXY(50, 5);
            printf("当前得分:%d", snake.length - 3);
            return 1;
        }
    }
    return 1;
}

/*速度调整函数*/
void SpeedControl() {
    switch (snake.length) {          //根据蛇长调整蛇的移动速度
    case 6:snake.speed = 200; break;
    case 9:snake.speed = 180; break;
    case 12:snake.speed = 160; break;
    case 15:snake.speed = 140; break;
    case 18:snake.speed = 120; break;
    case 21:snake.speed = 100; break;
    case 24:snake.speed = 80; break;
    case 27:snake.speed = 60; break;
    case 30:snake.speed = 40; break;
    default:break;
    }
}

void End() {
    system("cls");
```

```
    GotoXY(45, 14);
    printf("最终得分:%d\n", snake.length - 3);
    Score(c,snake.length - 3);
    printf("按任意键返回主菜单");
    char c = _getch();
    system("cls");
}
/*排名函数*/
void Score(char *c,int num) {
    FILE *fp = NULL;
    fp = fopen("..\\score\\score.txt","r");//在指定目录下创建.txt文件
    char ch;
    int k=0;
    score s[5];
    for(int i=0;i<5;i++){
        fscanf(fp,"%s %d",s[i].name,&(s[i].data));
    }
    fclose(fp); //关闭文件
    for(int i=0;i<4;i++){//排序
        for(int j=0;j<4-i;j++){
            score flag;
            if(s[j].data>s[j+1].data){
                flag=s[j];
                s[j]=s[j+1];
                s[j+1]=flag;
            }
        }
    }
    int flag=0;
    score me;
    strcpy(me.name,c);
    me.data=num;
    for(int i=4;i>=0;i--){//判断是否打破纪录,插入
        if(num>s[i].data){
            for(int j=1;j<=i;j++){
                s[j-1]=s[j];
            }
            s[i]=me;
            flag=1;
            break;
        }
    }

    GotoXY(45, 8);
    printf("当前用户排名:");
    for(int i=4;i>=0;i--){
    GotoXY(45,13-i);
        printf("第%d名%s的分数为%d",5-i,s[i].name,s[i].data);
    }
    fp = fopen("..\\score\\score.txt","w");
    for(int i=4;i>=0;i--){
        fprintf(fp,"%s %d\n",s[i].name,s[i].data);
    }
    fclose(fp);
    if(flag==1){
        GotoXY(45, 17);
        printf("恭喜你%s,打破了纪录!",c);
    }else{
```

```
        GotoXY(45, 17);
        printf("很遗憾,您未打破记录,不能进入我们的榜单");
    }
}
```