****

**《程序设计课程实践》设计文档**

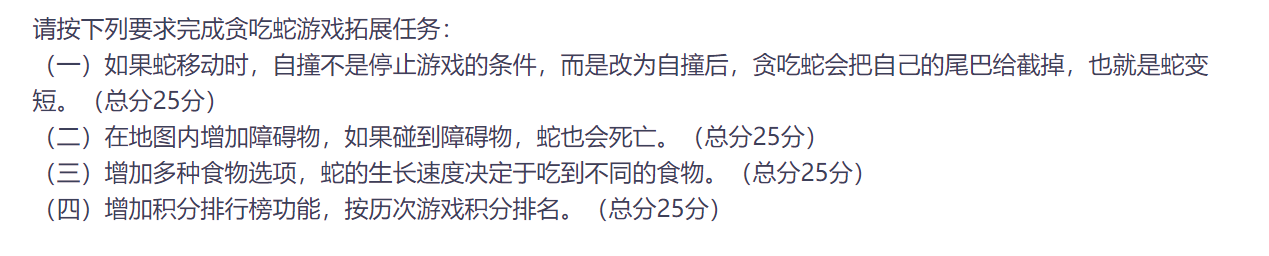
# 第7章 第1次作业

**学号： 21052115**

**姓名： 姜曜楠**

完成时间 2022 年 5 月

## 题目



## 2、程序设计思路

在贪吃蛇基础程序的设计上：

1. 在moveSnake()中添加自撞情况，若自撞则蛇长减一，积分和速度不受影响
2. 增加障碍物设计，大体上和食物产生类似，用#表示障碍物，最多产生OBSTACLE\_MAX个障碍物。同时，修改蛇死亡条件为撞墙或者撞到障碍物
3. 食物种类扩充为四种，每次随机产生三个食物（种类均随机），且食物产生条件限制增加“不能和障碍物重叠”。蛇每吃到一个食物，剩下两个食物消失，重新产生三个随机食物。因食物积分整体扩大，修改速度调整函数为每5积分修改一次速度
4. 积分排行榜用Score类型的降序数组存储，榜上只显示前RANKING\_MAX的积分，初始时榜上积分均为0。每场游戏结束，比较积分并更新排行榜，榜上同分不更新，退出程序排行榜清零
5. 修改Help()函数中的说明文字，使之与升级后的功能相对应

## 程序源码

#include "snake.h"

int main(void)

{

    srand((unsigned int)time(0));

    bool isContinue = true;

    while (isContinue)

    {

        int choice = PrintMenuAndGetChoice();

        switch (choice)

        {

        case 1:

            StartGame();

            break;

        case 2:

            PrintHelp();

            break;

        case 3:

            PrintAbout();

            break;

        default:

            isContinue = false;

            break;

        }

    }

    return 0;

}

#ifndef SNAKE\_H

#define SNAKE\_H

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>

#define MAP\_HEIGHT 20

#define MAP\_WIDTH 40

#define MAX\_SNAKE\_LEN 100

#define UP 'w'

#define DOWN 's'

#define LEFT 'a'

#define RIGHT 'd'

typedef int Score;

typedef char direction;

typedef struct

{

    int x;

    int y;

} SnakeNode;

typedef struct

{

    SnakeNode body[MAX\_SNAKE\_LEN];

    Score score;

    int length;

    int speed;

    direction nowForward, laterForward;

} Snake;

enum considerHeadOrNot

{

    Head,

    NoHead,

};

void GotoXY(int x, int y);

void Hide(void);

/\*

    startNode: Mainly whether or not to consider the snake head

        optional value reference: value in enum considerHeadOrNot

 \*/

bool IsOnSnake(int startNode, int x, int y);

int PrintMenuAndGetChoice(void);

void PrintMenu(void);

int GetChoice(void);

void StartGame(void);

void PrintHelp(void);

void PrintAbout(void);

#endif

#include <windows.h>

#include <conio.h>

#include "food.h"

#include "obstacle.h"

#include "rank.h"

#include "snake.h"

Snake snake;

/\* 局部函数定义 \*/

void init(void);

void initMap(void);

void initSnake(void);

void printScore(void);

void moveSnake(void);

/\* operate direction effectively \*/

bool isEffective(void);

/\* direction conflict \*/

bool isConflict(void);

void changeHead(void);

/\*

    snake.speed: sleeping delay time

    the higher score, the lower speed value

 \*/

void speedControl(void);

bool isDied(void);

bool isOnEdge(int x, int y);

/\* 接口函数实现 \*/

void GotoXY(int x, int y)

{

    HANDLE hout = GetStdHandle(STD\_OUTPUT\_HANDLE);

    COORD cor = {

        cor.X = x,

        cor.Y = y,

    };

    SetConsoleCursorPosition(hout, cor);

}

void Hide(void)

{

    HANDLE hout = GetStdHandle(STD\_OUTPUT\_HANDLE);

    CONSOLE\_CURSOR\_INFO corInfo = {1, false};

    SetConsoleCursorInfo(hout, &corInfo);

}

bool IsOnSnake(int startNode, int x, int y)

{

    for (int i = startNode; i < snake.length; i++)

    {

        if (x == snake.body[i].x && y == snake.body[i].y)

        {

            return true;

        }

    }

    return false;

}

int PrintMenuAndGetChoice(void)

{

    PrintMenu();

    int result = GetChoice();

    system("cls");

    return result;

}

void PrintMenu(void)

{

    GotoXY(40, 12);

    printf("Welcome to Snake Game");

    GotoXY(43, 14);

    printf("1. Start Game");

    GotoXY(43, 16);

    printf("2. View Help");

    GotoXY(43, 18);

    printf("3. Information about us");

    GotoXY(43, 20);

    printf("Any other key to exit the game");

    Hide();

}

int GetChoice(void)

{

    char ch = \_getch();

    switch (ch)

    {

    case '1':

        return 1;

    case '2':

        return 2;

    case '3':

        return 3;

    default:

        return 0;

    }

}

void StartGame(void)

{

    init();

    do

    {

        moveSnake();

    } while (!isDied());

    system("cls");

    GotoXY(45, 14);

    printf("Final Score: %d", snake.score);

    ChangeRanking(snake.score);

    GotoXY(45, 16);

    printf("You lose!");

    GotoXY(45, 18);

    printf("Any key to return to menu");

    char ch = \_getch();

    system("cls");

}

void PrintHelp(void)

{

    GotoXY(20, 12);

    printf("Operation:");

    GotoXY(20, 14);

    printf("'w' UP");

    GotoXY(20, 16);

    printf("'s' DOWN");

    GotoXY(20, 18);

    printf("'a' LEFT");

    GotoXY(20, 20);

    printf("'d' RIGHT");

    GotoXY(40, 12);

    printf("Food -> Score:");

    GotoXY(40, 14);

    printf("'$' -> 1");

    GotoXY(40, 16);

    printf("'A' -> 2");

    GotoXY(40, 18);

    printf("'M' -> 3");

    GotoXY(40, 20);

    printf("'U' -> 4");

    GotoXY(60, 18);

    printf("If snake bites itself, its length will shorten");

    GotoXY(60, 20);

    printf("Game over when the snake hits a wall or an obstacle '#' ");

    GotoXY(60, 22);

    printf("Any key to return to menu");

    Hide();

    char ch = \_getch();

    system("cls");

}

void PrintAbout(void)

{

    GotoXY(30, 12);

    printf("HDU - program synthesis design practice");

    GotoXY(43, 14);

    printf("Snake - Console Game");

    GotoXY(43, 16);

    printf("Any key to return to menu");

    Hide();

    char ch = \_getch();

    system("cls");

}

/\* 局部函数实现 \*/

void init(void)

{

    Hide();

    initMap();

    initSnake();

    PrintObstacle();

    PrintFood();

    printScore();

    PrintRanking();

}

void initMap(void)

{

    for (int i = 0; i < MAP\_WIDTH; i++)

    {

        GotoXY(i, 0);

        printf("-");

        GotoXY(i, MAP\_HEIGHT - 1);

        printf("-");

    }

    for (int i = 0; i < MAP\_HEIGHT; i++)

    {

        GotoXY(0, i);

        printf("|");

        GotoXY(MAP\_WIDTH - 1, i);

        printf("|");

    }

}

void initSnake(void)

{

    snake.score = 0;

    snake.length = 3;

    snake.speed = 250;

    snake.nowForward = snake.laterForward = RIGHT;

    snake.body[0].x = MAP\_WIDTH / 2 - 1;

    snake.body[0].y = MAP\_HEIGHT / 2 - 1;

    GotoXY(snake.body[0].x, snake.body[0].y);

    printf("@");

    for (int i = 1; i < snake.length; i++)

    {

        snake.body[i].x = snake.body[i - 1].x - 1;

        snake.body[i].y = snake.body[i - 1].y;

        GotoXY(snake.body[i].x, snake.body[i].y);

        printf("o");

    }

}

void printScore(void)

{

    GotoXY(MAP\_WIDTH + 10, 5);

    printf("Current Score: %d", snake.score);

}

void moveSnake(void)

{

    SnakeNode temp = {

        .x = snake.body[snake.length - 1].x,

        .y = snake.body[snake.length - 1].y,

    };

    for (int i = snake.length - 1; i >= 1; i--)

    {

        snake.body[i].x = snake.body[i - 1].x;

        snake.body[i].y = snake.body[i - 1].y;

    }

    GotoXY(snake.body[1].x, snake.body[1].y);

    printf("o");

    if (\_kbhit())

    {

        snake.laterForward = \_getch();

        if (isEffective() && !isConflict())

        {

            snake.nowForward = snake.laterForward;

        }

    }

    changeHead();

    if (EatFood(&snake))

    {

        snake.body[snake.length - 1].x = temp.x;

        snake.body[snake.length - 1].y = temp.y;

        ClearFood();

        PrintFood();

        printScore();

    }

    else

    {

        GotoXY(temp.x, temp.y);

        printf(" ");

    }

    speedControl();

    Sleep(snake.speed);

    if (IsOnSnake(NoHead, snake.body[0].x, snake.body[0].y))

    {

        GotoXY(snake.body[snake.length - 1].x, snake.body[snake.length - 1].y);

        printf(" ");

        snake.length--;

    }

}

bool isEffective(void)

{

    return snake.laterForward == UP ||

           snake.laterForward == DOWN ||

           snake.laterForward == LEFT ||

           snake.laterForward == RIGHT;

}

bool isConflict(void)

{

    return (snake.laterForward == UP && snake.nowForward == DOWN) ||

           (snake.laterForward == DOWN && snake.nowForward == UP) ||

           (snake.laterForward == LEFT && snake.nowForward == RIGHT) ||

           (snake.laterForward == RIGHT && snake.nowForward == LEFT);

}

void changeHead(void)

{

    switch (snake.nowForward)

    {

    case UP:

        snake.body[0].y--;

        break;

    case DOWN:

        snake.body[0].y++;

        break;

    case LEFT:

        snake.body[0].x--;

        break;

    case RIGHT:

        snake.body[0].x++;

        break;

    }

    GotoXY(snake.body[0].x, snake.body[0].y);

    printf("@");

}

void speedControl(void)

{

    switch (snake.score)

    {

    case 5:

        snake.speed = 200;

        break;

    case 10:

        snake.speed = 180;

        break;

    case 15:

        snake.speed = 160;

        break;

    case 20:

        snake.speed = 140;

        break;

    case 25:

        snake.speed = 120;

        break;

    case 30:

        snake.speed = 100;

        break;

    case 35:

        snake.speed = 80;

        break;

    case 40:

        snake.speed = 60;

        break;

    case 45:

        snake.speed = 40;

        break;

    default:

        break;

    }

}

bool isDied(void)

{

    return IsOnObstacle(snake.body[0].x, snake.body[0].y) || isOnEdge(snake.body[0].x, snake.body[0].y);

}

bool isOnEdge(int x, int y)

{

    return x == 0 || x == MAP\_WIDTH - 1 || y == 0 || y == MAP\_HEIGHT - 1;

}

#ifndef FOOD\_H

#define FOOD\_H

#include "stdbool.h"

#include "snake.h"

#define FOOD\_MAX 3

typedef char foodType;

typedef struct

{

    int x;

    int y;

    Score type;

} Food;

enum foodTypeToScore

{

    $,

    A,

    M,

    U,

    FOODTYPE\_MAX,

};

void PrintFood(void);

bool EatFood(Snake \*p);

void ClearFood(void);

#endif

#include "obstacle.h"

#include "food.h"

foodType scoreToFoodType[FOODTYPE\_MAX] = {

    '$',

    'A',

    'M',

    'U',

};

Food food[FOOD\_MAX];

void PrintFood(void)

{

    for (int i = 0; i < FOOD\_MAX; i++)

    {

        do

        {

            food[i].x = rand() % (MAP\_WIDTH - 2) + 1;

            food[i].y = rand() % (MAP\_HEIGHT - 2) + 1;

        } while (IsOnSnake(Head, food[i].x, food[i].y) || IsOnObstacle(food[i].x, food[i].y));

        food[i].type = rand() % FOODTYPE\_MAX;

        GotoXY(food[i].x, food[i].y);

        printf("%c", scoreToFoodType[food[i].type]);

    }

}

bool EatFood(Snake \*p)

{

    for (int i = 0; i < FOOD\_MAX; i++)

    {

        if (p->body[0].x == food[i].x && p->body[0].y == food[i].y)

        {

            p->length++;

            p->score += food[i].type + 1;

            return true;

        }

    }

    return false;

}

void ClearFood(void)

{

    for (int i = 0; i < FOOD\_MAX; i++)

    {

        if (!IsOnSnake(Head, food[i].x, food[i].y))

        {

            GotoXY(food[i].x, food[i].y);

            printf(" ");

        }

    }

}

#ifndef OBSTACLE\_H

#define OBSTACLE\_H

#include <stdbool.h>

#define OBSTACLE\_MAX 5

typedef struct

{

    int x;

    int y;

} Obstacle;

void PrintObstacle(void);

bool IsOnObstacle(int x, int y);

#endif

#include "snake.h"

#include "obstacle.h"

Obstacle obstacle[OBSTACLE\_MAX];

void PrintObstacle(void)

{

    for (int i = 0; i < OBSTACLE\_MAX; i++)

    {

        do

        {

            obstacle[i].x = rand() % (MAP\_WIDTH - 2) + 1;

            obstacle[i].y = rand() % (MAP\_HEIGHT - 2) + 1;

        } while (IsOnSnake(Head, obstacle[i].x, obstacle[i].x));

        GotoXY(obstacle[i].x, obstacle[i].y);

        printf("#");

    }

}

bool IsOnObstacle(int x, int y)

{

    for (int i = 0; i < OBSTACLE\_MAX; i++)

    {

        if (x == obstacle[i].x && y == obstacle[i].y)

        {

            return true;

        }

    }

    return false;

}

#ifndef RANK\_H

#define RANK\_H

#define RANKING\_MAX 3

void PrintRanking(void);

void ChangeRanking(int score);

#endif

#include "snake.h"

#include "rank.h"

Score rank[RANKING\_MAX];

void PrintRanking(void)

{

    for (int i = 0; i < RANKING\_MAX; i++)

    {

        GotoXY(MAP\_WIDTH + 10, 10 + i \* 2);

        printf("Rank %d - Score: %d", i + 1, rank[i]);

    }

}

void ChangeRanking(int score)

{

    int tempRank = -1;

    for (int i = 0; i < RANKING\_MAX; i++)

    {

        if (rank[i] == score)

        {

            break;

        }

        if (rank[i] < score)

        {

            tempRank = i;

            break;

        }

    }

    if (tempRank != -1)

    {

        for (int i = RANKING\_MAX - 1; i > tempRank; i--)

        {

            rank[i] = rank[i - 1];

        }

        rank[tempRank] = score;

    }

}

## 4、运行截图

