一、 程序结构概况

本程序由预处理命令、全局变量定义以及包括 main 函数在内的多个函数组成。

（一）预处理命令分为引用头文件和宏定义两部分

1. 头文件

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <math.h>
#include <windows.h>
#include <conio.h>
#include <io.h>
```

（1）stdio.h:标准输入输出函数；

（2）stdlib.h:srand()、rand()、malloc()、free()、system()等；

（3）time.h:time()等；

（4）string.h:strcmp()、strrchr()、strlen()、strcat()等；

（5）math.h:pow()等；

（6）windows.h:Sleep()等；

（7）conio.h:getch()等；

（8）io.h:struct _finddate_t、_findfirst()、_findnext()、_findclose()等；

2. 宏定义

```
#define HELP 13404          // hash value for \h
#define PRINT 14428         // hash value for \p
#define RANDOM 14044        // hash value for \m
#define CATALOG 12764       // hash value for \c
#define DESIGN 12892        // hash value for \d
#define QUIT 14556          // hash value for \q
#define LOAD 13916          // hash value for \l
#define SAVE 14812          // hash value for \s
#define GENERATE 13276      // hash value for \g
#define RUN 14684           // hash value for \r
#define EXIT 13020          // hash value for \e
#define END 1652581         // hash value for end
```

利用 hash() 将字符输入转化为数字，并分别将其作为关键操作词（宏名）的替换文本，所有宏定义如上图所示。

（二）全局变量

```
int x, y;           //x为地图行数，y为地图列数
int **game_array;   //声明二级指针
```

程序定义了三个全局变量：整型变量 x、y，二级指针变量 game_array。其中，用 x 表示细胞地图的行数，y 表示细胞地图的列数，且在程序使用中规定 0<x<=100,0<y<=70；通过 game_array 申请存储空间记录细胞生命状态。

（三）函数

```
25    void updatewithoutinput();      //与用户输入无关的步骤
26    void updatewithinput();         //与用户输入相关的步骤
27    long hash(char *a);             //哈希值计算
28    void getInput(char *buf);       //获取字符串输入
29    void help_display();            //打印命令提示
30    void print_map();               //打印当前地图
31    void random_map();              //生成随机地图
32    void catalog_display();         //打印文件目录
33    void design_map();              //设计地图
34    void load_map();                //导入地图
35    void save_map();                //保存地图
36    int alive_count(int p, int q);  //周围存活细胞计算
37    void step_map();                //单步运行
38    void auto_game();               //自动运行
39    void malloc_array();            //分配存储空间
40    void free_array();              //释放存储空间
41
```

以上为实现各功能的函数，注释中均说明其功能，后文会提供详细分析。

```
int main()      //主函数
{
    updatewithoutinput();
    updatewithinput();
    return 0;
}
```

main 函数如上图所示，其中，updatewithoutinput()实现了无需

用户输入的功能，而 updatewithinput()实现了通过用户输入进行相应判断的功能。

二、 主要函数的功能及分析

（一） long hash(char *a);

```
long hash(char * a)
{
    long result = 0;
    for(int index = 0; index < strlen(a); index++)
    {
        result +=  a[index] * pow(128,index);
    }
    return result;
}
```

此函数以字符指针 a 为参数，实现了将字符串转化为数字的功能，实现了二者的对应关系，便于 switch 语句的使用。

（二） void getInput(char *buf);

```
void getInput(char * buf)
{
    fgets(buf,1024,stdin);
    buf[strlen(buf) - 1] = '\0';
}
```

此函数实现了键盘输入字符串的功能。

（三） void malloc_array();

```
void malloc_array()
{
    game_array = (int **)malloc(sizeof(int*) * x);
    for (int i = 0; i < x; i++)
        game_array[i] = (int *)malloc(sizeof(int) * y);
}
```

此函数实现了二级指针进行动态分配。

（四） void free_array();

```
void free_array()
{
    for (int i = 0; i < x; i++)
        free(game_array[i]);
    free(game_array);
}
```

此函数用于动态分配空间的释放。

（五） void help_display();

```
void help_display()
{
    printf("\t[\\h]\t打印命令提示\n"
           "\t[\\p]\t打印当前地图\n"
           "\t[\\m]\t生成随机地图\n"
           "\t[\\c]\t打印文件目录\n"
           "\t[\\d]\t进入地图设计模式\n"
           "\t[\\q]\t退出地图设计模式\n"
           "\t[\\l <filename>]\t导入地图\n"
           "\t[\\s <filename>]\t保存地图\n"
           "\t[\\g]\t生成下一代生命\n"
           "\t[\\r]\t开始生命游戏\n"
           "\t[\\e]\t结束生命游戏\n"
           "\t[end]\t退出游戏\n");
}
```

此函数用于打印命令提示。

（六） void print_map();

```
void print_map()
{
    for (int i = 0; i < x; i++)
        for (int j = 0; j < y; j++)
        {
            if (game_array[i][j] == 1)
                printf("■");
            else
                printf("□");
            if (j == y - 1)
                printf("\n");
        }
}
```

依据数组元素值打印生命游戏图形。

（七） void random_map();

```
void random_map()
{
    srand((unsigned)time(NULL));
    x = rand() % 100 + 1;
    y = rand() % 70 + 1;
    malloc_array();
    for (int i = 0; i < x; i++)
        for (int j = 0; j < y; j++)
        {
            game_array[i][j] = rand() % 2;
        }
}
```

此函数利用随机生成随机数的方法，为生命游戏数组随机设置了行数、列数，并为每一个数组元素随机赋值（赋值大小均在设计要求之内）。利用此函数实现了两个功能，一是在进入游戏时便随机初始化生命游戏数组，确保了初始条件下“打印当前地图”等功能有意义；二是为构造生命游戏数组提供了一种新的方法，提高了游戏的可玩性与趣味性。

（八） void catalog_display();

```
void catalog_display()
{
    const char *path = {"./map/*.txt"};    //利用相对路径（map文件夹）
    long handle;
    struct _finddata_t files;
    handle = _findfirst(path, &files);  //第一次查找
    if(handle == -1)
        printf("无文件！\n");
    else
        printf("%s\n",files.name);
    while(!_findnext(handle, &files))    //循环查找其它地图文件
    {
        printf("%s\n",files.name);
    }
    _findclose(handle);
}
```

1. 代码分析：此函数利用了<io.h>中的结构体和函数，首先通过字

符指针 path 标明目标文件，之后利用_findfirst()将找的文件信息放入结构体 files 中（若查找失败，函数返回-1；若成功则返回查找用的句柄），再利用_findnext()和得到的句柄通过循环查找到其他符合条件的文件，并将所有此类文件的文件名打印出来。

2. 功能：此函数在程序中的运行结果为打印出 map 文件夹中所有的 txt 文件（map 文件夹用来存放生命游戏的位图），即所有的可导入文件，导入文件是更新细胞图的方法之一，打印出所有文件名称使此方法更加实用，增强了游戏的功能性。

（九） void design_map();

```
void design_map()
{
    int temx, temy;
    int count, i, j;
    char buff[100];

    printf("已进入地图设计模式(输入\\q以退出)\n");
    printf("请输入细胞图行列数(eg：6 8)(注：行数要求1~100，列数要求1~70):\n");
    while(1)
    {
        count = scanf("%d%d", &temx, &temy);
        if (count == 0)          //正确输入参数为0时，检验是否执行退出,并对错误输入进行回应
        {
            fgets(buff, 3, stdin);
            if(strcmp(buff, "\\q") == 0 && getchar() == '\n')
            {
                printf("已退出地图设计模式\n");
                return;
            }
            else
            {
                while(getchar() != '\n')
                continue;
                printf("错误！请重新输入:\n");
                continue;
            }
        }
        else if (count == 2 && temx <= 100 && temx > 0 && temy <= 70 && temy > 0 && getchar() == '\n')
            break;                                  //确定正确输入情况的条件，并对错误输入进行回应
        else
        {
            while(getchar() != '\n')
                continue;
            printf("错误！请重新输入:\n");
            continue;
        }
    }
```

```
    free_array();
    x = temx;
    y = temy;
    malloc_array();
    for (i = 0; i <  x; i++)
        for (j = 0; j < y; j++)
            game_array[i][j] = 0;                //分配内存并初始化数组
    printf("请输入存活细胞坐标(eg:0 0)(两坐标均为非负数且分别小于行列数):\n");
    while(1)
    {
        count = scanf("%d%d", &i, &j);
        if (count == 0)
        {
            fgets(buff, 3, stdin);
            if (strcmp(buff, "\\q") == 0 && getchar() == '\n')
            {
                system("cls");
                printf("地图已更新，可选择保存地图(\\s <filename>)\n");
                return;
            }
            else
            {
                printf("错误！请重新输入:\n");
                while (getchar() != '\n')
                    continue;
            }
        }
        else if (count == 2 && i < x && i >= 0 && j < y && j >= 0 && getchar() == '\n')
        {
            game_array[i][j] = 1;
            continue;
        }
        else    //对错误输入进行回应
        {
            printf("错误！请重新输入:\n");
            while(getchar() != '\n')
                continue;
        }
    }
}
```

1. 代码分析：首先定义 temx、temy 等变量，开始先进行新数组行数与列数的键入，将其分别赋予 temx、temy，利用 scanf()的返回值不同进行分类判断，将输入正确、输入\q 退出、输入其他三种不同情况分开，并给予不同回应；若得到 temx、temy 后对进行 free 操作，并赋值新的 x、y，申请空间（此处定义 temx、temy 的原因即为必须要真正得到新的 x、y 的值，才能进行 free 操作，而不能在 design 开始便 free）；之后则通过键入确定存活细胞坐标，同上述类似，对不同的输入进行相对应的反应。

2. 功能：此函数实现了自定义生命游戏细胞图的功能，并对玩家的无效输入给出了相应的错误提示。

（十） void load_map();

```
void load_map()
{
    int i;
    char fname[200] = {"./map/"};   //保证相对路径
    char name[50];                  //输入文件名
    char *pname;                    //保证文件类型为txt文件
    FILE *fp;

    printf("请输入导入文件的文件名：\n");
    for (i = 0; i < 50; i++)
    {
        name[i] = (char)getchar();
        if (name[i] == '\n')
            break;
        if (i == 49)            //防止数组溢出
        {
            printf("文件名称输入有误或文件名过长，请尝试重新导入！\n");
            while (getchar() != '\n')
                continue;
            return;
        }
    }
    name[i] = '\0';
    strcat(fname, name);            //拼接路径与文件名

    pname = strrchr(fname, '.');    //进行文件类型判断
    if (pname == NULL || strcmpi(pname, ".txt") != 0)
    {
        printf("文件名称输入错误或文件格式错误，请尝试重新导入！\n");
        return;
    }
```

```
    if ((fp = fopen(fname, "r")) != NULL)
    {
        free_array();
        fscanf(fp, "%d%d", &x, &y);
        malloc_array();
        for (int i = 0; i < x; i++)
            for (int j = 0; j < y; j++)
                fscanf(fp, "%d", &game_array[i][j]);
        print_map();
    }
    else
    {
        printf("文件打开失败，请尝试重新导入！\n");
        return;
    }
}
```

1. 代码分析：此函数首先定义了 i 等变量，且 fname 数组中放入了相对路径，之后利用 name 数组接受文件名，并且对文件名字

符数进行讨论，防止发生数组溢出，之后将 fname 和 name 中的字符串进行拼接，在 fname 中存有文件的相对路径与文件名；利用 strrchr()定位字符串中最后出现'.'的位置，判断文件类型，在此程序中要求文件后缀名为“txt”；之后利用 fopen()打开文件，并对失败情况进行回应，成功打开则利用文件信息更新细胞图。

2. 功能：此函数实现了向程序中导入文件来改变细胞图信息。

（十一） void save_map();

```
void save_map()
{
    FILE *fp;
    int j;
    char fname[200] = {"./map/"};
    char name[50];
    char *pname;

    printf("请输入保存文件的文件名：\n");
    for (j = 0; j < 50; j++)
    {
        name[j] = (char)getchar();
        if (name[j] == '\n')
            break;
        if (j == 49)
        {
            printf("名称有误，请重新保存\n");
            while (getchar() != '\n')
                continue;
            return;
        }
    }
    name[j] = '\0';
    strcat(fname, name);

    pname = strrchr(fname, '.');
    if (pname == NULL || strcmpi(pname, ".txt") != 0)
    {
        printf("文件名称输入错误或文件格式错误，请尝试重新导入！\n");
        return;
    }
```

```
    fp = fopen(fname, "w");
    fprintf(fp, "%d %d\n", x, y);
    for (int i = 0; i < x; i++)
        for (int j = 0; j < y; j++)
        {
            fprintf(fp, "%2d", game_array[i][j]);
            if (j == y - 1)
                fprintf(fp, "\n");
        }
    fclose(fp);
    printf("已保存地图。\n");
}
```

1. 代码分析：save_map()与 load_map()前半段基本相同，进行了文件名的输入、与相对路径的拼接、文件类型的检验，之后打开目标文件，依次写入行数、列数（即 x、y 的值），在逐行写入生命游戏数组的各个元素的值。
2. 功能:此函数实现了将当前的细胞图保存成为 txt 文件的功能。

（十二） int alive_count(int p, int q);

```
int alive_count(int p, int q)
{
    int count = 0;
    for (int i = -1; i <= 1; i++)
        for (int j = -1; j <= 1; j++)
            if (p + i >= 0 && p + i < x && q + j >= 0 && q + j < y)
                count += game_array[p + i][q + j];
    return (count - game_array[p][q]);
}
```

此函数对细胞图中的某个细胞进行分析，计算出其周围细胞的存活数之和，以便分析细胞图单步运行后的结果。

（十三） void step_map();

```
void step_map()
{
    int **alive_array;              //声明二级指针，申请存储空间，记录每个细胞的周围存活个数
    alive_array = (int **)malloc(sizeof(int*) *  x);
    for (int i = 0; i <  x; i++)
        alive_array[i] = (int *)malloc(sizeof(int) * y);
    for (int i = 0; i < x; i++)
        for (int j = 0; j < y; j++)
            alive_array[i][j] =  alive_count(i, j);

    for (int i = 0; i < x; i++)
        for (int j = 0; j < y; j++)
        {
            if (game_array[i][j] == 1)
            {
                if (alive_array[i][j] < 2 || alive_array[i][j] > 3)
                    game_array[i][j] = 0;
            }
            else if(game_array[i][j] == 0)
            {
                if (alive_array[i][j] == 3)
                    game_array[i][j] = 1;
            }
        }
    print_map();
    for (int i = 0; i < x; i++)
        free(alive_array[i]);
    free(alive_array);
}
```

1. 代码分析：首先定义二级指针 alive_array,并申请存储空间，数组中存放当前状态下各细胞的“周围存活细胞数目”；之后利用循环对每个细胞进行讨论，依据题目要求确定其下一步生死状态，改变数组元素值并进行打印；最后释放 alive_array 申请的空间。
2. 功能：此函数实现了打印当前细胞图单步运行的结果，实现了细胞图的更新。

（十四） void auto_game();

```
void auto_game()
{
    char ch1, ch2;
    print_map();
    while(1)
    {
        Sleep(1000);
        system("cls");
        step_map();

        while(1)
        {
            if (kbhit())        //判断键盘输入
            {
                ch1 = (char)getch();
                if (ch1 == '\r')    //若回车则暂停
                {
                    printf("输入回车继续生命游戏，输入\"\\e\"结束生命游戏，其他字符无效\n");
                    ch2 = (char)getch();
                    while(1)
                    {
                        if (ch2 == '\r')    //再次回车继续游戏
                            break;
                        else if ((ch2 == '\\') & ((ch2 = (char)getch()) == 'e'))    //"\e"退出自动模式
                            {
                                printf("退出自动模式，结束生命游戏\n");
                                return;
                            }
                    }
                }
            }
            else break;     //其他输入无效
        }

    }
}
```

1. 代码分析：首先，此函数定义了两个字符变量 ch1、ch2,并打印出当前的细胞图，随后进入 while 循环，以便实现持续不断地自动运行,Sleep()保证了自动运行时的间隔,system(“cls”)实现清屏,step_map()实现下一步细胞图的打印;后面紧跟的

while 循环,通过 kbhit()判断键盘输入,无输入跳出循环,即持续自动运行;代码中使用了 getch(),可以不显示用户输入,保证了无效输入不会影响自动运行的视觉效果;当 ch1 == '\r',即输入回车时,系统由于 getch()继续读取输入,视觉效果上即为生命游戏自动运行暂停;之后代码中的进行 if 判断,若读取回车,则跳出循环,生命游戏继续运行,若在读取到'\'后紧接读取'e',即对应键入"\e",此时结束函数,即结束自动运行模式。

2. 功能:此函数实现了细胞图更新自动运行,并能对自动运行进行暂停、离开等操作。

（十五） void updatewithoutinput();

```
void updatewithoutinput()
{
    random_map();   //随机初始化生命游戏数组
    puts("*********************************\n*********************************");
    puts("\n      生    命    游    戏\n\n       欢迎来到生命游戏       \n"
         " 在这里您将体会到元胞自动机的乐趣\n\n          准备好了吗");
    puts("\n*********************************\n*********************************");
    help_display();
}
```

此函数利用 random_map()随机初始化生命游戏数组，并打印出欢迎界面与命令提示。

（十六） void updatewithinput();

```
void updatewithinput()
{
    char buf[1024];
    while(1)
    {
        printf("请输入相应指令:");
        getInput(buf);
        long hashCode = hash(buf);
        system("cls");
        switch (hashCode)
        {
            case HELP:
            help_display();
            break;

            case PRINT:
            print_map();
            break;

            case RANDOM:
            free_array();
            random_map();
            print_map();
            break;
```

```
            case CATALOG:
            catalog_display();
            break;

            case DESIGN:
            design_map();
            break;

            case QUIT:
            printf("当前未处于地图设计模式，指令无效，请检查输入！\n");
            break;

            case LOAD:
            load_map();
            break;

            case SAVE:
            save_map();
            break;

            case GENERATE:
            step_map();
            break;

            case RUN:
            auto_game();
            break;

            case EXIT:
            printf("当前未处于自动运行游戏模式，指令无效，请检查输入！\n");
            break;

            case END:
            free_array();
            return;

            default:
            printf("错误！请检查输入！\n");
            break;
        }
    }
}
```

1. 代码分析：此函数首先定义字符数组，并在 while 循环中读取用户输入，通过 hash()进行转换，在 switch 中进行对应，分别对应不同功能，基本都已在其他函数中写过具体功能；输入“end”，先进行 free 操作再退出，输入不合命令提示，default 中打印出错误提示。
2. 功能：此函数实现了由用户输入而进入不同操作的功能，将上文多个函数通过 switch 串联在一起。

三、 程序结构设计与分析

程序结构设计方案为将程序所需各个功能封装成不同的函数，即将程序化作一个个小任务去执行；又由于大多数函数均需要生命游戏数组的相关信息，因此将细胞图的行数、列数（即 x、y）以及相应的二级指针均定义为全局变量；同时根据具体功能的需要增添相应的预处理命令；最后将所得函数进行整合，将 main 函数补充完整。具体每一个函数的功能与相关分析已在上文进行讨论。

四、 程序使用说明

源代码文件经过编译、链接后生成可执行文件，运行可执行文件即可启动程序，不过，源代码中默认程序同级路径中有一名为“map”的文件夹，以便进行文件的导入与保存。因此，推荐做法是新建一个名为“game_of_life”的文件夹，将源代码与可执行文件放于文件夹中，并在文件夹中新建一个名为“map”的文件夹，在其中放入可导入文件。运行可执行文件后，游戏界面有相关的命令提示，可根据相关说明运行游戏。

五、 编程遇到的困难及解决方案

（一） 开始未使用全局变量，进行地图设计时总是出现失败；后来发现 bug 原因，进行传参时使用的是二级指针，在函数中进行动态分配，原指针变量的值也不会改变，故改用三级指针作为参数，问题得以解决。

（二） 希望打印指定目录下所有的 txt 文件的文件名，但是没有办法；经过查询，找到学习了<io.h>中的结构体和函数，问题

得以解决。

（三） 游戏中玩家错误输入的回应问题，感觉错误输入可能性多，无从下手；经过不断地分类，确定更好的分类方法以分离出错误输入，同时，经常通过调试来检验代码的功能以不断完善。