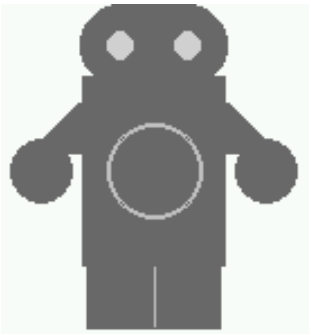
華中科技大學

C 课 设 验 收 报 告

家居机器人模拟系统



专业班级：自实 1901

小组成员：肖力文 U201915560

袁立凡 U201911484

指导老师：周纯杰教授

上交时间：2020 年 9 月 22 日

一、引言

1.背景

智能家居领域的技术突破将是通过机器人技术和物联网来促进业主和家庭之间的连接。通过了解业主需求并提供量身定制的服务，智能家居将实现更高水平的个性化体验。

智能家居，在几年内，将拥有协调人工智能软件或智能设备的力量，以聪明地管理家庭生活，并培养与我们之间的情感关系。令我们惊讶的是，我们现在已经进入了这个变革的时代。想知道怎么做吗？答案是，智能家居机器人。

简单来说，智能家居机器人是智能家庭的私人助理，它可以有效地管理家务，而无需主人亲自动手。利用人工智能和自然语言处理能力，家庭机器人将能够分析主人的需求，并提供所需的帮助。一旦所有的家庭成员都与智能家居机器人建立连接，那么家庭成员就可以与智能助理进行交谈，以获得有关家庭电器的任何信息。不仅仅是电器信息，家庭机器人也可以提供房子的安全和维护信息。有了家庭机器人，上班族照顾年迈父母和顽皮孩子所面临的共同问题也将被消除。通过从家中不同位置传感器收集的信息，机器人将分析情况，并在紧急情况下通知主人。此外，对于像“谁在家”这样的问题，机器人可以实时发送图像给主人。

从全球角度看，以智能服务机器人为主的机器人产业在不断发展，产业规模及市场空间持续扩张。目前全世界至少有 48 个国家在发展机器人，其中 25 个国家已涉足服务机器人开发，掀起一波服务机器人研发热潮。服务机器人是庞大机器人家族中的一个年轻成员，到目前为止尚没有一个严格的定义。根据其用途不同，可以划分为保洁机器人、教育机器人、医疗机器人、家用机器人、服务型机器人及娱乐机器人，应用范围非常广泛。

同时，随着工业革命的进一步发展，有两个主要原因迫使了智能家居机器人的出现：一个是人们的时间更多地投入到了生产生活中，而有更少的时间放在家务等琐碎的事情上，我们需要有一种劳动力来帮助投身于生产中的人来处理简单繁琐却又不可避免的家务事务中。另一方面，发达的电子技术、互联网技术和控制技术使得创造机器人来代替人工成为可能。

2.目的

通过对家居机器人的需求者和提供者的调查和分析，以及对一些技术的学习，查找资料，我们编写了这份软件需求分析和功能设计报告。

该项报告对于“智能家居机器人模拟系统”进行了用户需求和功能分析。包括需求分析，系统设计，模块设计，各模块之间的接口和调用，界面说明。

3.参考资料

①王士元. C 高级实用程序设计[M]. 北京：清华大学出版社，1996

②周纯杰. 标准 C 语言程序及应用[M]. 刘正林. 武汉：华中科技大学出版社，2005

③周纯杰. 程序设计教程：用 C/C++语言编程[M]. 何顶新，周凯波. 北京：机械工业出版社，2016

4.编写规范

1）命名规范

变量命名，尽量实用英文表达变量的准确定义，部分变量课实用拼音解释含义。

2）注释

函数功能都要在函数原型后著名,部分测试者难以理解的而算法和流程应该给出相应的注释

5.运行环境

1）硬件接口

处理器： Intel Pentium 166 MX 或以上

硬盘： 空间 500MB 以上

屏幕适配器： VGA 接口

系统运行内存： 要求 32MB 以上

2）软件接口

开发软件工具：更加先进的 Borland C 3.1

文字编辑工具： Notepad++

数据库： 文本存储（记事本）

操作系统： DOS WINDOWS 9X/ME/2000/XP/WINDOWS

3）控制

该系统通过鼠标与键盘直接进行控制。用户将鼠标移至需要操作的功能区进行点 击，

同时通过键盘来完成登陆，注册的输入功能。在相册，编辑界面通过鼠标和 快捷键相结合的形式简化操作。程序通过中断技术来获取鼠标的位置与键盘的输 入功能。

二、需求分析

1.需求调查

我们在参考深圳市赛梅斯凯科技有限公司的家居机器人功能说明网站（https://patents.google.com/patent/CN104571114A/zh），知乎“智能家居”的专栏“智能家居之机器人篇（https://zhuanlan.zhihu.com/p/88145165）和与家人朋友交流了他们期待的智能家居机器人的功能后，认为我们的家居机器人应该具有如下的功能（我们将功能分为 3 个部分：登录注册功能、自动功能、手动功能）：

自动功能：

自动功能是指事先就设计好的功能，到了某个时间点机器人就会去自动做的功能，具体功能有：

1）机器人应该有保证家庭安全的功能，所以我们设计有

安全检查功能：

当正门打开，识别是否有主人：有，问好；没有，警告并通过手机告知主人

2）机器人应该去为人做一些简单而又繁琐的事项，比如开关电视，开关空调，做饭，放浴缸水等，这里的选择由机器人与人的交流来决定，所以我们设计有：

主动聊天与服务功能：

<1>选择服务：

（1）选择娱乐还是工作：

娱乐：①看电视：

1.体育频道

2.娱乐频道

②听音乐：

1.古典

2.流行

工作：书房空调调到默认值、灯打开

(2)选择什么食物：

选项：

川菜、粤菜、闽菜、泰国菜、牛排、随便（电脑随机选一种）

(3)淋浴还是泡澡：

淋浴：打开热水器

泡澡：接水到浴缸

(4)是否需要拨打 120：

需要：拨打 120

不需要：回到原位

3）当主人在工作、洗澡或者做其他事情的时候，不希望有另一些繁琐的事情来打断自己，此时机器人要去做这些事情：比如开关空调、接水送达，所以我们设计有：

蓝牙自动控制家里电器和自动接水和送达的功能

4）自动换衣功能：

主人去洗澡的时候机器人从衣柜里面找一件衣服，将衣服挂在浴室外面。

5）机器人应该具有监控主人的健康状况和在主人健康状况不佳的情况下做出反应的功能，所以我们设计有

生病救援功能：

检测到主人温度过高，会端来一杯水并询问是否需要拨打 120

自动功能的逻辑线为：

16：00　有小偷闯入，触发报警模式

18：00　主人回家，询问吃什么，选择看电视还是听音乐，然后去做饭

18：30　主人吃饭，吃完饭把盘子收了，询问是要娱乐还是工作

21：00　询问是淋浴还是泡澡，然后根据主人选择执行相关功能，洗澡时机器人去换洗衣服,换洗完衣服后打开卧室空调。

（次日上午）

1：00　主人发烧，送水并询问是否需要拨打 120

6：00　做早饭

手动功能：

手动功能是指通过手机来控制机器人做事情的功能，具体功能有：

1）当主人无聊或者孤独的时候，机器人应该有能陪其聊天的功能，所以我们设计有被动聊天功能（要用户点击“聊天“才会开启）：
这里机器人会检索用户输入的文字并从里面找出关键字，根据关键字来生成回答

2）当主人觉得家里面需要打扫卫生的时候，在手机的“手动指令“界面里点击打扫卫生，机器人就会自动识别垃圾并打扫卫生，将垃圾放入垃圾桶中

3）当主人需要水的时候，在手机的“手动指令“界面里点击倒水，机器人就会先去饮水机处接一杯水，然后自动识别主人的位置，并将水送至主人处

4）在手机的“手动指令“界面里点击强制充电，机器人就会回到充电桩，并充电。

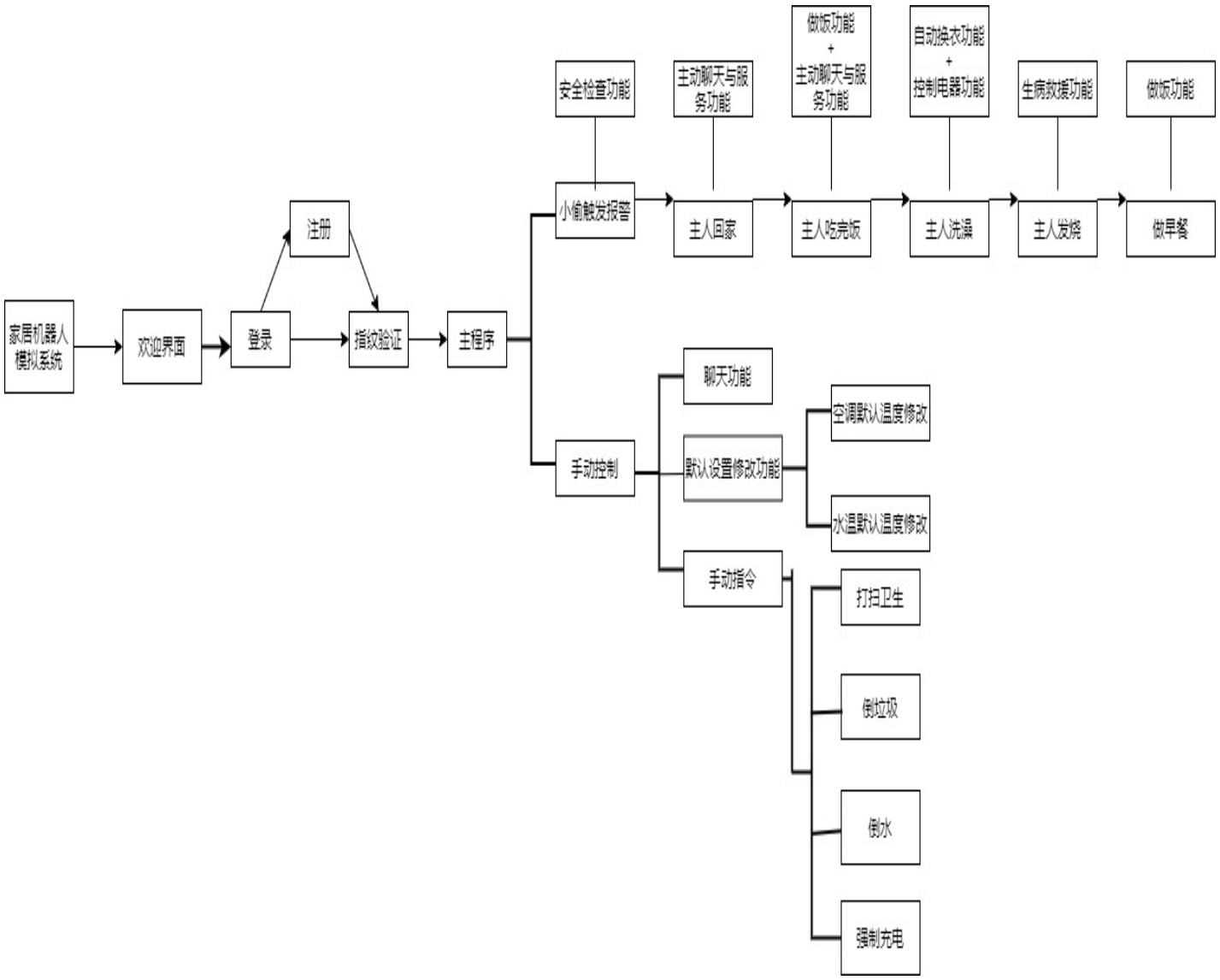
5）当主人想要修改家庭默认数值，比如空调默认温度，洗澡默认温度，就可以点击“默认设置修改“来修改默认数值

登录注册功能：

（登录注册我们会在接下来的部分详细讲述）
在登录或注册进入系统之后还有一个手机上的指纹验证功能，进一步加强系统的安全性

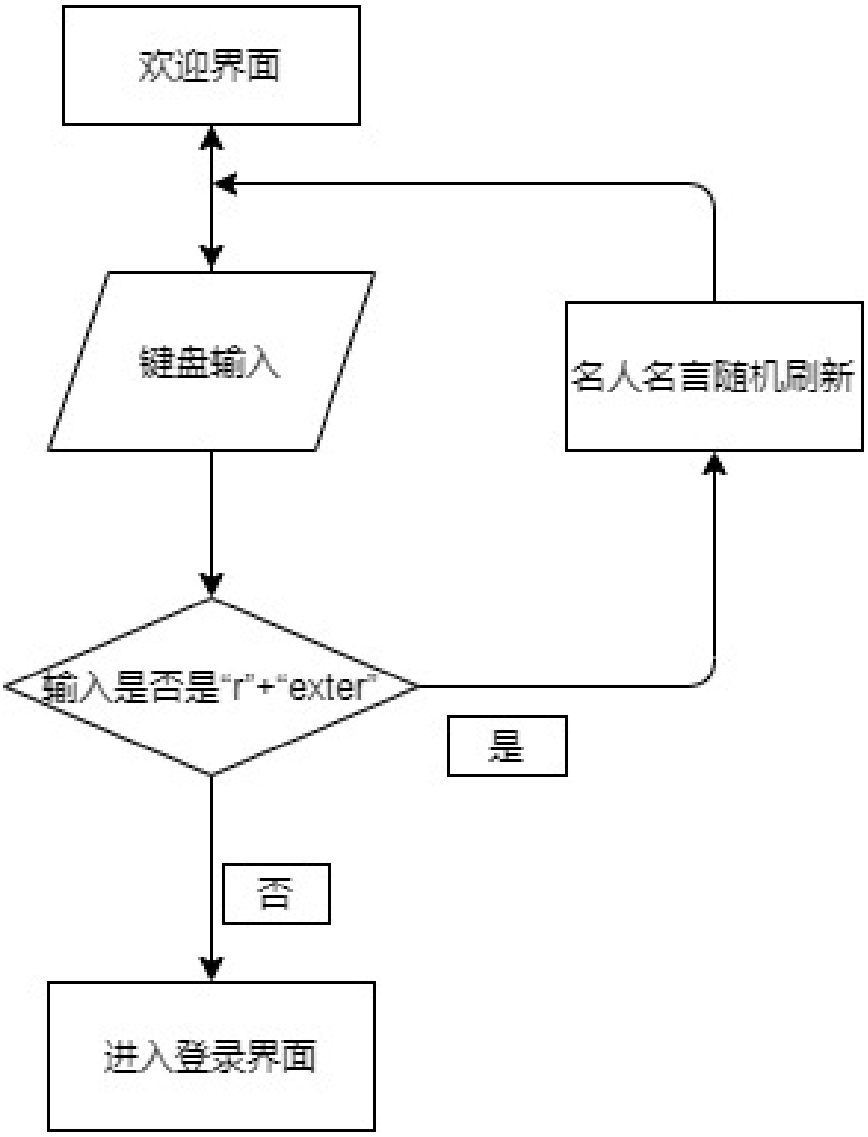
二、系统设计

1. 系统架构设计

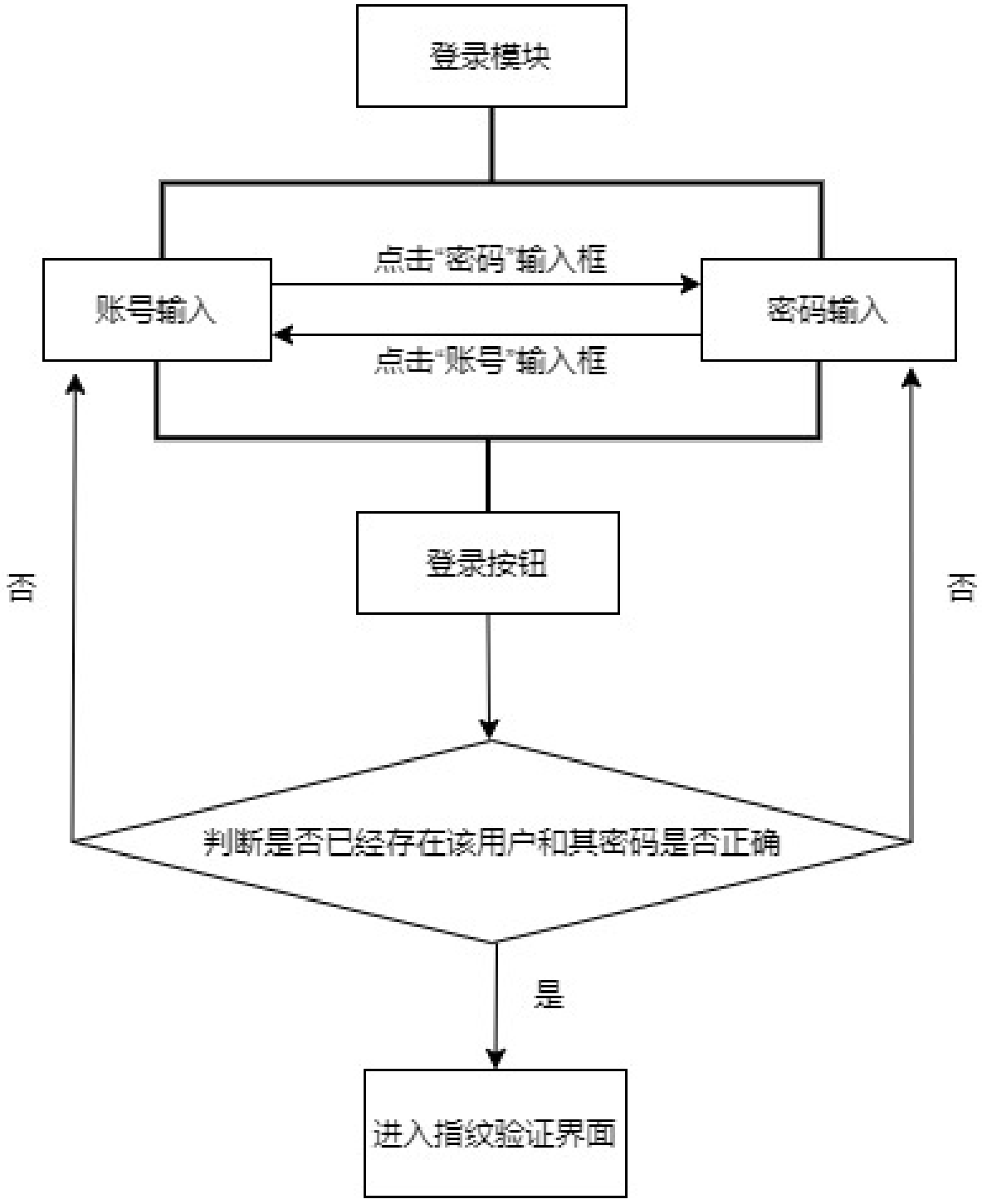


2. 软件结构设计

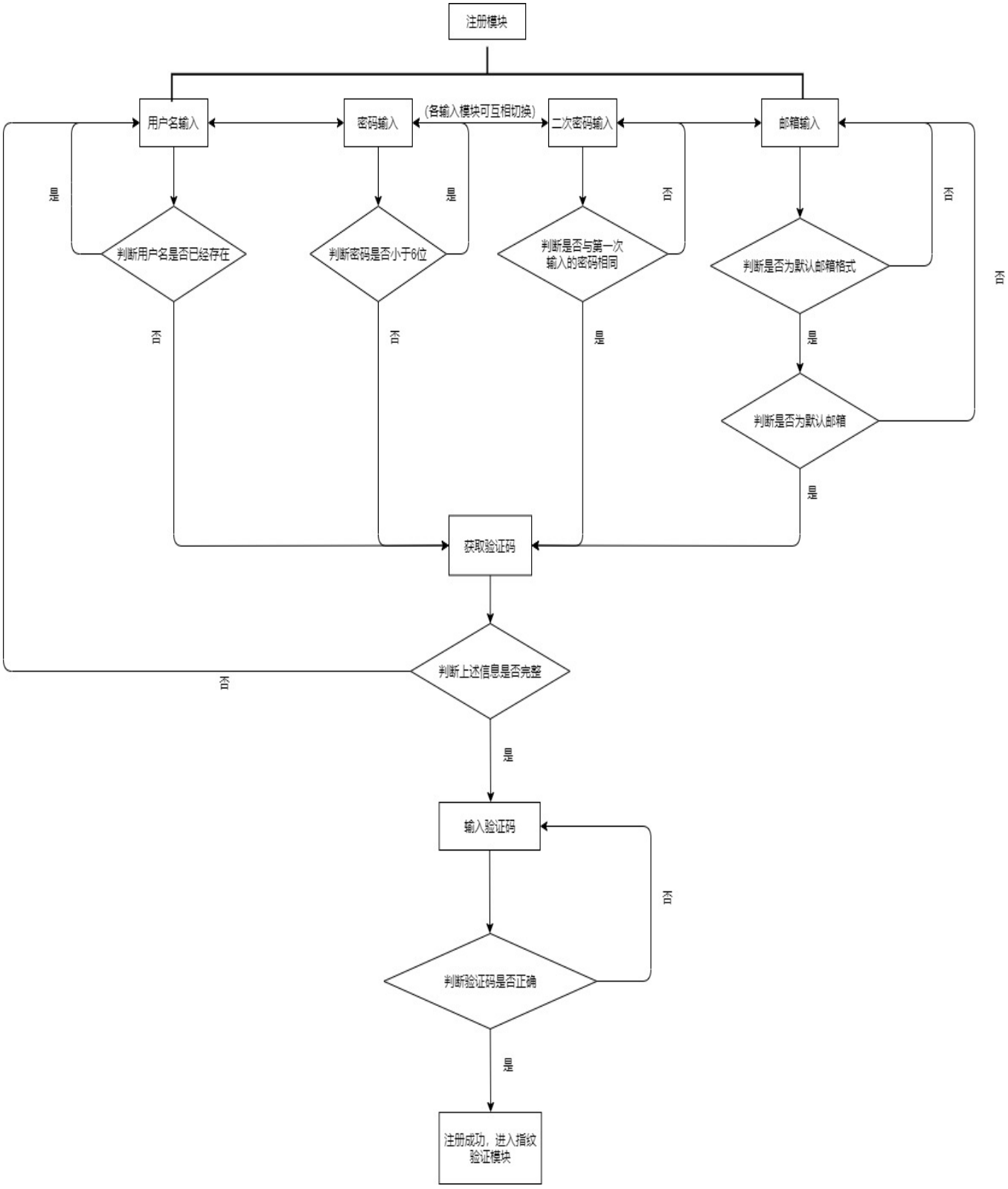
1）欢迎界面模块



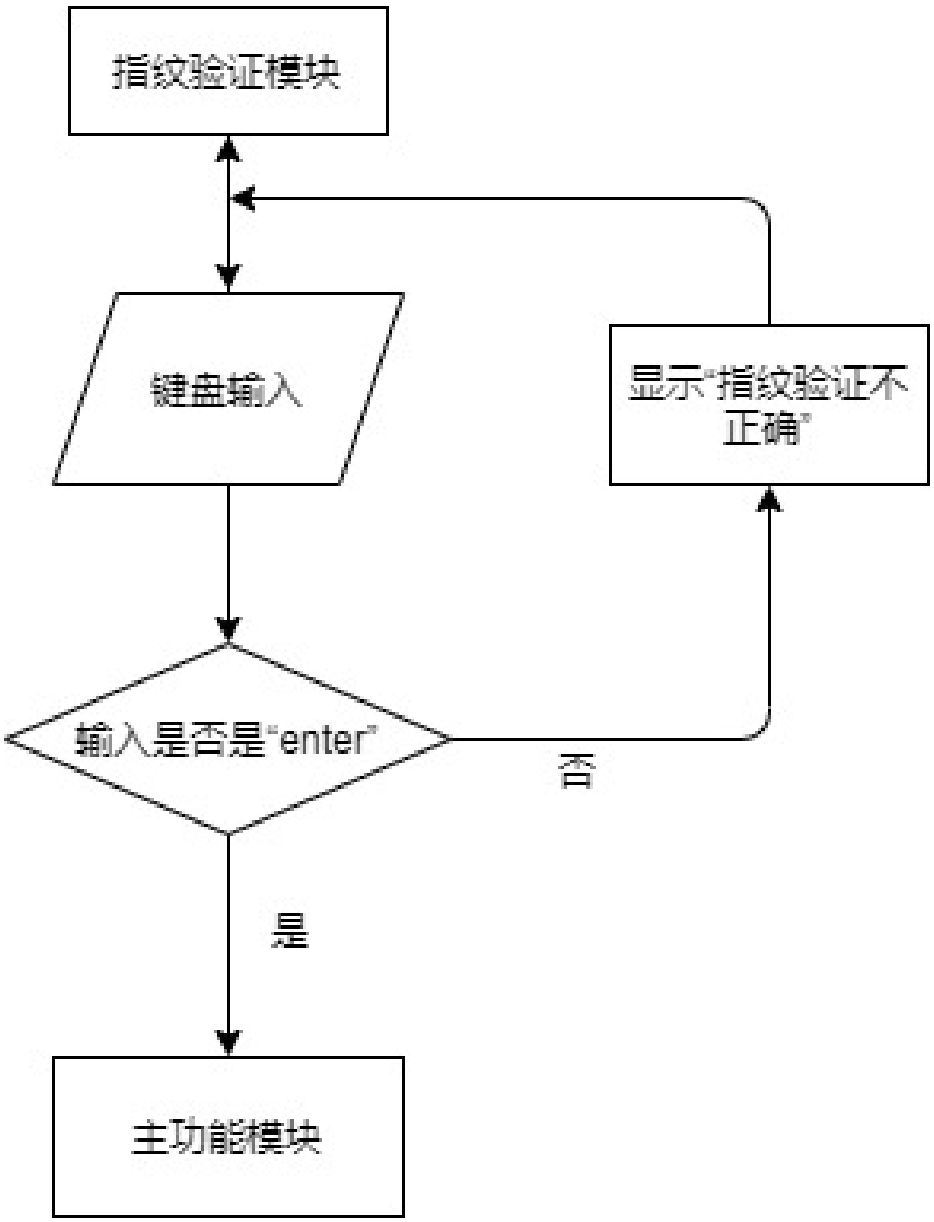
2）登录模块


登录模块
点击“密码”输入框
账号输入
密码输入
点击“账号”输入框
登录按钮
否
否
判断是否已经存在该用户和其密码是否正确
是
进入指纹验证界面

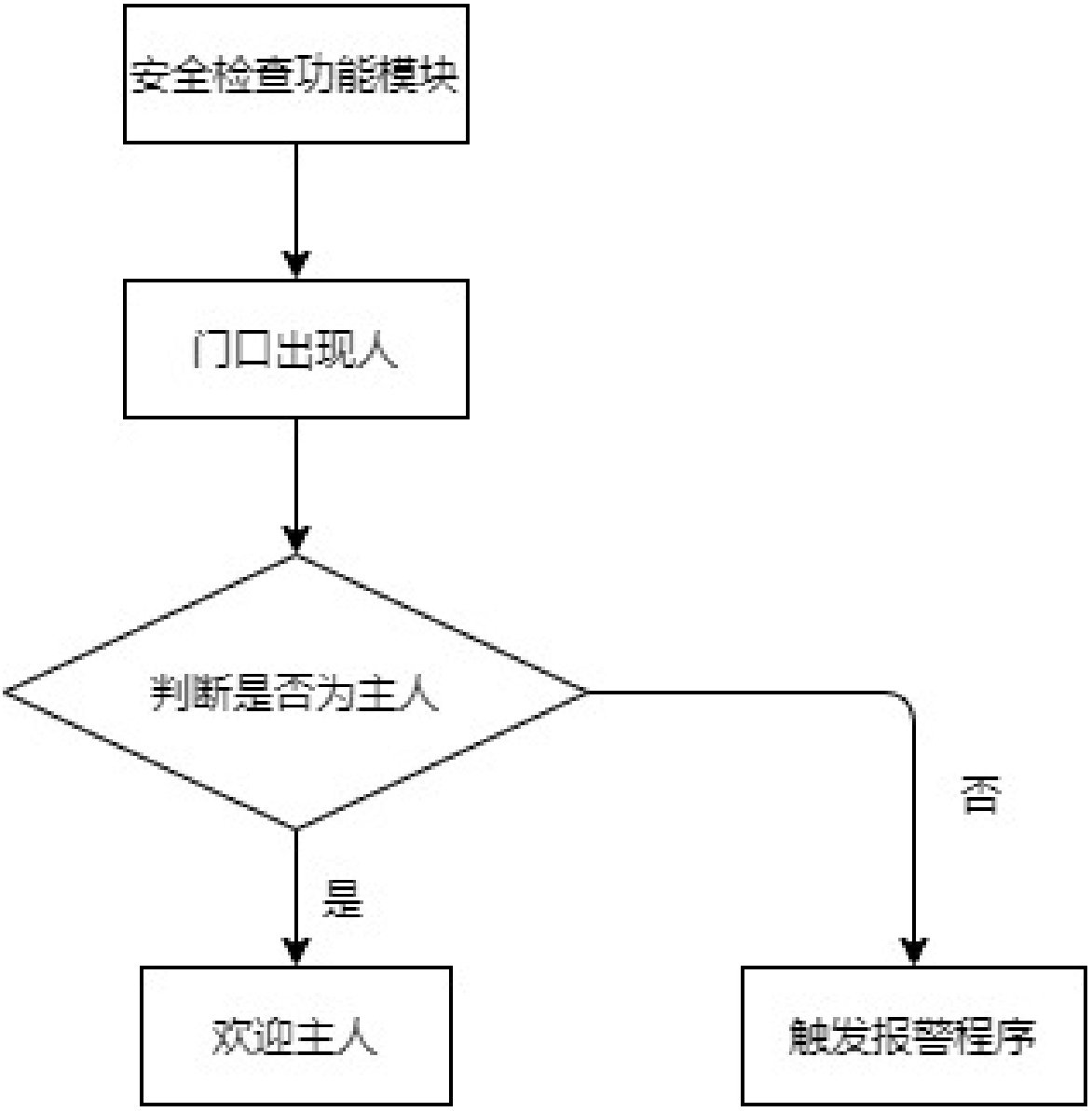
3）注册模块


注册模块
用户名输入
密码输入
(各输入模块可互相切换)
二次密码输入
邮箱输入
是
否
判断用户名是否已经存在
判断密码是否小于6位
判断是否与第一次
输入的密码相同
判断是否为默认邮箱格式
判断是否为默认邮箱
获取验证码
判断上述信息是否完整
输入验证码
判断验证码是否正确
注册成功，进入指纹
验证模块

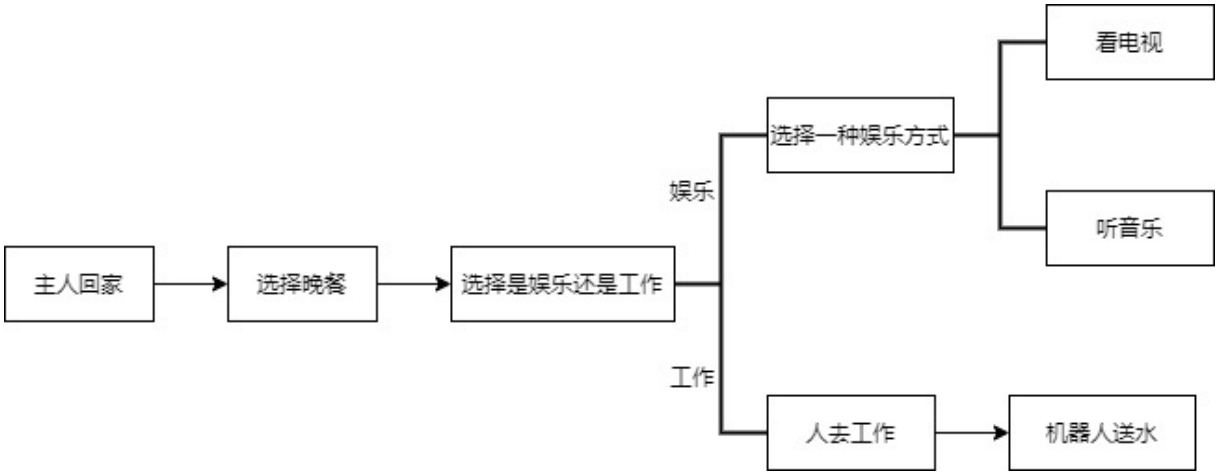
4）指纹验证模块



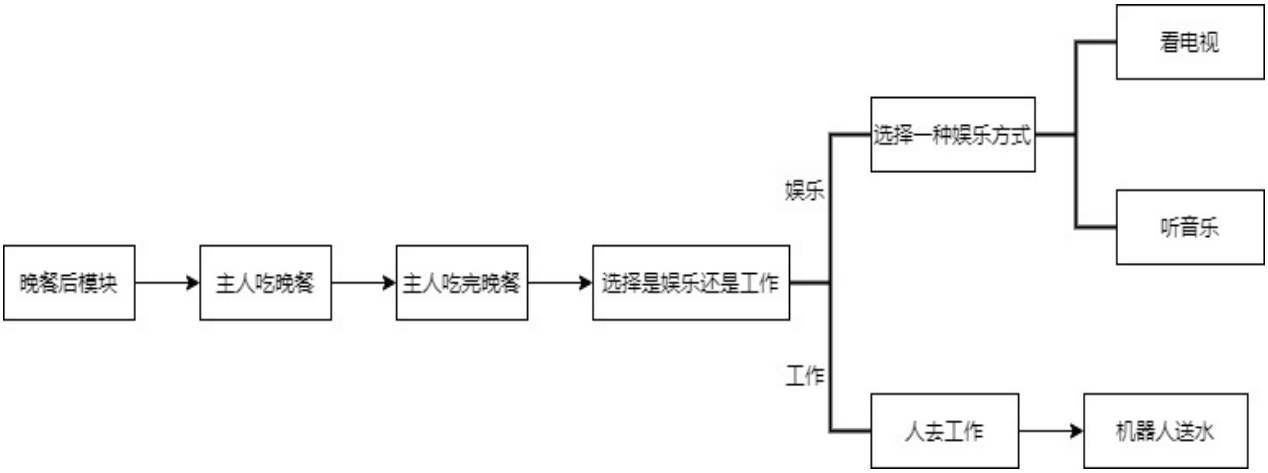
5）安全检查功能模块



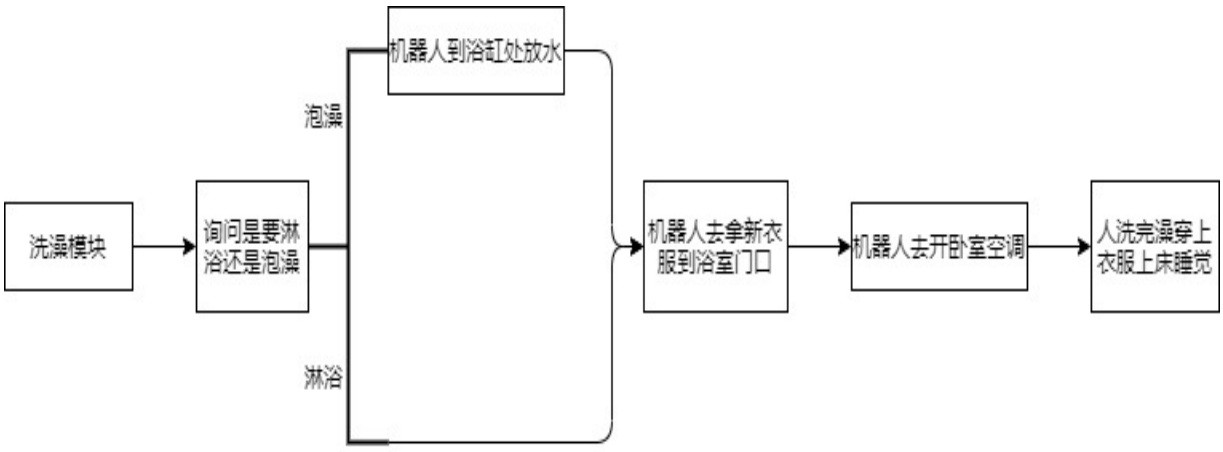
6）主人回家模块



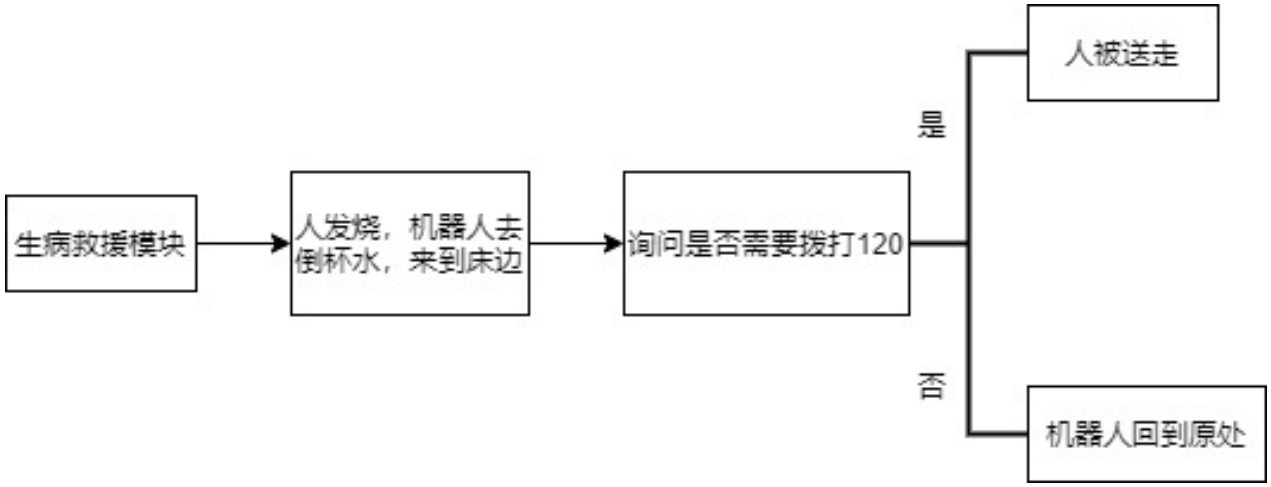
7）晚餐后模块



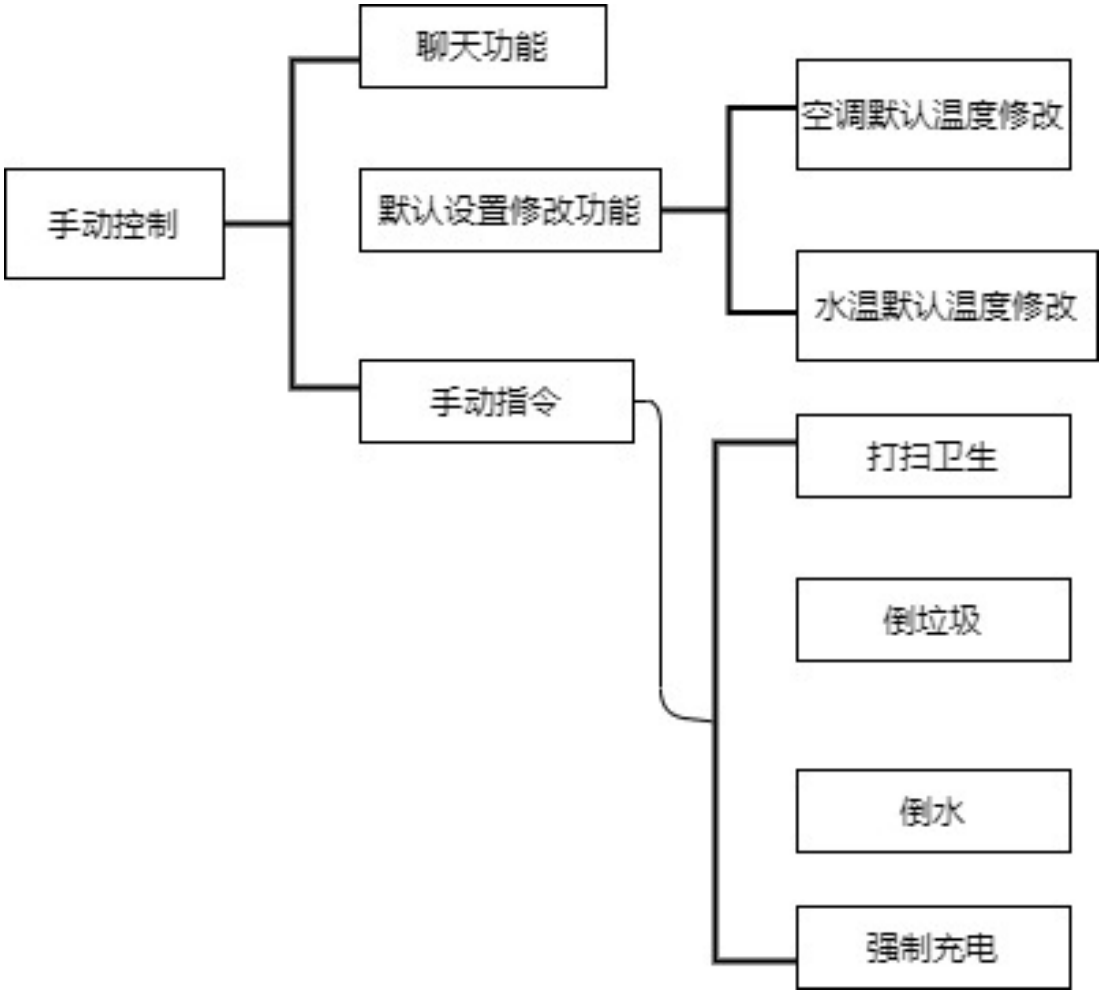
8）洗澡模块



9）生病救援模块



10）手动功能模块



11）数据接口


登录注册
加入用户信息
存储用户信息的文件
检验信息是否正确以确定登陆注册是否成功
主功能模块
修改时间和事件发生情况
存储时间和事件的信息
确定主功能函数的调用
确定人的选择
存储人的选择的信息
确定主功能函数的调用
确定状态
存储机器人和人的状态
确定主功能函数调用

12）.c 函数说明

advance.c	圆、矩形等基本图形（外源，有修改）
basicgf.c	画直线等画图基础函数（外源）
bricks.c	房间里的瓷砖
chat.c	机器人和主人聊天的主函数
chatHanz.c	汉字输出函数（主要给其他汉字函数打辅助）（外源）
chatInpu.c	汉字输入法（外源）
chatQhwh.c	汉字区号位号操作（外源）
chatShow.c	聊天功能函数
color1.c	颜色的相关处理（外源）
findway.c	数据结构基础操作、用链表储存房间地图、Dijkstra 算法寻找最短路径
finger.c	指纹解锁
frnture.c	房间里的家具
hzxs.c	汉字显示（外源）
input.c	键盘输入和数字英文输出（外源）
iph_page.c	手机界面
log_in.c	登录
MAIN_.c	主函数
process.c	程序主进程，使程序按“欢迎——注册——登录——功能展示”的顺序运行
manbody.c	画出人的前后左右各方向图像
module_b.c	当程序报错时，给出出错提示（外源）
mouse.c	鼠标操作（外源）
myhouse.c	房间界面
rbtbody..c	机器人持物和不持物时，各面的图像
rbtfunc.c	安保、医疗、打扫卫生等功能
rbtmove.c	根据 findway.c 给出的最短路径，移动机器人并画出动画
register.c	用户注册
sdzl.c	手机上的交互功能，通过手动传递指令让机器人完成某些功能
set_c.c	调整空调及热水的温度
SVGAHEAD.c	Svga 的操作（外源）
time_line.c	根据时间逻辑线，演示在日常生活中（前一天下午到第二天上午）机器人实现的功能
txt_save.c	动画相关操作，用文件存储、打印图像
user_list.c	用户的列表（与注册、登录相关）
wall.c	房间的墙壁
WELCOME.c	欢迎界面

3.数据结构设计

```
/*机器人和人的结构体,储存所有相关信息*/
typedef struct{
        char hand;                  //左手前为 1，右手为 0
        char hand_left;
        char hand_right;            //手的运动参数,为 θ
        char catch_th;    //是否持物，1 持物，对于人来说，这个参数没有意义
        unsigned int *cat;          //存储其图像信息
        int x;             //x,y 和 xpixel,ypixel 都是机器人 x 中心、y 顶上的位置
坐标
        int y;                      //小方格，16*19
        int xpixel;
        int ypixel;                 //像素点，1024*768
        int direction;              //方向，1 为右，2 为左，3 为上，4 为下
}CASE;
//findway
/**************队*************/
typedef int QElemtype;

/*QElemtype 是队里面元素的类型，Qnode 链表结点的类型，内含一个元素一个指针域*/
typedef struct Qnode
{
    QElemtype data;
    struct Qnode *next;
}Qnode, *QueuePtr;
typedef struct
{
    QueuePtr front; //队头指针
    QueuePtr rear;  //队尾
}LinkQueue;

/*****************图******************/
```

```
typedef struct LinkNode //表结点
{
    int x;
    int y;
    int direction;  //1 为右，2 为左，3 为上，4 为下
    struct LinkNode *next;
}LNode;

typedef struct Node
{
    int x;  //x 坐标（大坐标，40*40 的 block）
    int y;
    LNode *next;    //指向表结点
}VType, *Graph;     //Graph 就是定义一个 VType 型的数组

typedef struct
{
    int x;
    int y;
}Axis;

/**********寻路主要部分**************/
typedef struct
{
    int former;
    int direction;
}PathType;

/**********栈************/
//链表实现栈

typedef PathType SElemtype; //栈的元素类型就是 PathType，因为要入栈出栈的就是 path
```

```
typedef struct StackNode
{
    SElemtype data;
    struct StackNode  *next;     //指向栈顶
}SNode;

typedef struct LinkStack
{
    SNode *bottom;      //指向栈底
    SNode *top;         //指向栈顶
}LkStack;


typedef struct ChTab      //存储单个中文的信息
{
    int qhwh;
    char str[7];
}CH;

typedef struct EnTab      //存储单个英文的信息
{
    int qhwh;
    char str;
}EN;


typedef struct Coordinate  //存储输出文字时文字的坐标的信息
{
    int x;
    int y;
} Coordinate;

typedef struct Area       //存储输出文字的区域的信息
{
```

```
    Coordinate lt;
    Coordinate rb;
} Area;


typedef struct{                  //存储用户信息的单链表的单个元素
         char account[11];
         char code[11];
    struct USERS *next;
}USERS;


typedef struct tagBITMAPFILEHEADER{          //svga 的结构体
         int bfType;
         long bfsize;//文件大小，单位为字节
         int bfReserved1;//保留，必须为 0
         int bfReserved2;//保留，必须为 0
         long bfOffBits;
}BITMAPFILEHEADER;

/*BMP 信息头结构*/
typedef struct tagBITMAPINFOHEADER{
         long biSize; /*信息头大小*/
         long biWidth; /*图像宽度*/
         long biHeight; /*图像高度*/
         int biPlanes; /*必须为 1*/
         int biBitCount; /*每像素位数，必须为 1，4，8，24*/
         long     biCompression;    /*  压缩方法 */
         long     biSizeImage;      /* 实际图像大小，必须是 4 的倍数 */
         long     biXPelsPerMeter;  /*  水平方向每米像素数 */
         long     biYPelsPerMeter;  /*  垂直方向每米像素数*/
         long     biClrUsed;        /*  所用颜色数*/
         long     biClrImportant;   /* 重要的颜色数 */
} BITMAPINFOHEADER;
```

```
typedef struct tagRGBQUAD
{
        unsigned char B; /*蓝色分量，RED 缩写*/
        unsigned char G; /*绿色分量，GREEN 缩写*/
        unsigned char R; /*红色分量，BLUE 缩写*/
        unsigned char reserved;            /*保留字*/
} RGBQUAD;


typedef struct{                  //鼠标结构体
        int x;                           //left_up
        int y;                           //left_up
        int height;
        int wide;
        int click;
        int over;
}BUTTONS;

typedef struct
{
     char account[11];
     char code[11];
    struct USERS *next;
}USERS;
```

寻路的数据结构准备（Findway.c）：

(1)队的相关操作：队的初始化、销毁，判断队是否为空，以及入队、出队。

用法:类似于广度优先搜索 BFS 中队的用法(本工程的核心算法,即 Dijkstra 算法中权为 1 的情况)。

(2)图的相关操作：CreateGraph，FindAdjVex，LocateVex。

CreateGraph：根据房间的物品摆置情况建立地图，再用链表来表示图。

FindAdjVex：在链表中根据下标定位到初始点(VType 型)，再沿着其结构体

中 next 指针，按方向逐个遍历得到其邻接点。

LocateVex：查找某点在图中的编号。

(3)栈的相关操作：初始化栈、销毁栈、入栈、出栈。

作用：由 Dijkstra 算法得到的路径是逆序的，将该路径先入栈再出栈后恰好得到一个顺序的路径。

4.算法设计

1）登录注册中单链表的运用

说明：我们将用户的信息存储在 users.txt 文件中，关于此文件和链表的说明如下：

users.txt 文件说明：

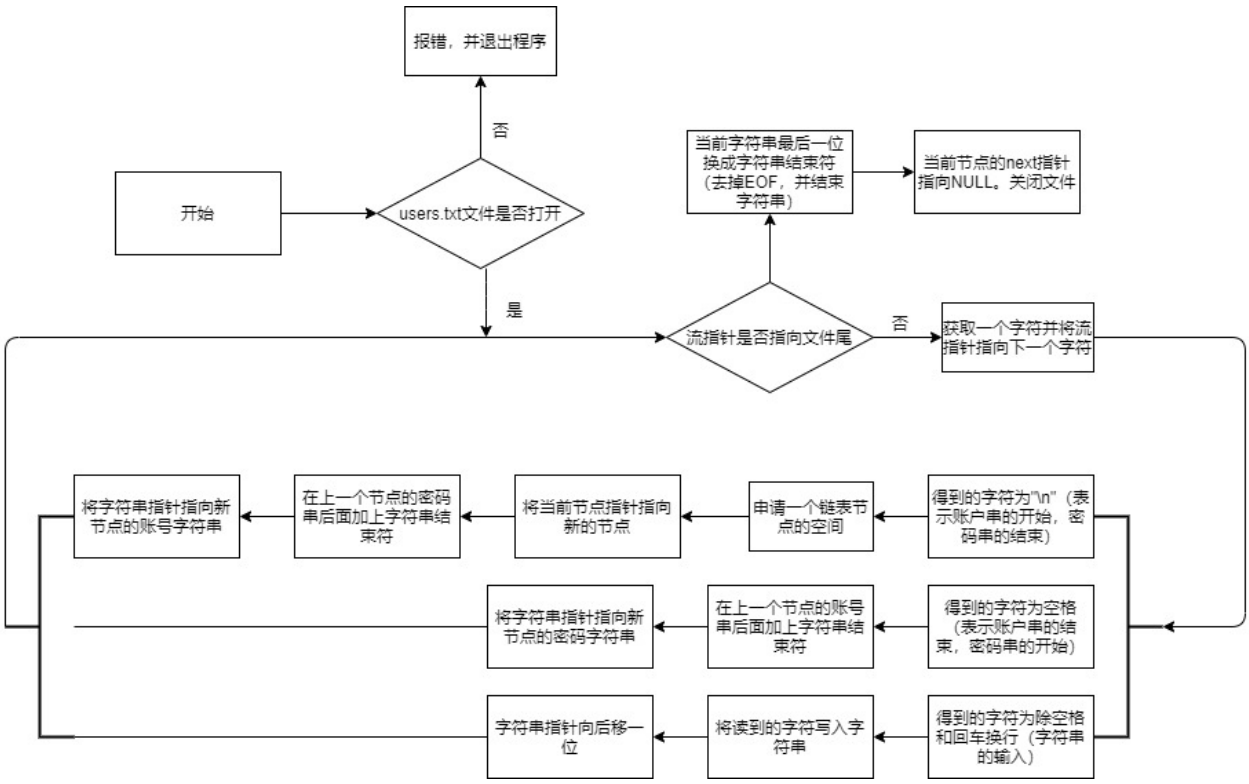
1.账号的起始符是"\n"，密码的起始符是" ".

2.一行为一个用户的信息，空格前是账号，空格后是密码

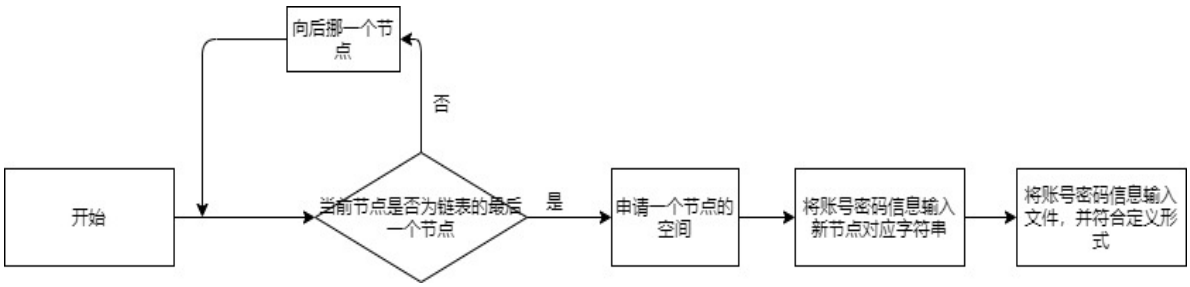
算法功能：利用链表来从文件中读取、存储用户的账号密码。在登录的时候检查账号密码是否匹配。在注册的时候在链表中添加新的结点并将信息写入文件中。

算法流程：

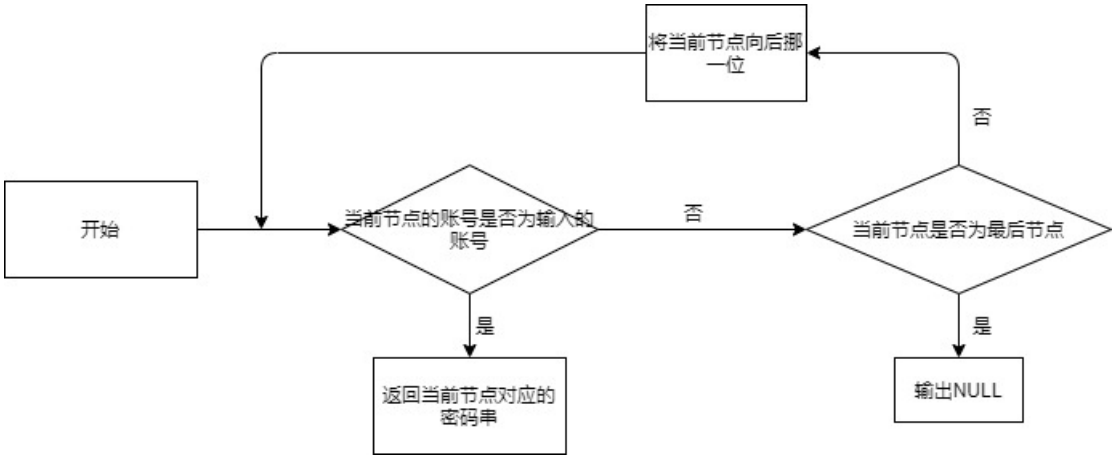
1）创建链表



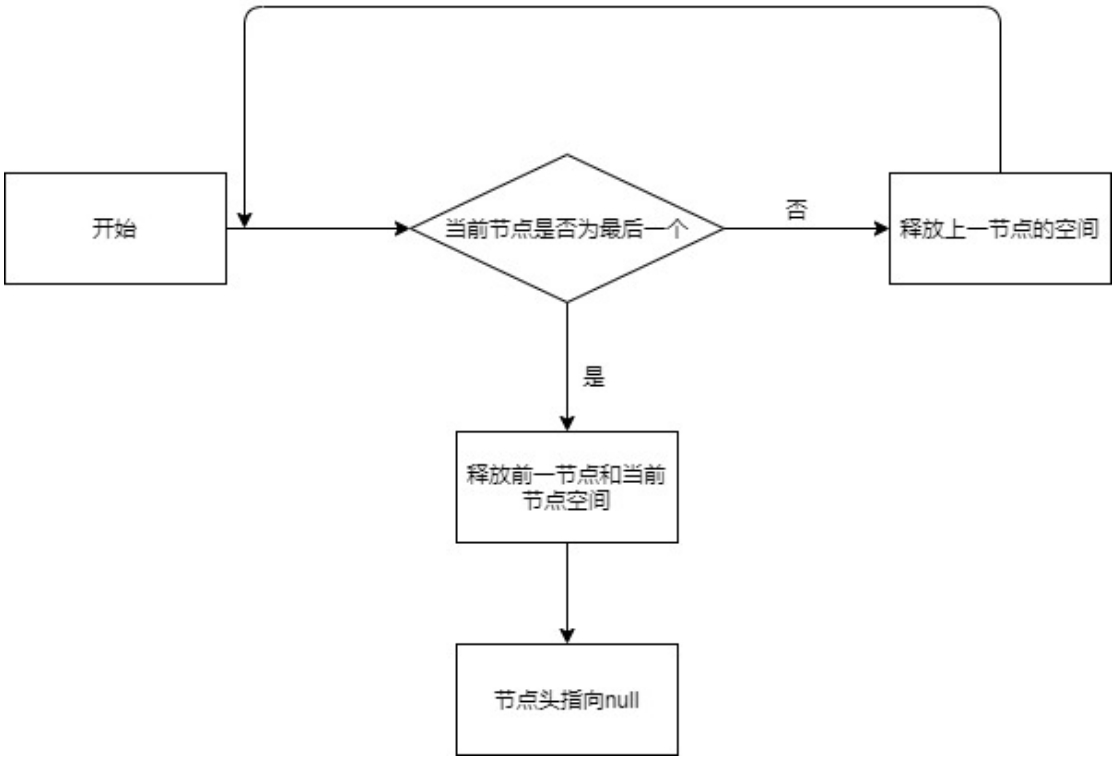
2）增加新用户



3）输出对应密码



4）释放链表



算法伪代码：

```
void create_list(USERS *head)                    /****在文件中读取用户信息进入内存，创建链表*****/
{
        打开文件；
  while(流指针未到达文件尾部)
```

```
  {
        从 users.txt 文件中提取一个字符
        if(ch=='\n')    //表示账户串的开始，密码串的结束
         {
                向系统申请空间；
          }
current=current->next;

           *p='\0';            //完善密码的字符串
           p=current->account;
         }
        else if(ch==' ')      //表示账户串的结束，密码串的开始
        {
          *p='\0';            //完善账号的字符串
          p=current->code;
        }
        else
        {
          将对应的账户串或密码串装入链表中 }
        }
    去掉 EOF，并给链表最后一个元素的密码的字符串结尾；
    current->next=NULL;
    fclose(fp);
}


void add_new_user(USERS *head,char *s1,char *s2)    /*******创建新用户的链表和
将新用户的信息写入文件中     ********/
{
        为新用户在链表中创建节点；
        将新用户的信息存入节点；
        将文件内部指针移到文件末端；
将新用户的信息写入 users.txt 文件中；
}
```

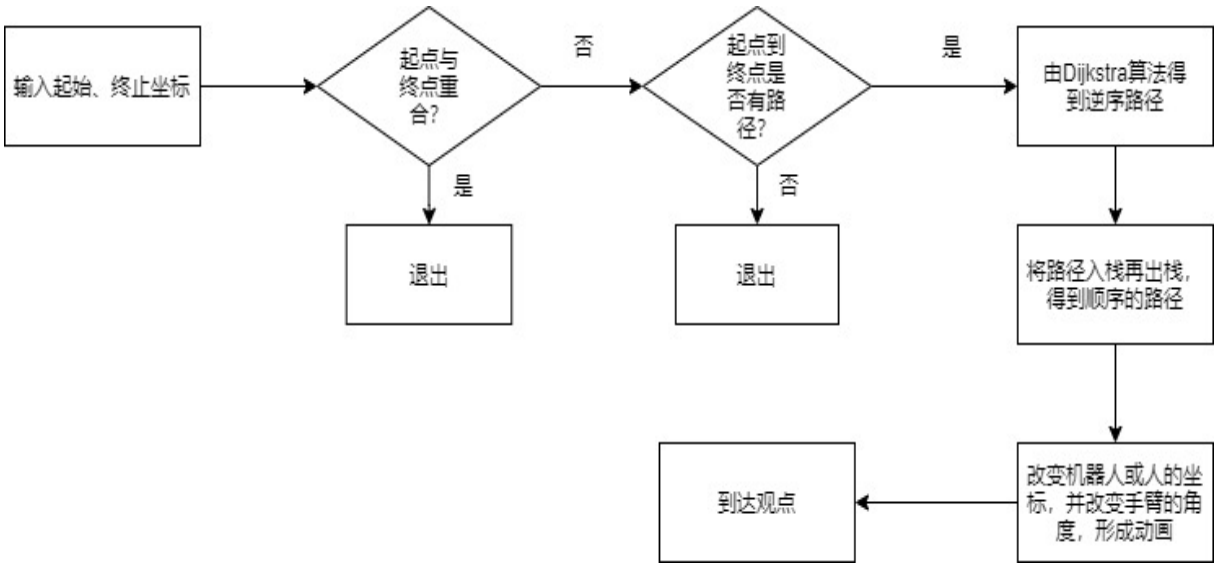
```
char *accounts_2_code(USERS *head,char *string)       /*****在已经生成的用户信息链表中查询目标账号的对应密码，并输出该密码字符串的首地址；如果没有该账号，则输出 NULL*****/
{
 while(当前节点不为最后一个且其账号串和输入账号串不对应)
 {
   节点向后移一位;
 }
 if (该节点不是最后一个)
 {
          判断为真;
 }
 else if (该节点是最后一个)
 {
          if (其账号串和输入账号串对应)
          {
                   判断为真;
          }
 }

 if (判断为假)
 {
          return NULL;
 }
 else
 {
        返回对应密码;
 }
}
```

```
void free_list(USERS *head)          /******释放链表********/
{

        if (链表为空)return 0;

        while (当前节点不是最后一个)
        {
                释放前一节点空间；
                向后移一个节点；
        }
释放迁前一节点；
        释放当前节点；
        head = NULL;
}
```

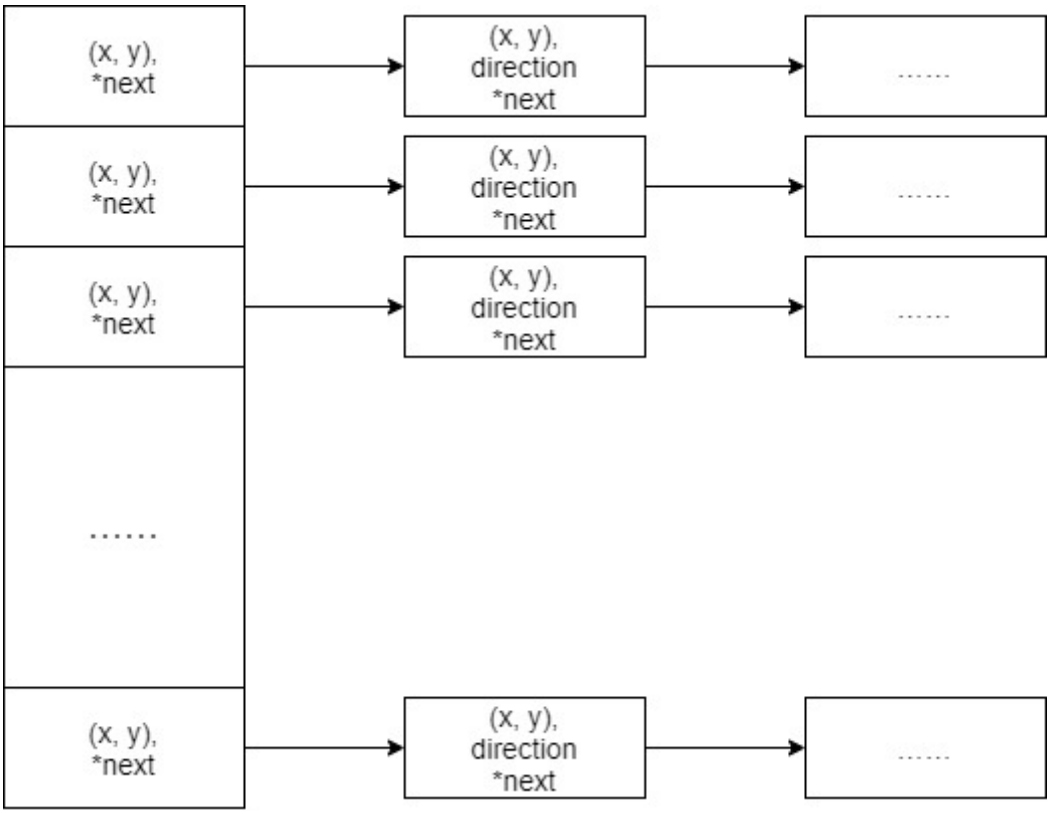
2）自动寻路

算法功能：利用 Dijkstra 算法，找出从指定起始点、终止点的最短路径。

算法流程：



数据结构：



① 图：

用链表储存房间中家具布置的地图，得到每个点与上、下、左、右四个点的邻接情况。图的链表表示如下：

② 队：

用队的操作实现 Dijkstra 算法，用以完成对图中每个点的遍历。

③ 栈：

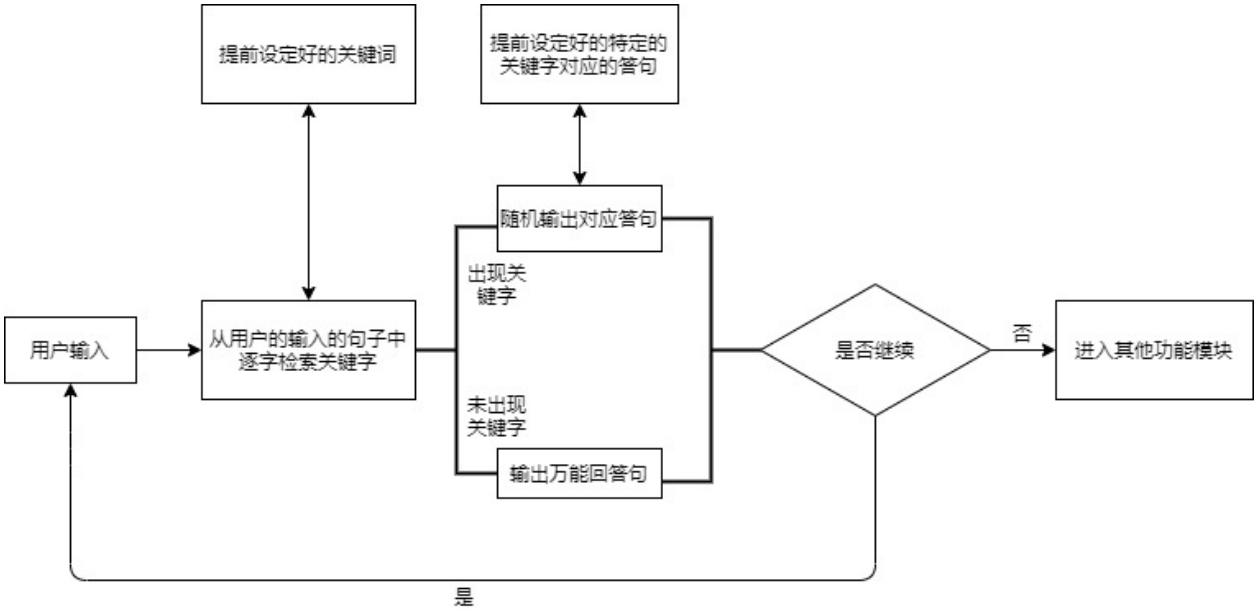
回溯 Dijkstra 的结果，得到的是逆序路径。将路径入栈后出栈，得到顺序路径。

算法的伪代码实现（只给出 Dijkstra 部分）：

```
while(队非空)                //说明还有点没有遍历完
    {
        初始化 adjvex;
        让队 Q 中的首元素 k 出队;     //队里放的是在图中的编号，所以首元素是一个数
        if(没邻接点)              //该点没有邻接点则直接跳过此段，遍历下一个点
        {                         //若该点有邻接点，则找出其邻接点，并记录方向
            for(上、下、左、右四个方向的邻接点 w)
            {
                if(往这个方向的邻接点可达，即该点没有放家具)
```

```
                {
                    if(该点还没被访问过还没被访问过)
                    {
                        更新 w 到起始点的距离（k 到起始点的距离+1）;
                        更新路径，使 w 的前一个结点是 k;
                        记录 k 走到 w 的方向;
                        w 入队;
                    }

                    if(找到终点)
                    {
                        销毁队;
                         退出;
                    }
                }
            }
        }
    }
```

3）聊天中的返回答句

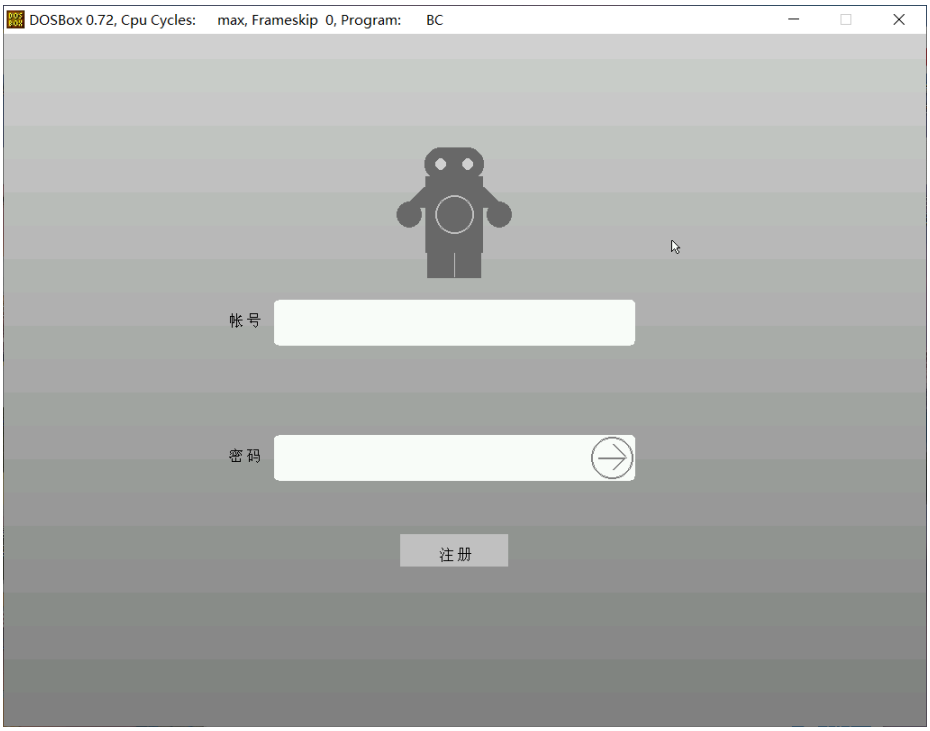


5. 界面设计

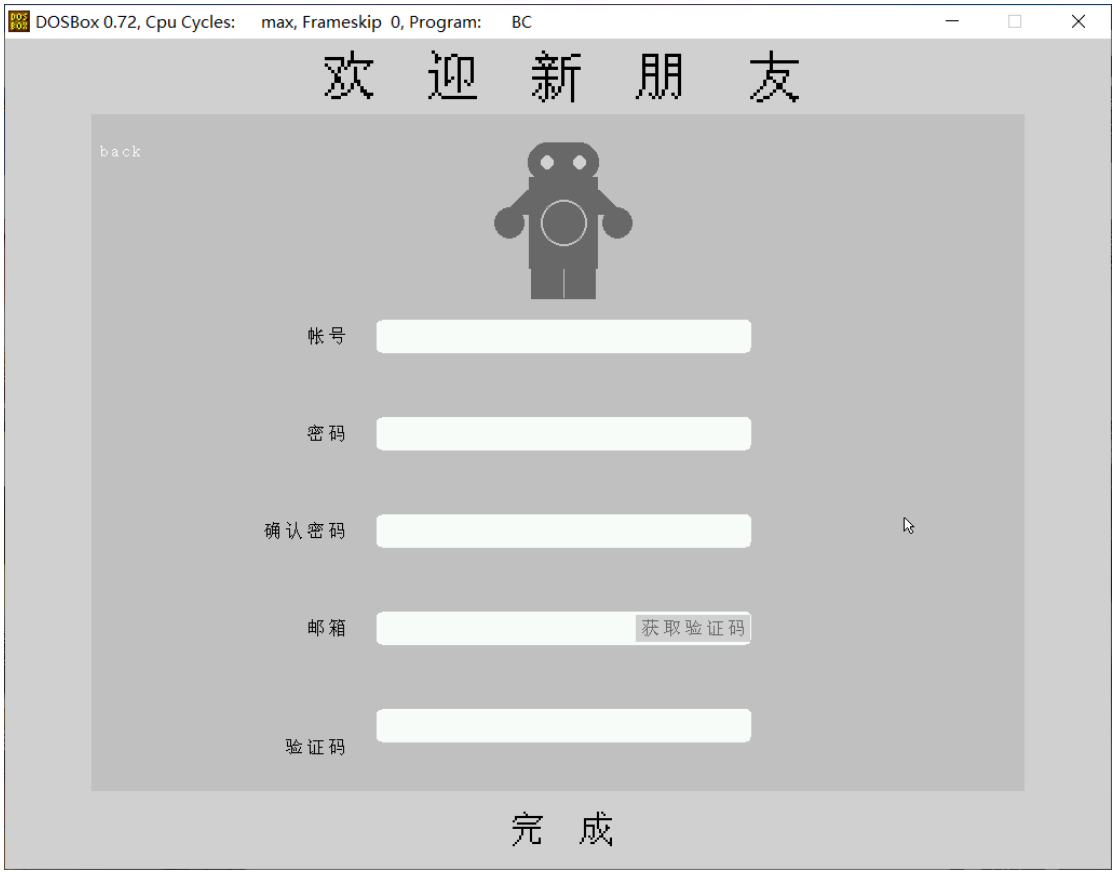
1）欢迎界面

界面选用的主题是模仿 iOS 早年的界面设计，以银色为主，并且加有渐变，机器人的 logo 是模仿苹果的 logo。界面的最下边有随机出现的乔布斯的名言，总共存了 5 句，当我们按下“r“后再按下”enter“键，就可以切换名言。

2）登录界面

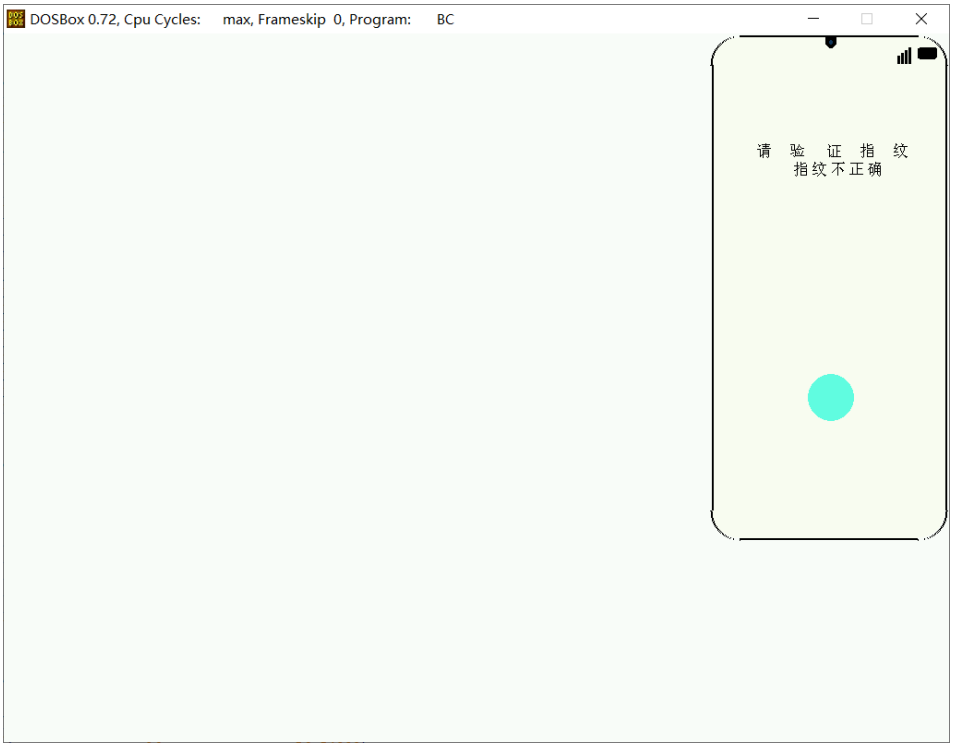


3）注册界面

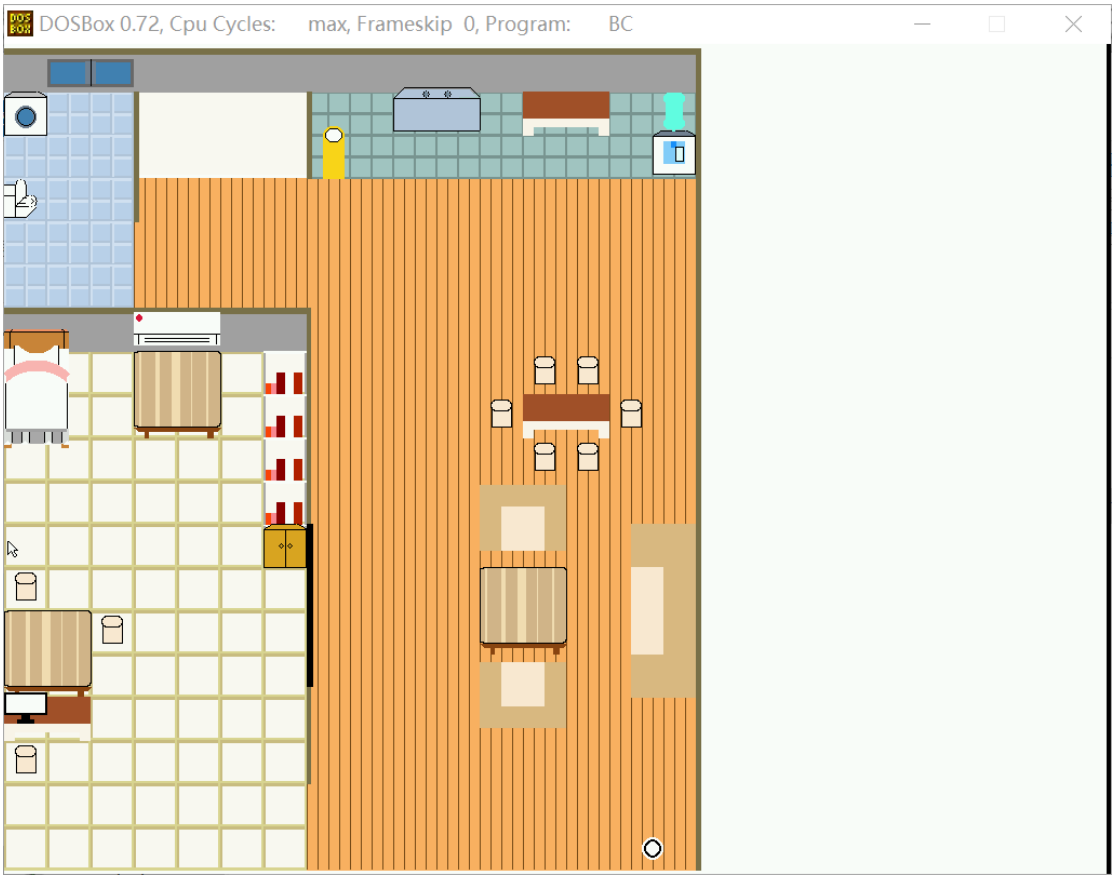


4）手机指纹验证界面

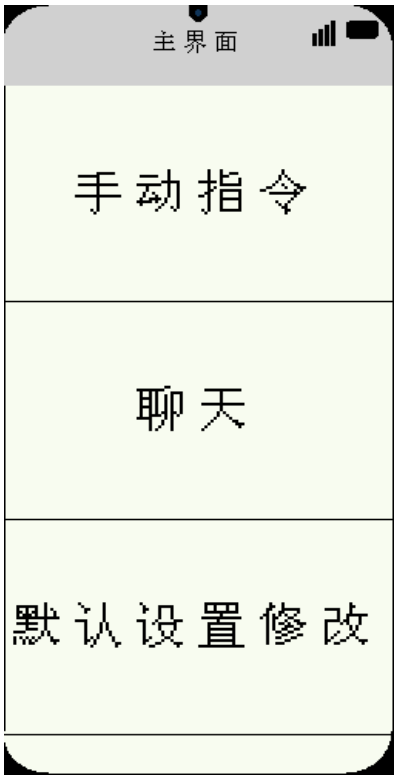
该手机样板的设计参考了华为 P30 和具有华为特色的前屏指纹解锁。



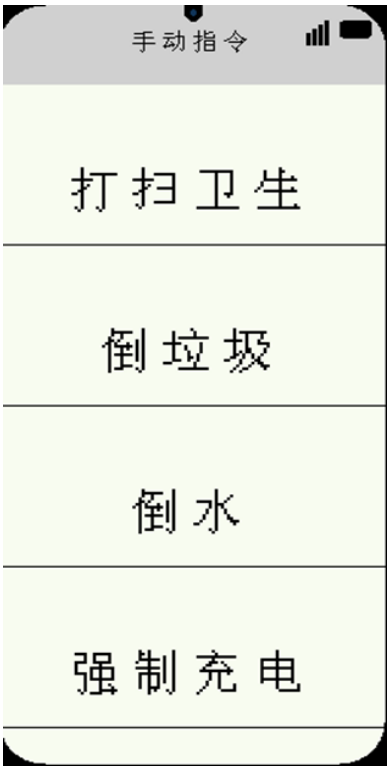
5）房间界面



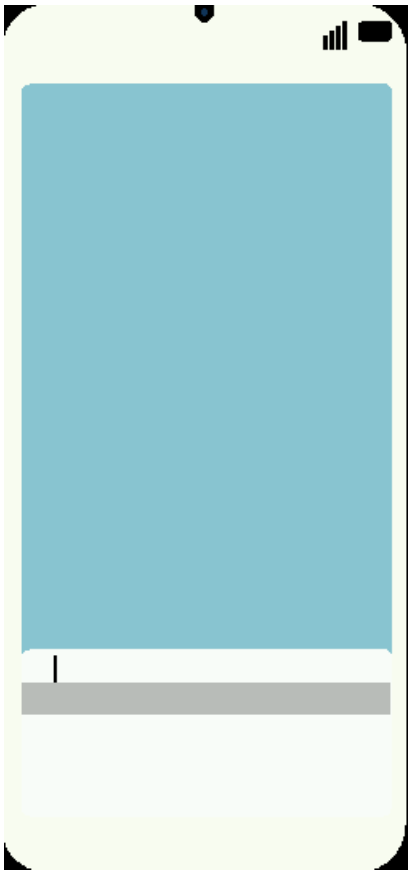
6）手机主界面



7）手动指令界面



8）聊天界面



9）默认设置修改界面

6. 时间安排

	任务	备注
暑假	需求分析与功能设计 欢迎、登录、注册模块 自动寻路实现 中文的基本实现 鼠标、键盘的交互的实现 所有界面（地图、手机、登陆注册界面）的设计	
第一周	时间线功能的实现	
第二周	时间线功能的调试 链表的优化	
第三周	手动功能的实现与人机交互 聊天功能的实现 登陆注册的优化 内存的优化	
第四周	集成测试与调试	中期验收
第五周	集成测试与调试	
第六周	集成测试与调试	
第七周	集成测试与调试	
第八周	集成测试与调试	最终验收

代码分配

源文件	行数	头文件	行数	合计	作者
advance.c	483	advance.h	35	518	外源
basicgf.c	264	basicgf.h	12	276	外源
bricwall.c	90	bricwall.h	22	112	袁立凡
chat.c	60	chat.h	10	70	肖力文
chatHanz.c	114	chatHanz.h	43	157	外源
chatInpu.c	395	chatInpu.h	63	458	外源
chatQhwh.c	215	chatQhwh.h	54	269	外源
chatShow.c	258	chatShow.h	30	288	肖力文
color1.c	113	color1.h	52	165	外源
findway.c	383	findway.h	42	425	袁立凡
finger.c	45	finger.h	10	55	肖力文
frnture.c	470	frnture.h	63	533	袁立凡 480，肖力文 54
hzxs.c	81	hzxs.h	7	88	外源
input.c	135	input.h	33	168	外源
iph_page.c	148	iph_page.h	26	174	肖力文
log_in.c	334	log_in.h	47	381	肖力文
MAIN_.c	60	MAIN_.h	10	70	肖力文
main2.c	267	main2.h	26	293	肖力文
manbody.c	216	manbody.h	24	240	袁立凡
module_b.c	182	module_b.h	22	204	外源
mouse.c	313	mouse.h	18	331	外源
myhouse.c	147	myhouse.h	14	161	袁立凡
rbtbody..c	282	rbtbody..h	28	310	袁立凡
rbtfunc.c	347	rbtfunc.h	12	359	袁立凡
rbtmove.c	259	rbtmove.h	12	271	袁立凡
register.c	905	register.h	117	1023	肖力文
sdzl.c	74	sdzl.h	15	89	肖力文
set_c.c	47	set_c.h	10	57	肖力文
SVGAHEAD.c	565	SVGAHEAD.h	81	644	外源
time_line.c	580	time_line.h	37	617	肖力文
txt_save.c	638	txt_save.h	145	783	肖力文 272，袁立凡 511
user_list.c	182	user_list.h	28	210	肖力文
WELCOME.c	130	WELCOME.h	26	156	肖力文
		typstrct.h	151	151	袁立凡 82，肖力文 69

肖力文 3815 行
袁立凡 2950 行
外源：3280 行

总结与感悟

袁立凡总结与感悟

C 课设是我第一次接触到大型项目的设计，让我学习了很多课外的编程知识，以及从需求分析开始的全套项目开发的流程，明白了团队的力量以及交流合作的重要性，也让我学会了遇到复杂问题需要自己多学、多问、多试，而不能坐以待毙，更让我在面对其他大型项目开发时更有信心。

在完成课设的过程中，需要完成大量的绘图、动画以及文件操作，这些都是课堂上受限于课时而没有学习的内容，所有东西都靠自学以及自己写程序尝试。尤其是我们的选题需要即时改变机器人的坐标并绘出相应的运动动画，这一功能困扰了我很久。而将文件操作运用于动画这一方法，大大节省了堆内存，为程序稳定性的提高起到了巨大的作用。我仍然清晰地记得在反复查阅资料和尝试以后，机器人开始可以持续、稳定运动时，自己的激动之情。正是这种不断产生的、小小的成就感，让我始终对课设饱含热情，暑假期间就整天沉迷于课设。这也让我明白，只要对所做之事充分投入、充满热情，就能把事情做得又快又好：正是暑假就全天候投入课设，才让我在暑假期间就完成了界面设计、自动寻路的数据结构与算法、机器人行走动画等保障程序功能实现的所有基本前提，完成了我负责的任务量的一半以上，再加上队友肖力文的巨大投入，我们小组在暑假完成了相当一部分的工作，让我们始终没有限于时间窘迫的困境中。

让我收获更大的是熟悉了大型项目开发的全套流程。由于在 C 语言课堂上，周老师就多次向我们强调需求分析的重要性，因此从 6 月 30 号选定题目后，我和肖力文就开始着手做需求分析，希望我们做出的家具机器人是真正符合市场需求的产品。在经过上网查资料，以及咨询长辈、朋友们对于家具机器人的看法后，最终确定了我们机器人的功能：安保（防贼防盗），便捷（日常生活服务），娱乐（提供娱乐选项），急救（生病或者意外时提供帮助，或帮忙拨打 120），我负责安保和急救部分。在完成这些功能后，我们将程序展示给同学时，听到他们评价说我们的机器人“很实用”的时候，欣慰之情也抹去了编写程序的艰辛之感。

团队交流的重要性以及小组全心投入所体现的力量也认我印象深刻。在暑假刚开始的几天，小组两个人各自都有一些事务，因此很多时候不能形成有效交流，有想法的时候无法得到及时的反馈，导致最初的几天进展极其缓慢。而之后两个人都全身心投入的时候，每次的交流都能得到及时有效的反馈，所有的想法都能经过充分的交流，开展进度大大加快。同时，我要特别感谢队友肖力文，在我开学后由于备战数模比赛而不能继续全天候投入完成课设时，他主动增加自己的任务量，使得进度没有陷入滞缓。

最后，感谢在我们需求分析时提出自己观点的长辈、朋友，感谢每一位认真解答我们问题的学长，感谢为我们的项目提出很多建设性意见的老师，没有他们的帮助，我们一定无法顺利完成此次课设。

这几个月收获的知识、对项目流程的熟悉以及对团队合作的理解一定会在日后的工作生活中给我带来巨大帮助，C 课设的经历也在我心里留下了不可磨灭的印记！

肖力文的总结与感悟

本次课设是我第一次接触软件的编写。在这次课设的完成过程中，我学到了很多东西，比如对于一款软件的需求分析，设计一个软件的高效流程，编写程序的规范，与队友的合作和在一次一次调 bug 时淬炼出来的耐心。这队友我以后的学习乃至未来工作都有着相当大

的意义。

首先我来说说我在编写软件方面的收获：

需求分析永远是第一位的。当时我和我的队友袁立凡看到了“家居机器人模拟系统”这个题目的时候，就眼前一亮，我们认为这会是一个很有趣的课题，并且有很多发挥的空间，这个题目的功能可以有很多创新的地方。在我们和学长们交流的过程中，不同的学长都多次给我们提到功能需求是第一位的，不要为了行数而去设计鸡肋的功能。于是我们在功能分析的时候上网查阅了现有的家居机器人网站上提供的功能介绍说明，也和家人朋友交流了他们理想的家居机器人的功能，最后定下了用户需求和功能：安保，便捷，娱乐，急救。并且在呈现形式上采用了时间线的方式，更加突出了我们的功能的实用性。这次实际设计软件的经历将会让我在以后的工科研，工作中更加重视用户的需求。

在程序开发的高效流程中，在完成需求分析后就是系统分析：总体设计和详细设计。在暑假的时候我和我的队友在完成了需求分析后就完成了程序的总体功能和逻辑的设计。但是我在编写的初期缺乏详细设计，一般就是拿着一个功能，啥也不想就开始编写，一边编写一边思考程序逻辑和程序要用到的变量数据等。这样轻则导致程序逻辑错误，重新编写，费时费力，重则导致程序逻辑混乱，出现 bug 时难以调试。所以在编程的前中期，我就先对要编写的功能的模块进行分割，每个模块要用到的变量和逻辑提前在纸上写清楚，然后才开始编写程序，这样编出来的程序不仅逻辑清晰，空间节省，而且编写程序效率高。所以周院长 mooc 里说的“系统分析不仅要包括总体设计还要包括详细设计”确实是真知灼见啊。

编写程序要有注释的好习惯，这是编程规范里很重要的一步。在刚开始编程的时候，我很是觉得新鲜，一心只想编写程序实现功能，而忘记了给程序写注释。不给程序写注释使得我在一周后再来看一周前的程序时，里面的具体逻辑就有点想不清楚了。特别的，在我和我的队友合代码的时候，在理解代码上出现了困难。所以我在接下来编程的过程中，尽量在重要节点写上详细的注释，比如链表的具体结构，重要变量的使用注意，多返回值函数的不同返回的功能。并且，在最后规范代码的时候，我尽量做到每个变量的定义都有说明，每个判断语句都有说明，尽量做到行行有代码，特别是重复性少的代码。我的 C++老师黎云教授也多次在我们面前强调注释的重要性。在此，我表达对黎云老师的感谢。这次的课设经历让我知道在以后的科研，工作中编写程序的时候，要常加注释，加强程序的可读性。

在编写 C 课设的过程中，我的耐心得到了极大的淬炼。我一直有急于求成的毛病，但是在 C 课设面前，越是急于求成，越是完成不了功能。特别是在最后集成调试的过程中，出了一个 bug，一个一万多行的程序要调试找错的难度可想而知，而且有的时候还是强退，卡掉，乱码这样的 bug，有时改了一个 bug，就是因为之前的改动，又来了一个 bug。在调试 bug 的过程中让我明白了焦虑是人的本能，但是沉下心来，找好方法，认真做事才是解决问题的良方。

在暑假的时候我和我的队友袁立凡就差不多完成了百分之四十的工作，包括：需求分析的完成，系统总体设计的完成，登陆注册的完成，中文输入输出的完成，用迪杰斯特拉算法实现自动寻路的功能的完成。所以在开学之后，我们并没有因为 C 课设而感到很窘迫。在此，我对我的队友袁立凡表达感谢，他的专业知识和耐心极大的促进了课设的完成。

我要对帮助过我们的学长表达感谢，余泽泰，刘心阳，兰梓皓等学长没有嫌我们烦，耐心地拨冗为我们解答问题，要是没有他们，我们的 C 课设不会有如今的成果。

我还要对周纯杰院长和其他所以 C 课设老师表达感谢。是周院长上学期 C 语言课程的扎实培养给我的 C 课设完成奠定了牢固的基石。同时，我也要感谢其他在机房默默为我们答疑的老师们，是他们解决了我们让我们寝食难安的程序问题。

这次课设不仅让我明白了很多编写程序，设计软件的经验和知识，让我明白了合作的重要性，还让我有更多的耐心来处理大的困难。

代码

1.头文件

----------advance.h----------

```
#ifndef advancegf.h
#define advancegf.h

extern void triangledown(int x,int y,int height,int color);

extern void triangleright(int x,int y,int height,int color);

extern void triangleleft(int x,int y,int height,int color);

/************************************************
function:            Fillellipse

description:        填充椭圆（仅限竖直方向）

Input:                x1,y1(上方圆心坐标),x2,y2(下方圆心坐标),r(两圆半径),color;

out:                  一个由上下两个填充圆形和一个矩形连接而成的椭圆
************************************************************/
extern void Fillellipse(int x1,int y1,int x2,int y2,int r,int color);

/************************************************
function:            ever_Fillellipse

description:        填充椭圆（水平方向）

Input:                x1,y1(左边圆心坐标),x2,y2(右边圆心坐标),r(两圆半径),color;

out:                  一个由左右两个填充圆形和一个矩形连接而成的椭圆
************************************************************/
extern void ever_Fillellipse(int x1,int y1,int x2,int y2,int r,int color);

/************************************************
function:            ellipse

description:        椭圆（仅限竖直方向）

Input:                x1,y1(上方圆心坐标),x2,y2(下方圆心坐标),r(两圆半径),color;
```

```
out:                    由上下两个半圆和两条直线连接而成的椭圆

quote:                  semicircle_up,semicircle_down;
************************************************************/
extern void ellipse(int x1,int y1,int x2,int y2,int r,int color);

/************************************************
function:           semicircle_up

description:        上半圆（非填充）

Input:              x,y,r,color;

out:                一个上半圆（非实心）
************************************************************/
extern void semicircle_up(int x0,int y0,int r,int color);

/************************************************
function:           semicircle_down

description:        下半圆（非填充）

Input:              x,y,r,color;

out:                一个下半圆（非实心）
************************************************************/
extern void semicircle_down(int x0,int y0,int r,int color);



/************************************************
function:           fill_bow_right_up

description:        右上填充扇形

Input:              x,y,r,color;

out:                右上角 1/4 扇形
************************************************************/
extern void fill_bow_right_up(int x,int y,int r,int color);

/************************************************
function:           fill_bow_left_up
```

```
description:        左上填充扇形

Input:              x,y,r,color;

out:                左上角 1/4 扇形
*************************************************************/
extern void fill_bow_left_up(int x,int y,int r,int color);

/************************************************
function:           fill_bow_left_down

description:        左下填充扇形

Input:              x,y,r,color;

out:                左下角 1/4 扇形
*************************************************************/
extern void fill_bow_left_down(int x,int y,int r,int color);

/************************************************
function:           fill_bow_right_down

description:        右下填充扇形

Input:              x,y,r,color;

out:                右下角 1/4 扇形
*************************************************************/
extern void fill_bow_right_down(int x,int y,int r,int color);

/************************************************
function:           fill_bow_down

description:        下填充扇形

Input:              x,y,r,color;

out:                下 1/4 扇形
*************************************************************/
extern void fill_bow_down(int x,int y,int r, int color);

/************************************************
function:           fill_bow_up
```

description:　　上填充扇形

Input:　　x,y,r,color;

out:　　上 1/4 扇形
**********/
extern void fill_bow_up(int x,int y,int r, int color);

/**********
function:　　fill_bow_left

description:　　左填充扇形

Input:　　x,y,r,color;

out:　　左 1/4 扇形
**********/
extern void fill_bow_left(int x,int y, int r, int color);

/**********
function:　　fill_bow_right

description:　　右填充扇形

Input:　　x,y,r,color;

out:　　右 1/4 扇形
**********/
extern void fill_bow_right(int x,int y,int r,int color);

/**********
function:　　bar_round

description:　　圆角矩形

Input:　　x,y(矩形中心坐标),length(矩形长度),height(矩形高度),r(圆角半径),thick(线段粗,目前只能改变直线段粗细-09/25),color;

out:　　圆角半径可变，线粗可变的圆角矩形

quote:
fill_bow_right_down,fill_bow_right_up,fill_bow_left_down,fill_bow_left_up,bow_right_up,bow_right_down,bow_left_up,bow_left_down;

```
************************************************************/
extern void bar_round(int x,int y,int length,int height,int r,int thick,int color);

/************************************************
function:            bar_round_2

description:        圆角矩形(无边框)

Input:                x0,y0(左上角坐标),x1,y1(右下角坐标),r(圆角半径),thick(线段粗),color;

out:                  无边框圆角矩形

quote:
fill_bow_right_down,fill_bow_right_up,fill_bow_left_down,fill_bow_left_up,bow_right_up,bow_right_down,bow_left_up,bow_left_down;
************************************************************/
extern void bar_round_2(int x0,int y0,int x1,int y1,int r,int thick,int color);

/************************************************
function:            bar_round_with_shadow

description:        带阴影效果的圆角矩形

Input:                x,y(矩形中心坐标),length(矩形长度),height(矩形高度),r(圆角半径),thick(线段粗,目前只能改变直线段粗细-09/25),color;

out:                  圆角半径可变，线粗可变的带阴影（阴影大小与线粗有关）圆角矩形

quote:
fill_bow_right_down,fill_bow_right_up,fill_bow_left_down,fill_bow_left_up,bow_right_up,bow_right_down,bow_left_up,bow_left_down;
************************************************************/
extern void bar_round_with_shadow(int x,int y,int length,int height,int r,int thick,int color);

/************************************************
function:            bow_right_down

description:        右下扇形

Input:                x,y,r,color;

out:                  右下 1/4 扇形（非填充）
************************************************************/
extern void bow_right_down(int x,int y,int r,int color);
```

```
/*************************************************
function:            bow_left_down

description:       左下扇形

Input:               x,y,r,color;

out:                  左下 1/4 扇形（非填充）
*************************************************************/
extern void bow_left_down(int x,int y,int r,int color);

/*************************************************
function:            bow_right_up

description:       右上扇形

Input:               x,y,r,color;

out:                  右上 1/4 扇形（非填充）
*************************************************************/
extern void bow_right_up(int x,int y,int r,int color);

/*************************************************
function:            bow_left_up

description:       左上扇形

Input:               x,y,r,color;

out:                  左上 1/4 扇形（非填充）
*************************************************************/
extern void bow_left_up(int x,int y,int r,int color);

/*************************************************
function:            lean_line

description:       倾斜直线

Input:               x,y(直线起点坐标),length(线长),theta,color;

out:                  倾斜直线

quote:
```

```
************************************************************/
extern void lean_line(int x,int y,int length,int theta,int color);

/************************************************
function:            lean_line_thick

description:       线粗可变的倾斜直线

Input:               x,y(直线起点坐标),length(线长),theta,thick,color;

out:                  线粗可变的倾斜直线

quote:               lean_line
************************************************************/
extern void lean_line_thick(int x,int y,int length,int theta,int thick,int color);

/************************************************
function:            theta_bar

description:       倾斜矩形(实质是用倾斜直线平移形成的平行四边形),但目前在 45-90 度区域输出不连续-09/25

Input:               x,y(矩形左上角点坐标),length(矩形边长),height(矩形高度),theta,color;

out:                  倾斜矩形

quote:               lean_line_thick
************************************************************/
extern void theta_bar(int x,int y,int length,int height,int theta,int color);


extern void robot_hand_left(int x,int y,int theta);

/************************************************
function:            red_cross

description:       一个红色的叉

Input:               x,y

out:                  图形

quote:               theta_bar
************************************************************/
```

```
extern void red_cross(int x,int y);

/************************************************
function:            green_tick

description:       一个绿色的勾

lnput:               x,y

out:                  图形

quote:               theta_bar
*************************************************************/
extern void green_tick(int x,int y);

/************************************************
function:            triangle1

description:       三角形(较瘦长)，为画树开发

lnput:               x,y,height,color

out:                  图形

quote:
*************************************************************/
extern void triangle1(int x,int y,int height,int color);

/************************************************
function:            triangle1

description:       三角形(较矮胖)

lnput:               x,y,height,color

out:                  图形

quote:
*************************************************************/
extern void triangle2(int x,int y,int height,int color);

/************************************************
function:            button
```

description:　　开关(ios 风格)

Input:　　x,y,judge(1 为开，0 为关)

out:　　图形

quote:　　ever_Fillellipse,FillCircle
**********/
extern void button(int x,int y,int judge);

#endif

——————————————————basicgf.h——————————————————————————
#ifndef basicgf.h
#define basicgf.h
/**********
功能说明：画水平线函数
参数说明：x0,y0 起始坐标　x1,y1 终止坐标　thick 厚度　color 颜色
无返回值
**********/
extern void linelevel(int x0,int y0,int x1,int y1,int thick,int color);

/**********/

/**********
功能说明：画竖线函数
参数说明：x0,y0 起始坐标　x1,y1 终止坐标　thick 厚度　color 颜色
无返回值
**********/
extern void linever(int x0,int y0,int x1,int y1,int thick,int color);

/**********/

/**********
Function:　　Horizline

Description：画水平线函数
可以接收超出屏幕范围的数据，只画出在屏幕内部分

因为没有防止整型变量溢出的判断，画超出屏幕的线时应防止输入特大数据

Calls:　　Selectpage

Called By:　　Line
Bar
Circlefill

Input:　　int x　　起始点横坐标，从左到右增加，0 为最小值（屏幕参考系）
int y　　起始点纵坐标，从上到下增加，0 为最小值（屏幕参考系）
int width　　水平长度，为正向右延伸，为负向左延伸
unsigned char color　　颜色数，共有 256 种

Output:　　屏幕上画出水平线

Return:　　None
Others:　　None
**********/
extern void Horizline(int x, int y, int width, int color);

/**********
功能说明：画水平矩形函数
参数说明: x0,y0 左上角坐标 x1,y1 右下角坐标 color 颜色
**********/
extern void bar(int x0, int y0, int x1, int y1, int color);

/**********
功能说明：以(x,y)点为圆心，以 radius 为半径画圆，没有防止超出屏幕的判断，
可以将没有超出的部分画出。
参数说明：x,y 为圆点，radius 为半径，color 为颜色。
无返回值：
**********/
extern void circle(int x0,int y0,int radius,int color);

/**********
function:　　bow

description:　　此函数是专为机器人正面嘴巴的绘制设计，生成一个弧度为 pi/6 的下圆弧

Input:　　x0,y0,r,color;

out:　　弧度 pi/6 的下圆弧

```
*************************************************************/
extern void bow(int x0,int y0,int r,int color);


/*****************************************************************************
        功能说明：画实心圆
        参数说明：x,y 为圆心
       无返回值
*****************************************************************************/
extern void FillCircle(/*int x, int y, int r, int color*/int xc, int yc, int radius, int color);

extern void eqver_tri(int x, int y, int length, int color);
 //等腰直角三角形，length 是直角边长度
    //方向是固定的，入口参数是直角点的坐标



extern void delay0(int time);
#endif
```

----------bricks.h----------

```
#ifndef bricwall.h
#define bricwall.h

///地板砖
extern void wood_ver(int x, int y);//木质条纹地板

extern void glass(int x, int y);//浴室地砖

void green_kitchen(int x, int y);//厨房地砖

void green_bedroom(int x, int y);//卧室地砖

///   各种墙
void w_blue(int x, int y);   //正面看上去最大的那堵大墙

void w_right(int x,int y);//在指定 x，y 代表的格子的右侧画墙壁

void w_left(int x, int y);//;    //在指定 x，y 代表的格子的左侧画墙壁

void w_down(int x, int y);    //在指定 x，y 代表的格子的下面画墙壁
```

#endif

----------chat.h----------

```
#ifndef chat_main.h
#define chat_main.h

#include "typstrct.h"

//功能：画聊天界面框架
//输入：无
//输出：无
extern void chat_interface();



//功能：聊天功能主函数
//输入；机器人的状态变量
//输出：无
extern void chatmain(CASE *robot);
#endif
```

----------chatHanz.h----------

```
#ifndef chatHanzi.h
#define chatHanzi.h
#include "typstrct.h"

/************************************************
function:       extern int ShowChinese(int *qhwh,int num_chinese,int color,Area show_area,int size,char *hzk_dir);

description:    打印汉字

Input:          int *qhwh,int num_chinese,int color,Area show_area,int size,char *hzk_dir

output:
************************************************************/
extern int ShowChinese(int *qhwh,int num_chinese,int color,Area show_area,int size,char *hzk_dir);

/************************************************
function:       extern int ShowPerCharacter(int qhwh,Area show_area,Coordinate * current_position,int color,int size,char *hzk_dir);
```

```
description:        每个汉字打印

Input:              int qhwh,Area show_area,Coordinate * current_position,int color,int
size,char *hzk_dir

output:
*****************************************************************/
extern int ShowPerCharacter(int qhwh,Area show_area,Coordinate * current_position,int
color,int size,char *hzk_dir);

/*************************************************
function:           extern int GetBit(int num,char ch);

description:        每位的像素点 get

Input:              int num,char ch

output:
*****************************************************************/
extern int GetBit(int num,char ch);

/*************************************************
function:           extern void CheckArea(Area position,Coordinate *current_position,int size);

description:        检验是否在聊天区域内，是否需要换行

Input:              Area position,Coordinate *current_position,int size

output:
*****************************************************************/
extern void CheckArea(Area position,Coordinate *current_position,int size);



#endif

————————————————————————————chatInpu.h————————————————————————————
#ifndef chatInput.h
#define chatInput.h
#include "typstrct.h"

/*************************************************
function:           extern void CursorWhite(Coordinate current_show_position,int show_size);
```

description:　　白色光标（盖住黑色光标）

Input:　　Coordinate current_show_position,int show_size

output:
**********/
extern void CursorWhite(Coordinate current_show_position,int show_size);

/**********
function:　　extern void CursorBlack(Coordinate current_show_position,int show_size);

description:　　黑色光标，短暂停留

Input:　　Coordinate current_show_position,int show_size

output:
**********/
extern void CursorBlack(Coordinate current_show_position,int show_size);

/**********
function:　　extern void DeleteShow(Area show_area,Coordinate * current_show_position,int show_size);

description:　　删掉汉字

Input:　　Area show_area,Coordinate * current_show_position,int show_size

output:
**********/
extern void DeleteShow(Area show_area,Coordinate * current_show_position,int show_size);

/**********
function:　　extern void ShowWhite(Coordinate * current_show_position,int show_size);

description:　　盖住删掉汉字的界面

Input:　　Coordinate * current_show_position,int show_size

output:
**********/
extern void ShowWhite(Coordinate * current_show_position,int show_size);

/**********

function: extern void DeleteTab(Coordinate * current_en_position,CH* ch,EN* en,char *temp ,int * num_tab,int *ch_qhwh);

description: 在输入框删拼音

Input: Coordinate * current_en_position,CH* ch,EN* en,char *temp ,int * num_tab,int *ch_qhwh

output:
***/
extern void DeleteTab(Coordinate * current_en_position,CH* ch,EN* en,char *temp ,int * num_tab,int *ch_qhwh);

/***
function: extern void ShowTxt(int *qhwh, Area show_area,Coordinate* current_show_position,int show_size);

description: 打印整段话

Input: int *qhwh, Area show_area,Coordinate* current_show_position,int show_size

output:
***/
extern void ShowTxt(int *qhwh, Area show_area,Coordinate* current_show_position,int show_size);

/***
function: extern int ShowChTab(Area show_area,Coordinate current_show_position, int show_size,int *qhwh);

description: 汉字输入法主逻辑

Input: Area show_area,Coordinate current_show_position, int show_size,int *qhwh

output:
***/
extern int ShowChTab(Area show_area,Coordinate current_show_position/*现在正在码的字的左上角坐标*/, int show_size/*有 48，36，28*/,int *qhwh);

#endif

----------chatQhwh.h----------

```
#ifndef    chatQhwh.h
#define    chatQhwh.h
#include "typstrct.h"

/*************************************************
function:              extern void SaveChQhwh(FILE *fp,CH * ch);

description:         将汉字区号位号保存在文件中

Input:                  FILE *fp,CH * ch

output:
*************************************************************/
extern void SaveChQhwh(FILE *fp,CH * ch);

/*************************************************
function:              extern int FindChQhwh(CH *ch,char *temp,int num_qhwh,int qhwh);

description:         传进读取文件的拼音库，从中找到对应汉字的区位号

Input:                  CH *ch,char *temp,int num_qhwh,int qhwh

output:
*************************************************************/
extern int FindChQhwh(CH *ch,char *temp,int num_qhwh,int qhwh);


/*************************************************
function:              extern int CheckQhwhNum(int *qhwh);

description:         看区位号数量（确认有几个字）

Input:                  int *qhwh

output:                 int  个数
*************************************************************/
extern int CheckQhwhNum(int *qhwh);

/*************************************************
function:              extern void qhwh2incode(int qwh,char *incode);

description:         将区号位号转化为内码，方便之后对比
```

```
Input:                int qwh,char *incode

output:
************************************************************/
extern void qhwh2incode(int qwh,char *incode);



extern void SaveEnQhwh(FILE *fp,EN *en);
extern int FindEnQhwh(EN *en,char temp);
extern int TxtToQhwh(int *qhwh,char *filename);
extern int AddQhwhToTxt(int *qhwh,char *filename);
extern int AddNumToTxt(int num,char *filename);
extern int CheckNumInTxt(char *filename);
extern int ReplaceQhwhToTxt(int *qhwh,char *filename);
extern int ReplacePassToTxt(int *qhwh,char *filename);

extern void ClearKey(void);




extern int QhwhToZero(int *qhwh);


#endif
```

----------chatShow.h----------

```
#ifndef chatShow.h
#define chatShow.h
#include "typstrct.h"


//功能：打印用户命令
//输入：区号位号和输出框的纵坐标
//输出：无
extern void show_order(int *qhwh,int *y);
```

```
//功能：得到用户输入的字符串（将区号位号改成内码）
//输入：区号位号指针和内码指针
//输出：无
extern void get_str(int *qhwh, char *incode);

//功能：根据输入的字符串得到返回语句
//输入：用户输入的字符串的指针，回复的字符串的指针，机器人的状态指针
//输出：无
extern void reply_match(char * str,char *reply,CASE* robot);

//功能：机器人回复输出
//输入：输出语句的内码，机器人的状态
//输出：无
extern void show_reply(char *incode,CASE* robot);

//功能：聊天输出的主函数
//输入：用户输入的字符串的内码，机器人状态
//输出：无
extern void show_main(int *qhwh,CASE* robot);

//功能：判断用户输入的句子中是否有关键字
//输入：用户输入的字符串的字符指针
//输出：int
//          返回 0：没有关键字
//          返回 1：有关键字
extern int no_keyword(char *str);
```

#endif // chatShow

----------color1.h----------

#include "typstrct.h"
#ifndef color1.h
#define color1.h

/***
function: transcolor

description: 颜色转换

Input: rgb

out: 64k 颜色值

quote:
***/
extern int transcolor(int r,int g,int b);

/***
function: linelevel_color

description: 水平线性渐变

Input: x1,y1（左上角坐标）,x2,y2（右下角坐标）,r1,g1,b1（出发颜色）,r2,g2,b2（结束颜色）

out:

quote:
***/
extern void linelevel_color(int x1,int y1,int x2,int y2,int r1,int g1,int b1,int r2,int g2,int b2);

/***
function: linever_color

description: 竖直线性渐变

Input: x1,y1（左上角坐标）,x2,y2（右下角坐标）,r1,g1,b1（出发颜色）,r2,g2,b2（结束颜色）

out:

quote:
***/
extern void linever_color(int x1,int y1,int x2,int y2,int r1,int g1,int b1,int r2,int g2,int b2);

/***
function: circle_color

description: 轴向渐变

Input: x0,y0（圆心坐标）,r1,g1,b1（出发颜色）,r2,g2,b2（结束颜色）,r（渐变圆半径）

out:

quote:
***/
extern void circle_color(int x0,int y0,int r1,int g1,int b1,int r2,int g2,int b2,int r);

#endif

----------findway.h----------

```
#ifndef findway.h
#define findway.h
/*队与图的一些基本操作，用于寻路。
由于用得不多，因此只写 队的初始化、入队、出队、判断队空，
图的初始化和第一邻接点函数
补了栈的操作，因为要把 path 路径正向找出*/

#include "typstrct.h"
/******采用链队列，因为不知道要装多少个点*****/

void InitQueue(LinkQueue *Q);    //传入队指针完成队的初始化操作，即：先申请一个队节点，并让队头、队尾指向该节点

void DestroyQueue(LinkQueue *Q);    //传入队指针销毁队，即：将元素逐个出队后释放节点空间，直到队空

int IsEmpty(LinkQueue Q);    //根据队头、队尾是否指向同一节点，判断队是否为空

void EnQueue(LinkQueue *Q, QElemtype e); //将数据 e 储存，即入队，申请一个队节点，存
```

放元素 e 信息后，插在队尾

void DeQueue(LinkQueue *Q, QElemtype *e);　//队头元素出队，传入指针 e 保存队头元素的信息，然后释放队头节点

///图
//入口是数组 G，在 main 函数中声明，将给定地图中每个可通行的坐标点放进数组中。结构体数组里有一个指针，依次、按方向顺次指向其可通行的邻接点
int CreateGraph(Graph G);

//入口参数：索引 k 和 adjvex[4]数组，数组各分量依次储存 G[k]右、左、上、下四个方向的可通行邻接点在 G 中的坐标
//adjvex 中第一个分量是向右的邻接点，第二个向左，以此类推
int FindAdjVex(Graph const　G, const int n, int k, QElemtype *adjvex);

int LocateVex(Graph const　G, const int n, Axis V);　　//在图 G 中找到 V 点（实际上就是 G 中数据域同 V 的点）

int FindWay(Graph const　G, PathType *path, const int n, Axis V0, Axis Vt);　//根据 Dijkstra 算法得到逆序最短路径 path

//栈
void InitStack(LkStack *S);　//初始化栈

void DestroyStack(LkStack *S);//销毁栈

void Push(LkStack *S, SElemtype e);//入栈

void Pop(LkStack *S, SElemtype *e);//出栈

#endif // findway

----------finger.h----------

#ifndef fingercheck.h
#define fingercheck.h
#include "typstrct.h"

//功能：指纹验证函数
//输入：无
//输出：int
//　　返回 1：验证成功（验证成功的按键是 enter 键）
extern int finger_check(void);

```
#endif // fingercheck
```

----------frnture.h----------

```
#include "typstrct.h"
#ifndef furniture.h
#define furniture.h

extern void trashbin(int x,int y);          //垃圾桶，入口参数是垃圾桶的左上角

extern void bed(int x,int y);              //床

extern void window_close(int x,int y);          //关着的窗户

extern void cupboard(int x,int y);          //衣柜

void aircon(int x,int y,int open);  //80*40,open==1 为开启，空调

void WashMach(int x,int y);  //40*40，洗衣机

void bookshelf(int x,int y);   //40*80，书架

void desk(int x,int y);    //80*40，小方桌

void seat(int x,int y);    //40*40,控制在 12 至 28 之间，小板凳

void trash1(int x,int y);  //40*40，纸张

void trash2(int x,int y);  //40*40,菜叶

void trash3(int x,int y);  //40*40,果核

void pc(int x,int y);       //电脑，要和上面那个桌子组合用

void TV(int x,int y);       //15*125

//木条纹方桌
void rect_table(int x,int y);   //80*80

extern void sofa_main(int x, int y);              //长条沙发

extern void sofa_up(int x, int y);                //沙发（短，上面的）
```

```
extern void sofa_down(int x, int y);                //沙发（短，下面的）

void toilet(int x, int y);                  //马桶

extern void water_dispenser(int x, int y);    //饮水机，40*80

extern void zaotai(int x, int y);             //灶台

extern void water_bottle(int x, int y);         //水杯

extern void clothes(int x, int y);    //衣服

extern void plate(int x,int y);    //盘子

void medical_kit(int x, int y); //医疗包

extern void TV_on(void);               //电视打开

extern void TV_off(void);             //电视关闭

extern void music_on(int x, int y);         //音符

extern void music_off( int x, int y);     //去除音符

#endif
```

----------hzxs.h----------

```
#ifndef _hzxs_h_
#define _hzxs_h_

//void    openhzk();
void gethz(char incode[],char    *bytes);
void dishz(int x0,int y0,int mx,int my,char *code,int color);
void fdhz(int x,int y,int mx,int my,char *s,int color);


#endif
```

----------input.h----------

```
#include "typstrct.h"
#ifndef input.h
#define input.h
```

```
/************************************************
function:          put_English

description :      在指定的地方输出英文

Input :            x1,y1 输出位置坐标，ascii 为该英文的 ASCII 码

out :              在指定位置输出英文字母

************************************************************/
extern void putEnglish(int x1,int y1,int ascii,int mx,int my,int color);
/************************************************
function:          outtextxy

description :      输出整个字符串

Input :            x,y（输出位置）,c（要输出的字符串）,mx,my（字母尺寸，横向/纵向）,mar（字符之间间距）,color

out :              图形模式下输出英文及数字

************************************************************/
extern void outtextxy(int x,int y,char *c,int mx,int my,int mar,int color);
/************************************************
Function: searchKeyValue
Description: 根据键值返回表中其对应字符
Calls:
Return: 若有则返回对应字符；若表中无此键值，则返回'\0'
************************************************/
extern char searchKeyValue(int value);
#endif
```

----------iph_page.h----------

```
#ifndef iph_page.h
#define iph_page.h
#include "typstrct.h"

//功能：手机外框等固定元素，任何界面都需要显示
extern void iph_frame();

//功能：手机界面附加元素
extern void iph_frame_plus();

//功能：手机主界面
```

```
extern void iph_page_1();

//功能：手动指令界面
extern void iph_page_2();

//功能：聊天界面
extern void iph_page_3(unsigned int *box_save);

//功能：默认设置修改界面
extern void iph_page_4(char *at,char *bt);

//功能：时间界面
extern void time_page();

#endif // iph_page
```

----------log_in.h----------

```
#ifndef log_in.h
#define log_in.h


#include "typstrct.h"

//功能：登录注册功能主逻辑
//输入：无
//输出：int 型
//          返回 1 为登陆成功，跳出循环，进入下一模块
//          返回 0 为登陆失败，继续循环
extern int enter(void);



//功能：登录的静止界面
//输入：无
//输出：无
extern void log_in_page(void);



//功能：账号输入函数
//输入：用户信息链表的头节点，指向账号字符串的字符指针，指向密码字符串的字符指针，
鼠标坐标指针
//输出：int 型
//          返回 1：登录或注册成功
```

```
//          返回 3：进入密码输入函数
extern int input_account(USERS *head,char *account,char *code,int *x,int *y);



//功能：账号输入函数
//输入: 用户信息链表的头节点, 指向账号字符串的字符指针, 指向密码字符串的字符指针, 鼠标坐标指针
//输出：int 型
//          返回 1：登录或注册成功
//          返回 2：进入 id 输入函数
extern int input_code(USERS *head,char *account,char *code,int *x,int *y);



//功能：验证账号密码是否正确
//输入：用户信息链表的头节点，指向账字符串的字符指针，指向密码字符串的字符指针
//输出：int 型
//          返回 1：验证成功
//          返回 5：验证失败
extern int log_in_check(USERS *head,char *account,char *code);
#endif // log_in
```

----------main_.h----------

```
#ifndef main_.h
#define main_.h

#include "typstrct.h"

//功能：主函数
extern void main();

#endif // main_
```

----------main2.h----------

```
#ifndef main_process.h
#define main_process.h
#include "typstrct.h"

//功能：实现功能的主进程
//输入：鼠标的坐标和按键状态
```

```
//输出：无
extern void mainprocess(int *x,int *y, int *pbutton);



//功能：将一位数字的分钟数调整成 2 位数字
//输入：分钟的字符串的首地址
//输出：无
extern void minute_adjust(char *s_minute);



//功能：点击“下一事件”后根据事件来调整时间
//输入：时间，事件，和事件是否发生变量的地址
//输出：无
extern void time_adjust_plus(int *time, int *times, int *wht_happen);

#endif // main_process
```

----------manbody.h----------

```
#ifndef manbody.h
#define manbody.h

#include "typstrct.h"

extern void paint_man(CASE case_state, int identity);     //第一次画人的时候调用，会画出人的正面

void man_forebody(CASE case_state, int identity);     //人的正面，包括主人与陌生人

void man_backbody(CASE case_state);                        //人的背面

void man_leftbody(CASE case_state);                       //人的右面

void man_rightbody(CASE case_state);                       //人的左面

void man_sleep(CASE case_state);                           //睡觉时的人像

void man_getup(CASE case_state);                           //起床

void sit_1(CASE case_state);                              //背面坐姿，工作和吃饭时调用
```

```
void sit_2(CASE case_state);                                    //侧面坐姿，娱乐时调用

#endif
```

----------module.h----------

```
#include "typstrct.h"
#ifndef module_box.h
#define module_box.h

/*************************************************
function:            overflow_box

description:        堆分配内存失败提示框

Input:               x,y(坐标)

out:                 提示框

quote:               advancegf.h,basicgf.h 中相关画图函数,hzxs.h,input.h 中文字函数
*************************************************************/
extern void overflow_box(int x,int y);

/*************************************************
function:            phone_module

description:        手机的功能外框

Input:               x,y(坐标)

out:

quote:               advancegf.h
*************************************************************/
extern void phone_module(int x,int y);

/*************************************************
function:            phone_back

description:        iphone 的 home 键

Input:               x,y(坐标)

out:
```

quote:　　advancegf.h,basicgf.h 中相关画图函数
**********/
extern void phone_back(int x,int y);

/**********
function:　　null_box

description:　　寻找文件失败提示框

Input:　　x,y(坐标)

out:　　提示框

quote:　　advancegf.h,basicgf.h 中相关画图函数,hzxs.h,input.h 中文字函数
**********/
extern void null_box(int x,int y);

extern void gg_bar();

/**********
function:　　FindWay_error

description:　　寻路失败提示框

Input:　　x,y(坐标)

out:　　提示框

quote:　　advancegf.h,basicgf.h 中相关画图函数,hzxs.h,input.h 中文字函数
**********/
extern void FindWay_error(int x,int y);

/**********
function:　　space_box

description:　　显示堆区剩余内存空间（调试）

Input:　　x,y(坐标)

out:　　提示框

quote:　　coreleft(alloc.h),advancegf.h,basicgf.h 中相关画图函数,hzxs.h,input.h 中文字函数

```
************************************************************/
extern void space_box(int x,int y);

/************************************************
function:            trap

description:       机器人行走失败提示框(周围有障碍物)

Input:               x,y(坐标)

out:                 提示框

quote:               advancegf.h,basicgf.h 中相关画图函数,hzxs.h,input.h 中文字函数
************************************************************/
extern void trap(int x,int y);

/************************************************
function:            pop_error

description:       栈弹出元素失败提示框(栈已空)

Input:               x,y(坐标)

out:                 提示框

quote:               advancegf.h,basicgf.h 中相关画图函数,hzxs.h,input.h 中文字函数
************************************************************/
extern void pop_error(int x,int y);

#endif
```

----------mouse.h----------

```
#ifndef _nmouse_H
#define _nmouse_H
#include "typstrct.h"


extern void cursor(int x, int y);
extern void setMouseShape(int mark, int mx, int my);
extern void mousehide(int x, int y);
extern void getMousebk(int x, int y);
extern int init();
extern void mouseInit(int *mx, int *my, int *mbutt);
extern int readxy(int *mx, int *my, int *mbutt);
```

```
extern void newxy(int *mx, int *my, int *mbutt);
extern void backgroundChange(int mx, int my, int x1, int y1, int x2, int y2);
extern void AddFrame(int mx, int my, int x1, int y1, int x2, int y2, int thick, int color);
extern void reset_mouse(int *x,int *y);
extern int esc_check(BUTTONS *esc1,int *x,int *y,int *button);

extern void esc_init(BUTTONS *esc1);

extern void CheckHeap(int i);
#endif
```

----------myhouse.h----------

```
#ifndef myhouse.h
#define myhouse.h

#include "typstrct.h"

extern void paint_house();//调用上述三个封装好的函数，画出房子界面

extern void paint_floor();//画地板

extern void paint_wall();//画墙壁

extern void paint_furniture();//画家具

#endif
```

----------rbtbody.h----------

```
#ifndef rbtbody.h
#define rbtbody.h

#include "typstrct.h"

void paint_robot(CASE case_state);          //第一次画机器人时调用，将画机器人处的背景储存，并画出机器人正面

void forebodyhead(CASE case_state);                    //机器人正面

void backbodyhead(CASE case_state);                    //机器人背面

void robot_left(CASE case_state);                     //机器人左面

void robot_right(CASE case_state);                     //机器人右面
```

```
void robot_hand_right(int x,int y,int theta);      //机器人的手，指向右侧，封装后在上面所述的几个函数中调用

void robot_hand_left(int x,int y,int theta);       //机器人的手，指向左侧，封装后在上面所述的几个函数中调用

void right_hold(CASE case_state);                   //机器人手持物品向右走

void left_hold(CASE case_state);                    //机器人手持物品向左走

void front_hold(CASE case_state);                   //机器人手持物品向下走（即正面）

void back_hold(CASE case_state);                    //机器人手持物品向上走（即背面）

#endif
```

----------rbtfunc.h----------

```
#ifndef rbtfunc.h
#define rbtfunc.h

#include "typstrct.h"

void rbtguard(CASE human, CASE robot, int identity, int *x, int *y, int *button);//安保功能

void treatment(CASE *human, CASE *robot, Graph G, int n, int *x, int *y, int *button, int *choice);//医疗照顾功能

void clean(CASE *robot, CASE *human, Graph G, int n);//打扫卫生功能

#endif // rbtfunc
```

----------rbtmove.h.----------

```
#ifndef rbtmove.h
#define rbtmove.h

#include "typstrct.h"

void dmove(CASE *case_state, int casetype);//改变坐标点

void move0(CASE *case_state,int casetype);//坐标的改变与动作叠加

int aimmove(CASE *case_state,int x0,int y0,int xt,int yt, const Graph G, const int n, int casetype);
//让机器人移动时需要直接调用的函数
```

#endif

----------register.h----------

#ifndef register.h
#define register.h

#include "title.h"
//功能：注册功能主逻辑函数
//输入: 用户信息链表的头节点, 指向账号字符串的字符指针, 指向密码字符串的字符指针, 鼠标坐标指针和按键信息指针
//输出：int 型
//　　返回 1：注册成功
//　　返回 5：按了返回键
int UserRegist(USERS *head,char *account,char *code,int *x,int *y,int *buttons);

//功能：decide which input function will be invoked
//输入：the pointer's information of mouse
//输出：int 型
//　　返回 1：进入账号输入函数
//　　返回 2：进入密码输入函数
//　　返回 3：进入二次确认密码输入函数
//　　返回 4：进入邮箱输入函数
//　　返回 6：进入输入验证码函数
//　　返回 7：点击了完成，进入注册验证函数
//　　返回 8：按了“back"
int input_area(int *x,int *y, int *buttons);

//功能：注册的各个模块的输入
//输入：
//　　char*inpu_c：将要输入的字符串的首地址
//　　int *x, int *y, int *buttons：鼠标的坐标和按键的指针
//　　int x_posi, int y_posi：输入起始位置（第一个字符的左上角坐标）
//　　int a_p：判断输入的是否是密码，是密码就显示实心圆，不是就显示该有的字符

```
//                                          0：不是密码
//                                          1：是密码
//输出：int 型
//                                  返回 1：进入账号输入函数
//                                  返回 2：进入密码输入函数
//                                  返回 3：进入二次确认密码输入函数
//                                  返回 4：进入邮箱输入函数
//                                  返回 6：进入输入验证码函数
//                                  返回 7：点击了完成，进入注册验证函数
//                                  返回 8：按了“back"
int inpu(char*inpu_c, int *x, int *y, int *buttons,int x_posi, int y_posi,int a_p);



//功能：静态注册界面
//输入：无
//输出：无
void regist_page(void);



//功能：获取验证码函数
//输入：指向邮箱字符串的字符指针，指向验证码字符串的字符指针，指针 flag
//输出：int 型
//                                  返回 1：账号密码信息完整，邮箱正确，生成验证码
//                                  返回 0：验证码生成错误
int get_verification(char *mail,char *real_veri, int flag[]);



//功能：判断注册是否成功
//输入：真的验证码的字符指针和自己写入的验证码的字符指针
//输出：int 型
//                                  返回 0：验证码不正确，失败
//                                  返回 1：验证码正确，成功
int regist_success(char *real_veri, char *veri);



//功能：判断邮箱是否为默认邮箱格式
```

```
//输入：指向邮箱字符串的字符指针
//输出：int 型
//                                返回 1：格式正确
//                                返回 0：格式错误
int whether_mail(char *str);

#endif
```

----------set_c.h----------

```
#ifndef set_c.h
#define set_c.h

#include "typstrct.h"

//功能：默认设置修改函数
//输入：空调温度和水温的指针
//输出：无
extern void set_change(int *air_t, int *bath_t);

#endif
```

----------svgahead.h----------

```
#ifndef svgahead.h
#define svgahead.h
#include<stdio.h>
#include<dos.h>
#include<conio.h>
#include<stdlib.h>
#include "typstrct.h"


/*设置 svga 显示模式  1024*768 256*/
extern void SetSVGA256();



/*模式  1024*768 64k*/
extern void SetSVGA64k();


/*获得当前 svga 显示模式的信息，返回显示模式号*/
extern unsigned int GetSVGA();
```

```
/*获取 SVGA 显示模式号 bx。摘录常用的模式号如下:
                    模式号        分辨率        颜色数
                    0x101         640*480       256
                    0x103         800*600       256
                    0x104         1024*768      16
                    0x105         1024*768      256
                    0x110         640*480       32K
                    0x111         640*480       64K
                    0x112         640*480       16.8M
                    0x113         800*600       32K
                    0x114         800*600       64K
                    0x115         800*600       16.8M
                    0x116         1024*768      32K
                    0x117         1024*768      64K
                    0x118         1024*768      16.8M
 ****************************************************/

typedef struct
{
      unsigned char B;   /*蓝色分量，BLUE 缩写*/
      unsigned char G;   /*绿色分量，GREEN 缩写*/
      unsigned char R;   /*红色分量，RED 缩写*/
} WESHEN;

 /*****************************************************
 功能说明：显存换页
 参数说明: page ,页面号
 *****************************************************/
extern unsigned int SelectPage(unsigned char page);

 /*****************************************************/

 /*****************************************************
      功能说明 ： 画点函数
      参数说明：x,y 所要画点位置 color 颜色
*****************************************************/
extern void      putpixel(int x,int y,int color);

/*****************************************************
   功能说明：得到某点的颜色值;
   参数说明：x,y 为该点的坐标;
   返回值：color 为该点的颜色值
*****************************************************/
```

```
extern int getpixel(int x,int y);



 /*******************************************************
 功能说明:     异或画点函数
 参数说明:      x,y 为像素位置，color  为异或的颜色
 返回值:        无
/*************************************************************/
extern void Xorpixel (int x, int y, int color);

/*****************************************************************
功能函数：用 64k 的模式画点
参数说明：画点的位置
返回值说明:   无返回
*****************************************************************/
extern void Putpixel64k(int x, int y,   int color);



/*********************************************************
     功能说明：从硬盘读取 8 位 BMP 直接到显存
     参数说明：x，y 坐标   name：文件路径
     返回值说明:
/*********************************************************/

extern int     Readbmp256(int x,int y,char * path);


 /*读取 24 位图片，参数 x,y 为图片位置，name 为路径，返回值：0 失败，1 成功*/
extern int Putbmp64k(int x,int y,const char *path);
extern unsigned int Getpixel64k(int x, int y);
extern void   put_image(int x0,int y0,int x1,int y1,unsigned int   *save);
extern void get_image(int x0,int y0,int x1,int y1,unsigned int   *save);
extern void   printf_image0(int x0, int y0, int x1, int y1);
extern void save_image0(int x0, int y0, int x1, int y1);
extern void   printf_image(int x0, int y0, int x1, int y1);
extern void save_image(int x0, int y0, int x1, int y1);
extern void   printf_image_2(int x0, int y0, int x1, int y1, int begin_y);

#endif
```

----------time_line.h----------

```
        #ifndef time_line.h
```

```
#define time_line.h
#include "typstrct.h"

//功能：主人回家函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
extern void come_home(CASE *robot, CASE *man, int *mx, int *my, int *button, int choice[],
VType G[], int n);



//功能：晚餐函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
extern void dinner(CASE *robot, CASE *man, int *mx, int *my, int *button, int choice[],VType
G[], int n);



//功能：娱乐工作函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
extern void entertain_and_work(CASE *robot, CASE *man, int *mx, int *my, int *button, int
choice[],VType G[], int n);



//功能：洗澡函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
extern void bath(CASE *robot, CASE *man, int *mx, int *my, int *button, int choice[],VType G[],
int n);



//功能：早餐函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
extern void breakfast(CASE *robot, CASE *man, int *mx, int *my, int *button,VType G[], int n);
#endif // time_line
```

----------title.h----------

```
#ifndef title.h
#define title.h
```

```
#include<stdio.h>
#include<conio.h>
#include<bios.h>
#include<fcntl.h>
#include<io.h>
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<bios.h>
#include<math.h>
#include<time.h>

#include"bricks.h"
#include"chat.h"
#include"chatHanz.h"
#include"chatInpu.h"
#include"chatQhwh.h"
#include"chatShow.h"
#include"color1.h"
#include"findway.h"
#include"finger.h"
#include"frnture.h"
#include"hzxs.h"
#include"input.h"
#include"iph_page.h"
#include"log_in.h"
#include"main_.h"
#include"main2.h"
#include"manbody.h"
#include"module_b.h"
#include"mouse.h"
#include"myhouse.h"
#include"rbtbody.h"
#include"rbtfunc.h"
#include"rbtmove.h"
#include"register.h"
#include"sdzl.h"
#include"svgahead.h"
#include"time_line.h"
#include"typstrct.h"
#include"user_list.h"
#include"wall.h"
#include"welcome.h"
```

```
#include"advance.h"
#include"basicgf.h"
#include"set_c.h"
#include"txt_save.h"

#define ORIGINX 767
#define ORIGINY 4
#define FINALX 1020
#define FINALY 547
#define MIDDLEX 894
#define ROBOTx 894
#define ROBOTy 404
#define TRUE 1
#define xmi 1
#define xma 1024
#define ymi 1
#define yma 767
#define PI 3.1415926
#define WITHOUT_THING 0
#define WITH_BOTTLE 1
#define WITH_CLOTHES 2
#define WITH_PLATE 3
#define MAXTXT 1000

#define MAN 1            //寻路用
#define ROBOT 0

#define INTRADER 0
#define MASTER 1
//关于颜色的说明
#define HAIRM 41605  //主人的头发，赭黄
#define SKINM 63222  //主人的皮肤，小麦色
#define EYESM 65535  //主人的眼睛，蓝
#define ARMSM 65504//主人的手臂，黄
#define PANTSM 29186      //主人的裤子，乌贼墨色

#define HAIRI 48006    //入侵者的头发，古铜色
#define SKINI 63222    //入侵者的皮肤，小麦色
#define EYESI 65535    //入侵者的眼睛，蓝
#define ARMSI 64342  //入侵者的手臂，暖粉红
#define PANTSI 52625 //入侵者的裤子，灰土色
#define PI 3.1415926
#define YG 40
#define M 4
```

```
#define UP 1
#define LEFT 2
#define DOWN 3
#define RIGHT 4
#define UP_DOWN 5
#define LEFT_RIGHT 6

#endif // _TITLE_H_
```

----------txt_save.h----------

```
#ifndef txt_save.h
#define txt_save.h

#include "typstrct.h"

//以下两个是机器人聊天框的背景存储函数
//第一个是得到聊天框的背景
//第二个是输出聊天框的背景，及覆盖聊天框
//输入：框的左上角
//输出：无
//尺寸：240*80
extern void saveimage_chat(int x,int y);

extern void putsave_chat(int x,int y);


//以下两个是机器人欢迎框的背景存储函数
//第一个是得到聊天框的背景
//第二个是输出聊天框的背景，及覆盖聊天框
//输入：框的左上角
//输出：无
//尺寸：240*80
extern void saveimage_welcome(int x,int y);

extern void putsave_welcome(int x,int y);


//以下两个是选择框的背景存储函数
//第一个是得到选择框的背景
//第二个是输出选择框的背景，及覆盖选择框
//输入：框的左上角
//输出：无
//尺寸：240*40
extern void saveimage_choose(int x,int y);
```

```
extern void putsave_choice(int x,int y);


//以下两个是机器人动作状态框的背景存储函数
//第一个是得到动作状态框的背景
//第二个是输出动作状态框的背景，及覆盖动作状态框
//输入：框的左上角
//输出：无
//尺寸：80*30
extern void saveimage_doing(int x,int y);

extern void putsave_doing(int x,int y);


//以下两个是 g_c 框的背景存储函数
//第一个是得到 g_c 框的背景
//第二个是输出 g_c 框的背景，及覆盖 g_c 框
//输入：框的左上角
//输出：无
//尺寸：90*40
extern void saveimage_g_c(int x,int y);

extern void putsave_g_c(int x,int y);


//以下两个是机器人 t_c 框的背景存储函数
//第一个是得到 t_c 框的背景
//第二个是输出 t_c 框的背景，及覆盖 t_c 框
//输入：框的左上角
//输出：无
//尺寸：120*40
extern void saveimage_t_c(int x,int y);

extern void putsave_t_c(int x,int y);


//以下两个是机器人 box 框的背景存储函数
//第一个是得到 box 框的背景
//第二个是输出 box 框的背景，及覆盖 box 框
//输入：框的左上角
//输出：无
//尺寸：200*40
extern void saveimage_box(int x,int y);
```

```
extern void putsave_box(int x,int y);


//以下两个是人背景存储函数
//第一个是得到人的背景
//第二个是输出人的背景，及覆盖人
//输入：框的左上角
//输出：无
//尺寸：70*90
extern void get_image_man(int x, int y);

extern void put_image_man(int x, int y);


//以下两个是机器人的背景存储函数
//第一个是得到机器人的背景
//第二个是输出机器人的背景，及覆盖机器人
//输入：框的左上角
//输出：无
//尺寸：70*90
extern void get_image_robot(int x, int y);

extern void put_image_robot(int x, int y);


//以下两个是垃圾 1 的背景存储函数
//第一个是得到垃圾 1 的背景
//第二个是输出垃圾 1 的背景，及覆盖垃圾 1
//输入：框的左上角
//输出：无
//尺寸：40*40
void get_image_trash1(int x, int y);

void put_image_trash1(int x, int y);


//以下两个是垃圾 2 的背景存储函数
//第一个是得到垃圾 2 的背景
//第二个是输出垃圾 2 背景，及覆盖垃圾 2
//输入：框的左上角
//输出：无
//尺寸：40*40
void get_image_trash2(int x, int y);
```

```
void put_image_trash2(int x, int y);


//以下两个是垃圾 3 的背景存储函数
//第一个是得到垃圾 3 的背景
//第二个是输出垃圾 3 的背景，及覆盖垃圾 3
//输入：框的左上角
//输出：无
//尺寸：40*40
void get_image_trash3(int x, int y);

void put_image_trash3(int x, int y);

#endif
```

----------typstrct.h----------

```
#ifndef typstrct.h
#define typstrct.h

///机器人和人的结构体,储存所有相关信息
typedef struct{

    char hand;                          //左手前为 1，右手为 0
    char hand_left;
    char hand_right;                    //手的运动参数,为 θ
    char  catch_th;                     //是否持物，1 持物，对于人来说，这个参数没有意
义
    int x;                                  //x,y 和 xpixel,ypixel 都是机器人 x 中心、y 顶上的
位置坐标
    int y;                              //小方格，16*19
    int xpixel;
    int ypixel;                         //像素点，1024*768
    int direction;                      //方向，1 为右，2 为左，3 为上，4 为下
}CASE;

///队

typedef int QElemtype;   //队的元素类型

typedef struct Qnode      //QElemtype 是队里面元素的类型，Qnode 链表结点的类型，内含
一个元素一个指针域
{
    QElemtype data;
```

```
    struct Qnode *next;
}Qnode, *QueuePtr;

typedef struct              //队指针
{
    QueuePtr front; //队头指针
    QueuePtr rear;   //队尾
}LinkQueue;

///图

typedef struct LinkNode //表结点
{
    int x;
    int y;
    char direction;   //1 为右，2 为左，3 为上，4 为下
    struct LinkNode *next;
}LNode;

typedef struct Node       //G 数组的元素类型，以及 G 的类型
{
    int x;   //x 坐标（大坐标，40*40 的 block）
    int y;
    LNode *next;     //指向表结点
}VType, *Graph;       //Graph 就是定义一个 VType 型的数组

typedef struct
{
    int x;
    int y;
}Axis;                    //x，y 坐标

///寻路主要部分
typedef struct
{
    int former;
    int direction;
}PathType;              //数组 path 的类型，第一维放前一个节点的索引，后一个表示方向

///栈

typedef PathType SElemtype;       //每个栈节点的 data 的数据类型，就是 PathType，因为要入栈出栈的就是 path
```

```
typedef struct StackNode              //栈节点
{
    SElemtype data;                    //指向栈底,整个栈的起点,在栈构造之前与销毁之后,
base 值为 NULL
    struct StackNode    *next;        //指向栈顶
}SNode;

typedef struct LinkStack              //链式栈
{
    SNode *bottom;          //指向栈底
    SNode *top;             //指向栈顶
}LkStack;


typedef struct ChTab
{
    int qhwh;
    char str[7];
}CH;

typedef struct EnTab
{
    int qhwh;
    char str;
}EN;

typedef struct Coordinate
{
    int x;
    int y;
} Coordinate;

typedef struct Area
{
    Coordinate lt;
    Coordinate rb;
} Area;

typedef struct USERS{
    char account[11];
    char code[11];
    struct USERS *next;
}USERS;
```

```
typedef struct tagBITMAPFILEHEADER{
    int bfType;
    long bfsize;//文件大小，单位为字节
    int bfReserved1;//保留，必须为 0
    int bfReserved2;//保留，必须为 0
    long bfOffBits;
}BITMAPFILEHEADER;

/*BMP 信息头结构*/
typedef struct tagBITMAPINFOHEADER{
    long biSize; /*信息头大小*/
    long biWidth; /*图像宽度*/
    long biHeight; /*图像高度*/
    int biPlanes; /*必须为 1*/
    int biBitCount; /*每像素位数，必须为 1，4，8，24*/
    long biCompression;    /*  压缩方法 */
    long biSizeImage; /* 实际图像大小，必须是 4 的倍数 */
    long biXPelsPerMeter; /*  水平方向每米像素数 */
    long biYPelsPerMeter; /*  垂直方向每米像素数*/
    long biClrUsed;    /*  所用颜色数*/
    long biClrImportant;    /* 重要的颜色数 */
} BITMAPINFOHEADER;

typedef struct tagRGBQUAD
{
    unsigned char B;  /*蓝色分量，RED 缩写*/
    unsigned char G;  /*绿色分量，GREEN 缩写*/
    unsigned char R;  /*红色分量，BLUE 缩写*/
    unsigned char reserved;          /*保留字*/
} RGBQUAD;

typedef struct{
    int x;                              //left_up
    int y;                              //left_up
    int height;
    int wide;
    int click;
    int over;
}BUTTONS;

#endif // typstrct
```

----------user_list.h----------

```
                    #ifndef filefun.h
#define filefun.h

#include "typstrct.h"

//功能：创建用户链表
//输入：用户信息链表的头指针
//输出：无
extern void create_list(USERS *head);


//功能：将新用户的信息写入文件
//输入：用户信息链表的头指针，密码字符指针，账号字符指针
//输出：无
extern void add_new_user(USERS *head,char *s1,char *s2);


//功能：通过账户来输出对应的密码
//输入：用户信息链表的头指针，账号字符指针
//输出：字符指针
extern char *accounts_2_code(USERS *head,char *string);


//功能：释放链表
//输入：用户信息链表的头指针
//输出：无
extern void free_list(USERS *head);
#endif
```

----------welcome.h----------

```
                    #ifndef welcome.h
#define welcome.h
#include "typstrct.h"

//功能：欢迎静止界面
//输入：无
//输出：无
extern void outwelcome(void);



//功能：goodbye 界面
//输入：无
```

```
//输出：无
extern void good_bye(void);



//功能：机器人 logo 输出
//输入：机器人的状态变量
//输出：无
extern void logo_robot(CASE robot_position);

#endif // welcome
```

2. 源文件

----------advance.c----------

```
#include "title.h"
/*****************************************************************************
void bow(int,int,int,int);
void fill_bow_right_up(int,int,int,int);
void fill_bow_left_up(int,int,int,int);
void fill_bow_right_down(int,int,int,int);
void fill_bow_left_down(int,int,int,int);
void fill_bow_down(int,int,int,int);
void fill_bow_up(int,int,int,int);
void fill_bow_left(int,int,int,int);
void fill_bow_right(int,int,int,int);
void bar_round(int,int,int,int,int,int,int);
void bow_right_up(int,int,int,int);
void bow_right_down(int,int,int,int);
void bow_left_up(int,int,int,int);
void bow_left_down(int,int,int,int);
void lean_line(int,int,int,int,int);
void theta_bar(int x,int y,int length,int wide,int theta,int color);
void robot_hand_left(int,int,int);
void robot_hand_right(int ,int ,int );
void red_cross(int x,int y);
void green_tick(int x,int y);
void triangle1(int x,int y,int height,int color);
*****************************************************************************/

/*只能画竖直方向的实心椭圆
```

```
输入：两个圆心坐标及半径，颜色*/
void Fillellipse(int x1,int y1,int x2,int y2,int r,int color)
{
	FillCircle(x1,y1,r,color);
	FillCircle(x2,y2,r,color);
	bar(x1-r,y1,x2+r,y2,color);
}

/*2018/10/1 新增
可画出水平方向的实心椭圆
输入：两个圆心坐标及半径，颜色*/
void ever_Fillellipse(int x1,int y1,int x2,int y2,int r,int color)
{
	FillCircle(x1,y1,r,color);
	FillCircle(x2,y2,r,color);
	bar(x1,y1-r,x2,y2+r,color);
}

/*四个画四分之一弧函数，非实心，均使用勾股定理计算*/


void bow_right_down(int x,int y,int r,int color)
{
	int tx=0,d;
	while(tx<=r)
	{
		d = sqrt(r*r-tx*tx);
		Putpixel64k(x+d,y+tx,color);
		tx++;
	}
}


void bow_left_down(int x,int y,int r,int color)
{
	int tx=0,d;
	while(tx<=r)
	{
		d = sqrt(r*r-tx*tx);
		Putpixel64k(x-d,y+tx,color);
		tx++;
	}
}
```

```
/*填充四分之一圆，右上*/
void fill_bow_right_up(int x,int y,int r,int color)
{
	int tx = 0, ty = r, d = 3 - 2 * r, i;
	while( tx < ty)
	{
		// 画水平两点连线(<45 度)
		for (i = x; i <= x + ty; i++)
		{
			Putpixel64k(i, y - tx, color);
		}

		if (d < 0)		// 取上面的点
			d += 4 * tx + 6;
		else			// 取下面的点
		{
			// 画水平两点连线(>45 度)
			for (i = x; i <= x + tx; i++)
			{
				Putpixel64k(i, y - ty, color);

			}

			d += 4 * (tx - ty) + 10, ty--;
		}

		tx++;
	}
	if (tx == ty)		// 画水平两点连线(=45 度)
		for (i = x; i <= x + ty; i++)
		{
			Putpixel64k(i, y - tx, color);

		}
}

void fill_bow_left_up(int x,int y,int r,int color)
{
	int tx=0,ty=r,i;
	double sx;
	while(tx<ty)
	{
		sx = sqrt(r*r-tx*tx);
		for(i=x-sx;i<=x;i++)
```

```
            {
                Putpixel64k(i,y-tx,color);
            }
            tx++;
        }
}

void fill_bow_right_down(int x,int y,int r,int color)
{
        int tx=0,ty=r,i;
        double sx;
        while(tx<ty)
        {
            sx = sqrt(r*r-tx*tx);
            for(i=x;i<=x+sx;i++)
            {
                Putpixel64k(i,y+tx,color);
            }
            tx++;
        }
}

void fill_bow_left_down(int x,int y,int r,int color)
{
        int tx=0,ty=r,i;
        double sx;
        while(tx<ty)
        {
            sx = sqrt(r*r-tx*tx);
            for(i=x-sx;i<=x;i++)
            {
                Putpixel64k(i,y+tx,color);
            }
            tx++;
        }
}

void fill_bow_down(int x,int y,int r,int color)
{
        int tx = 0, ty = r,i;
        float d = r/1.414;
        double sx;
        while(tx<ty)
        {
```

```
            if(tx<d)
            {
                for(i=x-tx;i<=x+tx;i++)
                {
                    Putpixel64k(i,y+tx,color);

                }
            }
            else
            {
                for(i=x-sx;i<=x+sx;i++)
                {
                    Putpixel64k(i,y+tx,color);
                }
            }
            tx++;
            sx = sqrt(r*r-tx*tx);
        }
}

void fill_bow_up(int x,int y,int r,int color)     //上半圆
{
        int tx = 0, ty = r,i;
        float d = r/1.414;
        double sx;
        while(tx<ty)
        {
            if(tx<d)
            {
                for(i=x-tx;i<=x+tx;i++)
                {
                    Putpixel64k(i,y-tx,color);

                }
            }
            else
            {
                for(i=x-sx;i<=x+sx;i++)
                {
                    Putpixel64k(i,y-tx,color);
                }
            }
            tx++;
            sx = sqrt(r*r-tx*tx);
```

```
    }
}


/*圆角矩形*/
void bar_round(int x,int y,int length,int height,int r,int thick,int color) //r 是圆角半径，thick 是粗细
{
    bar(x-length/2+r,y-height/2,x+length/2-r,y+height/2,color);
    bar(x-length/2,y-height/2+r,x+length/2,y+height/2-r,color);
    fill_bow_left_up(x-length/2+r,y-height/2+r,r,color);
    fill_bow_left_down(x-length/2+r,y+height/2-r,r,color);
    fill_bow_right_up(x+length/2-r,y-height/2+r,r,color);
    fill_bow_right_down(x+length/2-r,y+height/2-r,r,color);
    linelevel(x-length/2+r,y-height/2,x+length/2-r,y-height/2,thick,color);
    linelevel(x-length/2+r,y+height/2,x+length/2-r,y+height/2,thick,color);
    linever(x-length/2,y-height/2+r,x-length/2,y+height/2-r,thick,color);
    linever(x+length/2,y-height/2+r,x+length/2,y+height/2-r,thick,color);
    bow_right_up(x+length/2-r,y-height/2+r,r,color);
    bow_left_up(x-length/2+r,y-height/2+r,r,color);
    bow_left_down(x-length/2+r,y+height/2-r,r,color);
    bow_right_down(x+length/2-r,y+height/2-r,r,color);
}

void bar_round_2(int x0,int y0,int x1,int y1,int r,int thick,int color)
{
    int length, height, x, y;
    length = x1-x0;
    height = y1-y0;
    x = (x1+x0)/2;
    y = (y1+y0)/2;
    bar_round(x,y,length,height,r,thick,color);
}


/*斜线*/
void lean_line(int x,int y,int length,int theta,int color)//x,y 为线段的起点
{
    double right_x;
    double i,y0;
    double theta0 = ((double)(theta))/180*PI;
    right_x= x+cos(theta0)*(length);
    y0 = y;
    if((int)(theta)<=90)
```

```
    {
        for(i=x;i<=right_x;i++)
        {
            Putpixel64k(i,y0,color);
            y0 += tan(theta0);
        }
    }
    else
    {
        for(i=x;i>=right_x;i--)
        {
            Putpixel64k(i,y0,color);
            y0 = y0+tan(theta0);
        }
    }

}

/*斜矩形*/
void theta_bar(int x,int y,int length,int height,int theta,int color)//x,y 为矩形左上角,height 为线长,θ 为跟 x 正方向夹角
{
    lean_line_thick(x,y,height,theta,length,color);
}

/*树上方的三角形*/
void triangle1(int x,int y,int height,int color)//x,y 为三角形底边中点
{
    int i,j=0;
    for(i=0;i<height;i++)
    {
        for(j=0;j<=i;j++)
        {
            Putpixel64k(x-j/2,y+i,color);
            Putpixel64k(x+j/2,y+i,color);
        }

    }
}

void bow_right_up(int x,int y,int r,int color)//右上方
{
    int tx=0,d;
    while(tx<=r)
```

```
    {
        d = sqrt(r*r-tx*tx);
        Putpixel64k(x+d,y-tx,color);
        tx++;
    }
}

void bow_left_up(int x,int y,int r,int color)
{
    int tx=0,d;
    while(tx<=r)
    {
        d = sqrt(r*r-tx*tx);
        Putpixel64k(x-d,y-tx,color);
        tx++;
    }
}
/*通过加粗斜线画出平行四边形*/
void lean_line_thick(int x,int y,int length,int theta,int thick,int color)//x,y 为矩形的左边左上角坐标
{
    int i;
    for(i=0;i<thick;i++)
    {
        lean_line(x+i,y,length,theta,color);
    }
}

/*提示错误的红色叉*/
void red_cross(int x,int y)
{
    lean_line_thick(x,y,30,45,3,63488);
    lean_line_thick(x,y+20,30,-45,3,63488);
}

/*提示正确的绿色勾*/
void green_tick(int x,int y)
{
    lean_line_thick(x,y,15,45,3,2047);
    lean_line_thick(x+13,y+5,40,-45,3,2047);
}

void bar_round_with_shadow(int x,int y,int length,int height,int r,int thick,int color)
{
```

```
    bar(x-length/2+r,y-height/2,x+length/2-r,y+height/2,color);
    bar(x-length/2,y-height/2+r,x+length/2,y+height/2-r,color);
    fill_bow_left_up(x-length/2+r,y-height/2+r,r,color);
    fill_bow_left_down(x-length/2+r,y+height/2-r,r,color);
    fill_bow_right_up(x+length/2-r,y-height/2+r,r,color);
    fill_bow_right_down(x+length/2-r,y+height/2-r,r,color);
    linelevel(x-length/2+r,y-height/2,x+length/2-r,y-height/2,thick,65535);
    linelevel(x-length/2+r,y+height/2,x+length/2-r,y+height/2,thick*3,0);
    linever(x-length/2,y-height/2+r,x-length/2,y+height/2-r,thick,65535);
    linever(x+length/2,y-height/2+r,x+length/2,y+height/2-r,thick*2,0);
    bow_right_up(x+length/2-r,y-height/2+r,r,0);
    bow_left_up(x-length/2+r,y-height/2+r,r,65535);
    bow_left_down(x-length/2+r,y+height/2-r,r,65535);
    bow_right_down(x+length/2-r,y+height/2-r,r,0);
}
```

----------basicgf.c----------

```
    #include "title.h"

    /*****************************************************************************
    修改日志：2020/07/23，写了个等腰直角三角形的函数，直角方向上偏左 45°，不可变方向
    *****************************************************************************/
      /*********************************************************************
       功能说明：画水平线函数
       参数说明：x0,y0 起始坐标    x1,y1 终止坐标   thick 厚度   color 颜色
       无返回值
     *********************************************************************/
      void linelevel(int x0,int y0,int x1,int y1,int thick,int color)
      {
       int i,j,k;
       if(x0>x1)                                          /*确保 x0 为较小的一方*/
       {
        k=x0;
        x0=x1;
        x1=k;
       }

       if(x1<=0||x1>=1024||y1<=0||y1>=768||x1>=1024)   /*确保画线在屏幕范围之内*/
        {
              printf("the line is beyond the screen!");
              return;
        }
```

```
    for(i=0;i<thick;i++)
    {
      for(j=0;j<x1-x0;j++)
      {
            Putpixel64k(x0+j,y0+i,color);
      }
    }
 }

/*************************************************************/



/*****************************************************************
    功能说明：画竖线函数
    参数说明：x0,y0 起始坐标    x1,y1 终止坐标   thick 厚度   color 颜色
    无返回值
*****************************************************************/
void linever(int x0,int y0,int x1,int y1,int thick,int color)
  {
     int     i = 0;
     int     j = 0;
     //int     k=0;
    int high;
    high = y1 - y0;
   /* if(y0>y1)                                                  确保 y0 为较小的一方
    {
      k=y0;
      y0=y1;
      y1=k;
    }
    */
    if(x1<=0||x1>=1024||y1<=0||y1>=768||x1>=1024)   /*确保画线在屏幕范围之内*/
     {
           printf("the line is beyond the screen!");
           return;
     }

    for(i=0;i<thick;i++)
    {
      for(j=0;j<high;j++)
      {
           Putpixel64k(x0+i,y0+j,color);
```

```
    }
   }
 }


/*********************************************************/


/*********************************************************
Function:       Horizline

Description:  画水平线函数
可以接收超出屏幕范围的数据，只画出在屏幕内部分
因为没有防止整型变量溢出的判断，画超出屏幕的线时应防止输入特大数据

Calls:          Selectpage

Called By:      Line
Bar
Circlefill

Input:          int x                   起始点横坐标, 从左到右增加, 0 为最小值 (屏幕参考系)
int y                   起始点纵坐标，从上到下增加，0 为最小值（屏幕参考系）
int width               水平长度，为正向右延伸，为负向左延伸
unsigned char color         颜色数，共有 256 种

Output:         屏幕上画出水平线

Return:     None
Others:     None
*********************************************************/
void Horizline(int x, int y, int width, int  color)
{
    /*显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int  far * const video_buffer = (unsigned int  far *)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page = 0;

//  unsigned char old_page = 0;
    /*对应显存地址偏移量*/
    unsigned long int page;
```

```
/*i 是 x 的临时变量，后作循环变量*/
int i;

/*判断延伸方向，让起始点靠左*/
if (width < 0)
{
	x = x + width;
	width = -width;
}

i = x;

/*省略超出屏幕左边部分*/
if (x < 0)
{
	x = 0;
	width += i;
}

/*整条线在屏幕外时退出*/
if (x >= 1024)
{
	return;
}

/*整条线在屏幕外时退出*/
if (y < 0 || y >= 768)
{
	return;
}

/*省略超出屏幕右边部分*/
if (x + width > 1024)
{
	width = 1024 - x;
}

/*计算显存地址偏移量和对应的页面号，做换页操作*/
page = ((unsigned long int)y << 10) + x;
new_page = page >> 15;

	SelectPage(new_page);
```

```
    /*向显存写入颜色，水平线画出*/
    for (i = 0; i < width; i++)
    {
        *(video_buffer + page + i) = color;
    }
}


/************************************************************
    功能说明：画水平矩形函数
    参数说明: x0,y0  左上角坐标     x1,y1 右下角坐标  color  颜色
************************************************************/
void bar(int x0, int y0, int x1, int y1, int color)
{

    int i;

    int wide;                       /*计算矩形的长和宽*/
    wide = x1 - x0;


    for (i = y0; i <= y1; i++)
    {
        Horizline(x0, i, wide, color);
    }
}


/************************************************************
 功能说明：以(x,y)点为圆心，以 radius 为半径画圆，没有防止超出屏幕的判断，
 可以将没有超出的部分画出。
 参数说明：x,y 为圆点，radius  为半径，color 为颜色。
 无返回值：
 ************************************************************/
 void circle(int x0,int y0,int radius,int color)
 {

    int x, y, d;
    y = radius;
    d = 3 - radius << 1;

    for (x = 0; x <= y; x++)
    {
```

```
            Putpixel64k(x0 + x, y0 + y, color);
            Putpixel64k(x0 + x, y0 - y, color);
            Putpixel64k(x0 - x, y0 - y, color);
            Putpixel64k(x0 - x, y0 + y, color);
            Putpixel64k(x0 + y, y0 + x, color);
            Putpixel64k(x0 + y, y0 - x, color);
            Putpixel64k(x0 - y, y0 - x, color);
            Putpixel64k(x0 - y, y0 + x, color);

            if (d < 0)
            {
                d += x * 4 + 6;
            }

            else
            {
                d += (x - y) * 4 + 10;
                y--;
            }
        }
}
```

```
/*************************************************************************
        功能说明：画实心圆
        参数说明：x,y 为圆心
       无返回值
*************************************************************************/
 void FillCircle(/*int x, int y, int r, int color*/int xc, int yc, int radius, int color)
{
    /*画圆圈的定位变量和决策变量*/
    int x = 0,
        y = radius,
        dx = 3,
        dy = 2 - radius - radius,
        d = 1 - radius;


    /*半径必须为正，否则退出*/
    if (radius <= 0)
```

```
    {
        return;
    }

    /**********************************
    以下运用 Bresenham 算法生成实心圆。
    该算法是得到公认的成熟的快速算法。
    具体细节略去。
    **********************************/
    while (x <= y)
    {
        Horizline(xc - x, yc - y, x + x, color);
        Horizline(xc - y, yc - x, y + y, color);
        Horizline(xc - y, yc + x, y + y, color);
        Horizline(xc - x, yc + y, x + x, color);

        if (d < 0)
        {
            d += dx;
            dx += 2;
        }

        else
        {
            d += (dx + dy);
            dx += 2;
            dy += 2;
            y--;
        }

        x++;
    }
 }



 /*弧*/
void bow(int x0,int y0,int r,int color)//弧度固定
{
    int x,y,d;
    y = r;
    d = 3-r<<1;

    for(x=0;x<=y/4;x++)
```

```
        {
            Putpixel64k(x0 + x, y0 + y, color);
            Putpixel64k(x0 - x, y0 + y, color);
            if(d<0)
            {
                d+=x*4+6;
            }
            else{
                d+=(x-y)*4+10;
                y--;
            }
        }
    }

    void eqver_tri(int x, int y, int length, int color)     //等腰直角三角形，length 是直角边长
度
    {    //方向是固定的，入口参数是直角点的坐标
        int i;
        for(i = 0; i < length; i++)
        {
            linelevel(x, y + i, x + length - i, y + i, 1, color);
        }
    }




    void delay0(int time)
     {
        int i,j,k;
        for(i=0;i<time;i++)
        {
            for (j = 0; j < 1100; j++)
                for(k=0;k<100;k++);


        }
     }
```

----------bricwall.c----------

```
#include "title.h"
#define YG 40
#define M 4
```

```
#define UP 1
#define LEFT 2
#define DOWN 3
#define RIGHT 4
#define UP_DOWN 5
#define LEFT_RIGHT 6
/*****************************************************
地板瓷砖和墙壁
入口参数均为左上角坐标（40*40 的大坐标）
*****************************************************/

//客厅
void wood_ver(int x, int y)        //覆盖一个 40*40 的块
{
    bar(x*YG,y*YG+M,(x+1)*YG,(y+1)*YG+M,64908);          //    蜜橙色背景
    linever(x*YG,y*YG+M,(x+1)*YG,(y+1)*YG+M,1,31331);    //    五条竖线
    linever(x*YG+10,y*YG+M,(x+1)*YG+10,(y+1)*YG+M,1,31331); //隔十个像素点画一条
    linever(x*YG+20,y*YG+M,(x+1)*YG+20,(y+1)*YG+M,1,31331); //即 40*40 单位块上细分
    linever(x*YG+30,y*YG+M,(x+1)*YG+30,(y+1)*YG+M,1,31331);
    linever(x*YG+40,y*YG+M,(x+1)*YG+40,(y+1)*YG+M,1,31331);
}

//浴室
void glass(int x, int y)
{
    bar(x*YG,y*YG+M,(x+1)*YG,(y+1)*YG+M,48797);          //    蓝色背景
    bar(x*YG,y*YG+M,(x+1)*YG,y*YG+M+2,55070);      //    浅色边框
    bar(x*YG,y*YG+M,x*YG+2,(y+1)*YG+M,55070);
    bar(x*YG,(y+1)*YG+M-2,(x+1)*YG,(y+1)*YG+M,46651);              //    深色边框
    bar((x+1)*YG-2,y*YG+M,(x+1)*YG,(y+1)*YG+M,46651);

    bar(x*YG+2,y*YG+M+18,(x+1)*YG-2,y*YG+M+20,46651);                //中间十字
    bar(x*YG+2,y*YG+M+20,(x+1)*YG-2,y*YG+M+22,55070);
    bar(x*YG+18,y*YG+M+2,x*YG+20,(y+1)*YG+M-2,46651);
    bar(x*YG+20,y*YG+M+2,x*YG+22,(y+1)*YG+M-2,55070);
}

//    厨房
void green_kitchen(int x, int y)
{
    bar(x*YG,y*YG+M,(x+1)*YG,(y+1)*YG+M,42552);          //    绿色背景

    bar(x*YG,y*YG+M,(x+1)*YG,y*YG+M+1,31922);      //    深色边框
    bar(x*YG,y*YG+M,x*YG+1,(y+1)*YG+M,31922);
```

```
    bar(x*YG,(y+1)*YG+M-1,(x+1)*YG,(y+1)*YG+M,31922);
    bar((x+1)*YG-1,y*YG+M,(x+1)*YG,(y+1)*YG+M,31922);

    bar(x*YG+1,y*YG+M+19,(x+1)*YG-1,y*YG+M+20,31922);          //中间十字
    bar(x*YG+1,y*YG+M+20,(x+1)*YG-1,y*YG+M+21,31922);
    bar(x*YG+19,y*YG+M+1,x*YG+20,(y+1)*YG+M-1,31922);
    bar(x*YG+20,y*YG+M+1,x*YG+21,(y+1)*YG+M-1,31922);
}

//  卧室
void green_bedroom(int x, int y)
{
    bar(x*YG,y*YG+M,(x+1)*YG,(y+1)*YG+M,65502);      //  白色背景
    bar(x*YG,y*YG+M,(x+1)*YG,y*YG+M+2,52720);    //  深色边框
    bar(x*YG,y*YG+M,x*YG+2,(y+1)*YG+M,52720);
    bar(x*YG,(y+1)*YG+M-2,(x+1)*YG,(y+1)*YG+M,57010);          //  浅色边框
    bar((x+1)*YG-2,y*YG+M,(x+1)*YG,(y+1)*YG+M,57010);
}

//walls
//正面看上去最大的那堵大墙
void w_blue(int x, int y)
{
    bar(x*YG,y*YG+M,(x+1)*YG,(y+1)*YG+M,42260);
    bar(x*YG,y*YG+M,(x+1)*YG,y*YG+M+5,31625);
}

//在指定 x，y 代表的格子的右侧画墙壁
void w_right(int x,int y)              //w_开头的就是细细的边框
{                                          //right 是在指定格子的右边画
    bar((x+1)*YG,y*YG+M,(x+1)*YG+4,(y+1)*YG+M,31625);
}

//在指定 x，y 代表的格子的左侧画墙壁
void w_left(int x, int y)              //left 在指定格子的左边
{
    bar(x*YG,y*YG+M,x*YG+5,(y+1)*YG+M,31625);
}

void w_down(int x, int y)
{
    bar(x*YG,(y+1)*YG+M-2,(x+1)*YG,(y+1)*YG+M,31625);
}
```

----------chat.c----------

```
#include "title.h"


//功能：画聊天界面框架
//输入：无
//输出：无
void chat_interface()
{
	/***************
	输入框
	**************/
	bar_round_2(ORIGINX+13,ORIGINY+50,ORIGINX+243,ORIGINY+413,5,1,54938);
	bar_round_2(ORIGINX+13,ORIGINY+403,ORIGINX+243,ORIGINY+508,5,1,65535);
}

//功能：聊天功能主函数
//输入；机器人的状态变量
//输出：无
void chatmain(CASE *robot)
 {
	Area box_area = { {ORIGINX+15+2,ORIGINY+420} , {ORIGINX+240,ORIGINY+465} };
	//输入框左上角坐标
	Coordinate current_show_position = {ORIGINX+17,ORIGINY+415};		//现在正在码的字的左上角坐标

	int i;
	int flag;
	int *qhwh=(int *)malloc(sizeof(int)*MAXTXT);		//区号位号


	if(qhwh==NULL)
	{
		overflow_box(500,500);
		getch();
		exit(1);
	}
	//界面
	iph_frame(28153);
	chat_interface();

	while(1)
	{
		//初始化区位号
		QhwhToZero(qhwh);
```

```
            //开始输入法
            flag = ShowChTab(box_area,current_show_position/*现在正在码的字的左上角坐标
*/, 16,qhwh) ;
            //回车键之后刷新聊天窗口
            bar(ORIGINX+15, ORIGINY+405, ORIGINX+240, ORIGINY+460,65535);
            if(flag == 3)
            {
                  show_main(qhwh,robot);
            }
            else
            {
                  break;
            }
            getch();
      }
      free(qhwh);
      qhwh = NULL;
 }
```

----------chatHanz.c----------

```
             #include "title.h"
/******************函数清单**************************
1.    int ShowChinese(int *qhwh,int num_chinese,int color,Area show_area,int size,char
*hzk_dir)
2.    int ShowPerCharacter(int qhwh,Area show_area,Coordinate * current_position,int color,int
size,char *hzk_dir)
3.    int GetBit(int num,char ch)
4.    void CheckArea(Area area,Coordinate *current_position,int size)
*************************************************/
int ShowChinese(int *qhwh,int num_chinese,int color,Area show_area,int size,char *hzk_dir)//1
{

      int i=0;

      Coordinate current_position;//定义当前左上角坐标
      current_position.x=show_area.lt.x;//初始化当前左上角坐标
      current_position.y=show_area.lt.y;


    for(i=0;i<num_chinese;i++)//逐个汉字开始打点
    {
            ShowPerCharacter(qhwh[i],show_area,&current_position,color,size,  hzk_dir);// 打 出
单个字符
```

```
    }
    return 0;
}

int ShowPerCharacter(int qhwh,Area show_area,Coordinate * current_position,int color,int size,char *hzk_dir)//2
{
    int i=0,j=0,k=0;
    int x=current_position->x;
    int y=current_position->y;
    int real_rb_x= ((show_area.rb.x - show_area.lt.x)/size)*size + show_area.lt.x;

    FILE *hzk;//定义汉字库文件指针
    char *hzk_file;//定义汉字缓存数组
    int width_byte=0;//定义字节长度
    int qh=0,wh=0;//定义区号位号
    unsigned char *bitmap;//定义 bmp 指针，用来储存字形码
    long offset;//定义偏移量
    int size_read=0;//定义汉字读取有效长度
    if(x==real_rb_x) x=x-size;

    if((hzk_file=(char *)malloc(sizeof(char)*512))==NULL)//如果动态分配 bmp 指针内存失败，则直接返回
    {
            overflow_box(500,500);
            getch();
        perror("fail to malloc");
        exit(1);
    }

    if(size%4!=0&&size<=8)//如果字体太小或者不是 4 的倍数，则直接返回
    {
        perror("false size");
        exit(1);
    }
    else
    {
        size_read=((size+4)/8)*size;//初始化汉字读取有效长度
    }

    sprintf(hzk_file,"%s\\hzk%d",hzk_dir,size);//把汉字库的数据写入汉字缓存数组中

    if((hzk=fopen(hzk_file,"rb"))==NULL)//如果汉字库文件打开为空，则直接返回
```

```
{
        overflow_box(500,500);
        getch();
    perror("fail to open");
    exit(1);
}

    if((bitmap=(char *) malloc(sizeof(char)*size_read))==NULL)//如果动态分配 bmp 指针内存失败，则直接返回
{
        overflow_box(500,500);
        getch();
    perror("fail to malloc");
    exit(1);
}

    qh=qhwh/100;//初始化区号
    wh=qhwh%100;//初始化位号
    offset=(long) (94*((int) qh-1)+((int) wh-1))*size_read;//初始化偏移量

    fseek(hzk,offset,SEEK_SET);//根据偏移量寻找汉字库的该汉字的字形码
    fread(bitmap,sizeof(char),size_read,hzk);//读取汉字库中的该汉字的字形码

    width_byte=(size+4)/8;//计算字节数宽度

    for(i=0;i<size;i+=1)
    {
        for(j=0;j<width_byte;j+=1)
        {
            for(k=0;k<8;k+=1)
            {
                if(GetBit(k+1,bitmap[i*width_byte+j])==1)//如果该像素点需要打印
                {

                    Putpixel64k(x+j*8+k, y+i, color);

                }
            }
        }
    }
    CheckArea( show_area,current_position, size);
    free(hzk_file);
    free(bitmap);
        hzk_file = NULL;
```

```
            bitmap = NULL;
    fclose(hzk);
    return 0;
}

int GetBit(int num,char ch)//3
{
    int i;
    int temp=1;
    for(i=0;i<8-num;i++)
    {
        temp*=2;
    }
    return (temp&ch)&&1;
}

void CheckArea(Area area,Coordinate *current_position,int size)//4
{
    if( (current_position->x+size*2) <=area.rb.x)//如果当前位置 x 轴不越出边界，则初始位置后移
    {
        current_position->x+=size;
    }

    else if( (current_position->y+size*2)<= area.rb.y )//如果当前位置 y 轴不越出边界，则初始位置换行
    {
        current_position->y+=size;
        current_position->x=area.lt.x;
    }
    else current_position->x=((area.rb.x - area.lt.x)/size)*size + area.lt.x;   //stay where you are
}
```

----------chatInpu.c----------

```
#include "title.h"
/******************函数清单**************************
1.  void CursorWhite(Coordinate current_show_position,int show_size)
2.  void CursorBlack(Coordinate current_show_position,int show_size)
3.  void DeleteShow(Area show_area,Coordinate * current_show_position,int show_size )
4.  void ShowWhite(Coordinate * current_show_position,int show_size)
5.  void DeleteTab(Coordinate * current_en_position,CH* ch,EN* en,char *temp ,int * num_tab,int *ch_qhwh, unsigned int *box_save)
```

6. void ShowTxt(int *qhwh, Area show_area,Coordinate* current_show_position,int show_size)

7. int ShowChTab(Area show_area,Coordinate current_show_position, int show_size,int *qhwh)

**********/

```
//展示白色光标
void CursorWhite(Coordinate current_show_position,int show_size){
    bar(current_show_position.x,current_show_position.y,current_show_position.x+2,current_show_position.y+show_size,65535);
}

//展示黑色光标
void CursorBlack(Coordinate current_show_position,int show_size){
    bar(current_show_position.x,current_show_position.y,current_show_position.x+2,current_show_position.y+show_size,0);
}
```

/*************

似乎只是单纯的让定位改变，没有体现“删除”操作

*************/

```
void DeleteShow(Area show_area,Coordinate * current_show_position,int show_size )//1
{

    //初始化 real_rb_x，使其指向定位
    int real_rb_x=((show_area.rb.x - show_area.lt.x)/show_size)*show_size + show_area.lt.x;
    //int real_rb_y=((show_area.rb.y - show_area.lt.y)/show_size)*show_size + show_area.lt.y;
    if(current_show_position->x>show_area.lt.x || current_show_position->y > show_area.lt.y)//框内是否有字的检测
    {
        //有字有两种情况①光标不在最左侧即左侧有字符②光标在最左侧但不在第一行即上面有字符
        if(current_show_position->x > show_area.lt.x )                    //若① 则执行下一句
            current_show_position->x =current_show_position->x-show_size;     //定位往前移一格删除一个
        else //②的情况
```

```
            {
                current_show_position->x = real_rb_x- show_size;        //x 移到倒数第二
位
                current_show_position->y = current_show_position->y- show_size;
    //y 移到上一行
            }
        }
        ShowWhite( current_show_position, show_size);
    }



    /**********************

    似乎是写了个白 bar

    **********************/
    void ShowWhite(Coordinate * current_show_position,int show_size)//2
    {
        int x,y,i,j,k;
        int width_byte=0;//定义字节长度
        x=current_show_position->x;
        y=current_show_position->y;

        width_byte=(show_size+4)/8;//计算字节数宽度

        for(i=0;i<show_size;i+=1)
        {
            for(j=0;j<width_byte;j+=1)
            {
                for(k=0;k<8;k+=1)
                {
                    Putpixel64k(x+j*8+k, y+i, 0xFFFFFF);
                }
            }
        }
    }

    /***************

    在输入法的框中删除字母
    （区别于在输入框中删除字母

    ****************/
```

```
    void DeleteTab(Coordinate * current_en_position,CH* ch,EN* en,char *temp ,int * num_tab,int *ch_qhwh)//3
    {
        int en_qhwh,i;

        Area show_ch_area={ {ORIGINX+30,ORIGINY+365} , {ORIGINX+222,ORIGINY+380} };
        Area show_en_area={ {ORIGINX+30,ORIGINY+388} , {ORIGINX+222,ORIGINY+400} };
        Coordinate current_ch_position ={ORIGINX+30,ORIGINY+365};

        --(*num_tab);    //  不是很懂。。哪里要减一?
        /***************
        懂了懂了，这里说明一下这个 num_tab 吧
        这个就是拼音中的每个字母数
        之后有一个 FindChQhQh 这个函数就是将前几位拼音与汉字库中做对比，
        而这个 num_tab 就是某个汉字的总字母数
        每次按 backspace 触发这个 DeleteTab 函数意味着删去输入拼音中的刚刚那个字母
        ***************/
        *ch_qhwh = FindChQhwh(ch,temp, *num_tab, *ch_qhwh);          //在 num_tab 少了一个之后重新在汉字库中寻找匹配值
        current_en_position->x=ORIGINX+30;          //设置英文输入起始位置

        //重新 put 输入法框
        //Putbmp64k(589,633,"bmp\\ChTab.bmp");
        //put_image(ORIGINX+25,ORIGINY+365,ORIGINX+225,ORIGINY+400, box_save);
        putsave_box(ORIGINX+25,ORIGINY+365);

        //重新输出文字信息（都是保存好的地址的之前的内容
        for(i=0;i<*num_tab;i++)
        {
            en_qhwh = FindEnQhwh(en,temp[i]);
            ShowPerCharacter(en_qhwh,show_en_area, current_en_position ,0x000000,16,"hzk");
        }

        for(i=0;i<5;i++)
        {

ShowPerCharacter(217+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");

ShowPerCharacter((*ch_qhwh)+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");
        }
```

```
}

/***************

将刚刚输入法打出的字整合成一句话展示
ShowPerCharacter 用来决定文字是否需要换行
else 后的操作用来改变定位

****************/

void ShowTxt(int *qhwh, Area show_area,Coordinate* current_show_position,int show_size)//6
{
    int i=0,num_qhwh;//num_qhwh 相当于 current 的字数统计
    int real_rb_x=((show_area.rb.x - show_area.lt.x)/show_size)*show_size + show_area.lt.x;
    int real_rb_y=((show_area.rb.y - show_area.lt.y)/show_size)*show_size + show_area.lt.y;
    num_qhwh=CheckQhwhNum(qhwh);//看有几个字

    for(i=0;i<num_qhwh;i++)
    {
        if(qhwh[i]!=0) //打印整句话
        ShowPerCharacter(qhwh[i],show_area, current_show_position,0 , show_size,"hzk");
        else
        {
            ShowWhite( current_show_position, show_size);
            if((current_show_position->x+show_size)<=real_rb_x||(current_show_position->y+show_size)<=real_rb_y)//判断需不需要换行从最左边开始打印
            {
                if(current_show_position->x < real_rb_x - show_size)    //不需要换行，直接接在本行后面打印
                    current_show_position->x =current_show_position->x+show_size;
                else
                {           //需要换行的将定位按要求换行
                    current_show_position->x = show_area.lt.x;
                    current_show_position->y =current_show_position->y+show_size;

                }
```

```
                }

            }
        }
    }

    int ShowChTab(Area show_area,Coordinate current_show_position/*现在正在码的字的左上角坐标*/, int show_size/*有 48，36，28*/,int *qhwh)//7
    {
        union
        {
            int key;
            char c[2];
        }u;
        Coordinate first_show_position = current_show_position;
        CH * ch=NULL;//定义汉字符号结构体
        EN * en=NULL;//定义英文符号结构体
        char temp[500],temp1;                //定义临时变量缓存
        int i,ch_qhwh,en_qhwh,num_tab=0,num_qhwh=0;
        FILE *hanzi;                    //定义汉字符号文件指针
        FILE *fuhao;                    //定义英文符号文件指针

        Area            show_ch_area={          {ORIGINX+30,ORIGINY+365}          ,{ORIGINX+222,ORIGINY+380} };
        Area            show_en_area={          {ORIGINX+30,ORIGINY+388}          ,{ORIGINX+222,ORIGINY+400} };
        //Area input_tab_area={ {589,633},{1024,768} };
        Coordinate current_ch_position ={ORIGINX+30,ORIGINY+365};
        Coordinate current_en_position ={ORIGINX+30,ORIGINY+388};


        int real_rb_x= ((show_area.rb.x - show_area.lt.x)/show_size)*show_size + show_area.lt.x;
        //初始化 real_rb_x（最右点坐标），之后用最右值均用此

        if( ( ch=(CH *)malloc(sizeof(CH)*463) )==NULL )//如果动态分配内存失败，则直接返回
        {
            overflow_box(500,500);
            getch();
            perror("fail to malloc");
            return(1);
        }
        if( ( en=(EN *)malloc(sizeof(EN)*120) )==NULL )//如果动态分配内存失败，则直接
```

返回
```
        {
            overflow_box(500,500);
            getch();
            perror("fail to malloc");
            return(1);
        }

        if((hanzi = fopen("txt\\hanzi.txt","r")) == NULL)//hanzi 和 fuhao 是文件句柄
        {
            null_box(500,500);
            getch();
            perror("fail to open");
            return (1) ;
        }
        if((fuhao = fopen("txt\\fuhao.txt","r")) == NULL)
        {
            null_box(500,500);
            getch();
            perror("fail to open");
            return (1) ;
        }

        //StoreBk(input_tab_area.lt,435,135);        //保存 input_tab_area 的所有图像，1 左上角坐标 2 宽度 3 高度  这句就相当于 getimage
        //Putbmp64k(input_tab_area.lt.x,input_tab_area.lt.y,"bmp\\ChTab.bmp");  // 放输入框贴图

        SaveChQhwh(hanzi,ch);     //  初始化汉字输入库，将 hanzi.txt 里的汉字信息全部一个个转移到 ch 这个结构体中去
        SaveEnQhwh(fuhao,en);     //  初始化英文输入库，将 fuhao.txt 里的英文信息也全部一个个转移到 en 这个结构体中去

        num_qhwh=CheckQhwhNum(qhwh);//计算当前字符个数，总共有几个 qhwh 代表的字符，作用就是之后在整段聊天打印的时候需要
        //这里就是一开始光标找一下存在感
        CursorBlack(current_show_position,show_size);//光标涂黑（相当于出现光标）
        delay(400);
        //CursorWhite(current_show_position,show_size);//光标涂白（隐藏光标）

        ClearKey();//清除键盘缓存，不重要

        while(1)//进入按键循环
        {
```

```
u.key=bioskey(0);//获取按键信息，key 是 int 型（扫描码？

if(u.c[1]==0x1) //键入 esc 退出
{
    //归位
    free(ch);
    free(en);
    fclose(hanzi);
    fclose(fuhao);
    ClearKey();
    CursorWhite(current_show_position,show_size);//光标涂白
    //PutBk(input_tab_area.lt,435,135);          //把之前 StoreBk 的图片重新
putimage 上来
    return 0;
}

/**********
当 num_tab==0 时，
均对输入框操作
以下
**********/
else if(u.c[1]==0x1c&& num_tab==0)//键入回车发送消息
{
    //归位
    free(ch);
    free(en);
    ch = NULL;
    en = NULL;
    fclose(hanzi);
    fclose(fuhao);
    ClearKey();
    CursorWhite(current_show_position,show_size);//光标涂白
    //PutBk(input_tab_area.lt,435,135);          //把之前 StoreBk 的图片重新
putimage 上来
    return 3;


}


else if(u.c[1]==0x48 && num_tab==0)//键入 光标向上（UP 上键）
{
    if(current_show_position.y>show_area.lt.y )          //   判断之后是否会
```

```
出界
                    {
                        num_qhwh=num_qhwh-(real_rb_x-          show_area.lt.x)/show_size;
    //num_qhwh 减去整整一行的字符数
                        CursorWhite(current_show_position,show_size);
                        current_show_position.y-=show_size;
                        CursorBlack(current_show_position,show_size);
                    }

                }


                else if(u.c[1]==0x50 && num_tab==0)//键入 光标向下（DOWN 下键）
                {
                    if((current_show_position.y+2*show_size)<=show_area.rb.y) //这里*2 是
因为 current_show_position 定位的时左上角，故得判断顶端往下两行（即向下一行）的操作
之后不会出界
                    {
                        num_qhwh=num_qhwh+(real_rb_x-          show_area.lt.x)/show_size;
    //num_qhwh 加上整整一行的字符数
                        CursorWhite(current_show_position,show_size);
                        current_show_position.y+=show_size;
                        CursorBlack(current_show_position,show_size);
                    }

                }


                else if(u.c[1]==0x4b&& num_tab==0)//键入 光标向左（LEFT）
                {
                    if(          current_show_position.x          >          show_area.lt.x          ||
current_show_position.y>show_area.lt.y)//判断下一步不会出界
                    {
                        CursorWhite(current_show_position,show_size);
                        if(current_show_position.x > show_area.lt.x )//确认光标不会左移出输
入框
                            current_show_position.x = current_show_position.x-show_size;//
定位左移一个单位
                        else
                        {
                            //若左移出输入框则定位到上一行字符串尾
                            current_show_position.x =real_rb_x;
                            current_show_position.y =current_show_position.y-show_size;
```

```
                    }
                    --num_qhwh;//不论哪种情况，num_qhwh 都是要减 1（由键入 LEFT 键决定）
                    CursorBlack(current_show_position,show_size);//显示到现在定位的后面
                }

            }
            //在 y 处需要*2 也是 current)_show_position 定位在左上角惹的祸，用来判断下一行在不在界内
            else if(u.c[1]==0x4d&& num_tab==0)//键入 光标向右（RIGHT）
            {
                if((current_show_position.x+show_size)<=real_rb_x||(current_show_position.y+2*show_size)<=show_area.rb.y)//确认现在定位在正确框内
                {
                    CursorWhite(current_show_position,show_size);
                    if(current_show_position.x <= real_rb_x - show_size)//确认还在离最后面的字的定位有至少一个字以上的空间
                        current_show_position.x =current_show_position.x+show_size;//定位右移一个单位
                    else
                    {
                        //若左移出输入框则定位到下一行字符串首（似乎只对两行以上的输入方式有效
                        current_show_position.x = show_area.lt.x;
                        current_show_position.y =current_show_position.y+show_size;

                    }
                    ++num_qhwh;
                    CursorBlack(current_show_position,show_size);
                }

            }

            else if(u.c[1]==0xe && num_tab==0)//键入删除键（backspace），删除定位前一个字符
            {
                if(current_show_position.x>first_show_position.x || current_show_position.y>first_show_position.y)//如果不在第一个字的位置才可以进行
                {
                    CursorWhite(current_show_position,show_size);
                    DeleteShow( show_area,& current_show_position, show_size );//图像上的操作，逻辑上的操作见下一句
                    qhwh[num_qhwh-1]=0;//当前字符区号位号置 0
```

```
                --num_qhwh;//字符个数减去 1
            }
            CursorBlack(current_show_position,show_size);
        }

        else if(u.c[1]==0x39&& num_tab==0)//键入空格
        {
            if((current_show_position.x+show_size)<=real_rb_x ||
(current_show_position.y+2*show_size)<=show_area.rb.y)
            {
                CursorWhite(current_show_position,show_size);
                if(current_show_position.x <= real_rb_x - show_size)//确认还在离最
后面的字的定位有至少一个字以上的空间
                    current_show_position.x =current_show_position.x+show_size;//
图像上的空格，等下字符个数凭空加一
                else
                {
                    //满了就初始化重新输入
                    current_show_position.x = show_area.lt.x;
                    current_show_position.y =current_show_position.y+show_size;

                }
                ++num_qhwh;//字符个数+1
                CursorBlack(current_show_position,show_size);
            }
        }


        else if(u.c[0]>='a'&& u.c[0]<='z')//键入字母，注意 这里是拼音的字母
        {
            num_tab++;
            temp[num_tab-1]=(char) (u.c[0]);      //temp 的容量是 500，肯定够了
            ch_qhwh = FindChQhwh(ch,temp, num_tab,ch_qhwh);
            //去这个函数中与之前保存好的 ch 里面的所有汉字作比较，找到目标汉
字，
            //但此时有很多匹配值，所以在后面会全部 put 出来
            en_qhwh = FindEnQhwh(en,temp[num_tab-1]);     //到英文库中找到输
入的字母，直接显示出来
            if(num_tab<=6)    //限制了拼音最多输入 6 位
            ShowPerCharacter(en_qhwh,show_en_area,
&current_en_position ,0x000000,16,"hzk");//这里先把拼音的字母输出来
            //Putbmp64k(589,633,"bmp\\part.bmp");       //这里是输入法显示框（之
前是输入框操作
            bar(ORIGINX+25,ORIGINY+365,ORIGINX+225,ORIGINY+380,65535);
```

```
                current_ch_position.x = ORIGINX+30;//从这个位置输出可能匹配的汉字
                current_ch_position.y = ORIGINY+365;
                for(i=0;i<5;i++)    //一页显示五个可能值
                {
                    //显示序号 12345

ShowPerCharacter(217+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");
                    //显示汉字，拼音相同的两个汉字区位码相差 1

    ShowPerCharacter(ch_qhwh+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");
                }

            }


            else if(u.c[1]==0xe && num_tab!=0)//tab 框 键入删除键，删除输入法 tab 框
            {
                DeleteTab(& current_en_position, ch, en, temp ,& num_tab,& ch_qhwh);
            }


            else if(u.c[0]>='1'&&u.c[0]<='5'&& num_tab!=0)//键入 1-5   选择字符
            {
                CursorWhite(current_show_position,show_size);
                ++num_qhwh;             //为什么这里还需要让字符数加一。。。
                qhwh[num_qhwh-1]=ch_qhwh + u.c[0]-'1';
                /**********************
                u.c[0]-'1'是取出整型的输入数字（而不是 ascii 码），
                加上 ch_qhwh 之后就是指的是输入法框中的第几个汉字了
                **********************/

                ShowPerCharacter(qhwh[num_qhwh-1],show_area,
&current_show_position,0x000000 , show_size,"hzk");    //这次是打印到输入框中了
                //Putbmp64k(589,633,"bmp\\ChTab.bmp");    //重新刷图
                //put_image(ORIGINX+25,ORIGINY+365,ORIGINX+225,ORIGINY+400,
box_save);
                putsave_box(ORIGINX+25,ORIGINY+365);
                num_tab=0;  //拼音字母数归零
                current_ch_position.x = ORIGINX+30;//汉字出现位置归位
                current_ch_position.y = ORIGINY+365;
                current_en_position.x=ORIGINX+30;       //拼音出现位置归位
                current_en_position.y=ORIGINY+388;
```

```
                CursorBlack(current_show_position,show_size);//黑色光标重新出现

            }

            else if((u.c[1]==0x48/*UP*/||u.c[1]==0x4b/*LEFT*/)&& num_tab!=0)//键入 向
上或向左 翻页
            {
                ch_qhwh=ch_qhwh-5;        //汉字库中往前走五个（刚好翻页
                //Putbmp64k(589,633,"bmp\\part.bmp"); //又开始重新 put 所有信息
                bar(ORIGINX+25,ORIGINY+365,ORIGINX+225,ORIGINY+380,65535);

                current_ch_position.x = ORIGINX+30;
                current_ch_position.y = ORIGINY+365;
                for(i=0;i<5;i++)
                {

ShowPerCharacter(217+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");

ShowPerCharacter(ch_qhwh+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");
                }

            }

            else if((u.c[1]==0x50/*DOWN*/||u.c[1]==0x4d/*RIGHT*/) && num_tab!=0)//键
入 向下翻页
            {
                ch_qhwh=ch_qhwh+5;        //汉字库中往后走五个（刚好翻页
                //Putbmp64k(589,633,"bmp\\part.bmp"); //又开始重新 put 所有信息
                bar(ORIGINX+25,ORIGINY+365,ORIGINX+225,ORIGINY+380,65535);

                current_ch_position.x = ORIGINX+30;
                current_ch_position.y = ORIGINY+365;
                for(i=0;i<5;i++)
                {

ShowPerCharacter(217+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");

ShowPerCharacter(ch_qhwh+i,show_ch_area,&current_ch_position,0x000000,16,"hzk");
                }
            }

            else       //直接输出英文（不知道对应什么情况
            {
                temp1=(char) (u.c[0]);
```

```
                    en_qhwh = FindEnQhwh(en,temp1);
                    CursorWhite(current_show_position,show_size);
                    ++num_qhwh;
                    qhwh[num_qhwh-1]=en_qhwh;
                    ShowPerCharacter(qhwh[num_qhwh-1],show_area,
&current_show_position ,0x000000,show_size,"hzk");
                    CursorBlack(current_show_position,show_size);

                }
            }
        }
```

----------chatQhwh.c----------

```
#include "title.h"
/******************函数清单**************************
1.   void SaveChQhwh(FILE *fp,CH * ch)
2.   int FindChQhwh(CH *ch,char *temp,int num_tab,int qhwh)
3. int CheckQhwhNum(int *qhwh)
4.    void qhwh2incode(int qwh,char *incode)
*************************************************/
void SaveChQhwh(FILE *fp,CH * ch)//1
{
    int i=0;
    while(!feof(fp)&&i<463)
    {
        fscanf(fp,"%s", ch[i].str );//读取字符

        fscanf(fp,"%d",&(ch[i].qhwh));//读取区号位号

        i++;
    }
    fclose(fp);
}

void SaveEnQhwh(FILE *fp,EN *en)//2
{
    int i=0;
    while(!feof(fp)&&i<120)
    {
        fscanf(fp,"%s",&(en[i].str));//读取字符

        fscanf(fp,"%d",&(en[i].qhwh));//读取区号位号

        i++;
```

```
    }
    fclose(fp);
}

int FindChQhwh(CH *ch,char *temp,int num_tab,int qhwh)//3
{
    int   i;
    //对字符进行比对，找到则进行赋值
    for(i=0;i<463;i++)
    {
        if(strncmp(temp,ch[i].str,num_tab)==0)
            {
                qhwh=ch[i].qhwh;
                break;
            }
    }
    return qhwh;
}

int FindEnQhwh(EN *en,char temp)//4
{
    int i,qhwh;
    //对字符进行比对，找到则进行赋值
    for(i=0;i<120;i++)
    {
        if(strncmp( &temp,&(en[i].str),1)==0)
            {
                qhwh=en[i].qhwh;
                break;
            }
    }
    return qhwh;
}

/******************
将区号位号从文件中读取出来放到 qhwh 里，传回函数中打印
CheckQhwhNum 确认要写入的信息量
******************/
int TxtToQhwh(int *qhwh,char *filename)//5
{
    int i=0,j=0;
    char temp;
    FILE *txt;
    if((txt=fopen(filename,"r"))==NULL)
```

```
    {
        perror("fail to open");
        return (1) ;
    }
    while((!feof(txt))&&(i<MAXTXT))
    {
        for(j=0;j<4;j++)
        {
            temp=fgetc(txt);          //从 txt 中获取下一个字符（一个无符号字符），并把位置标识符往前移动
            qhwh[i]=qhwh[i]*10;
            qhwh[i]=qhwh[i]+ atoi(&temp);
        }
        i++;
    }
    fclose(txt);
    return 0;
}

int AddQhwhToTxt(int *qhwh,char *filename)//6
{
    int i=0;
    FILE *txt;
    if((txt=fopen(filename,"a"))==NULL)
    {
        perror("fail to open");
        return (1) ;
    }
    for(i=0;i<CheckQhwhNum(qhwh);i++)
    {
        fprintf(txt,"%4d", qhwh[i] );
    }
    fprintf(txt,"\n");
    fclose(txt);
    return 0;
}


/******************
将信息以区位号的形式写入文件中
CheckQhwhNum 确认要写入的信息量
******************/
int ReplaceQhwhToTxt(int *qhwh,char *filename)//7
{
```

```
    int i=0;
    FILE *txt;
    if((txt=fopen(filename,"w"))==NULL)
    {
        perror("fail to open");
        return (1) ;
    }
    for(i=0;i<CheckQhwhNum(qhwh);i++)
    {
       fprintf(txt,"%4d", qhwh[i] );
    }
    fprintf(txt, "\n" );    //输入完后换行
    fclose(txt);
    return 0;
}


/******************
将信息以整型（占位 4 个字符）写入文件中
通过 CheckQhwhNum 确定要写入的信息量
******************/
int ReplacePassToTxt(int *qhwh,char *filename)//8
{
    int i=0;
    FILE *txt;
    if((txt=fopen(filename,"w"))==NULL)
    {
        perror("fail to open");
        return (1) ;
    }
    for(i=0;i<CheckQhwhNum(qhwh);i++)
    {
       fprintf(txt,"%4d", qhwh[i] );
    }
    fclose(txt);
    return 0;
}

int AddNumToTxt(int num,char *filename)//9
{
    FILE *txt;
    if((txt=fopen(filename,"a"))==NULL)
    {
        perror("fail to open");
```

```
            return (1) ;
        }
        fprintf(txt,"%d", num );
        fclose(txt);
        return 0;
}

int CheckNumInTxt(char *filename)//10
{
        int i=0;
        char temp;
        FILE *txt;
        if((txt=fopen(filename,"r"))==NULL)
        {
            perror("fail to open");
            return (1) ;
        }
        while((!feof(txt)))
        {
            fgetc(txt);
            ++i;
        }
        fclose(txt);
        return i/2;
}

/**************
看看有几个区号位号（几个字）
**************/
int CheckQhwhNum(int *qhwh)//11
{
        int num_qhwh=0,i=MAXTXT;
        while(i >= 1)
        {
            if(qhwh[i-1]== 0)
            {
                ++num_qhwh;
                --i;

            }
            else break;
        }

        return (MAXTXT-num_qhwh);
```

```
}

void qhwh2incode(int qwh,char *incode)
{
	char qh,wh;
	qh = (char)(qwh/100);
	wh = (char)(qwh%100);
	incode[0] = qh+0xa0;
	incode[1] = wh+0xa0;
}

void ClearKey(void)//2
{
	union REGS key;

	key.h.ah = 0x0c;
	key.h.al = 0x00;
	int86(0x21, &key, &key);
}


/**************
区号位号归零操作
每次调用输入法都要使用
**************/
int QhwhToZero(int *qhwh)//6
{
	int i;
	for(i=0;i<MAXTXT;i++)
		qhwh[i]=0;
	return 0;
}
```

----------chatShow.c----------

```
#include "title.h"



//功能：打印用户命令
//输入：区号位号和输出框的纵坐标
//输出：无
void show_order(int *qhwh,int *y)
{
```

```
    int num_qhwh;      //字符数
    int num_line;  //需打印的行数
    int box_top;   //输入框上沿

    Area order_box;
    Coordinate order_begin;

    /***************
    检查有几个字并画对话框底
    ***************/
    num_qhwh = CheckQhwhNum(qhwh);     //得到字符数
    num_line = num_qhwh/10+1;          //确认打印行数
    box_top = 345-20*num_line;         //确认对话框上沿

    if(num_qhwh<=10)
    {
        bar_round_2(ORIGINX+239-num_qhwh*16-
4,ORIGINY+box_top,ORIGINX+242,ORIGINY+345,5,1,65535);     //打印对话框
    }
    else
    {
        bar_round_2(ORIGINX+239-10*16-
4,ORIGINY+box_top,ORIGINX+242,ORIGINY+345,5,1,65535);
    }

    /***************
    打印汉字并打印用户姓名
    ***************/
    if(num_qhwh<=10)
    {
        order_box.lt.x = ORIGINX+239-num_qhwh*16-4;
        order_box.lt.y = ORIGINY+box_top;
        order_box.rb.x = ORIGINX+243;
        order_box.rb.y = ORIGINY+345;
        order_begin.x = ORIGINX+239-num_qhwh*16;
        order_begin.y = ORIGINY+box_top+2;

        ShowTxt(qhwh, order_box, &order_begin,16);
    }
    else{
        order_box.lt.x = ORIGINX+239-10*16-4;
        order_box.lt.y = ORIGINY+box_top;
        order_box.rb.x = ORIGINX+243;
        order_box.rb.y = ORIGINY+345;
```

```
            order_begin.x = ORIGINX+239-10*16;
            order_begin.y = ORIGINY+box_top+2;

            ShowTxt(qhwh, order_box, &order_begin,16);
        }

        fdhz(ORIGINX+243-16*2,ORIGINY+box_top-20,1,1,"主",65535);
        fdhz(ORIGINX+243-16*1,ORIGINY+box_top-20,1,1,"人",65535);

        /*************
        改变回复区域的占位记录值
        *************/
        //*y = ORIGINY+box_top-25;
        *y = ORIGINY+50+20*num_line+25;
}

//功能：得到用户输入的字符串（将区号位号改成内码）
//输入：区号位号指针和内码指针
//输出：无
void get_str(int *qhwh, char *incode)
{
        char *p=incode;    //incode 的位置指针
        int num_qhwh = CheckQhwhNum(qhwh);
        int i;

        for(i=0;i<num_qhwh;i++)
        {
            qhwh2incode(*qhwh,p);
            p+=2;
            qhwh++;
        }
        *p = 0;
        *(p+1)=0;
}

//功能：根据输入的字符串得到返回语句
//输入：用户输入的字符串的指针，回复的字符串的指针，机器人的状态指针
//输出：无
void reply_match(char * str,char *reply, CASE* robot)
{
        int k;
        char *t_str=str;
        srand((unsigned int)time(0));
        while (*t_str!=0)
```

```
{
    if (!strncmp(t_str,"吃",2))
    {
        k=rand()%4;
        if (k==0)
            reply="火锅怎么样";
        else if (k==1)
            reply="我觉得还是牛排最好吃";
        else if (k==2)
            reply="还是推荐蔬菜沙拉";
        else if (k==3)
            reply="你都这么胖了还想吃";
    }
    else if (!strncmp(t_str,"气",2))
    {
        k=rand()%4;
        if (k==0)
            reply="阳光明媚，适合泡妹";
        else if (k==1)
            reply="外面下着小雨，建议出门带伞";
        else if (k==2)
            reply="大暴雨，还是开车吧";
        else if (k==3)
            reply="阴天，凉爽舒适，是个户外运动的好日子";
    }
    else if (!strncmp(t_str,"你",2))
    {
        k=rand()%3;
        if (k==0)
            reply="我是靠谱的智能机器人，让我来协助你生活是个明智的选择";
        else if (k==1)
            reply="我是智能机器人，请放心地将你的生活交给我";
        else if (k==2)
            reply="我是一个体贴的机器人，而且很智能";
    }
    else if (!strncmp(t_str,"游",2))
    {
        k=rand()%3;
        if (k==0)
            reply="我不具有游戏功能，去蒸汽游戏平台吧";
        else if (k==1)
            reply="你玩游戏玩得太多啦，今天还是歇歇吧";
        else if (k==2)
            reply="听说最近他们都在玩吃鸡呢，你要去试试吗？";
```

```
            }
            else if (!strncmp(t_str,"人",2))
            {
                k=rand()%1;
                if (k==0)
                    reply="除了你我只认识肖力文和袁立凡他们是所有人里最帅的";
            }
            else
                reply=NULL;
            t_str+=2;
            if (reply!=NULL)
            {
                saveimage_chat(robot->x*40+40,robot->y*40);
                show_reply(reply,robot);      //开始显示回复消息
                getch();
                putsave_chat(robot->x*40+40,robot->y*40);
            }
            reply=NULL;
        }
        if (no_keyword(str))
        {
            reply="嗯嗯";
            saveimage_chat(robot->x*40+40,robot->y*40);
            show_reply(reply,robot);      //开始显示回复消息
            getch();
            putsave_chat(robot->x*40+40,robot->y*40);
        }
}

//功能：机器人回复输出
//输入：输出语句的内码，机器人的状态
//输出：无
void show_reply(char *incode,CASE* robot)
{
        int x=robot->x*40+40+10,y;      //打印回复汉字的位置
        int box_top;   //对话框底顶部的纵坐标
        int i;              //temp
        int num_word=0;  //需要回复的字数
        int num_line;  //回复几行
        char *p_incode=incode;      //incode 的位置指针

        //检查有几个字
        while(*p_incode!=NULL)
        {
```

```
            p_incode+=2;
            num_word++;
        }


        //画对话框底

        bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+240,robot->y*40+80,5,2,655
35);

        //打印回复的汉字

        i = 0;      //作字数标记
        y = robot->y*40+5;
        while(*incode!=NULL)
        {
            while( (i<10)&&(*incode!=NULL) )
            {
                dishz(x,y,1,1,incode,0);
                i++;
                x+=16*1;
                incode+=2;
            }
            i=0;
            x=robot->x*40+40+10;
            y+=20*1;
        }


}



//功能：聊天输出的主函数
//输入：用户输入的字符串的内码，机器人状态
//输出：无
void show_main(int *qhwh, CASE* robot)
{
        int num_qhwh;      //统计输入的字符数量
        int begin_y;    //标记每次对话框输出时的顶端 y 坐标以确定 putimage 的范围
        char *reply =NULL;      //之后标记到文件中的答句部分
        char incode[30] ={0};

        //显示用户输入部分
```

```
        save_image(ORIGINX+13, ORIGINY+50, ORIGINX+243+1,ORIGINY+345+1);
        bar(ORIGINX+13,ORIGINY+250,ORIGINX+243,ORIGINY+345,54938);  //清空要显示用户输入消息的区域
        show_order(qhwh,&begin_y);      //开始显示用户输入消息
        printf_image_2(ORIGINX+13, ORIGINY+50, ORIGINX+243+1, ORIGINY+345+1,begin_y);

        delay(500);     //等待机器人回复

        //显示机器人回复部分
        //寻找匹配句
        get_str( qhwh, incode);
        reply_match( incode, reply,robot);
}


//功能：判断用户输入的句子中是否有关键字
//输入：用户输入的字符串的字符指针
//输出：int
//          返回 0：没有关键字
//          返回 1：有关键字
int no_keyword(char *str)
{
        int p=1;
        char *t_str=str;
        while (*t_str!=0)
        {
                if  (!strncmp(t_str,"吃",2)     ||     !strncmp(t_str,"气",2)     ||     !strncmp(t_str,"你",2)
||    !strncmp(t_str,"游",2)    ||    !strncmp(t_str,"人",2)    )
                        p=0;
                t_str+=2;
        }
        return    p;
}
```

----------color1.c----------

```
#include "title.h"
/*****************************************************
颜色转换及渐变色相关函数
int transcolor(int r,int g,int b)
void linelevel_color(int x1,int y1,int x2,int y2,int r1,int g1,int b1,int r2,int g2,int b2)
void linever_color(int x1,int y1,int x2,int y2,int r1,int g1,int b1,int r2,int g2,int b2)
void circle_color(int x0,int y0,int r1,int g1,int b1,int r2,int g2,int b2,int r)
*****************************************************/
```

```
/*将 rgb 颜色值转换为 64k 色*/
int transcolor(int r,int g,int b)
{
	int rgb;
	/*
	double r_mid,g_mid,b_mid;
	r_mid = (32*(double)(r)/255);
	g_mid = (64*(double)(g)/255);
	b_mid = (32*(double)(b)/255);
	rgb = ((int)(r_mid)<<11) || ((int)(g_mid)<<5) || ((int)(b_mid));

	*/
	/*0x117 模式下，原图红绿蓝各 8 位分别近似为 5 位、6 位、5 位*/
		r >>= 3;
		g >>= 2;
		b >>= 3;
		rgb = ((((unsigned int)r) << 11)
		| (((unsigned int)g) << 5)
		| ((unsigned int)b));
		return rgb;
}


/*竖直线性渐变*/
void linever_color(int x1,int y1,int x2,int y2,int r1,int g1,int b1,int r2,int g2,int b2)
{
	float fbegin = 0.0,fend = 0.0,rstep = 0.0,gstep = 0.0,bstep = 0.0;
	unsigned int rRed,rGreen,rBlue,rgb;
	int x,y,wide=y2-y1;
	for(y=y1;y<y2;y++)
	{
		fbegin = (float)(r1);
		fend = (float)(r2);
		rstep = (fend-fbegin)/(wide);
		rRed = (int)(fbegin+rstep*y);

		//green
		fbegin = (float)(g1);
		fend = (float)(g2);
		gstep = (fend-fbegin)/(wide);
		rGreen = (int)(fbegin+gstep*y);

		//blue
```

```
            fbegin = (float)(b1);
            fend = (float)(b2);
            bstep = (fend-fbegin)/(wide);
            rBlue = (int)(fbegin+bstep*y);
            rgb = transcolor(rRed,rGreen,rBlue);
            for(x=x1;x<=x2;x++)
            {
                  Putpixel64k(x,y,rgb);
            }

      }
}
```

----------findway.c----------

```
#include "title.h"
/*************************************************************
函数列表：1.队列的基本操作
                  void InitQueue(LinkQueue *Q)        //队的初始化
                  void DestroyQueue(LinkQueue *Q)   //销毁队
                  int IsEmpty(LinkQueue Q)              //判断队是否为空
                  void EnQueue(LinkQueue *Q, QElemtype e)        //入队
                  void DeQueue(LinkQueue *Q, QElemtype *e)       //出队
                2.图的基本操作
                  int CreateGraph(Graph G)              //图的初始化根据给定地图，以链表的方式构建图
                  int FindAdjVex(Graph const G, const int n, int k, QElemtype *adjvex)      //寻找某点在图中的邻接点
                  int LocateVex(Graph const G, const int n, Axis V)                         //给定某点的数据值，确定该点在图中的索引
                  int FindWay(Graph const G, PathType *path, const int n, Axis V0, Axis Vt) //Dijkstra 算法找到从 V0 到 Vt 的最短路径
                3.栈的基本操作
                  void InitStack(LkStack *S)          //栈的初始化
                  void DestroyStack(LkStack *S)     //销毁栈
                  void Push(LkStack *S, SElemtype e)   //入栈操作
                  void Pop(LkStack *S, SElemtype *e)   //出栈操作
*************************************************************/

//传入队指针完成队的初始化操作，即：先申请一个队节点，并让队头、队尾指向该节点
void InitQueue(LinkQueue *Q)
{
      Q->rear = (QueuePtr)malloc(sizeof(Qnode));        //指向头节点，不储存信息
      Q->front = Q->rear;
```

```
    if(!Q->front)      //分配失败
    {
        overflow_box(500, 500);
    }

    Q->front->next = NULL;
}

//传入队指针销毁队，即：将元素逐个出队后释放节点空间，直到队空
void DestroyQueue(LinkQueue *Q)        //从队头开始往队尾走，一个个释放
{
    while(Q->front)   //队还不空
    {
        Q->rear = Q->front->next;
        free(Q->front);
        Q->front = Q->rear;
    }
}

//根据队头、队尾是否指向同一节点，判断队是否为空
int IsEmpty(LinkQueue Q)
{
    if(Q.front == Q.rear)      //队头、队尾重合，就是队空
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

//将数据 e 储存，即入队，申请一个队节点，存放元素 e 信息后，插在队尾
void EnQueue(LinkQueue *Q, QElemtype e) //e 插入队尾
{
    QueuePtr p;
    p = (QueuePtr)malloc(sizeof(Qnode));
    if(!p)
    {
        overflow_box(500, 500);
    }
    p->data = e;
    p->next = NULL;
    Q->rear->next = p;      //连接到队尾
```

```
    Q->rear = Q->rear->next;                //队尾往后移动
}

//队头元素出队，传入指针 e 保存队头元素的信息，然后释放队头节点
void DeQueue(LinkQueue *Q, QElemtype *e)
{
    QueuePtr p;
    if(Q->front == Q->rear)
    {
        return ;            //队空，删除失败，退出函数
    }
    p = Q->front->next;
    *e = p->data;           //保存要出队的数据
    Q->front->next = p->next;     //队首后移
    if(Q->rear == p)        //删除之后，队空
    {
        Q->rear = Q->front;
    }
    free(p);     //删除以后就应该释放原结点
    p = NULL;
}

/*********队的基本操作结束，下面是图*************/
//入口是数组 G，在 main 函数中声明，将给定地图中每个可通行的坐标点放进数组中。结构体数组里有一个指针，依次、按方向顺次指向其可通行的邻接点
int CreateGraph(Graph G)
{
    int i, j, k = 0;
    LNode *p = NULL, *last = NULL;          //p 是每次加的临时节点，last 指向每个 G[k]接的链表的最后一个节点
    const char unaccessible[16][19]={
                                        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},   //第一列
                                        {1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1},
                                        {1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1},
                                        {1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                                        {1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                                        {1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                                        {1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                                        {1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
                                        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                                        {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                                        {1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},
                                        {1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1},
```

```
                                    {1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1},
                                    {1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1},
                                    {1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1},
                                    {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},   //第 16
列
                                };

    /*建立邻接链表来储存图的信息*/
    for(i = 1; i < 15; i++) //边界肯定不通，不浪费时间去遍历了
    {
        for(j = 1; j < 18; j++)//边界肯定不通，不浪费时间去遍历了
        {
            if(!unaccessible[i][j])    //这个点可以通过，则加到图中。不能通过就舍弃
            {
                //给 G[k]初始化，k 自增
                G[k].x = i;
                G[k].y = j;
                G[k].next = NULL;
                last = G[k].next;    //初始化指向 G[k]下一位，是 NULL，但可以简化后面
的操作
                //依次判断右边、左边、上面、下面可不可以通行
                if(i < 14 && !unaccessible[i+1][j])      //右边有方格且可以向右通行，自动
去除了边界（i+1 小于 15，确保了不会访问到外界）
                {
                    p = (LNode *)malloc(sizeof(LNode)); //为 p 申请空间
                    if(!p)
                    {
                        overflow_box(500,500);
                        getch();
                        exit(1);
                    }
                    p->x = i+1;                //初始化 p，p 是 G[k]点右边的点的坐标
                    p->y = j;
                    p->direction = 1;    //记录方向表示 p 点在 G[k]右边
                    p->next = NULL;
                    if(!last)          //如果这是第一个邻接点则 G[k].next 直接连上去，否则
靠 last
                    {
                        G[k].next = p;
                    }
                    else
                    {
                        last->next = p;
                    }
```

```
            last = p;    //不管是不是第一个结点，last 都得指向 p，因为 last 指向最后一个
        }

        if(i > 1 && !unaccessible[i-1][j])      //左边有方格且可以向右通行，自动去除了边界（i-1>0。确保不会乱访问）
        {
            p = (LNode *)malloc(sizeof(LNode));
            if(!p)
            {
                overflow_box(500,500);
                getch();
                exit(1);
            }
            p->x = i-1;         //初始化 p，p 是 G[k]点左边的点的坐标
            p->y = j;
            p->direction = 2;    //记录方向表示 p 点在 G[k]左边
            p->next = NULL;
            if(!last)           //如果这是第一个邻接点则 G[k].next 直接连上去，否则靠 last
            {
                G[k].next = p;
            }
            else
            {
                last->next = p;
            }
            last = p;    //不管是不是第一个结点，last 都得指向 p，因为 last 指向最后一个
        }

        if(j > 1 && !unaccessible[i][j-1])      //上面，自动去除了边界（j-1>0，不会越界）
        {
            p = (LNode *)malloc(sizeof(LNode));
            if(!p)
            {
                overflow_box(500,500);
                getch();
                exit(1);
            }
            p->x = i;           //初始化 p，p 是 G[k]点上面的点的坐标
            p->y = j-1;
            p->direction = 3;      //记录方向表示 p 点在 G[k]上面
```

```
                p->next = NULL;
                if(!last)          //如果这是第一个邻接点则 G[k].next 直接连上去，否则
靠 last
                {
                    G[k].next = p;
                }
                else
                {
                    last->next = p;
                }
                last = p;    //不管是不是第一个结点，last 都得指向 p，因为 last 指
向最后一个
            }

            if(j < 17 && !unaccessible[i][j+1])      //下面，自动去除了边界（j+1<18，
不会越界）
            {
                p = (LNode *)malloc(sizeof(LNode));
                if(!p)
                {
                    overflow_box(500,500);
                    getch();
                    exit(1);
                }
                p->x = i;          //初始化 p，p 是 G[k]点下面的点的坐标
                p->y = j+1;
                p->direction = 4;    //记录方向表示 p 点在 G[k]下面
                p->next = NULL;
                if(!last)          //如果这是第一个邻接点则 G[k].next 直接连上去，否则
靠 last
                {
                    G[k].next = p;
                }
                else
                {
                    last->next = p;
                }
                last = p;    //不管是不是第一个结点，last 都得指向 p，因为 last 指
向最后一个
            }

            k++;                //图中个数+1
        }
    }
```

```
    }
    return k;    //返回 G 中元素的个数
}

//入口参数：索引 k 和 adjvex[4]数组，数组各分量依次储存 G[k]右、左、上、下四个方向的可通行邻接点在 G 中的坐标
//函数功能：找 G[k]在 G 中的邻接点，储存在 adjvex[4]中，放的是 x，y 坐标
int FindAdjVex(Graph const G, const int n, int k, QElemtype *adjvex)
{    //adjvex 中第一个分量是向右的邻接点，第二个向左，以此类推。n 是图中点的个数，k 是邻接表中的下标
    int i, j;          //i 用来表示 adjvex 的下标，j 表示 LocateVex 的返回值
    LNode *p = NULL;
    Axis V;

    p = G[k].next; //指向 G 的第一个邻接点
    if(!p)         //没有邻接点
    {
        return 0;          //图里有这个点，但它没邻接点
    }

    while(p)    //p 非空，没到末尾
    {
        i = p->direction - 1;     //按方向存，direction 是从 1 开始的，而 i 代表下标，从 0 开始
        V.x = p->x;                  //V 就是 p 代表的点，也即 G[k]的邻接点
        V.y = p->y;
        p = p->next;             //往后移，找下一个邻接点
        j = LocateVex(G, n, V); //找 V 在 G 中的标号
        adjvex[i] = j;
    }
    return 1;                //找到了邻接点
}

//寻找 V 在 G 中的标号
int LocateVex(Graph const G, const int n, Axis V)
{
    int i;
    for(i = 0; i < n; i++)
    {
        if(G[i].x == V.x && G[i].y == V.y)
        {
            return i;                          //G 中找到 V 点后，把 V 在 G 中的下标返回
        }
    }
```

```
    return -1;  //没找到
}

//入口参数：路径 path，起点 V0,终点 Vt，图 G 及 G 的元素个数 n
//函数功能：根据 Dijkstra 算法得到逆序最短路径 path
///得到 path 之后，先入栈再出栈，就得到了我们想要的从 V0 到 Vt 的路径
int FindWay(Graph const G, PathType *path, const int n, Axis V0, Axis Vt)
{
    int i, j, w, *dist, v0, vt;    //v0，vt 是 V0 和 Vt 在 G 中的下标，w 是中间点的下标
    LinkQueue Q;
    QElemtype k, adjvex[4]; //adjvex 里面存的是下标，是要入队出队的，因此属于队的元素类型
    InitQueue(&Q);

    v0 = LocateVex(G, n, V0);
    vt = LocateVex(G, n, Vt);
    if(v0 == -1 || vt == -1)            //起始点、终止点错误，不玩了直接退出函数，啥也不干
    {
        return 0;           //error
    }
    dist = (int *)malloc(n * sizeof(int));
    for(i = 0; i < n; i++)
    {
        dist[i] = -1;
        path[i].former = -1;
    }
    dist[v0] = 0;

    EnQueue(&Q, v0);
    while(!IsEmpty(Q))
    {   //每个元素出队前，adjvex 重新初始化，元素出队后遍历其邻接点时要用
        adjvex[0] = -1;
        adjvex[1] = -1;
        adjvex[2] = -1;
        adjvex[3] = -1;
        DeQueue(&Q, &k);      //k 是队首元素，而队里放的是在图 G 中的编号
        if(FindAdjVex(G, n, k, adjvex)) //没邻接点的话返回 0，直接跳掉，如果有的话，按方向储存其邻接点的标号置于 adjvex 数组
        {
            for(j = 0; j < 4; j++)          //依次向四个方向遍历
            {
                if((w = adjvex[j]) != -1)    //说明这个方向的这条路通，k 有 w 这个邻接点
```

```
                {
                    if(dist[w] == -1)           //说明这条路不但是通的，还没被访问过
                    {
                        dist[w] = dist[k] + 1;  //V0 到 W 的最短距离是 V0 到 K 的最短距离+1
                        path[w].former = k;             //w 的路径上有 k 这一点
                        path[w].direction = j + 1;  //记录这是由 k 代表的点向右走一步来的，还是向左来的，等等
                        EnQueue(&Q, w);                 //在 dist 中标记过这个点后入队

                    }

                    if(w == vt)             //找到终点了，不浪费时间继续找了
                    {
                        path[v0].former=-1;
                        DestroyQueue(&Q);   //使用完毕，释放以节约内存
                        free(dist);
                        dist = NULL;
                        return 1;;
                    }
                }
            }
        }
    }
}

/****栈****/
//栈的作用仅在于将 path 逆序，不需要其他操作，因此只写四个函数

//初始化栈
void InitStack(LkStack *S)
{
    S->bottom = (SNode *)malloc(sizeof(SNode));     //申请头结点
    if(!S->bottom)
    {
        overflow_box(500, 500);
    }
    S->top = S->bottom;         //置空
}

//销毁栈
void DestroyStack(LkStack *S)
{
    SNode *p;
```

```
    while(S->top != S->bottom)   //栈还没空
    {
        p = S->top;
        S->top = p->next;
        free(p);
    }
    p = NULL;

}

//入栈
void Push(LkStack *S, SElemtype e)
{
    SNode *p;
    p = (SNode *)malloc(sizeof(SNode));
    p->data = e;
    p->next = S->top;
    S->top = p;
}

//出栈
void Pop(LkStack *S, SElemtype *e)
{
    SNode *p;
    p = S->top;
    S->top = p->next;
    *e = p->data;
    free(p);
    p = NULL;
}
```

----------finger.c----------

```
#include "title.h"

//功能：指纹验证函数
//输入：无
//输出：int
//        返回 1：验证成功（验证成功的按键是 enter 键）
int finger_check()
{
    int button=0,x=0,y=0,state=1;
    int key;        //表示键值的变量
```

```
    bar(0,0,1024,768,65535);
    iph_frame();
    fdhz(816,119,1,1,"请",0);
    fdhz(852,119,1,1,"验",0);
    fdhz(892,119,1,1,"证",0);
    fdhz(928,119,1,1,"指",0);
    fdhz(964,119,1,1,"纹",0);

    FillCircle(896,394,25,26620);
    while(1)
    {
        key=0;

        /*吸收键盘缓冲区数据*/
        if (kbhit() != 0)
        {
            key = bioskey(0);
        }

        /*按 esc 则退出*/
        if(key == 0x11b)
        {
            exit(0);
        }
        else if (key==0x1c0d)
        {
            bar(840,139,1024,159,65369);
            return 1;
        }
        else if (key!=0)
        {
            fdhz(856,139,1,1,"指纹不正确",0);
        }
    }
}
```

----------frnture.c----------

```
#include "title.h"
/*******************************************************************************
```

该源文件中存放所有家具的源代码

入口参数：1. 均为都是要画的地方的左上角，但很多函数中都有根据具体情况赋予一定的偏移量

2. 为寻路方便，大部分家具的入口参数是 40*40 的大坐标，刚好占据一个 block 坐标块

3. 很多函数里的 x *= 40 和 y *= 40 的操作都是将大坐标转化为小坐标。入口是像素点小坐标的部分会额外强调

函数列表：

```
void trashbin(int x,int y)          //垃圾桶
void bed(int x,int y)               //床
void window_close(int x,int y)      //窗户
void cupboard(int x,int y)          //衣柜
void aircon(int x,int y,int open)   //空调
void WashMach(int x,int y)          //40*40，洗衣机
void bookshelf(int x,int y)         //40*80，书架
void desk(int x,int y)              //80*40，小方桌
void seat(int x,int y)              //40*40，小板凳
void trash1(int x,int y)            //40*40，纸张
void trash2(int x,int y)            //40*40，菜叶
void trash3(int x,int y)            //40*40，果核
void pc(int x,int y)                //电脑
void TV(int x,int y)                //电视
void rect_table(int x,int y)        //条纹方卓
void sofa_main(int x, int y)        //沙发主要部分
void sofa_up(int x, int y)          //沙发，屏幕上面的那组
void sofa_down(int x, int y)        //沙发，屏幕下面的那组
void toilet(int x, int y)           //40*40，马桶
void water_dispenser(int x, int y)  //40*80，饮水机
void zaotai(int x, int y)           //灶台
void water_bottle(int x, int y)     //水杯，入口是像素点小坐标
void clothes(int x, int y)          //衣服（机器人去衣柜拿到浴室）
void plate(int x,int y)             //盘子
void medical_kit(int x, int y)      //医疗包
void TV_on(void)                    //电视打开状态
void TV_off(void)                   //电视关闭（和上面 TV 的不同之处是，要做额外绘画去恢复客厅模样）
void music_on(int x, int y)         //音符
void music_off( int x, int y)       //去除音符
*****************************************************************************/

void trashbin(int x,int y)          //垃圾桶，入口参数是垃圾桶的左上角
{
    x = x*40;
    y = y*40+4;
    bar(x+15,y,x+35,y+40,65187);
    ever_Fillellipse(x+23,y,x+27,y,8,65187);
    ever_Fillellipse(x+23,y,x+27,y,6,0);
    ever_Fillellipse(x+23,y,x+27,y,5,65535);
}
```

```
void bed(int x,int y)            //床
{
      x = x*40+10;                    //为防止图像溢出屏幕，多给 10 个偏移量
      y = y*40+4+30;                  //+30
      theta_bar(x-5,y-48,30,4,-45,62604);
      theta_bar(x+20,y-50,32,4,45,62604);
      lean_line(x,y-48,4,-45,0);
      lean_line(x+48,y-50,4,45,0);
      bar(x-5,y-48,x+55,y-32,52263);
      bar(x-5,y-32,x+55,y+42,65535);
      bar(x+4,y-35,x+46,y-15,0);
      bar(x+5,y-35,x+45,y-15,65535);
      fill_bow_right_down(x+25,y-48,20,52263);
      fill_bow_left_down(x+25,y-48,20,52263);
      fill_bow_up(x+25,y+21,43,64950);
      bar(x-3,y-7,x+53,y+10,64950);
      fill_bow_up(x+25,y+28,40,65535);
      bar(x-4,y,x+54,y+42,0);
      bar(x-3,y,x+53,y+42,65535);
      bar(x-5,y+53,x+2,y+59,52263);
      bar(x+48,y+53,x+55,y+59,52263);
      bar(x-5,y+42,x+55,y+55,44373);
      bar(x-5,y+42,x+2,y+55,57083);
      bar(x+11,y+42,x+18,y+55,57083);
      bar(x+32,y+42,x+39,y+55,57083);
      bar(x+48,y+42,x+55,y+55,57083);
      linelevel(x-3,y+42,x+53,y+42,3,42260);
      linever(x+2,y+45,x+2,y+55,1,0);
      linever(x+11,y+45,x+11,y+55,1,0);
      linever(x+18,y+45,x+18,y+55,1,0);
      linever(x+32,y+45,x+32,y+55,1,0);
      linever(x+39,y+45,x+39,y+55,1,0);
      linever(x+48,y+45,x+48,y+55,1,0);
      fill_bow_down(x-2,y+53,4,65535);
            fill_bow_down(x+14,y+53,4,65535);
      fill_bow_down(x+36,y+53,4,65535);
      fill_bow_down(x+53,y+53,4,65535);
      linelevel(x-5,y-48,x+55,y-48,1,0);
      linever(x,y-48,x,y-32,1,0);
      linever(x+50,y-48,x+50,y-32,1,0);
}

void window_close(int x,int y)          //关着的窗户
{
```

```
    x = x*40;
    y = y*40+4;
    bar(x,y+10,x+80,y+35,27469);
    bar(x+3,y+13,x+77,y+32,17430);
    bar(x+35,y+12,x+45,y+32,29714);
    linever(x+40,y+10,x+40,y+35,1,0);
}

void cupboard(int x,int y)          //衣柜
{
    x = x*40;
    y = y*40+4;
    theta_bar(x,y+5,20,10,-30,56612);
    theta_bar(x+12,y,20,10,30,56612);
    bar(x,y+5,x+40,y+40,0);
    bar(x+1,y+6,x+39,y+39,56612);
    linelevel(x,y+5,x+40,y+40,1,0);
    linever(x+20,y+5,x+20,y+40,1,0);
    circle(x+16,y+20,2,0);
    circle(x+24,y+20,2,0);
    lean_line(x,y+5,10,-30,0);
    lean_line(x+31,y,10,30,0);
    linelevel(x+8,y,x+30,y,1,0);
}

void aircon(int x,int y,int open)//80*40,open==1 为开启，空调
{

    int color[2] = {2016,55463};            //第一个元素是绿，第二个元素是红
    x = x*40;
    y = y*40-2;
    bar(x,y+10,x+80,y+40,65535);
    linelevel(x,y+30,x+80,y+30,1,0);
    linever(x+4,y+30,x+4,y+40,1,0);
    linever(x+76,y+30,x+76,y+40,1,0);
    linelevel(x+10,y+34,x+70,y+34,1,0);
    linelevel(x+10,y+37,x+70,y+37,1,0);

    if(open)            //空调开
    {
        FillCircle(x+5,y+15,3,color[0]);            //绿灯亮
    }
    else                //空调关
    {
```

```
            FillCircle(x+5,y+15,3,color[1]);              //红灯亮
        }

}

void WashMach(int x,int y)//40*40，洗衣机
{
        int color[3] = {17430/*钢蓝*/,29714/*石板灰*/,50712/*灰色，用于制作阴影效果*/};
        x = x*40+5;              //多 8 个偏移量
        y = y*40+4;
        theta_bar(x,y+5,20,10,-30,color[2]);
        theta_bar(x+12,y,20,10,30,color[2]);
        lean_line(x,y+5,10,-30,0);
        lean_line(x+31,y,10,30,0);
        linelevel(x+8,y,x+30,y,1,0);
        bar(x,y+5,x+40,y+40,0);
        bar(x+1,y+6,x+39,y+39,65535);
        linelevel(x,y+5,x+40,y+5,1,0);
        FillCircle(x+20,y+23,10,color[1]);
        FillCircle(x+20,y+23,7,color[0]);
        circle(x+20,y+23,8,0);

}

void bookshelf(int x,int y)//40*80，书架
{
        int color[5] = {44373,64033,64594,34816,45312};//深灰色，橙红色，深橙色，深红色，
耐火砖
        x = x*40;
        y = y*40+4;
        bar(x,y,x+40,y+2,65535);
        bar(x,y+40,x+40,y+42,65535);
        bar(x,y+2,x+2,y+40,color[0]);
        bar(x+38,y+2,x+40,y+40,color[0]);
        bar(x,y+42,x+2,y+80,color[0]);
        bar(x+38,y+42,x+40,y+80,color[0]);

        //书
        bar(x+2,y+30,x+7,y+40,color[1]);
        bar(x+7,y+30,x+12,y+40,color[2]);
        bar(x+12,y+20,x+20,y+40,color[3]);
        bar(x+20,y+20,x+28,y+40,65535);
        bar(x+28,y+20,x+36,y+40,color[4]);
}
```

```
void desk(int x,int y)//80*40，小方桌
{
	int color[2] = {41605,65469};//黄锗土色，海贝色
	x = x*40+5;
	y = y*40+4;
	bar(x,y,x+80,y+25,color[0]);
	bar(x,y+25,x+80,y+32,color[1]);
	bar(x,y+32,x+10,y+40,color[1]);
	bar(x+70,y+32,x+80,y+40,color[1]);
}

void seat(int x,int y)//40*40，小板凳
{
	int color = 65370;//古董白
	x = x*40;
	y = y*40+4;
	bar(x+11,y+10,x+30,y+30,0);
	bar(x+12,y+11,x+29,y+29,color);
	ever_Fillellipse(x+15,y+10,x+25,y+10,5,0);
	ever_Fillellipse(x+15,y+10,x+25,y+10,4,color);
}

void trash1(int x,int y)//40*40，纸张
{
	x = x*40;
	y = y*40+4;
	FillCircle(x+20,y+30,5,65535);
	triangle1(x+12,y+23,7,65535);
	triangle1(x+25,y+12,10,65535);
	lean_line(x+25,y+30,7,-45,0);
	bow(x+23,y+21,12,0);

}

void trash2(int x,int y)//40*40,菜叶
{
	x = x*40;
	y = y*40+4;
	linever(x+20,y+15,x+20,y+25,2,transcolor(127,255,0));
	linever(x+14,y+15,x+14,y+25,2,transcolor(127,255,0));
	linever(x+18,y+15,x+18,y+25,2,transcolor(127,255,0));
	lean_line(x+20,y+25,10,45,transcolor(127,255,0));
	lean_line(x+14,y+25,10,45,transcolor(127,255,0));
```

```
        lean_line(x+18,y+25,10,45,transcolor(127,255,0));
}

void trash3(int x,int y)//40*40,果核
{
        x = x*40;
        y = y*40+4;
        bar(x+18,y+10,x+22,y+25,transcolor(255,215,0));
        ever_Fillellipse(x+12,y+10,x+28,y+10,3,63488);
        linever(x+20,y,x+20,y+10,1,0);
}

void pc(int x,int y)       //电脑
{
        x = x*40+5;             //这个坐标是小方块的坐标，给 5 个偏移量
        y = y*40;
        bar(x,y,x+40,y+20,0);
        bar(x+2,y+2,x+38,y+18,65535);
        bar(x+16,y+20,x+24,y+25,0);
        bar(x+12,y+25,x+28,y+28,0);
}

void TV(int x,int y)       //15*125
{
        x = x*40;             //这个坐标是小方块的坐标
        y = y*40+4+40;
        bar(x,y,x+6,y+150,0);   //黑色电视
}

void rect_table(int x,int y)//80*80，木条纹方桌
{
        int i;
        int color[3]={63222,54705,35362};
        x = x*40+5;         //多 5 个偏移量
        y = y*40+4;
        bar_round(x+40,y+35,80,70,3,1,0);
        bar_round(x+40,y+35,78,68,3,1,color[1]);
        bar(x+5,y+1,x+7,y+69,color[0]);
        bar(x+5+1*15,y+1,x+9+1*15,y+69,color[0]);
        bar(x+5+2*15,y+1,x+13+2*15,y+69,color[0]);
        bar(x+5+48,y+1,x+9+3*15,y+69,color[0]);
        bar(x+5+64,y+1,x+7+4*15,y+69,color[0]);
        theta_bar(x,y+71,50,6,45,color[2]);
        theta_bar(x+40,y+74,38,5,-45,color[2]);
```

```
    bar(x+10,y+75,x+14,y+80,color[2]);
    bar(x+68,y+75,x+74,y+80,color[2]);

}

void sofa_main(int x, int y)      //沙发主要部分
{
    x = x * 40;
    y = y * 40 +4;
    bar(x+20, y, x + 80, y + 160, 56784);      //硬木色
    bar(x+20, y+40, x + 50, y + 120, 65370);       //古董白
}

void sofa_up(int x, int y)         //沙发，屏幕上面的那组
{
    x = x * 40;
    y = y * 40 +8;
    bar(x, y, x + 80, y + 60, 56784);      //硬木色
    bar(x + 20, y + 20, x + 60, y + 60, 65370);       //古董白
}

void sofa_down(int x, int y)      //沙发，屏幕下面的那组
{
    x = x * 40;
    y = y * 40 +12;
    bar(x, y, x + 80, y + 60, 56784);      //硬木色
    bar(x + 20, y, x + 60, y + 40, 65370);       //古董白
}

void toilet(int x, int y)          //马桶
{
    int i;
    x = x * 40 + 5;       //5 个偏移量
    y = y * 40;
    linelevel(x, y + 10, x + 10, y + 10, 1, 0);

    bar(x, y + 10, x + 10, y + 40, 65535);   //水箱
    Horizline(x, y + 10, 10, 0);
    Horizline(x, y + 20, 10, 0);             //水箱上的分割线

    bar(x + 10, y + 30, x + 20, y + 40, 65535); //底座
    eqver_tri(x + 20, y + 30, 10, 65535);
    lean_line(x + 20, y + 40, 10 * sqrt(2), -45, 0);       //三角形和外部的分界线
    Horizline(x, y + 40, 20, 0);
```

```
    FillCircle(x + 15, y + 10, 5, 65535);                //马桶盖
    bow_left_up(x + 15, y + 10, 5, 0);                    //马桶盖边框
    bow_right_up(x + 15, y + 10, 5, 0);
    bar(x + 10, y + 10, x +20, y + 20, 65535);
    linever(x + 20, y + 10, x + 20, y + 20, 1, 0);        //也是马桶盖与外界的边界

    bar(x + 10, y + 20, x + 25, y + 30, 65535);        //马桶...
    lean_line(x + 10, y + 30, 10 * sqrt(2), -45, 0); //盖与主体之间的斜线
    Horizline(x + 20, y + 20, 8, 0);                    //四条横线
    linelevel(x + 17.5, y + 22.5, x + 28, y + 22.5, 1, 0);
    linelevel(x + 12.5, y + 27.5, x + 28, y + 27.5, 1, 0);
    Horizline(x + 10, y + 30, 15, 0);
    FillCircle(x + 25, y + 25, 5, 65535);
    bow_right_up(x + 25, y + 25, 2.5, 0);
    bow_right_down(x + 25, y + 25, 2.5, 0);
    bow_right_up(x + 25, y + 25, 5, 0);
    bow_right_down(x + 25, y + 25, 5, 0);

    linever(x + 10, y + 10, x + 10, y + 40, 1, 0);   //水箱与马桶盖的分界线，如果在前面画会被覆盖
}

void water_dispenser(int x, int y)   //饮水机
{
    x = x * 40;
    y = y * 40;

    //底座
    y = y + 40; //第二个方格
    theta_bar(x,y+5,20,10,-30,29714);     //石板灰
    theta_bar(x+12,y,20,10,30,29714);
    lean_line(x,y+5,10,-30,0);
    lean_line(x+31,y,10,30,0);
    linelevel(x+8,y,x+30,y,1,0);
    bar(x,y+5,x+40,y+40,0);
    bar(x+1,y+6,x+39,y+39,65535);
    linelevel(x,y+5,x+40,y+5,1,0);

    //接水框
    bar(x + 10, y + 10, x + 30, y + 30, 34431); //亮天蓝
    bar(x + 17.5, y + 10, x + 22.5, y + 15, 7327);          //tap，道奇蓝

    //水桶
```

```
    y = y -40;   //第一个方格，变回来
    ever_Fillellipse(x + 15, y + 10, x + 25, y + 10, 5, 26620); //水蓝
    ever_Fillellipse(x + 15, y + 33, x + 25, y + 33, 5, 26620); //水蓝
    bar(x + 12, y + 10, x + 28, y + 33, 26620);         //水蓝
    bar(x + 18.5, y + 30, x + 21.5, y + 40, 26620);

}

void zaotai(int x, int y)            //灶台
{
    x = x * 40;
    y = y * 40;
    theta_bar(x, y + 10, 60, 10, -45, 46651);     //亮钢蓝，台面
    theta_bar(x + 10, y+2, 60, 10, 45, 46651);
    bar(x, y + 10, x + 80, y + 40, 46651);
    Horizline(x, y + 10, 80, 0);
    Horizline(x, y + 40, 80, 0);
    Horizline(x + 10, y, 60, 0);
    lean_line(x, y + 10, 10 * sqrt(2), -45, 0);
    lean_line(x + 70, y, 10 * sqrt(2), 45, 0);
    linever(x, y +10, x, y + 40, 1, 0);
    linever(x + 80, y + 10, x + 80, y + 40, 1, 0);

    //灶台的加热圈
    FillCircle(x + 30, y + 6, 2.5, 29779);       //亮岩灰
    FillCircle(x + 50, y + 6, 2.5, 29779);

    //边沿线
    bow_left_up(x + 30, y + 6, 3.5, 0);
    bow_left_down(x + 30, y + 6, 3.5, 0);
    bow_right_up(x + 30, y + 6, 3.5, 0);
    bow_right_down(x + 30, y + 6, 3.5, 0);

    bow_left_up(x + 50, y + 6, 3.5, 0);
    bow_left_down(x + 50, y + 6, 3.5, 0);
    bow_right_up(x + 50, y + 6, 3.5, 0);
    bow_right_down(x + 50, y + 6, 3.5, 0);
}

///左上角，画一个水杯，占 1/16 个方格，入口改为像素点，而非大坐标，因为机器人持物的图像要用
void water_bottle(int x, int y)
{
    bar(x+1, y, x+9, y+13, 0);
```

```
    bar(x+2, y+1, x+8, y+12, 59391);
}

void clothes(int x, int y)   //衣服
{
    bar(x+5, y, x+25, y+3, ARMSM);
    bar(x+5, y+4, x+25, y+6, PANTSM);
}

void plate(int x,int y)      //盘子
{
    theta_bar(x,y+5,20,10,-30,65535);
    theta_bar(x+12,y,20,10,30,65535);
    lean_line(x,y+5,10,-30,0);
    lean_line(x+31,y,10,30,0);
    linelevel(x+8,y,x+30,y,1,0);
    linelevel(x,y+5,x+30,y+5,1,0);
}

void medical_kit(int x, int y)   //医疗包
{
    bar(x+10, y+15, x+30, y+35, 65535);
    FillCircle(x+20, y+25, 7, 63488);
    bar(x+18, y+20, x+22, y+30, 65535);      //十字的横
    bar(x+15, y+23, x+25, y+27, 65535);      //十字的竖
}

void TV_on(void)                 //电视打开状态
{
    int i,j;

    for (i=0;i<34;i++)
    {
        for (j=0;j<150+2*i;j++)
        {
            Putpixel64k(287+i,445-i+j,57083);
        }
    }

}

void TV_off(void)                //电视关闭（和上面 TV 的不同之处是，要做额外绘画去恢复
客厅模样）
{
```

```
    int j;

    for(j = 10; j < 16; j++)          //刚刚被电视投影灰色覆盖部分的地板恢复
    {
        wood_ver(7, j);
    }

    for(j = 11; j < 17; j++)        //卧室与客厅交界的墙壁恢复
    {
        w_right(6, j);
    }

    TV(7, 10.5);
}

void music_on(int x, int y) //音符
{
    bar(x*40+15,y*40,x*40+20,y*40+32,0);
    FillCircle(x*40+9,y*40+32,8,0);
    bow_right_up(x*40+20,y*40+15,15,0);
    bow_right_up(x*40+20,y*40+14,14,0);
    bow_right_up(x*40+20,y*40+13,13,0);
    bow_right_up(x*40+20,y*40+12,12,0);
    bow_right_up(x*40+20,y*40+11,11,0);
}

void music_off( int x, int y)     //去除音符
{
    wood_ver(x, y);
    wood_ver(x, y-1);
}
```

----------hzxs.c----------

```
    #include "title.h"

    void gethz(char incode[],char    *bytes)
    {
        unsigned char qh, wh;
        unsigned long offset;
        FILE * fhzk_p;                                                    /*定义
文件指针*/
        fhzk_p = fopen("HZK\\hzk16", "rb");
```

```
    if (fhzk_p == NULL)
    {
        printf("the hzk can't open\n");

        delay(1000);
        exit(1);
    }


    qh=incode[0]-0xa0;
    wh=incode[1]-0xa0;
    offset=(94*(qh-1)+(wh-1))*32L;
    fseek(fhzk_p,offset,SEEK_SET);
    fread(bytes,1,32,fhzk_p);
    fclose(fhzk_p);
}



void dishz(int x0,int y0,int mx,int my,char *code,int color)//显示字符
{
    int x, y, i, j, k, xt, yt, bit;
    int count = 0;
    unsigned char mask=0x80;
    char *mat=(char *)malloc(32*sizeof(char));
    if (mat == NULL)
    {
        overflow_box(500,500);
        getch();
        printf("there is no place\n");
        exit(1);

    }



    gethz(code,mat);
    y=y0;
    x=x0;
    i=0;
    while(i<32)
    {
        for(yt=0;yt<my;yt++)
```

```
            {
                for(k=0;k<2;k++,count++)
                {
                    for(j=0;j<8;j++)
                    {
                        bit=mat[count]&mask;
                        for(xt=0;xt<mx;xt++)
                        {
                            if(bit)
                            {
                                Putpixel64k(x,y,color);
                            }
                            x++;
                        }
                        mask=mask>>1;
                    }
                    mask=0x80;
                }
                x=x0;
                count=count-2;
                y++;
            }
            i+=2;
            count+=2;
        }
        free(mat);
}



void fdhz(int x,int y,int mx,int my,char *s,int color)
{
   while(*s!=NULL)
   {
    while( ( (x+20*mx) <1024)&&(*s!=NULL) )
    {
    dishz(x,y,mx,my,s,color);
    x+=20*mx;
    s+=2;
    }
    x=20;
    y+=20*my;
   }
```

```
    }
```

----------input.c----------

```
#include "title.h"
/*结构体包涵：按键的键值与代表的字符*/
typedef struct
{
   int value;
   char ch;
}setKeyValue;


/*该数组收集了 26 个英文字母的大小写以及上、右数字键所对应的键值*/
setKeyValue KeyValue[74]={
    {0x1e61,'a'},{0x3062,'b'},{0x2e63,'c'},{0x2064,'d'},{0x1265,'e'},
    {0x2166,'f'},{0x2267,'g'},{0x2368,'h'},{0x1769,'i'},{0x246a,'j'},
    {0x256b,'k'},{0x266c,'l'},{0x326d,'m'},{0x316e,'n'},{0x186f,'o'},
    {0x1970,'p'},{0x1071,'q'},{0x1372,'r'},{0x1f73,'s'},{0x1474,'t'},
    {0x1675,'u'},{0x2f76,'v'},{0x1177,'w'},{0x2d78,'x'},{0x1579,'y'},
    {0x2c7a,'z'},
    {0x1e41,'A'},{0x3042,'B'},{0x2e43,'C'},{0x2044,'D'},{0x1245,'E'},
    {0x2146,'F'},{0x2247,'G'},{0x2348,'H'},{0x1749,'I'},{0x244a,'J'},
    {0x254b,'K'},{0x264c,'L'},{0x324d,'M'},{0x314e,'N'},{0x184f,'O'},
    {0x1950,'P'},{0x1051,'Q'},{0x1352,'R'},{0x1f53,'S'},{0x1454,'T'},
    {0x1655,'U'},{0x2f56,'V'},{0x1157,'W'},{0x2d58,'X'},{0x1559,'Y'},
    {0x2c5a,'Z'},
    {0x4f31,'1'},{0x5032,'2'},{0x5133,'3'},{0x4b34,'4'},{0x4c35,'5'},
    {0x4d36,'6'},{0x4737,'7'},{0x4838,'8'},{0x4939,'9'},{0x5230,'0'},
    {0x231,'1'},{0x332,'2'},{0x433,'3'},{0x534,'4'},{0x635,'5'},
    {0x736,'6'},{0x837,'7'},{0x938,'8'},{0xa39,'9'},{0xb30,'0'},{0x532e,'.'},
    {0x342e,'.'}};


/*
    补充几个功能键的键值：
    backspace:0xe08
    esc:0x11b
    enter:0x1c0d
    上方向键：0x4800
    下方向键：0x5000
    左方向键：0x4b00
    右方向键：0x4d00
```

```
 */

/************************************************
Function: searchKeyValue
Description: 根据键值返回表中其对应字符
Calls:
Return: 若有则返回对应字符；若表中无此键值，则返回'\0'
************************************************/
char searchKeyValue(int value)
{
   int i;
   for(i=0;i<74;i++)
   {
      if(value==KeyValue[i].value)break;
   }
   if(i<74)return KeyValue[i].ch;
   else return '\0';
}

/************************************************
function:          put_English

description :      在指定的地方输出英文

Input :            x1,y1 输出位置坐标，ascii 为该英文的 ASCII 码

out :              在指定位置输出英文字母

************************************************************/
void   putEnglish(int x1,int y1,int ascii,int mx,int my,int color)
{
      char * English_save;                                    /*定义指向字模存取的指针*/

      int x0;                                                 /*定义记录位置的变量个*/
      int y0;
      int yt;
      int xt;

      int i = 0;                                                /*循环变量*/
      int j = 0;
      int k = 0;
      unsigned char mask[] = { 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01 };
      unsigned long offest;                                   /*计算英文字母在汉字库中的偏
```

```
移量*/
    FILE * fh;                                                  /*定义一个文件指针*/
    const char *fname = "HZK\\ASC16.DZK";
    fh = fopen(fname, "rb");
   if (fh == NULL)
    {
        printf("the hzk can't open\n");
        delay(1000);
        exit(1);
    }



   English_save = (char *)malloc( 32* sizeof(char));
   if(English_save==NULL)
    {
        overflow_box(500,500);
        getch();
        exit(1);
    }



    offest = ascii * 32l;
    fseek(fh, offest, 0);
    fread(English_save, 32, 1, fh);

    x0 = x1;                                                    /*记录输入位置*/
    y0 = y1;
    fclose(fh);
    while (i <16)                                               /*循环判断输出英文字母*/
    {
        for (yt = 0; yt < my; yt++)
        {
            for (j = 0; j < 2; j++)
            {
                for (k = 0; k < 8; k++)
                {
                    for (xt = 0; xt < mx; xt++)
                    {
                        if ((English_save[2 * i + j] & mask[k % 8]) != 0)
                        {
                            Putpixel64k(x0 , y0 , color);
```

```
                        }
                        x0++;
                    }
                }

            }
            x0 = x1;
            y0++;
        }
        i++;
    }


    free(English_save);
}

/*仿照 graphic,调用 putEnglish 函数实现输出整行字符*/
void outtextxy(int x,int y,char *c,int mx,int my,int mar,int color)//x,y（输出位置）,c（要输出的字符串）,mx,my（字母尺寸，横向/纵向）,mar（字符之间间距）
{
    int a;
    char *p=c;
    while(*p!='\0')
    {
        a = (int)(*p);
        putEnglish(x,y,a,mx,my,color);
        p++;
        x+=mar;
    }
}
```

----------iph_page.c----------

```
#include "title.h"


//功能：手机外框等固定元素，任何界面都需要显示
void iph_frame()
{
    bar_round_2(ORIGINX,ORIGINY,FINALX,FINALY,30,1,65534);
    linelevel(ORIGINX+30,2,FINALX-30,2,2,0);
    linelevel(ORIGINX+30,FINALY,FINALX-30,547,2,0);
    linever(ORIGINX,ORIGINY+30,ORIGINX,FINALY-30,2,0);
    linever(FINALX,ORIGINY+30,FINALX,FINALY-30,2,0);
```

```
    bow_left_down(ORIGINX+30,FINALY-30,31,0);
    bow_left_down(ORIGINX+30,FINALY-30,32,0);
    bow_left_up(ORIGINX+30,ORIGINY+30,31,0);
    bow_left_up(ORIGINX+30,ORIGINY+30,32,0);
    bow_right_down(FINALX-30,FINALY-30,31,0);
    bow_right_down(FINALX-30,FINALY-30,32,0);
    bow_right_up(FINALX-30,ORIGINY+30,31,0);
    bow_right_up(FINALX-30,ORIGINY+30,32,0);
    bar (MIDDLEX-6,4,MIDDLEX+6,9,0);
    FillCircle(MIDDLEX,9,6,0);
    FillCircle(MIDDLEX,9,2,4523);                                                    //
    RGB__0 52 91
    //    信号&电源
    bar(FINALX-52,24,FINALX-49,32,0);
    bar(FINALX-48,21,FINALX-45,32,0);
    bar(FINALX-44,18,FINALX-41,32,0);
    bar(FINALX-40,15,FINALX-37,32,0);
                        //    RGB: 10 10 10
    bar_round(FINALX-20,21,20,12,2,1,0);

}

//功能：手机界面附加元素
void iph_frame_plus()
{
    linelevel(ORIGINX,ORIGINY+56,FINALX,ORIGINY+56,1,0);
//页面名称
    bar_round_2(ORIGINX,ORIGINY,FINALX,ORIGINY+56,30,1,54938);
    bar(ORIGINX,ORIGINY+26,FINALX,ORIGINY+56,54938);
    bar (MIDDLEX-6,ORIGINY,MIDDLEX+6,ORIGINY+5,0);
    FillCircle(MIDDLEX,ORIGINY+5,6,0);
    FillCircle(MIDDLEX,ORIGINY+5,2,4523);
    bar(FINALX-52,24,FINALX-49,32,0);
    bar(FINALX-48,21,FINALX-45,32,0);
    bar(FINALX-44,18,FINALX-41,32,0);
    bar(FINALX-40,15,FINALX-37,32,0);
                        //    RGB: 10 10 10
    bar_round(FINALX-20,21,20,12,2,1,0);
}

//功能：手机主界面
void iph_page_1()
{
    iph_frame();
```

```
    iph_frame_plus();
    fdhz(MIDDLEX-30,ORIGINY+18,1,1,"主界面",0);

    linelevel(ORIGINX,ORIGINY+209,FINALX,ORIGINY+209,1,0);
//手动指令按钮
    fdhz(MIDDLEX-80,ORIGINY+116,2,2,"手动指令",0);

    linelevel(ORIGINX,ORIGINY+363,FINALX,ORIGINY+363,1,0);
    fdhz(MIDDLEX-40,ORIGINY+269,2,2,"聊天",0);                    //聊
天按钮

    linelevel(ORIGINX,ORIGINY+515,FINALX,ORIGINY+515,1,0);
//默认设置修改按钮
    fdhz(MIDDLEX-120,ORIGINY+422,2,2,"默认设置修改",0);
}

//功能：手动指令界面
void                                                    iph_page_2()
//手动指令界面
{
    iph_frame();

    iph_frame_plus();
    fdhz(MIDDLEX-40,ORIGINY+18,1,1,"手动指令",0);

    linelevel(ORIGINX,ORIGINY+171,FINALX,ORIGINY+171,1,0);
//打扫卫生按钮
    fdhz(MIDDLEX-80,ORIGINY+116,2,2,"打扫卫生",0);

    linelevel(ORIGINX,ORIGINY+286,FINALX,ORIGINY+286,1,0);
    fdhz(MIDDLEX-60,ORIGINY+231,2,2,"倒水",0);

}

//功能：聊天界面
void            iph_page_3(unsigned          int          *box_save)
//聊天界面
{
    iph_frame();

    iph_frame_plus();
    fdhz(MIDDLEX-20,22,1,1,"聊天",0);
```

```
    /***************
    输入框
    **************/
    bar_round_2(ORIGINX+13,ORIGINY+50,ORIGINX+243,ORIGINY+468,5,1,54938);
    bar_round_2(ORIGINX+13,ORIGINY+363,ORIGINX+243,ORIGINY+468,5,1,65535);
    bar(ORIGINX+13,ORIGINY+384,ORIGINX+243,ORIGINY+403,48631);          //191     191
191

    get_image(ORIGINX+25,ORIGINY+365/*365*/,ORIGINX+225,ORIGINY+400/*400*/,
box_save);

    /********
    界面背景
    ********/
    FillCircle(ORIGINX+200,ORIGINY+157,40,54938);    //138 198 210
    fill_bow_right_up(ORIGINX+3,ORIGINY+305,120,32313);//121 198 204
    fill_bow_right_down(ORIGINX+3,ORIGINY+305,120,32313);  //121 198 204



}

//功能：默认设置修改界面
void iph_page_4(char *at,char *bt)                                                  //默
认设置修改界面
{
    iph_frame();

    iph_frame_plus();
    fdhz(MIDDLEX-60,ORIGINY+18,1,1,"默认设置修改",0);

    fdhz(MIDDLEX-100,ORIGINY+138,1,1,"空调温度",0);
    bar(FINALX-120,ORIGINY+136,FINALX-95,ORIGINY+161,33808);//900，140，925，165
减号
    bar(FINALX-117,ORIGINY+146,FINALX-98,ORIGINY+151,65535);

    bar(FINALX-85,ORIGINY+128,FINALX-45,ORIGINY+168,65535);  //温度显示区域
    outtextxy(FINALX-75,ORIGINY+138,at,1,1,10,0);

    bar(FINALX-35,ORIGINY+136,FINALX-10,ORIGINY+161,33808); //加号
    bar(FINALX-32,ORIGINY+146,FINALX-13,ORIGINY+151,65535);
    bar(FINALX-25,ORIGINY+139,FINALX-20,ORIGINY+158,65535);

    fdhz(MIDDLEX-100,ORIGINY+253,1,1,"洗澡水温",0);
    bar(FINALX-120,ORIGINY+251,FINALX-95,ORIGINY+276,33808);//900，140，925，165
```

```
减号
    bar(FINALX-117,ORIGINY+261,FINALX-98,ORIGINY+266,65535);

    bar(FINALX-85,ORIGINY+243,FINALX-45,ORIGINY+283,65535);   //温度显示区域
    outtextxy(FINALX-75,ORIGINY+253,bt,1,1,10,0);

    bar(FINALX-35,ORIGINY+251,FINALX-10,ORIGINY+276,33808); //加号
    bar(FINALX-32,ORIGINY+261,FINALX-13,ORIGINY+266,65535);
    bar(FINALX-25,ORIGINY+254,FINALX-20,ORIGINY+273,65535);
}

//功能：时间界面
void time_page()
{
    bar(764,572,1020,764,50712);//银色大框
    linelevel(764,690,1020,690,3,0);
    bar(794,597,990,665,65535);//时间显示框
    fdhz(894-40,727-8,1,1,"下一事件",0);
}
```

----------log_in.c----------

```
#include "title.h"
//功能：登录注册功能主逻辑
//输入：无
//输出：int 型
//        返回 1 为登陆成功，跳出循环，进入下一模块
//        返回 0 为登陆失败，继续循环
int enter(void)
{
    USERS user;                                          //用来存放当前用户
的信息
    USERS *head = NULL;                                  //链表的头节点
    int temp;                                          //用于存放键盘输入
    //int error=0;                                       //
    int button,x,y;
    int judge=0;
    user.account[0] = '\0';
    user.code[0] = '\0';

    if ((head = (USERS *)malloc(sizeof(USERS))) == NULL)
    {
        overflow_box(500,500);
        getch();
        exit(1);
```

```
}
create_list(head);
log_in_page();                                          //静止登录界面
mouseInit(&x, &y,&button);
while(1)
{
    //newxy(&x, &y, &button);
                    /*初始化，接受缓冲区数据*/
    if (kbhit() == 0)
    {
        newxy(&x, &y, &button);
    }
    if (kbhit() != 0)
    {
        temp = bioskey(0);
    }

    /*按 esc 则退出*/
    if(temp == 0x11b)
    {
        exit(0);
    }

    /*登录*/
    if(x>=650&&x<=700&&y<=495&&y>=445&&button)
    {
        judge = log_in_check(head,&user.account,&user.code);   //登录成功返回 1
    }


    /*注册*/
    if(x>=440&&x<=560&&y>=555&&y<=590&&button)
    {

        judge = UserRegist(head,&user.account,&user.code,&x,&y,&button);
    }

    if(x>=300&&x<=700&&y>=295&&y<=345&&button)  //选择 id 输入
    {
        judge = 2;
    }

    if(x>=300&&x<=700&&y>=445&&y<=495&&button)  //选择密码输入
    {
```

```
                judge = 3;
            }

            if(judge==2)//输入 id
            {
                judge = input_account(head,&user.account,&user.code,&x,&y);

            }
            if(judge==3)//输入 password
            {
                judge = input_code(head,&user.account,&user.code,&x,&y);
            }
            if(judge==1)
            {
                free_list(&head);
                return 1;
            }
            //返回登录界面
            if(judge==5)
            {
                free_list(&head);
                head=NULL;
                return 0;       //只要不是 1 就可以了
            }

        }
}


//功能：登录的静止界面
//输入：无
//输出：无
void log_in_page(void)
{
    CASE robot_position;
    robot_position.xpixel=500;
    robot_position.ypixel=120;

    linever_color(0,0,1024,768,211,211,211,128,128,128);
    bar_round(500,320,400,50,5,1,65535);//300,295,700,345
    bar_round(500,470,400,50,5,1,65535);
    fdhz(250,310,1,1,"帐",0);
    fdhz(270,310,1,1,"号",0);
    fdhz(250,460,1,1,"密",0);
```

```
        fdhz(270,460,1,1,"码",0);
        FillCircle(675,470,22,65535);
        circle(675,470,22,33808);
        circle(675,470,23,33808);
        linelevel(660,470,690,470,2,33808);
        lean_line_thick(690,470,20,225,2,33808);
        lean_line_thick(690,470,20,135,2,33808);
        bar (440,555,560,590,50712);
        fdhz(484,570,1,1,"注册",0);


        logo_robot(robot_position);

}


//功能：账号输入函数
//输入：用户信息链表的头节点，指向账号字符串的字符指针，指向密码字符串的字符指针，
鼠标坐标指针
//输出：int 型
//          返回 1：登录或注册成功
//          返回 3：进入密码输入函数
int input_account(USERS *head,char *account,char *code,int *x,int *y)
{
        int key;
        int i=0;//用于计算已输入的字符数目的变量
        char *p=account;//输入字符的中间指针变量
        char ch;//用于临时储存键值所对应字符的变量
        int buttons,judge;
        char temp[2] = {'\0','\0'};

        //查看已经输入了多少个字符
        while(*p != '\0')
        {
                i++;
                p++;
        }
        while(1)
        {
                //newxy(x,y,&buttons);
                key=0;
                if (kbhit() == 0)
                {
                        newxy(x,y,&buttons);
```

```
}
if (kbhit() != 0)
{
	key = bioskey(0);	//输入的是键值
}


 /*将按键对应的字符存入 account 中*/
ch = searchKeyValue(key);
if (ch != '\0'&&i<10)
{
	temp[0] = ch;
	outtextxy(300+i*18,305,temp,2,2,10,0);


/*将字符存入数组中*/
	*p = ch;
	p++;
	*p = '\0';
	i++;
}



if(key ==0xe08)		//按了回删键
{
	if(p != account)
	{
		bar(290 + i * 18, 295, 340 + i * 18, 345,65535);
		p--;
		i--;
	}
	*p = '\0';
}
/*按 esc 则退出*/
if(key == 0x11b)
{
	exit(0);
}

//删掉了防止鼠标遮挡的功能（2020/7/26）
```

```
            //密码输入
            if(*x>=300&&*x<=700&&*y>=445&&*y<=495&&buttons)
            {
                return 3;
            }

            //注册键
            if(*x>=440&&*x<=560&&*y>=555&&*y<=590&&buttons)
            {

                judge = UserRegist(head,account,code,x,y,&buttons);
                return judge;
            }




            //登录键
            if(*x>=650&&*x<=700&&*y<=495&&*y>=445&&buttons)
            {
                (400,535,400,655,2,31694);

                judge = log_in_check(head,account,code);
                return judge;
            }
        }
}

//功能：账号输入函数
//输入：用户信息链表的头节点，指向账号字符串的字符指针，指向密码字符串的字符指针，
鼠标坐标指针
//输出：int 型
//          返回 1：登录或注册成功
//          返回 2：进入 id 输入函数
int input_code(USERS *head,char *account,char *code,int *x,int *y)
{
    int key;
    int i=0;//用于计算已输入的字符数目的变量
    char *u = code;//输入字符的中间指针变量
    char ch;//用于临时储存键值所对应字符的变量
    int buttons,judge;

    while(*u != '\0')
    {
```

```
        i++;
        u++;
    }
    while(1)
    {
        newxy(x,y,&buttons);
        key=0;
        if (kbhit() != 0)
        {
            key = bioskey(0);
        }
        if(key ==0xe08)
        {
            if(u != code)
            {
                bar(275 + i * 25, 450, 305 + i * 25, 490,65535);
                u--;
                i--;
            }
            *u = '\0';
        }

        /*按 esc 则退出*/
        if(key == 0x11b)
        {
            exit(0);
        }

        //登录键
        if(*x>=650&&*x<=700&&*y<=495&&*y>=445&&buttons)
        {

            judge = log_in_check(head,account,code);
            return judge;
        }

        //删去防止输入时遮挡功能



        //输入账号
        if(*x>=300&&*x<=700&&*y>=295&&*y<=345&&buttons)
        {
```

```
            return 2;
        }

        //注册框
        if(*x>=440&&*x<=560&&*y>=555&&*y<=590&&buttons)
        {

            judge = UserRegist(head,account,code,x,y,&buttons);
            return judge;
        }


    /*将按键对应的字符存入 code 数组中*/
        ch = searchKeyValue(key);
        if (ch != '\0'&&i<10)
        {


            //bar(250 + i * 11, 218, 261 + i * 11, 242,65535);
            FillCircle(315+i*25,470,8,0);

        /*将字符存入数组中*/
            *u = ch;
            u++;
            *u = '\0';
            i++;
        }
    }
}



//功能：验证账号密码是否正确
//输入：用户信息链表的头节点，指向账字符串的字符指针，指向密码字符串的字符指针
//输出：int 型
//        返回 1：验证成功
//        返回 5：验证失败
int log_in_check(USERS *head,char *account,char *code)
{
    char *rightcode= NULL;
    rightcode = accounts_2_code(head, account);
    if (rightcode != NULL &&strcmp(rightcode, code) == 0)
        return 1; //登录成功
    else
```

```
    {
        fdhz(410, 500,1, 1, "用户名或密码输入错误", 0);
        getch();
        return 5;
    }
}
```

----------main_.c----------

```
#include "title.h"
//功能：主函数
void main()
{
    unsigned int a;
    int judge=0;//返回 1 为验证失败
    int x,y, button;
    mouseInit(&x, &y,&button);  //鼠标初始化
    newxy(&x, &y, &button);      //画鼠标



    //SetSVGA64k();



    //CreateHouseFile();
    SetSVGA64k();
    a = GetSVGA();              //svga 初始化

    srand((int)time(0));

/*
    outwelcome();           //欢迎界面

    getch();



    while(judge!=1)
    {
        judge=enter();      //进入登录注册界面，登录或者注册成功就可以退出这个循环
    }
    judge=0;
    while(judge!=1)
    {
        judge=finger_check();//进入指纹验证阶段
```

```
    }

*/
    mainprocess(&x,&y,&button);      //主流程


}
```

----------main2.c----------

```
#include "title.h"
//功能：实现功能的主进程
//输入：鼠标的坐标和按键状态
//输出：无
void mainprocess(int *x,int *y, int *pbutton)
{
    int button=*pbutton;      //鼠标按键
    int click=0;              //手动指令模块选择判断变量
    int i;
    int judge=0;              //时间线模块选择判断变量

    int air_t=24;             //空调温度初始化
    int bath_t=38;            //洗澡水温初始化

    int n;
    VType G[166];             //后期计算发现，只有不到 166 个点可走

    int time=15*60;           //时间初始化
    int hour;
    int minute;
    char s_hour[3];           //小时的字符串数组
    char s_minute[3];         //分钟的字符的数组
    int times=-1;             //事件是什么
    int wht_happen=0;         //事件是否发生          *********作用：当时间线上的函数执行过一遍之后这个变量就变成 1，在循环中就不会无限调用该函数
    int wht_happen2=0;        //事件是否发生          *********作用：当手动指令上的函数执行过一遍之后这个变量就变成 1，在循环中就不会无限调用该函数

    int choice[5]={0};        //选择数组定义及其初始化

    CASE robot,man;           //人和机器人的变量的初始化

    BUTTONS esc1;             //退出按钮
    esc_init(&esc1);          //退出按钮初始化
```

```
mouseInit(x, y,&button);//鼠标初始化

n = CreateGraph(G);      //建图


//初始化机器人位置方向，四肢状态
robot.x = 13;
robot.y = 3;
robot.direction = 4;

robot.hand = 0;
robot.hand_left = 0;
robot.hand_right = 0;
robot.xpixel=robot.x*40;
robot.ypixel=robot.y*40;



//初始化人位置方向，四肢状态
man.x = 4;
man.y = 1;
man.direction = 4;
man.hand = 0;
man.hand_left = 0;
man.hand_right = 0;
man.xpixel=man.x*40;
man.ypixel=man.y*40;

//绘制地图，手机主界面
mousehide(*x,*y);
paint_house();                          //画房子
iph_page_1();                           //手机主界面
time_page();                            //时间静态界面
hour=time/60;                               //小时
minute=time%60;                             //分钟
itoa(hour,s_hour,10);                   ///小时的信息存入字符串数组
itoa(minute,s_minute,10);               //分钟的信息存入字符串数组
minute_adjust(s_minute);                //分钟全部调为 2 位数
outtextxy(810,615,s_hour,3,3,20,0);
outtextxy(874,607,":",3,3,0,0);
outtextxy(920,615,s_minute,3,3,20,0);   //时间静态界面

paint_robot(robot);                         //画机器人
```

```
    //重置鼠标
    reset_mouse(x,y);

    while(1)
    {

        newxy(x,y,&button);                          //刷新鼠标位置和状态

        //检查是否点击退出按钮
        judge = esc_check(&esc1,x,y,&button);



        if (*x>=ORIGINX && *x <=FINALX&& *y>=ORIGINY+54 && *y
<=ORIGINY+209&& button)      //手动指令按钮
        {
            click=1;
            wht_happen2=0;
        }

        if (*x>=ORIGINX && *x <=FINALX&& *y>=ORIGINY+209 && *y
<=ORIGINY+363&& button)    //聊天按钮
        {
            click=2;
            wht_happen2=0;
        }
        if (*x>=ORIGINX && *x <=FINALX&& *y>=ORIGINY+363 && *y
<=ORIGINY+515&& button)    //默认设置修改按钮
        {
            click=3;
            wht_happen2=0;
        }
        if (*x>=0 && *x <=20&& *y>=0 && *y <=20&& button)
//退出按钮
        {
            click=4;
        }
        if (*x>=764 && *x <=1020&& *y>=690 && *y <=764&& button)
        {
            time_adjust_plus(&time, &times, &wht_happen);    //时间随事件调整
            hour=time/60;
            minute=time%60;
            itoa(hour,s_hour,10);
            itoa(minute,s_minute,10);
```

```
            minute_adjust(s_minute);

            bar(794,597,990,665,65535);//时间显示框
            outtextxy(810,615,s_hour,3,3,20,0);
            outtextxy(874,607,":",3,3,0,0);
            outtextxy(920,615,s_minute,3,3,20,0);
        }



        if (wht_happen==0 && times==0)
        {
            rbtguard(man, robot, INTRADER, x, y, &button);                  //安保功能实现
            wht_happen=1;
        }
        if (wht_happen==0 && times==1)
        {
            iph_page_1();
            come_home(&robot, &man, x, y, &button, choice,G, n); //回家后功能实现
            wht_happen=1;
        }
        if (wht_happen==0 && times==2)
        {
            dinner(&robot, &man, x, y, &button, choice,G, n);          //晚饭功能实现
            wht_happen=1;
        }
        if (wht_happen==0 && times==3)
        {
            bath(&robot, &man, x, y, &button, choice,G, n);                  //洗澡功能实现
            wht_happen=1;
        }
        if (wht_happen==0 && times==4)
        {
            treatment(&man, &robot, G, n, x, y, &button, choice);                //生病救援
功能实现
            wht_happen=1;
        }
        if (wht_happen==0 && times==5    && choice[4]==0)
    //choice[4]==0 表示没有去医院
        {
            breakfast(&robot, &man, x, y, &button, G, n);               //早餐功能实现
            wht_happen=1;
        }
        if (times==6)
```

```
        {
            good_bye();                                                    //结束界面
        }


        if (wht_happen2==0)
        {
            switch(click)
            {
                case 0:
                {
                    break;
                }
                //手动指令
                case 1:
                {
                    sdzl_main(x,y,&button,&robot,&man, G, n, times);          //进入手动
指令功能函数
                    iph_page_1();                                          //退出手动指令后画
主界面
                    wht_happen2=1;                                             //已经跑了一遍，
避免一直循环
                    break;
                }
                //聊天功能
                case 2:
                {
                    chatmain(&robot);                                      //进入聊天界面
                    //click=0;
                    iph_page_1();                                          //退出聊天后画主界
面
                    wht_happen2=1;                                             //已经跑了一遍，
避免一直循环
                    break;
                }
                //默认设置修改
                case 3:
                {
                    set_change(&air_t, &bath_t);                           //进入默认设置修改
界面
                    //click=0;
                    iph_page_1();                                          //退出默认设置修改
后画主界面
                    wht_happen2=1;                                             //已经跑了一遍，
```

```
避免一直循环
                                break;
                            }
                        }
                    }
                    //若点击退出系统，则跳出循环
                    if(judge==4)
                    {
                        break;
                    }
                }

}

//功能：将一位数字的分钟数调整成 2 位数字
//输入：分钟的字符串的首地址
//输出：无
void minute_adjust(char *s_minute)
{
    if(strcmp(s_minute,"0")==0)
        strcpy(s_minute,"00");
    if(strcmp(s_minute,"1")==0)
        strcpy(s_minute,"01");
    if(strcmp(s_minute,"2")==0)
        strcpy(s_minute,"02");
    if(strcmp(s_minute,"3")==0)
        strcpy(s_minute,"03");
    if(strcmp(s_minute,"4")==0)
        strcpy(s_minute,"04");
    if(strcmp(s_minute,"5")==0)
        strcpy(s_minute,"05");
    if(strcmp(s_minute,"6")==0)
        strcpy(s_minute,"06");
    if(strcmp(s_minute,"7")==0)
        strcpy(s_minute,"07");
    if(strcmp(s_minute,"8")==0)
        strcpy(s_minute,"08");
    if(strcmp(s_minute,"9")==0)
        strcpy(s_minute,"09");
}

//功能：点击“下一事件”后根据事件来调整时间
//输入：时间，事件，和事件是否发生变量的地址
//输出：无
```

```
void time_adjust_plus(int *time, int *times, int *wht_happen)
{
    (*times)++;
    *wht_happen=0;
    if (*times==0)              //当事件为 0，则说明是 16 点，调整相应时间
    {
        *time=60*16;
    }
    if (*times==1)              //当事件为 1，则说明是 18 点，调整相应时间
    {
        *time=60*18;
    }
    if (*times==2)              //当事件为 2，则说明是 18 点半，调整相应时间
    {
        *time=60*18+30;
    }
    if (*times==3)              //当事件为 3，则说明是 21 点，调整相应时间
    {
        *time=60*21;
    }
    if (*times==4)              //当事件为 4，则说明是 1 点，调整相应时间
    {
        *time=1*60;
    }
    if (*times==5)              //当事件为 5，则说明是 6 点，调整相应时间
    {
        *time=6*60;
    }
}
```

----------manbody.c----------

```
#include "title.h"
/*************************************************************
函数列表:    1.void paint_man(CASE case_state, int identity)          //第一次画人的时候调用，会画出人的正面
             2.void man_forebody(CASE case_state, int identity)       //人的正面，包括主人与陌生人
             3.void man_backbody(CASE case_state)                     //人的背面
             4.void man_rightbody(CASE case_state)                    //人的右面
             5.void man_leftbody(CASE case_state)                     //人的左面
             6.void  man_sleep(CASE  case_state)                      //睡觉时的人像
             7.void man_getup(CASE case_state)                        //起床
             8.void sit_1(CASE case_state)                            //背面坐姿，
```

```
工作和吃饭时调用
                9.void sit_2(CASE case_state)                                    //侧面坐姿，
娱乐时调用
*****************************************************************/

//函数功能：第一次画人时，将人的背景储存，并画出人的正面
//入口参数：表示人的结构体，以及一个表身份的形参 identity，
//          identity 为 MASTER(即 1)时为主人，为 INTRADER(即 0)时为入侵者
void paint_man(CASE case_state, int identity)
{
    //获取画人的空间。第一次画人的时候用此函数，主要是为了 get_image
    get_image_man(case_state.xpixel, case_state.ypixel);
    man_forebody(case_state, identity);
}

//函数功能：画出人的正面
//入口参数同上面的 paint_man
void man_forebody(CASE case_state, int identity)   //x-20,x+20,y+80,y
{
    int color[2][5] = {{HAIRI, SKINI, EYESI, ARMSI, PANTSI},{HAIRM, SKINM, EYESM, ARMSM,
PANTSM}};
    ///color[0]是画主人用的颜色，color[1]是陌生人的。
    ///注意到 identity 是 0 和 1，正好可以表示数组的第一维和第二维，这样处理可以减去
一大批重复代码

    //头
    bar(case_state.xpixel-9, case_state.ypixel+1, case_state.xpixel+9, case_state.ypixel+8,
color[identity][0]);      //头发
    bar(case_state.xpixel-9, case_state.ypixel+9, case_state.xpixel+9, case_state.ypixel+19,
color[identity][1]);      //皮肤
    FillCircle(case_state.xpixel-4, case_state.ypixel+10, 2, 0);
    FillCircle(case_state.xpixel-4, case_state.ypixel+10, 1, color[identity][2]);          //左眼，眼
珠
    FillCircle(case_state.xpixel+4, case_state.ypixel+10, 2, 0);
    FillCircle(case_state.xpixel+4, case_state.ypixel+10, 1, color[identity][2]);          //右眼，眼
珠
    Horizline(case_state.xpixel-3, case_state.ypixel+16, 6, 0);      //嘴

    //脖子
    bar(case_state.xpixel-4,              case_state.ypixel+20,              case_state.xpixel+4,
case_state.ypixel+25,color[identity][1]);

    //上身
    bar(case_state.xpixel-18,             case_state.ypixel+25,              case_state.xpixel+18,
```

```
case_state.ypixel+50, color[identity][3]);        //包含手的部分
    linever(case_state.xpixel-10,        case_state.ypixel+35,        case_state.xpixel-10,
case_state.ypixel+55, 1, 0);
    linever(case_state.xpixel+10,        case_state.ypixel+35,        case_state.xpixel+10,
case_state.ypixel+55, 1, 0);
    bar(case_state.xpixel-9, case_state.ypixel+50, case_state.xpixel+9, case_state.ypixel+55,
color[identity][3]);        //衣服
    bar(case_state.xpixel-18, case_state.ypixel+50, case_state.xpixel-11, case_state.ypixel+55,
color[identity][1]);        //手
    bar(case_state.xpixel+11,        case_state.ypixel+50,        case_state.xpixel+18,
case_state.ypixel+55, color[identity][1]);

    //腿
    bar(case_state.xpixel-9, case_state.ypixel+55, case_state.xpixel+9, case_state.ypixel+73,
color[identity][4]);        //裤子
    linever(case_state.xpixel, case_state.ypixel+55, case_state.xpixel, case_state.ypixel+80, 1,
0);

    //脚（鞋子）
    bar(case_state.xpixel-9, case_state.ypixel+74, case_state.xpixel+9, case_state.ypixel+80,
0);
}

//画人的背面，传入表示人的结构体提供画人像的位置
void man_backbody(CASE case_state)
{
     //头
    bar(case_state.xpixel-9, case_state.ypixel+1, case_state.xpixel+9, case_state.ypixel+19,
41605);     //赭黄，头发

    //脖子
    bar(case_state.xpixel-4,        case_state.ypixel+20,        case_state.xpixel+4,
case_state.ypixel+25,63222);     //米黄色，肤色

    //上身
    bar(case_state.xpixel-18,        case_state.ypixel+25,        case_state.xpixel+18,
case_state.ypixel+50, 65504);       //包含手的部分
    linever(case_state.xpixel-10,        case_state.ypixel+35,        case_state.xpixel-10,
case_state.ypixel+55, 1, 0);
    linever(case_state.xpixel+10,        case_state.ypixel+35,        case_state.xpixel+10,
case_state.ypixel+55, 1, 0);
    bar(case_state.xpixel-9, case_state.ypixel+50, case_state.xpixel+9, case_state.ypixel+55,
65504);       //穿着亚麻色的衣服
    bar(case_state.xpixel-18, case_state.ypixel+50, case_state.xpixel-11, case_state.ypixel+55,
```

```
63222);
    bar(case_state.xpixel+11, case_state.ypixel+50, case_state.xpixel+18,
case_state.ypixel+55, 63222);

    //腿
    bar(case_state.xpixel-9, case_state.ypixel+55, case_state.xpixel+9, case_state.ypixel+73,
29186);          //乌贼墨色的裤子
    linever(case_state.xpixel, case_state.ypixel+55, case_state.xpixel, case_state.ypixel+80, 1,
0);

    //脚（鞋子）
    bar(case_state.xpixel-9, case_state.ypixel+74, case_state.xpixel+9, case_state.ypixel+80,
0);
}

//画人的右面，传入表示人的结构体提供画人像的位置
void man_rightbody(CASE case_state)          //x-10, x+10, y+80,y
{
    //头
    bar(case_state.xpixel-10, case_state.ypixel, case_state.xpixel-1, case_state.ypixel+20,
41605);
    bar(case_state.xpixel-1, case_state.ypixel, case_state.xpixel+10, case_state.ypixel+8,
41605);
    bar(case_state.xpixel-1, case_state.ypixel+9, case_state.xpixel+10, case_state.ypixel+20,
63222);
    FillCircle(case_state.xpixel+4, case_state.ypixel+10, 2, 0);
    FillCircle(case_state.xpixel+4, case_state.ypixel+10, 1, 65535);
    Horizline(case_state.xpixel+6, case_state.ypixel+18, 3, 0);

    //脖子
    bar(case_state.xpixel-4, case_state.ypixel+21, case_state.xpixel+4, case_state.ypixel+25,
63222);

    //上身
    bar(case_state.xpixel-10, case_state.ypixel+26, case_state.xpixel+10,
case_state.ypixel+55, 65504);
    linever(case_state.xpixel-5, case_state.ypixel+28, case_state.xpixel-5,
case_state.ypixel+55, 1, 0);
    linever(case_state.xpixel+5, case_state.ypixel+28, case_state.xpixel+5,
case_state.ypixel+55, 1, 0);
    bar(case_state.xpixel-4, case_state.ypixel+50, case_state.xpixel+5, case_state.ypixel+55,
63222);    //手
    Horizline(case_state.xpixel-5, case_state.ypixel+55, 10, 0);
```

```
    //腿
    bar(case_state.xpixel-8, case_state.ypixel+55, case_state.xpixel+8, case_state.ypixel+73,
29186);          //乌贼墨色的裤子
    //脚（鞋子）

    bar(case_state.xpixel-8, case_state.ypixel+74, case_state.xpixel+8, case_state.ypixel+80,
0);
}

//画人的左面，传入表示人的结构体提供画人像的位置
void man_leftbody(CASE case_state)
{
    //头
    bar(case_state.xpixel+1, case_state.ypixel, case_state.xpixel+10, case_state.ypixel+20,
41605);
    bar(case_state.xpixel-10, case_state.ypixel, case_state.xpixel+1, case_state.ypixel+8,
41605);
    bar(case_state.xpixel-10, case_state.ypixel+9, case_state.xpixel+1, case_state.ypixel+20,
63222);
    FillCircle(case_state.xpixel-4, case_state.ypixel+10, 2, 0);
    FillCircle(case_state.xpixel-4, case_state.ypixel+10, 1, 65535);
    Horizline(case_state.xpixel-6, case_state.ypixel+18, -3, 0);

    //脖子
    bar(case_state.xpixel-4, case_state.ypixel+21, case_state.xpixel+4, case_state.ypixel+25,
63222);

    //上身
    bar(case_state.xpixel-10,          case_state.ypixel+26,          case_state.xpixel+10,
case_state.ypixel+55, 65504);
    linever(case_state.xpixel-5, case_state.ypixel+28, case_state.xpixel-5, case_state.y+55, 1,
0);
    linever(case_state.xpixel+5,          case_state.ypixel+28,          case_state.xpixel+5,
case_state.ypixel+55, 1, 0);
    bar(case_state.xpixel-4, case_state.ypixel+50, case_state.xpixel+5, case_state.ypixel+55,
63222);    //手
    Horizline(case_state.xpixel-5, case_state.ypixel+55, 10, 0);

    //腿
    bar(case_state.xpixel-8, case_state.ypixel+55, case_state.xpixel+8, case_state.ypixel+73,
29186);          //乌贼墨色的裤子

    //脚（鞋子）
    bar(case_state.xpixel-8, case_state.ypixel+74, case_state.xpixel+8, case_state.ypixel+80,
```

```
0);
}

//函数功能：time_line 中，人去睡觉时，将站着的人像以类似 put_image 的方式移除，恢复背景，再在床上画出人睡觉的图像
//入口参数：表示人的结构体，提供画人像的位置
void man_sleep(CASE case_state) //上床睡觉
{
    //直接床上画人的头即可，不用 get、put，但是要事先把站着的人的图像隐藏，也即，put
    put_image_man(case_state.xpixel, case_state.ypixel);
    case_state.xpixel = 0.75*40+5;          //头的位置，暂时改变，+5 是因为给床多加了 5 个偏移量
    case_state.ypixel = 6.8*40;

    //头
    bar(case_state.xpixel-9, case_state.ypixel+1, case_state.xpixel+9, case_state.ypixel+8, HAIRM);     //头发
    bar(case_state.xpixel-9, case_state.ypixel+9, case_state.xpixel+9, case_state.ypixel+19, SKINM);     //皮肤

    Horizline(case_state.xpixel-6, case_state.ypixel+10, 4, 0);       //左眼（闭眼）
    Horizline(case_state.xpixel+2, case_state.ypixel+10, 4, 0);       //右眼（闭眼）
    Horizline(case_state.xpixel-3, case_state.ypixel+16, 6, 0);      //嘴

    //脖子
    bar(case_state.xpixel-4, case_state.ypixel+20, case_state.xpixel+4, case_state.ypixel+21,SKINM);
}

//函数功能：把床上睡觉的人去除，并在相应位置画出站立的人
//入口参数：表示人的结构体，提供画人像的位置
void man_getup(CASE case_state) //起床
{
    //重画一张床，就可以把头覆盖了。再把人画出来。
    bed(0, 7);
    paint_man(case_state, MASTER);
}

//人工作时和吃饭时的背面坐姿，传入表示人的结构体提供画人像的位置
void sit_1(CASE case_state)   //背面坐姿
{
    //头，实际上只需要画头发
    bar(case_state.xpixel-9, case_state.ypixel+1, case_state.xpixel+9, case_state.ypixel+19,
```

```
41605);      //赭黄，头发

    //脖子
    bar(case_state.xpixel-4,    case_state.ypixel+20,    case_state.xpixel+4,
case_state.ypixel+25,63222);      //米黄色，肤色

    //上身
    bar(case_state.xpixel-18,    case_state.ypixel+25,    case_state.xpixel+18,
case_state.ypixel+40, 65504);      //包含手的部分
    bar(case_state.xpixel-9, case_state.ypixel+40, case_state.xpixel+9, case_state.ypixel+50,
65504);
    bar(case_state.xpixel-9, case_state.ypixel+50, case_state.xpixel+9, case_state.ypixel+55,
65504);      //穿着亚麻色的衣服
    linever(case_state.xpixel-10,    case_state.ypixel+35,    case_state.xpixel-10,
case_state.ypixel+55, 1, 0);
    linever(case_state.xpixel+10,    case_state.ypixel+35,    case_state.xpixel+10,
case_state.ypixel+55, 1, 0);

    //屁股
    bar(case_state.xpixel-9, case_state.ypixel+55, case_state.xpixel+9, case_state.ypixel+63,
29186);      //乌贼墨色的裤子
}

//人娱乐时的侧面坐姿，传入表示人的结构体提供画人像的位置
void sit_2(CASE case_state)  //侧面坐姿
{
     //头
    bar(case_state.xpixel+1,  case_state.ypixel,  case_state.xpixel+10,  case_state.ypixel+20,
41605);
    bar(case_state.xpixel-10,  case_state.ypixel,  case_state.xpixel+1,  case_state.ypixel+8,
41605);
    bar(case_state.xpixel-10, case_state.ypixel+9, case_state.xpixel+1, case_state.ypixel+20,
63222);
    FillCircle(case_state.xpixel-4, case_state.ypixel+10, 2, 0);
    FillCircle(case_state.xpixel-4, case_state.ypixel+10, 1, 65535);
    Horizline(case_state.xpixel-6, case_state.ypixel+18, -3, 0);

    //脖子
    bar(case_state.xpixel-4, case_state.ypixel+21, case_state.xpixel+4, case_state.ypixel+25,
63222);

    //上身
    bar(case_state.xpixel-10,    case_state.ypixel+26,    case_state.xpixel+10,
case_state.ypixel+55, 65504);
```

```
    linever(case_state.xpixel-5, case_state.ypixel+28, case_state.xpixel-5, case_state.y+55, 1, 0);
    linever(case_state.xpixel+5, case_state.ypixel+28, case_state.xpixel+5, case_state.ypixel+55, 1, 0);
    bar(case_state.xpixel-4, case_state.ypixel+50, case_state.xpixel+5, case_state.ypixel+55, 63222);    //手
    Horizline(case_state.xpixel-5, case_state.ypixel+55, 10, 0);

    //腿
    bar(case_state.xpixel-18, case_state.ypixel+55, case_state.xpixel+8, case_state.ypixel+62, 29186);          //乌贼墨色的裤子
    bar(case_state.xpixel-18, case_state.ypixel+62, case_state.xpixel-6, case_state.ypixel+70, 29186);
    bar(case_state.xpixel-18, case_state.ypixel+70, case_state.xpixel-6, case_state.ypixel+76, 0);
}
```

----------module.c.----------

```
#include "title.h"

/*************************************************
错误提示模块及组件
void overflow_box(int x,int y);
void null_box(int x,int y);
void space_box(int x,int y);
void phone_module(int x,int y);
void phone_back(int x,int y);
void gg_bar();
void FindWay_error(int x,int y);
void trap(int x,int y);
void pop_error(int x,int y);
*************************************************/

/*************************************************
这个文件里面有用栈实现的部分，要改成用图
*************************************************/



void overflow_box(int x,int y)//当 malloc 分配空间失败时提示，回车退出系统
{
    bar(x-350,y-100,x+350,y-70,255);
    bar(x-350,y-70,x+350,y+100,46651);
```

```
        bar_round(x+335,y-85,28,28,2,1,63488);

        fdhz(x-75,y+10,2,2,"内",0);
        fdhz(x-30,y+10,2,2,"存",0);
        fdhz(x+15,y+10,2,2,"溢",0);
        fdhz(x+60,y+10,2,2,"出",0);

        fdhz(x-75,y+53,2,2,"回",0);
        fdhz(x-45,y+53,2,2,"车",0);
        fdhz(x,y+53,2,2,"以",0);
        fdhz(x+45,y+53,2,2,"退",0);
        fdhz(x+75,y+53,2,2,"出",0);
}

void null_box(int x,int y)//当找不到相应文件路径时显示，回车退出系统
{
        bar(x-350,y-100,x+350,y-70,255);
        bar(x-350,y-70,x+350,y+100,46651);
        bar_round(x+335,y-85,28,28,2,1,63488);

        fdhz(x-115,y-20,2,2,"找",0);
        fdhz(x-75,y-20,2,2,"不",0);
        fdhz(x-30,y-20,2,2,"到",0);
        fdhz(x+15,y-20,2,2,"该",0);
        fdhz(x+60,y-20,2,2,"路",0);
        fdhz(x+100,y-20,2,2,"径",0);


        fdhz(x-75,y+23,2,2,"回",0);
        fdhz(x-45,y+23,2,2,"车",0);
        fdhz(x,y+23,2,2,"以",0);
        fdhz(x+45,y+23,2,2,"退",0);
        fdhz(x+75,y+23,2,2,"出",0);
}

void space_box(int x,int y)//显示堆区当前可分配空间
{
        unsigned long i,ki;
        int mi,gi;
        char a[8];                  //a 是以 b 为单位的数值，ki 是 k 为单位的数值，mi 是 M 为单位
的数值，gi 是 G
        i = coreleft();
        ki = i/1024;
        mi = ki/1024;
```

```
gi = mi/1024;
if(gi)
{
	itoa(gi,a,10);
	bar(x-350,y-100,x+350,y-70,255);
	bar(x-350,y-70,x+350,y+100,46651);
	bar_round(x+335,y-85,28,28,2,1,63488);
	fdhz(x-135,y+10,2,2,"剩余内存：",0);
	outtextxy(x+100,y+10,&a,1,1,10,0);
	outtextxy(x+300,y+10,"G",1,1,10,0);
	fdhz(x-75,y+53,2,2,"回",0);
	fdhz(x-45,y+53,2,2,"车",0);
	fdhz(x,y+53,2,2,"以",0);
	fdhz(x+45,y+53,2,2,"退",0);
	fdhz(x+75,y+53,2,2,"出",0);
}
else
{
	if(mi)
	{
		itoa(mi,a,10);
		bar(x-350,y-100,x+350,y-70,255);
		bar(x-350,y-70,x+350,y+100,46651);
		bar_round(x+335,y-85,28,28,2,1,63488);
		fdhz(x-135,y+10,2,2,"剩余内存：",0);
		outtextxy(x+100,y+10,&a,1,1,10,0);
		outtextxy(x+300,y+10,"M",1,1,10,0);
		fdhz(x-75,y+53,2,2,"回",0);
		fdhz(x-45,y+53,2,2,"车",0);
		fdhz(x,y+53,2,2,"以",0);
		fdhz(x+45,y+53,2,2,"退",0);
		fdhz(x+75,y+53,2,2,"出",0);
	}
	else
	{
		if(ki)
		{
			itoa(ki,a,10);
			bar(x-350,y-100,x+350,y-70,255);
			bar(x-350,y-70,x+350,y+100,46651);
			bar_round(x+335,y-85,28,28,2,1,63488);
			fdhz(x-135,y+10,2,2,"剩余内存：",0);
			outtextxy(x+100,y+10,&a,1,1,10,0);
			outtextxy(x+300,y+10,"k",1,1,10,0);
```

```
                fdhz(x-75,y+53,2,2,"回",0);
                fdhz(x-45,y+53,2,2,"车",0);
                fdhz(x,y+53,2,2,"以",0);
               fdhz(x+45,y+53,2,2,"退",0);
                fdhz(x+75,y+53,2,2,"出",0);
            }
            else
            {
                itoa((int)i,a,10);
                bar(x-350,y-100,x+350,y-70,255);
                bar(x-350,y-70,x+350,y+100,46651);
                bar_round(x+335,y-85,28,28,2,1,63488);
                fdhz(x-135,y+10,2,2,"剩余内存：",0);
                outtextxy(x+100,y+10,&a,1,1,10,0);
                outtextxy(x+300,y+10,"b",1,1,10,0);
                fdhz(x-75,y+53,2,2,"回",0);
                fdhz(x-45,y+53,2,2,"车",0);
                fdhz(x,y+53,2,2,"以",0);
                fdhz(x+45,y+53,2,2,"退",0);
                fdhz(x+75,y+53,2,2,"出",0);
            }
        }
    }


}


void phone_module(int x,int y)//手机功能模块外框
{
    bar_round_with_shadow(x,y,120,120,30,1,1788);
}
/*home 键不想要了，改*/
void phone_back(int x,int y)//home 键
{
    FillCircle(x,y,40,65535);
    circle(x,y,35,0);
    circle(x,y,40,0);
}

void gg_bar()
{
    bar(0,0,1024,768,65535);
    fdhz(200,200,3,3,"啥也没有，再见",0);
```

```
	getch();
	exit(1);
}

void FindWay_error(int x,int y)//寻路失败提示（退出）
{
	bar(x-350,y-100,x+350,y-70,255);
	bar(x-350,y-70,x+350,y+100,46651);
	bar_round(x+335,y-85,28,28,2,1,63488);
	fdhz(x-115,y-20,2,2,"寻",0);
	fdhz(x-75,y-20,2,2,"找",0);
	fdhz(x-30,y-20,2,2,"路",0);
	fdhz(x+15,y-20,2,2,"径",0);
	fdhz(x+60,y-20,2,2,"出",0);
	fdhz(x+100,y-20,2,2,"错",0);

	fdhz(x-75,y+23,2,2,"回",0);
	fdhz(x-45,y+23,2,2,"车",0);
	fdhz(x,y+23,2,2,"以",0);
	fdhz(x+45,y+23,2,2,"退",0);
	fdhz(x+75,y+23,2,2,"出",0);
}

void trap(int x,int y)//被障碍物挡住提示
{
	bar(x-350,y-100,x+350,y-70,255);
	bar(x-350,y-70,x+350,y+100,46651);
	bar_round(x+335,y-85,28,28,2,1,63488);
	fdhz(x-50,y-70,2,2,"被挡住了",0);
	fdhz(x-115,y,2,2,"请移开障碍物",0);
	fdhz(x-75,y+30,2,2,"回车以继续",0);
}
/*改成图的，别用栈了*/
void pop_error(int x,int y)//弹出栈顶元素失败（栈空）
{
	bar(x-350,y-100,x+350,y-70,255);
	bar(x-350,y-70,x+350,y+100,46651);
	bar_round(x+335,y-85,28,28,2,1,63488);
	fdhz(x-50,y-30,2,2,"栈已为空",0);
	fdhz(x-115,y,2,2,"取栈顶失败",0);


	fdhz(x-75,y+23,2,2,"回",0);
	fdhz(x-45,y+23,2,2,"车",0);
```

```
        fdhz(x,y+23,2,2,"以",0);
        fdhz(x+45,y+23,2,2,"退",0);
        fdhz(x+75,y+23,2,2,"出",0);
}
```

----------mouse.c----------

```
#include "title.h"

union REGS regs;

int arrowMouse[10][16] = {
{ 1,1,1,1,1,1,1,1,1,1,1,1,3,3,3 },
{ 1,0,0,0,0,0,0,0,0,0,0,1,3,3,3,3 },
{ 3,1,0,0,0,0,0,0,0,0,1,3,3,3,3,3 },
{ 3,3,1,0,0,0,0,0,0,1,3,3,3,3,3,3 },
{ 3,3,3,1,0,0,0,0,0,0,1,1,3,3,3,3 },
{ 3,3,3,3,1,0,0,0,0,0,0,0,1,1,3,3 },
{ 3,3,3,3,3,1,0,0,1,1,1,0,0,0,1,3 },
{ 3,3,3,3,3,3,1,0,1,3,3,1,0,0,1,3 },
{ 3,3,3,3,3,3,3,1,1,3,3,3,1,1,3,3 },
{ 3,3,3,3,3,3,3,3,1,3,3,3,3,3,3,3 },
};
int AddMouse[10][16]={
{ 3,3,3,3,3,3,3,1,1,1,1,1,3,3,3,3 },
{ 3,3,3,3,3,3,3,1,0,0,0,1,3,3,3,3 },
{ 3,3,3,3,3,3,3,1,0,0,0,1,3,3,3,3 },
{ 3,3,1,1,1,1,1,1,0,0,0,1,1,1,1,3 },
{ 3,3,1,0,0,0,0,0,0,0,0,0,0,1,3,3 },
{ 3,3,1,0,0,0,0,0,0,0,0,0,0,1,3,3 },
{ 3,3,1,1,1,1,1,1,0,0,0,1,1,1,1,3 },
{ 3,3,3,3,3,3,3,1,0,0,0,1,3,3,3,3 },
{ 3,3,3,3,3,3,3,1,0,0,0,1,3,3,3,3 },
{ 3,3,3,3,3,3,3,1,1,1,1,1,3,3,3,3 },
};

int InputMouse[10][16]={
{ 0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
```

```
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0 },
{ 0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0 },
};
int Mouse[10][16] = {
	{ 1,1,1,1,1,1,1,1,1,1,1,1,3,3,3 },
	{ 1,0,0,0,0,0,0,0,0,0,0,1,3,3,3,3 },
	{ 3,1,0,0,0,0,0,0,0,0,1,3,3,3,3,3 },
	{ 3,3,1,0,0,0,0,0,0,1,3,3,3,3,3,3 },
	{ 3,3,3,1,0,0,0,0,0,0,1,1,3,3,3,3 },
	{ 3,3,3,3,1,0,0,0,0,0,0,0,1,1,3,3 },
	{ 3,3,3,3,3,1,0,0,1,1,1,0,0,0,1,3 },
	{ 3,3,3,3,3,3,1,0,1,3,3,1,0,0,1,3 },
	{ 3,3,3,3,3,3,3,1,1,3,3,3,1,1,3,3 },
	{ 3,3,3,3,3,3,3,3,1,3,3,3,3,3,3,3 },
};
int MouseSave[10][16] = {0};


/*mark 为 0 设置为箭头鼠标,mark 为 1 设置为加号鼠标*/
void setMouseShape(int mark,int mx,int my)
{
	int i;
	int j;


	if (mark == 0)
	{
		for(i=0;i<10;i++)
			for (j = 0;j < 16;j++)
				Mouse[i][j] = arrowMouse[i][j];
	}
	else if (mark == 1)
	{
		for (i = 0;i<10;i++)
			for (j = 0;j < 16;j++)
				Mouse[i][j] = AddMouse[i][j];
	}
	else
	{
		printf("MouseShape doesn't exit!");
		getch();
		exit(1);
	}
```

```
        mousehide(mx,my);
        cursor(mx, my);
}


void cursor(int x, int y)/*画鼠标*/
{
        int i, j;
        for (i = 0;i<10;i++)
                for (j = 0;j<16;j++)
                {
                        if (Mouse[i][j] == 0)
                                Putpixel64k(x + i, y + j, 65535);
                        else if (Mouse[i][j] == 1)
                                Putpixel64k(x + i, y + j, 0);
                }
}


void getMousebk(int x, int y)/*获取点的颜色*/
{
        int i, j;
        for(i=0;i<10;i++)
                for(j=0;j<16;j++)
                                MouseSave[i][j] = Getpixel64k(x + i, y + j);
}


void mousehide(int x, int y)
{
        int i, j;
        for (i = 0;i<10;i++)
                for (j = 0;j<16;j++)
                {
                        Putpixel64k(x + i, y + j, MouseSave[i][j]);
                }
}

int init()   //鼠标器初始化操作
{
        int retcode;
        regs.x.ax = 0;
        int86(51, &regs, &regs);
        retcode = regs.x.ax;
```

```
    if (retcode == 0)
        return 0;
    regs.x.ax = 7;
    regs.x.cx = xmi;
    regs.x.dx = xma;
    int86(51, &regs, &regs);
    regs.x.ax = 8;
    regs.x.cx = ymi;
    regs.x.dx = yma;
    int86(51, &regs, &regs);
    return retcode;
}
void mouseInit(int *mx,int *my, int *button)
{
    int u=init();
    if ( u == 0)
    {
        //closegraph();
        printf("Mouse or Mouse Driver Absent,Please Install");
        delay(5000);
        exit(1);
    }

    *mx = 3;
    *my = 460;
    *button = 0;

    getMousebk(*mx, *my);
    cursor(*mx, *my);

}


int readxy(int *mx, int *my, int *button)          //读取鼠标的位置
{
    static int mark = 0;   //按键按松开标志
    int xx0 = *mx, yy0 = *my, buto = *button;
    int xnew, ynew;
    do
    {

        regs.x.ax = 3;
        int86(51, &regs, &regs);
```

```
            xnew = regs.x.cx;
            ynew = regs.x.dx;
            if (mark == 0 && regs.x.bx != 0)
            {
                mark = 1;
                //delay(10);
                if(regs.x.bx != 0)*button = regs.x.bx;
            }
            else if (regs.x.bx == 0)
            {
                mark = 0;
                *button = 0;
            }
            else *button = 0;


        } while (xnew == xx0&&ynew == yy0&&*button == buto);
        *mx = xnew;
        *my = ynew;
        if (*button)
        {
            *mx = xnew;
            *my = ynew;
            return -1;
        }
        else
        {
            *mx = xnew;
            *my = ynew;
            return 1;
        }
}

void newxy(int *mx, int *my, int *mbutt)       //在新的位置处画鼠标
{
        static int i = 0;
        int ch, xx0 = *mx, yy0 = *my;
        int xm, ym;


        readxy(&xm, &ym, mbutt);


        if (xm != xx0 || ym != yy0)
```

```
    {
        mousehide(xx0, yy0);
        getMousebk(xm, ym);
        cursor(xm, ym);
        //mousehide(xx0, yy0);
        *mx = xm;
        *my = ym;
    }
}

void backgroundChange(int mx, int my,int x1,int y1,int x2,int y2)/*为了输入的时候不遮住鼠标*/
{
    int i, j;
    int mark = 0;

    for(i=0;i<10;i++)
        for (j = 0;j < 16;j++)
        {
            if (mx + i >= x1&&mx + i <= x2&&my + j >= y1&&my + j <= y2)
            {
                MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                mark = 1;
            }
        }
    if (mark == 1)
    {
        mousehide(mx,my);
        getMousebk(mx, my);
        cursor(mx, my);
    }
}

void AddFrame(int mx, int my, int x1, int y1, int x2, int y2,int thick,int color)
{
    int i, j;

    bar(x1, y1, x2, y2,color);



    if (thick == 3)
    {
        for (i = 0;i < 10;i++)
```

```
                for (j = 0;j < 16;j++)
                {
                    if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 || my + j == y2))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                    if (my + j <= y2&&my + j >= y1&&mx + i == x1 || mx + i == x2)
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);

                    if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 + 1 || my + j == y2 + 1))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                    if (my + j <= y2&&my + j >= y1&&(mx + i == x1 + 1 || mx + i == x2 + 1))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);

                    if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 - 1 || my + j == y2 - 1))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                    if (my + j <= y2&&my + j >= y1&&(mx + i == x1 - 1 || mx + i == x2 - 1))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                }
        }
        else if (thick == 1)
        {
            for (i = 0;i < 10;i++)
                for (j = 0;j < 16;j++)
                {
                    if (mx + i <= x2&&mx + i >= x1&&(my + j == y1 || my + j == y2))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                    if (my + j <= y2&&my + j >= y1&&(mx + i == x1 || mx + i == x2))
                        MouseSave[i][j] = Getpixel64k(mx + i, my + j);
                }
        }
}

void reset_mouse(int *x,int *y)
{
        getMousebk(*x,*y);
}

void esc_init(BUTTONS *esc1)
{
        int x=900,y=50,wide=50,height=25;
        bar(x-wide/2,y-height/2,x+wide/2,y+height/2,transcolor(176,224,230));
        outtextxy(x-10,y-5,"esc",1,1,10,44373);

        esc1->x = x-wide/2;
        esc1->y = y-height/2;
```

```
	esc1->wide = wide;
	esc1->height = height;
	esc1->click=0;
	esc1->over=0;

}

int esc_check(BUTTONS *esc1,int *x,int *y,int *button)
{

	if(*x>=esc1->x	&&	*y>=esc1->y	&&	*x<=esc1->x+esc1->wide	&&
*y<=esc1->y+esc1->height)
	{
		esc1->over=1;

	}
	if(*x<=esc1->x	||	*y<=esc1->y	||	*x>=esc1->x+esc1->wide	||
*y>=esc1->y+esc1->height)
	{
		esc1->over=0;

	}
	/*if(esc1->over)
	{
		outtextxy(890,45,"esc",1,1,10,0);
	}
	else
	{
		outtextxy(890,45,"esc",1,1,10,44373);
	}*/

	if(esc1->over && *button)
	{
		return 1;//退出系统
	}
}
/*

void CheckStack()
{
	_SP;
	outtextxy(10,100,,1,1,10,65535);
}
*/
```

```
void CheckHeap(int i)
{
    int num;
    static int j=0;
    if(j<i)
    {
        j++;
        return ;
    }
    num=0;
    bar(0,0,1024,768,0);
    while(malloc(1024))num++;
    textmode(0);
    printf("%d",num);
    getch();
    exit(0);
}
```

----------myhouse.c----------

```
#include "title.h"
/*****************************************************************************
函数列表:    1.paint_floor();        //画地板
             2.paint_wall();         //画墙壁
             3.paint_furniture();    //画家具
功能:        调用封装在 bricks.c、wall.c 和 frnture.c 中的函数，画出房子主界面
*****************************************************************************/

void paint_house()
{
    paint_floor();        //画地板
    paint_wall();         //画墙壁
    paint_furniture();    //画家具
}

void paint_floor()        //地板
{
    int i, j;

    bar(3*40, 1*40, 7*40, 3*40, 65502);          //整体背景画成白色，如果不这样处理，背景是黑的

    for(i = 3; i < 7; i++)        //大门处
    {
        for(j = 3; j < 6; j++)
```

```
        {
            wood_ver(i, j);
        }
    }

    for(i = 7; i < 16; i++)          //客厅主体
    {
        for(j = 3; j < 19; j++)
        {
            wood_ver(i, j);
        }
    }

    ///以上把客厅部分画完了，下面是各个小房间

    for(i = 0; i < 3; i++)       //bathroom
    {
        for(j = 1; j < 6; j++)
        {
            glass(i, j);
        }
    }

    for(j = 1; j < 3; j++)       //kitchen
    {
        for(i = 7; i < 16; i++) //i 的循环更长，放里面提高效率
        {
            green_kitchen(i, j);
        }
    }

    for(i = 0; i < 7; i++)       //bedroom
    {
        for(j = 7; j < 19; j++)
        {
            green_bedroom(i, j);
        }
    }

}

void paint_wall()            //墙壁
{
    int i, j;
```

```
    for(i = 0; i < 7; i++)          //顶上的和卧室的
    {
        w_blue(i, 0);
        w_blue(i, 6);
    }

    for(i = 7; i < 16; i++)        //顶上的（因为卧室只画到 i = 6 为止，顶上的剩余部分单独遍历）
    {
        w_blue(i, 0);
    }

    for(j = 11; j < 19; j++)        //卧室与客厅交界
    {
        w_right(6, j);
    }

    for(j = 0; j < 19; j++)        //房子最右
    {
        w_right(15, j);
    }

    for(j = 1; j < 3; j++)   //厨房左
    {
        w_left(7, j);
    }

    for(j = 1; j < 4; j++)
    {
        w_right(2, j);            //浴室
    }

}

void paint_furniture()   //家具
{
    //bedroom
    bed(0, 7);
    rect_table(3, 7);
    aircon(3, 6, 0);
    bookshelf(6, 11);
    bookshelf(6, 12);
    bookshelf(6, 13);
```

```
    bookshelf(6, 14);
    cupboard(6, 11);
    rect_table(0, 13);
    desk(0, 15);
    seat(0, 12);
    seat(2, 14);
    seat(0, 16);
    pc(0, 15);

    //reception room
    rect_table(11, 12);
    desk(12, 8);
    seat(12, 7);      //饭桌旁边的小板凳
    seat(13, 7);
    seat(11, 8);
    seat(14, 8);
    seat(12, 9);
    seat(13, 9);
    TV(7, 10.5);
    sofa_main(14, 11);
    sofa_up(11, 10);
    sofa_down(11, 14.5);

    //bathroom
    window_close(1, 0);
    WashMach(0, 1);
    toilet(0, 3);

    //kitchen
    trashbin(7, 2);
    water_dispenser(15, 1);
    zaotai(9, 1);
    desk(12, 1);      //放水杯的桌子
    water_bottle(15.5*40, 2.4*40);
}
```

----------rbtbody.c----------

```
#include "title.h"
/*****************************************************
函数列表:    1.void paint_robot(CASE case_state)      //第一次画机器人时调用，将画机器人处的背景储存，并画出机器人正面
             2.void forebodyhead(CASE case_state)     //机器人正面
             3.void backbodyhead(CASE case_state)     //机器人背面
             4.void robot_left(CASE case_state)       //机器人左面
```

```
                5.void robot_right(CASE case_state)                    //机器人右面
                6.void robot_hand_right(int x,int y,int theta)      //机器人的手，指向右侧，
封装后在上面所述的几个函数中调用
                7.void robot_hand_left(int x,int y,int theta)       //机器人的手，指向左侧，封
装后在上面所述的几个函数中调用
                8.void right_hold(CASE case_state)                    //机器人手持物品向右
走
                9.void left_hold(CASE case_state)                     //机器人手持物品向左
走
                10.void front_hold(CASE case_state)                   //机器人手持物品向下
走（即正面）
                11.void back_hold(CASE case_state)                     //机器人手持物品向
上走（即背面）
*******************************************************/

//函数功能：第一次画人时，将机器人的背景储存，并画出机器人的正面
void paint_robot(CASE case_state)
{
      //获取机器人背景并绘制机器人
    get_image_robot(case_state.xpixel, case_state.ypixel);
    forebodyhead(case_state);
}

//画机器人的正面，传入表示机器人的结构体提供画机器人像的位置
void forebodyhead(CASE case_state)//范围：case_state.xpixel+-30;
{
    robot_hand_right(case_state.xpixel+4,case_state.ypixel+56,45);
    robot_hand_left(case_state.xpixel-28,case_state.ypixel+65,-45);
    //头
    ever_Fillellipse(case_state.xpixel-7.5,     case_state.ypixel+12,     case_state.xpixel+7.5,
case_state.ypixel+12, 9, 0);     //先画黑再覆盖，相当于有了轮廓
    ever_Fillellipse(case_state.xpixel-7.5,     case_state.ypixel+12,     case_state.xpixel+7.5,
case_state.ypixel+12, 8, 65535);

    FillCircle(case_state.xpixel-7.5, case_state.ypixel+12, 3, 1023);    //眼睛
    FillCircle(case_state.xpixel+7.5, case_state.ypixel+12, 3, 1023);
    //不要嘴巴也行
   // bow(case_state.xpixel-7.5, case_state.ypixel+12, 7.5/sin(3.14159/12), 0);
    //脚
    bar(case_state.xpixel-15,             case_state.ypixel+60,             case_state.xpixel+15,
case_state.ypixel+75, 65535);
    linever(case_state.xpixel-15,           case_state.ypixel+60,           case_state.xpixel-15,
case_state.ypixel+75, 1, 0);
    linever(case_state.xpixel+15,           case_state.ypixel+60,           case_state.xpixel+15,
```

```
case_state.ypixel+75, 1, 0);
    Horizline(case_state.xpixel-15, case_state.ypixel+70, 30, 0);
    linever(case_state.xpixel, case_state.ypixel+60, case_state.xpixel, case_state.ypixel+75, 1,
0);
    Horizline(case_state.xpixel-15, case_state.ypixel+75, 30, 0);
    //身体
    bar(case_state.xpixel-16, case_state.ypixel + 19,case_state.xpixel+16, case_state.ypixel +
61, 0);//边框
    bar(case_state.xpixel-15, case_state.ypixel + 20,case_state.xpixel+15, case_state.ypixel +
60, 65535);//入口是中心。。。
    circle(case_state.xpixel, case_state.ypixel + 40, 10, 0);    //不填充
    Horizline(case_state.xpixel-15, case_state.ypixel+60, 30, 0);
    //暂时不画红十字

}

//画机器人的背面，传入表示机器人的结构体提供画机器人像的位置
void backbodyhead(CASE case_state)//范围：x+30,-30,y-80~y;
{
    robot_hand_right(case_state.xpixel+4,case_state.ypixel+56,45);
    robot_hand_left(case_state.xpixel-28,case_state.ypixel+65,-45);
    ever_Fillellipse(case_state.xpixel-7.5,    case_state.ypixel+12,    case_state.xpixel+7.5,
case_state.ypixel+12, 9, 0);    //先画黑再覆盖，相当于有了轮廓
    ever_Fillellipse(case_state.xpixel-7.5,    case_state.ypixel+12,    case_state.xpixel+7.5,
case_state.ypixel+12, 8, 65535);
    putpixel(case_state.xpixel-7.5, case_state.ypixel+12, 0);    //眼睛
    putpixel(case_state.xpixel+7.5, case_state.ypixel+12, 0);

    //脚
    bar(case_state.xpixel-15,    case_state.ypixel+60,    case_state.xpixel+15,
case_state.ypixel+75, 65535);
    linever(case_state.xpixel-15,    case_state.ypixel+60,    case_state.xpixel-15,
case_state.ypixel+75, 1, 0);
    linever(case_state.xpixel+15,    case_state.ypixel+60,    case_state.xpixel+15,
case_state.ypixel+75, 1, 0);
    Horizline(case_state.xpixel-15, case_state.ypixel+70, 30, 0);
    linever(case_state.xpixel, case_state.ypixel+60, case_state.xpixel, case_state.ypixel+75, 1,
0);
    Horizline(case_state.xpixel-15, case_state.ypixel+75, 30, 0);

     //身体
    bar(case_state.xpixel-16, case_state.ypixel + 19,case_state.xpixel+16, case_state.ypixel +
61, 0);//边框
    bar(case_state.xpixel-15, case_state.ypixel + 20,case_state.xpixel+15, case_state.ypixel +
```

```
60, 65535);//入口是中心。。。
    Horizline(case_state.xpixel-15, case_state.ypixel+60, 30, 0);
}

//画机器人的左面，传入表示机器人的结构体提供画机器人像的位置
void robot_left(CASE case_state)//x-10~x+10,y-80~y
{
    //头
    FillCircle(case_state.xpixel, case_state.ypixel+15,12, 65535);
    FillCircle(case_state.xpixel-4, case_state.ypixel+15, 3, 1023); //眼睛

    ///手向左，头也向左，故不论左手还是右手，调用的都是 robot_hand_left 函数
    //右手
    robot_hand_left(case_state.xpixel-20,case_state.ypixel+67,-45+case_state.hand_right);

    //身体
    bar(case_state.xpixel-12,  case_state.ypixel+20,  case_state.xpixel+12,
case_state.ypixel+60, 0);
    bar(case_state.xpixel-11,  case_state.ypixel+21,  case_state.xpixel+11,
case_state.ypixel+59, 65535);
    Horizline(case_state.xpixel-5, case_state.ypixel+30, 10, 0);

    //脚
    bar(case_state.xpixel-11,  case_state.ypixel+59,  case_state.xpixel+11,
case_state.ypixel+74, 65535);
    linever(case_state.xpixel-12,  case_state.ypixel+60,  case_state.xpixel-12,
case_state.ypixel+75, 1, 0);
    linever(case_state.xpixel+12,  case_state.ypixel+60,  case_state.xpixel+12,
case_state.ypixel+75, 1, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+60, 24, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+70, 24, 0);
    Horizline(case_state.xpixel-15, case_state.ypixel+75, 30, 0);

    //左手
    robot_hand_left(case_state.xpixel-20,case_state.ypixel+71,-45-case_state.hand_right);

}

//画机器人的右面，传入表示机器人的结构体提供画机器人像的位置
void robot_right(CASE case_state)//x-10~x+10,y-80~y，手和头都向右
{
    //头
    FillCircle(case_state.xpixel, case_state.ypixel+15,12, 65535);
    FillCircle(case_state.xpixel+4, case_state.ypixel+15, 3, 1023); //眼睛
```

```
    ///手向右，头也向右，故不论左手还是右手，调用的都是 robot_hand_right 函数
    //左手
    robot_hand_right(case_state.xpixel-4,case_state.ypixel+58,45-case_state.hand_left);

    //身体
    bar(case_state.xpixel-12,  case_state.ypixel+20,  case_state.xpixel+12,
case_state.ypixel+60, 0);
    bar(case_state.xpixel-11,  case_state.ypixel+21,  case_state.xpixel+11,
case_state.ypixel+59, 65535);
    Horizline(case_state.xpixel-5, case_state.ypixel+30, 10, 0);

    //脚
    bar(case_state.xpixel-11,  case_state.ypixel+59,  case_state.xpixel+11,
case_state.ypixel+74, 65535);
    linever(case_state.xpixel-12,  case_state.ypixel+60,  case_state.xpixel-12,
case_state.ypixel+75, 1, 0);
    linever(case_state.xpixel+12,  case_state.ypixel+60,  case_state.xpixel+12,
case_state.ypixel+75, 1, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+60, 24, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+70, 24, 0);
    Horizline(case_state.xpixel-15, case_state.ypixel+75, 30, 0);

    //右手
    robot_hand_right(case_state.xpixel-4,case_state.ypixel+62,45+case_state.hand_left);

}

//函数功能：画出机器人的手，指向右侧，封装后在上面所述的几个函数中调用
//入口参数：小像素点，表示画这只手的位置；theta 角控制手摆动的角度，传入以后给
theta_bar 函数调用
void robot_hand_right(int x,int y,int theta)
{
    ///屏幕上看，这只手在大白右边，实际上是大白的左手
    ///如果想要在(0,0)弄上这只手的顶点，用的是(xpixel+10, ypixel+40)
    //color 为 0 的稍微大一个像素点的 theta_bar，就是留一个边沿（先画大一点，之后被
覆盖），color 为 65535 就是大白的颜色
    theta_bar(x-1,y-
29,(int)(12/sin((double)(theta)/180*PI)),(int)(12/sin((double)(theta)/180*PI)),theta,0);
    theta_bar(x,y-
30,(int)(10/sin((double)(theta)/180*PI)),(int)(10/sin((double)(theta)/180*PI)),theta,65535);
    FillCircle(x+(int)(16/tan((double)(theta)/180*PI))+5,y-
(int)(24*sin((double)(theta)/180*PI))-2,7,0);
    FillCircle(x+(int)(16/tan((double)(theta)/180*PI))+5,y-
```

```
(int)(24*sin((double)(theta)/180*PI))-2,6,65535);
}

//函数功能：画出机器人的手，指向左侧，封装后在上面所述的几个函数中调用
//入口参数同 robot_hand_right
void robot_hand_left(int x,int y,int theta)
{
    ///注意事项也同上面的 robot_hand_right
    ///在(xpixel,ypixel)=(40,40)画这只手，大约是(xpixel-20, ypixel+50),即(20,90)
    theta_bar(x-1,y-29,(int)(12/sin((double)(-1*theta)/180*PI)),(int)(12/sin((double)(-
1*theta)/180*PI)),theta,0);
    theta_bar(x,y-30,(int)(10/sin((double)(-1*theta)/180*PI)),(int)(10/sin((double)(-
1*theta)/180*PI)),theta,65535);
    FillCircle(x+(int)(16/tan((double)(-1*theta)/180*PI))-13,y-(int)(24*sin((double)(-
1*theta)/180*PI))-10,7,0);
    FillCircle(x+(int)(16/tan((double)(-1*theta)/180*PI))-13,y-(int)(24*sin((double)(-
1*theta)/180*PI))-10,6,65535);
}

///以下每个持物寻走的函数中，放物品的位置都是需要具体微调的，所以无法用一个统一的
画物品的函数去封装
//画出机器人手持物品向右走的图像，传入表示机器人的结构体提供画机器人像的位置
void right_hold(CASE case_state)      //向右转，看到右手侧与物体
{
    ///以下每个部件的代码和 robot_right 完全一致，但是为了得到手->物品->身体的层次
感，
    ///在画右手前插入了物品，因此不能直接调用函数解决问题只能出现重复代码代码

    //头
    FillCircle(case_state.xpixel, case_state.ypixel+15,12, 65535);
    FillCircle(case_state.xpixel+4, case_state.ypixel+15, 3, 1023); //眼睛

    //左手
    robot_hand_right(case_state.xpixel-4,case_state.ypixel+58,45+case_state.hand_left);

    //身体
    bar(case_state.xpixel-12,          case_state.ypixel+20,          case_state.xpixel+12,
case_state.ypixel+60, 0);
    bar(case_state.xpixel-11,          case_state.ypixel+21,          case_state.xpixel+11,
case_state.ypixel+59, 65535);
    Horizline(case_state.xpixel-5, case_state.ypixel+30, 10, 0);

    //脚
    bar(case_state.xpixel-11,          case_state.ypixel+59,          case_state.xpixel+11,
```

```
case_state.ypixel+74, 65535);
    linever(case_state.xpixel-12,  case_state.ypixel+60,  case_state.xpixel-12,
case_state.ypixel+75, 1, 0);
    linever(case_state.xpixel+12,  case_state.ypixel+60,  case_state.xpixel+12,
case_state.ypixel+75, 1, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+60, 24, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+70, 24, 0);
    Horizline(case_state.xpixel-15, case_state.ypixel+75, 30, 0);

    if(case_state.catch_th == WITH_BOTTLE)          //拿水杯
    {
        water_bottle(case_state.xpixel+16, case_state.ypixel+31);
    }
    else if(case_state.catch_th == WITH_CLOTHES)    //拿衣服
    {
        clothes(case_state.xpixel+5, case_state.ypixel+31);
    }
    else if(case_state.catch_th == WITH_PLATE)      //端碟子
    {
        plate(case_state.xpixel-10, case_state.ypixel+31);
    }

    //右手
    robot_hand_right(case_state.xpixel-4,case_state.ypixel+62,45+case_state.hand_left);

}

//画出机器人手持物品向左走的图像，传入表示机器人的结构体提供画机器人像的位置
void left_hold(CASE case_state)//向左拿东西的图像
{
    //右手
    robot_hand_left(case_state.xpixel-20,case_state.ypixel+67,-45-case_state.hand_right);

    if(case_state.catch_th == WITH_BOTTLE)          //拿水杯
    {
        water_bottle(case_state.xpixel-25, case_state.ypixel+28);
    }
    else if(case_state.catch_th == WITH_CLOTHES)    //拿衣服
    {
        clothes(case_state.xpixel-20, case_state.ypixel+28);
    }
    else if(case_state.catch_th == WITH_PLATE)      //端碟子
    {
        plate(case_state.xpixel-20, case_state.ypixel+32);
```

```
    }

    //头
    FillCircle(case_state.xpixel, case_state.ypixel+15,12, 65535);
    FillCircle(case_state.xpixel-4, case_state.ypixel+15, 3, 1023); //眼睛

    //身体
    bar(case_state.xpixel-12, case_state.ypixel+20, case_state.xpixel+12,
case_state.ypixel+60, 0);
    bar(case_state.xpixel-11, case_state.ypixel+21, case_state.xpixel+11,
case_state.ypixel+59, 65535);
    Horizline(case_state.xpixel-5, case_state.ypixel+30, 10, 0);

    //脚
    bar(case_state.xpixel-11, case_state.ypixel+59, case_state.xpixel+11,
case_state.ypixel+74, 65535);
    linever(case_state.xpixel-12, case_state.ypixel+60, case_state.xpixel-12,
case_state.ypixel+75, 1, 0);
    linever(case_state.xpixel+12, case_state.ypixel+60, case_state.xpixel+12,
case_state.ypixel+75, 1, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+60, 24, 0);
    Horizline(case_state.xpixel-12, case_state.ypixel+70, 24, 0);
    Horizline(case_state.xpixel-15, case_state.ypixel+75, 30, 0);

    //左手
    robot_hand_left(case_state.xpixel-20,case_state.ypixel+71,-45-case_state.hand_right);

}

//画出机器人手持物品向下走的图像（正面），传入表示机器人的结构体提供画机器人像的
位置
void front_hold(CASE case_state)
{
    //水杯直接贴上去，覆盖在机器人的正面就好了。注意是先画机器人，再画物品，注意
层次感。
    forebodyhead(case_state);
    if(case_state.catch_th == WITH_BOTTLE)
    {
        water_bottle(case_state.xpixel-30, case_state.ypixel+23);
    }
    else if(case_state.catch_th == WITH_CLOTHES)
    {
        clothes(case_state.xpixel-20, case_state.ypixel+27);
    }
```

```
    else if(case_state.catch_th == WITH_PLATE)
    {
        plate(case_state.xpixel-20, case_state.ypixel+29);
    }
}

//画出机器人手持物品向上走的图像（背面），传入表示机器人的结构体提供画机器人像的位置
void back_hold(CASE case_state)
{
    //注意是先画物品，再画机器人，注意层次感。
    if(case_state.catch_th == WITH_BOTTLE)
    {
        water_bottle(case_state.xpixel+20, case_state.ypixel+23);
    }
    else if(case_state.catch_th == WITH_CLOTHES)
    {
        clothes(case_state.xpixel+10, case_state.ypixel+27);
    }
    else if(case_state.catch_th == WITH_PLATE)
    {
        plate(case_state.xpixel+10, case_state.ypixel+27);
    }
    backbodyhead(case_state);
}
```

----------rbtfunc.c----------

```
#include "title.h"
/*********************************************************************************************
  函数列表:    1.void guard(CASE human, CASE robot, int identity);                    //安保功能
               2.void treatment(CASE *human, CASE *robot, Graph G, int n);    //医疗照顾功能
               3.void clean(case *robot);                                              //打扫卫生功能
*********************************************************************************************/
/*函数描述：安保：来人 check，检查是主人还是陌生人。若是主人，上前欢迎问候；若是陌生人，
                则捕捉陌生人图像，通过手机告知主人，并且传送到手机屏幕上，让主人进行识别。
                如果主人点击“报警”, 则自动拨打 110, 并且机器人会喊: “发现入侵者”, 以警告入侵者
                房子里有机器人在防卫，吓跑入侵者；若主人认为闯入的是熟人，即点击“欢迎”，则机器人会来访者。
```

```
  入口参数：分别表示人（闯入者）和机器人的结构体 human、robot；
            identity 表示人身份（就是闯入者），因为里面调用的 paint_man 函数需要该参数
            *x , *y , *button 是鼠标需要的参数
*/
void rbtguard(CASE human, CASE robot, int identity, int *x, int *y, int *button)      //identity 表明来的是主人还是陌生人
{
    int judge=0, j;
    paint_man(human, identity);
    if(identity)              //是主人
    {
        delay0(300);      //腾一点时间出来，做识别时间
        saveimage_g_c(robot.xpixel-105, robot.ypixel-40);       //腾出气泡空间

                bar(robot.xpixel-105, robot.ypixel-40, robot.xpixel-15, robot.ypixel,65535);
        fdhz(robot.xpixel-105, robot.ypixel-40,1,1,"主人，",0);
        fdhz(robot.xpixel-105, robot.ypixel-20,1,1,"欢迎回家",0);
        delay0(1500);  //汉字显示持续一段时间
        //void fdhz(int x,int y,int mx,int my,char *s,int color)
        putsave_g_c(robot.xpixel-105, robot.ypixel-40);      //复原
    }
    else
    {
        delay0(300);      //腾一点时间出来，做识别时间
        FillCircle(robot.xpixel-7.5, robot.ypixel+12, 3, 63776);    //眼睛
        FillCircle(robot.xpixel+7.5, robot.ypixel+12, 3, 63776);//调用画眼睛的语句，眼睛变红

        //机器人会说出来，类似于警告入侵者房子里有机器人在监控
        saveimage_g_c(robot.xpixel-105, robot.ypixel-40);       //腾出气泡空间
        bar(robot.xpixel-105, robot.ypixel-20, robot.xpixel-5, robot.ypixel,65535);
        fdhz(robot.xpixel-105, robot.ypixel-20,1,1,"发现入侵者",0);
        //不着急 put，最后再来

        //手机上提示主人有人入侵
        iph_frame(65534);
        fdhz(ORIGINX+28, ORIGINY+70,2,2,"发现入侵者",0);
        bar(ORIGINX+45, ORIGINY+155, FINALX-45, ORIGINY+250, 65535);            //人像背景

        //手机上画放大的人像
        human.xpixel = MIDDLEX;         //手机上的人
        human.ypixel = ORIGINY+165;
```

```
        man_forebody(human, INTRADER);

        //询问是否报警
        bar(ORIGINX+45, ORIGINY+295, MIDDLEX-10, ORIGINY+355, 63488);    //红色
        fdhz(ORIGINX+65, ORIGINY+315,1,1,"报警",0);
        bar(MIDDLEX+10, ORIGINY+295, FINALX-45, ORIGINY+355, 2016);    //绿色
        fdhz(MIDDLEX+30, ORIGINY+315,1,1,"欢迎",0);

        while(1)
        {
            newxy(x, y, button);

            if(ORIGINX+45<=*x && *x<=MIDDLEX-10 && ORIGINY+295<=*y &&
*y<=ORIGINY+355 && *button)              //点击“报警”，坐标待定
            {
                judge = 0;
                break;
            }
            else if(MIDDLEX+10<=*x && *x<=FINALX-45 && ORIGINY+295<=*y &&
*y<=ORIGINY+355 && *button)       //点击“忽略”
            {
                judge = 1;
                break;
            }
        }

                mousehide(*x,*y);

        if(!judge)          //报警
        {
            iph_frame(65534);

            fdhz(ORIGINX+45, ORIGINY+100,2,2,"正在呼叫",0);
            outtextxy(ORIGINX+55, ORIGINY+140,"110",2,2,50, 0);
            delay(2000);
            human.xpixel = human.x*40;
            human.ypixel = human.y*40;          //前面改变了 pixel 位置，但是 x,y 大坐标
没变，现在变回去，把人覆盖掉
            put_image_man(human.xpixel, human.ypixel);
            putsave_g_c(robot.xpixel-105, robot.ypixel-40);      //气泡
            iph_frame(65534);                       //手机恢复

        }
        else
```

```
        {
            putsave_g_c(robot.xpixel-105, robot.ypixel-40);      //气泡
            saveimage_g_c(robot.xpixel-105, robot.ypixel-40);       //腾出气泡空间
            bar(robot.xpixel-105,  robot.ypixel-40,  robot.xpixel-10,  robot.ypixel,43001);
//苍色气泡
            fdhz(robot.xpixel-105, robot.ypixel-40,1,1,"你好，欢",0);
            fdhz(robot.xpixel-105, robot.ypixel-20,1,1,"迎你的到来",0);

            delay(2000);
            putsave_g_c(robot.xpixel-105, robot.ypixel-40);      //气泡
                    put_image_man(human.xpixel, human.ypixel);
                    bar(3 * 40, 1 * 40, 7 * 40, 3 * 40, 65502); //防止人不消失，把门口那
块画一遍

                    for(j = 1; j < 4; j++)
            {
                w_right(2, j);          //浴室
            }

        }

        FillCircle(robot.xpixel-7.5, robot.ypixel+12, 3, 1023);    //眼睛变回去
        FillCircle(robot.xpixel+7.5, robot.ypixel+12, 3, 1023);

    }

                mousehide(*x,*y);
                iph_page_1();

}
///不需要*choice，main2 里面直接传&choice[4]就好了
/*
函数描述：当检测到主人发烧时，端一杯水，询问是否拨打 120。如果不拨打，则
          机器人把药箱放在床头供主人自取。
入口参数：人和机器人的结构体指针（因为要移动，改坐标，故必须传指针）
          图 G 以及图中元素个数 n，用于寻路，*x，*y，*button 是鼠标用的
          choice 指针是 main2 中传入的 choice 数组，这里修改 choice 用来记录是否去了
医院
*********************************************/
/**void fdhz(int x,int y,int mx,int my,char *s,int color)*/
void treatment(CASE *human, CASE *robot, Graph G, int n, int *x, int *y, int *button, int *choice)
{
    int i, judge=0;
                iph_page_1();
```

```
    aimmove(robot, robot->x, robot->y, 14, 2, G, n, ROBOT);//移动到水杯处
    robot->catch_th = WITH_BOTTLE;    //拿水杯，改变了这个量以后，画图就用 fore_hold 等持水杯的函数了
                water_dispenser(15, 1);              //重画饮水机，原来的上面覆盖有水杯，现在水杯被拿走了，画空的饮水机

    aimmove(robot, robot->x, robot->y, 2, 8, G, n, ROBOT);          //到床头

    water_bottle(3*40+5, 8*40);                      //水杯放在床头的桌上
    robot->catch_th = WITHOUT_THING;              //水杯放下后，回到不持物状态
    saveimage_t_c(robot->xpixel, robot->ypixel-36);      //腾出气泡空间
    bar(robot->xpixel, robot->ypixel-36, robot->xpixel+120, robot->ypixel, 43001); //画气泡，苍色
    fdhz(robot->xpixel, robot->ypixel-36,1,1,"主人，你",0);
    fdhz(robot->xpixel, robot->ypixel-18,1,1,"好像发烧了",0);
    delay0(500);
    bar(robot->xpixel, robot->ypixel-36, robot->xpixel+120, robot->ypixel, 43001); //画气泡，苍色
    fdhz(robot->xpixel, robot->ypixel-36,1,1,"需要拨打",0);
    outtextxy(robot->xpixel+90, robot->ypixel-36,"120",1,1,10,0);
    fdhz(robot->xpixel, robot->ypixel-18,1,1,"吗",0);
    delay(500);
    putsave_t_c(robot->xpixel, robot->ypixel-36);      //复原

    //这里是手机上的东西
    iph_frame(65534);
    bar(ORIGINX+45, ORIGINY+155, FINALX-45, ORIGINY+205, 54938);
    bar(ORIGINX+45, ORIGINY+210, FINALX-45, ORIGINY+250, 54938);
    fdhz(ORIGINX+45, ORIGINY+155,1,1,"拨打",0);
    outtextxy(ORIGINX+98, ORIGINY+155,"120",1,1,10,0);
    fdhz(ORIGINX+45, ORIGINY+210,1,1,"休息一下就好",0);

    while(1)
    {
            newxy(x, y, button);

            if(*x>=ORIGINX+45 && *x<=FINALX-45 && *y>=ORIGINY+155 && *y<=ORIGINY+205 && *button)
            //if(ORIGINX+45<=*x && *x<=FINALX-45 && ORIGINY+155<=*y && *y<=ORIGINY+205 && *button)          //点击“拨打 120”，坐标待定
            {
                judge = 0;
                break;
```

```
            }
            else if(*x>=ORIGINX+45 && *x<=FINALX-45 && *y>=ORIGINY+210 &&
*y<=ORIGINY+250 && *button)
            //else if(ORIGINX+45<=*x && *x<=FINALX-45 && ORIGINY+210<=*y &&
*y<=ORIGINY+250 && *button)       //点击“休息一下就好”
            {
                judge = 1;
                break;
            }
    }

                mousehide(*x,*y);

    if(!judge)                      //鼠标交互，点击“拨打 120”的区域
    {
        iph_frame(65534);
        fdhz(ORIGINX+50, ORIGINY+215,2,2,"正在呼叫",0);
        outtextxy(ORIGINX+50, ORIGINY+255,"120",2,2,50, 0);
        delay0(50);
        bed(0, 7);
        choice[4] = 1;          //打了 120，人去医院，第二天不煮饭
    }
    else
    {
        //药箱放到床头方桌
        medical_kit(3*40+5, 7*40);
        saveimage_t_c(robot->xpixel, robot->ypixel-36);      //腾出气泡空间
        bar(robot->xpixel, robot->ypixel-36, robot->xpixel+120, robot->ypixel, 43001);
//画气泡，苍色
        fdhz(robot->xpixel, robot->ypixel-36,1,1,"那我把医疗",0);
        fdhz(robot->xpixel, robot->ypixel-18,1,1,"包放这啦",0);
        putsave_t_c(robot->xpixel, robot->ypixel-36);      //复原
        choice[4] = 0;          //没打 120，第二天煮饭
    }

    aimmove(robot, robot->x, robot->y, 13, 3, G, n, ROBOT);       //回到原来的位置

                iph_page_1();

}

/********随机生成垃圾，再自动寻路拾取******/
void clean(CASE *robot, CASE *human, Graph G, int n)       //一大堆坐标点没解决
{
```

```
	Axis V1 = {12, 13}, V2 = {12, 13}, V3 = {12, 13}, Vtemp, Vex[3];
	int i, j, a, b, c;    //a， b， c 是用来存随机数，选定垃圾样式的

	//三种垃圾的横纵坐标（大坐标），这样初始化的目的是：确保数组不越界，又保证会进入 while
	//V1，V2,V3 就是要产生垃圾的点，Vex 根据各点与机器人的位置进行排序
	//下面这张图专门给 trash 用，和机器人寻路没关系
	 const static int unaccessible[16][19]={
						//[4][14],[5][4],[9][9]
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},   //第一列
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1},
						{1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
						{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},   //第 16 列
					};

	//这个图和 findway 里面有一点区别，因为要求机器人在 trash 左边捡垃圾，
	//所以现在把这个数组中，本身是 0 但上一行（也就是左边）为 0 的都设为 1，不可达
	get_image_man(human->xpixel, human->ypixel);
	srand(time(NULL));          //随机数种子

	while(unaccessible[V1.x-1][V1.y] && unaccessible[V1.x][V1.y-1])     //前一个条件保证机器人能到达垃圾左边，后一个条件保证垃圾出现的地方没有别的东西
	{
		V1.x = rand()%15+1;
		V1.y = rand()%18+1;
	}

	while(unaccessible[V2.x-1][V2.y] && unaccessible[V2.x][V2.y-1])
	{
		V2.x = rand()%15+1;
```

```
        V2.y = rand()%18+1;
    }

    while(unaccessible[V3.x-1][V3.y] && unaccessible[V3.x][V3.y-1])
    {
        V3.x = rand()%15+1;
        V3.y = rand()%18+1;
    }

    //接下来按距离排序（选择排序法）
    Vex[0] = V1;
    Vex[1] = V2;
    Vex[2] = V3;

    for(i = 0; i < 2; i++)
    {
        for(j = i + 1; j < 3; j++)
        {    //排序准则是 1 范数距离，不是欧式距离
             if(abs(Vex[i].x - robot->x) + abs(Vex[i].y - robot->y) > abs(Vex[j].x - robot->x)
+ abs(Vex[j].y - robot->y)) //1 范数距离，不是欧式距离
             {
                 Vtemp = Vex[i];
                 Vex[i] = Vex[j];
                 Vex[j] = Vtemp;
             }
        }
    }

    //把三种垃圾的背景存起来
    get_image_trash1(Vex[0].x*40, Vex[0].y*40);
    get_image_trash2(Vex[1].x*40, Vex[1].y*40);
    get_image_trash3(Vex[2].x*40, Vex[2].y*40);

    //下面产生随机数，来决定三个产生垃圾的坐标点，各自产生的垃圾的种类
    a = rand()%3+1;

    do{
        b = rand()%3+1;
    }while(b == a);        //确保 a、b 不相等

    do{
        c = rand()%3+1;
    }while(c == a || c == b);    //确保 c 不与 a、b 相等
```

```
    //下面根据随机数的值来确定各位置垃圾的样式
    switch(a)
    {
        case 1:        //若 a 为 1，Vex[0]处画第一种垃圾，即纸张
            trash1(Vex[0].x, Vex[0].y);
            break;
        case 2:        //若 a 为 2，Vex[0]处画第二种垃圾
            trash2(Vex[0].x, Vex[0].y);
            break;
        case 3:        //若 a 为 3，Vex[0]处画第三种垃圾，即纸张
            trash3(Vex[0].x, Vex[0].y);
            break;
    }

    switch(b)
    {
        case 1:        //若 b 为 1，Vex[1]处画第一种垃圾，即纸张
            trash1(Vex[1].x, Vex[1].y);
            break;
        case 2:        //若 b 为 2，Vex[1]处画第二种垃圾
            trash2(Vex[1].x, Vex[1].y);
            break;
        case 3:        //若 b 为 3，Vex[1]处画第三种垃圾
            trash3(Vex[1].x, Vex[1].y);
            break;
    }

    switch(c)
    {
        case 1:        //若 c 为 1，Vex[2]处画第一种垃圾，即纸张
            trash1(Vex[2].x, Vex[2].y);
            break;
        case 2:        //若 c 为 2，Vex[2]处画第二种垃圾
            trash2(Vex[2].x, Vex[2].y);
            break;
        case 3:        //若 c 为 3，Vex[2]处画第三种垃圾
            trash3(Vex[2].x, Vex[2].y);
            break;
    }

    //跑到 trash 左边一格，把 trash 处背景释放
    aimmove(robot, robot->x, robot->y, Vex[0].x-1, Vex[0].y, G, n, ROBOT);        //到垃圾
a 处，Vex[0]
    put_image_trash1(Vex[0].x*40, Vex[0].y*40);                                    //释
```

放背景

aimmove(robot, robot->x, robot->y, Vex[1].x-1, Vex[1].y, G, n, ROBOT); //到垃圾 b 处，Vex[1]
put_image_trash2(Vex[1].x*40, Vex[1].y*40); //释放背景

aimmove(robot, robot->x, robot->y, Vex[2].x-1, Vex[2].y, G, n, ROBOT); //到垃圾 c 处，Vex[2]
put_image_trash3(Vex[2].x*40, Vex[2].y*40); //释放背景

aimmove(robot, robot->x, robot->y, 8, 2, G, n, ROBOT); //扔到垃圾箱

aimmove(robot, robot->x, robot->y, 13, 4, G, n, ROBOT); //回到原位
}

----------rbtmove.c----------

```
#include "title.h"
/*****************************************************************
函数列表：  1.void dmove(CASE *case_state, int casetype)          //改变坐标点
            2.void move0(CASE *case_state, int casetype)          //坐标的改变与动作叠加
            3.int aimmove(CASE *case_state,int x0,int y0,int xt,int yt,const Graph G, const int n, int casetype)      //让机器人移动时需要直接调用的函数
*****************************************************************/
/*函数描述：dmove 是每一次移动的最小单位函数，只改变坐标，然后直接画出这个时候的机器人或者人的图像，并不涉及到手的摆动；
  入口参数：表示机器人或人的状态的结构指针 case_state，因为改变坐标需要址传递
            casetype 用于表示是人还是机器人，MAN，即 1，是人，ROBOT，即 0，是机器人
*/
void dmove(CASE *case_state, int casetype)      //要改变状态，所以应该传指针，casetype 为 0 是机器人，1 是人
{
    int x0 = case_state->xpixel, y0 = case_state->ypixel;

    //判断方向并移动相应距离（以像素点计）
    if(!casetype)    //是机器人
    {
        switch(case_state->direction)
    {
        case 1:             //向右走
```

```
{
    if(case_state->catch_th == WITHOUT_THING)    //手上没拿东西
    {
        robot_right(*case_state);          //画出向右走的图像
    }
    else    //手上有水杯或衣服或盘子
    {
        right_hold(*case_state);
    }
    x0 = x0 + 10;
    break;
}
case 2:             //向左
{
    if(case_state->catch_th == WITHOUT_THING)    //手上没拿东西
    {
        robot_left(*case_state);          //画出向左走的图像
    }
    else    //手上有水杯或衣服或盘子
    {
        left_hold(*case_state);
    }
    x0 = x0 - 10;
    break;
}
case 3:             //向上
{
    if(case_state->catch_th == WITHOUT_THING)    //手上没拿东西
    {
        backbodyhead(*case_state);
    }
    else    //手上有水杯或衣服或盘子
    {
        back_hold(*case_state);
    }
    y0 = y0 - 10;
    break;
}
case 4:             //向下
{
    if(case_state->catch_th == WITHOUT_THING)
    {
        forebodyhead(*case_state);
    }
```

```
                else      //手上有水杯或衣服或盘子
                {
                    front_hold(*case_state);
                }
                y0 = y0 + 10;
                break;
            }
        }

        delay(80); //保持图像
        put_image_robot(case_state->xpixel, case_state->ypixel);

         }
    else          //是人
    {
        switch(case_state->direction)
        {
            case 1:          //向右走
            {
                man_rightbody(*case_state);
                x0 = x0 + 10;
                break;
            }
            case 2:          //向左
            {
                man_leftbody(*case_state);
                x0 = x0 - 10;
                break;
            }
            case 3:          //向上
            {
                man_backbody(*case_state);
                y0 = y0 - 10;
                break;
            }
            case 4:          //向下
            {
                man_forebody(*case_state, 1);    //一定是主人移动，陌生人只是出现并
check，不需要他移动
                y0 = y0 + 10;
                break;
            }
        }
```

```
            delay(80); //保持图像
            put_image_man(case_state->xpixel, case_state->ypixel);

                }

                case_state->xpixel = x0;   //改变位置
                case_state->ypixel = y0;
                case_state->x = (case_state->xpixel + 2)/40;          //根据现在小坐标的情
况调整大坐标
                case_state->y = (case_state->ypixel + 2)/40;          //给两个偏移量是因为
2 个像素点本身并无大碍，为了防止边界处发生潜在的故障

                if(!casetype)          //机器人
    {
        get_image_robot(case_state->xpixel, case_state->ypixel);
    }
    else                        //人
    {
        get_image_man(case_state->xpixel, case_state->ypixel);
    }

}

/*函数描述：将 dmove 中坐标的改变（即位置的移动）与动作的改变融合，相当于运动的
叠加
  入口参数：同 dmove*/
void move0(CASE *case_state, int casetype)
{
                int i;
                if(!casetype)          //机器人
    {
        put_image_robot(case_state->xpixel, case_state->ypixel);
    }
    else
    {
        put_image_man(case_state->xpixel, case_state->ypixel);
    }

                //按一定的间隔改变机器人的位置
                for(i=0;i<4;i++)
                {
                    //改变手的角度
                    if(case_state->hand)
                    {
```

```
                    case_state->hand_left++;
                    case_state->hand_right--;
                }
                else//初始化时为 0，即动右手
                {
                    case_state->hand_right++;
                    case_state->hand_left--;
                }

            if((case_state->hand_right>=10)||(case_state->hand_left>=10)||(case_state->
hand_right<=-10)||(case_state->hand_left<=-10))
                {
                    case_state->hand=!case_state->hand;
                }
                //前面这些部分是在改变动作，相当于做一个走路的图画
                //后面调用 dmove 就是真的在移动了
///可以这样理解, 走路的动画是: “在原地改变动作 (即这段注释上面的部分) +不在意动作,
给位置平移 2 个像素点”的叠加
                //调用最小移动单元
                dmove(case_state,casetype);
            }
            //更新位置
            case_state->x = case_state->xpixel/40;
            case_state->y = case_state->ypixel/40;

            if(!casetype)
    {
        forebodyhead(*case_state);
    }
    else
    {
        man_forebody(*case_state, 1);
    }

}

/*
函数综述:  case_state 的 x、y 坐标会到 move0，然后再进到 demove0，然后才被改变
            demove0 中，根据方向，移动两个像素点，改变 case_state
函数功能:  给定从起始点到终止点的最短路径，人和机器人共用此函数
入口参数:  表示机器人或人的状态的结构指针 case_state，起始点、终止点的 x、y 坐标（大
坐标）
            图 G 以及 G 中元素个数 n，casetype 用于表示是人还是机器人
*/
```

```
int aimmove(CASE *case_state,int x0,int y0,int xt,int yt,const Graph G, const int n, int casetype)//以(0,0)为开始，大像素点
{
        Axis V0, Vt;            //传给 findway

        ///path 传给 findway，然后用入栈 S，再把 S 弹出放回 path
        PathType *path = NULL, *t = NULL;       //t 是入栈出栈时用的暂存变量
        LkStack S;
        int i;

        V0.x = x0, V0.y = y0;               //对 V0、Vt 初始化，因为 findway 需要的是 Axis 型的点
        Vt.x = xt, Vt.y = yt;

        InitStack(&S);                      //初始化栈

    path = (PathType *)malloc(n*sizeof(PathType));      //为 path 分配空间，path 数组元素数目最多只有 G 的个数个

        if(!path)                           //分配失败
        {
            overflow_box(500,500);
            getch();
            exit(1);
        }

        t = (PathType *)malloc(sizeof(PathType));   //为暂存节点分配空间

        if(!t)                              //分配失败
        {
            overflow_box(500,500);
            getch();
            exit(1);
        }

        //如果起点等于终点
        if(x0==xt && y0==yt)
        {
            return 0;           //返回错误信息
        }

        if(!FindWay(G, path, n, V0, Vt))//进入 FindWay 函数，如果找到路径，则得到逆序的最短路径 path，否则得到错误信息并提示寻路出错
        {
```

```
                    //寻路出错
                    FindWay_error(500,500);
                    getch();
                    exit(1);
                }
                else
                {
                    //将 path 中的元素弹出，压入 S
                    for(i = LocateVex(G, n, Vt); i != -1; i = path[i].former)        //从终点开始向
前回溯，i!=-1 说明到达起始点后还要把起始点入栈
                    {
                        Push(&S, path[i]);
                    }

                    //弹出 S 中元素，并单步移动
                    while(S.top != S.bottom)       //栈非空
                    {
                Pop(&S, t);                    //出栈一个点，即得到现在该走向的下一个点
                case_state->direction = t->direction;        //记方向，知道下一个点由上一个
点往什么方向走而到达，等下给 move0 的时候就知道方向，可以做动画，具体坐标不重要
                move0(case_state, casetype);                    //每一步的移动通过 move0 实现
                    }
                }

                //释放堆
                free(t);
    free(path);
    DestroyStack(&S);    //销毁栈

                //将指针置为空
                path = NULL;
                t = NULL;
}
```

----------register.c----------

```
            #include "title.h"
//功能：注册功能主逻辑函数
//输入：用户信息链表的头节点，指向账号字符串的字符指针，指向密码字符串的字符指针，
鼠标坐标指针和按键信息指针
//输出：int 型
//              返回 1：注册成功
//              返回 5：按了返回键
int UserRegist(USERS *head,char *account,char *code,int *x,int *y,int *buttons)
```

```
{
        int judge = 0; //判断应该调用那些函数的变量
        int flag[4] = {0};   //分别为为用户名、密码、二次密码是否正确的判断变量
        int temp;   //用于吸收键盘缓冲区的变量
        char secondcode[11];   //第二次输入密码串
        char mail[20]={0};
        char real_veri[5]={0};
        char veri[5]={0};

        /*必要的初始化过程*/
        secondcode[0] = '\0';
        *account = '\0';
        *code = '\0';
        mousehide(*x,*y);              //画鼠标
    regist_page();                    //登录界面
        reset_mouse(x,y);              //存鼠标样子
        while (1)
        {

       newxy(x, y, buttons);//在新的位置画鼠标
       //账号输入
            judge = input_area(x,y,buttons);

            switch(judge)
            {

                case 0:
                    break;
                //调用账号输入函数
                case 1:
                    //judge = RegisteraccountsInput(account, x, y);
                    judge = inpu(account, x, y, buttons,349, 260,0);
                    if (account[0]==0)
                    {
                        bar(695,240,750,310,50712);          //清空×勾的位置处的图像
                        red_cross(700,275);
                        fdhz(450,242,1,1,"账号不能为空",0);
                        flag[0] = 0;
                    }
                    else if (accounts_2_code(head, account) != NULL)
                    {
                        bar(695,240,750,310,50712);          //清空×勾的位置处的图像
                        red_cross(700,275);
                        fdhz(450,242,1,1,"该账号已被注册",0);
```

```
            flag[0] = 0;
        }
        else
        {
            bar(695,240,750,310,50712);          //清空×勾的位置处的图像
            bar(440,241,690,259,50712);          //清空“该账号已被注册”
            green_tick(700,275);
            flag[0] = 1;

        }
        break;
/*case 2:调用密码输入函数*/
case 2:
        //judge = RegistercodeInput(code, x, y);
        judge = inpu(code, x, y, buttons,349, 350,1);
        if (strlen(code) < 6)
        {
            /*画红色的叉并提示密码不得少于六位&flag2=0*/
            bar(695,330,750,400,50712);          //清空×勾的位置处的图像
            bar(440,331,690,349,50712);          //清空“密码不得少于六位”
            red_cross(700,365);
            fdhz(442,332,1, 1, "密码不得少于六位",   0);
            flag[1] = 0;
        }
        else
        {
            /*画一个绿色的勾&flag2=1*/
            bar(695,330,750,400,50712);          //清空×勾的位置处的图像
            bar(440,331,690,349,50712);          //清空“密码不得少于六位”
            green_tick(700,365);
            flag[1] = 1;
        }
        break;

/*case 3:调用二次密码输入函数*/
case 3:
        //judge = RegistersecondcodeInput(secondcode, x, y);
        judge = inpu(secondcode, x, y, buttons,349, 440,1);
        if (strcmp(code, secondcode) != 0)
        {
            /*画红色的叉并提示两次密码输入不同&&flag3=0*/
            bar(695,420,750,490,50712);          //清空×勾的位置处的图像
            bar(400,421,690,439,50712);          //清空“两次输入的密码不同”
```

```
            red_cross(700,455);
            fdhz(442,422, 1, 1, "两次输入的密码不同", 0);
            flag[2] = 0;
        }
        else
        {
            /*画一个绿色的勾&flag3=1*/
            bar(695,420,750,490,50712);          //清空×勾的位置处的图像
            bar(400,421,690,439,50712);          //清空“两次输入的密码不同”
            green_tick(700,455);

            flag[2] = 1;
        }
        break;

/*case 4:邮箱输入*/
case 4:
        //judge=mail_input(mail,x,y);
        judge = inpu(mail, x, y, buttons,349, 530,0);
        break;

/*case 5:获取验证码*/
case 5:
        flag[3]=get_verification(mail,real_veri,flag);
        judge=4;
        break;


/*case 6:验证码输入*/
case 6:
        //judge=verification_input(veri, x, y);
        judge = inpu(veri, x, y, buttons,349, 620,0);
        break;

/*case 7:完成注册键*/
case 7:
        if (regist_success(real_veri, veri) && flag[1] && flag[2] && flag[0])
        {
            add_new_user(head,account,code);     //计入文件
            bar(695,600,750,670,50712);          //清空×勾的位置处的图像
            bar(400,601,690,619,50712);          //清空“验证码不正确”
            return 1;
        }
        else
```

```
                {
                    bar(695,600,750,670,50712);          //清空×勾的位置处的图像
                    bar(400,601,690,619,50712);          //清空“验证码不正确”

                    red_cross(700,635);
                    fdhz(622,602, 1, 1, "验证码不正确", 0);
                }
                break;
            case 8:
                return 5;
        }

    }
}


//功能：decide which input function will be invoked
//输入：the pointer's information of mouse
//输出：int 型
//          返回 1：进入账号输入函数
//          返回 2：进入密码输入函数
//          返回 3：进入二次确认密码输入函数
//          返回 4：进入邮箱输入函数
//          返回 6：进入输入验证码函数
//          返回 7：点击了完成，进入注册验证函数
//          返回 8：按了“back"
int input_area(int *x,int *y, int *buttons)
{
        if(*x>=344 &&*x<=690&&*y>=260&&*y<=290&&*buttons)
        {
            return 1;
        }
        //密码
        if(*x>=344&&*x<=690&&*y>=350&&*y<=380&&*buttons)
        {
            return 2;
        }
        //二次确认密码
        if(*x>=344&&*x<=690&&*y>=440&&*y<=470&&*buttons)
        {
            return 3;
        }
        //邮箱输入
        if(*x>=344&&*x<=584&&*y>=530&&*y<=560&&*buttons)
```

```
            {
                return 4;
            }
            //获取验证码
            if(*x>=584&&*x<=690&&*y>=530&&*y<=560&&*buttons)
            {
                return 5;
            }
            //输入验证码
            if(*x>=344&&*x<=690&&*y>=620&&*y<=650&&*buttons)
            {
                return 6;
            }
            //注册并登录
            if(*x>=468&&*x<=580&&*y>=710&&*y<=752&&*buttons)
            {
                return 7;
            }
        //返回键
            if(*x>=75&&*x<=125&&*y>=75&&*y<=125&&*buttons)
            {
                return 8;
            }
            else
                return 0;

}




//功能：注册的各个模块的输入
//输入：
//          char*inpu_c：将要输入的字符串的首地址
//          int *x, int *y, int *buttons：鼠标的坐标和按键的指针
//          int x_posi, int y_posi：输入起始位置（第一个字符的左上角坐标）
//          int a_p：判断输入的是否是密码，是密码就显示实心圆，不是就显示该有的字
符
//                  0：不是密码
//                  1：是密码
//输出：int 型
//          返回 1：进入账号输入函数
//          返回 2：进入密码输入函数
//          返回 3：进入二次确认密码输入函数
```

```
//                返回 4：进入邮箱输入函数
//                返回 6：进入输入验证码函数
//                返回 7：点击了完成，进入注册验证函数
//                返回 8：按了“back"
int inpu(char*inpu_c, int *x, int *y, int *buttons,int x_posi, int y_posi,int a_p)   //a_p 是 0 表示非密码
{
        int key;        //表示键值的变量
        int i = 0;     //用于计算已输入的字符数目的变量
        char *p = inpu_c;        //输入字符的中间指针变量
        char ch;        //用于临时储存键值所对应字符的变量
    char temp[2]={0};//用于 outtextxy 的输出
        int judge=0;

        /*使 p 指向'\0',i 表示当前字符数*/
        while (*p != '\0')
        {
            i++;
            p++;
        }

        while (1)
        {
            newxy(x, y, buttons);//在新的位置画鼠标

            judge = input_area(x,y,buttons);

            if (judge!=0)
                return judge;

          key=0;

            if (a_p==0)
            {
                /*吸收键盘缓冲区数据*/
                if (kbhit() != 0)
                {
                    key = bioskey(0);
                }

                /*按 esc 则退出*/
                if(key == 0x11b)
                {
                    exit(0);
```

```
                }
                /*按了回删键*/
                if(key == 0xe08)  //点击退格键时的操作
                {
                    if(p != inpu_c)      //检测是否是首位（地址是否相同）
                    {
                        bar(x_posi+  i  *  10+4,  y_posi,  x_posi+16+  i  *  10+4,
y_posi+30,65535);
                        p--;            //指针          往前移
                        i--;            //记字数 往前移
                    }
                    *p = '\0';          //   同样将最后一位换成'\0'
                }
                //   将按键对应的字符一个个存入 ph_num 数组中

                ch = searchKeyValue(key);
                temp[0] = ch;
                if (ch != '\0'&&i<20)
                {
                    outtextxy(x_posi+i*10+10,y_posi+7,temp,1,1,8,0);

                    //   将字符存入数组中
                    *p = ch;
                    p++;
                    *p = '\0';      // 每次操作都要归'\0'
                    i++;

                }
            }

            else if(a_p==1)
            {
                /*吸收键盘缓冲区数据*/
                if (kbhit() != 0)
                {
                    key = bioskey(0);
                }

                /*按 esc 则退出*/
                if(key == 0x11b)
                {
                    exit(0);
                }
                /*按了回删键*/
```

```
                    if(key == 0xe08)   //点击退格键时的操作
                    {
                        if(p != inpu_c)      //检测是否是首位（地址是否相同）
                        {
                            bar(x_posi + i * 15 -15, y_posi, x_posi+15 + i * 15,
y_posi+30,65535);
                            p--;             //指针          往前移
                            i--;             //记字数 往前移
                        }
                        *p = '\0';           //    同样将最后一位换成'\0'
                    }
                    //    将按键对应的字符一个个存入 ph_num 数组中

                    ch = searchKeyValue(key);
                    temp[0] = ch;
                    if (ch != '\0'&&i<20)
                    {
                        FillCircle(x_posi + i * 15+8, y_posi+15, 5, 0);

                        //    将字符存入数组中
                        *p = ch;
                        p++;
                        *p = '\0';       //  每次操作都要归'\0'
                        i++;

                    }
                }
            }

}

//功能：静态注册界面
//输入：无
//输出：无
void regist_page(void)
{
    CASE robot_position;
    robot_position.xpixel=517;
    robot_position.ypixel=90;
        bar(0,0,1024,768,54938);
        bar(80,70,944,695,50712);
        fdhz(296,11,3,3,"欢",0);
        fdhz(392,11,3,3,"迎",0);
        fdhz(488,11,3,3,"新",0);
```

```
        fdhz(584,11,3,3,"朋",0);
        fdhz(690,11,3,3,"友",0);
        //bar_round(100,100,50,50,3,1,255);
        bar_round(517,275,346,30,5,1,65535);//用户名        344,260     690,290
    bar_round(517,365,346,30,5,1,65535);//密码            344,350     690,380
    bar_round(517,455,346,30,5,1,65535);//再次输入密码   344,440     690,470
    bar_round(517,545,346,30,5,1,65535);//输入邮箱   344,530     690,560
    bar_round(517,635,346,30,5,1,65535);//输入验证码   344,620     690,650
    bar(584,533,690,557,54938);              //获取验证码       584，533     690，557
        //bar_round(517,540,234,50,5,1,33808);
        logo_robot(robot_position);
        fdhz(280,267,1,1,"帐",0);
        fdhz(300,267,1,1,"号",0);
        fdhz(280,357,1,1,"密",0);
        fdhz(300,357,1,1,"码",0);
        fdhz(240,447,1,1,"确",0);
        fdhz(260,447,1,1,"认",0);
        fdhz(280,447,1,1,"密",0);
        fdhz(300,447,1,1,"码",0);
        fdhz(280,537,1,1,"邮",0);
        fdhz(300,537,1,1,"箱",0);
        fdhz(590,537,1,1,"获取验证码",27469);
        fdhz(260,647,1,1,"验",0);
        fdhz(280,647,1,1,"证",0);
        fdhz(300,647,1,1,"码",0);
        fdhz(468,715,2,2,"完",0);
        fdhz(532,715,2,2,"成",0);
        outtextxy(83,95,"back",1,1,10,65535);
}

//功能：获取验证码函数
//输入：指向邮箱字符串的字符指针，指向验证码字符串的字符指针，指针 flag
//输出：int 型
//              返回 1：账号密码信息完整，邮箱正确，生成验证码
//              返回 0：验证码生成错误
int get_verification(char *mail,char *real_veri, int flag[])
{
    char real_mail[17] = "123456aqq.com";
    int a,i;
    char b[5];
    int flag0=0;
    srand((unsigned)time(0));

        if (flag[0]==1 && flag[1]==1   &&   flag[2]==1)
```

```
{
        if (whether_mail(mail)==0)
    {
        bar(390,505,690,530,50712);
        fdhz(454,510,1,1,"邮箱格式不正确",65535);
        bar(695,530,750,580,50712);          //清空×勾的位置处的图像
        red_cross(700,545);
    }
    if (whether_mail(mail)==1)
    {
        if (strcmp(mail,real_mail)==0)
        {
            bar(390,505,690,530,50712);
            bar(695,530,750,580,50712);          //清空×勾的位置处的图像
            green_tick(700,545);
            //get_image(290,0,740,96,image_save);
            bar (290,0,740,50,65535);
            bar_round_2(290,30,740,96,10,1,65535);
            fdhz(300,5,1,1,"尊敬的用户，您好：",0);
            fdhz(332,25,1,1,"欢迎使用本智能家居机器人系统，本次您",0);
            fdhz(332,45,1,1,"注册的验证码为：",0);
            linelevel(380,90,620,90,3,0);
            a=rand()%10000;
            itoa(a,b,5);

            b[4]='\0';

            for(i=0;i<5;i++)
            {
                real_veri[i] = b[i];
            }


            outtextxy(450,50,b,2,2,15,0);

            fdhz(300,75,1,1,"祝您使用愉快",0);
            delay(1500);
            //put_image(290,0,740,96,image_save);
            flag0=1;
        }
    }
}

else
```

```
            {
                bar(390,505,690,530,50712);
                fdhz(454,510,1,1,"账号密码信息不完整",65535);
                bar(695,530,750,580,50712);          //清空×勾的位置处的图像
                red_cross(700,545);
            }
    return flag0;
}

//功能：判断注册是否成功
//输入：真的验证码的字符指针和自己写入的验证码的字符指针
//输出：int 型
//              返回 0：验证码不正确，失败
//              返回 1：验证码正确，成功
int regist_success(char *real_veri, char *veri)
{
    int p=0;
    if (strcmp(real_veri,veri)==0)
    {
        p=1;
    }
    return p;
}

//功能：判断邮箱是否为默认邮箱格式
//输入：指向邮箱字符串的字符指针
//输出：int 型
//              返回 1：格式正确
//              返回 0：格式错误
int whether_mail(char *str)
{
    int p=0;
    if (strstr(str,"aqq.com"))
    {
        p=1;
    }
    return p;
}
```

----------sdzl.c----------

```
#include "title.h"
/*********************************************************************************************
*****************************************
```

函数列表：

1.void sdzl_main(int *x, int *y, int *button, CASE *robot, CASE *man, Graph G, int n, int times, int choice) //选择具体哪种手动指令

2.void water(CASE *robot, int x, int y, Graph G, int n, int times) //给主人送水

**********/

```
/*
函数功能：对鼠标点击进行分析，选择不同种类的手动指令服务（打扫卫生，聊天，默认设置修改）
入口参数：*x，*y，*button 是鼠标操作所需，*robot，*man 分别是代表机器人和人的结构体指针，用于传给函数后续要用的 aimmove
          图 G 和 G 的元素个数 n 也是寻路要用的，times 和 choice 来自 main2，分别表示目前表示的事件，以及人是否去了医院
*/
void sdzl_main(int *x, int *y, int *button, CASE *robot, CASE *man, Graph G, int n, int times, int choice)
{
    int click=0;
    int wht_happen=0;
    iph_page_2();
    while (1)
    {
        newxy(x, y, button);//在新的位置画鼠标

        if (*x>=ORIGINX && *x <=FINALX&& *y>=ORIGINY+54 && *y <=ORIGINY+171&& *button)  //打扫卫生
        {
            click=1;
            wht_happen=0;
            break;
        }
        if (*x>=ORIGINX && *x <=FINALX&& *y>=ORIGINY+171 && *y <=ORIGINY+286&& *button)  //倒水
        {
            click=2;
            wht_happen=0;
            break;
        }
        if (*x>=ORIGINX && *x <=FINALX&& *y>=FINALY-30 && *y <=FINALY&& *button) //返回主页面
        {
            click=3;
            wht_happen=0;
            break;
```

```
            }
        }
        if(wht_happen==0)
        {
            switch (click)
            {
                case 1:
                    if(times == 1 || times == 5)                    //做饭时不能打扫卫生
                    {
                        return;
                    }
                    clean(robot, man, G, n);
                    wht_happen=1;
                    break;

                case 2:
                    water(robot, man->x, man->y, G, n, times, choice);     //送到人处
                    wht_happen=1;
                    break;

                if (click==3)
                {
                    mousehide(*x, *y);
                    return ;
                }
            }
        }
}

/*
函数功能：送水，并且根据不同情况
入口参数：*robot 是代表机器人的结构体指针，用于传给函数后续要用的 aimmove
            图 G 和 G 的元素个数 n 也是寻路要用的，times 和 choice 来自 main2，分别
            表示目前表示的事件，以及人是否去了医院
*/
void water(CASE *robot, int x, int y, Graph G, int n, int times, int choice)      //入口是送水的
终点，times 是事件，来自 time_line
{
    if(times==-1 || times == 0 || times == 1 || times == 5 || choice)                  //人还没回
家，或者做饭时，或者经过 treatment 以后去了医院，都不能送水
    {
        return;
    }
```

```
    aimmove(robot, robot->x, robot->y, 14.5, 2, G, n, ROBOT);//移动到水杯处
    robot->catch_th = WITH_BOTTLE;    //拿水杯，改变了这个量以后，画图就用 fore_hold 等持水杯的函数了
    water_dispenser(15, 1);              //重画饮水机，原来的上面覆盖有水杯，现在水杯被拿走了，画空的饮水机

    //人工作时送水送到电脑桌
    if(x == 2 && y == 15)
    {
        aimmove(robot, robot->x, robot->y, 2, 15, G, n, ROBOT);    //走到电脑桌旁
    }
    else if(x == 2 && y == 10)        //人在睡觉
    {
        aimmove(robot, robot->x, robot->y, 2, 8, G, n, ROBOT);    //走到床头
        water_bottle(3*40+5, 8*40);                    //水杯放在床头的桌上
    }
    else
    {
        aimmove(robot, robot->x, robot->y, x, y, G, n, ROBOT);    //走到人所在的地方
    }

    robot->catch_th = WITHOUT_THING;     //手上没东西了
    aimmove(robot, robot->x, robot->y, 13, 3, G, n, ROBOT);        //回到常待的点
}----------set_c.c----------
        #include "title.h"

//功能：默认设置修改函数
//输入：空调温度和水温的指针
//输出：无
void set_change(int *air_t, int *bath_t)
{
        int x,y,button;          //鼠标的变量
        char at[3]={0};              //空调温度字符串
        char bt[3]={0};              //洗澡水温字符串
        mouseInit(&x, &y,&button); //鼠标初始化
        mousehide(x,y);              //隐藏鼠标
        reset_mouse(&x,&y);      //设置鼠标
        itoa(*air_t,at,10);      //将空调温度转换成字符串
        itoa(*bath_t,bt,10);   //将洗澡水温度转换成字符串
        iph_page_4(at,bt);           //画默认设置修改界面
        while(1)
        {
          newxy(&x,&y,&button);//刷新鼠标
```

```
            if (x>=FINALX-120 && y>=ORIGINY+136 && x<=FINALX-95 && y<=ORIGINY+161 && button) //点击了空调温度减温
            {
                (*air_t)--;
                itoa(*air_t,at,10);
                iph_page_4(at,bt);
            }
            if (x>=FINALX-35 && y>=ORIGINY+136 && x<=FINALX-10 && y<=ORIGINY+161 && button) //点击了空调温度升温
            {
                (*air_t)++;
                itoa(*air_t,at,10);
                iph_page_4(at,bt);
            }
            if(x>=FINALX-120 && y>=ORIGINY+251 && x<=FINALX-95 && y<=ORIGINY+276 && button ) //点击了洗澡水温温度减温
            {
                (*bath_t)--;
                itoa(*bath_t,bt,10);
                iph_page_4(at,bt);
            }
            if(x>=FINALX-35 && y>=ORIGINY+251 && x<=FINALX-10 && y<=ORIGINY+276 && button ) //点击了洗澡水温升温
            {
                (*bath_t)++;
                itoa(*bath_t,bt,10);
                iph_page_4(at,bt);
            }
            if(x>=ORIGINX && y>=FINALY-30 && x<=FINALX && y<=FINALY && button )
                break;
        }
}
```

----------svgahead.c----------

```
#include "title.h"


/*模式 1024*768 64k*/
void SetSVGA64k()
{
    union REGS in;
    in.x.ax = 0x4f02;
    in.x.bx = 0x117;  /*对应的模式*/
```

```
    int86(0x10,&in,&in);
    if(in.x.ax!=0x004f)//若调用成功则返回 0x004f
    {
        printf("Error in setting SVGA64k,0x%x\n",in.x.ax);
        exit(1);
    }
}

/*获得当前 svga 显示模式的信息，返回显示模式号*/
unsigned int GetSVGA()
{
    union REGS out;
    out.x.ax = 0x4f03;
    int86(0x10,&out,&out);
    if(out.x.ax!=0x004f)
    {
        printf("error!: ox%x\n",out.x.ax);
        exit(1);
    }
    return(out.x.bx);
}

/*获取 SVGA 显示模式号 bx。摘录常用的模式号如下:
                    模式号      分辨率      颜色数
                    0x101       640*480     256
                    0x103       800*600     256
                    0x104       1024*768    16
                    0x105       1024*768    256
                    0x110       640*480     32K
                    0x111       640*480     64K
                    0x112       640*480     16.8M
                    0x113       800*600     32K
                    0x114       800*600     64K
                    0x115       800*600     16.8M
                    0x116       1024*768    32K
                    0x117       1024*768    64K
                    0x118       1024*768    16.8M
 ****************************************************/


 /*****************************************************
    功能说明 ： 画点函数
    参数说明: x,y 所要画点位置 color 颜色
*****************************************************/
```

```
    void     putpixel(int x,int y,int color)
 {
   unsigned char far*VideoBuffer=(unsigned char far*)0xa0000000L;
   unsigned long int pos;
   int Newpage=0;
   /*计算显存地址偏移量和对应的页面号，做换页操作 */
   pos=((unsigned long int)y<<10)+x;
   Newpage=pos>>16;
   SelectPage(Newpage);
   VideoBuffer[pos]=color;
   }
/*************************************************************
   功能说明：得到某点的颜色值；
   参数说明：x,y 为该点的坐标；
   返回值：color 为该点的颜色值
*************************************************************/

int   getpixel(int x,int y)
 {
   int color;
   unsigned char far*VideoBuffer=(unsigned char far*)0xa0000000L;
   unsigned long int pos;
   int Newpage=0;
   /*计算显存地址偏移量和对应的页面号，做换页操作 */
   pos=((unsigned long int)y<<10)+x;
   Newpage=pos>>15;
   SelectPage(Newpage);
   color=VideoBuffer[pos];
   return color;
   }


/*************************************************************
功能函数：用 64k 的模式画点
参数说明：画点的位置
返回值说明：  无返回
*************************************************************/
void Putpixel64k(int x, int y,   int color)
{
      if(x>=0&&x<=1024&&y>=0&&y<=768)
      {
      /*显存指针常量，指向显存首地址，指针本身不允许修改*/
      unsigned int far * const video_buffer = (unsigned int far *)0xa0000000L;
```

```
    /*要切换的页面号*/
    unsigned char newpage=0;

//  unsigned char oldpage=0;
    /*对应显存地址偏移量*/
    unsigned long int page;


    /*计算显存地址偏移量和对应的页面号，做换页操作*/
    page = ((unsigned long int)y << 10) + x;
    newpage = page >> 15;    /*32k 个点一换页，除以 32k 的替代算法*/


        SelectPage(newpage);



    /*向显存写入颜色，对应点画出*/
    video_buffer[page] = color;
    }
}


unsigned int Getpixel64k(int x, int y)
 {
    /*显存指针常量，指向显存首地址，指针本身不允许修改*/
    unsigned int far * const video_buffer = (unsigned int far *)0xa0000000L;

    /*要切换的页面号*/
    unsigned char new_page;

    /*对应显存地址偏移量*/
    unsigned long int page;

    /*判断点是否在屏幕范围内，不在就退出*/
    if (x < 0 || x >(1024 - 1) || y < 0 || y >(768 - 1))
    {
        return 0;
    }

    /*计算显存地址偏移量和对应的页面号，做换页操作*/
    page = ((unsigned long int)y << 10) + x;
    new_page = page >> 15;  /*32k 个点一换页，除以 32k 的替代算法*/
    SelectPage(new_page);
```

```
        /*返回颜色*/
        return video_buffer[page];
 }

/*************************************************
                        9 月 26 日编写
function:           get_image

description:        得到一片区域的图像信息

Input:              x0,y0,左上角起始坐标，X1,Y2 右下角坐标，save 指向存储信息的数组

out:                    无输出
*************************************************************/

void       get_image(int x0,int y0,unsigned int   *save)
{

      FILE *fp;
      int i,j;
      unsigned int t;
      int fail_amount=0;
      fp=fopen("save//image","w");
      if(fp==NULL)
      {
            null_box(500,500);
            getch();
            exit(1);
      }
      rewind(fp);
      for(i=0;i<70;i++)
            for(j=0;j<85;j++)
            {
                  t=Getpixel64k(x0+i,y0+j);
                  fwrite(&t,2,1,fp);
            }
      fclose(fp);
}



/*************************************************
                        9 月 26 日编写
```

```
function:            put_image

description:         显示出存储空间的图像信息

Input:               x0,y0,左上角起始坐标，X1,Y2 右下角坐标，save 指向存储信息的数组

out:                 无输出
************************************************************/

void       put_image(int x0,int y0,unsigned int    *save)
{

      int i,j;
      unsigned int t;
      FILE *fp;
      fp=fopen("save//image","r+");
      if(fp==NULL)
      {
            null_box(500,500);
            getch();
            exit(1);
      }
      rewind(fp);
      for(i=0;i<70;i++)
            for(j=0;j<85;j++)
            {
                  fread(&t,2,1,fp);
                  Putpixel64k(x0+i,y0+j,t);
            }
      fclose(fp);


}


void    printf_image_2(int x0, int y0, int x1, int y1, int begin_y)
{
      FILE * fpo;                                    /*定义文件指针*/

      int i = 0;                                          /*循环变量*/
      int j = 0;
      int wide = x1 - x0;
      int high = y1 - y0;
      int dy = begin_y - y0;
```

```
    unsigned     int save = 0;

    char    fas[20]="saveim";
    fpo = fopen(fas, "rb");                          /*建立保存背景的文件*/
    if (fpo == NULL)
    {
        printf("the file cant creat\n");
        exit(1);
    }

    for (i = 0; i<dy; i++)
    {
        for (j = 0; j<wide; j++)
        {
            fread(&save,sizeof(int),1,fpo);
        }
    }

    for (i = 0; i<high - dy; i++)
    {
        for (j = 0; j<wide; j++)
        {


            fread(&save,sizeof(int),1,fpo);
            Putpixel64k(x0 + j, y0 + i, save);

        }
    }

    fclose(fpo);

}

/************************************************
10 月 26 日编写
function:           save_image

description:       显示出存储空间的图像信息

Input:               x0,y0,左上角起始坐标，X1,Y2 右下角坐标,将图像信息存到文件里面

out:
```

```
*************************************************************/
void save_image(int x0, int y0, int x1, int y1)
{
	FILE * fp;                                      /*定义文件指针*/

	int i = 0;                                              /*循环变量*/
	int j = 0;
	int wide = x1 - x0;
	int high = y1 - y0;


	int save = 0;



	char fg[20] = "saveim";
	fp = fopen(fg, "wb");                           /*建立保存背景的文件*/
	if (fp == NULL)
	{
		printf("the file cant creat\n");
		exit(1);
	}


	for (i = 0; i<high; i++)
	{
		for (j = 0; j<wide; j++)
		{

			save = Getpixel64k(x0+j,y0+i);
			fwrite(&save,sizeof(int),1,fp);

		}
	}

	fclose(fp);

}

 /*****************************************************
 功能说明:  显存换页
 参数说明: page ,页面号
 *****************************************************/
unsigned int SelectPage(unsigned char page)
```

```
{
    union REGS r;
    static unsigned char old_page = 0;
    static unsigned char flag = 0;
    r.x.ax = 0x4f05;
    r.x.bx = 0;
    if (flag == 0)
    {
        old_page = page;
        r.x.dx = page;
        int86(0x10, &r, &r);
        flag++;
    }
    if (page != old_page)
    {
        old_page = page;
        r.x.dx = page;
        int86(0x10, &r, &r);
    }

    return 0;
}
```

----------time_line.c----------

```
#include "title.h"
//功能：主人回家函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
void come_home(CASE *robot, CASE *man, int *mx, int *my, int *button, int choice[],VType
G[], int n)
{
    paint_man(*man, MASTER);
                    //画人，人的参数已经在主进程函数中写了
    aimmove(robot,robot->x,robot->y,4,4,G,n,ROBOT);
                        //让机器人到 4，4 去迎接主人


    saveimage_chat(robot->x*40+40,robot->y*40);
                        //聊天框后面的背景的存储
    saveimage_choose(robot->x*40+40,robot->y*40+40*2);
                        //选择框后面的背景的存储

    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);    //聊天白框
```

```
    show_reply("欢迎回家，主人",robot);

    delay(500);

    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
    show_reply("您晚上想吃川菜还是粤菜呢？",robot);

    bar_round_2(robot->x*40+40,robot->y*40+2*40,robot->x*40+40+118,robot->y*40+
3*40,5,1,65535);  //选项白框
    fdhz(robot->x*40+40+40,robot->y*40+2*40+12,1,1,"川菜",0);

    bar_round_2(robot->x*40+40+122,robot->y*40+2*40,robot->x*40+40+240,robot->y
*40+3*40,5,1,65535);  //选项白框
    fdhz(robot->x*40+40+120+40,robot->y*40+2*40+12,1,1,"粤菜",0);

    while (1)
    {
        newxy(mx,my,button);
        if (*mx>=robot->x*40+40 && *mx<=robot->x*40+40+120 &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)  //选择川菜
        {
            choice[1]=1;
            break;
        }
        if (*mx>=robot->x*40+40+120 && *mx<=robot->x*40+40+240 &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)  //选择粤菜
        {
            choice[1]=2;
            break;
        }
    }

    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
    show_reply("好的，我会记住您的选择",robot);

    putsave_chat(robot->x*40+40,robot->y*40);  //恢复聊天框的背后的背景
    putsave_choose(robot->x*40+40,robot->y*40+40*2);  //选择框消失
    delay(300);

    entertain_and_work(robot,man,mx,my,button,choice,G, n);
    //进入工作娱乐函数功能模块
    delay0(200);
```

```
    aimmove(robot,robot->x,robot->y,10,2,G,n,ROBOT);
    saveimage_doing(robot->x*40-40,robot->y*40-35);
        //做事状态框背后的图像的存储

    bar(robot->x*40-40,robot->y*40-35,robot->x*40+40,robot->y*40-5,65535);
//做饭状态白框
    fdhz(robot->x*40-30,robot->y*40-30,1,1,"做饭中",0);
}

//功能：晚餐函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
void dinner(CASE *robot, CASE *man, int *mx, int *my, int *button, int choice[],VType G[], int
n)
{
    int t;                                          //随机变量，用来判断随机输出菜名
    char *s=NULL;                                       //用于指向菜名的字符数组的
字符指针
    srand((unsigned int)time(0));
    t=rand()%4;                                         //给 t 赋随机值

    putsave_doing(robot->x*40-40,robot->y*40-35);   //去掉"做饭中"的状态

    robot->catch_th = WITH_PLATE;                   //机器人的状态改为拿了盘子
    aimmove(robot,robot->x,robot->y,11,6,G,n,ROBOT);//机器人走到桌子旁
    robot->catch_th = WITHOUT_THING;                    //机器人将盘子放到桌上
    plate(12*40,8*40+20);


    //**********这里是还原人工作或休息时家具的状态

    if(choice[0]==1)                //假如人是在休息
    {
        if(choice[2]==1)        //假如人选择了看电视
        {
            TV_off();           //关掉电视机
            delay(200);
        }
        else if(choice[2]==2)   //假如人选择了听音乐
        {
            music_off(8,10);
            music_off(9,10);    //去掉音乐的音符
            delay(200);
        }
```

```
        //**********以下为吃饭的桌子恢复原样，用于吃饭后桌子恢复原样使用
        wood_ver(14, 12);
        wood_ver(14, 13);
        wood_ver(15, 12);
        wood_ver(15, 13);
        wood_ver(14, 14);
        wood_ver(15, 14);
        sofa_main(14, 11);
        man->xpixel=man->x*40;   //人的位置恢复到坐下来之前的位置
        man->ypixel=man->y*40;
        man_forebody(*man,1);      //画人的正面
        //***********
    }
    else if(choice[0]==2)   //假如人去工作
    {
        //******************   以下是工作的桌子恢复原样，用于工作后桌子恢复原样使用
        green_bedroom(0, 14);
        green_bedroom(0, 15);
        green_bedroom(0, 16);
        green_bedroom(1, 14);
        green_bedroom(1, 15);
        green_bedroom(1, 16);
        rect_table(0, 13);
        desk(0, 15);
        seat(0, 16);
        pc(0, 15);
        man->xpixel=man->x*40;
        man->ypixel=man->y*40;
        man_forebody(*man,1);
        //*********************************
    }

    aimmove(man,man->x,man->y,9,10,G,n,MAN);      //人到饭桌旁



    put_image_man(man->xpixel,man->ypixel);       //将人盖住
    man->xpixel=500;
    man->ypixel=315;                              //设置人坐下时的坐标
    sit_1(*man);                                  //画人坐在桌前

    if (choice[1]==1)                             //如果选择的是川菜，则随机输出一个川
菜
```

```
    {
        if(t==0)
        {
            s="麻婆豆腐";
        }
        else if (t==1)
        {
            s="鱼香肉丝";
        }
        else if (t==2)
        {
            s="冒菜";
        }
        else if (t==3)
        {
            s="蚂蚁上树";
        }
    }
    else if (choice[1]==2)                          //如果选择的是川菜，则随机输出一个川
菜
    {
        if(t==0)
        {
            s="蜜汁叉烧";
        }
        else if (t==1)
        {
            s="脆皮烤肉";
        }
        else if (t==2)
        {
            s="红烧乳鸽";
        }
        else if (t==3)
        {
            s="上汤焗龙虾";
        }
    }
    saveimage_chat(robot->x*40+40,robot->y*40);   //聊天框后面的背景的存储

    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);   //聊天白框
    fdhz(robot->x*40+40+10,robot->y*40+20,1,1,"今天是",0);
    fdhz(robot->x*40+40+10,robot->y*40+40,1,1,s,0);
```

```
    fdhz(robot->x*40+40+10,robot->y*40+60,1,1,"请慢用",0);

    getch();                                          //需要按一下键
    putsave_chat(robot->x*40+40,robot->y*40);  //将聊天框背后的背景放出来

    //**********************
    //桌子上有盘子（表示人在吃饭）
    //*************
    saveimage_doing(man->x*40-40,man->y*40-35); //存放人的动作状态框后面的背景

    bar(man->x*40-40,man->y*40-35,man->x*40+40,man->y*40-5,65535);
//吃饭
    fdhz(man->x*40-20,man->y*40-30,1,1,"吃饭中",0);
    getch();

    putsave_doing(man->x*40-40,man->y*40-35);          //动作状态背景 put 出

    entertain_and_work(robot,man,mx,my,button,choice,G, n);   //进入工作娱乐函数功能模
块

}

//功能：娱乐工作函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
void entertain_and_work(CASE *robot, CASE *man, int *mx, int *my, int *button, int
choice[],VType G[], int n)
{

    saveimage_chat(robot->x*40+40,robot->y*40);   //聊天框后面的背景的存储
    saveimage_choose(robot->x*40+40,robot->y*40+40*2);   //选择框后面的背景的存储

    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);    //聊天白框
    show_reply("您现在是选择娱乐还是工作呢？",robot);
    bar_round_2(robot->x*40+40,robot->y*40+2*40,robot->x*40+40+118,robot->y*40+
3*40,5,1,65535);   //选项
    fdhz(robot->x*40+40+40,robot->y*40+2*40+12,1,1,"娱乐",0);

    bar_round_2(robot->x*40+40+122,robot->y*40+2*40,robot->x*40+40+240,robot->y
*40+3*40,5,1,65535);   //选项
    fdhz(robot->x*40+40+120+40,robot->y*40+2*40+12,1,1,"工作",0);

    while (1)
```

```
    {
        newxy(mx,my,button);
        if (*mx>=robot->x*40+40 && *mx<=robot->x*40+40+120 &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)          //选择娱乐
        {
            choice[0]=1;
            break;
        }
        if (*mx>=robot->x*40+40+120 && *mx<=robot->x*40+40+240 &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)     //选择工作
        {
            choice[0]=2;
            break;
        }
    }

    putsave_choose(robot->x*40+40,robot->y*40+40*2);  //选择框消失
    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
    fdhz(robot->x*40+40+20,robot->y*40+20,1,1,"好的",0);

    delay(200);


    if(choice[0]==1)  //娱乐
    {

bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
        show_reply("您要哪种娱乐呢",robot);


bar_round_2(robot->x*40+40,robot->y*40+2*40,robot->x*40+40+118,robot->y*40+3*40,
5,1,65535);  //选项
        fdhz(robot->x*40+40+40,robot->y*40+2*40+12,1,1,"看电视",0);


bar_round_2(robot->x*40+40+122,robot->y*40+2*40,robot->x*40+40+240,robot->y*40+
3*40,5,1,65535);  //选项
        fdhz(robot->x*40+40+120+40,robot->y*40+2*40+12,1,1,"听音乐",0);

        while (1)
        {
            newxy(mx,my,button);
```

```
            if  (*mx>=robot->x*40+40  &&  *mx<=robot->x*40+40+120  &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)  //选择看电视
            {
                choice[2]=1;
                break;
            }
            if  (*mx>=robot->x*40+40+120  &&  *mx<=robot->x*40+40+240  &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)  //选择听音乐
            {
                choice[2]=2;
                break;
            }
        }

        putsave_choose(robot->x*40+40,robot->y*40+40*2);  //选择框消失

bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
        fdhz(robot->x*40+40+20,robot->y*40+20,1,1,"好的",0);

        delay(200);

        putsave_chat(robot->x*40+40,robot->y*40);  //恢复聊天框的背后的背景

        //*********以下为人起身，吃饭的桌子恢复原状

        wood_ver(12, 7);
        wood_ver(12, 8);
        wood_ver(12, 9);
        desk(12, 8);
        seat(12, 9);
        seat(12, 7);
        man->xpixel=man->x*40;
        man->ypixel=man->y*40;
        man_forebody(*man,1);          //吃饭的桌子恢复原状

        //****************
        aimmove(man,man->x,man->y,14,15,G,n,MAN);
        //*************这一行是人坐在沙发上的图像

        put_image_man(man->x*40,man->y*40);
        man->xpixel=580;
        man->ypixel=500;
        sit_2(*man);
```

```
        //*********************

        if(choice[2]==1)       //假如选择了看电视
        {
            TV_on();        //打开电视机

        }
        else if (choice[2]==2) //假如人选择了听音乐
        {

            music_on(8,10);
            music_on(9,10);          //画出两个音符
        }
}
else if (choice[0]==2)               //假如人去工作
{
        delay(200);

        putsave_chat(robot->x*40+40,robot->y*40);   //恢复聊天框的背后的背景

        //************以下为人起身，吃饭的桌子恢复原状
        wood_ver(12, 7);
        wood_ver(12, 8);
        wood_ver(12, 9);
        desk(12, 8);
        seat(12, 9);
        seat(12, 7);
        man->xpixel=man->x*40;
        man->ypixel=man->y*40;
        man_forebody(*man,1);               //吃饭的桌子恢复原状
        //***********************
        aimmove(man,man->x,man->y,2,15,G,n,MAN);      //人前往工作桌前
        //**************以下是人坐在桌前工作的图像

        put_image_man(man->x*40,man->y*40);
        man->xpixel=20;
        man->ypixel=600;
        sit_1(*man);
        //**********************

        aimmove(robot,robot->x,robot->y,14,2,G,n,ROBOT);//机器人去接水

        saveimage_doing(robot->x*40-40,robot->y*40-35); //机器人动作状态框背景存储
```

```
            bar(robot->x*40-40,robot->y*40-35,robot->x*40+40,robot->y*40-5,65535);
//接水
            fdhz(robot->x*40-30,robot->y*40-30,1,1,"接水中",0);
            delay(500);

            putsave_doing(robot->x*40-40,robot->y*40-35);  //动作状态框恢复
            robot->catch_th = WITH_BOTTLE;                  //   这里机器人的手中有水杯

            aimmove(robot,robot->x,robot->y,3,14,G,n,ROBOT);   //这行的目的是送杯水过去
            robot->catch_th = WITHOUT_THING;

            water_bottle(50,605);//把水杯放在桌上
            aimmove(robot,robot->x,robot->y,13,2,G,n,ROBOT);   //回到默认位置处
        }
}

//功能：洗澡函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
void bath(CASE *robot, CASE *man, int *mx, int *my, int *button, int choice[],VType G[], int n)
{
        if (choice[0]==1)   //假如选择淋浴
        {
            aimmove(robot,robot->x,robot->y,13,16,G,n,ROBOT);       //机器人回到默认位置
        }
        if (choice[0]==2)   //假如选择泡澡
        {
            aimmove(robot,robot->x,robot->y,3,15,G,n,ROBOT);       //机器人前往厕所放水

        }

        saveimage_chat(robot->x*40+40,robot->y*40);   //聊天框后面的背景的存储
        saveimage_choose(robot->x*40+40,robot->y*40+40*2);   //选择框后面的背景的存储


        bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);   //聊天白框
        show_reply("您今天是选择淋浴还是泡澡？",robot);


        bar_round_2(robot->x*40+40,robot->y*40+2*40,robot->x*40+40+118,robot->y*40+
3*40,5,1,65535);   //选项
        fdhz(robot->x*40+40+40,robot->y*40+2*40+12,1,1,"淋浴",0);
```

```
    bar_round_2(robot->x*40+40+122,robot->y*40+2*40,robot->x*40+40+240,robot->y
*40+3*40,5,1,65535);  //选项
    fdhz(robot->x*40+40+120+40,robot->y*40+2*40+12,1,1,"泡澡",0);

    while (1)
    {
        newxy(mx,my,button);
        if (*mx>=robot->x*40+40 && *mx<=robot->x*40+40+120 &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)  //选择淋浴
        {
            choice[3]=1;
            break;
        }
        if (*mx>=robot->x*40+40+120 && *mx<=robot->x*40+40+240 &&
*my>=robot->y*40+2*40 && *my<=robot->y*40+3*40 && *button)  //选择泡澡
        {
            choice[3]=2;
            break;
        }
    }

    putsave_choose(robot->x*40+40,robot->y*40+40*2);  //选择框消失
    bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
    fdhz(robot->x*40+40+20,robot->y*40+20,1,1,"好的",0);

    delay(200);

    if(choice[3]==2)
    {

bar_round_2(robot->x*40+40,robot->y*40,robot->x*40+40+40*6,robot->y*40+2*40,
7,2,65535);  //聊天白框
        show_reply("马上为您准备热水",robot);

        delay(1000);
        putsave_chat(robot->x*40+40,robot->y*40);  //恢复聊天框的背后的背景

        aimmove(robot,robot->x,robot->y,1,3,G,n,ROBOT);//机器人到浴缸旁边

        saveimage_doing(robot->x*40-40,robot->y*40+82);
        bar(robot->x*40-40,robot->y*40+82,robot->x*40+40,robot->y*40+112,65535);
//准备热水
        fdhz(robot->x*40-30,robot->y*40+85,1,1,"放水中",0);
```

```
            delay(1000);
            putsave_doing(robot->x*40-40,robot->y*40+82);
        }
        else
        {
            putsave_chat(robot->x*40+40,robot->y*40);   //恢复聊天框的背后的背景
        }
            aimmove(robot,robot->x,robot->y,13,3,G,n,ROBOT);   //回到默认位置处

        //*********以下为人起身，吃饭的桌子恢复原状 或 以下为人起身，工作的桌子恢复原
状
        if(choice[0]==2)     //假如人选择了工作
        {
            green_bedroom(0, 14);
            green_bedroom(0, 15);
            green_bedroom(0, 16);
            green_bedroom(1, 14);
            green_bedroom(1, 15);
            green_bedroom(1, 16);
            rect_table(0, 13);
            desk(0, 15);
            seat(0, 16);
            pc(0, 15);
            man->x=2;
            man->y=15;
            man->xpixel=2*40;
            man->ypixel=15*40;
            man_forebody(*man,1);
        }
        else if(choice[0]==1) //假如人选择了娱乐
        {
            if(choice[2]==1)   //假如人选择了看电视
            {
                TV_off();
                delay(200);
            }
            else if(choice[2]==2)//假如人选择了听音乐
            {
                music_off(8,10);
                music_off(9,10);
                delay(200);
            }
            wood_ver(14, 12);
            wood_ver(14, 13);
```

```
        wood_ver(15, 12);
        wood_ver(15, 13);
        wood_ver(14, 14);
        wood_ver(15, 14);
        sofa_main(14, 11);
        man->x=14;
        man->y=15;
        man->xpixel=14*40;
        man->ypixel=15*40;
        man_forebody(*man,1);
    }
    //*******************
    aimmove(man,man->x,man->y,1,3,G,n,MAN);   //人去洗澡

    put_image_man(man->x*40,man->y*40);
    bar(40,40,120,120,65535);
    fdhz(45,45,1,1,"洗澡中",0);                    //把人盖住

    aimmove(robot,robot->x,robot->y,5,11,G,n,ROBOT);    //机器人去拿衣服
    delay(500);
    delay(500);
    robot->catch_th = WITH_CLOTHES;                          //此时将机器人的状态调整为拿了衣服
    aimmove(robot,robot->x,robot->y,1,3,G,n,ROBOT);      //机器人前往厕所放水
    robot->catch_th = WITHOUT_THING;                         //此时将机器人的状态调整为没拿衣服


    //***********
    //将衣服画到这个方格内

    clothes(2*40,4*40);

    aimmove(robot,robot->x,robot->y,3,11,G,n,ROBOT);   //去开空调
    delay(500);
    //      空调红灯变绿灯
    aircon(3, 7, 1);
    delay(500);
    aimmove(robot,robot->x,robot->y,13,3,G,n,ROBOT);   //回到默认位置处

    //*****重新画出浴缸和地板和人
    glass(1, 1);
    glass(1, 2);
    glass(2, 1);
```

```
    glass(2, 2);
    glass(2, 3);

    man_forebody(*man,1);
    glass(2, 4);
    //*****************************
    delay(500);
    aimmove(man,man->x,man->y,3,10,G,n,MAN);          //人走到床旁边

    man_sleep(*man);                                   //画人睡觉的图
}

//功能：早餐函数
//输入：机器人结构体指针，人结构体指针，鼠标指针，传入选择数组的指针，图
//输出：无
void breakfast(CASE *robot, CASE *man, int *mx, int *my, int *button,VType G[], int n)
{

    aimmove(robot,robot->x,robot->y,10,2,G,n,ROBOT);    //机器人到灶台旁边
    saveimage_doing(robot->x*40-40,robot->y*40+82);
    bar(robot->x*40-40,robot->y*40+82,robot->x*40+40,robot->y*40+112,65535);
//准备早餐
    fdhz(robot->x*40-30,robot->y*40+85,1,1,"做饭中",0);
    getch();
    putsave_doing(robot->x*40-40,robot->y*40+82);

    robot->catch_th = WITH_PLATE;
    aimmove(robot,robot->x,robot->y,11,6,G,n,ROBOT);      // 此时手里拿着盘子
    robot->catch_th = WITHOUT_THING;
    aimmove(robot,robot->x,robot->y,13,3,G,n,ROBOT);       //此时手里没有盘子

    //把盘子放在桌上
    plate(12*40,8*40);

    //重新画一遍床（人起床了），并且在旁边把人画上
    man_getup(*man);


    aimmove(man,man->x,man->y,10,8,G,n,MAN);          //人到饭桌旁


    //画一次地板，把人画没，再画一个有人吃饭的桌子
    put_image_man(man->x*40,man->y*40);
    man->xpixel=500;
```

```
        man->ypixel=370-40;
        sit_1(*man);
        saveimage_doing(man->xpixel-40,man->ypixel+82);
        bar(man->xpixel-40,man->ypixel+82,man->xpixel+40,man->ypixel+112,65535);
//吃饭
        fdhz(man->xpixel-30,man->ypixel+85,1,1,"吃饭中",0);
        getch();
        putsave_doing(man->xpixel-40,man->ypixel+82);

        //把人画出来，再画一个空的桌子
        wood_ver(12, 8);
        wood_ver(12, 9);
        desk(12, 8);
        seat(12, 9);
        man->xpixel=man->x*40;
        man->ypixel=man->y*40;
        man_forebody(*man,1);              //吃饭的桌子恢复原状


        aimmove(robot,robot->x,robot->y,10,2,G,n,ROBOT);
        saveimage_doing(robot->x*40-40,robot->y*40+82);
        bar(robot->x*40-40,robot->y*40+82,robot->x*40+40,robot->y*40+112,65535);
//洗碗
        fdhz(robot->x*40-30,robot->y*40+85,1,1,"洗碗中",0);
        getch();
        putsave_doing(robot->x*40-40,robot->y*40+82);

        aimmove(man,man->x,man->y,1,4,G,n,MAN);   //人到厕所
        put_image_man(man->x*40,man->y*40);         //把人画没
        man_leftbody(*man);                                //人侧着站
        saveimage_doing(man->x*40+40,man->y*40+82);
        bar(man->x*40+40,man->y*40+82,man->x*40+120,man->y*40+112,65535);
//洗碗
        fdhz(man->x*40+50,man->y*40+85,1,1,"洗漱中",0);
        getch();
        putsave_doing(man->x*40+40,man->y*40+82);

        man_forebody(*man,1);
        aimmove(man,man->x,man->y,4,1,G,n,MAN);
        put_image_man(man->x*40,man->y*40);              //把人画没，代表出门

        aimmove(robot,robot->x,robot->y,13,3,G,n,ROBOT);   //回到默认位置处
}
```

----------txt_save.c----------

```
#include "title.h"
//以下两个是机器人聊天框的背景存储函数
//第一个是得到聊天框的背景
//第二个是输出聊天框的背景，及覆盖聊天框
//输入：框的左上角
//输出：无
//尺寸：240*80
void saveimage_chat(int x,int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	fp=fopen("save//welcome","w");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<245;i++)
		for(j=0;j<85;j++)
		{
			t=Getpixel64k(x+i,y+j);
			fwrite(&t,2,1,fp);
		}
	fclose(fp);
}

/******************将 240*80 的弹窗遮挡区域显示*****/

void putsave_chat(int x,int y)
{
	int i,j;
	unsigned int t;
	FILE *fp;
	fp=fopen("save//welcome","r+");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
```

```
    for(i=0;i<245;i++)
        for(j=0;j<85;j++)
        {
            fread(&t,2,1,fp);
            Putpixel64k(x+i,y+j,t);
        }
    fclose(fp);
}

//以下两个是机器人欢迎框的背景存储函数
//第一个是得到聊天框的背景
//第二个是输出聊天框的背景，及覆盖聊天框
//输入：框的左上角
//输出：无
//尺寸：240*80
void saveimage_welcome(int x,int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;                    ///？？？要这个干嘛
    fp=fopen("save//chat","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<250;i++)
        for(j=0;j<83;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
        }
    fclose(fp);
}

/******************将 710*40 的弹窗遮挡区域显示*****/

void putsave_welcome(int x,int y)
{
    int i,j;
    unsigned int t;
```

```
    FILE *fp;
    fp=fopen("save//chat","r+");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<250;i++)
        for(j=0;j<83;j++)
        {
            fread(&t,2,1,fp);
            Putpixel64k(x+i,y+j,t);
        }
    fclose(fp);
}


//以下两个是选择框的背景存储函数
//第一个是得到选择框的背景
//第二个是输出选择框的背景，及覆盖选择框
//输入：框的左上角
//输出：无
//尺寸：240*40
void saveimage_choose(int x,int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//choose","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<250;i++)
        for(j=0;j<43;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
```

```
            }
        fclose(fp);
}

/******************将 240*40 的弹窗遮挡区域显示*****/

void putsave_choose(int x,int y)
{
        int i,j;
        unsigned int t;
        FILE *fp;
        fp=fopen("save//choose","r+");
        if(fp==NULL)
        {
                null_box(500,500);
                getch();
                exit(1);
        }
        rewind(fp);
        for(i=0;i<250;i++)
                for(j=0;j<43;j++)
                {
                        fread(&t,2,1,fp);
                        Putpixel64k(x+i,y+j,t);
                }
        fclose(fp);
}


//以下两个是机器人动作状态框的背景存储函数
//第一个是得到动作状态框的背景
//第二个是输出动作状态框的背景，及覆盖动作状态框
//输入：框的左上角
//输出：无
//尺寸：80*30
void saveimage_doing(int x,int y)
{
        FILE *fp;
        int i,j;
        unsigned int t;
        int fail_amount=0;
        fp=fopen("save//doing","w");
        if(fp==NULL)
        {
```

```
            null_box(500,500);
            getch();
            exit(1);
        }
        rewind(fp);
        for(i=0;i<85;i++)
        {
            for(j=0;j<33;j++)
            {
                t=Getpixel64k(x+i,y+j);
                fwrite(&t,2,1,fp);
            }
        }
        fclose(fp);
}

/******************将 80*30 的弹窗遮挡区域显示*****/

void putsave_doing(int x,int y)
{
        int i,j;
        unsigned int t;
        FILE *fp;
        fp=fopen("save//doing","r+");
        if(fp==NULL)
        {
            null_box(500,500);
            getch();
            exit(1);
        }
        rewind(fp);
        for(i=0;i<85;i++)
        {
            for(j=0;j<33;j++)
            {
                fread(&t,2,1,fp);
                Putpixel64k(x+i,y+j,t);
            }
        }
        fclose(fp);
}

//以下两个是 g_c 框的背景存储函数
//第一个是得到 g_c 框的背景
```

```
//第二个是输出 g_c 框的背景，及覆盖 g_c 框
//输入：框的左上角
//输出：无
//尺寸：90*40
void saveimage_g_c(int x,int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	int fail_amount=0;
	fp=fopen("save//g_c","w");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<105;i++)
		for(j=0;j<43;j++)
		{
			t=Getpixel64k(x+i,y+j);
			fwrite(&t,2,1,fp);
		}
	fclose(fp);
}

/******************将 90*40 的弹窗遮挡区域显示*****/

void putsave_g_c(int x,int y)
{
	int i,j;
	unsigned int t;
	FILE *fp;
	fp=fopen("save//g_c","r+");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<95;i++)
		for(j=0;j<43;j++)
```

```
            {
                fread(&t,2,1,fp);
                Putpixel64k(x+i,y+j,t);
            }
    fclose(fp);
}

//以下两个是机器人 t_c 框的背景存储函数
//第一个是得到 t_c 框的背景
//第二个是输出 t_c 框的背景，及覆盖 t_c 框
//输入：框的左上角
//输出：无
//尺寸：120*40
void saveimage_t_c(int x,int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//t_c","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<125;i++)
        for(j=0;j<43;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
        }
    fclose(fp);
}

/******************将 120*40 的弹窗遮挡区域显示*****/

void putsave_t_c(int x,int y)
{
    int i,j;
    unsigned int t;
    FILE *fp;
    fp=fopen("save//t_c","r+");
```

```
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<125;i++)
        for(j=0;j<43;j++)
        {
            fread(&t,2,1,fp);
            Putpixel64k(x+i,y+j,t);
        }
    fclose(fp);
}

//以下两个是机器人 box 框的背景存储函数
//第一个是得到 box 框的背景
//第二个是输出 box 框的背景，及覆盖 box 框
//输入：框的左上角
//输出：无
//尺寸：200*40
void saveimage_box(int x,int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//box","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<210;i++)
        for(j=0;j<45;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
        }
    fclose(fp);
}
```

```
/******************将 200*40 的弹窗遮挡区域显示*****/

void putsave_box(int x,int y)
{
	int i,j;
	unsigned int t;
	FILE *fp;
	fp=fopen("save//box","r+");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<210;i++)
		for(j=0;j<45;j++)
		{
			fread(&t,2,1,fp);
			Putpixel64k(x+i,y+j,t);
		}
	fclose(fp);
}

//以下两个是人背景存储函数
//第一个是得到人的背景
//第二个是输出人的背景，及覆盖人
//输入：框的左上角
//输出：无
//尺寸：70*90

void get_image_man(int x, int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	int fail_amount=0;
	fp=fopen("save//man","w");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
```

```
	}
	rewind(fp);
	for(i=-35;i<35;i++)
		for(j=-5;j<85;j++)
		{
			t=Getpixel64k(x+i,y+j);
			fwrite(&t,2,1,fp);
		}
	fclose(fp);
}

void put_image_man(int x, int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	int fail_amount=0;
	fp=fopen("save//man","r+");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=-35;i<35;i++)
		for(j=-5;j<85;j++)
		{
			fread(&t,2,1,fp);
			Putpixel64k(x+i,y+j,t);
		}
	fclose(fp);
}

//以下两个是机器人的背景存储函数
//第一个是得到机器人的背景
//第二个是输出机器人的背景，及覆盖机器人
//输入：框的左上角
//输出：无
//尺寸：70*90
void get_image_robot(int x, int y)
{
	FILE *fp;
	int i,j;
```

```
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//robot","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=-35;i<35;i++)
        for(j=-5;j<85;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
        }
    fclose(fp);
}

void put_image_robot(int x, int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    //while(1);
    fp=fopen("save//robot","r+");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=-35;i<35;i++)
        for(j=-5;j<85;j++)
        {
            fread(&t,2,1,fp);
            Putpixel64k(x+i,y+j,t);
        }
    fclose(fp);
}

//以下两个是垃圾 1 的背景存储函数
```

```
//第一个是得到垃圾 1 的背景
//第二个是输出垃圾 1 的背景，及覆盖垃圾 1
//输入：框的左上角
//输出：无
//尺寸：40*40
void get_image_trash1(int x, int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	int fail_amount=0;
	fp=fopen("save//trash1","w");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<43;i++)
		for(j=0;j<43;j++)
		{
			t=Getpixel64k(x+i,y+j);
			fwrite(&t,2,1,fp);
		}
	fclose(fp);
}

void put_image_trash1(int x, int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	int fail_amount=0;
	fp=fopen("save//trash1","r+");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<43;i++)
		for(j=0;j<43;j++)
```

```
            {
                fread(&t,2,1,fp);
                Putpixel64k(x+i,y+j,t);
            }
    fclose(fp);
}

//以下两个是垃圾 2 的背景存储函数
//第一个是得到垃圾 2 的背景
//第二个是输出垃圾 2 背景，及覆盖垃圾 2
//输入：框的左上角
//输出：无
//尺寸：40*40
void get_image_trash2(int x, int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//trash2","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<43;i++)
        for(j=0;j<43;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
        }
    fclose(fp);
}

void put_image_trash2(int x, int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//trash2","r+");
    if(fp==NULL)
```

```
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<43;i++)
        for(j=0;j<43;j++)
        {
            fread(&t,2,1,fp);
            Putpixel64k(x+i,y+j,t);
        }
    fclose(fp);
}

//以下两个是垃圾 3 的背景存储函数
//第一个是得到垃圾 3 的背景
//第二个是输出垃圾 3 的背景，及覆盖垃圾 3
//输入：框的左上角
//输出：无
//尺寸：40*40
void get_image_trash3(int x, int y)
{
    FILE *fp;
    int i,j;
    unsigned int t;
    int fail_amount=0;
    fp=fopen("save//trash3","w");
    if(fp==NULL)
    {
        null_box(500,500);
        getch();
        exit(1);
    }
    rewind(fp);
    for(i=0;i<43;i++)
        for(j=0;j<43;j++)
        {
            t=Getpixel64k(x+i,y+j);
            fwrite(&t,2,1,fp);
        }
    fclose(fp);
}
```

```
void put_image_trash3(int x, int y)
{
	FILE *fp;
	int i,j;
	unsigned int t;
	int fail_amount=0;
	fp=fopen("save//trash3","r+");
	if(fp==NULL)
	{
		null_box(500,500);
		getch();
		exit(1);
	}
	rewind(fp);
	for(i=0;i<43;i++)
		for(j=0;j<43;j++)
		{
			fread(&t,2,1,fp);
			Putpixel64k(x+i,y+j,t);
		}
	fclose(fp);
}
```

----------user_list.c----------

```
		#include "title.h"
/*
users.txt 文件说明:
1.账号的起始符是"\n"，密码的起始符是" ".
2.一行为一个用户的信息，空格前是账号，空格后是密码

链表说明:
1.链表的头是一个指针，指向链表结构体
2.链表的最后一个元素的 next 指向为 null,但是其 account 和 code 不是空的
*/



//功能：创建用户链表
//输入：用户信息链表的头指针
//输出：无
void create_list(USERS *head)
{
		USERS *current = NULL;//指向当前结点的指针变量
	FILE *fp=NULL;			//用于打开 users.txt 的文件指针
```

```
    char ch;       //用于接收并传送文件内部字符的中间变量
    char *p=NULL;     //指向需要接收字符的地址的指针变量

  //初始化
  current = head;
  p = head->account;
  //防止直接点击登陆进入的情况发生
  if ((fp = fopen("user\\users.txt", "r+")) == NULL)
  {
            null_box(500,500);
            getch();
            exit(1);
  }
  while(!feof(fp))//判断指针是否已到达文件尾部
  {
        ch=fgetc(fp);       //从 users.txt 文件中提取一个字符
        if(ch=='\n')       //表示账户串的开始，密码串的结束
         {
        /*向系统申请空间*/
           if((current->next=(USERS *)malloc(sizeof(USERS)))==NULL)
        {
            overflow_box(500,500);
            getch();
            exit(1);
        }

            current=current->next;

            *p='\0';                  //上一个用户的密码最后面加一个 0 表示密码读入完成，完
善密码的字符串
            p=current->account;
         }
        else if(ch==' ')          //表示账户串的结束，密码串的开始
        {
          *p='\0';                  //用户的账号最后面加一个 0 表示账号读入完成，完善账号
的字符串
          p=current->code;
        }
        else              //将对应的账户串或密码串装入链表中
        {
          *p=ch;
          p++;
        }
  }
```

```
    p--;
    *p=0;                //去掉 EOF，并给链表最后一个元素的密码的字符串结尾

    current->next=NULL;
    fclose(fp);
}


//功能：将新用户的信息写入文件
//输入：用户信息链表的头指针，密码字符指针，账号字符指针
//输出：无
void add_new_user(USERS *head,char *s1,char *s2)      /*******创建新用户的链表和将新用户的信息写入文件中       s1 是账号，s2 是密码********/
{
            USERS *current = head;
            FILE *fp;
            char *p;
            int length = strlen(s2);
       int i = 0;

            //以下程序块是为新用户在链表中创建节点
            while(current->next != NULL)
            {
             current = current->next;
            }
            if((current->next = (USERS *)malloc(sizeof(USERS))) == NULL)
            {
             overflow_box(500,500);
             getch();
             exit(1);
            }
            current = current->next;
            strcpy(current->account,s1);
            strcpy(current->code,s2);
            current->next = NULL;

            //以下程序块是将将新用户的信息写入 users.txt 文件中
            if((fp = fopen("user\\users.txt","r+")) == NULL)
            {
             null_box(500,500);
             getch();
             exit(1);
            }
            //将文件内部指针移到文件末端
```

```
        fseek(fp,0L,2);

              p="\n";
              putc(*p,fp);
        p=s1;
        while(*p!='\0')
        {
                    putc(*p,fp);//输入字符到指定文件
                    p++;
        }
        putc(' ',fp);//账户串结束，密码串开始

        p=s2;
        while(*p!='\0')
        {
                    putc(*p,fp);
                    p++;
        }
  fclose(fp);
}

//功能：通过账户来输出对应的密码
//输入：用户信息链表的头指针，账号字符指针
//输出：字符指针
//在已经生成的用户信息链表中查询目标账号的对应密码，并输出该密码字符串的首地址；如果没有该账号，则输出 NULL
char *accounts_2_code(USERS *head,char *string)
{
  USERS *current=head;
  int p=0;                    //判断是否有对应的账号

  while(current->next&&strcmp(current->account,string)!=0)      //在除了最后一个节点的其他节点中寻找是否有对应的账号
  {
     current=current->next;
  }
  if (current->next!=NULL)                //找到了对应的账号，p 置为 1
  {
              p=1;
  }
  else if (current->next==NULL) //如果现在指针指向了最后一个节点
  {
              if (strcmp(current->account,string)==0)   //如果该账号是目标账号
              {
```

```
            p=1;
           }
 }

 if (p==0)                    //如果没有找到对应账号，那么就输出 0
 {
           return NULL;
 }
 else                         //如果找到了对应的账号，那么就输出对应账号对应的密码
 {
           return current->code;
 }
}


//功能：释放链表
//输入：用户信息链表的头指针
//输出：无//释放链表
void free_list(USERS *head)
{
          USERS *previous = head;              //将 previous 指针指向头节点
          USERS *current = head->next;   //将 current 指针指向第一个节点

          if (head == NULL)return 0;         //如果是空表，就返回 0

          while (current->next != NULL)     //如果 current 指针没有指空
          {
           free(previous);                    //释放 previous 指针指向的节点
           previous = current;                //将 previous 指针指向 current 指针
           current = current->next;     //将 current 指向下一个节点
          }
          free(previous);                     //此时 current 指向空，那么就释放 previous 的空
间
          previous=NULL;
          current=NULL;
          head = NULL;
}
```

----------welcome.c----------

```
#include "title.h"
//功能：欢迎静止界面
//输入：无
//输出：无
void outwelcome(void)
```

```
{

    CASE robot_position;                                        //机器人的状态的变量的定义（用于输出机器人 logo）
    int i;                                                      //决定输出的名人名言的随机变量
    int key;                                                    //用于存储键盘输入的信息（判断是否为“r”时有用）
    char *str;                                                  //字符指针，用于指向要输出的句子
    char t[2];                                                  //用来得到输入的键的中间字符串数组
    unsigned int *image_save=(unsigned int *)malloc(708*40*sizeof(unsigned int)); //    保存验证码弹窗背后的图案
    robot_position.xpixel=512;
    robot_position.ypixel=320;                                  //初始化 logo 状态
    linever_color(0,0,1024,768,211,211,211,128,128,128);        //渐变色
    fdhz(250,50,3,3,"家",27469);
    fdhz(318,50,3,3,"居",27469);
    fdhz(386,50,3,3,"机",27469);
    fdhz(454,50,3,3,"器",27469);
    fdhz(522,50,3,3,"人",27469);
    fdhz(590,50,3,3,"模",27469);
    fdhz(658,50,3,3,"拟",27469);
    fdhz(726,50,3,3,"器",27469);
    outtextxy(364,120,"iRobot",4,4,48,0);
    //get_image(192,600,900,640,image_save);
    saveimage_chat(192,600);                                    //保存输出名人名言的地方的背景到文件中
    logo_robot(robot_position);                                 //画 logo
    srand((unsigned int)time(0));
    while(1)                                    //随机名人名言输出
    {
        putsave_chat(192,600);                                  //清空名人名言部分的输出
        i=rand()%5;
        key=0;
        if (i==0)
        {
            str="活着就是为了改变世界，难道还有其他原因吗？";
            fdhz(332,600,1,1,str,0);
            linelevel(650,628,695,628,1,0);
            str="乔布斯";
```

```
            fdhz(702,620,1,1,str,0);
        }
        else if (i==1)
        {
            str="要有勇气追随心声，听从直觉。";
            fdhz(372,600,1,1,str,0);
            linelevel(650,628,695,628,1,0);
            str="乔布斯";
            fdhz(702,620,1,1,str,0);
        }
        else if (i==2)
        {
            str="你若能绕过经验，便会有创新之举。";
            fdhz(352,600,1,1,str,0);
            linelevel(650,628,695,628,1,0);
            str="乔布斯";
            fdhz(702,620,1,1,str,0);
        }
        else if (i==3)
        {
            str="领袖和跟风者的区别就在于创新。";
            fdhz(372,600,1,1,str,0);
            linelevel(650,628,695,628,1,0);
            str="乔布斯";
            fdhz(702,620,1,1,str,0);
        }
        else if (i==4)
        {
            str="佛教中有一句话：初学者的心态；拥有初学者的心态是件了不起的事情。
";
            fdhz(212,600,1,1,str,0);
            linelevel(650,628,695,628,1,0);
            str="乔布斯";
            fdhz(702,620,1,1,str,0);
        }
        gets(t);                                                    //输入完之后还
要点一次回车
        key=t[0];
        if (key=='r')                                               //如果输入“r”，那
么就刷新名人名言
        {
            continue;
        }
        else
```

```
            {
                break;
            }
        }
        free(image_save);
        image_save=NULL;
}


//功能：goodbye 界面
//输入：无
//输出：无
void good_bye(void)
{
        CASE robot_position;
        robot_position.xpixel=512;
        robot_position.ypixel=220;
        linever_color(0,0,1024,768,211,211,211,128,128,128);
        logo_robot(robot_position);
        fdhz(220,460,3,3,"与您度过了美好的一天",27469);
        fdhz(250,520,3,3,"希望下次与您的见面",27469);
        outtextxy(364,120,"iRobot",4,4,48,0);
        getch();
        exit(1);
}

//功能：机器人 logo 输出
//输入：机器人的状态变量
//输出：无
void logo_robot(CASE robot_position)
{
        theta_bar(robot_position.xpixel,robot_position.ypixel+50,(int)(24/sin((double)(45)/180*PI
)),(int)(24/sin((double)(45)/180*PI)),45,27469);
        FillCircle(robot_position.xpixel+16+(int)(32/tan((double)(45)/180*PI))+2,robot_position.y
pixel+80,14,27469);
        theta_bar(robot_position.xpixel-
(int)(24/sin((double)(45)/180*PI)),robot_position.ypixel+50,(int)(24/sin((double)(135)/180*PI)),
(int)(24/sin((double)(135)/180*PI)),225,27469);

        FillCircle(robot_position.xpixel-16+(int)(32/tan((double)(-45)/180*PI))-
2,robot_position.ypixel+80,14,27469);
        //头
        ever_Fillellipse(robot_position.xpixel-15,                    robot_position.ypixel+24,
robot_position.xpixel+15, robot_position.ypixel+24, 18, 27469);      //先画黑再覆盖，相当于
```

有了轮廓

```
    FillCircle(robot_position.xpixel-15, robot_position.ypixel+24, 6, 54938);    //眼睛
    FillCircle(robot_position.xpixel+15, robot_position.ypixel+24, 6, 54938);
    bar(robot_position.xpixel-30, robot_position.ypixel+120, robot_position.xpixel+30, robot_position.ypixel+150, 27469);
    linever(robot_position.xpixel, robot_position.ypixel+120, robot_position.xpixel, robot_position.ypixel+150, 1, 50712);
    //身体
    bar(robot_position.xpixel-32, robot_position.ypixel + 38,robot_position.xpixel+32, robot_position.ypixel + 122, 27469);//边框
    circle(robot_position.xpixel, robot_position.ypixel + 80, 20, 50712);    //不填充
    circle(robot_position.xpixel, robot_position.ypixel + 80, 21, 50712);    //不填充
}
```