程设大作业报告

刘青乐(2024010854) 马子润(2024010846)

〇、 视频链接

视频网盘链接： https://cloud.tsinghua.edu.cn/f/f4ffa7cb48e647fcaf6b/
直接下载链接： https://cloud.tsinghua.edu.cn/f/f4ffa7cb48e647fcaf6b/?dl=1

```
//视频时间轴
00:00-01:02 文件夹内容和结构说明;
01:02-02:24 GUI版本：第一关至第三关成功运行展示;
02:24-04:44 GUI版本：自主创新关卡和extra关卡成功运行展示;
04:44-06:02 GUI版本：Fail、Error情况展示;
06:02-09:04 GUI版本：“外部关卡配置文件”加载关卡的功能展示;
09:04-10:36 GUI版本：机器人指令的“文件输入”功能展示(补录);
10:36-11:50 CLI版本：简单演示（容错）;
11:50-12:56 CLI版本：机器人指令的“文件输入”功能展示(补录);
12:56-13:32 结语;
```

一、 设计思路

我们从最简单的命令行界面CLI版本开始，开发完毕后又基于QT框架开发了GUI版本，也实现了自定义数据配置文件加载自定义关卡的拓展功能。

整体思想：

用命名为$current$的结构体储存机器人的位置、当前积木块数字两个信息；
用两个STL容器$input$和$output$储存输入和输出两个传送带上的数据；
分别用函数实现$inbox$、$outbox$、add、sub、$copyfrom$、$copyto$、$jump$、$jumpifzero$等操作；
用命名为opr的结构体储存每条指令的名称、操作位置数字、是否合法三个信息。
在数据变动时，进行界面输出的更新，（在CLI中为调用$output_level()$函数，在GUI中为调用$op_demonstrate()$函数）。

运行逻辑：

游戏初始进入选关页面，在加载存档关卡后等待用户选择关卡，随后调用关卡函数，加载关卡页面，游戏运行结束后返回选关页面。其中，存档的加载依靠$record.txt$文件中信息的读写。

在关卡页面中，用户输入指令（**键盘输入与文件输入两种方式皆可**），确认后开始执行。执行过程中循环遍历每条输入指令，每次调用指令对应的函数、改变$current$、$input$和$output$等信息，直至运行完毕。运行完毕后，检查最终输出的结果是否和正确答案相同。

二、 工程结构

（一）$CLI\ version$

1. 主函数：

```
int main(){
        game();//基本不做任何事，调用game()函数即可
        system("pause>nul");
        return 0;
}
```

2. $game()$函数：

（1） 从$record.txt$中读取存档。
（2） 初始化变量和stl容器。
（3） 清屏，打印游戏欢迎页面，等待用户输入关卡。
（4） 读入目标关卡数字$levelselect$，判断是否合法。
（5） 若合法，调用$level()$函数。其返回值若为1，则表示成功通过，输出成功信息并更新存档，并重新调用$game()$函数继续游玩；为0表示未成功通过，输出$Fail$并重新调用$game()$函数继续游玩。

//下为具体实现，即按照上述逻辑依次翻译成代码语句，（可以直接太长不看hhh）

```
void game(){
        //读入存档
        FILE *file1 = fopen("record.txt","r");
        fscanf(file1, "%d", &record);
        fclose(file1);
        //初始化变量、队列与栈
        for(int i=0;i<=100;i++){
                rob[i].oprname='\0';
                rob[i].oprpos=-1;
        }
        while(!output.empty()) output.pop();
        while(!input.empty()) input.pop();
        //清屏
        system("cls");
        //打印欢迎界面，并由用户选择关卡
                //中间的代码功能简单，写法臃肿，故省略，功能上相当于下面两行代码
                //printf("Welcome\n");
                //cin >> levelselect;
        //判断选择的关卡是否合法
        if(!(record==4 && levelselect==5) && (levelselect<1 || levelselect>4 || levelselect>record
                gotoxy(0,7);printf("NO ACCESS!!!");Sleep(250);
                game();
        }
        //调用level()函数，根据返回值确定是否通关
        if(level()){
                gotoxy(0,23);//移动光标
                printf("Success");//打印success
                print_info();        //打印具体执行步数
                if(record<levelselect) {        //更新存档
                        FILE *file1 = fopen("record.txt","w");
                        fprintf(file1, "%d", record+1);
                        fclose(file1);
                }
                system("pause>nul");
                game();//自调用，游戏继续运行
        }
        else {
                gotoxy(0,23);//移动光标
                printf("Fail");//打印fail
```

```
                system("pause>nul");
                game();//自调用，游戏继续运行
        }
}
```

3. $level()$函数：

（1） 根据已读入的$levelselect$，按照不同关卡的要求初始化$space[\,]$（空地上的数据）、$input$（输入传送带中初始的数字）。
（2） 清屏，并调用$output_level()$函数输出对应关卡的初始界面。
（3） 读入指令总数$number$和每一条指令$rob[i].name$。
（4） 定义$worker$类型的$current$变量，并初始化。
（5） 循环遍历每条指令，调用相关操作函数，改变$current$、$input$和$output$中的数据。
（6） 执行过程中，当发现指令非法时，输出$error$信息，并重新调用$game()$函数继续游玩。
（7） 执行完毕后，返回$input.empty()\ \&\&\ anscheck()$的逻辑判断值。

//下为具体实现，即按照上述逻辑依次翻译成代码语句，（同样可以太长不看hhh）

```
bool level(){
        //初始化空地
        for (int i = 1;i <= 25;i++)
                space[i] = -inf;
        if(levelselect==4) space[4]=0;//第四关设计带来的特殊设置
        else if(levelselect==5)//外部导入的第五关，独立初始化
                for(int i=1;i<=selfspaceaccess;i++)
                        space[i]=selfspacevalue[i-1];
        //初始化input队列
        if(levelselect==1) input.push(1), input.push(2);
        if(levelselect==2) input.push(3), input.push(9), input.push(5), input.push(1), input.push(
        if(levelselect==3) input.push(6), input.push(2), input.push(7), input.push(7), input.push(
        if(levelselect==4) input.push(7), input.push(4), input.push(7), input.push(3), input.push(
        if(levelselect==5) {//外部导入的第五关，独立初始化
                for(int i=0;i<selfinboxtotal;i++)
                        input.push(selfoffer[i]);
        }
    //初始化cnt、清屏与重新打印
        cnt = 0;
        system("cls");
        output_level(1,-114514,0,input,output,0);
        gotoxy(5*max(4,spaceaccess[levelselect])+2,3); printf("TOTAL NUMBER OF INSTRUCTIONS?");
        gotoxy(5*max(4,spaceaccess[levelselect])+34 + 2,3); printf("(Press 0 to file input)");
        gotoxy(5*max(4,spaceaccess[levelselect])+34,3); cin >> number;
        //规定：若总指令数为0，则为文件读入，if分支为fin处理
        if (number == 0) {
                system("cls");
                printf("Please enter the file address:");
                char t[10086];//t为文件地址
                cin >> t;
                ifstream fin;
            fin.open(t);
            fin >> number;
            string s;
            getline(fin, s);
            for(int i=1;i<=number;i++){
                    getline(fin, s);
                op_input(i, s);
            }
```

```
        fin.close();
    }
    else {//若总指令数不为0，则为键盘输入，else分支为cin处理
            string s;
            getline(cin, s);//多读入一个回车符
        for(int i=1;i<=number;i++){
                gotoxy(5*max(4,spaceaccess[levelselect])+36,i+3);
                getline(cin, s);
            op_input(i, s);
        }
    }
    //定义并初始化current
    worker current;
    current.curnum=-114514;
    current.curpos=1;
    //循环执行每条指令
    for(int i=1;i<=number;i++){
            if (rob[i].illegal) {//非法指令
                    gotoxy(5*max(4,spaceaccess[levelselect])+50,i+3);
                    printf("Error on instruction %d", i);
                    system("pause>nul");
                    game();//自调用，游戏继续运行
            }
            if(instructionaccess[levelselect][0] && rob[i].oprname=="inbox"){
                    if (input.empty()) break;//如果input队列为空，则停止
                    current=inbox(i,current);
            }
            else if(instructionaccess[levelselect][1] && rob[i].oprname=="outbox"){
                    current=outbox(i,current);
            }
            else if (instructionaccess[levelselect][2] && rob[i].oprname == "copyfrom"){
                    current = copyfrom(i, current);
            }
            else if (instructionaccess[levelselect][3] && rob[i].oprname == "copyto") {
                    current = copyto(i, current);
            }
            else if (instructionaccess[levelselect][4] && rob[i].oprname == "add") {
                    current = add(i, current);
            }
```

```
                else if (instructionaccess[levelselect][5] && rob[i].oprname == "sub") {
                        current = sub(i, current);
                }
                else if (instructionaccess[levelselect][6] && rob[i].oprname == "jump") {
                        current = jump(i, current);
                        i = rob[i].oprpos - 1;
                        if (time_to_break()) break;
                }
                else if (instructionaccess[levelselect][7] && rob[i].oprname == "jumpifzero") {
                        current = jumpifzero(i, current);
                        if (current.curnum == 0) i = rob[i].oprpos - 1;
                        if (time_to_break()) break;
                }
                else if (instructionaccess[levelselect][8] && rob[i].oprname == "jumpifnegative")
                        current = jumpifnegative(i, current);
                        if (current.curnum < 0) i = rob[i].oprpos - 1;
                        if (time_to_break()) break;
                }
                else if (instructionaccess[levelselect][9] && rob[i].oprname == "bump+") {
                        current = bump(i, current,'+');
                }
                else if (instructionaccess[levelselect][10] && rob[i].oprname == "bump-") {
                        current = bump(i, current,'-');
                }
                else{//其他情况，非法指令
                        gotoxy(5*max(4,spaceaccess[levelselect])+50,i+3);
                        printf("Error on instruction %d", i);
                        system("pause>nul");
                        game();
                }
        }
        return input.empty() && anscheck();//返回队列为空&&最终结果正确
}
```

4. **操作函数：**

```
//所有函数的声明
worker inbox(int codenum,worker current);
worker outbox(int codenum,worker current);
worker copyfrom(int codenum, worker current);
worker copyto(int codenum, worker current);
worker add(int codenum, worker current);
worker sub(int codenum, worker current);
worker jump(int codenum, worker current);
worker jumpifzero(int codenum, worker current);
worker jumpifnegative(int codenum, worker current);
worker bump(int codenum, worker current, char whattodo);
bool time_to_break();
bool anscheck();
bool selfanscheck(int total, int* array);
void game();
bool level();
void op_input(int codenum);
```

（1） $inbox$，$outbox$，$copyfrom$，$copyto$，add，sub，$jump$，$jumpifzero$，$jumpifnegative$，$bump$函数执行相关指令的操作。在执行过程中，会修改小机器人的$oprnum$、$oprpos$等信息，同时更新输出的图形界面。

（2） $time_to_break()$负责检查$jump()$何时应该停止；$anscheck()$负责检查最终结果是否和正确答案相同;$op_input()$负责输入操作指令，判断这些指令是否合法；$selfanscheck()$和自定义关卡有关，负责检查最终输出传送带上的结果是否与自定义的正确答案相同。

（3） $op_input(int\ codenum)$函数是字符串处理函数，将用户的输入指令转化为rob数组中的$oprname$、$oprpos$等信息，具体实现如下：

```
void op_input(int codenum, string s)
{
    for (int i = 0; i < s.length();i++) {//按位处理
        if (s[i] == ' ') {//查询到空格，说明指令结束
            rob[codenum].oprname = s.substr(0, i);
            //inbox和outbox不应该有空格，否则非法
                        if (rob[codenum].oprname == "inbox" || rob[codenum].oprname == "outbox") {
                rob[codenum].illegal = 1;
                return ;
            }
                        //继续读空格后面的数字，作为指令的参数，下为字符转数字的操作
            int op_num = 0;
            for (int j = i + 1;j < s.length();j++) {
                if (s[j] >= '0' && s[j] <= '9') {
                    op_num = op_num * 10 + (s[j] - '0');
                }
                else {
                    rob[codenum].illegal = 1;
                    return ;
                }
            }
            rob[codenum].oprpos = op_num;
            return ;
        }
    }
    if (s == "inbox" || s == "outbox") {//特判inbox和outbox
        rob[codenum].oprname = s;
        return ;
    }
    else {//其他情况，非法指令
        rob[codenum].illegal = 1;
        return ;
    }
    return ;
}
```

5. 界面输出函数：

（1） $output_level()$负责输出（更新）关卡中的图形界面，包括机器人、传送带、空地、指令等。

//这部分代码极为dirtywork，毫无技术力可言，反复移动光标并输出罢了，可以直接跳过这部分的阅读

```
void output_level(int pos, int robcurrent,int codecurrent, queue <int> input, stack <int> output,b
        //打印关卡信息
        gotoxy(0,0);
        if(levelselect==1){
                printf("Level information:Level 1 --- First Things First\n");
                printf("Output everything in order.\n");
                printf("Available Space:0        Accepted Command:inbox outbox\n");
        }
        else if(levelselect==2){
                printf("Level information:Level 2 --- Do Some Subtractions\n");
                printf("For every two numbers A and B, output A-B and B-A in order.\n");
                printf("Available Space:3        Accepted Command:inbox outbox copoyfrom copyto ad
        }
        else if(levelselect==3){
                printf("Level information:Level 3 --- We Are Homo!\n");
                printf("Get two numbers. If they are the same, output one of them. Otherwise, thro
                printf("Available Space:3        Accepted Command:inbox outbox copoyfrom copyto ad
        }
        else if(levelselect==4){
                printf("Level information:Level 4 --- Let's try division!\n");
                printf("Get two numbers A and B. Output A/B omitting the remainder.\n");
                printf("Available Space:4        Accepted Command:inbox outbox copoyfrom copyto ad
        }
        else if(levelselect==5)
                selfprintinfo();

        //打印inbox传送带
        gotoxy(0,4); printf("IN");
        for(int i=1;i<=6;i++){
                gotoxy(3,3*i+1);        printf("+---+\n");
                gotoxy(3,3*i+2);
                if(input.empty())
                        printf("| X |\n");
                else{
                        printf("| %d |\n",input.front());
                        input.pop();
                }
                gotoxy(3,3*i+3);        printf("+---+\n");
        }
```

```
//打印outbox传送带
gotoxy(5*max(4,spaceaccess[levelselect])+26,4); printf("OUT");
for(int i=1;i<=6;i++){
        gotoxy(5*max(4,spaceaccess[levelselect])+20,3*i+1);        printf("+---+\n");
        gotoxy(5*max(4,spaceaccess[levelselect])+20,3*i+2);
        if(output.empty())
                printf("| X |\n");
        else{
                printf("| %d |\n",output.top());
                output.pop();
        }
        gotoxy(5*max(4,spaceaccess[levelselect])+20,3*i+3);        printf("+---+\n");
}

//根据小机器人位置和所持的数字，打印小机器人
gotoxy(5*pos+9,4);  printf("+---+\n");
gotoxy(5*pos+9,5);
        if(robcurrent==-114514)
                    printf("|   |\n",robcurrent);
        else
                    printf("| %d |\n",robcurrent);
gotoxy(5*pos+9,6);  printf("+---+\n");
gotoxy(5*pos+9,7);  printf("@   @\n");
gotoxy(5*pos+9,8);  printf("-----\n");
gotoxy(5*pos+9,9);  printf("|@ @|\n");
gotoxy(5*pos+9,10); printf("  +  \n");
gotoxy(5*pos+9,11); printf("/   \\\n");
gotoxy(5*pos+9,12); printf(" | | \n");

//打印草地
for(int i=1;i<=max(4,spaceaccess[levelselect]);i++){
        gotoxy(5*i+9,14);printf("+---+\n");
        gotoxy(5*i+9,15);
                if (space[i] == -inf) printf("| X |\n");
                else printf("| %d |\n", space[i]);
        gotoxy(5*i+9,16);printf("+---+\n");
        gotoxy(5*i+11,17);printf("%d\n",i-1);
}
```

```
        //打印代码栏左边的竖线、代码序号和相应的指示箭头“>”
        for(int i=1 ; i<=18; i++){
                gotoxy(5*max(4,spaceaccess[levelselect])+30,i+3);printf("| ");
                if(i==codecurrent)        printf(">%2d ",i);
                else printf(" %2d ",i);//50,55
        }

        //当关卡已经开始运行后，每一次需要重新打印用户输入的指令行
        if(situa==1)
                for(int i=1;i<=number;i++){
                        gotoxy(5*max(4,spaceaccess[levelselect])+36,i+3);//56 61
                        if(rob[i].oprpos==-1)
                                printf("%s",rob[i].oprname.c_str());
                        else
                                printf("%s %d",rob[i].oprname.c_str(),rob[i].oprpos);
                }
        return;
}
```

（2） $print_info()$负责在$Success$时同时输出执行的指令总数。

```
void print_info(){
        gotoxy(0,24);
    printf("The total number of orders executed is %d", cnt);
        return ;
}
```

（3） $gotoxy()$函数负责在命令行中移动光标，控制输出的位置。

```
void gotoxy(int x, int y) {
    COORD pos = {x,y};
    HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(hOut, pos);
}
```

（二）*GUI version*

我们使用了QT框架来实现图形化界面。

$main.cpp$文件是主程序，包含了$widget.h$头文件和QT相关头文件。

$widget.h$头文件中定义了一个$widget$类，这个类继承了$QWidget$类。$widget$类中声明了QT相关函数和$HumanResourceMachine$程序的相关函数。其中最重要的是在$private$部分声明了所有需要用到的信号和槽函数。

```
private slots:
    void on_BtnQuit_clicked();
    void on_button1_clicked();
    void on_button2_clicked();
    void on_button3_clicked();
    void on_button4_clicked();
    void on_buttonextra_clicked();
    void on_codeline1_returnPressed();
    //省略16行，中间是其它的void on_codelineXX_returnPressed();
    void on_codeline18_returnPressed();
    void on_button_retry_clicked();
    void on_button_execute_clicked();
    void on_button_file_clicked();

private:
    Ui::Widget *ui;
    void op_input(QString s, int codenum);
    void op_demonstrate(int robcurrent,int codecurrent, QQueue <int> input, QStack <int> output);
    void op_clear();
    void op_errormessage(int codenum);
    void op_successmessage();
    void op_failmessage();
    void inbox(int codenum);
    void outbox(int codenum);
    void copyfrom(int codenum);
    void copyto(int codenum);
    void add(int codenum);
    void sub(int codenum);
    void jump(int codenum);
    void jumpifzero(int codenum);
    void jumpifnegative(int codenum);
    void bump(int codenum, char whattodo);
    bool time_to_break();
    bool anscheck();
    bool selfanscheck(int total, int* array);
    void move_robot(int destination);
    void delay(int milliseconds);
```

*widget.cpp*文件是*widget*类的实现文件，具体定义了在*widget.h*头文件中声明的函数，

也是工作的核心，下面是对这部分代码的详细解释。

1. **核心交互设计：**

使用一个$stackedWidget$存放两个页面，在选关页面中，用户点击$L1$、$L2$、$L3$、$L4$、$EXTRA$按钮可以分别进入对应关卡（即$stackedWidget$翻到下一页，点击$Quit$按钮可以结束游戏。在关卡执行页面，点击$Retry$按钮可以回到选关页面（即$stackedWidget$翻回上一页），在右侧每一个文本框中可以输入指令，按回车后该文本框设为禁用，代码总数自动加1，点击$Execute$按钮可以顺序执行所有指令，此外，我们还设计了$FILEIN$按钮，按下它会弹出资源管理器，提示用户选择“机器人指令文件”，输入完毕后会自动呈现在右侧代码区，同时所有$QLineEdit$都被设为$unable$，用户继续点击$Execute$按钮可以顺序执行所有指令。

分别以第一关按钮和第一行文本框为例，展示对应的槽函数设计。

```
void Widget::on_button1_clicked(){//这是第一关按钮的槽函数
    levelselect = 1;

    op_clear();

    ui->label_info->setText("Level 1 --- First Things First");
    ui->label_require->setText("Output everything in order.");
    ui->label_avspace->setText("Available Space:0");
    ui->label_avcommand->setText("inbox outbox");

    input.enqueue(1);
    input.enqueue(2);
    op_demonstrate(-114514,0,input,output);
    ui->stackedWidget->setCurrentIndex(1);

}
void Widget::on_codeline1_returnPressed(){//这是第一行文本框的槽函数
    QString code = ui->codeline1->text();
    ui->codeline1->setEnabled(false);
    op_input(code, 1);
    number += 1;
}
```

此外，游戏运行的核心槽函数是$Widget::on_button_execute_clicked()$，这一槽函数相当于$CLI$版本中的$level()$函数，初始化$current$并逐行运行指令，过程中判断指令的合法性，结束后弹出$Success$或$Fail$消息框，点击$Retry$按钮可以回到选关页面。由于运行逻辑和

CLI版本中的$level()$函数基本一致，故下面出示的代码可以略过。

```
void Widget::on_button_execute_clicked()
{
    //对current进行初始化
    current.curnum=-114514;
    current.curpos=1;
    //逐行执行指令
    for(int i=1;i<=number;i++)
    {
        if (rob[i].illegal){//非法情况
            op_demonstrate(current.curnum, i, input, output);//需要演示一次当前状态
            op_errormessage(i);//弹出error的消息框
            return;
        }
        if(instructionaccess[levelselect][0] && rob[i].oprname=="inbox"){
            if (input.empty()) break;//如果inbox为空，跳出循环
            inbox(i);
        }
        else if(instructionaccess[levelselect][1] && rob[i].oprname=="outbox"){
            if(current.curnum==-114514){//不合法指令，报错
                op_demonstrate(current.curnum, i, input, output);
                op_errormessage(i);
                return;
            }
            outbox(i);
        }
        else if (instructionaccess[levelselect][2] && rob[i].oprname == "copyfrom"){
            int pos = rob[i].oprpos + 1;
            if (space[pos] == -inf || pos < 1 || pos > spaceaccess[levelselect]){//不合法指令，报错
                op_demonstrate(current.curnum, i, input, output);
                op_errormessage(i);
                return;
            }
            copyfrom(i);
        }
        else if (instructionaccess[levelselect][3] && rob[i].oprname == "copyto") {
            int pos = rob[i].oprpos + 1;
            if (current.curnum == -114514 || pos < 1 || pos > spaceaccess[levelselect]){//不合法指
                op_demonstrate(current.curnum, i, input, output);
                op_errormessage(i);
```

```
            return;
        }
        copyto(i);
    }
    else if (instructionaccess[levelselect][4] && rob[i].oprname == "add") {
        int pos = rob[i].oprpos + 1;
        if (current.curnum == -114514 || pos < 1 || pos > spaceaccess[levelselect] || space[po
            op_demonstrate(current.curnum, i, input, output);
            op_errormessage(i);
            return;
        }
        add(i);
    }
    else if (instructionaccess[levelselect][5] && rob[i].oprname == "sub") {
        int pos = rob[i].oprpos + 1;
        if (current.curnum == -114514 || pos < 1 || pos > spaceaccess[levelselect] || space[po
            op_demonstrate(current.curnum, i, input, output);
            op_errormessage(i);
            return;
        }
        sub(i);
    }
    else if (instructionaccess[levelselect][6] && rob[i].oprname == "jump") {
        int jumpto = rob[i].oprpos;
            if (jumpto < 1 || jumpto > number){//不合法指令，报错
                op_demonstrate(current.curnum, i, input, output);
                op_errormessage(i);
                return;
            }
        jump(i);
        i = rob[i].oprpos - 1;
        if (time_to_break()) break;
    }
    else if (instructionaccess[levelselect][7] && rob[i].oprname == "jumpifzero") {
        int jumpto = rob[i].oprpos;
            if (jumpto < 1 || jumpto > number || current.curnum == -114514){//不合法指令，报错
                op_demonstrate(current.curnum, i, input, output);
                op_errormessage(i);
                return;
```

```
            }
        jumpifzero(i);
        if (current.curnum == 0) i = rob[i].oprpos - 1;
        if (time_to_break()) break;
    }
    else if (instructionaccess[levelselect][8] && rob[i].oprname == "jumpifnegative") {
        int jumpto = rob[i].oprpos;
            if (jumpto < 1 || jumpto > number || current.curnum == -114514){//不合法指令，报错
                op_demonstrate(current.curnum, i, input, output);
                op_errormessage(i);
                return;
            }
        jumpifnegative(i);
        if (current.curnum < 0) i = rob[i].oprpos - 1;
        if (time_to_break()) break;
    }
    else if (instructionaccess[levelselect][9] && rob[i].oprname == "bump+") {
        int pos = rob[i].oprpos + 1;
        if (space[pos] == -inf || pos < 1 || pos > spaceaccess[levelselect]){//不合法指令，报错
            op_demonstrate(current.curnum, i, input, output);
            op_errormessage(i);
            return;
        }
        bump(i, '+');
    }
    else if (instructionaccess[levelselect][10] && rob[i].oprname == "bump-") {
        int pos = rob[i].oprpos + 1;
        if (space[pos] == -inf || pos < 1 || pos > spaceaccess[levelselect]){//不合法指令，报错
            op_demonstrate(current.curnum, i, input, output);
            op_errormessage(i);
            return;
        }
        bump(i, '-');
    }
    else{//其他不合法指令，报错
        op_demonstrate(current.curnum, i, input, output);
        op_errormessage(i);
        return;
    }
```

```
    }
    if (input.empty() && anscheck()){//成功过关，输出通关信息
        op_successmessage();//弹出success消息框
        if(record<levelselect) {//存档更新
            FILE *file1 = fopen("C:\\FinalWork_by_LQL&MZR\\code\\record.txt","w");
            fprintf(file1, "%d", record+1);
            fclose(file1);
                }
    }
    else{
        op_failmessage();//弹出success消息框
    }
}
```

这里，值得一提的是$Widget::on_button_file_clicked()$槽函数，它实现了机器人指令的文件读入，实现方法非常有趣：

```
void Widget::on_button_file_clicked()
{
    ui->codeline1->setEnabled(false);
    ui->codeline2->setEnabled(false);
    ui->codeline3->setEnabled(false);
    ui->codeline4->setEnabled(false);
    ui->codeline5->setEnabled(false);
    ui->codeline6->setEnabled(false);
    ui->codeline7->setEnabled(false);
    ui->codeline8->setEnabled(false);
    ui->codeline9->setEnabled(false);
    ui->codeline10->setEnabled(false);
    ui->codeline11->setEnabled(false);
    ui->codeline12->setEnabled(false);
    ui->codeline13->setEnabled(false);
    ui->codeline14->setEnabled(false);
    ui->codeline15->setEnabled(false);
    ui->codeline16->setEnabled(false);
    ui->codeline17->setEnabled(false);
    ui->codeline18->setEnabled(false);

    QString file_path = QFileDialog::getOpenFileName(this,tr("璇烽€夋嫨鏂囦欢"),"C:\\FinalWork_by_L
    QFile FileToInput(file_path);
    FileToInput.open(QIODevice::ReadOnly | QIODevice::Text);
    QString data;

    QTextStream fin(&FileToInput);

    fin >> number;
    QString nullstring = fin.readLine();
    QString codelines;

    for(int i = 1; i <= number; i++){
        codelines = fin.readLine();
        op_input(codelines, i);
        QString codelinename = "codeline";

        QString singlenumber = QString::number(i);
        QString codelinenameaddnumber = codelinename + singlenumber;
```

```
        QLineEdit *db = findChild<QLineEdit *>(codelinenameaddnumber);
        db->setText(codelines);
    }

    FileToInput.close();
}
```

2. 函数功能说明

op_clear函数：将各统计用变量、队列和栈中的数据、$inbox$和$outbox$传送带中的数字、空地中的数字、机器人的位置及所持数字、指令栏、所有按钮和文本行的$enablity$全部重置。在选关页点击某个按钮进入关卡后，需要执行该函数，将所有东西初始化。（操作非常基础，代码不必展示）

op_input函数：字符串处理，将用户输入的指令转化为$QString$类型。这部分字符串的处理和CLI版本完全一致，只是从$string$类型变成了$QString$类型，不再展示。

$op_demonstrate$函数：相当于原CLI版本中的$output_level$函数，在数据发生变化时调用$op_demonstrate$函数，将变化后的图形界面更新输出到界面上。同样非常dirtywork，下面代码可以直接跳过阅读。

```
void Widget::op_demonstrate(int robcurrent,int codecurrent, QQueue <int> tmpinput, QStack <int> tm
    //inbox strip
    QString inbox = "inline";
    for(int i=1;i<=6;i++){
        QString singlenumber = QString::number(i);
        QString inboxaddnumber = inbox + singlenumber;
        QLineEdit *db = findChild<QLineEdit *>(inboxaddnumber);
        if(tmpinput.empty())
            db->setText("X");
        else{
            QString inlineexactnumber = QString::number(tmpinput.first());
            db->setText(inlineexactnumber);
            tmpinput.dequeue();
        }
    }
    //outbox strip
    QString outbox = "outline";
    for(int i=1;i<=6;i++){
        QString singlenumber = QString::number(i);
        QString outboxaddnumber = outbox + singlenumber;
        QLineEdit *db = findChild<QLineEdit *>(outboxaddnumber);
        if(tmpoutput.empty())
            db->setText("X");
        else{
            QString inlineexactnumber = QString::number(tmpoutput.top());
            db->setText(inlineexactnumber);
            tmpoutput.pop();
        }
    }
    //robot's number
    if(robcurrent==-114514)
        ui->currentline->setText("");
    else{
        QString singlenumber = QString::number(robcurrent);
        ui->currentline->setText(singlenumber);
    }

    //code's yajirushi
    QString yajirushi = "yajirushi";
```

```
    for(int i = 1; i <= 18; i++){
        QString singlenumber = QString::number(i);
        QString yajirushiaddnumber = yajirushi + singlenumber;
        QLabel *db = findChild<QLabel *>(yajirushiaddnumber);
        if(i == codecurrent)
            db->setText(">");
        else
            db->setText("");
    }

    //grass
    QString spacename = "spaceline";
    for(int i = 0; i < spaceaccess[levelselect]; i++){
        QString singlenumber = QString::number(i);
        QString spacenameaddnumber = spacename + singlenumber;
        QLineEdit *db = findChild<QLineEdit *>(spacenameaddnumber);
        if(space[i+1] == -inf)
            db->setText(" ");
        else{
            QString spaceexactnumber = QString::number(space[i+1]);
            db->setText(spaceexactnumber);
        }
    }
    for(int i = spaceaccess[levelselect]; i < 10; i++){
        QString singlenumber = QString::number(i);
        QString spacenameaddnumber = spacename + singlenumber;
        QLineEdit *db = findChild<QLineEdit *>(spacenameaddnumber);
        db->setText("");
    }
}
```

$op\_errormessage$、$op_successmessage$和$op_failmessage$三个函数：均用来弹出消息框，提示玩家游戏状态用的。下面以$error$的提示框为例，说明消息框函数的实现。$success$和$fail$的消息框均如法炮制。

```
void Widget::op_errormessage(int codenum){
    QMessageBox *error = new QMessageBox();
    error->setIcon(QMessageBox::Critical);
    error->setWindowTitle("Error");
    QPushButton *confirmbutton = new QPushButton();
    QString getnum = QString::number(codenum);
    QString errorinfo = "Error on Instruction " + getnum;
    error->setText(errorinfo);
    confirmbutton = error->addButton("Retry",QMessageBox::AcceptRole);
    error->show();
    connect(confirmbutton,&QPushButton::clicked,this,[=](){
        op_clear();
        ui->stackedWidget->setCurrentIndex(0);
    });
    return;
}
```

*move_robot*函数：单独控制机器人及其所持数字块的移动，使用了*QPropertyAnimation*。

```
void Widget::move_robot(int destination){

    int x = ui->robot->pos().x();
    int y1 = 110, y2 = 70;

    QPropertyAnimation *transrobot = new QPropertyAnimation(ui->robot, "geometry");
    transrobot->setDuration(500);
    transrobot->setStartValue(QRect(x, y1, 53, 121));
    transrobot->setEndValue(QRect(destination, y1, 53, 121));
    transrobot->setEasingCurve(QEasingCurve::OutBounce);

    QPropertyAnimation *transcurrent = new QPropertyAnimation(ui->currentline, "geometry");
    transcurrent->setDuration(500);
    transcurrent->setStartValue(QRect(x, y2, 53, 41));
    transcurrent->setEndValue(QRect(destination, y2, 53, 41));
    transcurrent->setEasingCurve(QEasingCurve::OutBounce);

    QParallelAnimationGroup *AnimationGroup = new QParallelAnimationGroup;

    AnimationGroup->addAnimation(transcurrent);
    AnimationGroup->addAnimation(transrobot);

    AnimationGroup->start();
}
```

$delay$函数用于延时，在机器人工作间进行停顿。

```
void Widget::delay(int milliseconds) {
    QEventLoop loop;
    QTimer::singleShot(milliseconds, &loop, &QEventLoop::quit);
    loop.exec();
}
```

$inbox$、$outbox$、$copyfrom$、$copyto$、add、sub、$jump$、$jumpifzero$、$jumpifnegative$、$bump$、$time\_to\_break$、$anscheck$、$selfanscheck$函数，和CLI版本没有太大差别，不再赘述。

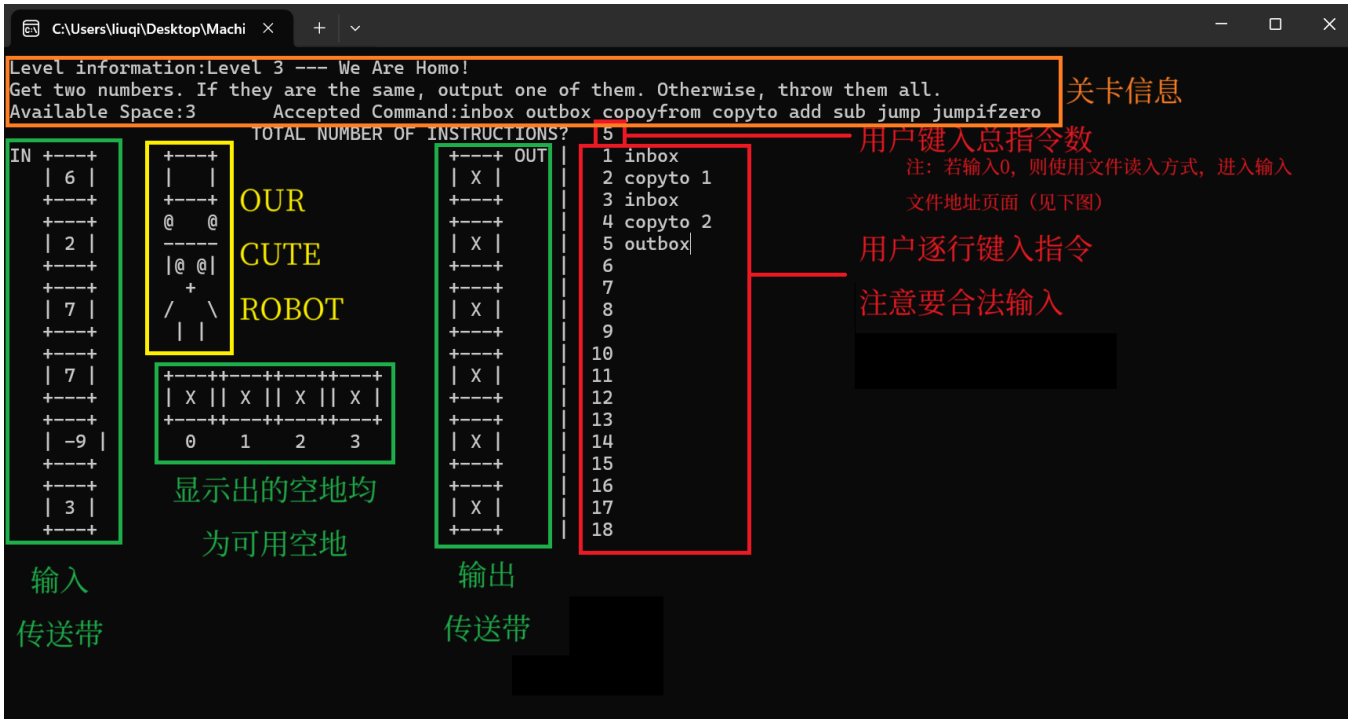
三、整体界面设计及玩法介绍

（一）*CLI version*

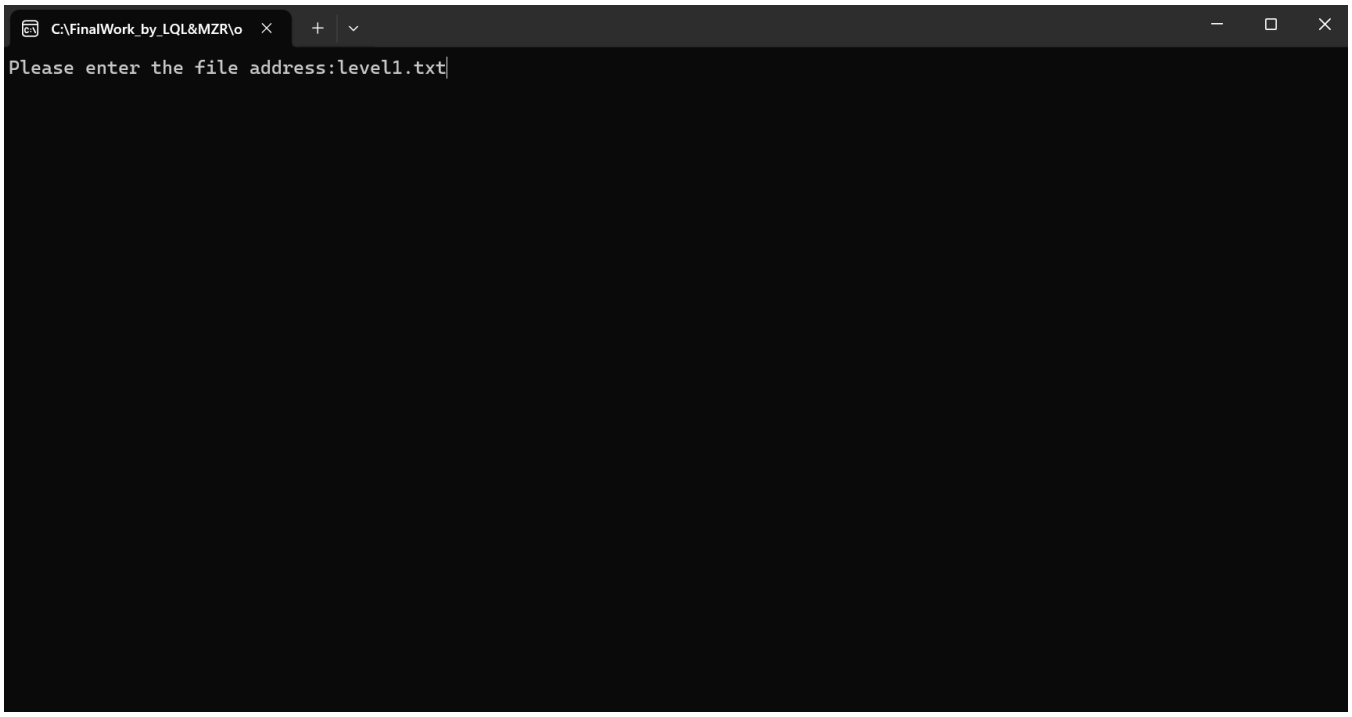
进入游戏后，可以根据从已经解锁的关卡中选择一个，输入对应的数字，即可进入该关卡。如果输入的数字不合法，则会提示”*No Access*”并要求用户重新输入。用户输入大写字母*Q*，可以退出游戏。

进入关卡页面后，用户需要首先输入总共的指令数，随后一行一行输入。当输入完毕后，游戏会自动开始运行。运行结束后，无论成功、失败或故障，都会回到主页面。

↑图3.1.1，*CLI*选关页面说明



↑图3.1.2，*CLI*关卡页面说明



↑图3.1.3，*CLI*机器人指令“文件读入”之新页面

（二）*GUI version*

进入游戏后，玩家首先可以看到我们的幽默标题，随后可以点击按钮进入关卡（只有已解锁

的关卡才能被点击），也可以按下$Quit$按钮退出游戏。

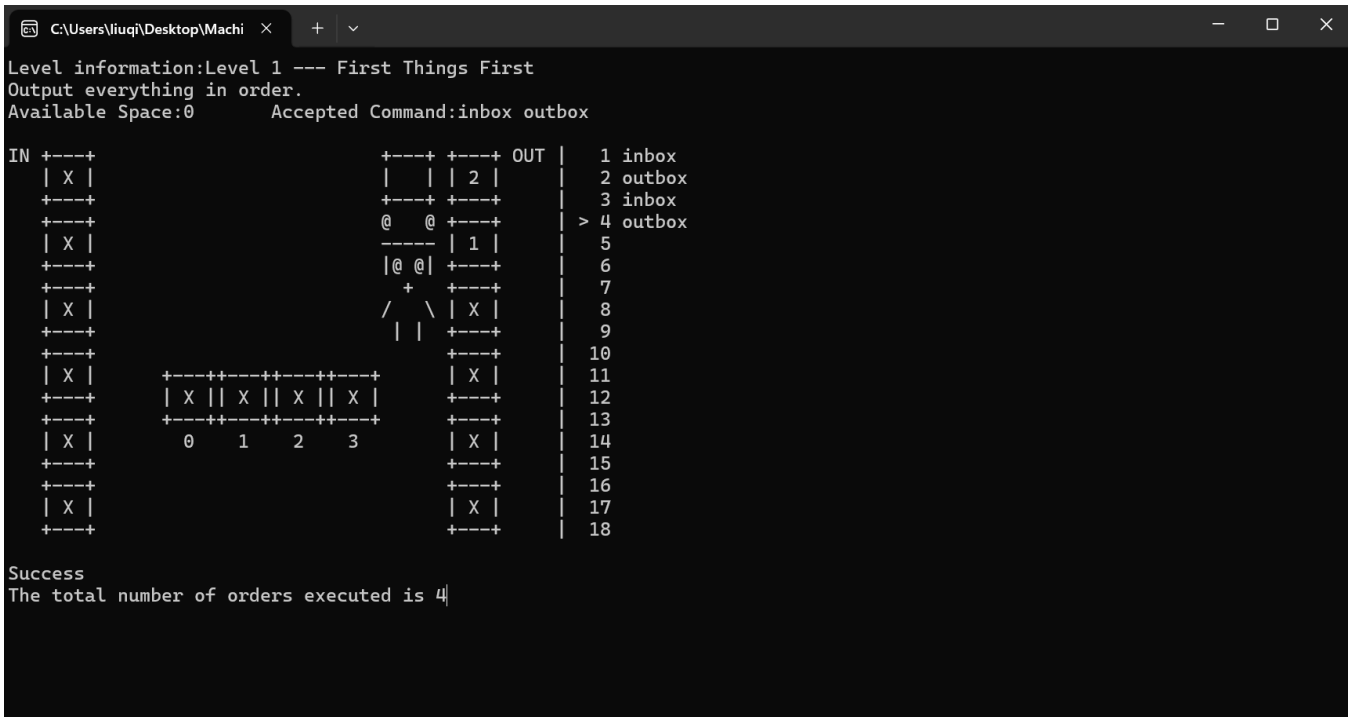
进入关卡页面后，用户不再需要输入总共的指令数。只需要一行一行键入指令（注意，每输入完一行指令需要按下回车，同时该行会被锁定）。键入完毕后，按下$Execute$按钮，可以开始执行。运行结束后，会弹出对应消息框（成功或失败或故障），点击$Retry$或者$Next$都可以回到选关界面。

↑图3.2.1，GUI选关页面说明

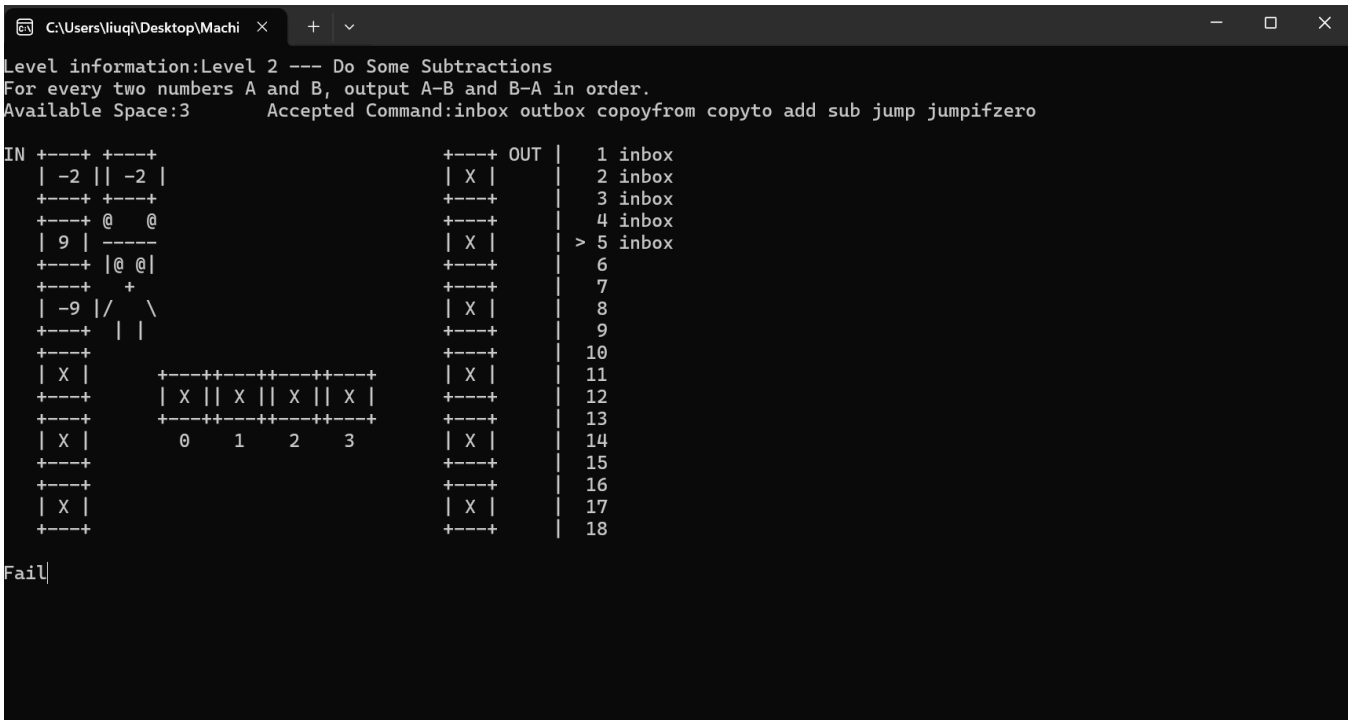
↑图3.2.2，*GUI*关卡页面说明

四、游戏测试

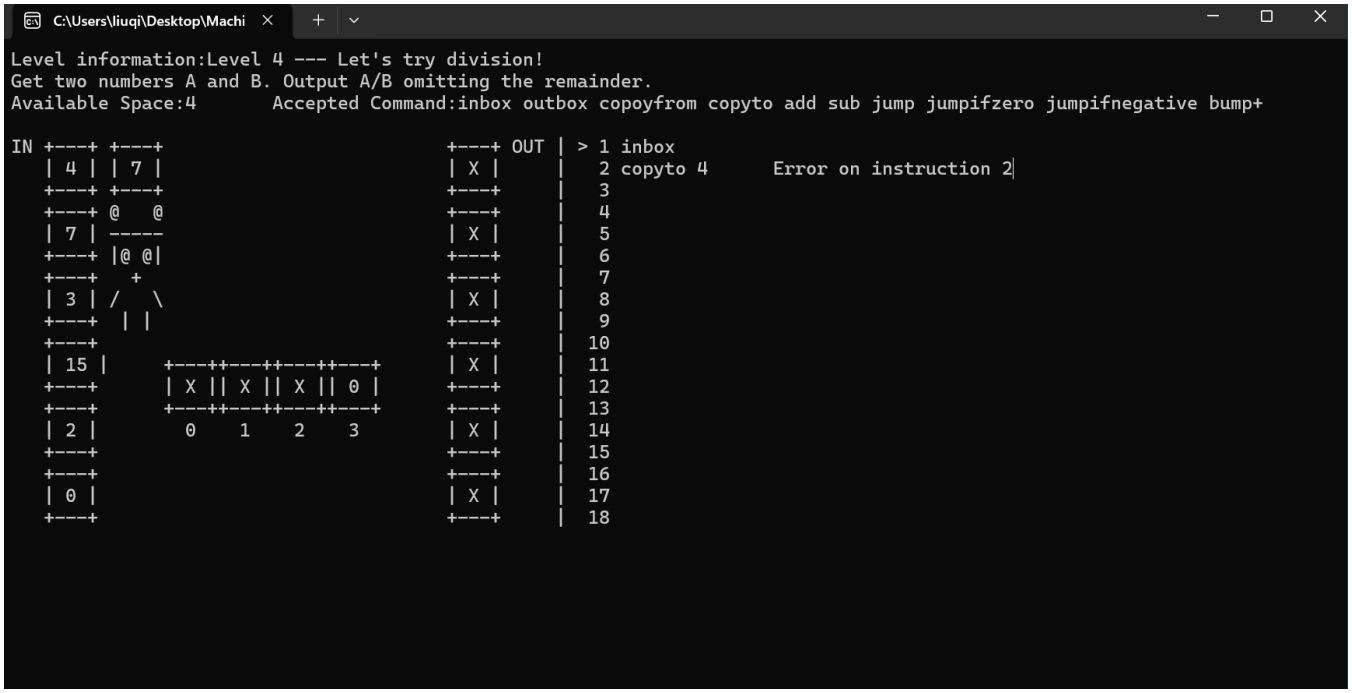
（1） CLI版本：



↑图4.1.1，CLI运行成功提示（$Success$）

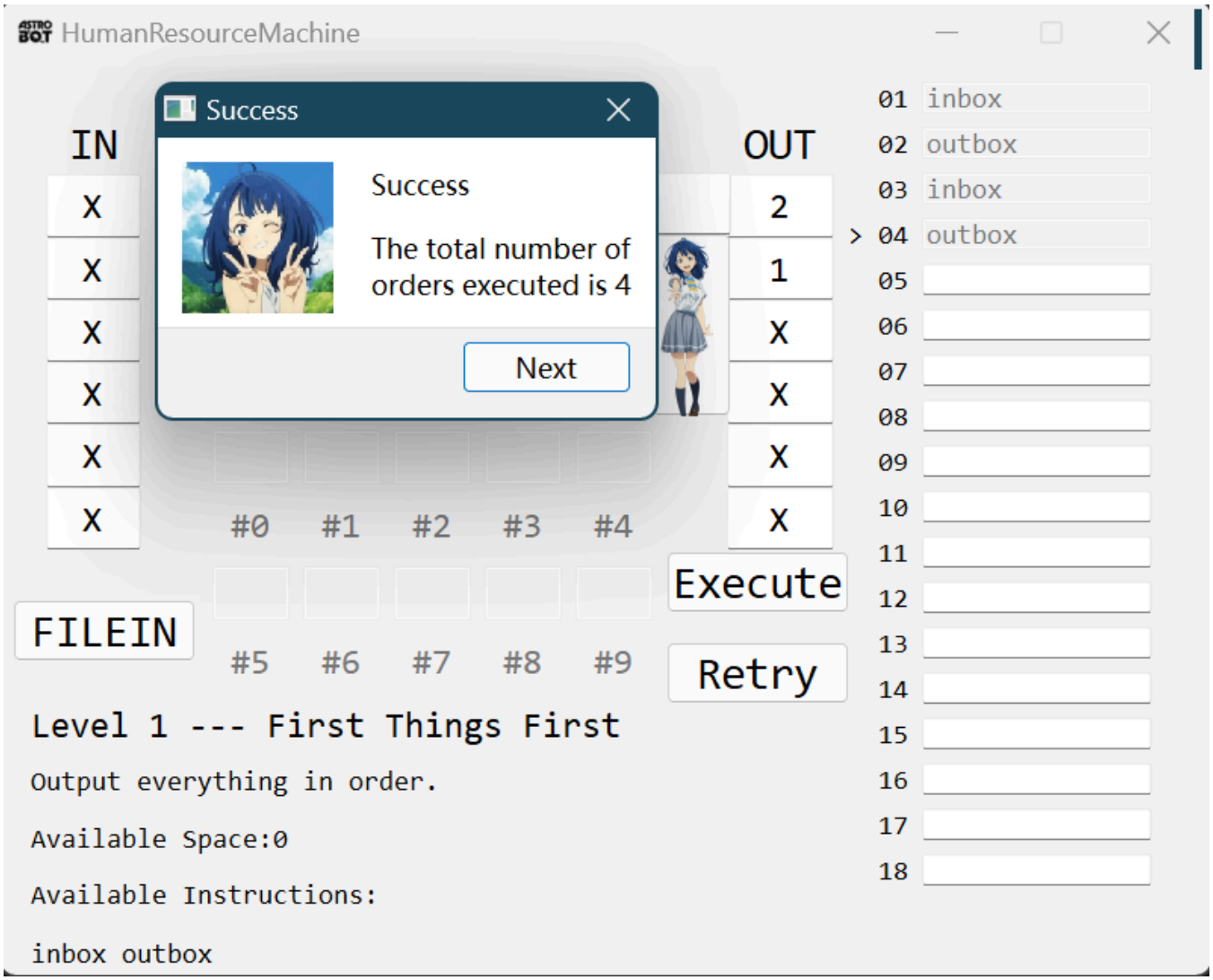


↑图4.1.2，CLI运行失败提示（$Fail$）

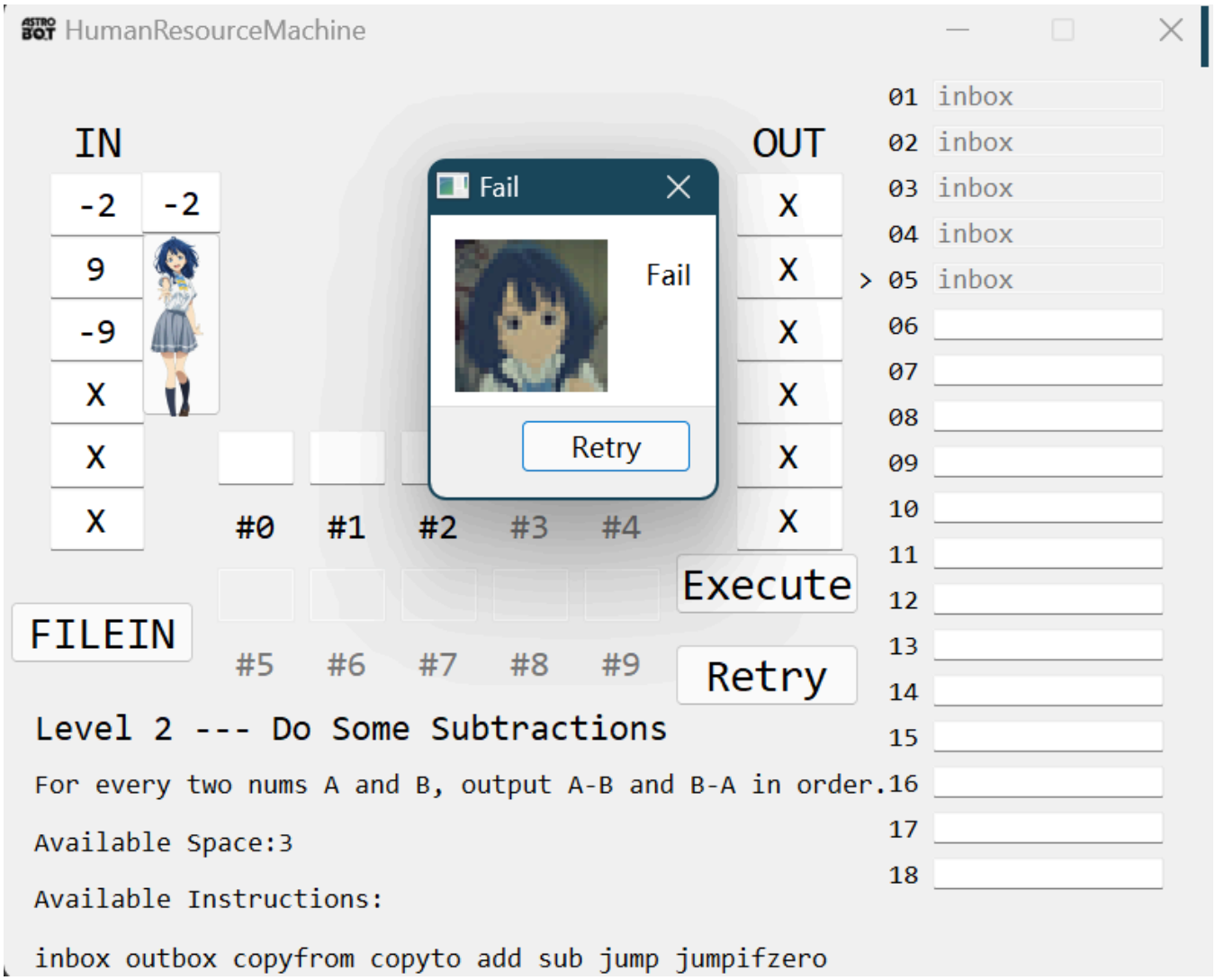


↑图4.1.3，CLI运行故障提示（$Error$）

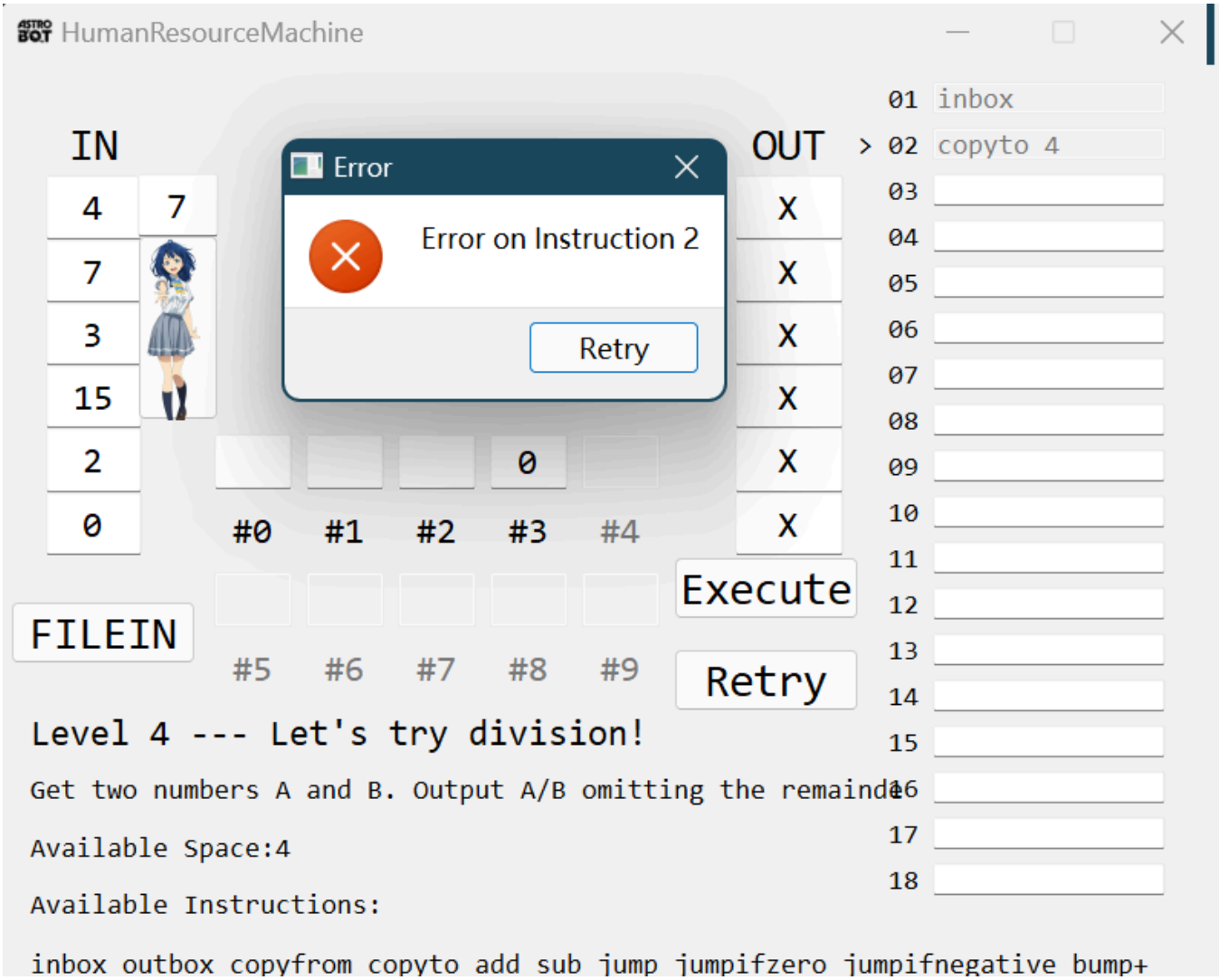
（2） *GUI*版本：



↑图4.2.1，*GUI*运行成功提示（*Success*）



↑图4.2.2，*GUI*运行失败提示（*Fail*）



↑图4.2.3，GUI运行故障提示（$Error$）

五、自由创新关卡

（1） 关卡描述：

从输入序列中依次取2个数字A和B，输出A/B的商（忽略余数）。重复直至传送带为空。

（2） 权限参数：

可用空地数：4

可用指令：$inbox$，$outbox$，$copyfrom$，$copyto$，add，sub，$jump$，$jumpifzero$，$jumpifnegative$，$bump+$，$bump-$

其中，$jumpifnegative\ x$表示如果当前积木小于0，则将后续执行的指令改为从第x条指令开始执行。如果当前积木不为0，则不做任何操作；$bump+$表示把第i块空地上的数$+1$，并复制到机器人手中；$bump-$表示把第i块空地上的数-1，并复制到机器人手中。

（3） 输入输出：

输入序列：7，4，7，3，15，2，0，9
目标序列：1，2，7，0

（4） 参考解法：

```
15
copyfrom 3
copyto 2
inbox
copyto 1
inbox
copyto 0
copyfrom 1
sub 0
jumpifnegative 13
copyto 1
bump+ 2
jump 7
copyfrom 2
outbox 1
jump 1
```

六、小组分工

刘青乐：整体框架的搭建、实现图形化界面GUI、录制讲解视频

马子润：实现操作函数，OJ平台debug、撰写作业报告

附开发日志：

```
000 Oct.15, Liu Qingle: Level_1
001 Oct.17, Ma Zirun: relevant operation functions
002 Oct.24, Ma Zirun: Level_2
003 Oct.26, Liu Qingle: Level_3 , Level_4 and levelselection_page
004 Nov.02, Liu Qingle: Self-Defined-Level (external file input)
005 Nov.22, Ma Zirun: debug, and OJ checking
006 Nov.30, Ma Zirun: updating and integrating the whole program
007 Dec.11, Liu Qingle: QT GUI designing
008 Dec.17, Liu Qingle: QT GUI perfect version
009 Dec.30, Liu Qingle: QT GUI file input function!!!!!
010 Dec.30, Ma Zirun: CLI file input function!!!!!
```

※七、其他的重要补充说明

1. 变量含义的说明（CLI与GUI大体相同）

（1） 结构体$opr\ rob[]$储存每个指令的名称($oprname$)、指令后接的数字($oprpos$，例如“$jump\ 10$”的“10”就是储存在$oprpos$中的；$inbox$和$outbox$的$oprpos = -1$)、当前指令是否合法。

（2）结构体$worker$储存机器人的当前位置($curpos$)、当前手上拿的数字($curnum$，初始化为-114514)。

（3）$record$记录从$record.txt$文件中读入的、之前通过的关卡序号；$levelselect$为用户输入的、当前想玩的关卡序号；$number$为用户输入的指令总数。

（4）$stringjustdoit$也是记录用户输入的想玩的关卡序号。如果$justdoit$为‘Q’，程序退出；如果$justdoit$为其他字符串，输出“$NO\ \ ACCESS$”；如果$justdoit$为一个合法的关卡数，则将$justdoit$转化为$levelselect$。

（5）$queue\ \ input$记录输出传送带的情况；$stack\ \ output$记录输出传送带的情况。

（6）$space[]$记录空地上的情况。

（7）cnt记录程序总共执行了多少次指令，在成功通过关卡的同时输出。

（8）$instructionaccess[i][j]$记录第i关的第j条指令是否能被使用，1表示可以，0表示不可以。其中，11种指令的标号分别为：$inbox:0$, $outbox:1$, $copyfrom:2$, $copyto:3$, $add:4$, $sub:5$, $jump:6$, $jumpifzero:7$, $jumpifnegative:8$, $bump+:9$, $bump-:10$.

（9）$spaceaccess[i]$记录第i关可以被使用的空地个数

```
struct opr{
    string oprname;
    int oprpos=-1;
    bool illegal = 0;
}rob[105];
struct worker{
        int curpos;
        int curnum;
};
int record,levelselect,number;
string justdoit;
queue <int> input;
stack <int> output;
int space[25] = {0};
int cnt = 0;
int instructionaccess[10][11]={{0,0,0,0,0,0,0,0,0,0,0},{1,1,0,0,0,0,0,0,0,0,0},{1,1,1,1,1,1,1,1,0,
//inbox0 outbox1 copyfrom2 copyto3 add4 sub5 jump6 jumpifzero7 jumpifnegative8 bump+9 bump-10
int spaceaccess[10]={0,0,3,3,4};
```

2. $selfdefine$自定义关卡外部导入的说明

下面是$selfdefine.txt$自定义关卡外部导入的格式，我们对其含义进行解释说明。

```
9
9 11 2 0 18 15 5 3 0
2
2 3
1 1 1 1 1 1 1 1 1 1
2
-1000000000 -1000000000
DLC Level --- MIN
Find the minimum of every 0-ending subsequence.
Available Space:2
inbox outbox copyfrom copyto add sub jump jumpifzero jumpifnegative bump+-
```

第一行一个数字，表示$inbox$传送带的数字个数，记作A。
第二行A个数字，表示$inbox$传送带分别有哪些数字。
第三行一个数字，表示正确输出到$outbox$传送带应该有几个数字，记作B。

第四行B个数字，表示正确输出到$outbox$传送带的数字分别是谁。

第五行11个数字，表示11种指令分别是否有权限，1为有，0为无。顺序为：$inbox:0$, $outbox:1$, $copyfrom:2$, $copyto:3$, $add:4$, $sub:5$, $jump:6$, $jumpifzero:7$, $jumpifnegative:8$, $bump+:9$, $bump-:10$.

第六行2个数字，表示可用的空地的数量，记为C。

第七行C个数字，表示可用的空地分别的初始化值，若要将空地初始为空，则标记为-10^9。

第八至十一行分别为一个字符串，表示要输出的关卡信息，包括关卡名、关卡要求、关卡可用空地数与可用指令。

另外，CLI版本的自定义关卡是通过包含另一个cpp文件实现的，$selfdefine.cpp$格式如下，与上面大体相同。

```
#include<iostream>
#include<queue>
# define inf 1e9

void selfprintinfo();
int selfinboxtotal=2;
int selfoffer[2]={20,6};
int selfoutboxtotal=12;
int selfexamine[12]={1,1,2,3,5,8,13,1,1,2,3,5};
int selfinstructionaccess[11]={1,1,1,1,1,1,1,1,1,1,1};
int selfspaceaccess=5;
int selfspacevalue[5]={-inf,-inf,-inf,-inf,0};

void selfprintinfo(){
        printf("Level information:Self-Defined Level --- Fibonacci\n");
        printf("For every number N, output fibonacci elements which < N.\n");
        printf("Available Space:5        Accepted Command:inbox outbox copyfrom copyto add sub jum
        return;
}
```

※※3. 我们的文件夹内容

（1） 麻烦助教老师把本文件夹（注意是FinalWork_by_LQL&MZR，而不是“41组-程设大作业-刘青乐马子润”！！！）以该名字直接置于C盘根目录运行，否则代码中路径无法识别。

（2） "C:\FinalWork_by_LQL&MZR\code"文件夹包含GUI项目全部文件，包含项目文件、头

文件、源代码文件、资源文件、配置文件。其中，源代码包括头文件"widget.h"和两个cpp文件"main.cpp""widget.cpp"。配置文件包括：用来记录游戏存档的"record.txt"和自定义关卡的配置文件"selfdefine.txt"。

（3） "C:\FinalWork_by_LQL&MZR\release"文件夹中包含可执行程序"Astrobot.exe"及其dll应用程序扩展，可以直接运行。

（4）“C:\FinalWork_by_LQL&MZR\oldversion(CLI)”文件夹包含旧的CLI版本代码，以及记录存档的"record.txt"，自定义关卡的配置文件"selfdefine.cpp"和关卡的参考通关指令"op.txt"。

（5） 其余文件是QtCreator在调试、生成和注入文件过程中产生的杂七杂八的文件，烦请您忽略它们。

※※4. 代码调试和游戏运行

（1）若要检测我们的旧版本（命令行界面），请进入“C:\FinalWork_by_LQL&MZR\oldversion(CLI)”，具体的用法在大作业报告中所有的CLI部分有详细介绍。

（2）（推荐）若要检测我们的GUI版本，请进入“C:\FinalWork_by_LQL&MZR\release”，运行其中的"Astrobot.exe"进行游戏，游戏功能、玩法在大作业报告中均有呈现。

（3）假如遇到不幸（几率比较小），在您的电脑缺少一些配置文件导致无法运行"Astrobot.exe"，则请您在QtCreator中打开"C:\FinalWork_by_LQL&MZR\code\Firstthingsfirst.pro"项目文件并运行，或参考演示视频。

（4）我们的报告写的比较长，代码也比较臃肿，给您添麻烦了！！助教老师辛苦了！！！