# Human Resource Machine 实验报告

## 一、设计思路

**1.需求分析：**

程序需要实现一个游戏的功能。要求存储输入的积木序列、输出序列、空地，目标序列和当前积木，以及实现一系列的指令。此外还有选择关卡并且保存通关信息、显示游戏界面等基础要求。

**2.具体解决：**

我们用a数组存储输入的积木序列，b数组存储输出序列，c数组存储空地，d数组存储目标序列，用r表示当前积木数，用1024表示当前位置没有积木，用instruction数组和x数组记录下指令。我们声明了inbox等函数，来实现指令。

选择关卡部分我们用文件相关的知识解决，详见“三、选择关卡界面的设计”。

显示游戏界面的函数实现如下：

```
4   int getConsoleHeight() {//计算控制台高度
5       HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
6       CONSOLE_SCREEN_BUFFER_INFO consoleInfo;
7       GetConsoleScreenBufferInfo(hConsole, &consoleInfo);
8       int Height = consoleInfo.srWindow.Bottom - consoleInfo.srWindow.Top + 1;
9       return Height;
10  }
11  int getConsoleWidth() {//计算控制台宽度
12      HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
13      CONSOLE_SCREEN_BUFFER_INFO consoleInfo;
14      GetConsoleScreenBufferInfo(hConsole, &consoleInfo);
15      int Width = consoleInfo.srWindow.Right - consoleInfo.srWindow.Left + 1;
16      return Width;
17  }
18  void setCursorPosition(int x, int y) {
19      HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
20      COORD pos;
21      pos.X = x;
22      pos.Y = y;
23      SetConsoleCursorPosition(hConsole, pos);//一个已有的函数
24  }
25  void printVerticalLine(int x, int y, int length) {
26      setCursorPosition(x, y);
27      for (int i = 0; i < length; i++)
28      {
29          cout << "|";
30          setCursorPosition(x, y + i + 1);
31      }
32  }
33  void printsquare(int x, int y, int a) {
34      setCursorPosition(x, y); cout << "+---+" << endl;
35      setCursorPosition(x, y + 1);
36      if (a != 1024)cout << "|" << setw(3) << a << "|" << endl;
37      if (a == 1024)cout << "|" << "   " << "|" << endl;
38      setCursorPosition(x, y + 2); cout << "+---+" << endl;
39  }
40  void printRobot(int x, int y, int a) {
41      setCursorPosition(x, y);
42      printsquare(x, y, a);
43      setCursorPosition(x, y + 3); cout << "@   @" << endl;
44      setCursorPosition(x, y + 4); cout << "-----" << endl;
45      setCursorPosition(x, y + 5); cout << "|@ @|" << endl;
46      setCursorPosition(x, y + 6); cout << "  +  " << endl;
47      setCursorPosition(x, y + 7); cout << "/   \\" << endl;
48      setCursorPosition(x, y + 8); cout << " | | " << endl;
49  }
```

```
void level1n(int a[], int b[], int c[], int d[], int m1, int m2, int m3) {
    setCursorPosition(0, 2);
    cout << "IN" << endl;
    for (int i = 0; i < m1; i = i + 1)
        printsquare(0, 3 * i + 3, a[i]);
    setCursorPosition(wid * 2 / 3 - 5, 2);
    cout << "OUT" << endl;
    for (int i = 0; i < m2; i = i + 1)
        printsquare(wid * 2 / 3 - 5, 3 * i + 3, b[i]);
    for (int k = 0; k < m3; k = k + 1)
    {
        printsquare(wid / 3 - 4 * numc + 8 * k, 15, c[k]);
        setCursorPosition(wid / 3 - 4 * numc + 2 + 8 * k, 18); cout << k;
    }
    setCursorPosition(28, 20);
    cout << "Goal:";
    for (int i = 0; i < m2; i++) {
        if (i != m2 - 1)cout << d[i] << ",";
        else cout << d[i];
    }
    setCursorPosition(wid * 5 / 6 - 6, 2);
    cout << "Instructions";
    printVerticalLine(wid * 2 / 3, 2, hei - 1);
}
void clearRobot(int x, int y) {
    for (int k = y; k <= y + 8; k = k + 1) {
        setCursorPosition(x, k); cout << "      " << endl;
    }
}
void moveRobot(int x, int y, int z, int w, int a, int b) {
    int pos = x;
    while (pos != z) {
        clearRobot(pos, y);
        printRobot(pos + (z - x) / abs(z - x), w, a);
        pos = pos + (z - x) / abs(z - x);
        this_thread::sleep_for(chrono::milliseconds(10));
    }
    printRobot(z, w, b);
}
```

通过计算出控制台的高度和宽度以及设置光标的位置，实现在对应位置打印出机器人空地等图形。moverobot函数实现了对于机器人运行的逐步模拟。具体实现效果可以参见下图以及自行调试。


**3.扩展功能:**

我们实现了玩家自定义增加关卡的功能。为了主函数的简明性，我们将不同关卡的信息都保存在对应的文件中，可以直接进行读入，从而避免了大量重复的if分支结构，也便于玩家自定义增加关卡。实现代码详见“三、选择关卡界面的设计”。

## 二、工程结构

为了主函数的简明性，我们将文件拆分为function.h,function1.cpp,主函数.cpp三个文件。另外还有一系列的资源文件。

其中function.h文件声明函数和全局变量、结构体（如下图）：

```
#ifndef HEADER_FILE_NAME_H  // 如果未定义HEADER_FILE_NAME_H宏，则执行下面的代码
#define HEADER_FILE_NAME_H  // 定义HEADER_FILE_NAME_H宏
#pragma once
#include<iostream>
#include<cmath>
#include<string>
#include<iomanip>
#include<cstdlib>
#include<windows.h>
#include<vector>
#include<fstream>
#include<thread>
#include<chrono>
#include<sstream>
using namespace std;

struct Level {
    int number;
    bool passed;
};//关卡结构体，包含关卡编号和是否通过的状态
extern string instruction[1000], input[1000],inst1[10];//记录指令
extern int m, x[1000], v[1000], r, total;//m表示指令条数，x记录指令参数
extern vector<Level> levels; //加载关卡状态
```

function1.cpp对于声明的函数进行了定义，主要实现显示界面、指令集、选择关卡等功能，主函数.cpp中调用对应的函数实现游戏功能。

资源文件分别以“level+关卡编号+.txt”和”level+关卡编号+data”命名，前者对应非自定义关卡的正确答案，后者对应各关卡的关卡信息。其中第五关以后文件初始为空，可以由玩家自行设计。

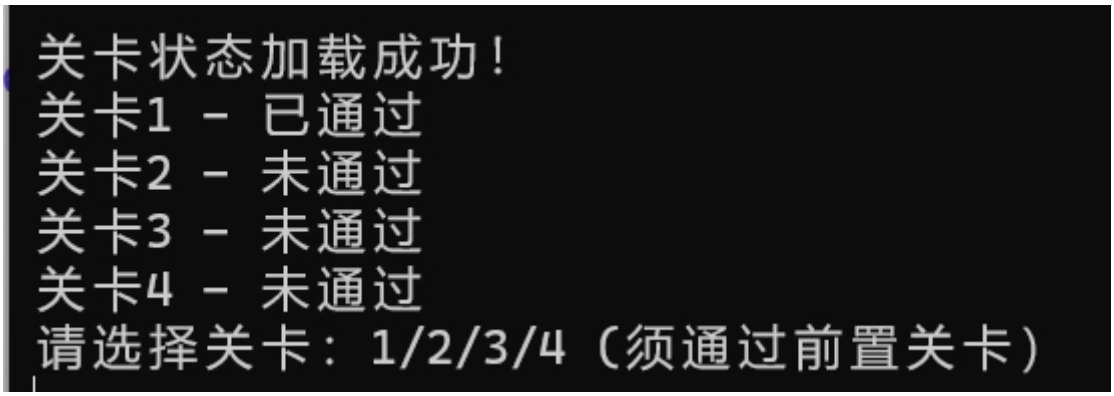
## 三、选择关卡界面的设计

函数实现如下图：

```cpp
vector<Level> load() {//从文件加载关卡状态
    vector<Level> levels;
    ifstream fin("levels.txt");
    if (fin) {
        int number;
        bool passed;
        while (fin >> number >> passed) {
            Level leveltemp;
            leveltemp.number = number;
            leveltemp.passed = passed;
            levels.push_back(leveltemp);
        }
        fin.close();
        setCursorPosition(0, 0); cout << "关卡状态加载成功！" << endl;
    }
    else {
        setCursorPosition(0, 0); cout << "无法加载关卡状态！" << endl;
    }
    return levels;
}
void chooselevel() {//选择关卡
    for (int i = 0; i < levels.size(); i++) {
        setCursorPosition(0, i + 1);
        cout << "关卡" << levels[i].number;
        if (levels[i].passed) cout << " - 已通过";
        else cout << " - 未通过";
        cout << endl;
    }
}
bool checklevel(int selected) {
    if (selected > levels.size()) return 0;//超出了关卡数
    for (int i = 0; i < selected - 1; i++) {//前置关卡没有通过
        if (!levels[i].passed) return 0;
    }
    return 1;
}
void save(vector<Level>& levels) {//采用一个引用运算符，减少复制的浪费
    ofstream fout("levels.txt");
    if (fout) {
        for (int i = 0; i < levels.size(); i++) {//遍历levels中的每个元素
            fout << levels[i].number << " " << levels[i].passed << endl;
        }
        fout.close();
        cout << "关卡状态保存成功!" << endl;
    }
    else {
        cout << "无法保存关卡状态!" << endl;
    }
}//保存关卡状态到文件
void renew(vector<Level>& levels){
    ofstream fout("levels.txt");
    if (fout) {
        for (int i = 0; i < levels.size(); i++) {//遍历levels中的每个元素
            levels[i].passed=0;
            fout << levels[i].number << " " << levels[i].passed << endl;
        }
        fout.close();
        cout << "关卡状态重置成功!" << endl;
    }
    else {
        cout << "关卡状态重置失败!" << endl;
    }
}
```

load函数通过声明一个新的结构体vector数组，并且从文件中读入，然后不断加入数组，最后将数组返回，实现从文件中的加载。

chooselevel函数将文件中的信息输出到屏幕上，让用户对前面关卡的通关情况有所了解。实现效果如下图：




save函数将结构体vector数组中的元素读入文件中并且实现对原文件的覆盖，实现通关信息的记录，并且在关闭后也可以保存。

此外，我们还实现了玩家自定义增加关卡、删除关卡以及重置通关状态的扩展功能。重置通关状态以及增加关卡实现代码如下：

```
27          if (ans0 == 999) {
28              renew(levels); clearScreen(); chooselevel(cntk);
29          }
30          if (ans0 == 0) {
31              clearPart(50, cntk + 5); setCursorPosition(0, 0);
32              string stra;
33              stringstream ss0; ss0 << "level" << cntk + 1 << "data" << ".txt"; stra = ss0.str();
34              ofstream file(stra.c_str());
35              int m1, m2, m3, m4;//m1表示输入的积木数，m2表示输出的积木数，m3表示空地数，m4表示指令数
36              if (!file) {
37                  cout << "关卡文件打开失败" << endl;
38              }
39              cout << "请输入输入序列的积木个数" << endl;
40              cin >> m1;
41              file << m1 << endl;
42              cout << "请输入" << m1 << "个积木的数字" << endl;
43              for (int i = 1; i <= m1; i++) {
44                  int temp;
45                  cin >> temp;
46                  file << temp << " ";
47              }
48              file << endl;
49              cout << "请输入输出序列的积木个数" << endl;
50              cin >> m2;
51              file << m2 << endl;
52              cout << "请输入" << m2 << "个积木的数字" << endl;
53              for (int i = 1; i <= m2; i++) {
54                  int temp;
55                  cin >> temp;
56                  file << temp << " ";
57              }
```

```
            file << endl;
            cout << "请输入可用的空地数（最多8个）" << endl;
            cin >> m3;
            while (m3 < 0 || m3 >= 9) {
                cout << "空地数目过多！\n"; cin >> m3;
            }
            file << m3 << endl;
            cout << "请输入可用的指令数" << endl;
            cin >> m4;
            file << m4 << endl;
            cout << "请输入" << m4 << "个可用的指令" << endl;
            for (int i = 1; i <= m4; i++) {
                string temp;
                cin >> temp;
                while (!checkInstruction(2, temp)) {
                    cout << "输入的指令不在指令集中，请重新输入" << endl;
                    cin >> temp;
                }
                if (i != m4) {
                    file << temp << " "; inst1[i] = temp;
                }
                else file << temp;
            }
            file << endl;
            file.close();
            Level temp;
            temp.number = cntk + 1;
            temp.passed = 0;
            levels.push_back(temp);
            save(levels);
```

玩家通过自行读入一组文件数据到“level+关卡编号+data.txt”中并将关卡信息插入结构体数组levels当中，实现了关卡信息的保存。

删除关卡实现代码如下：

```
        if (ans0 == 1024) {
            setCursorPosition(0, cntk + 4);
            for (int i = 5; i <= cntk; i++) {
                string filename, filen;
                stringstream ss1; ss1 << "level" << i << "data" << ".txt"; filename = ss1.str();
                stringstream ss2; ss2 << "level" << i << ".txt"; filen = ss2.str();
                remove(filename.c_str()); remove(filen.c_str());
            }
            ifstream fin("levels.txt");
            ofstream fout("temp.txt");
            string line1;
            for (int i = 1; i <= 4; i++) {
                getline(fin, line1);
                fout << line1 << endl;
            }
            fin.close(); fout.close();
            remove("levels.txt");
            rename("temp.txt", "levels.txt");
            cout << "是否继续？（1：初始化选关界面/其他：退出）\n";
            char outgame; cin >> outgame;
            if (outgame == '1') { clearScreen(); chooselevel(4); }
            else return 0;
        }
```

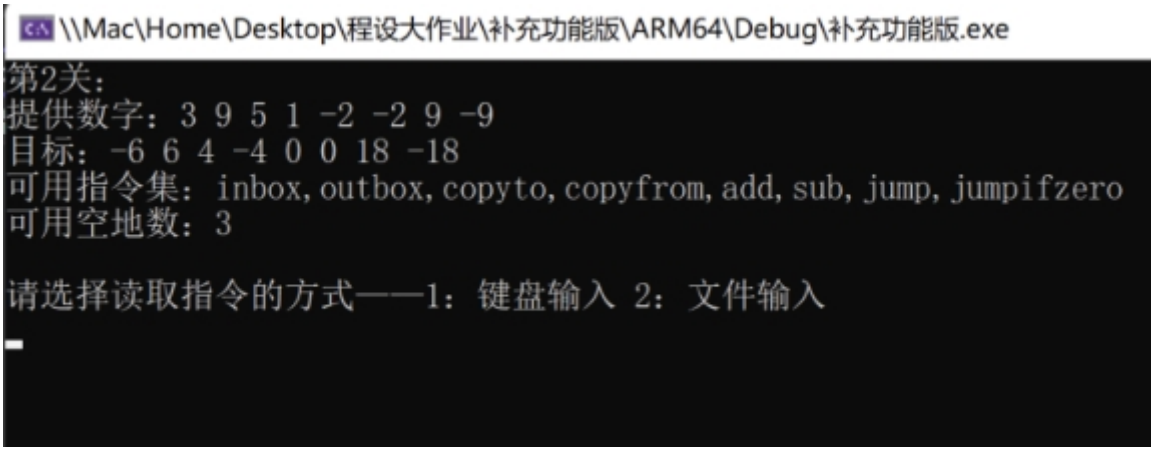
## 四、游戏测试

**1、选关界面操作：**

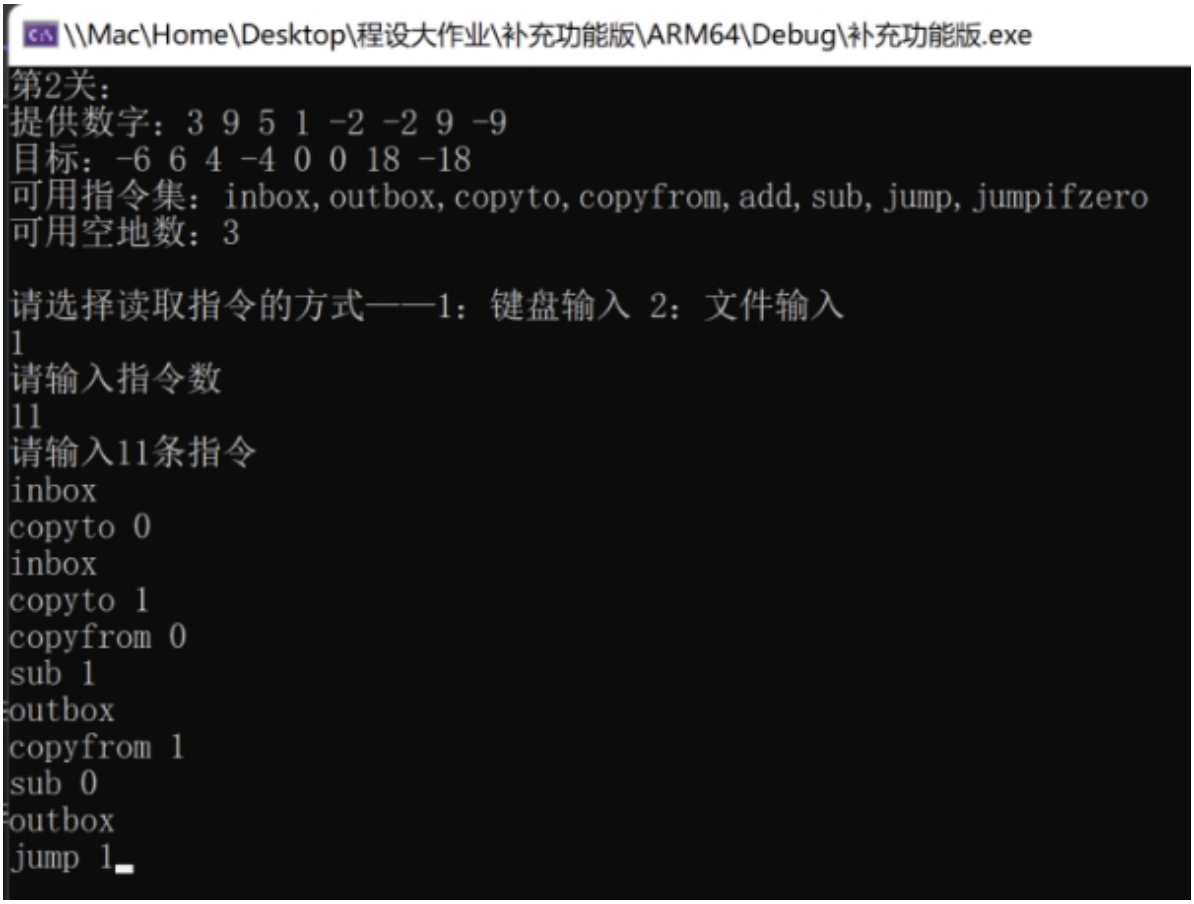

打开游戏进入选关界面，游戏会提示玩家选择一个关卡。初始状态下只能选择1-4关。如果前置关卡没有完成，而选择了后面的关卡，或者选择了不存在的关卡，游戏会提示当前选择不合理。

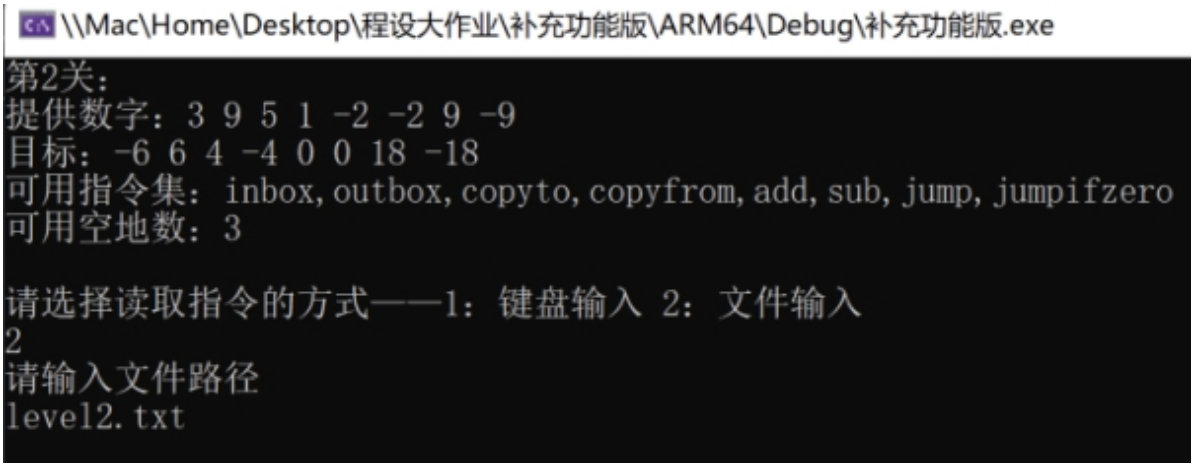
在选关界面还可以进行一些其他操作：输入0，会进入新增关卡页面，玩家可以自定义增加关卡；输入999，系统会重置目前的通关状态（所有关卡显示“未通过”）；输入1024，系统会清除新增的自定义关卡及其文件。

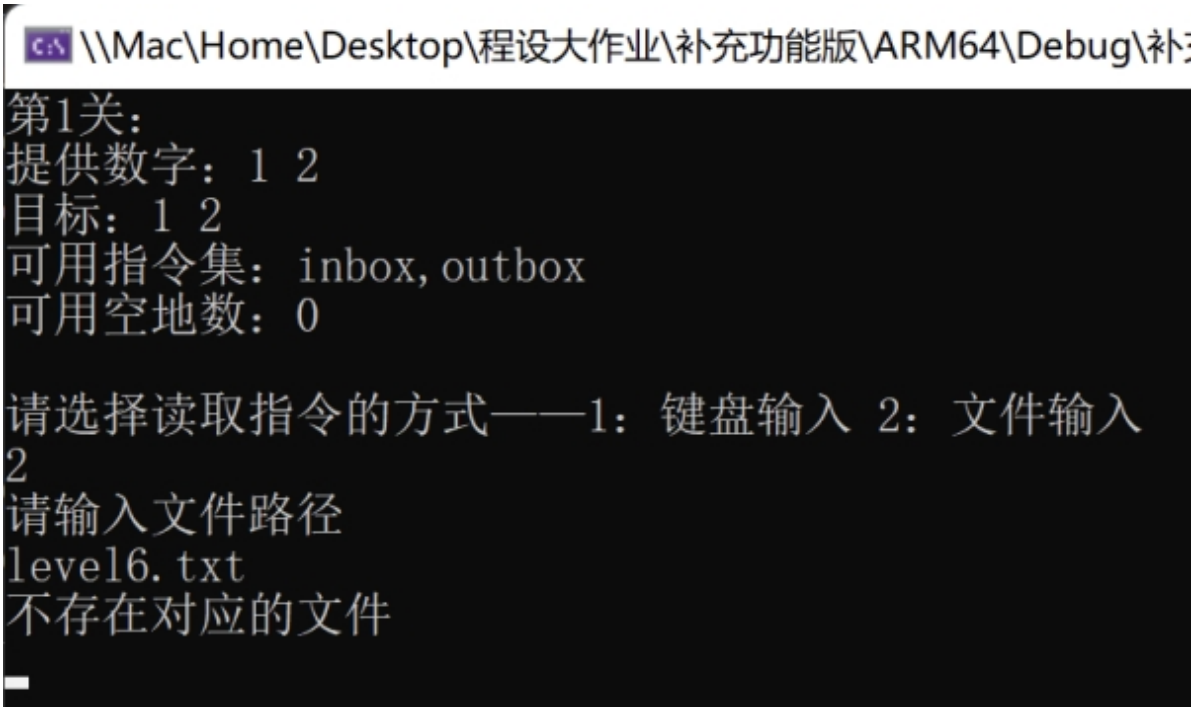
**2、关卡准备界面操作（以第二关为例）**




选择一个关卡，系统从存储中读取这个关卡的配置信息，进入准备界面。系统会询问玩家是要选择用键盘手动输入指令（1）还是用文件输入指令（2）。选择1，则系统询问玩家要输入多少条指令，并且等待玩家输入完所有指令。选择2，输入对应的文件路径，则游戏会根据玩家输入的文件路径，从存储中读取玩家提前预设好的指令文件。如果文件不存在，则提示“不存在对应的文件”。

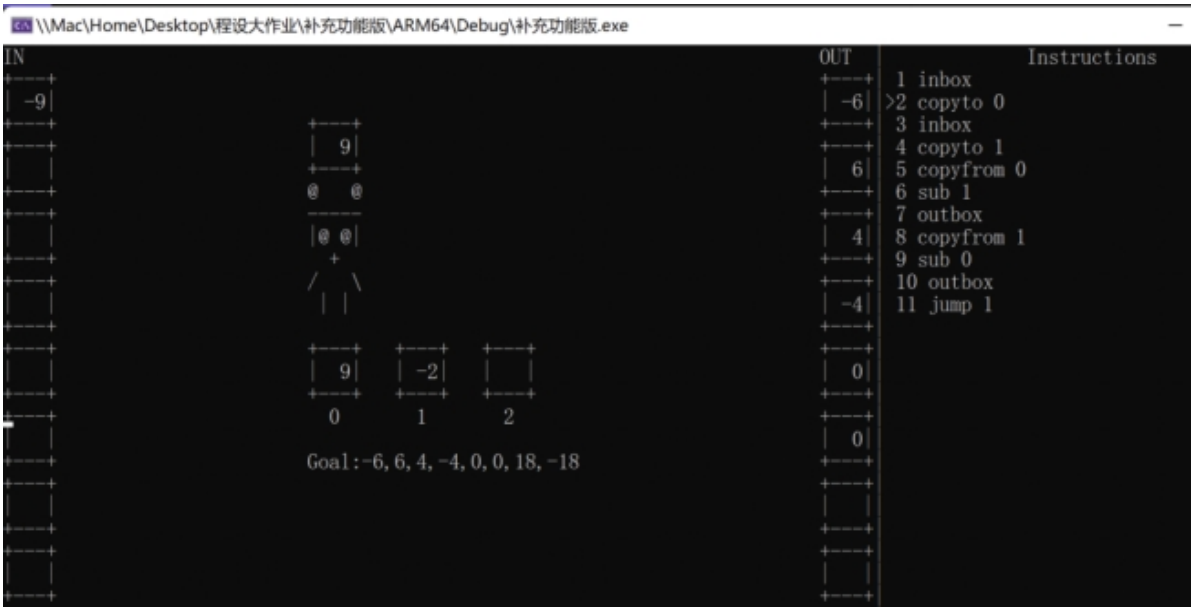

\\Mac\Home\Desktop\程设大作业\补充功能版\ARM64\Debug\补充功能版.exe
第2关:
提供数字: 3 9 5 1 -2 -2 9 -9
目标: -6 6 4 -4 0 0 18 -18
可用指令集: inbox,outbox,copyto,copyfrom,add,sub,jump,jumpifzero
可用空地数: 3
请选择读取指令的方式——1: 键盘输入 2: 文件输入
1
请输入指令数
11
请输入11条指令
inbox
copyto 0
inbox
copyto 1
copyfrom 0
sub 1
outbox
copyfrom 1
sub 0
outbox
jump 1
键盘输入


\\Mac\Home\Desktop\程设大作业\补充功能版\ARM64\Debug\补充功能版.exe
第2关:
提供数字: 3 9 5 1 -2 -2 9 -9
目标: -6 6 4 -4 0 0 18 -18
可用指令集: inbox,outbox,copyto,copyfrom,add,sub,jump,jumpifzero
可用空地数: 3
请选择读取指令的方式——1: 键盘输入 2: 文件输入
2
请输入文件路径
level2.txt
文件输入



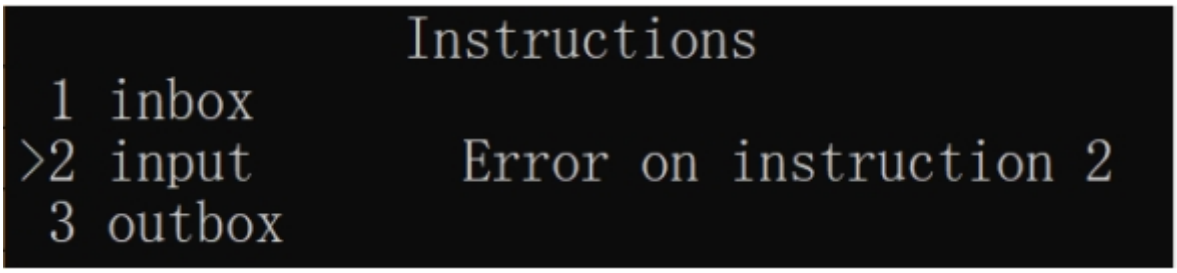

错误文件路径

**3、关卡运行显示（以第二关为例）**

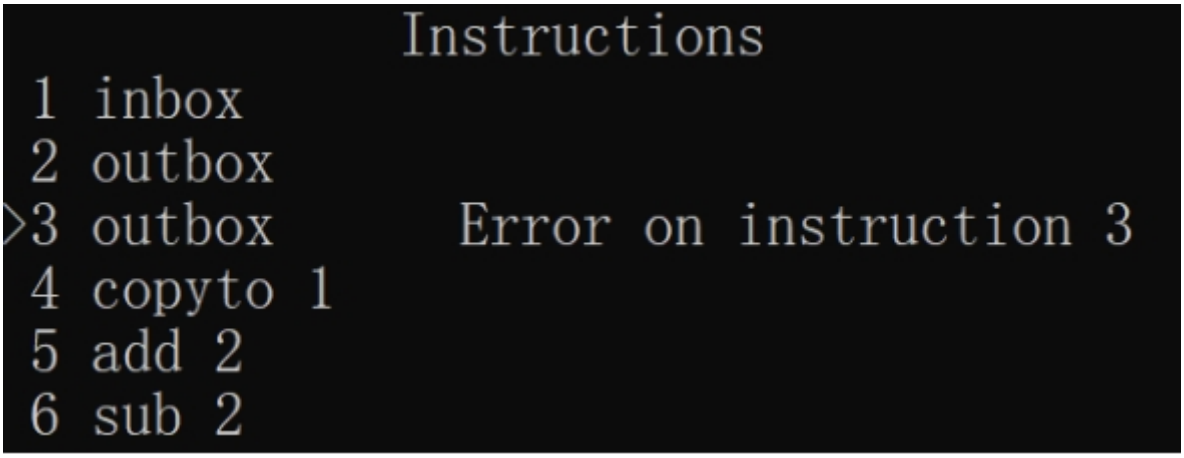



运行中

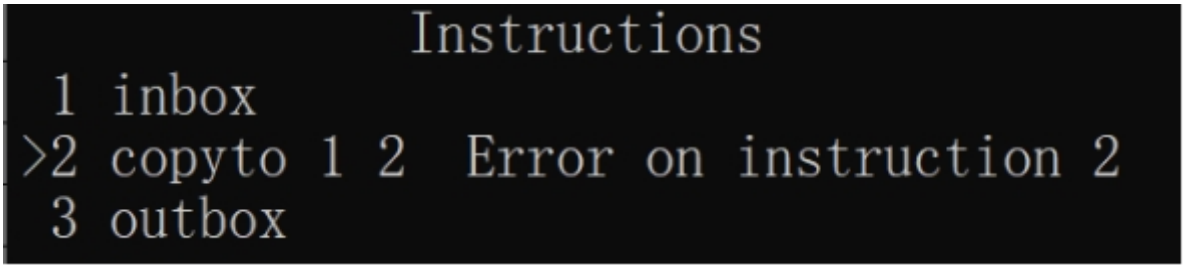
指令输入完成后，点击回车键进入关卡运行界面。机器人会按照玩家给出的指令运动并完成各项操作。如果检测到无效指令（不在指令集中、不符合该指令运行条件、操作数数量和要求不符、操作数超出范围、操作数不是整数等），则机器人停止运行，在无效指令旁边显示“Error on instruction X”（X是无效指令的编号），关卡结束。



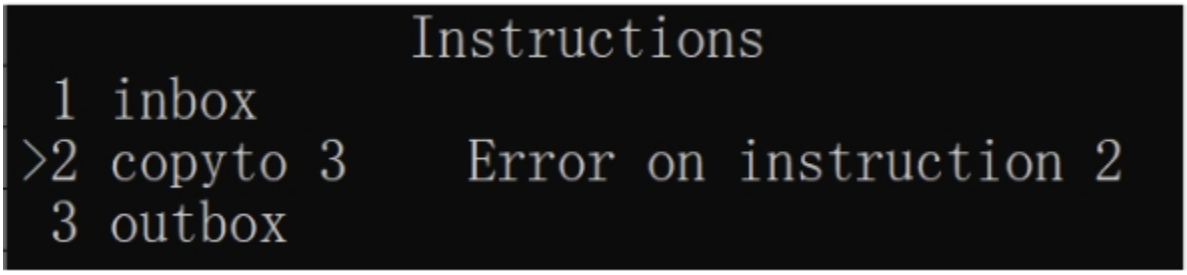

不在指令集中



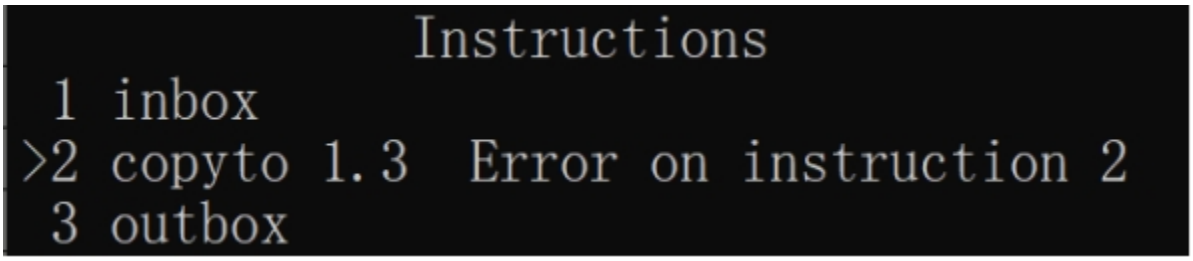
运行异常



多个操作数



操作数超限

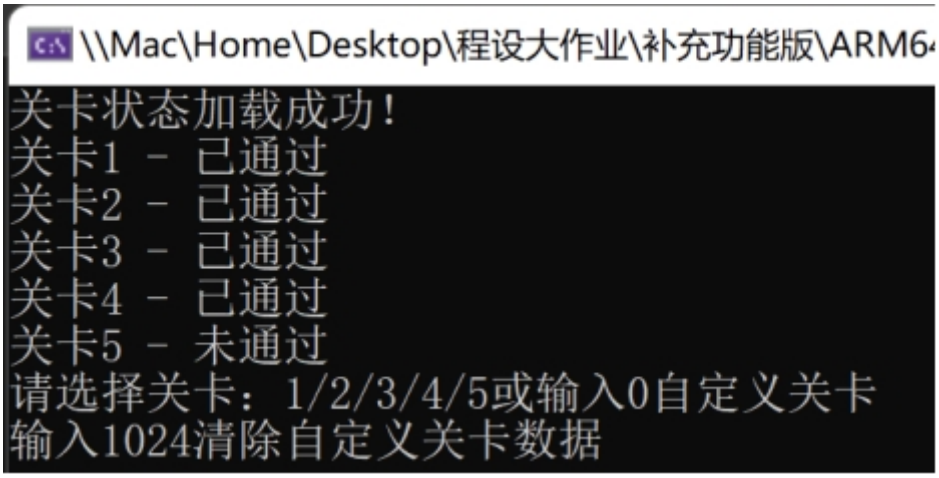


非整操作数

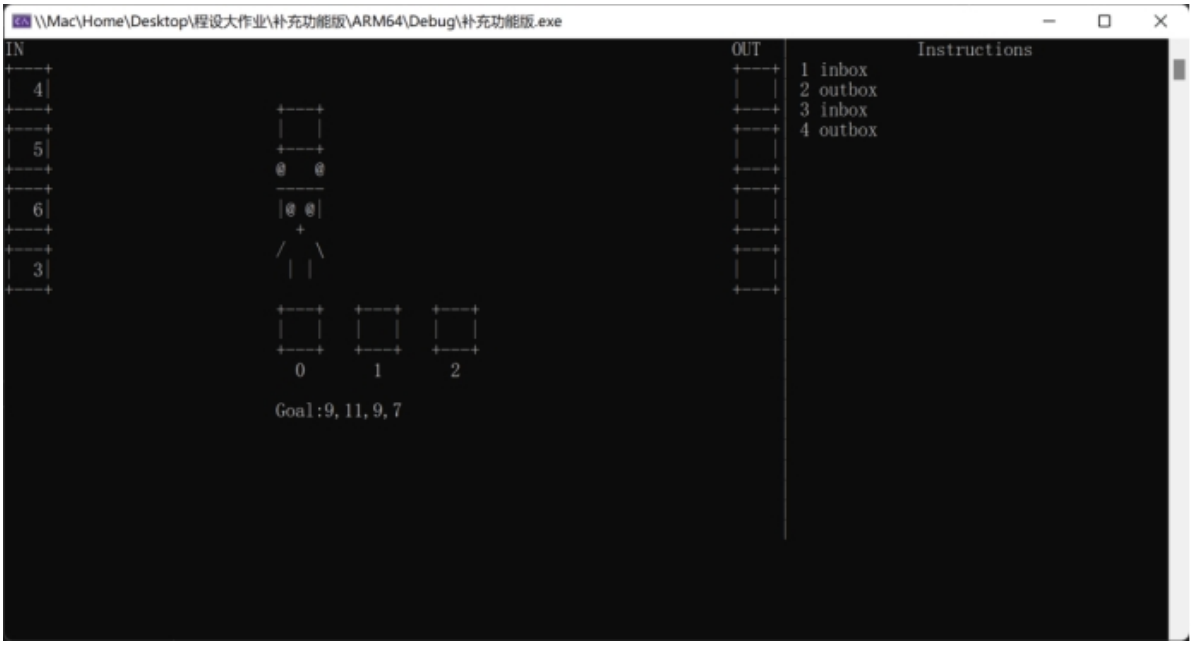
若没有出现异常指令，则当指令执行完成，或者执行“inbox”指令时输入传送带上没有积木，则关卡结束，系统判定运行结果。如果运行正确，则输出“Success”，否则输出“Fail”。只要关卡正常完成，都会在左下角输出实际执行的指令数。此时按1即可回到选关界面。


**4、新增关卡操作&清除关卡数据操作**

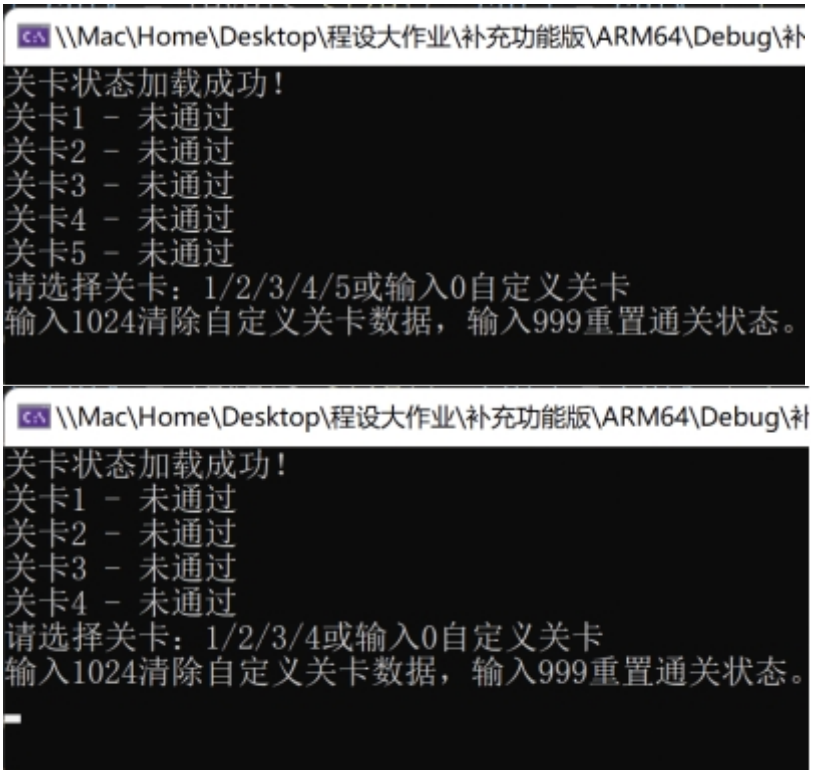
在选关界面输入0，即可进入自定义关卡界面。在此界面中按照指示，依次输入输入传送带上的积木数目及提供的数字、输出传送带上的积木数及目标数字、可用的空地数（最多八个）、可用的指令数、可用指令集，全部输入完成后按1返回选关界面，此时可以看到新增的关卡（默认状态是“未通过”）。




新增关卡的运行方法与前四个关卡相同。选择新增关卡，进入准备界面，按要求输入指令，即可进入关卡运行界面。


在选关界面输入1024，即可清除所有新增的自定义关卡，只保留四个基本关卡。输入999，即可清除所有关卡的通关数据，将各关卡设置为“未通过”。




输入999后的结果　　输入1024后的结果

## 五、自由创新关卡：

在三个固定关卡之外，第四关是创新关卡。下面介绍这一关卡的配置文件、输入、输出要求及一种可行思路。


第四关的配置文件内容为：

8

9 6 7 1 5 0 -2 4

4

16 11 5 -2

3

5

inbox outbox copyto copyfrom add

即：输入序列为：9，6，7，1，5，0，-2，4；输出序列为：16，11，5，-2。可用空地数为3，除sub，jump，jumpifzero外其余指令可用。

参考思路：通过观察可得，16=9+7，11=6+5，5=4+1，-2=0+（-2），即输出序列的第k个数是输入序列中第2k-1和第2k大的数之和。此关卡的难点在于，输入序列中的数字不是按大小顺序排列的，因此利用空地，将这些数字按照合适的先后顺序相加，具有一定技巧性。

一种解法为：inbox，copyto 0，inbox，copyto 1，inbox，add 0，outbox，inbox，copyto 2，inbox，add 1，outbox，inbox，copyto 1，inbox，copyto 0，inbox，add 2，outbox，copyfrom 0，outbox。