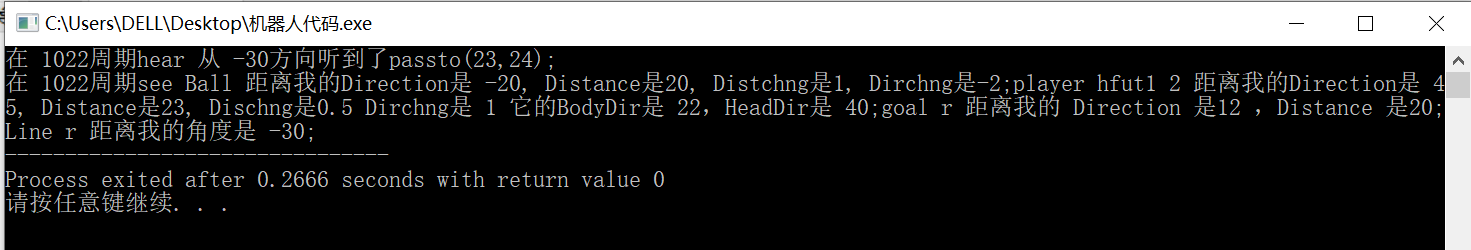
作者：计科19-4 曾宇杰

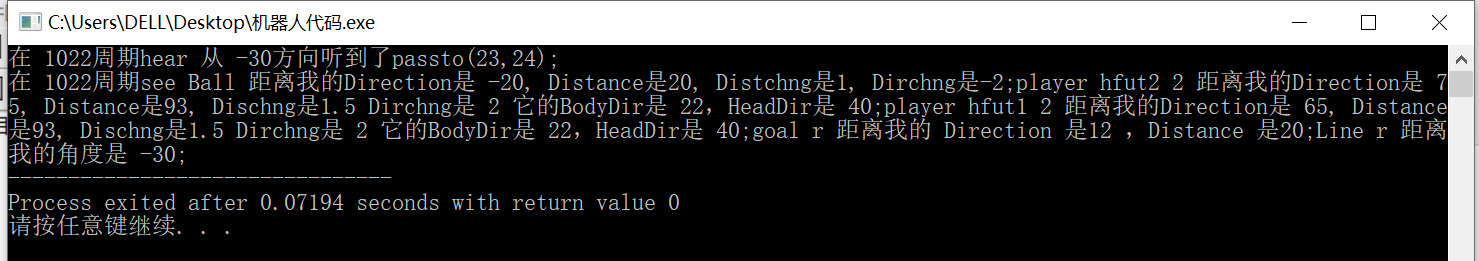
学号：2019218164

运行结果截图：

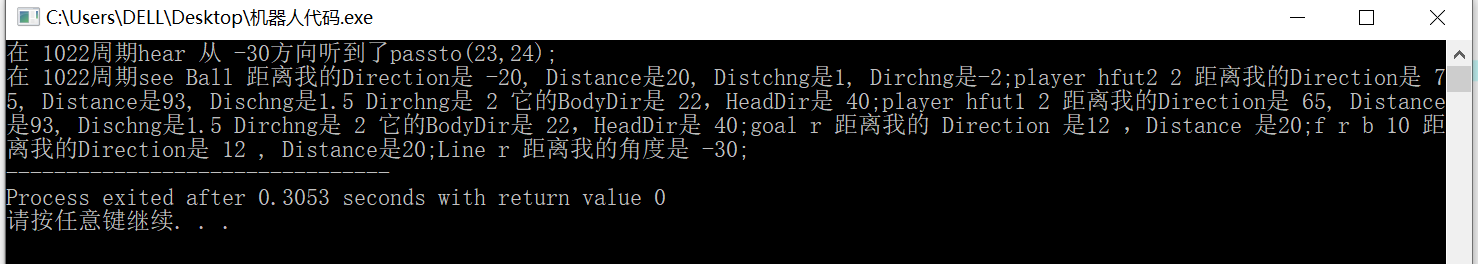
输入：(hear 1022 -30 passto(23,24))(see 1022 ((ball) -20 20 1 -2) ((player hfut1 2) 45 23 0.5 1 22 40 ) ((goal r) 12 20) ((Line r) -30))



输入：(hear 1022 -30 passto(23,24))(see 1022 ((ball) -20 20 1 -2) ((player hfut2 2) 75 93 1.5 2 22 40 ) ((player hfut1 2) 65 93 1.5 2 22 40 ) ((goal r) 12 20) ((Line r) -30))



输入：(hear 1022 -30 passto(23,24))(see 1022 ((ball) -20 20 1 -2) ((player hfut2 2) 75 93 1.5 2 22 40 ) ((player hfut1 2) 65 93 1.5 2 22 40 ) ((goal r) 12 20) ((f r b 10) 12 20) ((Line r) -30))



源代码：

#ifndef FIRST\_HOMEWORK\_OF\_ROBOCUP

#define FIRST\_HOMEWORK\_OF\_ROBOCUP

#include <iostream>

#include <string>

#include <vector>

#endif // !FIRST\_HOMEWORK\_OF\_ROBOCUP

using namespace std;

class message\_manager

{

private:

    string messages;                   //消息原始字符串

    string messages\_see;

    string messages\_hear;

    vector<string> HEAR;              //经过处理后的消息分段存入vector中

    vector<string> SEE;

    string time;                      //周期

    string sender;                    //听到的信息

    string message;

    vector<string> ball;              //看到的信息

    vector<string> flag;

    vector<string> Line;

    vector<string> goal;

    vector<string> teammates;

    vector<string> opponent;

public:

    message\_manager(string message);

    ~message\_manager();

    void message\_print();

};

/\*\*

 \* description : 将原始消息切割分类存入小数组并且实现格式化输出

 \*/

void message\_manager::message\_print()

{

    cout << "在 " << time << "周期" << \*(HEAR.begin()) << " 从 " << sender << "方向听到了" << message << ";" << endl;

    cout << "在 " << time << "周期" << \*(SEE.begin());

    vector<string>::iterator itt = SEE.begin()+2;

    while (itt != SEE.end())

    {

        vector<string> container;

        int first = (\*itt).find('(')+2;

        int end = (\*itt).find(')');

        string message\_name = (\*itt).substr(first, end - first);

        container.push\_back(message\_name);

        int index = end+1;

        while (index < (\*itt).size())                              //现将原始消息切割成小段方便但会输出

        {

            string temp = "";

            while ((\*itt)[index] != ' ' && (\*itt)[index] != '(' && (\*itt)[index] != ')' && index < (\*itt).size())

            {

                temp.append((\*itt).substr(index, 1));

                index++;

            }

            if (temp != "")

            {

                container.push\_back(temp);

            }

            index++;

        }

        if ((\*itt).find("ball") != (\*itt).npos)                 //将SEE到的信息存储到对应数组并且格式化输出

        {

            ball.push\_back(\*itt);                               //储存原始消息字符串到相应数组

            cout << " Ball 距离我的Direction是 " << container[1] << ", Distance是" << container[2]

                    << ", Distchng是" << container[3] << ", Dirchng是" << container[4] << ";";

        }

        else if ((\*itt).find("player") != (\*itt).npos)

        {

            if ((\*itt).find("hfut1") != (\*itt).npos)

            {

                teammates.push\_back(\*itt);

            }

            else if ((\*itt).find("hfut2") != (\*itt).npos)

            {

                opponent.push\_back((\*itt));

            }

            cout << container[0] << " 距离我的Direction是 " << container[1] << ", Distance是"

                    << container[2] << ", Dischng是" << container[3] << " Dirchng是 " << container[4]

                            << " 它的BodyDir是 " << container[5] << "，HeadDir是 " << container[6] << ";";

        }

        else if ((\*itt).find("goal") != (\*itt).npos)

        {

            goal.push\_back((\*itt));

            cout << container[0] << " 距离我的 Direction 是" << container[1] << " ，Distance 是"

                << container[2] << ";";

        }

        else if ((\*itt).find("f ") != (\*itt).npos)

        {

            flag.push\_back((\*itt));

            cout << container[0] << " 距离我的Direction是 " << container[1] << " , Distance是"

                    << container[2] << ";";

        }

        else if ((\*itt).find("Line") != (\*itt).npos)

        {

            Line.push\_back((\*itt));

            cout << container[0] << " 距离我的角度是 " << container[1] << ";";

        }

        ++itt;

    }

}

message\_manager::message\_manager(string original\_message)

{

    original\_message.replace(original\_message.size()-1, 1, " ");

    messages.append(original\_message.replace(0, 1, " "));

    messages.replace(messages.find("see")-2, 2, " ");                      //去左右两个括号

    //将字符串切割存到对应变量

    messages\_hear.append(messages,messages.find("hear"), messages.find("see") - messages.find("hear"));

    messages\_see.append(messages, messages.find("see"), messages.size() - messages.find("see"));

    int index = 0;

    int flag = 0;

    while (index < messages\_hear.size())            //将听到的信息分割

    {

        string temp = "";

        while (((messages\_hear[index] != ' ') && (index < messages\_hear.size())) || (flag != 0))

        {

            if (messages\_hear[index] == '(')

            {

                ++flag;

            }

            if (messages\_hear[index] == ')')

            {

                --flag;

            }

            temp.append(messages\_hear,index,1);

            ++index;

        }

        ++index;

        if (temp != "")

        {

            HEAR.push\_back(temp);

        }

    }

    index = 0;

    flag = 0;

    while (index < messages\_see.size())                        //将看到的信息分割

    {

        string temp = "";

        while (((messages\_see[index] != ' ') && (index < messages\_see.size())) || (flag != 0))

        {

            if (messages\_see[index] == '(')

            {

                ++flag;

            }

            if (messages\_see[index] == ')')

            {

                --flag;

            }

            temp.append(messages\_see,index,1);

            ++index;

        }

        ++index;

        if (temp != "")

        {

            SEE.push\_back(temp);

        }

    }

    vector<string>::iterator itt = HEAR.begin();             //将HEAR到对应信息存储在对应成员变量里面

    while (itt != HEAR.end())

    {

        switch (itt - HEAR.begin()-1)

        {

        case 0:

            time.append((\*itt));

            break;

        case 1:

            sender.append((\*itt));

            break;

        case 2:

            message.append((\*itt));

            break;

        }

        ++itt;

    }

}

message\_manager::~message\_manager()

{

}

int main()

{

    string s = "(hear 1022 -30 passto(23,24))(see 1022 ((ball) -20 20 1 -2) ((player hfut1 2) 45 23 0.5 1 22 40 )

                        ((goal r) 12 20) ((Line r) -30))";

    message\_manager s1(s);

    s1.message\_print();

    return 0;

}