Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра МОП ЭВМ

Направление 09.03.01 – «Информационные системы и технологий»

Расчетно-графическое задание по дисциплине «Логическое программирование»

Студент группы 5ИСБ-1 Д.Д. Азизов

Преподаватель Е.Б. Абарникова

2017

**Тема:** Методы поиска решения в пространстве состояний.

**Цель:** Изучить методы поиска решения в пространстве состояний.

Задание: Существует план лабиринта, в котором из 100комнат только 8 заперты, а все остальные открыты настежь. Открыть запертую комнату можно в том случае, если она будет точно девятой по счету, то есть если перед этим будет пройдено 8 открытых дверей. При этом должны быть открыты и пройдены все запертые комнаты, каждая из них открывается сама, если перед этим пройдено ровно 8 открытых комнат. Как только какаянибудь комната пройдена, то вернуться в неё уже нельзя. Найти путь в лабиринте. Вход в лабиринт находится в клетке с координатами 10,10.

**Содержание**

**1 Оглавление**

2 Теоретическое описание ......................................................................... 4

2.1 Типичные операции: ....................................................................... 4

2.2 Стратегия поиска в глубину: .......................................................... 5

2.3 Стратегия поиска в ширину: .......................................................... 5

2.4 Граф состояний для данной программы ....................................... 6

3 Описание программы: ............................................................................. 7

4 Текст программы ..................................................................................... 9

5 Программа и методика испытаний ...................................................... 23

5.1 Объект испытаний ......................................................................... 23

5.2 Цель испытаний ............................................................................. 23

6 Средства и порядок испытаний............................................................ 23

6.1 Технические средства, используемые во время испытаний ........ 23

6.2 Программные средства, используемые во время испытаний ….. 24

7 Методы испытаний ............................................................................... 24

7.1 Проверка на корректность ........................................................... 24

7.2 Проверка на корректную обработку выбора решения .............. 25

7.3 Проверка на правильность ........................................................... 25

7.4 Проверка на надежность ............................................................... 25

8 Эскизы экранных форм ......................................................................... 27

Список использованных источников ...................................................... 29

**2 Теоретическое описание**

Граф – множество вершин вместе с множеством ребер. Каждое ребро задается парой вершин. Если ребра направлены, то называются дугами. Дуги задаются упорядоченными парами. Такие графы называются направленными. Ребрам можно приписывать имена или метки, в зависимости от конкретного приложения.

Каждое ребро – отдельное предложение:

link ( a, b ).

link ( b, c ).

Весь граф – как один объект, состоящий из двух множеств: множество вершин и множество ребер ( дуг ). Каждое множество - список

G = graf ( [ a, b, c, d, e ], [ p( a, b ), p( b, c ),…,p( d, e ) ] ).

Для направленного графа:

G1= graf ( [ a | \_ ], [ d( a, b, 1 ), d( b, c, 4 ) …] )

**2.1 Типичные операции:**

а) Найти путь между двумя заданными вершинами.

б) Найти подграф, обладающий некоторыми заданными свойствами.

Пространство состояний – некоторое множество уникальных состояний системы, некоторые из которых связаны между собой. Если два состояния связаны, это значит, что возможен переход системы из одного состояния в другое. Пространство состояний может быть представлено в виде графа, в котором узлами являются состояния системы, а рёбра –

5

возможными переходами. Рёбра могут быть направленными. Это значит, что возможен переход системы только от первого связанного состояния ко второму, но не наоборот. Поиск в пространстве состояний – это поиск пути в этом графе от начального состояния к конечному.

2.2 Стратегия поиска в глубину:

«В глубину» - это порядок, в котором рассматриваются альтернативы в пространстве состояний. Всегда, когда алгоритму поиска в глубину надлежит выбрать из нескольких вершин ту, в которую следует перейти для продолжения поиска, он предпочитает самую «глубокую» из них. Самая глубокая вершина – это вершина, расположенная дальше других от стартовой вершины (см. Рисунок 1).

2.3 Стратегия поиска в ширину:

В противоположность поиску в глубину стратегия поиска в ширину предусматривает переход в первую очередь к вершинам, ближайшим к стартовой вершине. В результате процесс поиска имеет тенденцию развиваться более в ширину, чем в глубину (см. Рисунок 2). Поиск в ширину программируется не так легко, как поиск в глубину. Причина состоит в том, что нужно сохранить все множество альтернативных вершин-кандидатов, а не только одну вершину, как при поиске в глубину.

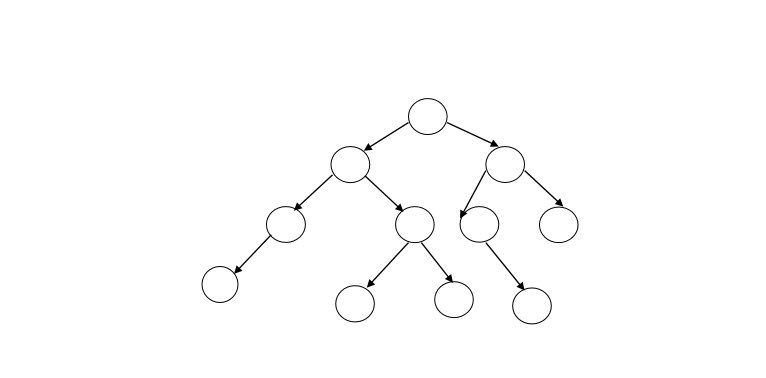


Рисунок 1 . Стратегия поиска в ширину

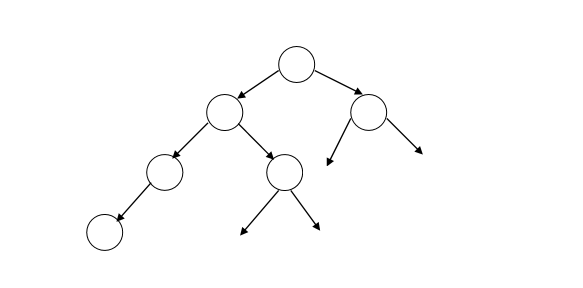


Рисунок 2. Стратегия поиска в глубину

**2.4 Граф состояний для данной программы**

В данном алгоритме используется метод «Образовать – Проверить». Начальным состоянием программы является «Нет пути» - S1. Конечными состояниями «Есть путь» - Y1. Состояния S2, S3 описывают нахождение в

определённой клетке. Ниже на рисунке 3 показано граф состояний для данной задачи. Условиями перехода в каждое состояние являются:

Если комната не была ранее посещена

Если комната закрыта, то проверить, сколько перед ней было посещено открытых комнат. Если 8, то условия перехода выполняется

Координаты комнаты должны ограничиваться координатами (1,1) – (10,10).

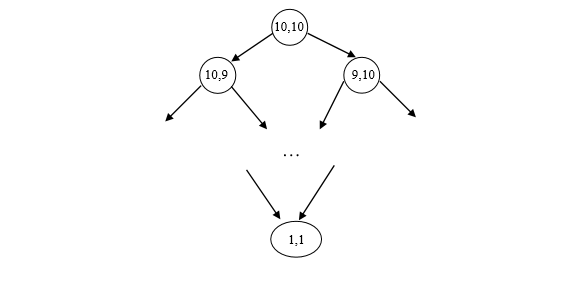


Рисунок3. Граф состояний

**3 Описание программы:**

Пространством состояний в данной задаче является совокупность всевозможных комбинаций обходов лабиринта, размером 10x10.

В данной программе используется метод обхода графа в глубину. Первоначально происходит проверка на заданные условия. Если условие выполняется, то пройденная вершина записывается в очередь. Иначе происходит откат назад. Данный алгоритм является довольно продуктивным,

но при неявном пути обхода требует некоторое количество времени для его реализации.

Стартовая вершина – клетка лабиринта с координатами (10,10). Выход из лабиринта находится в клетке (1,1).

Пользователь вводит координаты закрытых комнат в поля диалогового окна, затем нажимает на кнопку «Нахождение пути». Если путь будет найден, то в окно диалога будет выведено графическое изображение лабиринта с комнатами, которые будут пронумерованы в порядке их обхода. Закрытые комнаты отображаются чёрными квадратами. В противном случае пользователю будет выведено сообщение о некорректно введённых данных, или отсутствии пути.

В программе определены следующие предикаты:

finish(i,j) - предикат проверки пришли ли мы на финиш

nondetermmember(a,m)- предикат проверки были ли мы ранее в клетке

nondetermway(a,m,m,integer,integer) - предикат формирования пути

nondetermall(m) – предикат запуска формирования пути

квадрат (window,i,i) - предикат рисования квадратов для закрытых дверей

цифры(window,m,i) – предикат отображения последовательности обхода комнат

**4 Текст программы**

include "lab1.inc"

include "lab1.con"

include "hlptopic.con"

%BEGIN\_WIN Task Window

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Event handling for Task Window

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

GLOBAL DOMAINS

m=a\*

a=a(i,j)

j=integer

i=integer

GLOBAL DATABASE

wall(i,j)

predicates

%wall(i,j) %Есть стена

finish(i,j) %Финиш

nondetermmember(a,m) %Проверяем были мы в данной клетке ранее или нет

nondetermway(a,m,m,integer,integer) %3 аргумент - сколько было пройдено клеток, 4-кол-во открытых дверей

nondeterm all(m)

квадрат (window,i,i) %вывод квадрата

цифры(window,m,i)

10

clauses

%Конец пути

finish(1,1).

%Вход в лабиринт

all(P):-way(a(10,10),P,[a(10,10)],1,0),!. %Начальная координата 10,10. Была пройдена 1 клетка (ступили на 10,10). Открыто 0 дверей. Отсечение выводит 1 вариант

%Если финиш

way(a(I,J),[a(I,J)],\_,\_,9):- finish(I,J). %Должно быть открыто 9 дверей перед тем как можно будет выйти

%Идём вверх

%Если двери нет и не были в данной клетке, то путь есть

way(a(I,J),[a(I,J)|P],Was,KolOpen,KolDoor):-

K = I-1,

not(wall(K,J)), %Если нет стены

K>=1, K<=10, %Проверка

not(member(a(K,J), Was)), %Если не были в данной комнате

KolOpen2=KolOpen+1, %Количество пройденных дверей увеличиваем на 1

way(a(K,J),P,[a(K,J)|Was],KolOpen2,KolDoor). %Идём дальше

%Если есть дверь, не были в данной клетке и было пройдено 8 клеток ранее, то путь через стену есть

way(a(I,J),[a(I,J)|P],Was,8,KolDoor):-

K = I-1,

wall(K,J), %Если стена

not(member(a(K,J), Was)), %Если не были в данной комнате

KolOpen2=0, %Количество пройденных дверей обнуляем, т.к. прошли через дверь

KolDoor2=KolDoor+1, %Количество открытых дверей увеличивается

way(a(K,J),P,[a(K,J)|Was],KolOpen2,KolDoor2). %Идём дальше

11

%Идём вниз

%Если двери нет и не были в данной клетке, то путь есть

way(a(I,J),[a(I,J)|P],Was,KolOpen,KolDoor) :-

K = I+1,

not(wall(K,J)), %Если нет стены

K>=1, K<=10, %Проверка

not(member(a(K,J), Was)), %Если не были в данной комнате

KolOpen2=KolOpen+1, %Количество пройденных дверей увеличиваем на 1

way(a(K,J),P,[a(K,J)|Was],KolOpen2,KolDoor). %Идём дальше

%Если есть дверь, не были в данной клетке и было пройдено 8 клеток ранее, то путь через стену есть

way(a(I,J),[a(I,J)|P],Was,8,KolDoor) :-

K = I+1,

wall(K,J), %Если стена

not(member(a(K,J), Was)), %Если не были в данной комнате

KolOpen2=0, %Количество пройденных дверей обнуляем, т.к. прошли через дверь

KolDoor2=KolDoor+1, %Количество открытых дверей увеличивается

way(a(K,J),P,[a(K,J)|Was],KolOpen2,KolDoor2). %Идём дальше

%Идём влево

%Если двери нет и не были в данной клетке, то путь есть

way(a(I,J),[a(I,J)|P],Was,KolOpen,KolDoor) :-

L = J-1,

not(wall(I,L)), %Если нет стены

L>=1, L<=10, %Проверка

not(member(a(I,L), Was)), %Если не были в данной комнате

KolOpen2=KolOpen+1, %Количество пройденных дверей увеличиваем на 1

way(a(I,L),P,[a(I,L)|Was],KolOpen2,KolDoor). %Идём дальше

12

%Если есть дверь, не были в данной клетке и было пройдено 8 клеток ранее, то путь через стену

есть

way(a(I,J),[a(I,J)|P],Was,8,KolDoor) :-

L = J-1,

wall(I,L), %Если стена

not(member(a(I,L), Was)), %Если не были в данной комнате

KolOpen2=0, %Количество пройденных дверей обнуляем, т.к. прошли через дверь

KolDoor2=KolDoor+1, %Количество открытых дверей увеличивается

way(a(I,L),P,[a(I,L)|Was],KolOpen2,KolDoor2). %Идём дальше

%Идём вправо

%Если двери нет и не были в данной клетке, то путь есть

way(a(I,J),[a(I,J)|P],Was,KolOpen,KolDoor) :-

L = J+1,

not(wall(I,L)), %Если нет стены

L>=1, L<=10, %Проверка

not(member(a(I,L), Was)), %Если не были в данной комнате

KolOpen2=KolOpen+1, %Количество пройденных дверей увеличиваем на 1

way(a(I,L),P,[a(I,L)|Was],KolOpen2,KolDoor). %Идём дальше

%Если есть дверь, не были в данной клетке и было пройдено 8 клеток ранее, то путь через стену

есть

way(a(I,J),[a(I,J)|P],Was,8,KolDoor) :-

L = J+1,

wall(I,L), %Если стена

not(member(a(I,L), Was)), %Если не были в данной комнате

KolOpen2=0, %Количество пройденных дверей обнуляем, т.к. прошли через дверь

KolDoor2=KolDoor+1, %Количество открытых дверей увеличивается

way(a(I,L),P,[a(I,L)|Was],KolOpen2,KolDoor2). %Идём дальше

13

%Проверяем были мы в данной клетке ранее или нет

member(I,[I|\_]).

member(I,[\_|P]):-member(I,P).

predicates

task\_win\_eh : EHANDLER

constants

%BEGIN Task Window, CreateParms, 20:14:02-11.3.2016, Code automatically updated!

task\_win\_Flags = [wsf\_SizeBorder,wsf\_TitleBar,wsf\_Close,wsf\_Maximize,wsf\_Minimize,wsf\_ClipSiblings]

task\_win\_Menu = res\_menu(idr\_task\_menu)

task\_win\_Title = "Lab1"

task\_win\_Help = idh\_contents

%END Task Window, CreateParms

clauses

task\_win\_eh(\_Win,e\_Create(\_),0):-!,

dlg\_ввод\_и\_вывод\_информации\_Create (\_Win),

win\_Destroy (\_Win),

!.

task\_win\_eh(\_Win,e\_Menu(id\_help\_contents,\_ShiftCtlAlt),0):-!,

vpi\_ShowHelp("lab1.hlp"),

!.

task\_win\_eh(Win,e\_Menu(id\_help\_about,\_ShiftCtlAlt),0):-!,

14

dlg\_about\_dialog\_Create(Win),

!.

task\_win\_eh(Win,e\_Menu(id\_file\_exit,\_ShiftCtlAlt),0):-!,

win\_Destroy(Win),

!.

task\_win\_eh(\_Win,e\_Size(\_Width,\_Height),0):-!,

ifdef use\_tbar

toolbar\_Resize(\_Win),

enddef

ifdef use\_message

msg\_Resize(\_Win),

enddef

!.

%END Task Window, e\_Size

%END\_WIN Task Window

project\_ShowHelpContext(HelpTopic):-

vpi\_ShowHelpContext("lab1.hlp",HelpTopic).

goal

ifdef use\_mdi

vpi\_SetAttrVal(attr\_win\_mdi,b\_true),

enddef

ifdef ws\_win

ifdef use\_3dctrl

vpi\_SetAttrVal(attr\_win\_3dcontrols,b\_true),

enddef

enddef

15

vpi\_Init(task\_win\_Flags,task\_win\_eh,task\_win\_Menu,"lab1",task\_win\_Title).

constants

%BEGIN About dialog, CreateParms, 20:14:02-11.3.2016, Code automatically updated!

dlg\_about\_dialog\_ResID = idd\_dlg\_about

dlg\_about\_dialog\_DlgType = wd\_Modal

dlg\_about\_dialog\_Help = idh\_contents

%END About dialog, CreateParms

predicates

dlg\_about\_dialog\_eh : EHANDLER

clauses

dlg\_about\_dialog\_Create(Parent):-

win\_CreateResDialog(Parent,dlg\_about\_dialog\_DlgType,dlg\_about\_dialog\_ResID,dlg\_about\_dial og\_eh,0).

%BEGIN About dialog, idc\_ok \_CtlInfo

dlg\_about\_dialog\_eh(\_Win,e\_Control(idc\_ok,\_CtrlType,\_CtrlWin,\_CtrlInfo),0):-!,

win\_Destroy(\_Win),

!.

%END About dialog, idc\_ok \_CtlInfo

%MARK About dialog, new events

dlg\_about\_dialog\_eh(\_,\_,\_):-!,fail.

constants

16

dlg\_ввод\_и\_вывод\_информации\_ResID = idd\_ввод\_и\_вывод\_информации

dlg\_ввод\_и\_вывод\_информации\_DlgType = wd\_Modal

dlg\_ввод\_и\_вывод\_информации\_Help = idh\_contents

predicates

dlg\_ввод\_и\_вывод\_информации\_eh : EHANDLER

dlg\_ввод\_и\_вывод\_информации\_handle\_answer(INTEGER EndButton,DIALOG\_VAL\_LIST)

dlg\_ввод\_и\_вывод\_информации\_update(DIALOG\_VAL\_LIST)

clauses

dlg\_ввод\_и\_вывод\_информации\_Create(Parent):-

%MARK Ввод и вывод информации, new variables

dialog\_CreateModal(Parent,dlg\_ввод\_и\_вывод\_информации\_ResID,"",

[

df(idc\_текст1,editstr("",[]),nopr),

df(idc\_текст3,editstr("",[]),nopr),

df(idc\_текст5,editstr("",[]),nopr),

df(idc\_текст7,editstr("",[]),nopr),

df(idc\_текст9,editstr("",[]),nopr),

df(idc\_текст11,editstr("",[]),nopr),

df(idc\_текст13,editstr("",[]),nopr),

df(idc\_текст15,editstr("",[]),nopr),

df(idc\_текст16,editstr("",[]),nopr),

df(idc\_текст14,editstr("",[]),nopr),

df(idc\_текст12,editstr("",[]),nopr),

df(idc\_текст10,editstr("",[]),nopr),

17

df(idc\_текст8,editstr("",[]),nopr),

df(idc\_текст6,editstr("",[]),nopr),

df(idc\_текст4,editstr("",[]),nopr),

df(idc\_текст2,editstr("",[]),nopr)

%END Ввод и вывод информации, ControlList

],

dlg\_ввод\_и\_вывод\_информации\_eh,0,VALLIST,ANSWER),

dlg\_ввод\_и\_вывод\_информации\_handle\_answer(ANSWER,VALLIST).

dlg\_ввод\_и\_вывод\_информации\_handle\_answer(idc\_ok,VALLIST):-!,

dlg\_ввод\_и\_вывод\_информации\_update(VALLIST).

dlg\_ввод\_и\_вывод\_информации\_handle\_answer(idc\_cancel,\_):-!. % Handle Esc and Cancel here

dlg\_ввод\_и\_вывод\_информации\_handle\_answer(\_,\_):-

errorexit().

dlg\_ввод\_и\_вывод\_информации\_update(\_VALLIST):-

%BEGIN Ввод и вывод информации, Update controls, 20:50:02-3.6.2016, Code automatically updated!

\_IDC\_ТЕКСТ1\_VALUE = dialog\_VLGetstr(idc\_текст1,\_VALLIST),

\_IDC\_ТЕКСТ3\_VALUE = dialog\_VLGetstr(idc\_текст3,\_VALLIST),

\_IDC\_ТЕКСТ5\_VALUE = dialog\_VLGetstr(idc\_текст5,\_VALLIST),

\_IDC\_ТЕКСТ7\_VALUE = dialog\_VLGetstr(idc\_текст7,\_VALLIST),

\_IDC\_ТЕКСТ9\_VALUE = dialog\_VLGetstr(idc\_текст9,\_VALLIST),

\_IDC\_ТЕКСТ11\_VALUE = dialog\_VLGetstr(idc\_текст11,\_VALLIST),

\_IDC\_ТЕКСТ13\_VALUE = dialog\_VLGetstr(idc\_текст13,\_VALLIST),

\_IDC\_ТЕКСТ15\_VALUE = dialog\_VLGetstr(idc\_текст15,\_VALLIST),

\_IDC\_ТЕКСТ16\_VALUE = dialog\_VLGetstr(idc\_текст16,\_VALLIST),

\_IDC\_ТЕКСТ17\_VALUE = dialog\_VLGetstr(idc\_текст17,\_VALLIST),

\_IDC\_ТЕКСТ18\_VALUE = dialog\_VLGetstr(idc\_текст18,\_VALLIST),

\_IDC\_ТЕКСТ14\_VALUE = dialog\_VLGetstr(idc\_текст14,\_VALLIST),

\_IDC\_ТЕКСТ12\_VALUE = dialog\_VLGetstr(idc\_текст12,\_VALLIST),

18

\_IDC\_ТЕКСТ10\_VALUE = dialog\_VLGetstr(idc\_текст10,\_VALLIST),

\_IDC\_ТЕКСТ8\_VALUE = dialog\_VLGetstr(idc\_текст8,\_VALLIST),

\_IDC\_ТЕКСТ6\_VALUE = dialog\_VLGetstr(idc\_текст6,\_VALLIST),

\_IDC\_ТЕКСТ4\_VALUE = dialog\_VLGetstr(idc\_текст4,\_VALLIST),

\_IDC\_ТЕКСТ2\_VALUE = dialog\_VLGetstr(idc\_текст2,\_VALLIST),

%END Ввод и вывод информации, Updatecontrols

true.

%MARK Ввод и вывод информации, newevents

%BEGIN Ввод и вывод информации, idc\_ввод \_CtlInfo

dlg\_ввод\_и\_вывод\_информации\_eh(\_Win,e\_Control(idc\_ввод,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-

%Вывод картинки

Picture = pict\_GetFromRes (idb\_s),

pict\_Draw(\_Win,Picture,pnt(20,20),rop\_SrcCopy),

Текст1 = win\_GetCtlHandle (\_Win,idc\_текст1), Текст2 = win\_GetCtlHandle (\_Win,idc\_текст2),

Текст3 = win\_GetCtlHandle (\_Win,idc\_текст3), Текст4 = win\_GetCtlHandle (\_Win,idc\_текст4),

Текст5 = win\_GetCtlHandle (\_Win,idc\_текст5), Текст6 = win\_GetCtlHandle (\_Win,idc\_текст6),

Текст7 = win\_GetCtlHandle (\_Win,idc\_текст7), Текст8 = win\_GetCtlHandle (\_Win,idc\_текст8),

Текст9 = win\_GetCtlHandle (\_Win,idc\_текст9), Текст10 = win\_GetCtlHandle (\_Win,idc\_текст10),

Текст11 = win\_GetCtlHandle (\_Win,idc\_текст11), Текст12 = win\_GetCtlHandle (\_Win,idc\_текст12),

Текст13 = win\_GetCtlHandle (\_Win,idc\_текст13), Текст14 = win\_GetCtlHandle (\_Win,idc\_текст14),

Текст15 = win\_GetCtlHandle (\_Win,idc\_текст15), Текст16 = win\_GetCtlHandle (\_Win,idc\_текст16),

19

Текст17 = win\_GetCtlHandle (\_Win,idc\_текст17), Текст18 = win\_GetCtlHandle (\_Win,idc\_текст18),

ТекстНовый1 = win\_GetText (Текст1), ТекстНовый2 = win\_GetText (Текст2),

ТекстНовый3 = win\_GetText (Текст3), ТекстНовый4 = win\_GetText (Текст4),

ТекстНовый5 = win\_GetText (Текст5), ТекстНовый6 = win\_GetText (Текст6),

ТекстНовый7 = win\_GetText (Текст7), ТекстНовый8 = win\_GetText (Текст8),

ТекстНовый9 = win\_GetText (Текст9), ТекстНовый10 = win\_GetText (Текст10),

ТекстНовый11 = win\_GetText (Текст11), ТекстНовый12 = win\_GetText (Текст12),

ТекстНовый13 = win\_GetText (Текст13), ТекстНовый14 = win\_GetText (Текст14),

ТекстНовый15 = win\_GetText (Текст15), ТекстНовый16 = win\_GetText (Текст16),

ТекстНовый17 = win\_GetText (Текст17), ТекстНовый18 = win\_GetText (Текст18),

ТекстНовый1 <> "", ТекстНовый2 <> "", ТекстНовый3 <> "", ТекстНовый4 <>

"",

ТекстНовый5 <> "", ТекстНовый6 <> "", ТекстНовый7 <> "", ТекстНовый8 <>

"",

ТекстНовый9 <> "", ТекстНовый10 <> "", ТекстНовый11 <> "", ТекстНовый12 <>

"",

ТекстНовый13 <> "", ТекстНовый14 <> "", ТекстНовый15 <> "", ТекстНовый16 <>

"",

ТекстНовый17 <> "", ТекстНовый18 <> "",

%ПРОВЕРКИ НА ТО ЧТОБЫ БЫЛИ ВВЕДЕНЫ РАЗНЫЕ НАЧАЛЬНЫЕ ГРАНИЦЫ

str\_int(ТекстНовый1,Число1), str\_int(ТекстНовый2,Число2),

str\_int(ТекстНовый3,Число3), str\_int(ТекстНовый4,Число4),

str\_int(ТекстНовый5,Число5), str\_int(ТекстНовый6,Число6),

str\_int(ТекстНовый7,Число7), str\_int(ТекстНовый8,Число8),

str\_int(ТекстНовый9,Число9), str\_int(ТекстНовый10,Число10),

str\_int(ТекстНовый11,Число11), str\_int(ТекстНовый12,Число12),

str\_int(ТекстНовый13,Число13), str\_int(ТекстНовый14,Число14),

str\_int(ТекстНовый15,Число15), str\_int(ТекстНовый16,Число16),

20

str\_int(ТекстНовый17,Число17), str\_int(ТекстНовый18,Число18),

Число1>=1,Число1<=10,Число2>=1,Число2<=10,Число3>=1,Число3<=10,Число4>=1,Число4

<=10,

Число5>=1,Число5<=10,Число6>=1,Число6<=10,Число7>=1,Число7<=10,Число8>=1,Число8

<=10,

Число9>=1,Число9<=10,Число10>=1,Число10<=10,Число11>=1,Число11<=10,Число12>=1, Число12<=10,

Число13>=1,Число13<=10,Число14>=1,Число14<=10,Число15>=1,Число15<=10,Число16>= 1,Число16<=10,

квадрат(\_Win,Число1,Число2),

квадрат(\_Win,Число3,Число4),

квадрат(\_Win,Число5,Число6),

квадрат(\_Win,Число7,Число8),

квадрат(\_Win,Число9,Число10),

квадрат(\_Win,Число11,Число12),

квадрат(\_Win,Число13,Число14),

квадрат(\_Win,Число15,Число16),

квадрат(\_Win,Число17,Число18),

retractall(wall(\_,\_)),

assert(wall(Число1,Число2)),

assert(wall(Число3,Число4)),

assert(wall(Число5,Число6)),

assert(wall(Число7,Число8)),

assert(wall(Число9,Число10)),

assert(wall(Число11,Число12)),

assert(wall(Число13,Число14)),

assert(wall(Число15,Число16)),

21

assert(wall(Число17,Число18)),

all(P), !,

цифры(\_Win,P,0),

!.

dlg\_ввод\_и\_вывод\_информации\_eh(\_Win,e\_Control(idc\_ввод,\_CtrlType,\_CtrlWin,\_CtlInfo),0):-!,

dlg\_note ("Некорректный ввод данных или путь не найден!"),

!.

dlg\_ввод\_и\_вывод\_информации\_eh(\_,\_,\_):-!,fail.

%ВЫВОД КВАДРАТА

квадрат (\_Win,XX,YY):-

win\_SetBrush(\_Win,brush(pat\_Solid,color\_Black)),

%draw\_Rect (\_Win, rct(60,55,110,105)),

X=XX-1,

X1=X\*50,

X11=X1+58,

X22=X1+108,

Y=YY-1,

Y1=Y\*50,

Y11=Y1+52,

Y22=Y1+103,

draw\_Rect (\_Win, rct(Y11,X11,Y22,X22)),!.

цифры(\_Win,[],N).

цифры(\_Win,[a(XX,YY)|P],N):-

X=XX-1,

X1=X\*49,

X11=X1+73,

Y=YY-1,

Y1=Y\*49,

Y11=Y1+68,

N1=N+1,

str\_int(Text,N1),

draw\_Text(\_Win, Y11, X11, Text),

%вывод в листбокс

HandleListBox = win\_GetCtlHandle (\_Win,idc\_lb),

str\_int(I1,XX),

str\_int(J1,YY),

concat("Координата: (", I1, String0),

concat(String0, ",", String2),

concat(String2, J1, String3),

concat(String3, ")", String4),

Len = lbox\_CountAll(HandleListBox),%Длина

lbox\_add(HandleListBox, Len, String4),

цифры(\_Win,P,N1),!

**5 Программа и методика испытаний**

**5.1 Объект испытаний**

Испытуемая программа RGZ.exe предназначена для нахождения пути в лабиринте с условием прохождения всех закрытых комнат.

**5.2 Цель испытаний**

Испытание проводилось с целью выявления корректности выполнения программы; отсутствия ошибочных, либо неопределенных ситуаций, возникающих в результате выполнения программы; отсутствия сбоев систем компьютера в результате работы программы.

**6 Средства и порядок испытаний**

**6.1 Технические средства, используемые во время испытаний**

Во время испытаний использовались технические средства:

процессор Intel I7 с тактовой частотой 3200 MHz;

оперативная память 512 Mб;

видеокарта и монитор, поддерживающие разрешение 1024x768 точек и больше;

свободное дисковое пространство 1.9 Гб;

клавиатура;

24

манипулятор мышь.

**6.2 Программные средства, используемые во время испытаний**

Во время испытаний использовалось программное обеспечение:

операционнаясистемаMicrosoftWindows 7

VisualProlog 5.2

Испытание программы проводились в следующем порядке:

испытание на корректность (адекватно ли программа реагирует на выбор решения из списка);

испытание на правильность;

испытание на надежность (процент отказа системы).

Если итог проведения испытаний удовлетворяет заданным критериям, то проведение испытаний считается успешным.

**7 Методы испытаний**

**7.1 Проверка на корректность**

Порядок выполнения теста:

Запускаем на выполнение программный модуль RGZ.exe (Рисунок 1).

При нажатии кнопки «Нахождение пути» выводится полученный путь обхода лабиринта (Рисунок 2).

25

При вводе некорректных данных будет выведено сообщение об ошибке (Рисунок 3).

При нажатии кнопки «Помощь» вызывается диалоговое окно справки (Рисунок4).

**7.2 Проверка на корректную обработку выбора решения**

Результат работы программы приводит к нахождению пути обхода лабиринта и выводу решения на экран (Рисунок 2).

Если пользователь не указал все координаты, то будет выведено сообщение об ошибке (Рисунок 3)

Если пользователь вводит несколько раз одинаковые координаты комнат, то будет выведено сообщение об ошибке (Рисунок 3)

Поэтому можно сказать, что программа работает корректно.

**7.3 Проверка на правильность**

При выполнении программы результат должен совпадать с результатами, полученными на основе эмпирических закономерностей. В процессе тестирования программа отреагировала правильно, результат ввода решений совпал с ожидаемым.

При проверке на правильность, результат не отличался от ожидаемого, поэтому можно сказать, что программа работает правильно.

**7.4 Проверка на надежность**

26

При работе со списком решений не должен изменяться внешний вид пользовательского интерфейса кроме изображения лабиринта. Возможный вариант обхода был графически показан на форме. Пользовательский интерфейс при этом не изменился.

**8 Эскизы экранных форм**

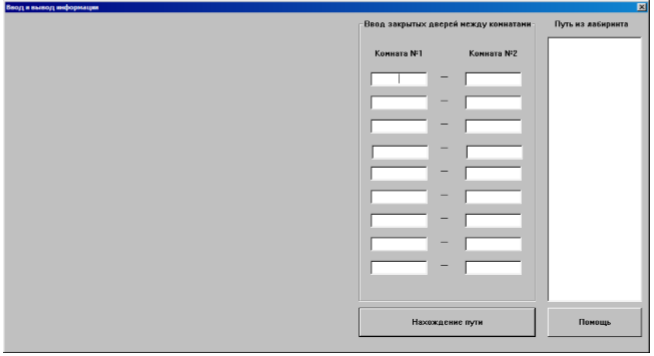


Рисунок 1. Запуск приложения

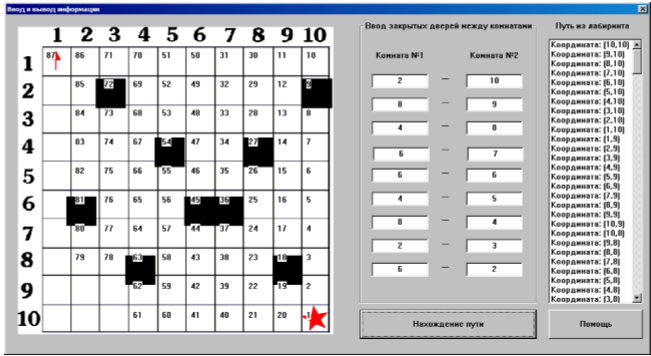


Рисунок 2. Нажатие на кнопку «Нахождение пути»

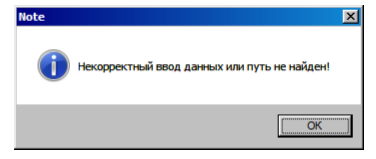


Рисунок 3. Сообщение об ошибке

**Список использованных источников**

1. Адаменко А.Н., Кучуков А.М. Логическое программированиеVisual Prolog. – СПб.: БХВ-Петербург, 2003

2. Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с англ.-М.Мир,1990

3. Дьяконов В.П., Борисов А.В. Основы искусственного интеллекта. Смоленск, 2007.

4. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний в интеллектуальных системах. Уч. пос. для вузов, СПб.:Питер, 2000

5. Корнеев В.В. и др. Базы данных. Интеллектуальная обработка информации. Рос.ассоц.издат. компьют.лит-ры.-М.:НОЛИДЖ, 2000