**Постановка задачи.**

Сформировать модель принятия решений в виде совокупности таблиц решений и написать модуль-интерпретатор для данной модели.

**Описание предметной области.**

Объекты моделирования – поле размером 10 х 10, на котором имеются препятствия в виде стен, и морковки. Действующие лица – заяц и волк. Объектом охоты для зайца являются морковки, а объект для охоты волка – сам заяц.

Волк и заяц осуществляют свои шаги по очереди. Первым делает шаг заяц. Если заяц видит стену, то он обязательно должен ее обойти. А если морковка находится рядом с зайцем, то он ее обязательно должен съесть.

Волк обходит препятствия аналогично зайцу.

Игра заканчивается, когда либо заяц съест все морковки, либо волк съест зайца.

**Описание табличной модели.**

Модель включает в себя 5 таблиц, которые содержат правила поведения зайца, волка и правила для начала игры, преодоления препятствий и переключения таблиц.

**Таблица 1. Готовность для начала игры.**

|  |  |  |
| --- | --- | --- |
| **Условия** | **Правила** | |
| **1** | **2** |
| Готов? | F | T |
| **Действия** |  | |
| Ждать | 1 |  |
| \* |  | 1 |

Данная таблица предназначена для определения момента начала игры. Когда будет нажата кнопка «Старт», то на вопрос «Готов?» будет получен положительный ответ и игра начнется.

Таблица полна.  
Таблица не противоречива.

**Таблица 2. Заяц.**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Условия** | **Правила** | | | | | | | | | | **Е** |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| Остались еще морковки? | F | T | T | T | T | T | T | T | T | T |  |
| Есть морковка спереди? |  | T | F | F | T | T | F | T | F |  |  |
| Есть морковка справа? |  | F | T | F | T | F | T | T | F |  |  |
| Есть морковка слева? |  | F | F | T | F | T | T | T | F |  |  |
| Заяц съеден? |  | F | F | F | F | F | F | F | F | T |  |
| **Действия** |  | | | | | | | | | |  |
| Повернуть направо |  |  | 1 |  |  |  |  |  |  |  |  |
| Повернуть налево |  |  |  | 1 |  |  |  |  |  |  |  |
| Съесть морковку |  | 1 | 2 | 2 | 2 | 2 | 2 |  |  |  |  |
| Повернуть направо / не поворачивать |  |  |  |  | 1 |  |  |  |  |  |  |
| Повернуть налево / не поворачивать |  |  |  |  |  | 1 |  |  |  |  |  |
| Повернуть направо / налево |  |  |  |  |  |  | 1 |  |  |  |  |
| Повернуть направо / налево / не поворачивать |  |  |  |  |  |  |  | 1 |  |  |  |
| \* | S | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 3 |

Данная таблица координирует движения зайца в условии, если рядом находится морковка. Заяц может ходить на одну клетку вперед, или назад, или вправо, или влево, предварительно повернувшись в нужную сторону.

Таблица полна, так как в ней присутствует правило иначе.

Таблица не противоречива.

**Таблица 3. Волк.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Условия** | **Правила** | | | | | | **Е** |
| **1** | **2** | **3** | **4** | **5** | **6** |
| Есть заяц справа? | T | F | F | F | F |  |  |
| Есть заяц слева? | F | T | F | F | F |  |  |
| Есть заяц спереди? | F | F | T | F | F |  |  |
| Есть заяц сзади? | F | F | F | T | F |  |  |
| Заяц с волком находятся на одной клетке? | F | F | F | F | F | T |  |
| **Действия** |  | | | | | |  |
| Повернуть направо | 1 |  |  |  |  |  |  |
| Повернуть налево |  | 1 |  |  |  |  |  |
| Повернуть назад |  |  |  | 1 |  |  |  |
| Съесть зайца | 2 | 2 | 1 | 2 |  |  |  |
| \* | S | S | S | S | 5 | S | 2 |

Данная таблица координирует движения волка в условии, если рядом находится заяц. Волк может ходить на одну клетку вперед, или назад, или вправо, или влево, предварительно повернувшись в нужную сторону.

Таблица полна, так как в ней присутствует правило иначе.

Таблица не противоречива.

**Таблица 4. Стена для зайца.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Условия** | **Правила** | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Есть стена прямо? | F | F | F | F | T | T | T | T |
| Есть стена справа? | F | T | F | T | T | F | T | F |
| Есть стена слева? | F | F | T | T | T | F | F | T |
| **Действия** |  | | | | | | | |
| Повернуть направо |  |  | 1 |  |  |  |  |  |
| Повернуть направо / налево | 1 |  |  |  |  |  |  |  |
| Повернуть налево |  | 1 |  |  |  |  |  |  |
| Шаг вперед | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 |
| Повернуть назад |  |  |  | 1 |  |  |  |  |
| Повернуть направо / налево / не поворачивать |  |  |  |  |  | 1 |  |  |
| Повернуть налево / не поворачивать |  |  |  |  |  |  | 1 |  |
| Повернуть направо / не поворачивать |  |  |  |  |  |  |  | 1 |
| \* | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Данная таблица координирует движения зайца в условии, если рядом находится стена.

Таблица полна и не противоречива.

**Таблица 5. Стена для волка.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Условия** | **Правила** | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Есть стена прямо? | F | F | F | F | T | T | T | T |
| Есть стена справа? | F | T | F | T | T | F | T | F |
| Есть стена слева? | F | F | T | T | T | F | F | T |
| **Действия** |  | | | | | | | |
| Повернуть направо |  |  | 1 |  |  |  |  |  |
| Повернуть направо / налево | 1 |  |  |  |  |  |  |  |
| Повернуть налево |  | 1 |  |  |  |  |  |  |
| Шаг вперед | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 |
| Повернуть назад |  |  |  | 1 |  |  |  |  |
| Повернуть направо / налево / не поворачивать |  |  |  |  |  | 1 |  |  |
| Повернуть налево / не поворачивать |  |  |  |  |  |  | 1 |  |
| Повернуть направо / не поворачивать |  |  |  |  |  |  |  | 1 |
| \* | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

Данная таблица аналогична таблице 4, только для волка.

Таблица полна и не противоречива.

**Оптимизированная трансляция**

Таблица 1

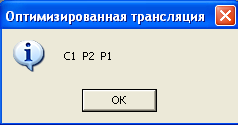
****

Таблица 2

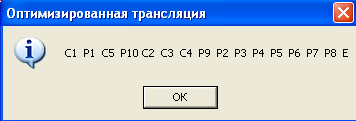
****

Таблица 3

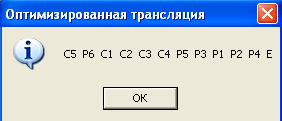
****

Таблица 4

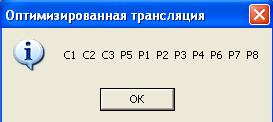
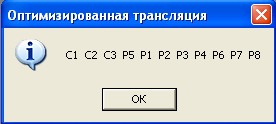
****

Таблица 5

****

**Вид программы.**

**Основное окно программы.**



**Возможные положения зайца:**

RabbitBottom.bmpRabbitLeft.bmpRabbitRight.bmpRabbitTop.bmp

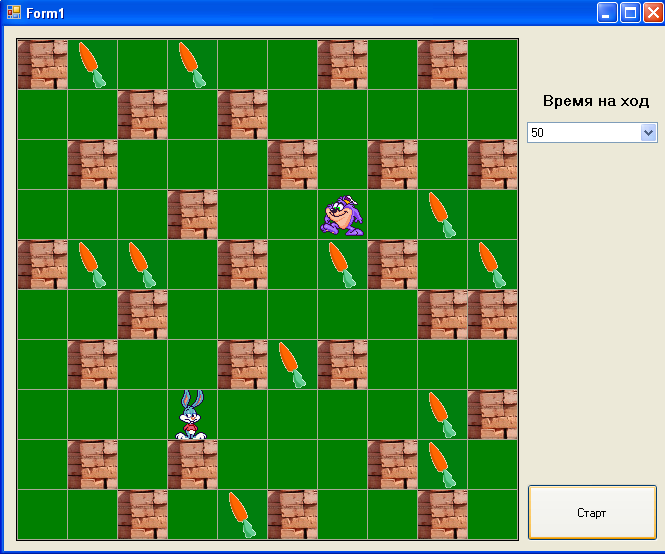
**Возможные положения волка:**

WolfTop.bmpWolfBottom.bmpWolfLeft.bmpWolfRight.bmp

Предусмотрена возможность выбора времени на ход: от 50 мс до 150 мс.

Программа реализована в среде Microsoft Visual Studio 2008.

**Пример работы программы.**



**Приложение.**

Модуль-интерпретатор:

using System;

using System.Windows.Forms;

using System.Threading;

using System.Runtime.InteropServices;

namespace RabbitWolf

{

public class MyHookClass : NativeWindow

{

[DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Auto)]

static extern uint RegisterWindowMessage(string lpString);

uint simpr;

Form1 f;

public MyHookClass(Form1 af)

{

simpr = RegisterWindowMessage("RabbitWolfMessage");

this.AssignHandle(af.Handle);

f = af;

}

protected override void WndProc(ref Message m)

{

int wparamhi,wparamlo,wparam;

int lParam = Convert.ToInt32("" + m.LParam);

if (m.Msg == simpr)

{

wparam = Convert.ToInt32("" + m.WParam);

wparamhi = wparam / 65536;

wparamlo = wparam - wparamhi \* 65536;

#region Условия

if (wparamhi == 0)

{

if (wparamlo == 1) // Таблица 1

{

switch (lParam) // Условия

{

case (1): m.Result = f.IsReady(); break;

}

}

else if (wparamlo == 2) // Таблица 2

{

switch (lParam) // Условия

{

case (1): m.Result = f.IsCarrotElse(); break;

case (2): m.Result = f.IsCarrotInFront(); break;

case (3): m.Result = f.IsCarrotRight(); break;

case (4): m.Result = f.IsCarrotLeft(); break;

case (5): m.Result = f.IsRabbitEaten(); break;

}

}

else if (wparamlo == 3) // Таблица 3

{

switch (lParam)

{

case (1): m.Result = f.IsRabbitRight(); break;

case (2): m.Result = f.IsRabbitLeft(); break;

case (3): m.Result = f.IsRabbitInFront(); break;

case (4): m.Result = f.IsRabbitBack(); break;

case (5): m.Result = f.IsRabbitWolfTogether(); break;

}

}

else if (wparamlo == 4) // Таблица 4

{

switch (lParam)

{

case (1): m.Result = f.IsWallInFrontRabbit(); break;

case (2): m.Result = f.IsWallRightRabbit(); break;

case (3): m.Result = f.IsWallLeftRabbit(); break;

}

}

else if (wparamlo == 5) // Таблица 5

{

switch (lParam)

{

case (1): m.Result = f.IsWallInFrontWolf(); break;

case (2): m.Result = f.IsWallRightWolf(); break;

case (3): m.Result = f.IsWallLeftWolf(); break;

}

}

}

#endregion

#region Действия

else if (wparamhi == 1)

{

if (wparamlo == 1) // Таблица 1

{

switch (lParam)

{

case (1): f.Sleep(); break;

}

}

else if (wparamlo == 2) // Таблица 2

{

switch (lParam)

{

case (1): f.TurnRightRabbit(); break;

case (2): f.TurnLeftRabbit(); break;

case (3): f.StepRabbit(); break;

case (4): f.TurnRightNotTurnRabbit(); break;

case (5): f.TurnLeftNotTurnRabbit(); break;

case (6): f.TurnLeftRightRabbit(); break;

case (7): f.TurnLeftRightNotTurnRabbit(); break;

}

}

else if (wparamlo == 3) // Таблица 3

{

switch (lParam)

{

case (1): f.TurnRightWolf(); break;

case (2): f.TurnLeftWolf(); break;

case (3): f.TurnBackWolf(); ; break;

case (4): f.StepWolf(); break;

}

}

else if (wparamlo == 4) // Таблица 4

{

switch (lParam)

{

case (1): f.TurnRightRabbit(); break;

case (2): f.TurnLeftRightRabbit(); break;

case (3): f.TurnLeftRabbit(); break;

case (4): f.StepRabbit(); break;

case (5): f.TurnBackRabbit(); break;

case (6): f.TurnLeftRightNotTurnRabbit(); break;

case (7): f.TurnLeftNotTurnRabbit(); break;

case (8): f.TurnRightNotTurnRabbit(); break;

}

}

else if (wparamlo == 5) // Таблица 5

{

switch (lParam)

{

case (1): f.TurnRightWolf(); break;

case (2): f.TurnLeftRightWolf(); break;

case (3): f.TurnLeftWolf(); break;

case (4): f.StepWolf(); break;

case (5): f.TurnBackWolf(); break;

case (6): f.TurnLeftRightNotTurnWolf(); break;

case (7): f.TurnLeftNotTurnWolf(); break;

case (8): f.TurnRightNotTurnWolf(); break;

}

}

System.Threading.Thread.Sleep(f.ind);

// Ответом на запрос действия со стороны СИМПР должна быть единица

m.Result = new IntPtr(1);

}

#endregion

Application.DoEvents();

}

else

{

base.WndProc(ref m);

}

}

}

}