4 Образцы решения прямой и обратной задач с исполнителями Робот и редакторСреды

*(Фрагменты использования библиотеки классов «**библиотекаМетодовИсполнителяРобот**»)*

4.1 Введение

В данном документе мы познакомимся с постановкой и общими подходами при решении задач с исполнителем Робот. Как было отмечено в первой части документации по работе с рассматриваемым исполнителем[1], прежде чем решать задачу с Роботом надо каким-то образом создать среду (совокупность особым образом расположенных объектов: между клеточных стен, закрасок, загрязненностей и помеченностей клеток), действуя в которой исполнитель должен выполнить некоторую работу. Чаще всего эта работа сводится к разметке отдельных участков поля за счет закрашивания.

Работа по созданию среды, которую должен «обработать» Робот носит название ***обратной задачи***. Для обратной задачи может быть сформулировано особое условие, но, как правило, дело ограничивается постановкой ***прямой задачи***. Суть этой задачи – в созданной среде средствами исполнителя Робот произвести определенные действия или расчеты.

Обратная задача может быть решена в *непосредственном режиме управления* редактирования среды. Для этих целей можно с успехом воспользоваться средствами редактирования среды поля Робота, содержащимися в пакете Windows КУМир. Такой прием уместен и на первых порах удобен. Именно использование этого режима разработки среды делает «ненужным» формулировку обратной задачи в виде самостоятельной. Вместе с тем, замечено, что статичность создаваемых сред, нередко порождает невыгодную для преподавания ситуацию. Во-первых, преподавателю приходится постоянно создавать все новые и новые обстановки с учетом замеченных им (преподаватем) недочетов в алгоритмах решения задачи обучаемым. Это с одной стороны может породить у обучаемого чувство «предвзятого» отношения к нему со стороны учителя, в с другой – отсуствие стимула к решению задачи в *общем виде*, что требуется при формировании *алгоритмического стиля мышления*, являющегося основной целью курса основ алгоритмизации (программирования). Во-вторых, это отнимает драгоценное время на уроке, которое может быть потрачено более продуктивно в образовательном смысле.

Мы предлагаем раздельное решение *обратной* (исполнитем ***редакторСреды***) и *прямой* задач (исполнителем ***инфоРобот***).

4.2 Алгоритм начала работы при решении задач с исполнителями

4.2.1 Особенности работы с исполнителем и визуализацией результатов работы

1. По входе в среду программирования *Visual Studio Начало работы* выбирается пункт меню «*Создать проект*».

2. На форме «*Создать проект*» в разделе «*Visual C# Windows*» выбирается (средняя часть формы) шаблон «*Приложение Windows Forms*», а в нижней части формы в окне «*Имя*» указывается имя проекта (*имя пространства имен программы*), а в окне «*Расположение*», используя возможности кнопки «*Обзор*», выбирается папка, в которой будет раз-

[1] Смотри введение к «СК исполнителя РедакторСреды исполнителя Робот»

мещено решение задачи. После этого нажимается кнопка «*OK*». Через некоторое время (большее, чем при работе в консольном режиме) на экране разворачивается стандартное окно начальной установки, включая форму «*Forms1*».

3. В разделе формы «*Обозреватель решений*» нажатим правой клавиши мыши на пункте «*Forms1.cs*» добиваемся появления нового меню, в котором выбираем пункт «*Удалить*» и после запроса на подтвержение, удаляем форму «*Forms1*».

4. Дважды щелкаем левой клавишей мыши на пункте «*Program.cs*» того же раздела. В результата разворачивается окно вида:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace решениеЗадачи_00
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

5. В пункте меню «**Проект**» выбрать пункт «**Добавить ссылку**», а по открытии формы – подпункт «**Обзор**». Далее, найти файл «*библиотекаМетодовРоботаНаСтруктуреПоле.dll*»[2]

6. Раздел подключения библиотек классов дополнить строкой «**using** библиотекаМетодовРоботаНаСтруктуреПоле;» [3]

7. Рассмотрим случай последовательного решения *обратной* и *прямой* задач в рамках одного проекта. В таком случае в блоке класса «**class** *Program*» необходимо вставить строки объявления объектов

   « **static** ***редакторСреды*** редактор;

[2]Адрес файла следует узнать у преподавателя.

[3]В случае, если по ходу решения задачи предполагается использовать готовые средства библиотеки построения сред, пункты данного описания 5, 6 надо повторить, с целью подключения библиотеки «*библиотекаФормированияСред.dll*»

«**static** ***инфоРобот*** робот;» и

«**static** ***ситуацияНаПоле*** поле;»

8. Блок метода «**static void** Main(**string**[] args)» преобразовать следующим образом:

```
static void Main()
{
     Application.EnableVisualStyles();
     Application.SetCompatibleTextRenderingDefault(false);
     редактор = new редакторСреды(20, 15);
     поле = решениеОбратнойЗадачи();
     Application.Run(new средаРобота(поле));
}
```

В данном случае приведенный фрагмент кода отражает ситуацию предназначенную для решения и отображение решения только *обратной задачи*. Сам код «решениеОбратнойЗадачи» будет приведен ниже. где

– строка *редактор = new редакторСреды(20, 15);* означает вызов конструктора редактора среды с дополнительным указанием предельных количеств столбцов и рядов поля;

– строка *поле = решениеОбратнойЗадачи();* означает вызов кода решения;

4.3 Обратная задача

4.3.1 Постановка задачи

В произвольном ряду поля построить единственную горизонтальную стену, которая

– не примыкала бы к внешним вертикальным стенам поля;

– не совпадала с внешними горизонтальными стенами поля;

– пометить все клетки, расположенные под построенной стеной в их верхней части символом «Z» (это внешний признак того, что все эти клетки в ходе решения прямой задачи должны быть ракрашены).

Самого исполнителя Робот надо расположить в СЗ углу поля.

4.3.2 Исходный код решения обратной задачи

1. Исходный код метода «решениеОбратнойЗадачи» может быть расположен непосредственно за методом Main. В данном случае он имеет вид:

```
static ситуацияНаПоле решениеОбратнойЗадачи()
{
     Random случайноеЧисло = new Random();
     int ширина = редактор.ширинаПоля;
     int высота = редактор.высотаПоля;
     int рядСоСтеной = случайноеЧисло.Next(2, высота - 1);
```

```
    int колонкаНачалаСтены = случайноеЧисло.Next(1, ширина / 2);
    int колонкаКонцаСтены = случайноеЧисло.Next(ширина / 2, ширина - 2);
    редактор.положениеРобота(0, 0);
    // Установка стены и пометка клеток, которые должны быть закрашены
    for (int колонка = колонкаНачалаСтены; колонка < колонкаКонцаСтены; колонка++)
   {
        редактор.поставитьСтену(колонка, рядСоСтеной, 8); // Стена сверху ряда
        for (int ряд = рядСоСтеной; ряд < высота; ряд++)
       {
            редактор.пометить(колонка, ряд, 0, 'Z');
       }
    }
    редактор.кончитьРаботу("Inlet.in");
    // Подготовка файла шаблона ответа не производится
    return редактор.результатРаботы;
}
```

где

– обратим внимание, что значения параметров «рядСоСтеной», «колонкаНачалаСтены» и «колонкаКонцаСтены» строятся с использованием датчика случайных чисел. Следовательно, формируемая среда будет действительно динамически меняться от одного запуска программы к другому. Более того, значения формируются так, чтобы ряд стены не был первым (в таком случае стена слилась бы с верхней стеной поля), а начало стены берется в «любой» колонке, но не ближе второй и не дальше середины поля. Аналогично этому строится и конец стены.

– возвращаемое значение реализуется вызовом свойства «редактор.результатРаботы;».

4.4 Прямая задача

4.4.1 Постановка задачи

Где-то на поле имеется единственная горизонтальная стена, не примыкающая к внешним стенам поля Робота. Сам Робот в СЗ углу поля.

Закрашен прямоугольник клеток поля, расположенных под горизонтальной стеной до нижней внешней стены поля.

4.4.2 Дополнение кода метода Main вызовом решения прямой задачи

Приведенный выше код метода Main должен быть пополнен инициированием объекта-исполнителя Робот, вызовом метода «решениеПрямойЗадачи» и визуализаций выполненного решения. Пополнение кода будет иметь вид:

```
. . . . . .
робот = new инфоРобот(поле);
поле = решениеПрямойЗадачи();
Application.Run(new средаРобота(поле));
```

Заметим, что инициализация объекта Робот сопровождается передачей ему параметра-значения «поле», через который Робот получает информацию о построенной исходной обстановке, которую он (Робот) должен решить.

4.4.3 Исходный код решения прямой задачи

1. Исходный код метода «решениеПрямойЗадачи» может быть тоже расположен непосредственно за методом Main. В данном случае он имеет вид:

```
static ситуацияНаПоле решениеПрямойЗадачи()
{
    поискСтены();
    обойтиСтену();
    закраситьПодСтеной();
    return робот.результатРаботы;
}
static void закраситьПодСтеной()
{
    while (робот.сверхуСтена)
    {
        закраситьКолонку();
    }
}
static void закраситьКолонку()
{
    робот.закрасить();
    while (робот.снизуСвободно)
    {
        робот.вниз();
        робот.закрасить();
    }
    while (робот.сверхуСвободно)
    {
        робот.вверх();
    }
    робот.вправо();
}
static void обойтиСтену()
{
    робот.влево();
    робот.вниз();
    робот.вправо();
}
static void поискСтены()
{
    поискВРяду();
    while (робот.снизуСвободно)
    {
```

```
            возврат();
            поискВРяду();
        }
    }
    static void возврат()
    {
        while (робот.снизуСвободно && робот.слеваСвободно)
        {
            робот.влево();
        }
        робот.вниз();
    }
    static void поискВРяду()
    {
        while (робот.справаСвободно && робот.снизуСвободно)
        {
            робот.вправо();
        }
    }
```

В заключение сделаем несколько замечений:

1. *Замечание 1.* По запуске представленного кода на выполнение – сначала появляется «картинка» исходной обстановки, на которой, как было замечено ранее, клетки помеченные символом «Z» должны быть закрашенными в результате работы Робота.

   Если первую «картинку» закрыть, то тут же появляется новая – результат решения прямой задачи. В данном случае надо обратить внимание на то, чтобы все нужные клетки были закрашены причем по одному разу.

2. *Замечание 2.* В приведенном решении «решениеОбратнойЗадачи» и «решениеПрямойЗадачи» оформлены как методы класса ***Program***. Это потребовало их оформления со спецификатором **static** в силу того, что таким спецификатором необходимо снабжен основной метод **Main**. Можно было бы для обоих решений создать отдельные, самостоятельные классы. В таком случае потребовалось бы создать объекты соотвествующих классов и вызывая нужные методы решать задачи в истинном стиле *объектно ориентированного программмирования*.

3. *Замечание 3.* Продолжая идею создания самостоятельных классов, можно создать библиотеку конструкторов сред для всего множества задач, которые преподаватель предлагает учащимся. Имея подобную библиотеку и подключив ее к равертываемому проекту точно так, как это делалось для библиотеки «***библиотекаМетодовРоботаНаСтруктуреПоле.dll***» можно обеспечить работой всю группу обучаемых, предоставив им возможность самостоятельной визуальной проверки результатов. При таком подходе, преподаватель высвобождается для консультативной работы по сути решаемых *прямых* задач с наибольшим количеством обучаемых.