Проверяемые задания
для компьютерного исполнителя Робот

Теоретические положения и практические рекомендации педагогам

А. И. Занько,
учитель информатики высшей категории
ГУО «Люсинский детский сад –
средняя школа имени Якуба Коласа» Ганцевичского района

В соответствие с учебной программой по предмету «Информатика» в 7 классе предполагается изучение компьютерного исполнителя Робот, что содействует быстрому обучению семиклассников основным алгоритмическим конструкциям. Ускорению этого процесса и облегчению труда преподавателя способствуют так называемые проверяемые задания, содержащиеся в электронном задачнике, который включен в учебную систему программирования PascalABC.Net. Примеры задач из встроенного задачника приведены в учебном пособии для 7 класса. Преподаватель может самостоятельно создавать такие задания.

В данной статье рассмотрим процесс разработки заданий для исполнителя Робот, который сводится к построению модуля, содержащего группу (группы) заданий. Такой модуль имеет следующий общий вид:

Unit *имя_модуля;*
Interface
Uses RobotTaskMaker;
Implementation
Процедуры заданий
Begin
Процедуры регистрации групп заданий
Процедуры регистрации заданий в группах
End.

При разработке заданий используется конструктор RobotTaskMaker, реализованный в одноименном модуле. Познакомимся со стандартными процедурами этого конструктора.

Формулировка текста задания осуществляется с помощью процедуры TaskText (s), где s – строка с описанием задания.

Поле размером m x n задается процедурой Field(m,n), где m – количество клеток по горизонтали, n – количество клеток по вертикали. Отметим, что клетки нумеруются по горизонтали слева направо (начиная с 1) и по вертикали – сверху вниз (начиная с 1).

Для задания начального положения Робота используется процедура RobotBegin(x1,y1), где (x1,y1) – координаты клетки, определяющей начальное положение Робота (внутри такой клетки будет большой желтый квадрат). Процедурой RobotEnd(x2,y2) задается конечное положение Робота, здесь (x2,y2) – координаты клетки для конечного положения Робота (такая клетка будет содержать в левом верхнем угле маленький желтый квадрат). Вместо этих двух процедур можно использовать одну: RobotBeginEnd (x1,y1,x2,y2).

Построение горизонтальной стены выполняет процедура HorizontalWall (x,y,L), где (x,y) – левая точка стены, а L – длина стены. Процедура VerticalWall (x,y,L) используется для построения вертикальной стены; здесь (x, y) – верхняя точка стены, а L – её длина. Отметим, что горизонтальные линии нумеруются сверху от границы поля вниз (начиная с 0), а вертикальные – слева от границы поля направо (начиная с 0).

Процедура Tag(x,y) помечает клетку (x,y), которую в задании необходимо закрасить (такая клетка в задании будет отмечена маленьким черным квадратиком в центре). Процедура TagRect (x1,y1,x2,y2) помечает прямоугольник из клеток, задаваемый координатами противоположных вершин (x1,y1) и (x2,y2), для закрашивания в задании. Процедура MarkPainted (x,y) создаёт закрашенную клетку (x,y) (в задании эта клетка будет уже закрашена).

При разработке процедур удобно пользоваться функцией случайных чисел Random(a,b), которая формирует целое случайное число в диапазоне от a до b включительно. Это позволит получать разнообразные начальные обстановки исполнителя при каждом новом вызове задания.

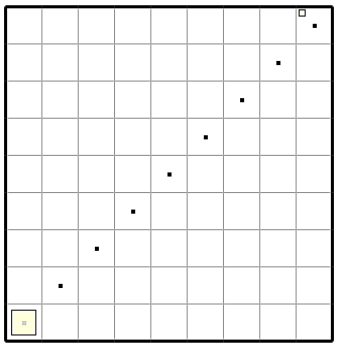
Все группы и задания должны быть определенным образом зарегистрированы. Для регистрации группы используется стандартная процедура RegisterGroup (ig,og,im,kz), где ig – имя группы, og – описание группы, im – имя модуля, в котором описана группа, kz– количество заданий в группе. Имя группы должно содержать не более 7 символов (цифр и латинских букв) и не должно оканчиваться цифрой; количество заданий – не более 999.

Для регистрации задания используется стандартная процедура RegisterTask (iz,ip), где iz – имя задания, ip – имя процедуры, в которой реализовано задание. Имя задания, входящего в группу, должно состоять из имени этой группы и числа. Данную процедуру следует вызывать для каждого задания.

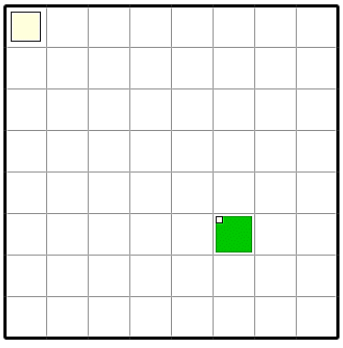
Процедуры RegisterGroup и RegisterTask прописываются в секции инициализации модуля, содержащего реализацию новой группы заданий для Робота.

Пример. Создадим модуль RobotTasks.pas, содержащий группу Robиз трех заданий Rob1, Rob2 и Rob3.

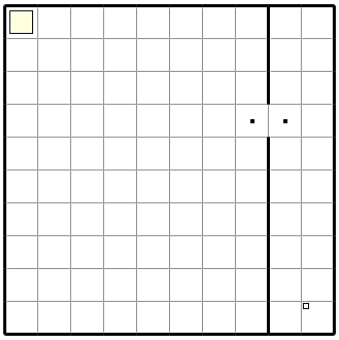
Задание 1. Поле Робота имеет квадратную форму произвольного размера. Робот находится в левой нижней клетке. Составить программу закрашивания Роботом всех клеток по диагонали от левой нижней клетки до правой верхней клетки.



Задание 2. Поле Робота имеет форму квадрата произвольного размера. Исходное положение Робота – левая верхняя клетка. Где-то на поле имеется единственная закрашенная клетка. Роботу необходимо зайти в эту клетку.



Задание 3. Поле Робота имеет форму квадрата 10х10. Исходное положение Робота – левая верхняя клетка. Где-то правее Робота имеется вертикальная стена с единственным проходом (проход может быть в любой клетке, кроме верхней и нижней). Роботу необходимо перейти в противоположный угол поля, при этом закрасить две клетки против прохода в стене.



Запустим среду программирования PascalABC.Net и наберем следующий текст:

```
Unit Robot Tasks;
Interface
Uses RobotTaskMaker;
Implementation
//Процедура создания задания1
procedureRobot_1;
varn: integer;
begin
TaskText('ЗаданиеRob1. Закрасить клетки по диагонали квадрата');
n:=Random(3,20);
Field(n,n);
RobotBegin(1,n);
RobotEnd (n, 1);
For vari:=1 to ndoTag(i,n+1-i);
end;
//Процедура создания задания2
procedureRobot_2;
varn,k,m: integer;
begin
```

```
TaskText('Задание Rob2. Зайти в закрашенную клетку');
n:=Random(3,20);
Field(n,n);
k:=Random(1,n);
m:=Random(1,n);
MarkPainted(k,m);
RobotBeginEnd(1,1,k,m);
end;
//Процедура создания задания3
procedureRobot_3;
vark,m: integer;
begin
TaskText('Задание Rob3. Зайти в противоположный угол и закрасить две клетки
против проема в стене');
Field(10,10);
k:=Random(1,9);
m:=Random(1,8);
VerticalWall(k,0,m);
VerticalWall(k,m+1,9-m);
Tag(k,m+1);
Tag(k+1,m+1);
RobotBeginEnd(1,1,10,10);
end;
begin
//Процедура регистрации группы
RegisterGroup('rob','Задания для Робота','RobotTasks',3);
//Процедуры регистрации заданий 1,2,3
RegisterTask('rob1', Robot_1);
RegisterTask('rob2', Robot_2);
RegisterTask('rob3', Robot_3);
end.
```

Сохраним текст в файле RobotTasks.pas (имя файла должно совпадать с именем модуля).

Всё готово. Можно пользоваться созданными заданиями. Так, для составления программы решения первого задания (оно у нас имеет имя ‘rob1’) набираем шаблон (в строке подключения модулей после стандартного модуля Robot указываем также имя созданного нами модуля – RobotTasks):

```
Uses Robot, RobotTasks;
Begin
  Task('rob1');
End.
```

Если запустить программу на выполнение, то будет выведено соответствующее задание для Робота. Осталось дописать программу до конца и проверить её работоспособность (оставим это читателю для самостоятельной работы). Аналогично и для других заданий.

На этом процесс разработки проверяемых заданий можно закончить. Но для того чтобы наши задания появлялись в окне редактора по команде “модули → создать шаблон программы” или Ctrl+Shift+L (как это делается со всеми встроенными заданиями разработчиков системы), необходимо выполнить дополнительную работу.

После сохранения файла, содержащего наш модуль, выполним компиляцию:

Программа → Компилировать (в папке Output, содержащейся в папке PABCWork.Net, будет создан файл RobotTasks.pcu).

Созданный модуль (файл с расширением .pcu) из папки, в которой был создан, перенесемв папку *Lib*:

Program Files (x86) → PascalABC.Net → Lib

Внесем изменения в файл *loadpabc.dat*, открыв его с помощью Блокнота **(**ProgramFiles (x86) → PascalABC.Net → PT4 → loadpabc.dat). В конце списка наборов заданий для Робота добавим строку с содержанием:

==rob|задания для Робота|3

Сохраним изменения.

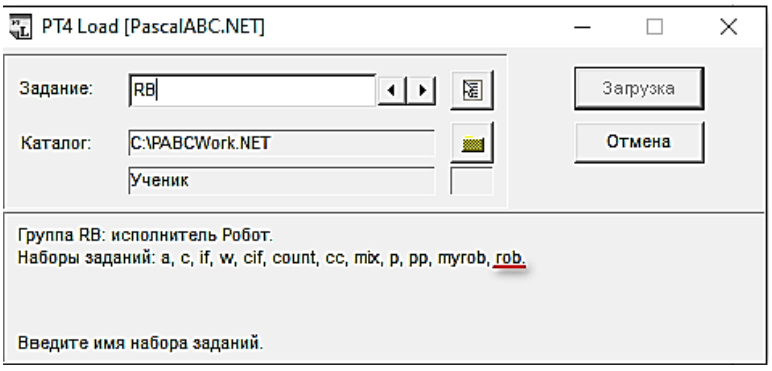
Для создания шаблона задания 1 в новом окне редактора введем текст:

```
Uses Robot, RobotTasks;
Begin
  Task('rob1');
End.
```

Запустим программу на выполнение. Будет выведено соответствующее задание для Робота. Сохранять программу-шаблон в каком-либо файле не надо.

Аналогично поступаем с шаблонами для заданий 2 и 3.

При первом запуске программы с подключенным модулем *RobotTasks* созданная нами группа *rob* автоматически зарегистрируется в мастере по созданию программ-заготовок *PT4Load*. Если теперь нажать кнопку  «создать шаблон программы» и в появившемся окне *PT4Load* ввести префикс *RB* в поле «Задание», то окно примет следующий вид:



Мы видим, что группа заданий *rob* появилась в списке доступных групп для исполнителя Робот. Если приписать имя одного из созданных заданий, например, *rob2*, то после нажатия Enter в рабочем каталоге будет создан новый файл *RBrob2.pas* со следующим содержимым:

```
Uses Robot, RobotTasks;
Begin
  Task('rob2');
End.
```

Таким образом, теперь вместо ручного набора шаблона программы можно вызывать готовый шаблон с помощью соответствующей команды окна редактора.

Созданный модуль можно обновлять, добавлять новые задания, применяя описанный выше механизм. Использование собственных проверяемых заданий позволит учителю разнообразить учебные задания для школьников, сделать более

интересным учебно-воспитательный процесс. Будет полезным привлечение мотивированных ребят к выполнению проектов по разработке таких заданий.

Литература

1. Информатика. Учебное пособие для 7 класса учреждений общего среднего образования с русским языком обучения / В. М. Котов, А. И. Лапо, Е. Н. Войтехович. – Минск: Народная асвета, 2017.
2. Осипов, А. В. PascalABC.Net: Введение в современное программирование / А. В. Осипов. – Ростов н/Д, 2019.