ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ
«ШАГ В БУДУЩЕЕ, МОСКВА»

947

регистрационный номер

Информатика и системы управления
название факультета

Программное обеспечение ЭВМ и информационные технологии
название кафедры

Разработка языка программирования и системы обучения детей младшего школьного возраста основам программирования
название работы

Автор: Алексашин Антон Павлович
фамилия, имя, отчество

МАОУ СОШ №36 г.Владимир, 10 «А» класс
наименование учебного заведения, класс

Москва – 2020

Аннотация

Данный проект является составной частью предлагаемой мной концепции обучения школьников программированию. Его целью является разработка системы обучения детей младшего школьного возраста основам программирования.

В основе всех подобных систем лежит некий исполнитель (черепаха в лого, робот-исполнитель в кумире), который выполняет написанную на встроенном языке программу. Наряду с рядом других инноваций, мной предложен не имеющий аналогов, принципиально новый подход в организации функционирования подобных систем. В основе данного подхода лежит создание для исполнителя недетерминированной внешней среды, что, во-первых, позволяет использовать для обучения максимально простой язык и при этом создавать программы высокой сложности, во-вторых, существенно повышает мотивацию школьников через увеличение игровой составляющей обучающей системы.

Результат практического применения системы в обучении школьников младшего школьного возраста показал, что даже школьники, имеющие неудовлетворительные оценки по основным школьным предметам (в том числе по математике) успешно осваивают предложенный курс обучения, после которого они в состоянии свободно решать ряд задач из ОГЭ по информатике для выпускников 9-го класса общеобразовательных школ и готовы приступить к изучению универсальных языков программирования

Содержание

1 Введение

Вероятно, в мире не существует ни одного человека, кто хотя бы раз не задумывался о будущем. Что ожидает меня, страну, человечество? Ответ на этот вопрос во многом очевиден - новые технологии: искусственный интеллект, виртуальная реальность, роботы.

Мир новых технологий очень многообразен, но есть то, что их объединяет- информация. А если есть информация, значит возникает потребность в ее поиске, хранении, обработке, т.е. в информационных технологиях [1]. Если мы хотим уверенно смотреть в будущее и не бояться оказаться на задворках мирового прогресса, то изучение информационных технологий должно стать краеугольным камнем в образовании современного школьника. И особое внимание при изучении информационных технологий необходимо обратить на изучение программирования. Для того, чтобы в полной мере отвечать современным требованиям, школьники должны начинать освоение соответствующих технологий начиная с младшего школьного возраста. Естественно, это должно происходить с использованием соответствующих программных средств.

Прародителем подобных систем обучения можно считать язык Лого, разработанный в 1967 г. в образовательных целях для обучения детей дошкольного и младшего школьного возраста основным концепциям программирования [2]. Язык Лого известен первым графическим исполнителем черепахой- воображаемым роботоподобным устройством, которое перемещается по экрану и поворачивается в заданных направлениях, при этом оставляя (или, по выбору, не оставляя) за собой нарисованный след заданного цвета и ширины. В настоящее время существует множество подобных систем обучения. Большую популярность получил созданный в Массачусетском технологическом институте язык визуального программирования Scratch [3]. В нашей стране широкое распространение получила разработанная НИИСИ РАН система Кумир [4].

2 Ключевые требования к системам обучения основам программирования

Ключевыми требованиями, предъявляемыми к данным программным средствам, являются:

1) Игровая форма обучения (*edutainment*-обучение через развлечение) [5,6].

Общеизвестно, что обучение происходит более успешно, если в процессе обучения присутствует игровой компонент, что особенно актуально для школьников младшего школьного возраста.

2) Обеспечение возможности плавного перехода от использования адаптированных программных средств к изучению универсальных языков программирования. Для этого:

2.1) встроенный язык программирования должен соответствовать парадигме структурного программирования [7] и содержать базовые управляющие конструкции в т.ч.:

а) последовательности;

б) конструкции ветвления как в полной, так и в сокращенной форме;

в) различные варианты циклических конструкций;

г) подпрограммы.

Для успешного освоения базовых конструкций языков программирования обучающая система должна не только содержать данные конструкции, но и стимулировать обучаемых для их активного использования.

2.2) с одной стороны встроенный язык программирования должен быть максимально прост для обучения, с другой стороны, обучающая система должна обеспечивать возможность написания программ любой сложности от самых простых до достаточно сложных, требующих от обучаемого высокого уровня алгоритмического мышления и навыков алгоритмизации решаемых задач. В этом заключается существенное противоречие, которое я бы назвал «парадоксом учебного языка». Стандартный подход, который используют

разработчики обучающих систем для решения данной проблемы- это усложнение языка. Я считаю данный подход неверным и предлагаю иное решение.

Для обучения школьников младшего школьного возраста основам программирования, мной была разработана **обучающая система «Робот-сапер».**

3 Основные задачи, решаемые данной системой

Основные задачи, решаемые данной системой:

1) обучение школьников основам программирования, навыкам написания эффективных программ;
2) подготовка к изучению универсальных языков программирования;
3) выработка у обучаемых алгоритмического мышления;
4) подготовка школьников к сдаче ОГЭ по информатике (решению задач типа «Исполнитель Робот»).

4 Основные отличия от существующих аналогов

Основные отличия разработанной мной системы от существующих аналогов:

1) *недетерминированная внешняя для исполнителя среда;*

Характерной чертой всех существующих систем обучения является детерминированная внешняя для исполнителя среда. Я предлагаю принципиально иной подход, основанный на хаотичном изменении внешней среды. Фактически обучаемому предстоит создать систему искусственного интеллекта, управляющую роботом в автономном режиме. В рамках данной системы робот вынужден постоянно анализировать внешнюю среду и взаимодействовать со случайно возникающими объектами, что является упрощенным аналогом систем, разрабатываемых для управления реальными

объектами (например, система "автопилота", управляющая автомобилем и т.д.). Данный подход позволяет разрешить противоречие, связанное с «парадоксом учебного языка» и существенно повысить мотивацию школьников через увеличение игровой составляющей обучающей системы.

2) *мультиязычный синтаксис языка программирования;*

Как показывает практика, часть школьников испытывает трудности при изучении программирования, связанные с тем, что синтаксис современных языков программирования основан на использовании английского языка. Для решения данной проблемы встроенный в обучающую систему учебный язык программирования имеет как русскоязычный, так и англоязычный синтаксис. На начальном этапе обучение происходит с использованием русскоязычной версии языка, затем происходит переход на англоязычную версию. Это позволяет плавно подвести обучаемого к освоению базовых языков программирования.

3) *система тестирования полученных знаний*:

- тестирует полученные школьником знания (при этом позволяет тестировать строго определенные знания, вплоть до определенных управляющих конструкций);

- ориентирует обучаемого на разработку наиболее эффективных алгоритмов, оптимальных с точки зрения использования ресурсов компьютера (быстродействие, память);

- существенно повышает мотивацию школьников через использование игровых элементов.

4) *система автоматической проверки решений;*

Любое решение любой задачи может быть сразу проверено с использованием встроенной системы автоматического тестирования без предварительной настройки системы.

5) *система автодополнения текущих лексем;*

Позволяет существенно упростить и ускорить процесс текстового ввода программы.

6) *система ускоренного ввода команд*:

- ускоряет и упрощает процесс написания программы;

- позволяет избежать синтаксических ошибок, что особенно важно на первоначальном этапе обучения.

5 Описание системы

В ходе работы с обучающей системой перед школьником стоит задача - написать программу, управляющую роботом-сапером, который должен разминировать минное поле. Минное поле расположено на плоскости, разбитой на клетки. Клетка может быть свободна, занята миной, которую нужно разминировать, либо занята препятствием (стена), которое нужно обойти. Попытка пройти сквозь стену или мину приведет к разрушению робота.

Для управления роботом обучаемый может использовать встроенный язык программирования.

Встроенный язык программирования содержит:

1) команды, определяющие начало и конец основной программы:

начало программы **begin program**

конец программы **end program**

2) команды-приказы (конструкции последовательности)

вверх, вниз, вправо, влево, разминировать сверху, разминировать снизу, разминировать справа, разминировать слева;

up, down, right, left, clear top, clear bottom, clear right, clear left;

3) конструкции ветвления

3.1) полная форма:

если <условие> **то**	**if** <условие> **then**
<команды>	<команды>
иначе	**else**
<команды>	<команды>
все	**all**

3.2) сокращенная форма:

если <условие> **то**	**if** <условие> **then**
<команды>	<команды>
все	**all**

Допускается использование вложенных условных конструкций.

4) циклические конструкции

4.1)

нц пока <условие>	**bl while** <условие>
<команды>	<команды>
кц	**el**

4.2)

нц выполнить <значение> **раз**	**bl loop** <значение> **do**
<команды>	<команды>
кц	**el**

где <значение> - это целое число больше 0.

Допускается использование вложенных циклических конструкций.

5) подпрограммы

подпрограмма <имя>	**procedure** < имя >
<команды>	<команды>
конец подпрограммы	**end procedure**

где < имя > - это название подпрограммы, в котором могут быть использованы буквы, цифры и символ подчеркивания. Название подпрограммы должно начинаться с буквы.

Допускается использование вложенных подпрограмм, а также рекурсивных алгоритмов.

В качестве команд проверки условий используются:

1) команды, проверяющие свободен ли путь (отсутствуют препятствия на пути):

сверху свободно, снизу свободно, справа свободно, слева свободно

top free, bottom free, right free, left free

2) команды, проверяющие не заминирован ли путь:

сверху бомба, снизу бомба, справа бомба, слева бомба

bomb top, bomb bottom, bomb right, bomb left

3) команды, проверяющие текущие координаты робота:

<координата> <знак> <значение>

где <координата> - **строка** или **столбец**, **row** или **col**

<знак> - знак отношения >, <, =,

<значение> - целое число больше 0.

При проверке условий допускается использование логического оператора “**не**”, “**not**”.

Для комментариев используется зарезервированное слово “//”.

Экранная форма обучающей системы разбита на 2 части (рисунок 1).

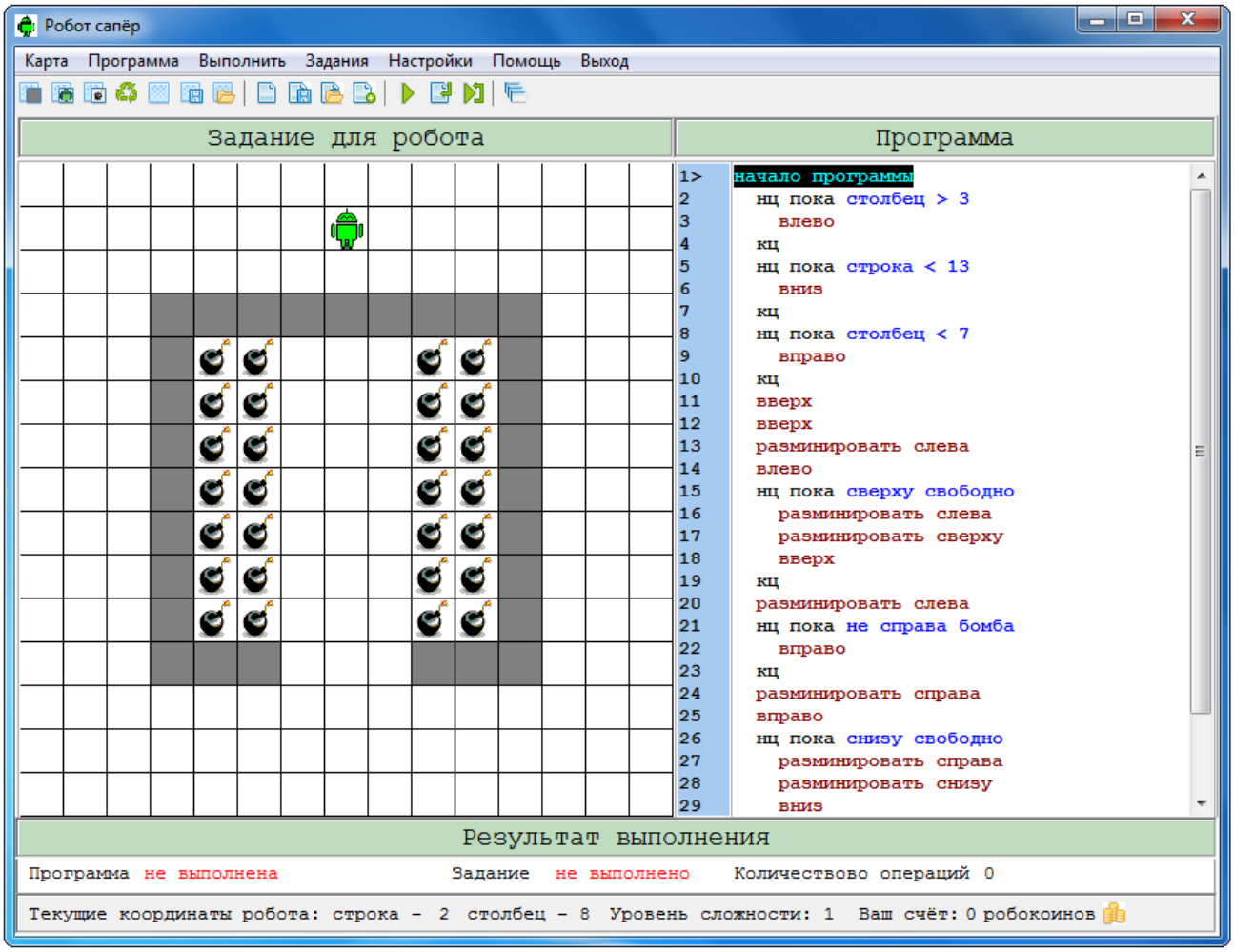

Рисунок 1 - экранная форма обучающей системы

Левая часть содержит задание для робота и включает в себя робота-сапера, лабиринт, по которому должен перемещаться робот и минное поле. Пользователь имеет возможность самостоятельно создать лабиринт, загрузить уже имеющийся (и при желании отредактировать его), сохранить лабиринт.

Правая часть содержит интегрированную среду разработки, включающую в себя:

- текстовый редактор для ввода и редактирования программ, с автоматической подсветкой синтаксиса, системой автодополнения текущих лексем, системой ускоренного ввода команд (рисунок 2);
- компилятор встроенного языка программирования;
- систему автоматизированного тестирования;
- средства отладки.

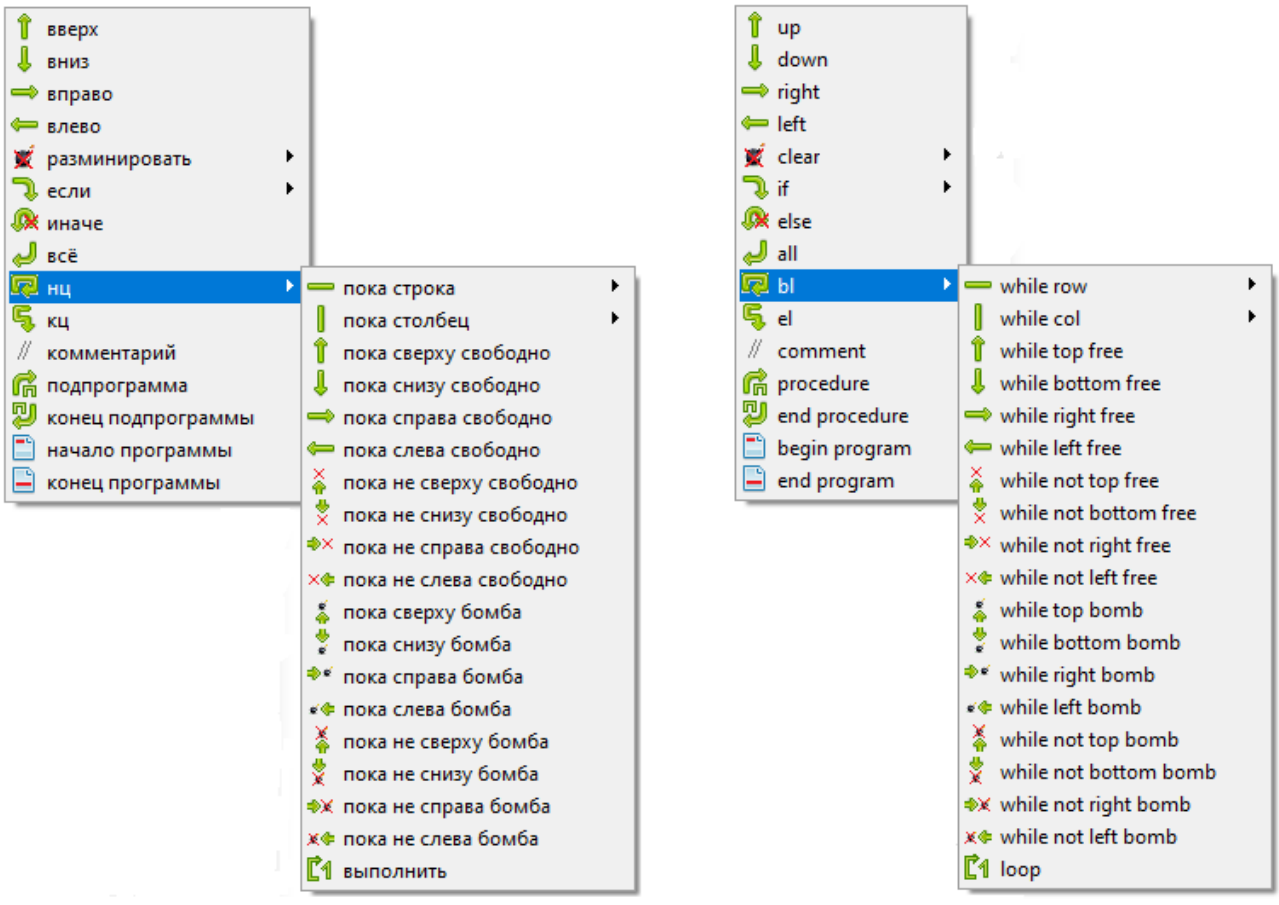

Рисунок 2 - система ускоренного ввода команд

При написании программ для управления роботом пользователь имеет возможность создавать программы с нуля, загружать, редактировать и сохранять уже созданные.

Готовую программу пользователь может запустить на выполнение в обычном режиме, режиме пошаговой отладки, режиме тестирования.

Для проверки усвоенных знаний в системе существует комплекс тестовых заданий (рисунок 3).

В общей сложности обучаемому предстоит решить 118 задач различного уровня сложности. При решении тестовых задач существуют определенные ограничения на размер используемой памяти, что побуждает обучаемого искать наиболее оптимальные алгоритмы решения и на используемые конструкции, что позволяет закрепить строго определенный учебный материал.

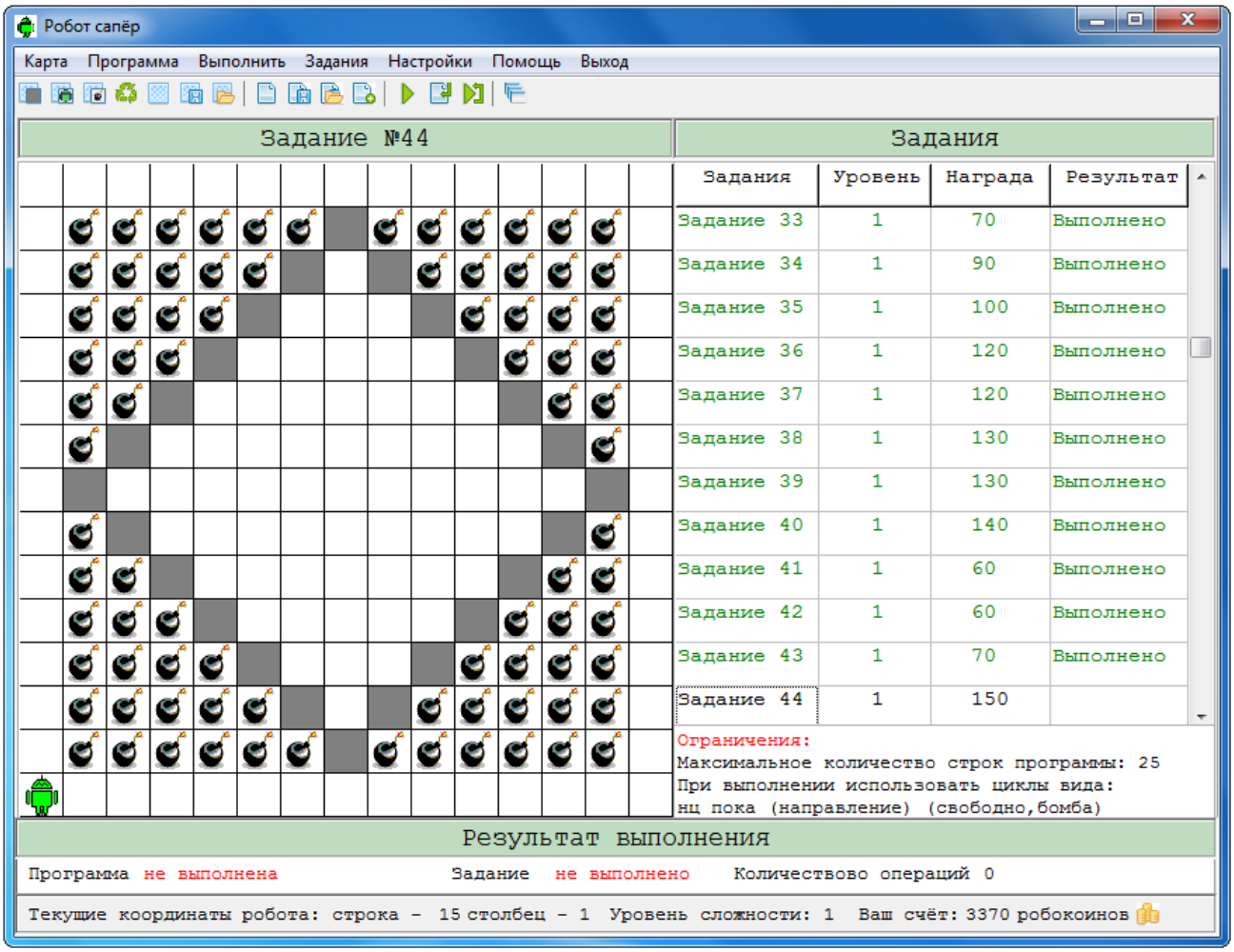

Рисунок 3 - комплекс тестовых заданий

Например, для решения задания №44 из комплекса тестовых заданий (рисунок 3) обучаемому, в соответствии с имеющимися ограничениями, предстоит написать программу, состоящую из вложенных циклических конструкций вида нц пока (направление) (свободно, бомба).

В соответствии с теорией геймификации, основными составляющими, требующими внимания при внедрении гемифицированной системы в неигровую деятельность, являются мотивация и игровые элементы из триады PBL – баллы, бейджи, лидерборды [8]. В соответствии с данной теорией за решение тестовых заданий обучаемый получает некоторое количество местной виртуальной валюты «робокоинов», которое зависит от сложности выполняемых заданий.

В обучающей системе существует 4 базовых уровня сложности.

На 1-м уровне сложности обучаемому предстоит создавать несложные программы, управляющие поведением робота. При этом состояние игровой среды детерминировано.

Цели 1-го уровня сложности:

1) знакомство с обучающей системой;
2) изучение последовательностей команд;
3) изучение различных вариантов циклических конструкций, в том числе вложенных циклов.

Пример программы 1-го уровня сложности для карты, изображенной на рисунке1, приведен в приложении А. Как видно из примера, данная программа обладает небольшим уровнем сложности и состоит из нескольких циклов и последовательностей команд.

На 2-м уровне сложности появляются элементы хаотичного изменения игровой среды в виде случайно возникающих в окружении робота объектов (мин). При этом робот должен постоянно отслеживать изменения окружающей среды и взаимодействовать с возникающими объектами (обезвреживать мины).

Цели 2-го уровня сложности:

1) изучение конструкций ветвления;

2) изучение подпрограмм;

3) увеличение игровой составляющей обучающей системы, что повышает мотивацию школьников;

4) увеличение сложности исполняемых алгоритмов.

Пример программы 2-го уровня сложности для карты, изображенной на рисунке1, приведен в приложении Б. Как видно из примера, сложность программы существенно выросла. Размер программы вырос вдвое. Успешное решение задания требует применения условных конструкций и подпрограмм.

На 3-м уровне сложности изменяется вид возникающих объектов (вместо мин – стены), что принципиально меняет характер взаимодействия робота и объектов- робот должен искать альтернативные пути к достижению цели.

Цели 3-го уровня сложности:

1) закрепление изученных базовых управляющих конструкций;

2) дальнейшее увеличение игровой составляющей проекта (развитие ситуации на игровом поле может идти по абсолютно непредсказуемому сценарию);

3) дальнейшее увеличение сложности исполняемых программ.

Пример программы 3-го уровня сложности для карты, изображенной на рисунке1, приведен в приложении В. Как видно из примера, сложность программы выросла на порядок. Успешное решение задания требует свободного владения основными конструкциями языков программирования, выработанного алгоритмического мышления, навыков алгоритмизации задач.

На 4-м уровне сложности характер возникающих объектов может быть любым. Робот должен анализировать текущую ситуацию, классифицировать возникающие объекты и реагировать должным образом- обходить препятствия, обезвреживать мины. Фактически обучаемому предстоит создать систему

искусственного интеллекта, автоматически управляющую поведением робота в зависимости от изменения окружающей среды.

Для успешного освоения в обучающую систему встроена подробная справка (рисунок 4)

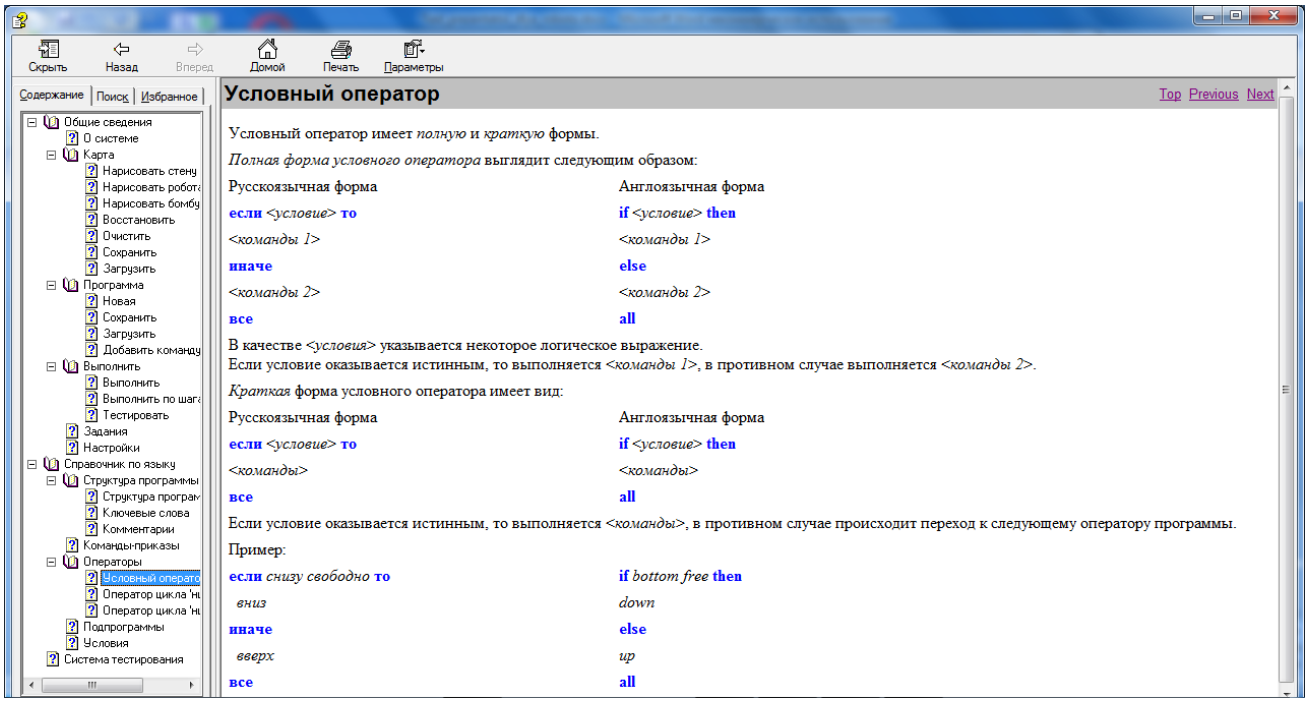

Рисунок 4 - встроенная справка

6 Описание основного алгоритма

Основной алгоритм, реализующий исполнение разработанной обучаемым программы управления роботом состоит из следующих этапов:

1) проверка наличия на карте робота и минного поля. В случае отсутствия выдаются сообщения:

- «Ошибка: на поле отсутствует робот»;

- «Ошибка: на поле отсутствуют мины»;

2) компиляция программы;

2.1) лексический анализ текста программы – распознавание и выделение лексем из входной последовательности символов;

2.2) в случае, если используется англоязычная версия языка производится трансляция в русскоязычную версию;

2.3) синтаксический анализ текста программы – процесс сопоставления линейной последовательности лексем языка с его формальной грамматикой. В случае, если последовательность лексем не соответствует синтаксису языка, выдаются сообщения:

- «Ошибка синтаксиса. Строка № <номер строки>»;

- «Ошибка синтаксиса: имя подпрограммы должно начинаться с буквы»;

2.4) структурирование и определение логических взаимосвязей между управляющими конструкциями программы, проверка правильности расстановки операторов начала и конца управляющих конструкций. Данная проверка осуществляется реализацией программного стека. В случае, если встречается команда начала ветвления или цикла- данная команда добавляется в стек. Если встречается команда окончания ветвления или цикла, данная команда сопоставляется с головным элементом стека. Если элементы представляют собой структуру вида: начало ветвления-конец ветвления или начало цикла-конец цикла, происходит удаление головного элемента стека и чтение следующей команды. В противном случае операторы начала и конца управляющих конструкций расставлены неправильно определяется ошибка (рисунок 5). В случае, если встречается команда конца управляющей конструкции, а стек пуст - определяется ошибка отсутствия начала управляющей конструкции. В случае, если все команды программы проанализированы, а стек не пуст – определяется ошибка отсутствия конца

управляющей конструкции.

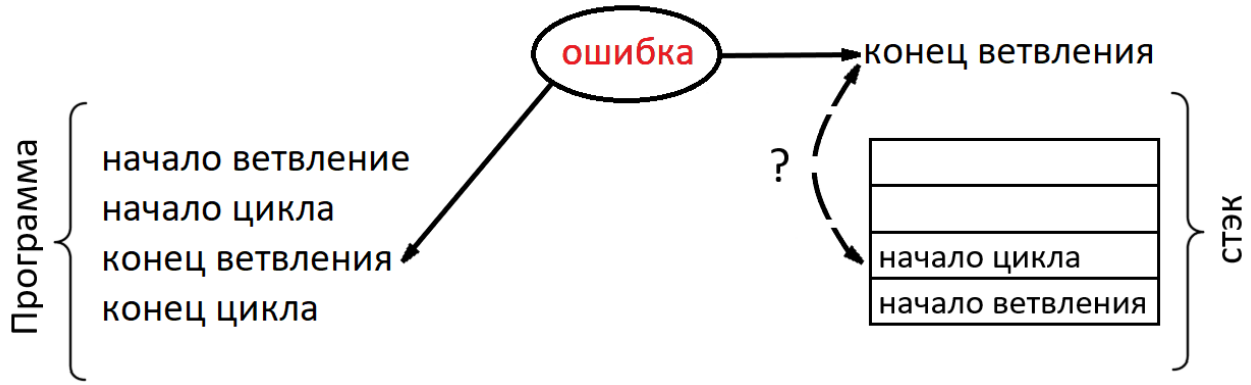

Рисунок 5 - проверка правильности расстановки начала и конца управляющих конструкций

В случае определения ошибки выдаются сообщения:

- «Ошибка: отсутствует команда "если"»;

- «Ошибка: отсутствует команда "всё"»;

- «Ошибка: отсутствует команда "нц"»;

- «Ошибка: отсутствует команда "кц"»;

- «Ошибка: встретилась команда "всё", ожидалась команда "кц". Строка № <номер строки>»;

- «Ошибка: встретилась команда "кц", ожидалась команда " всё ". Строка № <номер строки>»;

- «Ошибка: команда "кц" находиться до команды "нц". Строка № <номер строки>»;

- «Ошибка: команда "всё" находиться до команды "если". Строка № <номер строки>»;

- «Ошибка: отсутствует начало программы»;

- «Ошибка: отсутствует конец программы»;

- «Ошибка: ожидалась подпрограмма или начало программы. Строка № <номер строки>»;

- «Ошибка: отсутствует конец подпрограммы»;

2.5) определение точек входа и выхода в управляющих конструкциях;

В соответствии с теоремой Бема-Якопини [9], любая программа, заданная в виде блок-схемы, может быть представлена в виде трех управляющих структур:

а) последовательность, обозначается

```
BLOCK
  f, g
ENDBLOCK
```

б) ветвление, обозначается

```
IF p THEN
  f
ELSE
  g
ENDIF
```

с) цикл, обозначается

```
WHILE p DO
  f
ENDDO
```

где f, g – блок-схемы с одним входом и одним выходом (рисунок 6),

p – условие,

BLOCK, ENDBLOCK, IF, THEN, ELSE, ENDIF, WHILE, DO, ENDDO – ключевые слова

Точки входа и выхода необходимы для определения, какому оператору передается управление ходом выполнения программы в зависимости от выполнения, либо не выполнения условия.

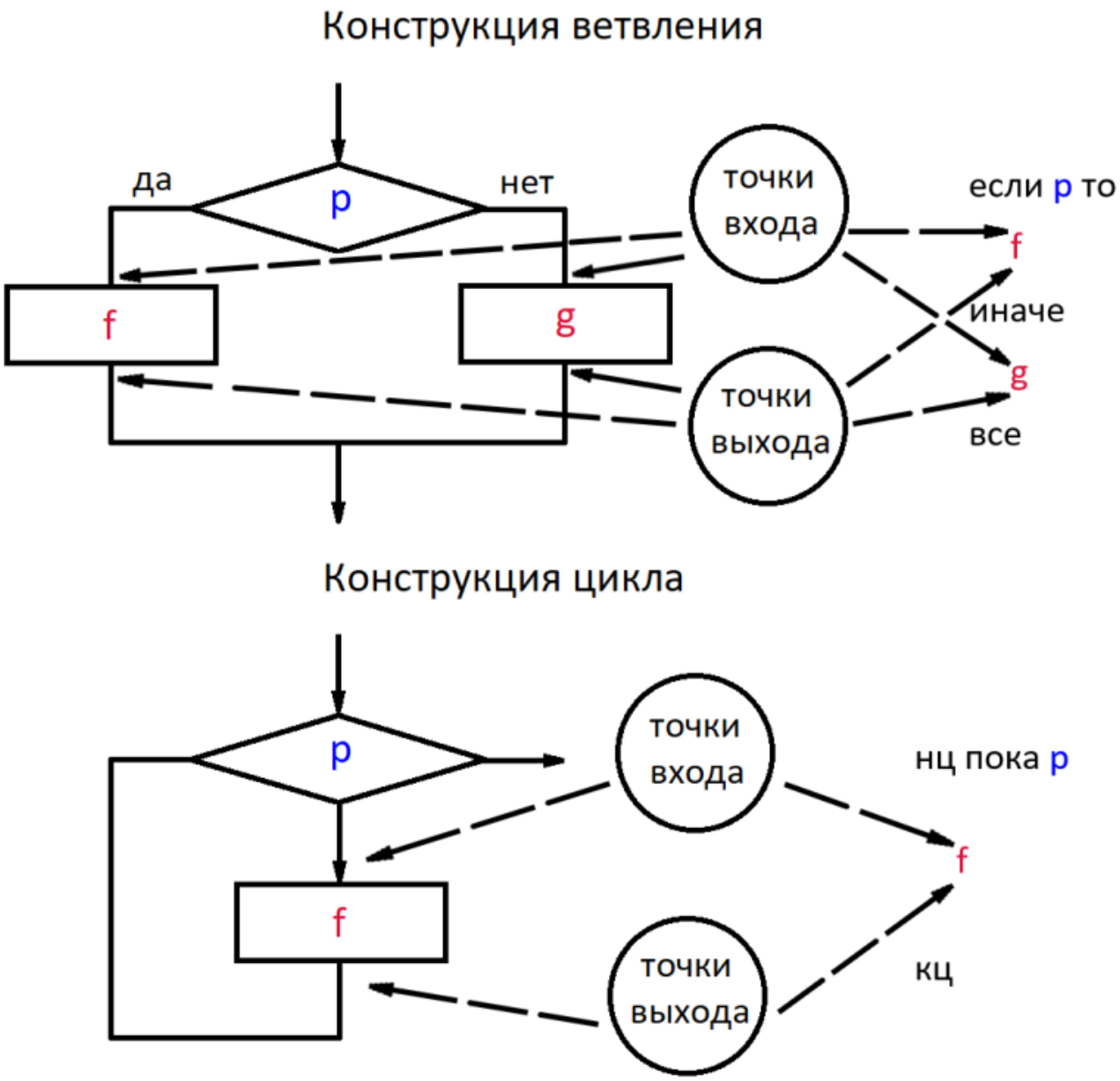

Рисунок 6 - точки входа и выхода в управляющих конструкциях

2.6) компиляция – трансляция программы в исполняемый код;

Исполняемый код сохраняется в массиве, каждый элемент которого состоит из 8 ячеек (рисунок 7). Пример участка программы и соответствующего ей исполняемого кода приведен на рисунке 8.

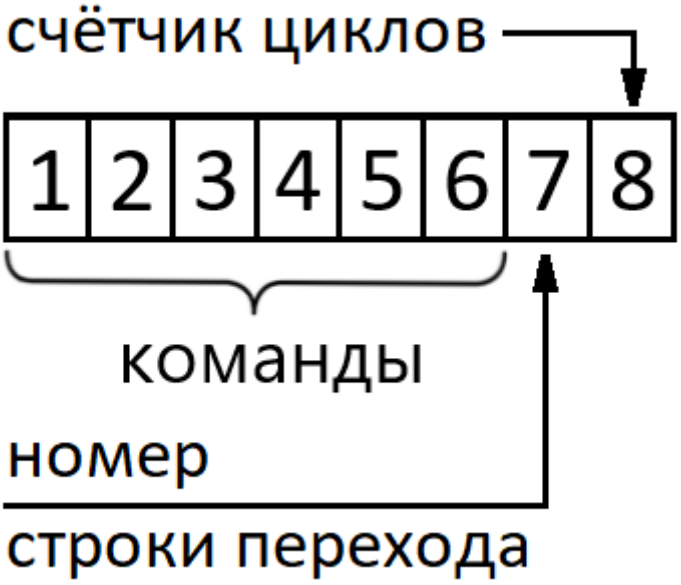

Рисунок 7 - элемент исполняемого кода

Фрагмент программы

```
...
14 если справа свободно то
15   вправо
16 иначе
17   влево
18 всё
...
24 нц выполнить 10 раз
25   вправо
26 кц
```

Фрагмент исполняемого кода

1	2	3	4	5	6	7	8
если	справа	свободно	то			17	
вправо							
иначе						19	
влево							
всё							
нц	выполнить	10	раз			27	10
вправо							
кц						24	

Рисунок 8 - пример исполняемого кода

3) выполнение скомпилированного файла.

В случае использования подпрограмм, для хранения информации о вызовах используется программный стек (рисунок 9). При завершении текущей подпрограммы происходит передача управления команде, номер которой указан в головном элементе стека и удаление данного элемента.

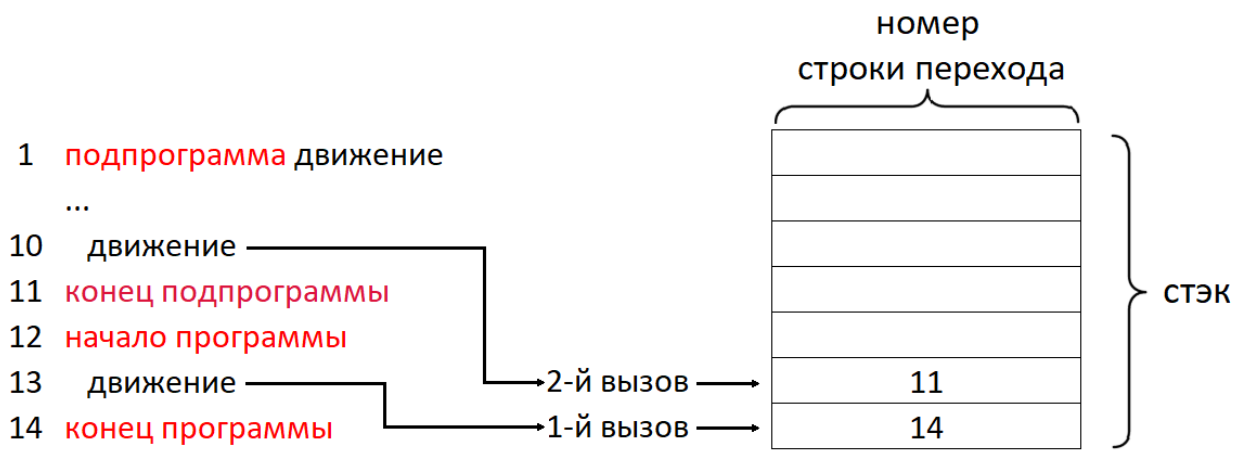

Рисунок 9 - выполнение подпрограмм

При выполнении скомпилированного файла происходит отслеживание ошибок выполнения. В случае обнаружения ошибок выводятся соответствующие сообщения:

- «Ошибка: робот наступил на мину»;
- «Ошибка: робот врезался в стену»;
- «Ошибка: попытка разминирования за границами поля»;
- «Ошибка: попытка разминирования стены»;
- «Ошибка: попытка разминирования пустого поля»;
- «Ошибка: зацикливание программы» (система автоматически отслеживает зацикливание выполняемой программы, прерывает выполнение и выдает сообщение об ошибке).

При реализации данного проекта использовалась интегрированная среда разработки Delphi, как одно из лучших средств для создания десктопных приложений под Windows с графическим интерфейсом пользователя.

7 Результаты применения обучающей системы

С целью проверки эффективности разработанной системы было проведено обучение основам программирования школьницы 10 лет (закончила 4 класса общеобразовательной школы, успеваемость низкая, средняя оценка по математике за 4-й класс 2,67 балла). В общей сложности было проведено 16

занятий. В ходе проведенных занятий школьница освоила базовые управляющие конструкции языков программирования и успешно решила все тестовые задания 1 и 2 уровня сложности.

После изучения данного курса школьнице было предложено решить задания из ОГЭ по информатике для выпускников 9-го класса. С решением данных заданий школьница успешно справилась.

8 Перспективы дальнейшего развития

Система постоянно совершенствуется, расширяется её функционал. В данный момент работа ведется по следующим направлениям:

1) расширение базы тестовых заданий;
2) проверка синтаксиса введенных команд в режиме реального времени;
3) автоматическое структурирование текста программы;
4) выполнение программы в режиме отладки до точки прерывания;
5) пошаговая отладка программы без захода в подпрограмму.

В качестве перспективы дальнейшего развития системы рассматриваются следующие варианты:

1) создание сетевой версии с возможностью соревнования программистов:
   - большая карта;
   - 2 участника, 2 робота, каждый участник управляет своим роботом;
   - каждый участник имеет доступ к координатам робота противника и может препятствовать его продвижению, создавая вокруг робота различные объекты;
   - цель - обезвредить наибольшее количество мин.
2) создание сайта с реализацией встроенного языка программирования на стороне клиента.

3) разработка учебно-методического робототехнического комплекса с реальными, физически существующими, роботом, полем и хаотично возникающими объектами. Работа в данном направлении уже ведется и проект уже частично реализован.

9 Заключение

В ходе работы над данным проектом была разработана система обучения детей основам программирования, включающая в себя:

9.1) игровую подсистему;

9.2) развитую IDE (интегрированную среду разработки), в том числе:

- текстовый редактор;
- компилятор учебного языка;
- систему автоматизированного тестирования;
- средства отладки;

9.3) подсистему тестирования полученных знаний.

Благодаря реализации новых подходов к функционированию обучающей системы, данная система полностью соответствует заявленным требованиям. Цели и задачи, которые были поставлены при реализации данного проекта выполнены в полном объеме.

Система успешно прошла апробацию и может быть рекомендована к использованию в образовательном процессе в качестве составной части методики раннего обучения школьников навыкам программирования.

Список использованных источников

1. Федеральный закон от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации»

2. Николов Р., Сендова Е. Начала информатики. Язык Лого / Под ред. Б. Сендова, Пер. с болг. Под ред. А. В. Гиглавого. М.: Гл. ред. физ.-мат. лит., 1989

3. https://scratch.mit.edu/

4. https://www.niisi.ru/kumir/index.htm

5. Гнатюк О. Л. Основы теории коммуникации. — М.: КНОРУС, 2010. — 256 с.

6. Зиновкина М. М. Педагогическое творчество / Модульно-кодовое учебное пособие. — М.: МГИУ, 2007. — 258 с.

7. Дал У., Дейкстра Э., Хоор К. Структурное программирование. — М.: Мир, 1975. — 247с.

8. Вовлекай и властвуй. Игровое мышление на службе бизнеса / Кевин Вербах, Дэн Хантер: Манн, Иванов и Фербер; Москва; 2015 - 160с.

9. Harlan D. Mills Mathematical Foundations for Structured Programming. — February 1972. — Federal Systems Division. IBM Corporaton. Gaithersburg, Maryland.

Приложение А. Пример программы 1-го уровня сложности.

```
1  начало программы
2    нц пока столбец > 3
3      влево
4    кц
5    нц пока строка < 13
6      вниз
7    кц
8    нц пока столбец < 7
9      вправо
10   кц
11   вверх
12   вверх
13   разминировать слева
14   влево
15   нц пока сверху свободно
16     разминировать слева
17     разминировать сверху
18     вверх
19   кц
20   разминировать слева
21   нц пока не справа бомба
22     вправо
23   кц
24   разминировать справа
25   вправо
26   нц пока снизу свободно
27     разминировать справа
28     разминировать снизу
```

29 **вниз**

30 **кц**

31 **разминировать справа**

32 **конец программы**

Приложение Б. Пример программы 2-го уровня сложности.

```
1  подпрограмма разминировать_вокруг
2   если сверху бомба то
3    разминировать сверху
4   всё
5   если снизу бомба то
6    разминировать снизу
7   всё
8   если справа бомба то
9    разминировать справа
10  всё
11  если слева бомба то
12   разминировать слева
13  всё
14 конец подпрограммы
15 начало программы
16  нц пока столбец > 3
17   разминировать_вокруг
18   влево
19  кц
20  нц пока строка < 13
21   разминировать_вокруг
22   вниз
23  кц
24  нц пока столбец < 7
25   разминировать_вокруг
26   вправо
27  кц
```

```
28  нц пока строка > 11
29    разминировать_вокруг
30    вверх
31  кц
32  разминировать_вокруг
33  влево
34  нц пока сверху свободно
35    разминировать_вокруг
36    вверх
37  кц
38  разминировать_вокруг
39  нц пока столбец < 9
40    разминировать_вокруг
41    вправо
42  кц
43  разминировать_вокруг
44  вправо
45  нц пока снизу свободно
46    разминировать_вокруг
47    вниз
48  кц
49  разминировать_вокруг
50 конец программы
```

Приложение В. Пример программы 3-го уровня сложности.

```
1   начало программы
2    нц пока столбец > 3
3     если слева свободно то
4      влево
5     иначе
6      если не снизу свободно то
7       вверх
8      иначе
9       вниз
10     все
11    все
12   кц
13   нц пока строка < 13
14    если снизу свободно то
15     вниз
16    все
17    если строка < 13 то
18     если не снизу свободно то
19      если слева свободно то
20       влево
21       если снизу свободно то
22        вниз
23       иначе
24        влево
25       все
26       если снизу свободно то
27        вниз
28       все
```

```
29          если справа свободно то
30            вправо
31          все
32        все
33        если столбец = 1 то
34          если справа свободно то
35            вправо
36          все
37        все
38      все
39    все
40  кц
41  нц пока столбец < 7
42    если справа свободно то
43      вправо
44    иначе
45      если снизу свободно то
46        вниз
47      все
48    все
49  кц
50  нц пока строка > 11
51    если сверху свободно то
52      вверх
53    иначе
54      вправо
55      если сверху свободно то
56        вверх
57      все
58    все
```

```
59  кц
60  нц пока столбец > 7
61   если слева свободно то
62    влево
63   иначе
64    если сверху свободно то
65     вверх
66    все
67   все
68  кц
69  разминировать слева
70  влево
71  если строка = 9 то
72   разминировать снизу
73   вниз
74  все
75  если строка = 10 то
76   разминировать снизу
77   вниз
78   разминировать слева
79   вверх
80  все
81  нц пока сверху свободно
82   разминировать слева
83   если сверху бомба то
84    разминировать сверху
85   все
86   вверх
87  кц
88  разминировать слева
```

```
89   нц пока столбец < 9
90    если справа свободно то
91     вправо
92    иначе
93     если снизу свободно то
94      вниз
95     иначе
96      влево
97     все
98    все
99   кц
100  разминировать справа
101  вправо
102  нц пока снизу свободно
103   разминировать снизу
104   вниз
105  кц
106  разминировать справа
107  вправо
108  нц пока сверху свободно
109   разминировать сверху
110   вверх
111  кц
112  если слева бомба то
113   разминировать слева
114   влево
115  все
116  нц пока снизу бомба
117   разминировать снизу
118   вниз
```

```
119    кц
120  конец программы
```