УДК 519.682: 372.8+004.421:372.8

ПИКТОГРАММНЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ «ПИКТО»

Н. О. Бесшапошников, А. Г. Леонов
Федеральный научный центр
Научно-исследовательский институт системных исследований РАН,
nbesshaposhnikov@vip.niisi.ru, dr.l@vip.niisi.ru

Одним из важнейших способов восприятия информации является визуальный. Схематичное изображение действия или предмета можно использовать как реализацию самого действия или воплощение предмета. Человек может воспринимать данные изображения независимо от родного языка. Поэтому схематичные изображения, или иначе пиктограммы, иконки, можно использовать как элементы языка программирования, для изучения которого даже не обязательно уметь писать или читать. Ограниченность в наборе синтаксических конструкций и размерах тела программы такого языка позволяет избежать множество трудностей обычных текстовых языков программирования, таких как: синтаксически неверные программы, знание иностранного языка (в основном, английского), немалое количество материала для изучения. Посредством таких ограничений порог входа для изучения пиктографического языка довольно низок, и это позволяет использовать его для обучения детей дошкольного возраста. В данной работе рассмотрена реализация такого пиктографического языка – Пикто. Система ПиктоМир используется как часть курса обучения алгоритмическому программированию в дошкольных и начальных классах. Следующей частью курса для средних и старших классов является школьный алгоритмический язык (система КуМир). В рамках реализации описана контекстно-свободная и нерегулярная грамматика для языка, а также процесс компиляции и синтаксический анализ программ, составленных на нем, произведено сравнение с другим пиктографическим языком программирования Lightbot.

Ключевые слова: ПиктоМир, язык программирования Пикто, контекстно свободная грамматика, компиляция, пиктограммы, дошкольник, младшеклассник**.**

PICTOGRAPHIC PROGRAMMING LANGUAGE “PIKTO”

N. O. Besshaposhnikov, A. G. Leonov
System Research Institute, Russian Academy of Sciences,
nbesshaposhnikov@vip.niisi.ru, dr.l@vip.niisi.ru

One of the most important ways of perceiving information is visual. A schematic depiction of an action or object can be used as the implementation of the action itself or the embodiment of the object. A person can perceive the image data regardless of the native language. Therefore, as part of the programming language, you can use schematic images, as well as icons, learning which you do not even need know how to write or read. Limitedness in the set of syntactic constructions and the size of the body code of such language allows avoiding many difficulties of conventional textual programming languages, such as: syntactically incorrect programs, the need to know a foreign language (mainly English) and a considerable amount of material for study. By means of such restrictions, the entry threshold for studying pictographic language is quite low, and this allows it to be used for teaching preschool children. The article discusses the implementation of such pictographic language that is Pikto. The PiktoMir system is used as part curriculum of algorithmic programming in preschool and elementary classes. The next part of the course for middle and senior schools is the school algorithmic language (the KuMir system). The implementation describes the

context-free and irregular grammar for the language, as well as the compilation process and the syntactic parsing of programs compiled on it. The paper also compares Pikto with another pictographic programming language Lightbot.

Keywords: PiktoMir, Pikto programming language, context-free grammar, compilation, pictographs, preschooler, primary school pupil.

В основе символизации лежит процесс моделирования, создание некоторого образа, который в существенных чертах соответствует моделируемому предмету или процессу. В древности наскальной живописью первобытный человек стремился сделать картинки максимально узнаваемыми его соплеменниками. Основоположник семиотики, Чарльз Сандерс Пирс, в конце XIX в. классифицировал знаки по типам отношений между внешней стороной знака – означающим, и внутренней его стороной – означаемым. Знаки, основанные на фактическом сходстве, Пирс назвал иконическими, или иконами [1]. Пиктограмма – знак (иконка), отображающий важнейшие узнаваемые черты объекта, предмета или явления, на которые он указывает. Пиктограммы, как элементы графического интереса – иконки, были внедрены в 1970 г. в исследовательском центре PARC компании Xerox для облегчения работы с компьютером[2–3].

Одним из примеров пиктографических языков является язык в игре Lightbot [4–5]. В данной игре Робот с помощью команд «Вперед», «Подпрыгнуть», «Налево», «Направо», и «Зажечь» двигается по полю и «зажигает» выделенные клетки поля. Язык Робота довольно просто устроен – он позволяет составлять только линейные программы с возможность вызова линейных же подпрограмм. Грамматика языка Lightbot является регулярной и может быть задана алфавитом Σ = {«Вперед», «Подпрыгнуть/Спрыгнуть», «Налево», «Направо», «Зажечь», «Подпрограмма 1», «Подпрограмма 2», I}, где I – набор обозначений начала подпрограммы (так же в алфавите могут быть аналогичные процедуры, которые выполняются только на определенных клетках) и двумя правилами вывода:

1. ПОДПРОГРАММА $\longrightarrow$ e | $e \in \Sigma$
2. ПОДПРОГРАММА $\longrightarrow$ e ПОДПРОГРАММА | $e \in \Sigma$
3. ПРОГРАММА $\longrightarrow i$ ПОДПРОГРАММА | $i \in$ I
4. ПРОГРАММА $\longrightarrow$ ПРОГРАММА i ПОДПРОГРАММА | $i \in$ I

Циклы в языке реализуются с помощью рекурсивного вызова подпрограммы. Условия возможны только на выполнение конкретной команды Робота.

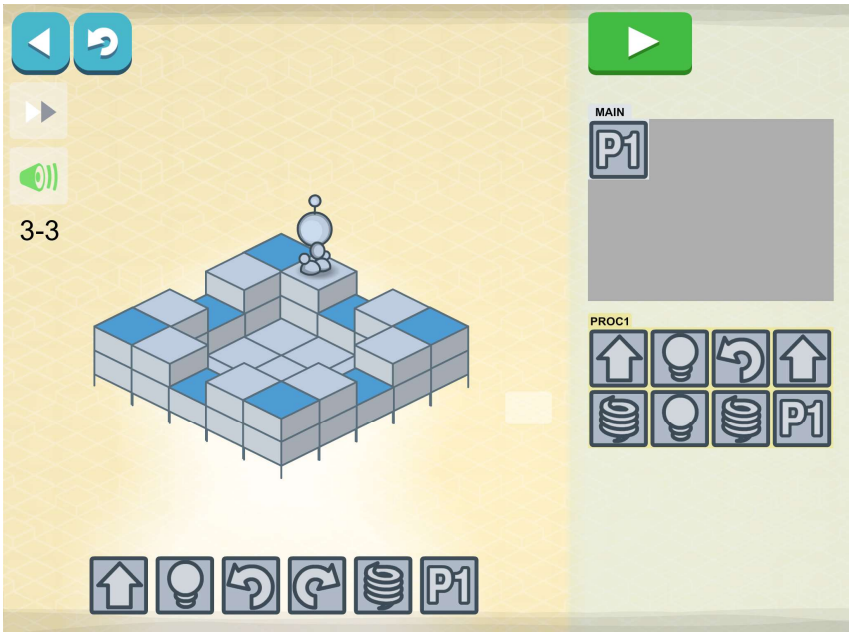

Рис. 1. Программа в Lightbot

Рассмотрим на примере рис. 1 синтаксический разбор программы (для краткости записи: ’В’ – «Вперед», ‘П’ – «Подпрыгнуть/Спрыгнуть», ‘З’ – «Зажечь», ‘Л’ – «Налево»):

- ПРОГРАММА (3) ⟶ *‘MAIN’* ПОДПРОГРАММА

- *MAIN’* ПОДПРОГРАММА (4) ⟶ *‘MAIN’* ПОДПРОГРАММА *‘PROC1’* ПОДПРОГРАММА

- *MAIN’* ПОДПРОГРАММА *‘PROC1’* ПОДПРОГРАММА (2) ⟶ *‘MAIN’* ‘П’ ПОДПРОГРАММА *‘PROC1’* ПОДПРОГРАММА,

применяя несколько раз Правило 2.

- ‘*MAIN’* ПОДПРОГРАММА ‘*PROC1’* ‘В’ ‘З’ ‘Л’ ‘В’ ‘П’ ‘З’ ‘П’ ПОДПРОГРАММА (1) ⟶ ‘*MAIN’* ПОДПРОГРАММА ‘*PROC1’* ‘В’ ‘З’ ‘Л’ ‘В’ ‘П’ ‘З’ ‘П’ ‘Р1’

- ‘*MAIN’* ПОДПРОГРАММА ‘*PROC1’* ‘В’ ‘З’ ‘Л’ ‘В’ ‘П’ ‘З’ ‘П’ ‘Р1’ (1) ⟶ ‘*MAIN’* ‘Р1’ ‘*PROC1’* ‘В’ ‘З’ ‘Л’ ‘В’ ‘П’ ‘З’ ‘П’ ‘Р1’.

Язык программирования системы ПиктоМир. Система ПиктоМир бестекстового, пиктограммного программирования позволяет ребенку «собрать» из пиктограмм на экране компьютера несложную программу, управляющую виртуальными исполнителями-роботами. ПиктоМир, в первую очередь, ориентирован на дошкольников, еще не умеющих писать, или на младшеклассников, не очень любящих писать [6–7].

Язык Пикто – это пиктографический язык программирования, используемый в системе ПиктоМир. Особенность языка в том, что любая составленная в нем программа, является синтаксически верной. Существует расширение этого языка, позволяющее «безошибочный» ввод арифметических и логических выражений. Данный язык содержит такие синтаксические конструкции, как: подпрограммы, деление подпрограммы на блоки, Цикл «Пока верно условие», цикл «повторить *N* раз», условное выполнение части или всего блока. С помощью данного языка пользователь управляет Роботом на экране (а также аналогом переменной – Кувшином) для достижения цели Роботом, например, починить все клетки на поле [8].

Выполнение программы начинается с главной «безымянной» подпрограммы алгоритма, не отмеченной никакой буквой. В зависимости от сложности задачи, пользователю также доступны подпрограммы А, Б, В (а также, при необходимости, Г, Д, Е… и т. д.). Подпрограммы поделены на один или более блоков, каждый из которых может содержать цикл (кружок, в который может быть поставлено число от 1 до 6, условие, или значение, вычисляемое в процессе работы программы) или условие (ромб) для его выполнения. Также доступна конструкция «Если, то иначе», внутри которой есть два блока. Каждый блок состоит из квадратов-слотов, в которые можно поставить либо метод Робота или Кувшина, либо вызов подпрограммы.

Грамматика языка Пикто является контекстно-свободной, но не регулярной. Она может быть описана следующим образом:

Терминальный алфавит $\Sigma = \{R, C, F, I, BS, BE\}$, где R – это набор из повторителей (циклов) 1–6, также повторитель «Пока условие» и повторители Кувшина; C – набор условий для конкретного Робота, в том числе и пустое условие; I – набор обозначений начала подпрограммы; *BS, BE* – обозначения начала и конца блока соответственно; F – набор методов Робота, а также обозначение подпрограмм – А, Б, В, … .

Нетерминальный алфавит: {‘БЛОК’, ‘БЛОК С ПОВТОРИТЕЛЕМ’, ‘ПОДПРОГРАММА’, ‘ПРОГРАММА’}.

Начальный нетерминал: БЛОК.

Правила вывода:

1. ‘БЛОК’ $\longrightarrow f \mid f \in F$
2. ‘БЛОК’ $\longrightarrow f$ ‘БЛОК’ $\mid f \in F$
3. ‘БЛОК’ $\longrightarrow c$ ‘БЛОК’ $\mid c \in C$
4. ‘БЛОК С ПОВТОРИТЕЛЕМ’ $\longrightarrow r$ ‘БЛОК’ $\mid r \in R$
5. ‘ПОДПРОГРАММА’ $\longrightarrow$ *BS* ‘БЛОК’ *BE*

6. ‘ПОДПРОГРАММА’
7. ММА’ ⟶ *BS* ‘БЛОК С ПОВТОРИТЕЛЕМ’ *BE*‘ПОДПРОГРАММА’ ⟶ ‘ПОДПРОГРАММА’ *BS* ‘БЛОК’ *BE*
8. ‘ПОДПРОГРАММА’⟶ ‘ПОДПРОГРАММА’*BS* ‘БЛОК С ПОВТОРИТЕЛЕМ’*BE*
9. ‘ПРОГРАММА’ ⟶ *i* ‘ПОДПРОГРАММА’ | $i \in$ I
10. ‘ПРОГРАММА’ ⟶ ‘ПРОГРАММА’ *i* ‘ПОДПРОГРАММА’ | $i \in$ I

Таблица 1

Типы инструкций

Тип	Описание
EXECUTE	Выполняет метод
CHECK_CONDITION	Выполняет проверку условия Робота
START_LOOP	Является индикатором начала цикла, задает начальное значение итератору цикла
END_LOOP	После тела цикла, проверяет достиг ли итератор цикла значения повторителя
RETURN	Выполняет выход из функции, иначе: получает из стека позицию в программе во время вызова этой функции
JUMP_IF	Переход по ссылке, если условие верно
JUMP_NIF	Переход по ссылке, если условие неверно
JUMP	Безусловный переход по ссылке

Для синтаксического разбора языка Пикто используется *LL(1)* синтаксический анализатор. Ввиду того, что синтаксическая диаграмма программы уже составлена заранее и, соответственно, задает заведомо корректную программу, синтаксический разбор совмещен с процессом однопроходной интерпретации кода в набор инструкций. Получаемые инструкции перечислены в табл. 1.

Процесс интерпретации программы выглядит следующим образом:

1. Процесс компиляции блока:
    - если блок с обычным повторителем
    - создать ссылку *blockEndLabel*
    - создать инструкцию *START_LOOP* (переход по *blockEndLabel*)
    - положить инструкцию в список
    - присвоить ссылке *loopBodyStartLabel* текущий размер списка
    - цикл по всем методам:
        - создать инструкцию выполнения метода *EXECUTE*
        - положить инструкцию в список
    - присвоить ссылке *loopEndLabel* текущий размер списка
    - создать инструкцию *END_LOOP* (переход по *loopBodyStartLabel*)
    - положить инструкцию в список
    - присвоить ссылке *blockEndLabel* текущий размер списка
- если блок с условиями:
    - создать ссылку *condEndLabel*
    - цикл по высоте блока

- если создана ссылка *blockEndLabel*
  - присвоить ссылке *subBlockLabel* текущий размер списка
  - создать инструкцию *JUMP* (переход по *condEndLabel*)
  - положить инструкцию в список
- если *foundElse* прервать цикл
- если на данной строке есть условие, и оно не пустое
  - создать инструкцию *CHECK_CONDITION*
  - положить инструкцию в список
  - создать ссылку *subBlockLabel*
  - создать инструкцию *JUMP_NIF* (переход по *subBlockLabel*)
  - положить инструкцию в список
- если на данной строке есть условие, и оно пустое, то установить флаг *foundElse = true*
- цикл по ширине блока
- если метод на данной высоте и ширине не пустой
  - создать инструкцию выполнения метода *EXECUTE*
  - положить инструкцию в список

• если не *foundElse* и создана ссылка *subBlockLabel*
- присвоить ссылке *subBlockLabel* текущий размер списка

• если *foundElse*
- присвоить ссылке *condEndLabel* текущий размер списка

• если блок с обычным повторителем
- создать ссылку *blockStartLabel*
- создать инструкцию *CHECK_CONDITION*
- положить инструкцию в список
- создать инструкцию *JUMP_NIF* (переход по *blockEndLabel*)
- цикл по всем методам:
  - создать инструкцию выполнения метода *EXECUTE*
  - положить инструкцию в список
- создать инструкцию *JUMP* (переход по *blockStartLabel)*
- присвоить ссылке *blockEndLabel* текущий размер списка

• если обычный блок
- цикл по всем методам:
  - создать инструкцию выполнения метода *EXECUTE*
  - положить инструкцию в список.

2. Процесс компиляции алгоритма:
- присвоить ссылке на ID текущего алгоритма текущий размер списка
- цикл по количеству блоков
  - скомпилировать блок
- создать инструкцию *RETURN*
- положить инструкцию в список

Процесс компиляции программы состоит из последовательной компиляции всех алгоритмов.

Рис. 2. Программа в ПиктоМире

Сделаем синтаксический разбор программы Пикто на рис. 2:

• ПРОГРАММАМА (9) ⟶ *‘ГЛ П’* ПОДПРОГРАММА

• *‘ГЛ П’* ПОДПРОГРАММА (10) ⟶ *‘ГЛ П’* ПОДПРОГРАММА ‘А’ ПОДПРОГРАММА

• *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ ПОДПРОГРАММА (6) ⟶ *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS* ‘БЛОК С ПОВТОРИТЕЛЕМ’ *BE*

• *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS* ‘БЛОК С ПОВТОРИТЕЛЕМ’ *BE* (4) ⟶ *‘ГЛ П’* ПОДПРОГРАММА *BS* ‘ПП А’  ‘БЛОК’ *BE*

• *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS*  ‘БЛОК’ *BE* (2) ⟶ *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS*  ‘БЛОК’ *BE*

• *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS*  ‘БЛОК’ *BE* (1) ⟶ *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS*  *BE*

• *‘ГЛ П’* ПОДПРОГРАММА ‘ПП А’ *BS*  *BE* (7,8,7,6,4,4) ⟶*ГЛ П’ BS* ‘БЛОК’ *BE BS*  ‘БЛОК’ *BE BS* ‘БЛОК’ *BE BS*  ‘БЛОК’ *BE* ‘ПП А’ *BS*  *BE*

• *ГЛ П’ BS* ‘БЛОК’ *BE BS*  ‘БЛОК’ *BE BS* ‘БЛОК’ *BE BS*  ‘БЛОК’ *BE* ‘ПП А’ *BS*  *BE* (Несколько раз 1, 2) ⟶ Программа на рис. 2.

Набор инструкций, получаемых после компиляции данной программы, представлены в табл. 2.

Таблица 2

Результат компиляции программы

Метки	Инструкция
MAIN	*EXECUTE*
	EXECUTE
	EXECUTE
	EXECUTE
	EXECUTE
	EXECUTE
	EXECUTE
	EXECUTE
BS_1	*CHECK_CONDITION*
	JUMP_NIF (переход по ссылки *BE_1*)
	EXECUTE
	EXECUTE
	JUMP (переход по ссылки BS_1)
BE_1	*EXECUTE*
BS_2	*CHECK_CONDITION*
	JUMP_NIF (переход по ссылки *BE_2*)
	EXECUTE
	JUMP (переход по ссылки BS_2)
BE_2	*RETURN*
A, BS_3	*CHECK_CONDITION*
	JUMP_NIF (переход по ссылки *BE_3*)
	EXECUTE
	EXECUTE
	JUMP (переход по ссылки BS_3)
BE_3	*RETURN*

ПиктоМир, как педагогический программный продукт, направлен на пропедевтику использования алгоритмического языка программирования [9–10]. В системе КуМир [11–15] использован школьный алгоритмический язык с русской лексикой и встроенными исполнителями Робот и Чертежник. Программы, написанные школьником на алгоритмическом языке, уже не всегда будут синтаксически корректны. Но среда программирования КуМир позволяет вставлять управляющие конструкции и имена величин и алгоритмов целиком, чтобы обучаемый смог избежать большинства синтаксических ошибок. Для облегчения перехода от

пиктограммного стиля программирования разработано расширение языка Пикто и система ПиктоМир-К, позволяющее безошибочный ввод простейших программ управления роботами-исполнителями в тестовой форме [5].

Настоящая работа выполнена по теме госзадания 2017 г. (№ 0065-2015-0105).

Литература

1. Пирс Ч. С. Что такое знак? // Вестн. Томск. гос. ун-та. Сер. Философия. Социология. Политология. 2009. № 3 (7). С. 88–95.

2. Kay A. Microelectronics and the Personal Computer // Scientific American. 1977. Sept. V. 237. Is. 3. P. 230–244.

3. Раскин Д. Интерфейс: новые направления в проектировании компьютерных систем. М. : Символ-Плюс, 2005.

4. Lightbot. URL: http://lightbot.com (дата обращения: 01.10.2017).

5. Lightbot programming puzzles android apps on google play. URL: http://topbestbook.info/lightbot/lightbot-programming-puzzles-android-apps-on-google-play.htm (дата обращения: 01.10.2017).

6. Rogozhkina I. B., Kushnirenko A. G., PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment // Procedia – Social and Behavioral Sciences. 2011. V. 28. P. 601–605.

7. Леонов А. Г. Логическое проектирование педагогических программных средств // Ярослав. пед. вестн. Т. III. Естеств. науки. 2013. № 4. С. 134–141.

8. Леонов А. Г., Бесшапошников Н. О., Ерёмин Д. Б., Дедков А. Н. ПиктоМир для планшетных компьютеров как инструмент пропедевтического курса информатики // Тр. Большого Московского семинара по методике раннего обучения информатике. М., 2014. Т. 1. № 4.

9. Кушниренко А. Г., Леонов А. Г. Программирование для дошкольников и младших школьников // Первое сентября. Информатика. 2011. № 15. С. 20–23.

10. Бесшапошников Н. О. Дедков А. Н., Еремин Д. Б., Леонов А. Г. ПиктоМир как кооперативная среда для обучения основам программирования дошкольников и младших школьников // Тр. НИИСИ РАН. 2015. Т. 5. № 1. С. 138–141.

11. Ершов А. П., Кушниренко А. Г., Лебедев Г. В., Семенов А. Л., Шень А. Х. Пробный учебник для средних учебных заведений. М. : Просвещение, 1988. 207 с.

12. Кушниренко А. Г., Лебедев Г. В., Сворень Р. А. Основы информатики и вычислительной техники : пробн. учеб. для сред. учеб. заведений. М. : Просвещение, 1996.

13. Кушниренко А. Г., Лебедев Г. В., Зайдельман Я. Н. Информатика. 7–9 кл. М. : Дрофа, 2003.

14. Поляков К. Ю. Практикумы в системе КуМир. URL: http://kpolyakov.spb.ru/download/kumkurs.pdf, 2017 (дата обращения: 01.10.2017).

15. КуМир. URL: https://www.niisi.ru/kumir/ (дата обращения: 01.10.2017).