



Р. Б. Воронов

*Удельнинская гимназия, п. Удельная, г.о. Раменский, Московская область, Россия;
Методический центр «Раменский дом учителя», г. Раменское, Московская область, Россия*

ПСЕВДОГРАФИКА СРЕДСТВАМИ PASCAL И PYTHON КАК НАГЛЯДНЫЙ ИНСТРУМЕНТ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ*

Аннотация

В статье приводится пример представления обучающимся такого сложного материала, как создание программ с вложенными циклами и ветвлением внутри них на языках программирования Pascal или Python. Псевдографика по сравнению с другими обучающими материалами, с одной стороны, дает большую наглядность работы программ с вложенными циклами вперемежку с ветвлением, а с другой — прививает учащимся творческий подход к решению сложных задач при написании программ. Также статья наглядно демонстрирует через псевдографику межпредметную связь информатики с математикой и технологией. Знание из алгебры уравнений для построения различных линий на плоскости позволяет создавать эти «линии» в псевдографике, начиная от прямых линий и окружностей, парабол и гипербол и заканчивая овалами и синусоидами разных типов и размеров. Построенные с помощью псевдографики, эти красивые фигуры можно вывести не только на экран, но и на принтер, например, для создания аппликации на уроках технологии.

Статья будет полезна учителям информатики, студентам педагогических вузов и всем, кто интересуется программированием на Pascal и Python.

Ключевые слова: обучение программированию, Python, Pascal, псевдографика, вложенные циклы.

Информация об авторе

Воронов Роман Борисович, учитель информатики, Удельнинская гимназия, п. Удельная, г.о. Раменский, Московская область, Россия; методист, преподаватель ИКТ, методический центр «Раменский дом учителя», г. Раменское, Московская область, Россия; e-mail: namor1968@yandex.ru

1. Введение

Алгоритм работы любой программы на любом языке программирования опирается всего на пять базовых конструкций:

- 1) способ отделения команд друг от друга и способ объединения команд в модули;
- 2) ввод/вывод данных в программу/из программы;

- 3) присваивание переменным значений или через присвоение связывание переменных арифметической зависимостью;
- 4) конструкции ветвления;
- 5) циклические конструкции.

Если первые три конструкции осваиваются учащимися достаточно легко, то про последние две этого сказать нельзя. Но, как правило, именно последние две

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_06-2025/

R. B. Voronov

Udelninskaya Gymnasium, Udelnaya, Ramenskoye Urban District, Moscow Region, Russia;
Methodology Center «Ramensky Teacher's House», Ramenskoye, Moscow Region, Russia

PSEUDOGRAPHICS USING PYTHON AND PASCAL AS A VISUAL TOOL FOR TEACHING PROGRAMMING

Abstract

The article provides an example of presenting complex material to students, such as creating programs with nested loops and branching within them, in the Pascal or Python programming languages. Compared to other educational materials, pseudographics, on the one hand, provides greater clarity on how programs with nested loops interspersed with branching work, and on the other, instills in students a creative approach to solving complex programming problems. The article also clearly demonstrates through pseudographics the interdisciplinary connections between informatics, mathematics, and technology. Knowledge of algebraic equations for constructing various lines on a plane allows one to create these "lines" in pseudographics, ranging from straight lines and circles, parabolas and hyperbolas, to ovals and sinusoids of various types and sizes. Constructed using pseudographics, these beautiful figures can be displayed not only on the screen but also printed, for example, to create appliques in technology lessons.

The article will be useful for informatics teachers, students at pedagogical universities, and anyone interested in programming in Pascal and Python.

Keywords: teaching programming, Python, Pascal, pseudographics, nested loops.

Information about the author

Roman B. Voronov, an informatics teacher, Udelninskaya Gymnasium, Udelnaya, Ramenskoye Urban District, Moscow Region, Russia; a methodologist, an ICT teacher, Methodology Center "Ramensky Teacher's House", Ramenskoye, Moscow Region, Russia; e-mail: namor1968@yandex.ru

базовые конструкции — ветвление и циклы — в специфических сочетаниях определяют логику работы любой программы.

В псевдографике, которую мы определим как создание узоров или примитивных рисунков из вывода в консоли редактора различных символов, нет сложных математических, технических или других расчетов. Это позволяет обучающимся сосредоточить внимание на логике взаимодействия конструкций циклов и ветвления между собой. То есть, если ученик научится правильно сочетать в своей программе конструкции ветвления и цикла, а также составлять сложные условия для этих конструкций, то, по существу, он научится основам программирования.

Л. Л. Босова в учебном пособии [2] и в статье [1] указывает на необходимость изменений в преподавании информатики в школе. Она с соавторами предлагает увлечь обучающихся изучением информатики на примерах красивых и наглядных решений интересных задач [3]. Этому способствуют полезные решения разных повседневных, занимательных [3] или исследовательских [5], а также межпредметных задач [10].

В данной статье мы покажем:

- как научить обучающихся в наглядном виде сочетать в программах конструкции цикла как между собой (вложенные циклы), так и с конструкциями ветвления [4];
- как научить составлять сложные условия в конструкциях ветвления [7];
- как стимулировать интерес обучающихся к изучению языков программирования [5].

2. Базовый программный конструктив псевдографики

Рассмотрим простую задачу:

Вывести несколько раз (например, 11) на экран символ нуля, т. е. «0», в строку и затем перевести курсор на новую строку (листинг 1).

Для любой псевдографики нам нужно чистое полотно. Лучше всего сначала нарисовать его из ноликов, что символично для начала начал. Сделаем наше полотно прямоугольным и размером, например, 5 на 11. Для этого мы, конечно, не будем повторно прописывать команды пять раз, а поместим циклическую конструкцию с переводом курсора на новую строку (`writeln()` в Pascal или `print()` в Python) в другой, внешний цикл с повтором пять раз (листинг 2). Программисты говорят в таких случаях: «завернем» все во внешний цикл [6].

Мы получили определенного размера лист, или холст, на котором в дальнейшем будем рисовать или «крестиком вышивать» узор.

3. Математические основы псевдографики

Чтобы создавать разные узоры, нужно понять, что каждый символ нолик при выводе на холст имеет свои индивидуальные индексы-координаты *i* и *j*. То есть мы в любом месте нашего холста, а точнее, там, где захотим, можем вывести любой другой символ, например, можем

Pascal	Python
<pre>program uzor; var i,j: integer; begin for j:=0 to 10 do write(' 0'); writeln(); end.</pre>	<pre>for j in range(11): print(' 0', end='') print()</pre>
0 0 0 0 0 0 0 0 0 0 0 0	

Листинг 1 и результат работы программы

Pascal	Python
<pre>program uzor; var i,j: integer; begin for i:=0 to 4 do begin for j:=0 to 10 do write(' 0'); writeln(); end; end.</pre>	<pre>for i in range(5): for j in range(11): print(' 0', end='') print()</pre>
0 0	

Листинг 2 и результат работы программы

Pascal	Python
<pre> program uzor; var i,j: integer; begin for i:=0 to 4 do begin for j:=0 to 10 do if (i=1) and (j=4) then write(' +') else write(' 0'); writeln(); end; end. </pre>	<pre> for i in range(5): for j in range(11): if i=1 and j=4: print(' +', end='') else: print(' 0', end='') print() </pre>
<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 + 0 </pre>	

Листинг 3 и результат работы программы

вывести символ «+» во второй строке пятого столбца. Помним, что нумерация в математике и информатике начинается всегда с нуля. Тогда вторая строка имеет индекс строки $i = 1$ и индекс столбца $j = 4$. В наш вывод ноликов мы можем добавить условие: если координаты нолика равны $i = 1$ и $j = 4$, то вместо нолика выводится «+», а иначе будет выводиться «0» (листинг 3).

Теперь рассмотрим графики алгебраических уравнений на координатной плоскости. В алгебре ось абсцисс OX направлена вправо, а ось ординат OY — вверх [7]. В псевдографике, условно говоря, ось абсцисс тоже направлена вправо, и отвечает за нее (здесь уместно задать вопрос ученикам: какая переменная — i или j ?). Правильно, j ! Чем больше j , тем шире получается у нас холст. Вторым вопросом: а куда у нас направлена тогда ось ординат в псевдографике? Правильный ответ: вниз. Чем больше i , тем длиннее наш холст по вертикали. И тут возникает вопрос: как перевести графики алгебраических уравнений, связывающие координаты x и y , в псевдографику? Например, возьмем самое простое уравнение линейной функции: $y = x$ (рис. 1).

Мы видим, что все графики уравнений $y = f(x)$ в псевдографике будут превращаться в графики $i = f(j)$ полностью симметрично относительно горизонтальной оси координат — оси абсцисс OX .

Итак, построим диагональ квадрата размера 11 на 11 символов (листинг 4).

А как добавить вторую диагональ? В математике эта вторая диагональ проходила бы через точки (10; 0) и (0; 10). Отсюда ее уравнение: $y = 10 - x$. Тогда в псевдографике получаем аналогичное уравнение: $i = 10 - j$. Остается вопрос: а каким образом мы можем вывести сразу обе диагонали? При понимании процесса вывода символов в псевдографике он переходит в вопрос: какой логической операцией нам следует воспользоваться, чтобы вывелись обе диагонали? Приходим к выводу: на выводе, если в условии координаты символа удовлетворяют хотя бы одному из уравнений, выводим «+», иначе «0». То есть объединять эти уравнения в условии можно только логической операцией ИЛИ (дизъюнкцией).

Чтобы полотно приобрело форму, близкую к флагу, добавим к каждому выводимому символу сзади про-

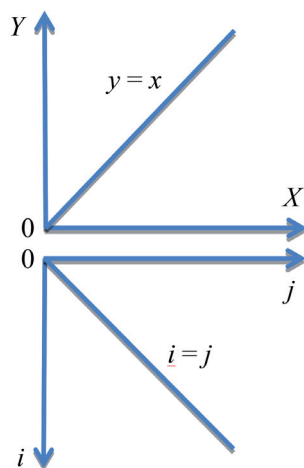


Рис. 1. Уравнение линейной функции

Pascal	Python
<pre> program uzor; var i,j: integer; begin for i:=0 to 10 do begin for j:=0 to 10 do if i=j then write(' + ') else write(' 0 '); writeln(); end; end; end. </pre>	<pre> for i in range(11): for j in range(11): if i == j: print(' + ', end='') else: print(' 0 ', end='') print() </pre>
<pre> + 0 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + </pre>	

Листинг 4 и результат работы программы

бел. Например, вместо ' 0' будем выводить ' 0 ' (листинг 5).

Мы получили андреевский или шотландский флаг, кому как нравится больше.

Теперь попробуйте аналогично построить другие линии в нашем узоре. Какие уравнения сюда нужно добавить, чтобы получить британский флаг? Сделайте самостоятельно.

Pascal	Python
<pre> program uzor; var i,j: integer; begin for i:=0 to 10 do begin for j:=0 to 10 do if (i=j) or (i=10-j) then write(' + ') else write(' 0 '); writeln(); end; end; end. </pre>	<pre> for i in range(11): for j in range(11): if i==j or i==10-j: print(' + ', end='') else: print(' 0 ', end='') print() </pre>
<pre> + 0 0 0 0 0 0 0 0 0 0 + 0 + 0 0 0 0 0 0 0 0 + 0 0 0 + 0 0 0 0 0 0 + 0 0 0 0 0 + 0 0 0 0 + 0 0 0 0 0 0 0 + 0 0 + 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 + 0 0 0 + 0 0 0 0 0 + 0 0 0 0 0 + 0 0 0 0 + 0 0 0 0 0 0 0 + 0 0 + 0 0 0 0 0 0 0 0 0 0 + </pre>	

Листинг 5 и результат работы программы

Pascal	Python
<pre> program uzor; var i,j: integer; begin for i:=0 to 10 do begin for j:=0 to 10 do if (i-j) mod 3=0 then write(' + ') else write(' 0 '); writeln(); end; end; end. </pre>	<pre> for i in range(11): for j in range(11): if (i-j)%3 ==0: print(' + ', end='') else: print(' 0 ', end='') print() </pre>
<pre> + 0 0 + </pre>	

Листинг 6 и результат работы программы

Если к этим уравнениям подключить такие операнды языков, как остаток и целая часть от деления, то можно целую россыпь уравнений для диагональных полосок

типа: $i=j-6$, $i=j-3$, $i=j$, $i=j+3$, $i=j+6$ и т. д. заменить в псевдографике одним изящным: $(i-j) \bmod 3=0$ в Pascal [6] или $(i-j)\%3==0$ в Python [2] (листинг 6).

Pascal	Python
<pre> program uzor; var i,j: integer; begin for i:=0 to 10 do begin for j:=0 to 10 do if (i>5-j) and (i<15-j) and (i<5+j) and (i>j-5) then write(' + ') else write(' 0 '); writeln(); end; end; end. </pre>	<pre> for i in range(11): for j in range(11): if i>5-j and i<15-j and i<5+j and i>j-5: print(' + ', end='') else: print(' 0 ', end='') print() </pre>
<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 + + + 0 0 0 0 0 0 0 + + + + + 0 0 0 0 0 + + + + + + + 0 0 0 + + + + + + + + 0 0 0 + + + + + + + 0 0 0 0 0 + + + + + 0 0 0 0 0 0 0 + + + 0 0 0 0 0 0 0 0 0 + 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 </pre>	

Листинг 7 и результат работы программы

Листинг 8 и результат работы программы

Рис. 2. Образец вывода кубка в домашнем задании

Задания для самостоятельного выполнения.

Попробуйте сами вывести:

- флаг из горизонтальных полос;
- потом флаг в клеточку;
- затем «шахматку», где «0» чередуется с другим символом в шахматном порядке.

Задание повышенного уровня сложности.

Вывести греческий флаг и шахматную доску размером 16×16 , где каждая клетка имеет размер два символа в ширину и в длину, т. е. представляет собой квадратик из четырех символов.

Все условия в программах, кроме уравнений диагоналей на флаге, должны состоять из одного уравнения!

Если перейти от линий к разным фигурам, то выясняется, что здесь нам тоже поможет математика. Только не уравнениями, а неравенствами. Например, если нам понадобится изобразить бразильский флаг, т. е. ромб в центре, то проще всего это сделать с помощью системы неравенств: $i > 5 - j, i < 15 - j, i < 5 + j, i > j - 5$, которые надо связать конъюнкцией, т. е. логическим И (листинг 7).

С помощью неравенств мы можем вывести не только многоугольники, но и любые округлые фигуры, описываемые уравнениями круга, овала, параболами, гиперболами и т. д.

Для наглядного примера, чтобы кривые шли более плавно, надо взять холст побольше — 21×21 . Изобразим круг с центром в точке (10; 10) с помощью классического уравнения: $(i - 10)^2 + (j - 10)^2 \leq 10^2$ (листинг 8).

Домашнее задание.

Построить кубок за первое место (рис. 2).

4. Поэтапное построение программы рисования новогодней елочки

Написание серьезной программы схоже с написанием если не романа, то как минимум повести — в основе лежит идея. Наша **идея** заключается в том, чтобы нарисовать с помощью псевдографики новогоднюю елочку.

Под идею подбирается оптимальный инструментарий. Под оптимальным понимается наиболее простой, наиболее быстрый и наиболее понятный способ вывода элементов рисунка. В нашем случае базовым элементом елочки является треугольник. Конечно, мы можем его построить «в лоб», опираясь на рассмотренную выше теорию алгебраических уравнений или неравенств. Но если знать свойства вложенного цикла, то можно построить в псевдографике любой треугольник безо всяких уравнений и даже без условий.

4.1. Этап 1. Построение равнобедренного треугольника**Шаг 1. Вывод треугольника с помощью вложенного цикла.**

Принцип простой: в первой строке выводим один символ, во второй — два символа, в третьей — три, и т. д. То есть внутренний цикл теперь будет каждый раз выводить не одинаковое количество символов, а по возрастающей — в прямой зависимости от номера строки (листинг 9).

Шаг 2. Вывод левой части равнобедренного треугольника.

Мы получили правую часть равнобедренного треугольника, а нам сначала нужна левая. Нарисовать ее можно с помощью убывающих с каждой строкой отступов перед выводом ноликов. Чтобы было понятнее, временно обозначим отступы символом подчеркивания «_» (листинг 10).

Шаг 3. Равнобедренный треугольник.

Можно было бы заменить $7-i$ на $6-i$, поскольку в последней строке отступ нам особо не нужен. Но нас устраивает и так. А теперь догадайтесь, во сколько раз надо увеличить количество ноликов в каждой строке, чтобы получить юбочку для елочки — равнобедренный треугольник? Конечно, в два раза. И обратите внимание на одну тонкость: первый нолик в первой строке не удвоится, т. е. конструкция `for j:=0 to 0 do`, как

Pascal	Python
<pre> program ubka; var i,j: integer; begin for i:=0 to 6 do begin for j:=0 to i do write(' 0'); writeln(); end; end. </pre>	<pre> for i in range(7): for j in range(i+1): print(' 0', end='') print() </pre>
<pre> 0 </pre>	

Листинг 9 и результат работы программы

Pascal	Python
<pre> program ubka; var i,j: integer; begin for i:=0 to 7 do begin for j:=0 to 7-i do write(' _'); for j:=0 to i do write(' 0'); writeln(); end; end. </pre>	<pre> for i in range(7): for j in range(7-i): print(' _', end='') for j in range(i+1): print(' 0', end='') print() </pre>
<pre> _ _ _ _ _ _ _ 0 _ _ _ _ _ _ 0 0 _ _ _ _ _ 0 0 0 _ _ _ _ 0 0 0 0 _ _ _ 0 0 0 0 0 _ _ 0 0 0 0 0 0 _ 0 0 0 0 0 0 0 _ 0 0 0 0 0 0 0 </pre>	

Листинг 10 и результат работы программы

и `for j:=0 to 0*2 do`, выведет в первой строке всего один символ. А далее, так как индекс первого символа в строочном выводе начинается с 0, а не с 1 и заканчивается четным числом, количество символов в каждой строке будет всегда нечетным. Это нам и нужно, для того чтобы наш равнобедренный треугольник имел центральную ось и лучше смотрелся. Заменяем заодно символ «`_`» на пробел (листинг 11).

4.2. Этап 2. Настройка елочки

Итак, главного мы добились.

Классическая елочка состоит из трех «юбочек». Все, что нам остается, это «завернуть» весь наш код в еще один внешний цикл (листинг 12).

Вроде бы похоже на елочку. Но в новогодней песенке поется, что она стройная была. Чего мы пока что про нашу елочку сказать не можем. Чтобы она стала стройнее, надо сделать верхние «юбочки» немного меньше нижних. Размер юбочки, как не трудно догадаться, зависит от количества строк в ней. Поэтому уменьшим верхнюю юбочку до пяти строк и также подрежем до пяти строк вторую и третью. И тут переменная внешнего цикла k нам в помощь (листинг 13).

Ну вот, теперь елочку можно забирать домой!

Домашнее задание.

Построить кубок за первое место без использования в программе ветвлений (рис. 3).

Pascal	Python
<pre> program ubka; var i,j: integer; begin for i:=0 to 7 do begin for j:=0 to 7-i do write(' '); for j:=0 to 2*i do write(' 0'); writeln(); end; end. </pre>	<pre> for i in range(7): for j in range(7-i): print(' ', end='') for j in range(2*i+1): print(' 0', end='') print() </pre>
<pre> 0 </pre>	

Листинг 11 и результат работы программы

Pascal	Python
<pre> program elka; var i,j,k: integer; begin for k:=0 to 2 do for i:=0 to 7 do begin for j:=0 to 7-i do write(' '); for j:=0 to 2*i do write(' 0'); writeln(); end; end; end. </pre>	<pre> for k in range(3): for i in range(7): for j in range(7-i): print(' ', end='') for j in range(2*i+1): print(' 0', end='') print() </pre>
<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 </pre>	

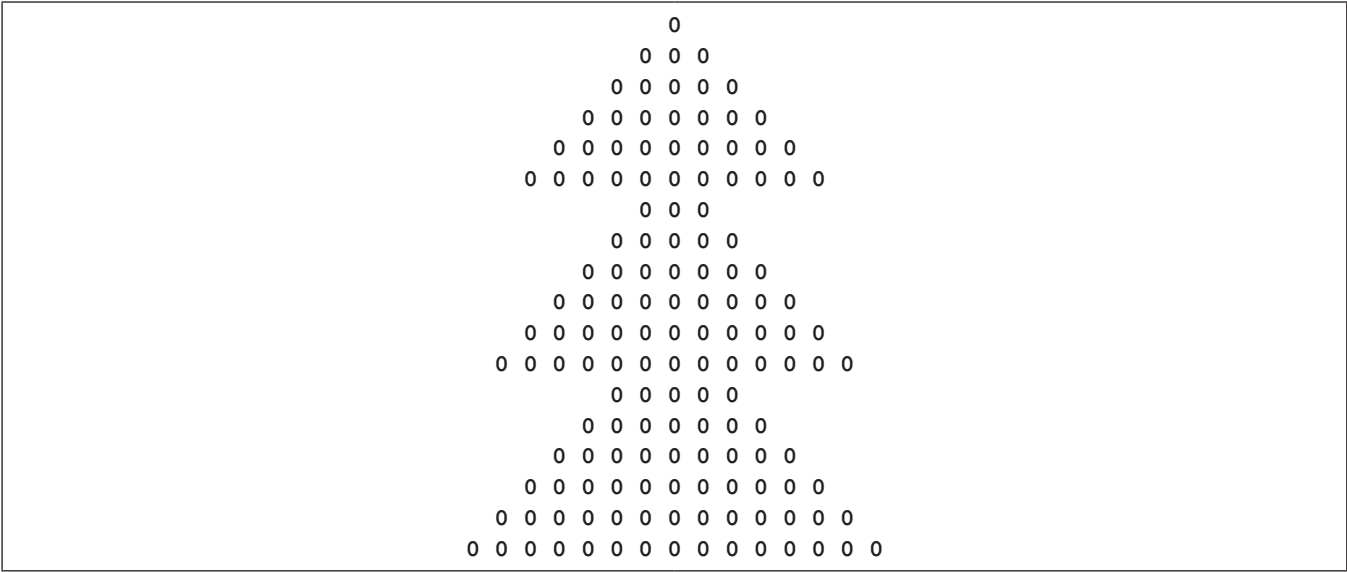
Листинг 12 и результат работы программы

На данном примере очень удобно объяснять в старших классах смысл процедуры в Pascal или функции в Python, которые будут выводить юбочку для елочки с парой наглядных параметров. Один параметр, обязательный, будет отвечать за количество строк в юбочке,

а второй, необязательный, по умолчанию равный 0, — за количество символов в первой строке. А потом в цикле от 0 до 2 включительно в Pascal и от 0 до 3, не включая 3, в Python запускаем эту функцию с возрастающим на единицу параметром количества символов в первой строке.

Pascal	Python
<pre> program elka; var i,j,k: integer; begin for k:=0 to 2 do for i:=k to k+5 do begin for j:=0 to 7-i do write (' '); for j:=0 to 2*i do write (' 0'); writeln(); end; end; end. </pre>	<pre> for k in range(3): for i in range(k,k+5): for j in range(7-i): print(' ', end='') for j in range(2*i+1): print(' 0', end='') print() </pre>

Листинг 13



Результат работы программы листинга 13

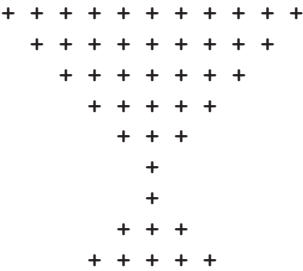


Рис. 3. Образец вывода кубка в домашнем задании без использования ветвлений

При этом количество символов будет в самой функции увеличиваться на 2.

оставалось нечетным, будем выводить ' + ', примыкающий к остальным (листинг 14).

4.3. Этап 3. Украшение елочки

Шаг 1. «Колючая» елочка.
Чтобы елочка лучше смотрелась, сделаем ее «колючей» и более плотной, заменив символы ' 0 ' на ' ++ '. Но в начале каждой строки, чтобы количество плюси

Шаг 2. Гирлянда.
С чего начинают украшать елочку? Правильно, с гирлянды из маленьких лампочек.
Маленькой лампочкой нам послужит вот такой кружочек «о» (из символов Word). В стандартной таблице

Pascal	Python
<pre>program elka; var i,j,k: integer; begin for k:=0 to 2 do for i:=k to k+5 do begin for j:=0 to 7-i do write(' '); for j:=0 to 2*i do if j=0 then write(' +') else write('++'); writeln(); end; end; end.</pre>	<pre>for k in range(3): for i in range(k,k+5): for j in range(7-i): print(' ', end='') for j in range(2*i+1): if j==0: print(' +',end='') else: print('++',end='') print()</pre>

Листинг 14

[illegible]

Результат работы программы листинга 14

Unicode это символ номер 9675. Выводим его специальной функцией chr(9675). Как вывести? Очень просто: каждый, например, четвертый вывод '++' заменим на 'o+'. Обратите внимание: не четвертый от края, а кратный четырем! И сдвинем на один вывод влево —

так будет выглядеть аккуратнее. Отступы для блоков в Python придется поменять здесь с четырех пробелов на два (листинг 15).

Теперь осталось включить нашу елочку. Как работали лампочки в старых гирляндах? Через одну: сначала

Pascal	Python
<pre> program elka; var i,j,k: integer; begin for k:=0 to 2 do for i:=k to k+5 do begin for j:=0 to 7-i do write(' '); for j:=0 to 2*i do if j=0 then write(' +') else if (j+1) mod 4=0 then write(Chr(9675),'+') else write('+++'); writeln(); end; end. </pre>	<pre> for k in range(3): for i in range(k,k+5+1): for j in range(7-i): print(' ', end='') for j in range(2*i+1): if j==0: print(' +',end='') elif (j+1)%4==0: print(chr(9675)+'+',end='') else: print('++',end='') print() </pre>
<pre> + +++++ +++++○+++ +++++○+++++++ +++++○+++++++○+++ +++++ +++++○+++ +++++○+++++++ +++++○+++++++○+++ +++++○+++++++○+++++++ +++++○+++ +++++○+++++++ +++++○+++++++○+++ +++++○+++++++○+++++++ +++++○+++++++○+++++++○+++ </pre>	

Листинг 15 и результат работы программы

включались одни, потом другие. В нашем случае получаем, что если в нечетных строках лампочки должны гореть, то в четных — нет. Еще помним, какая у нас переменная за строки отвечает? Unicode потухшей лампочки, которую обозначим символом «●», — 9679 (листинг 16).

Шаг 3. Украшения и игрушки на елочку.

Остался последний шаг — «повесить» на елочку «игрушки». Где их взять? В кодировочной таблице Unicode. Но в ней больше 10 000 символов! А где там прячутся самые красивые игрушки? Выведем же некоторые из них на свет вместе с их кодами [9] (листинг 17).

Как надеть на самый верх елки какое-то украшение, вам должно быть уже понятно. А давайте развесим на ней снежинки ❄ с кодами 10052–10054 в хаотичном порядке! В Pascal воспользуемся встроенной функцией random(), а вот в Python придется подключать модуль random.

Сначала выберем какую-нибудь одну снежинку из трех, например, с кодом `s:=10052+random(3)`. Функция random в Pascal сгенерирует либо 0, либо 1, либо 2,

но не 3! В Python она под заданным нами именем `rnd` работает предельно просто.

А потом — рулетка: если вдруг текущий индекс столбца случайно совпадет со сгенерированным числом для каждого нового индекса, то снежинка появится с вероятностью около 1/8. И может легко получиться, что в одной строке снежинок несколько, а в следующей их нет совсем. Но снежинка больше плюсика. В итоге некоторые строки некрасиво расползаются (листинг 18).

Чтобы этого избежать, нужно разместить строго по одной снежинке в строку. Для этого необходимо сгенерировать номер индекса, например `n`, где и будет стоять снежинка перед выводом строки, благо длина строки нам будет известна. И когда индекс `j` сравняется с `n`, выведется снежинка. Ровно один раз, поскольку для каждой строки этот номер `n` генерируется только один раз в пределах значений индекса `j` для этой строки. И еще, снежинки должны перекрывать лампочки, т. е. условие для них должно стоять выше условия вывода лампочек (листинг 19).

Pascal	Python
<pre> program elka; var i,j,k,u: integer; begin for k:=0 to 2 do for i:=k to k+5 do begin if i mod 2=0 then u:=9675 else u:=9679; for j:=0 to 7-i do write(' '); for j:=0 to 2*i do if j=0 then write (' +') else if (j+1) mod 4=0 then write(Chr(u), '+') else write ('++'); writeln(); end; end; end. </pre>	<pre> for k in range(3): for i in range(k,k+5+1): if i%2==0: u=9675 else: u=9679 for j in range(7-i): print(' ', end='') for j in range(2*i+1): if j==0: print(' +', end='') elif (j+1)%4==0: print(chr(u)+'+',end='') else: print('++', end='') print() </pre>
<pre> + +++++ +++++o+++ +++++●+++++ +++++o+++++o+++ +++++●+++++●+++++ +++++ +++++o+++ +++++●+++++ +++++o+++++o+++ +++++●+++++●+++++ +++++o+++++o+++++o+++ +++++o+++ +++++●+++++ +++++o+++++o+++ +++++●+++++●+++++ +++++o+++++o+++++o+++ +++++●+++++●+++++●+++++ </pre>	

Листинг 16 и результат работы программы

Pascal	Python
<pre>for i:= 10022 to 10054 do write(i, ' ', Chr(i), ',')</pre>	<pre>for i in range(10022, 10054+1): print(i, chr(i), ', ', end='')</pre>
<pre>10022 ✦ ,10023 ✧ ,10024 ✨ ,10025 ☆ ,10026 ☼ ,10027 ☆ ,10028 ☆ ,10029 ☆ ,10030 ☆ , 10031 ☆ ,10032 ☆ ,10033 ✱ ,10034 ✱ ,10035 ☼ ,10036 ☼ ,10037 ✱ ,10038 ✱ ,10039 ✱ , 10040 ✱ ,10041 ✱ ,10042 ✱ ,10043 ✱ ,10044 ✱ ,10045 ✱ ,10046 ✱ ,10047 ✱ ,10048 ✱ , 10049 ✱ ,10050 ✱ ,10051 ✱ ,10052 ✱ ,10053 ✱ ,10054 ✱</pre>	

Листинг 17 и результат работы программы

Pascal	Python
<pre>program elka; var i,j,k,u,s: integer; begin for i:=0 to 16 do write(' '); writeln(Chr(10031)); for k:=0 to 2 do for i:=k to k+5 do begin if i mod 2=0 then u:=9675 else u:=9679; for j:=0 to 7-i do write(' '); for j:=0 to 2*i do begin s:=10053; if j=0 then write(' +') else if (j+1) mod 4=0 then write(Chr(u), '+') else if j=random(8) then write(Chr(s), '+') else write('+++'); end; writeln(); end; end; end.</pre>	<pre>from random import randint as rnd for i in range(15): print(' ', end='') print(chr(10031)) for k in range(3): for i in range(k,k+5+1): if i%2==0: u=9675 else: u=9679 for j in range(7-i): print(' ', end='') for j in range(2*i+1): s=rnd(10052,10054) if j==0: print(' +', end='') elif (j+1)%4==0: print(chr(u)+'+', end='') elif j==rnd(0,7): print(chr(s)+'+', end='') else: print('+++', end='') print()</pre>
<pre> ☆ + +++++ +++✱+O+++ ++++++●+++++++ ++++++O+++ ✱ ++++O+++ ++++++●+++++++ ✱ +●+++++++ ++++++ +++ ✱ +O+++ ++++++●+++++++ ++++++O+++++++O+++ ++++++●+++++++●+++++++ ++++++O+++++++O+++++++O+++ ++++++O+++ ++++++●+++++++ ++++++O+++++++O+++ ++++++●++++ ✱ ++++●+++++++ ++++++O+ ✱ ++++++O+++++++O+++ ++++++●++++ ✱ ++++●+++++++●+++++++ </pre>	<pre> ☆ + +++++ ++++++O+++ ++++++●+++++++ ++++++O+ ✱ ++++++O+++ ++++++●++++ ✱ ++++●+++++++ ++++++ +++ ✱ + +++ ✱ +O+++ ++++++●+++++++ +++ ✱ +O++++ ✱ +O+++ + ✱ ++++●++++ ✱ ++++●+++++++ ++++++O++++ ✱ +O+++++++O+++ + ✱ ++++O+++ ++++++●++++ ✱ + ++++++O+++++++O+++ ++++++●++++ ✱ ++++●+++++++ ++++++O+++++++O+++++++O+++ ++++++●++++ ✱ ++++●+++++++●+++++++ </pre>

Листинг 18 и результат работы программы

Pascal	Python
<pre> program elka; var i,j,k,u,s,n: integer; begin for i:=0 to 16 do write(' '); writeln(Chr(10031)); for k:=0 to 2 do for i:=k to k+5 do begin if i mod 2=0 then u:=9675 else u:=9679; s:=10052+random(3); n:=random(2*i+1); for j:=0 to 7-i do write(' '); for j:=0 to 2*i do begin if j=0 then write(' +') else if j=n then write (Chr(s),'+') else if (j+1) mod 4=0 then write (Chr(u),'+') else write('++'); end; writeln(); end; end; end. </pre>	<pre> from random import randint as rnd for i in range(15): print(' ', end='') print(chr(10031)) for k in range(3): for i in range(k,k+5+1): s=rnd(10052,10054) n=rnd(0,2*i) if i%2==0: u=9675 else: u=9679 for j in range(7-i): print(' ', end='') for j in range(2*i+1): if j==0: print(' +', end='') elif j==n: print(chr(s)+'+', end='') elif (j+1)%4==0: print(chr(u)+'+', end='') else: print('++', end='') print() </pre>
<pre> ☆ + +++++ ++++++O+++ ++++++●+++ ❄️ +++ ++++++O++++++ ❄️ +++ +++ ❄️ +●+++++++●+++++++ +++++ ❄️ + ++++++O+++ ++++++●+++ ❄️ +++ ++++++O++++++ ❄️ +++ ++++++●+++ ❄️ +++●+++++++ ++++++O++++++ ❄️ +++++++O+++ ++++++O+++ + ❄️ +++●+++++++ ++++++O+++ ❄️ ++++O+++ ++++++●+++++++ ❄️ +++++++ + ❄️ ++++O+++++++O+++++++O+++ ++++++●+++ ❄️ +++●+++++++●+++++++ </pre>	<pre> ☆ + +++ ❄️ + ++++++O+++ ++++++●+++ ❄️ ++++ + ❄️ ++++O+++++++O+++ ++++++ ❄️ ++++++++●+++++++ +++++ ❄️ + +++ ❄️ +O+++ +++++++ ❄️ + ++++++O+++ ❄️ ++++O+++ ++++++●+++++++ ❄️ +++++++ ++++++O+++++++O+ ❄️ ++++++O+++ ++++++O+++ +++ ❄️ +●+++++++ +++ ❄️ +O+++++++O+++ ++++++●+++ ❄️ +++●+++++++ ++++++ ❄️ ++++++++O+++++++O+++ ++++++●+++++++ ❄️ ++++++++●+++++++ </pre>

Листинг 19 и результат работы программы

Домашнее задание.

Украшения, которые мы рассмотрели, достаточно сложные. Поэтому теперь, если вы разобрались с тонкостями случайного вывода, для вас не составит большого труда добавить на елочку разнообразные игрушки.

5. Заключение

В данной статье мы разобрали сначала теорию — основной конструктив, на котором базируется псевдографика, далее — алгебраические уравнения линий на

плоскости, а затем — практику: поэтапное и пошаговое моделирование изделия.

Что касается практики, написания реальной программы, то это очень похоже на изготовление материального изделия, например, коленчатого вала двигателя внутреннего сгорания. Сначала мы отлили болванку, заготовку под деталь — это были контуры елочки из ноликов, только на основе вложенных циклов без ветвления. Затем с помощью вывода двойного плюсики '++' мы «отшлифовали» елочку, сделав ее более плотной и пушистой, а потом применили ветвление — как

режущий инструмент для детали, которым выпиливают отверстия, выемки, резьбы. И в конце с помощью более тонкого сочетания конструкций цикла и ветвления, разобравшись с индексами символов в строках, добавили аккуратные штрихи, чтобы от разницы в ширине символов между плюсиками и снежинками наша елочка не разъехалась построчно. Это был аналог финальной полировки детали.

То есть можно сказать, что процесс создания программы — это тот же производственный процесс. И его невозможно освоить без длительной учебной практики, без нее продукт не достигнет товарного качества.

А умение сочетать вложенные циклы с ветвлением, по существу, и определяет уровень владения языком программирования. И псевдографика, как наглядное представление работы программы в процессе ее выполнения, позволяет как видеть ошибки в работе алгоритма программы, так и демонстрировать свое мастерство в построении рационального и оригинального алгоритма программы (рис. 4). Поэтому, только научившись легко и свободно плести узоры и выводить красивые картинки с помощью псевдографики, можно считать себя реальным программистом.

С Новым годом!

Список источников

1. Босова Л. Л. О новых подходах к изучению школьной информатики в условиях цифровой трансформации общества // Информатика в школе. 2022. № 4 (177). С. 5–14. EDN: DKRLZV. DOI: 10.32517/2221-1993-2022-21-4-5-14.

2. Босова Л. Л., Аквилянов Н. А., Кочергин И. О., Штепа Ю. Л., Бурцева Т. А. Информатика. 8–9 классы. Начала программирования на языке Python. Дополнительные главы к учебникам. М.: БИНОМ. Лаборатория знаний, 2020. 96 с. EDN: LPMXCB.

3. Босова Л. Л., Филиппов В. И., Босова А. Ю. Программирование графики и анимации в курсе информатики основной школы // Информатика в школе. 2022. № 5 (178). С. 5–15. EDN: LYUPWI. DOI: 10.32517/2221-1993-2022-21-5-5-15.

4. Гербут С. С. Методические аспекты обучения началам программирования в среде Pascal ABC.NET (Исполнитель чертёжник) // Информатика в школе. 2021. № 4 (167). С. 24–33. EDN: VRHAXB. DOI: 10.32517/2221-1993-2021-20-4-24-33.

5. Ивкина Е. С. Разработка и применение заданий межпредметного характера, как средство повышения мотивации учащихся на уроках информатики // Хроники объединенного фонда электронных ресурсов наука и образование. 2014. Т. 1. № 12 (67). С. 110. EDN: TGUHUR.

6. Касторнов А. Ф., Касторнова В. А., Козлов О. А. Программирование на языке Паскаль: учебное пособие. Н. Новгород: Нижегородский государственный педагогический университет, 2012. 152 с. EDN: RJBKAC.

7. Коберник Ю. В., Скубко В. В. Возможные пути преодоления трудностей учащихся в изучении графиков функций и уравнений // Инновационные научные исследования. 2020. № 12-3 (2). С. 98–108. EDN: NKWQZN.

8. Криволясова Е. В., Нефёдова В. Ю., Прилепина А. В. Методика обучения основам программирования на языке PYTHON // Информатика в школе. 2020. № 3 (156). С. 24–30. EDN: NCKNDR. DOI: 10.32517/2221-1993-2020-19-3-24-30.

9. Таблица символов Unicode. <http://www.hdsr.ru/?p=25>

10. Филиппов В. И. Достижение метапредметных результатов при изучении раздела «Алгоритмизация и программирование» // Информатика в школе. 2015. № 8 (111). С. 4–8. EDN: VADQCH.

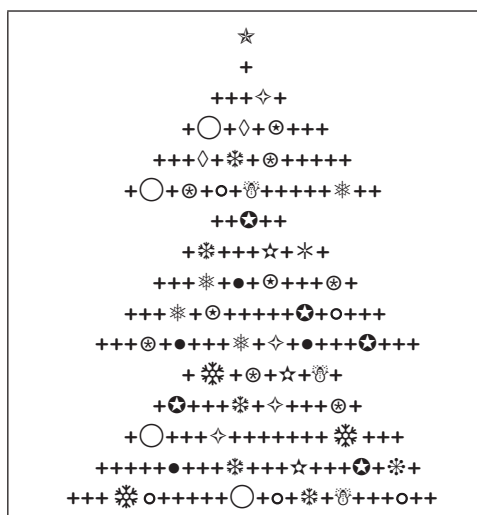


Рис. 4. С Новым годом!