ПРОБЛЕМЫ ПРЕПОДАВАНИЯ В ВЫСШЕЙ ШКОЛЕ

Физико-математические науки

СИНТЕЗ МЕТОДИЧЕСКИХ ВОЗМОЖНОСТЕЙ УЧЕБНЫХ ИСПОЛНИТЕЛЕЙ «ЧЕРЕПАХА» И «ЧЕРТЕЖНИК» В ЯЗЫКЕ PASCALABC

А.И. Бочкин, Д.Р. Кузьмичев
Витебск, ВГУ имени П.М. Машерова

Успешность языка PascalABC в обучении объясняется тем, что его разработчики, снимая проблемы в школьной информатике, соединили язык управления исполнителями и язык из разработки. Такой подход удалось осуществить с помощью открытого кода самих исполнителей в стиле, указанном еще А.Г. Кушниренко.

Целью работы является обогащение инструментария изучающего информатику путем комбинаторного расширения команд управления исполнителем, расширение набора типов данных на элементарном уровне**.**

Материал и методы. Материалом являются исполнители Чертежник и Черепаха, реализуюшее их программное обеспечение школьной информатики для изучения элементов алгоритмизации и программирования. Методом является синтез исполнителей на уровне их команд.

Результаты и их обсуждение. Исполнитель Чертежник является методическим инструментом и средой для освоения на элементарном и наглядном уровне таких понятий, как «процедура с параметрами», «глобальные и локальные переменные» (координаты начала рисунка и относительные смещения), разработка программы по технологии «сверху вниз».

Исполнитель Чертежник унаследовал от своего аналога (псевдографического Робота) отсутствие направления, что ограничивает его возможности функциями графопостроителя.

Реализованная на практике успешность исполнителя Черепаха в языке Лого объясняется его подобием не только черепахе, но и человеку – наличие направления для движения, «лица». Этим упрощается присвоение учащимся алгоритма на первом этапе обучения, подстановка себя на место исполнителя. С другой стороны, «незнание» координат своего положения затрудняет возврат Черепахи в исходную точку после фрагмента рисования. Поэтому стоит объединить этих исполнителей и получить мощного, наглядного и при том простого исполнителя.

Фактически реализованные разработчиками языка PASCALABC в модуле Turtle процедуры On_Vector и To_Point этот синтез и выполняют, однако, напрашивается полезное добавление – запоминать координаты исполнителя перед фрагментом рисования для простого возврата после построения фрагмента (Листинг 1).

Листинг 1. *Возврат исполнителя в начальную точку.*

```
procedure mem;  {запомнить положение}
begin
 xm:=turtleX;
 ym:=turtleY;
end;

procedure rem;  {вспомнить положение}
begin
 topoint(xm,ym); {перейти туда, где был}
end;
```

Стоит изменить названия поворотов на процедуры с более понятными именами (Листинг 2) или, методически правильнее, – предложить это сделать учащемуся.

Листинг 2. *Собственные процедуры поворотов для учебного исполнителя.*

```
Procedure napravo(dfi:integer);
Begin
 Turn(-dfi)
End;
```

Возможность нарисовать целое дерево через рекурсивную процедуру с единственной простой проверкой условия уже стоит того, чтобы только для этого дерева учащемуся освоить пару команд Черепахи. Это полностью наглядный, динамически, управляемый и медленно, наглядно рисующий алгоритм полностью исчерпывает тему «Рекурсия» и наполовину тему – «структуры данных».

```
procedure derevo(n:real); {рекурсивное дерево со стволом длины n}
const h=1.5;      {шаг укорочения ветки для остановки рекурсии}
begin
 if n<=1 then exit;  {дошли до коротких веток - откат}
 vpered(n-h);    {Основная часть: рисуем ветку или ствол}
 nalevo(45);     {чтобы рисовать дерево короче - поворот налево}
 derevo(n-h);    {рисовать налево рекурсивно меньшее дерево}
 napravo(90);     {после возврата поворот для рисования дерева направо}
 derevo(n-h);    {рисовать направо рекурсивно меньшее дерево}
 nalevo(45);     {вернуть исходное направление до рисования 2 веток}
 vpered(-(n-h));   {спуститься назад по стволу после рисования}
end;
```

Наличие Sin и Cos (скрытых в модуль Turtle) при расчете приращений координат двойственно. С одной стороны, это позволяет учащемуся, не знакомому с тригонометрией, не вникать в устройство Черепахи и не рассматривать ее код. С другой стороны, учащийся, знакомый с тригонометрией, может увидеть, что синус и косинус используются практически для рисования на ЭВМ. Кроме того, возможно, именно здесь он может, наконец, увидеть их геометрический, наглядный смысл.

Не стоит бояться раннего введения процедур. По существу команда OnVector – уже готовая процедура для применения, а как называть того, кто ею пользуется – не так важно: если ученик самоутверждается и считает себя программистом, не стоит ему мешать. Кроме того, нынешняя информатика инверсна – то, что появилось позже – изучается и применяется раньше (графика, мышь, сенсорный экран и так далее).

Заключение. Полагаем, что применение синтеза учебных алгоритмических исполнителей «Чертежника» и «Черепахи» представляет собой новый и эффективный методический инструмент в руках учителя. Расширение системы команд «Чертежника» тремя процедурами «vpered», «nalevo», «napravo» позволяет перенести все учебные программы, созданные и опубликованные ранее для «Черепахи», в среду языка PascalABC. Указанные процедуры предлагаются с одним параметром, а не с двумя, и опираются на идею о направлении исполнителя, поэтому они проще в изучении и понятнее для учащегося на практике, чем процедуры с двумя параметрами OnVector() или ToPoint().

СОПОСТАВЛЕНИЕ ПОНЯТИЙ «СОБЫТИЕ» И «УСЛОВИЕ» В ОБРАЗОВАТЕЛЬНЫХ КОНЦЕПЦИЯХ И ПРОФЕССИОНАЛЬНОМ ПРОГРАММИРОВАНИИ

А.И. Бочкин, Д.Р. Кузьмичев
Витебск, ВГУ имени П.М. Машерова

Как указывалось нами ранее [1], понятие «событие» является с точки зрения пользователя первичным, более близким к человеческой, а не компьютерной логике, чем понятие «условие» в языках программирования. В силу указанной нами инверсии средств при изучении информатики оно должно изучаться ранее или одновременно. Следующим