УДК 38:378.016

Г. А. Заборовский,
кандидат физико-математических наук,
доцент кафедры информатики и методики преподавания информатики БГПУ

ИЗУЧЕНИЕ ОБЪЕКТНО ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ В СРЕДЕ PASCAL ABC

Изучение различных подходов к программированию в среде Pascal ABC актуально в связи с использованием этого языка в школах республики и необходимостью обеспечить не только базовый, но и углубленный уровень обучения. Ранее нами рассмотрены примеры реализации структурно-модульного и событийно-ориентированного программирования в этой среде [1]. Объектно ориентированное программирование (ООП) -парадигма программирования, основными понятиями которой являются объект и класс. В отличие от процедурного программирования, где данные и подпрограммы их обработки (процедуры, функции) формально не связаны, в ООП данные и методы их обработки интегрированы (инкапсулированы) в одном объекте. Программа рассматривается как набор взаимодействующих объектов. ООП есть по сути императивное программирование, дополненное принципами инкапсуляции данных и методов в объект, наследования, полиморфизма.

Программированию на языке Pascal посвящено громадное количество работ, однако практически все они реализуют версии Turbo Pascal или Delphi. Даже в лучших, на наш взгляд, учебных пособиях рассмотрение ООП в среде Turbo Pascal весьма краткое и рассчитано на университетский уровень [2; 3], а в среде Delphi основное внимание уделяется использованию графических компонентов [3; 4]. Изучению программирования в среде Pascal ABC посвящено гораздо меньше работ. Все доступные нам пособия рассчитаны на начальный уровень и не затрагивают ООП [5-8].

Рассмотрим на примерах возможности реализации объектно ориентированного программирования в среде Pascal ABC, формирование основных понятий и принципов. Важнейшим понятием ООП является класс. Класс - это тип данных, описывающий устройство объектов. Класс создается программистом во время написания кода и служит шаблоном для объектов (экземпляров класса), которые создаются во время работы программы. В языке Pascal класс представляет собой объектный тип данных, похожий на тип *запись.* Запись - составной тип данных, содержащий набор элементов разных типов, каждый из которых имеет свое имя и называется полем записи.

Покажем формирование понятия *класс* на примере решения одной и той же задачи двумя способами, переходя от использования записи (процедурный подход) к проектированию класса (объектно ориентированный подход).

Пример 1. Составить программу, которая выводит данные об учащихся: фамилию fam, имя name, класс klas, а также возраст age.

Рассмотрим сначала процедурный вариант, использующий тип *запись.*

```
Program Stud_zapis; t
  ype
  Stud = record          {описание типа запись)
fam:   string;  name:   string; klas: 1..11;
age:   6. .18; end;
```

Код программы начинается с объявления типа переменных st1 и st2. Для обращения к элементам записи используется точечная нотация: указывается имя переменной и имя поля, разделенные точкой.

```
var st1,   st2   :   Stud;
begin        {присваивание и вывод значений)
stl.fam :=  'Иванов';  stl.name  := 'Ваня';
stl.klas   := 4;  stl.age   := 10;
writeln(stl.fam:8,  stl.name:8, stl.klas:4,
'класс',  stl.age:4, 'лет');
end.
```

Заметим, что присваивание и вывод значений в коде программы довольно громоздко. Для упрощения кода введем процедуры пользователя Create и Print. Мы преднамеренно используем для нашей процедуры имя Create, которое в объектном подходе принято для конструктора.

```
var st:   Stud;
procedure Create (fm: string; nm:string;
kl:l..11; ag:6..18) ;
begin
st.fam := fm; st.name := nm; st.klas := kl;
st.age  := ag;
end;
procedure Print
; begin
writeln(st.fam:8,  st.name:8,  st.klas:4,
'класс',  st.age:4,   'лет');
 end;
```

Теперь код присваивания и вывода значений существенно упростился:

```
begin
 Create('Влськин',   'Вася',  9,  15);
Print;
 end.
```

В объектно ориентированном варианте решения этой же задачи будем использовать понятие *класс.* Класс является составным типом, включающим данные (поля и свойства), методы (процедуры и функции), конструкторы, события. Поля класса описываются по тем же правилам, что и поля в записях. Описание методов идентично описанию обычных процедур и функций. Объявление метода является заголовком процедуры или функции, которая должна быть описана либо внутри, либо вне тела класса. При описании вне класса имя метода предваряется именем класса с точкой (точечная нотация).

Опишем класс **Stud,** задав в нем поля fam, name, klas, age, а также методы **Create** и **Print.** Метод **Print** является обычной процедурой вывода данных. Специальный метод **Create,** предназначенный для инициализации данных в ООП, называется конструктором. **Конструктор** представляет собой функцию, создающую объект в динамической памяти, инициализирующую его поля и возвращающую указатель на созданный объект. При описании конструктора вместо служебного слова **function** используется слово **constructor.** Для конструктора не указывается никакой тип возвращаемого значения. При вызове конструктора пишется имя класса, за которым следует точка-разделитель, имя конструктора и список параметров. При создании объекта поля не инициализируются автоматически, все поля следует инициализировать явно. В Pascal ABC, как и в Borland Delphi, для конструктора принято использовать имя **Create.**

```
Program
 Stud_obj; type
Stud = class                {объявление класса}
fam:  string; name:string; klas:l..ll;
age:6..18;                  {объявление полей}

Constructor Create (fm:   string; nm: string;
kl: 1. .11;  ag:6. .18) ;
begin{конструктор с параметрами)
fam := fm; name:= nm; klas := kl;
age  := ag;
end;

procedure Print ;
 begin {описание метода- процедуры)
writeln(fam:8,  name:8, klas:4,'  класс',
age: 4,   'лет');
end;
end;
```

Код фрагмента программы с присваиванием и выводом значений имеет предельно простой вид:

```
var st:  Stud;{объявление переменной
                    типа Stud)
begin          {создание экземпляров класса,
              присваивание и вывод значений)
st:= Stud.Create('Иванов',   'Ваня',  9,
15); st.Print;
st:= Stud.Create('Петров','Петя',  8,  14);
st.Print; end.
```

В Pascal ABC можно использовать массивы объектов, например, задать информацию о 20 учащихся, а затем вывести для первых трех:

```
var spis:  array[1..20]   of Stud; var i:
integer;
spistl]   := Stud.CreateМальков',   'Коля',
1, 6) ;
spis[2]   := Stud.Create('Ясная',   'Маша',
2, 7) ;
spis[3]   := Stud.Create('Краснова','Оля',
V,  12) ;
for i   := 1 to 3 do spis [i] .Print;
```

Рассмотрим теперь пример реализации важнейшего принципа ООП - наследования. **Наследованием** называют возможность порождать один класс от другого с сохранением всех свойств и методов **класса-предка.** Использование свойств и методов классов-предков позволяет упростить большие программы вследствие исключения повторяющегося кода.

Пример 2. Составить программу, которая выводит сообщение о состоянии компьютера или ноутбука, а также параметры: название name, оперативная память ram, время зарядки аккумулятора ноутбука tim.

Опишем сначала класс **Computer,** задав в нем поля name, ram, конструктор **Create,** а также методы-процедуры **Start** и **Stop,** которые выводят сообщения о названии компьютера, его состоянии и оперативной памяти.

```
Program Computer;
 type
  Computer=class            (объявление класса)
   name:  string; ram:integer;
   {объявление
                                      полей)
   constructor Create (nm:   string;  rm:
   1..16);
                {конструктор с параметрами)
begin
  name  := nm;  ram := rm;
end;
procedure Start;
begin          {описание метода-процедуры)
write(name,'работает,  память =', ram,
'Гбайт.'); end;
procedure Stop ;
begin          {описание метода-процедуры)
  writeln(name,   'выключается');
 end;
end;
```

Опишем теперь класс **Notebook,** наследующий и расширяющий возможности класса **Computer,** для чего после служебного слова **class** в скобках укажем имя класса-предка. В классе **Notebook** зададим дополнительно поле tim (время зарядки аккумулятора), переопределим конструктор, поскольку он должен инициализировать уже не две, а три переменные. Переопределим также метод **Stop,** поскольку для ноутбука он должен выводить еще и сообщение о зарядке аккумулятора.

```
Notebook=class(Computer)    {объявление класса -
                            наследника}
tim: integer;    {объявление нового поля}
constructor Create (nm:   string;  rm:   1..16;
tm:  20..150);
begin      {вызов предка, переопределение}
  inherited Create(nm,rm);  tim:=tm; end;
procedure Stop ;
begin              {переопределение метода)
  writeln(name,'выключается, заряжается',
 tim,   ' мин');
end; end;
```

Код программы и в этом случае имеет простой вид и сводится к объявлению переменных, присваиванию и выводу значений:

```
var compl,  comp2:  Computer;  {объявление
                               переменных}
var nbl,  nb2   :
Notebook; begin
compl   := Computer.Create('IBM',   4);
compl.Start;  compl.Stop;
  nbl   := Notebook.Create('Asus',  2,
  30);
 nbl.Start; nbl.Stop; comp2  :=
Notebook.Create('Sony',  3,  120);
comp2.Start;  comp2.Stop; end.
```

Результаты работы программы выглядят так:

IBM работает, память = 4 Гбайт	IBM выключается
Asus работает, память = 2 Гбайт	Asus выключается, заряжается 30 мин
Sony работает, память = 3 Гбайт	Sony выключается, заряжается 120 мин

Обратим особое внимание на третий результат. Несмотря на то, что переменная comp2 объявлена типом Computer, возможно использование конструктора для класса-наследника Notebook, что автоматически приводит к «превращению» компьютера Sony в ноутбук! Такое поведение объектов в ООП называют **полиморфизмом.** Это явление, при котором методу с одним и тем же именем может соответствовать разный программный код (полиморфный код) в зависимости от того, объект какого класса используется при вызове данного метода. Наряду с наследованием это важнейший принцип ООП.

Еще одно проявление полиморфизма -**перегрузка метода.** В этом случае имя метода неизменно, а количество или тип параметров отличается.

Пример 3. Составить программу, которая выводит периметр фигур: квадрата, прямоугольника, треугольника.

Опишем три процедуры с одинаковым именем **Perimetr,** но разными параметрами:

```
Program Perimetr3; procedure
 Perimetr (a:  integer); begin
   writeln('Периметр квадрата =  ',  4*a);
end;
procedure Perimetr (a, b:  integer);
begin
writeln('Периметр прямоугольника =  ', 2*a +
2*b); end;
procedure Perimetr (a, ь,  c: integer) ; begin
  writeln('Периметр треугольника =  ',
 a + b + c) ;
end; end.
```

Будем вызывать их, задавая различные значения параметров:

```
begin
 Perimetr(4);  Perimetr(2,3);
Perimetr(2,3,7); end.
```

Результат выполнения программы будет выглядеть так:

```
Периметр квадрата = 16. Периметр
прямоугольника = 10. Периметр
треугольника = 12.
```

В заключение отметим, что рассмотренные подходы к формированию понятий и принципов ООП в среде Pascal ABC апробированы в курсе «Теория и методика обучения информатике» и могут быть использованы при изучении программирования на углубленном уровне в средней школе, а также при подготовке учителей информатики в университетах и учреждениях повышения квалификации. Они позволяют заложить фундамент для дальнейшего изучения объектно ориентированного программирования.

Литература

1. *Заборовский, Г. А.* Структурно-модульное и событийно-ориентированное программирование в среде Pascal ABC / Г. А. Заборовский // Весці БДПУ. - Серыя 3. - 2014. - № 2. - С. 63-65.
2. *Павловская, Т. А.* Паскаль. Программирование на языке высокого уровня / Т. А. Павловская. - СПб. : Питер, 2010.-464 с.
3. *Мироненко, А. С.* Императивное и объектно-ориентированное программирование на TurboPascal и Delphi / А. С. Мироненко. -Одесса : ВМВ, 2007. - 408 с.
4. *Бабушкина, И. А.* Практикум по объектно-ориентированному программированию / И. А. Ба-

бушкина, С. М. Окулов. - М. : БИНОМ, 2012. -366 с.
5. *Тервщук, В. А.* Информатика в школе. Pascal ABC в теории и на практике / В. А. Терещук, Г. Т. Филиппова. - Минск : Аверсэв, 2009. -128 с.
6. *Пенкрат, В. В.* Программирование на языке Pascal ABC / В. В. Пенкрат. - Минск : БГПУ, 2011. - 67 с.
7. *Цветков, А. С.* Язык программирования Pascal. Система программирования ABC Pascal / А. С. Цветков. - СПб.: 2013.-46 с.
8. *Долинер, Л. И.* Основы программирования в среде PascalABC.NET/Jl. И. Долинер. - Екатеринбург : Изд-во Урал, ун-та, 2014. - 128 с.

Поступила в редакцию 14.04.2015 г.