УДК 004. 42(07)

Г.А. Заборовский, *кандидат физико-математических наук,*
доцент кафедры информатики и основ электроники БГПУ

СТРУКТУРНО-МОДУЛЬНОЕ И СОБЫТИЙНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В СРЕДЕ PASCAL ABC

Теория и практика программирования в последнее время претерпели существенные изменения. Появились и стали популярными новые языки и технологии. Эволюция парадигм программирования подробно прослежена для многих языков [1, 2].

Изучение различных подходов к программированию в среде Pascal ABC актуально в связи с использованием этого языка в школах республики [3, 4] и необходимостью обеспечить не только базовый, но и углубленный (факультативный) уровень обучения. Рассмотрим возможности реализации структурно-модульного и событийно-ориентированного программирования в учебной среде Pascal ABC.

С точки зрения методологии **структурного программирования** любая программа представляет собой структуру, построенную из трёх типов базовых конструкций: следование, ветвление, цикл. Повторяющиеся либо логически целостные фрагменты программы оформляются в виде подпрограмм, а в текст основной программы вставляются команды их вызова. В языке Pascal используют подпрограммы двух типов: процедуры и функции. Наряду со стандартными процедурами и функциями можно создавать и использовать собственные. Все используемые процедуры и функции предварительно должны быть описаны: стандартные в модулях системы (System, GraphABC…), а собственные - в разделе описаний программы и/или в модулях пользователя.

Модульность в языках программирования — принцип, согласно которому программа разделяется на отдельные фрагменты. Модуль представляет собой именованный фрагмент программного кода, оформленный в виде отдельного файла и предназначенный для использования в других программах. В модули могут выделяться: структуры данных, библиотеки процедур и функций, классы и другие программные единицы, реализующие некоторую функциональность. Модули могут объединяться в пакеты, библиотеки и т.д.

Рассмотрим реализацию структурно-модульного подхода на примере программы **Doroga**, которая рисует в графическом окне три светофора и три дорожных знака.

Поместим описания процедур рисования светофора и знака в модуле Gai. Изобразим светофор в виде серого прямоугольника с тремя кругами красного, желтого и зеленого цвета. Пусть параметрами процедуры svetofor(x,y,R) являются целочисленные координаты середины верхнего края светофора и радиус круга. Относительно них задаются все координаты внутри процедуры, например красный круг Circle(x, y+R+4, R). В процедуре рисования знака (белый прямоугольник на красном круге) znak(x,y,R) используются целочисленные координаты центра знака и радиус круга.

```
unit Gai;
uses GraphABC;
procedure svetofor(x,y,R: integer);
  begin
    SetPenColor(clBlack);
    SetBrushColor(clGray);  Rectangle(x-R-4, y, x+R+4, y+6*R+12);
    SetBrushColor(clRed);    Circle(x,y+R+4, R);
    SetBrushColor(clYellow); Circle(x,y+3*R+6, R);
    SetBrushColor(clGreen);  Circle(x, y+5*R+8, R);
  end;
procedure znak(x,y,R: integer);
  begin
    SetPenColor(clBlack);  Circle(x,y,R);
    SetBrushColor(clRed); Circle(x,y,trunc(0.8*R));
    SetBrushColor(clWhite);
    Rectangle(x-trunc(0.6*R),y-trunc(0.3*R),x+trunc(0.6*R),y+trunc(0.3*R));
  end;
end.
```

В результате структурирования и размещения процедур в отдельном модуле программа **Doroga** предельно упрощается и сводится к их вызову с заданными параметрами (установке светофоров и знаков).

```
Program Doroga;
uses GraphABC, Gai;
begin
   setWindowSize(600,300);
   svetofor(120,40,10);  svetofor(50,120,20); svetofor(380,20,40);
   znak(40,40,30);  znak(230,180,90);  znak(500,100,60);
end.
```

Конечно, можно нарисовать три светофора и знака без процедур и модулей, но код программы значительно удлинится, а читабельность ухудшится.

Структурно-модульный подход оказывается весьма эффективным при разработке больших однотипных программ, предоставляя программистам удобную для многократного использования функциональность в виде набора процедур, функций, классов, констант. Так, для программирования задач по геометрии или черчению полезно создать модуль figures и описать в нем необходимые для дальнейшей работы фигуры, которые отсутствуют в стандартном модуле GraphABC, например, треугольники, параллелограммы, трапеции. Для программирования задач на построение графиков полезен модуль с набором координатных осей и сеток и т. п. В программах с повторяющимися расчетами (например, вычислениями расстояний, площадей, объемов, напряженностей электрических и магнитных полей и т.п.) бывает полезным использовать функции пользователя, особенно, если вычисленные значения используются в дальнейшем.

Важным качеством современных программ является возможность интерактивного управления объектами. Под **интерактивным управлением** понимают изменение параметров (данных) пользователем в процессе работы программы с помощью клавиатуры и мыши. В настоящее время интерактивное управление используется в большинстве программ из различных предметных областей, наибольшую эффективность оно показывает в моделирующих, графических и игровых программах.

Традиционно управление объектами основано на обработке кодов нажатых клавиш в цикле *repeat … c:=ReadKey … until c=#27* (условием завершения цикла является, например, нажатие клавиши Esc - код 27). Текст программы при этом бывает весьма сложен, а реализовать управление с помощью мыши крайне затруднительно.

Способ построения программы, при котором в коде выделяются фрагменты, состоящие из двух частей: выборки события и обработки события, называют **событийно-ориентированным** программированием (event-driven programming). Реализацию программ, управляемых событиями, в системе Pascal ABC обеспечивает модуль Events. Он работает в паре с модулем GraphABC, поскольку в Pascal ABC все события связаны с графическим окном.

Событийно-ориентированная программа после запуска и выполнения своего основного блока *begin….end* не завершается, а продолжает отслеживать возникающие события. Каждому событию соответствует своя процедурная переменная: OnMouseDown и OnMouseUp - нажатие и отпускание кнопки мыши; OnMouseMove - перемещение мыши; OnKeyDown и OnKeyUp - нажатие и отпускание клавиши; OnKeyPress - нажатие символьной клавиши; OnResize и

OnClose - изменение размеров и закрытие графического окна. До начала работы программы эти переменные имеют нулевые значения. Чтобы при некотором событии выполнялось определенное действие, необходимо процедурной переменной присвоить конкретную **процедуру – обработчик события**. Эта процедура вызывается всякий раз при возникновении данного события.

Рассмотрим реализацию событийно-ориентированного подхода на примере программы графический редактор GrafRed. Пусть наш простейший редактор позволяет рисовать линии, задавать их цвет и толщину, использовать ластик, очищать графическое окно. Все это несложно реализовать с помощью процедур обработки событий мыши и клавиатуры.

Прежде всего определим две процедуры – обработчики нажатия клавиши мыши mDown и перемещения мыши mMove. Параметры x и y определяют координаты указателя мыши в момент наступления события; параметр mb=0 (кнопка мыши не нажата), mb=1 (нажата левая кнопка мыши). Если при перемещении мыши нажата левая кнопка, то рисуется линия.

Цвет линий будем изменять нажатием буквенных клавиш: красный 'r', зеленый 'g', синий 'b', черный 'k', белый (стирание) 'w' , а толщину увеличивать или уменьшать клавишами управления курсором (стрелки вверх Up и вниз Down). Очищать окно будем нажатием клавиши Del. Напишем процедуры обработки событий нажатия соответствующих клавиш kPress и kDown. Параметр ch в обработчике kPress определяет нажатый символ. Параметр key в обработчике kDown определяет *виртуальный код* нажатой клавиши. В модуле Events определены именованные целые константы, представляющие виртуальные коды клавиш, например, управления курсором: VK_Up, VK_Down и т.п. Символьные клавиши '0'..'9' и 'A'..'Z' имеют виртуальный код, равный коду соответствующего символа. Используем полученные значения цвета и толщины линии в процедуре перемещения мыши mMove.


```
Program GrafRed;
uses GraphABC, Events;
  var w, col: integer;
    { процедура - обработчик нажатия кнопки мыши }
procedure mDown(x,y,mb: integer);
  begin
    MoveTo(x,y);
  end;
```

```
    { процедура - обработчик перемещения мыши }
procedure mMove(x,y,mb: integer);
  begin
    setPenColor(col); SetPenWidth(w);
    if mb=1 then LineTo(x,y);
  end;
    { процедура - обработчик нажатия  символьных клавиш }
procedure kPress(ch: char);
  begin
    if ch='r'  then col:=clRed;
    if ch='g'  then col:=clGreen;
    if ch='b'  then col:=clBlue;
    if ch='w'  then col:=clWhite;
    if ch='k'  then col:=clBlack;
  end;
    { процедура - обработчик нажатия клавиш управления курсором }
procedure kDown(key: integer);
  begin
    if key = VK_Up  then  w:=w+1;
    if key = VK_Down  then  w:=w-1;
    if key = VK_Delete  then ClearWindow(clWhite);
  end;
    { основная часть программы }
begin
  OnKeyPress:=kPress;
  OnKeyDown:=kDown;
  OnMouseDown:=mDown;
  OnMouseMove:=mMove;
end.
```


Основная часть программы получается чрезвычайно простой и сводится к привязке четырех процедур-обработчиков к событиям мыши и клавиатуры.

В заключение отметим, что рассмотренные подходы к программированию и их реализация в среде Pascal ABC 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. Кауфман В. Ш. Языки программирования. Концепции и принципы. / В. Ш. Кауфман - М.: ДМК Пресс. 2010. - 464 с.
2. Мироненко. А. С. Императивное и объектно-ориентированное программирование на Turbo Pascal и Delphi / А. С. Мироненко. - Одесса.: ВМВ. 2007. – 408 с.
3. Заборовский, Г.А. Информатика: учеб. пособие для 11 кл. общеобразоват. учреждений / Г. А. Заборовский, А. Е. Пупцев. - Минск: Нар. Асвета. 2010. – 150 с.
4. Заборовский, Г.А. Информатика в 11 классе : учеб-метод. пособие для учителей учреждений общ. сред. образования / Г. А. Заборовский, О. Н. Лапко. - Минск: Нар. Асвета. 2012. – 112 с.