Сопоставление с образцом в языке программирования PascalABC.NET

Михалкович С.С.

Южный федеральный университет,
факультет математики,
механики и компьютерных наук
miks@math.sfedu.ru

Доклад на XXVI научной конференции
«Современные информационные технологии: тенденции и перспективы развития (СИТО 2019)»
(Ростов-на-Дону, 18-19 апреля 2019 г.)

Этапы развития языка PascalABC.NET

- 2003 г. – выпуск Pascal ABC
- **2007 г.** – выпуск первой версии PascalABC.NET
- 2010 г. – методы расширения
- 2010 г. – лямбда-выражения
- 2015 г. – переход на свободную лицензию LGPLv3
- 2015 г. – тип последовательности sequence of T
- 2016 г. – кортежи, кортежное присваивание
- 2016 г. – срезы массивов, списков, строк
- 2018 г. – сопоставление с образцом, классы типов
- **2019 г.** – улучшенные конструкции сопоставления с образцом

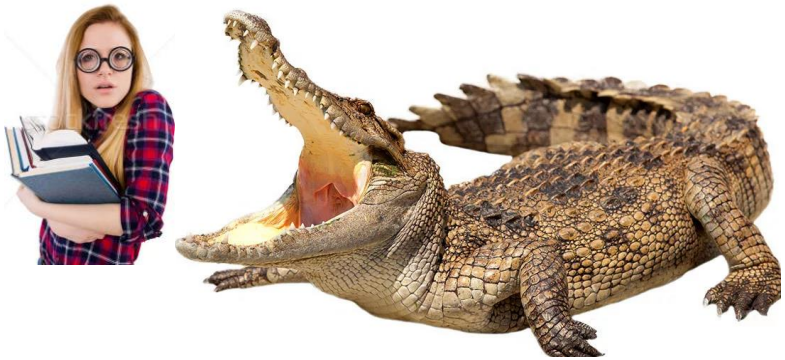
Pattern Matching в языках программирования

- Pattern Matching (сопоставление с образцом) – прерогатива функциональных языков программирования (ML, Haskell, OCaml, Erlang, F#)
- Pattern Matching в последние годы стал активно внедряться в "большие" языки программирования: Scala, C#
- Пока нет Pattern Matching в C++, Java, JavaScript, Python
- Отдельные конструкции Pattern Matching появились в PascalABC.NET в 2018 году.
- Улучшенные конструкции Pattern Matching - PascalABC.NET, лето 2019 года.

Пример: расширенная операция is

Обычная операция is:

```
var c := new Crocodile;
var st := new Student;
var x: Object := c;
if x is Crocodile then
  (x as Crocodile).Eat(st);
```



Недостаток. Чтобы съесть студента, тип приводится дважды.

Расширенная операция is:

```
if x is Crocodile (var croco) then
  croco.Eat(st);
```

Переменная x преобразуется в переменную croco типа Crocodile непосредственно в условии if

Пример: несколько возможных типов

Для разных типов выполняются различные операции

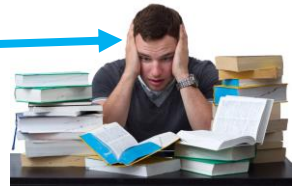
```
var st := new Student;
var x: Object := new Teacher;
if x is Crocodile (var croco) then
  croco.Eat(st)
else if x is Student (var stud) then
  stud.PassExams(3)
else if x is Teacher (var t) then
  t.TakeExam(st);
```

Это невозможно реализовать с помощью полиморфизма – типы Student, Teacher и Crocodile не имеют ничего общего.

Пример: оператор match сопоставления с образцом

Тот же код можно написать, используя сопоставление с образцом:

```
var st := new Student;
var x: Object := new Teacher;
match x with
  Crocodile(croco): croco.Eat(st);
  Student(stud): stud.PassExams(3)
  Teacher(t): t.TakeExam(st);
end;
```



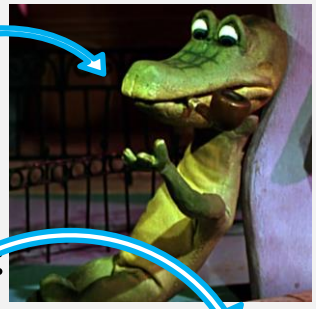
Таким образом, идёт **сопоставление с типом**.

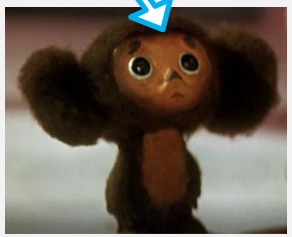
А можно – со значениями? **ДА!**

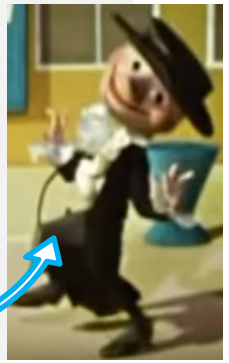
Полиморфный список, где каждый объект занимается своим делом

Обработка всех объектов **полиморфного** списка. Сопоставление со значениями.

```
var l := new List<Object>;
l.Add(new OldLady("Шапокляк",65));
l.Add(new Animal("Чебурашка",1));
l.Add(new Crocodile("Гена",20));
foreach var x in l do
  match x with
    Crocodile(c) when c.Name = "Гена":
      c.Employ("крокодилом");
    Crocodile(c): c.Eat(new Student);
    Animal(a) when a.Name = "Чебурашка":
      a.BeFriendOf(l[1]);
    OldLady(ol) when ol.Name = "Шапокляк":
      ol.Advice("Не тратить время зря");
  end;
```







Полиморфный список графических фигур

Изменение графических фигур в зависимости от типа и свойств

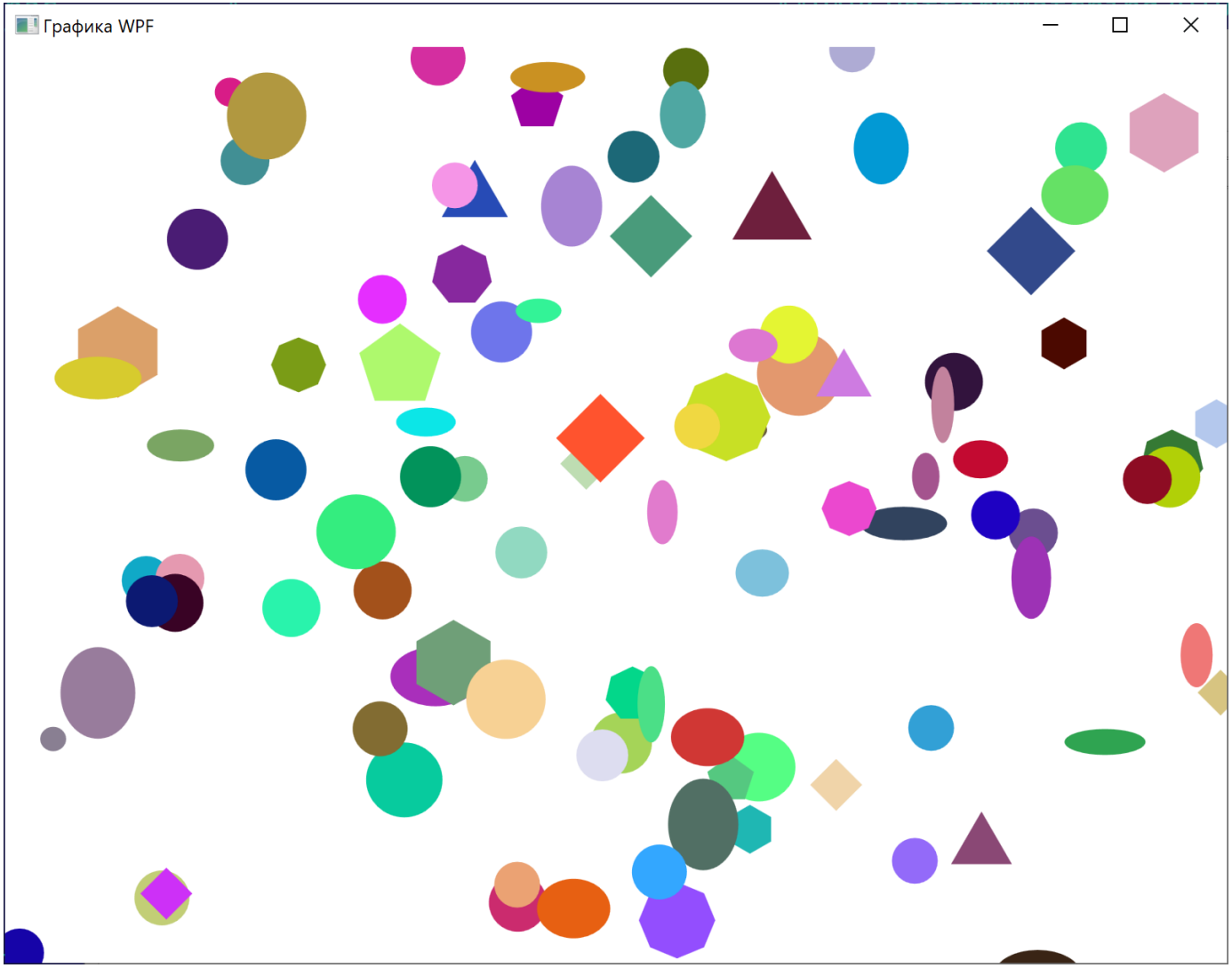
```
uses WPFObjects;

begin
  loop 100 do
    GenRandomWPF;

  Sleep(1000);
  foreach var o in Objects do
    match o with
      CircleWPF(c): c.Radius += 10;
      EllipseWPF(e): e.AnimMoveOn(Random(-50,50),Random(-50,50),1);
      RegularPolygonWPF(r) when r.Count<6: r.WithBorder;
    end
end.
```

Круги увеличивают радиус на 10.

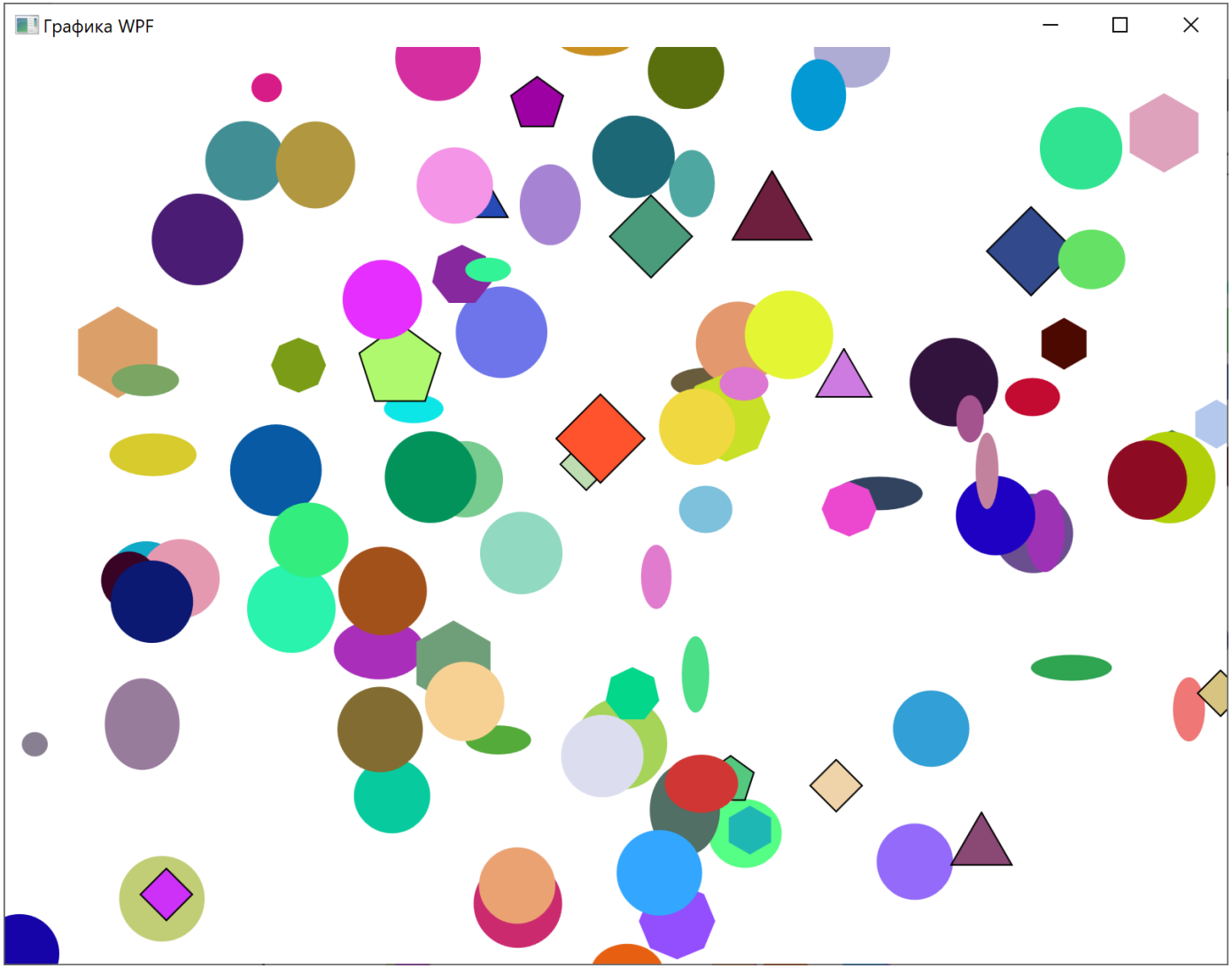
Эллипсы анимируются в случайном направлении.

Многоугольники с количеством углов до 6 получают границу

Полиморфный список графических фигур (2)



Полиморфный список графических фигур (3)



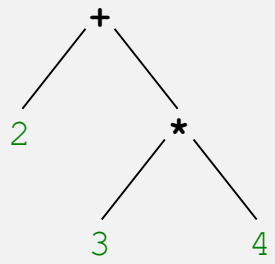
Более серьёзный пример – интерпретатор выражения, заданного деревом разбора

Узлы дерева – Cons (константа), Add, Mult. Функция Eval

```
type
  Expr = interface
  end;
  Cons = auto class(Expr)
    r: real;
  end;
  Add = auto class(Expr)
    left,right: Expr;
  end;
  Mult = auto class(Expr)
    left,right: Expr;
  end;
```

Автоклассы дерева разбора

```
function Eval(e: Expr): real;
begin
  match e with
    Cons(c): Result := c;
    Add(l,r): Result := Eval(l) + Eval(r);
    Mult(l,r): Result := Eval(l) * Eval(r);
  end;
end;

begin
  // 2 + 3 * 4
  var r := AddC(
    ConsC(2),
    MultC(ConsC(3),ConsC(4))
  );
  Eval(r).Print;
end.
```



Глубокий Pattern matching. Упрощение выражения

Упрощение выражения **5 + 1 * (x + 0) + 0 * (2 + a)**, заданного деревом

```
function Simplify(e: Expr): Expr;
begin
  match e with
    Mult(Cons(c),Cons(c1)): Result := ConsC(c*c1);
    Mult(Cons(1),ex): Result := Simplify(ex);
    Mult(Cons(0),ex): Result := ConsC(0);
    Add(Cons(0),ex): Result := Simplify(ex);
    Add(ex,Cons(0)): Result := Simplify(ex);
    Add(Cons(c),Cons(c1)): Result := ConsC(c+c1);
    Mult(e1,e2): Result := MultC(Simplify(e1),Simplify(e2));
    Add(e1,e2): Result := AddC(Simplify(e1),Simplify(e2));
    else Result := e;
  end;
end;
begin
  var r: Expr := AddC(AddC(ConsC(5),MultC(ConsC(1),AddC(VC('x'),ConsC(0)))),
                 MultC(ConsC(0),AddC(ConsC(2),VC('a'))));
  // ((5 + (1 * (x + 0))) + (0 * (2 + a)))
  r := Simplify(r);
  Println(Str(r)); // (5 + x)
end.
```

Выводы

Использование оператора сопоставления с образцом позволяет:

- в увлекательной форме рассказывать школьникам основы объектно-ориентированного программирования
- проще записывать алгоритмы обработки полиморфных структур данных (списки, деревья)
- компактнее излагать достаточно сложные алгоритмы, связанные с построением компиляторов: создание интерпретаторов и упрощение выражений
- позволяет языку программирования PascalABC.NET оставаться современным развивающимся языком