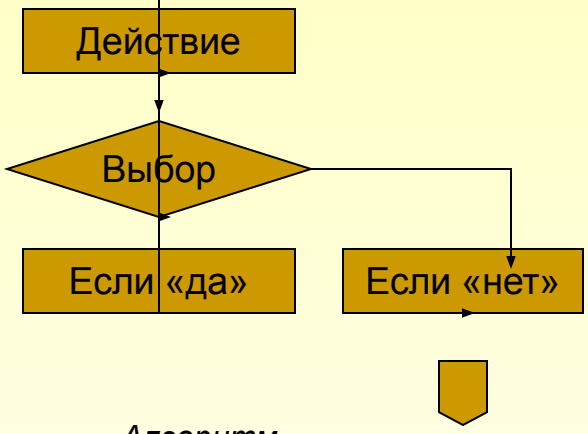
Функциональное программирование

Курс лекций для студентов
4 курса ЕНФ

Глава 1. Элементы функционального программирования

1.1. Введение в функциональное программирование

- Архитектура фон Неймана диктует стиль программирования?
- Средства программирования:
  - Арифметические и логические операции;
  - Присваивания;
  - Последовательное исполнение шагов алгоритма;
  - Управление (управляющие конструкции);
  - Процедуры и функции;
  - Модули, исключительные ситуации, структуры данных,...
- Программа: описание процесса (алгоритма) или «черный ящик»?



Алгоритм

Функция

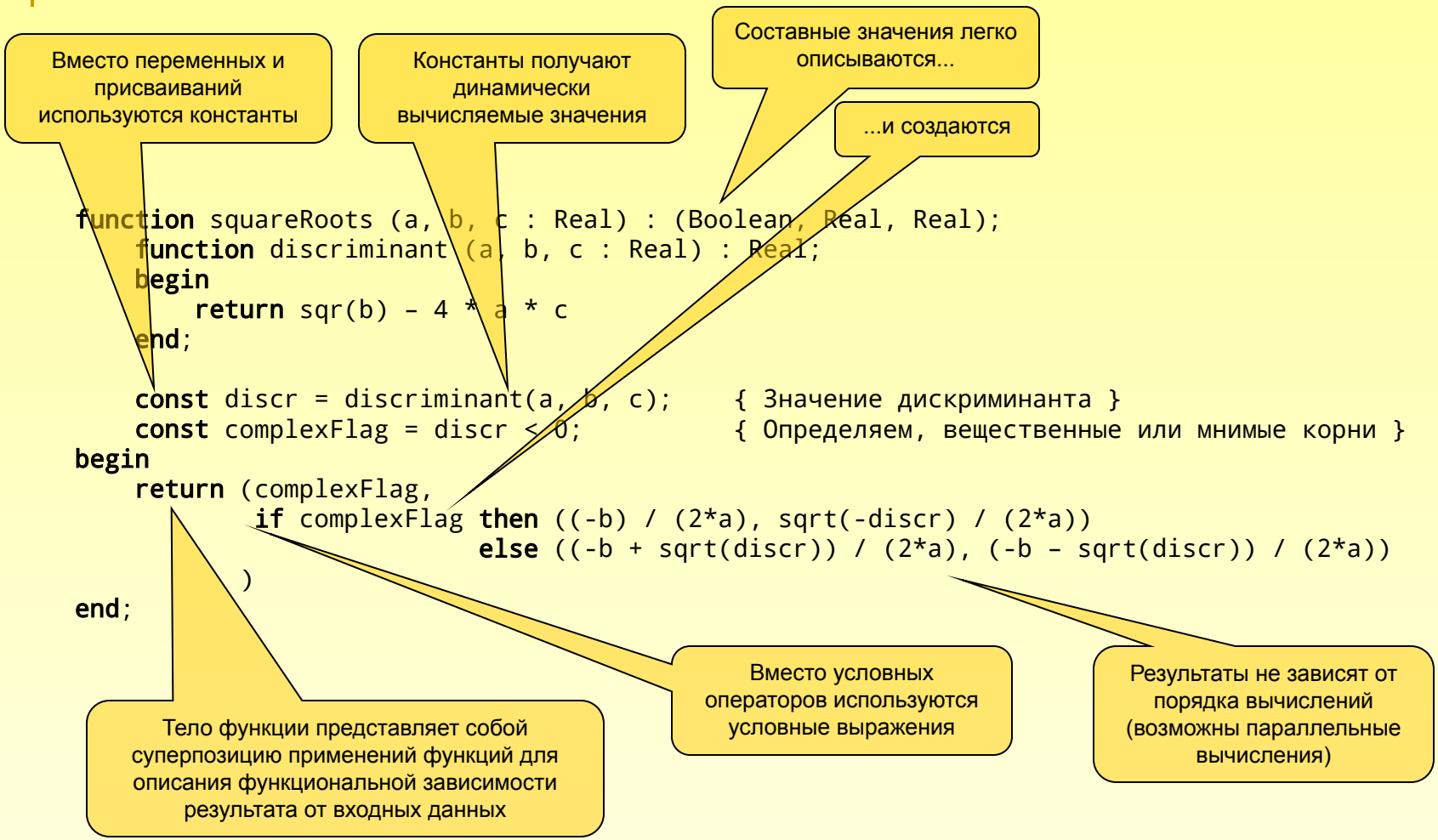
Задача о вычислении значений квадратных корней уравнения (процедурный стиль программирования)

```
{ Процедура вычисляет вещественные или комплексные корни квадратного трехчлена,
  в предположении, что первый коэффициент (a) отличен от нуля.
  Аргументы:  a, b, c – коэффициенты квадратного трехчлена;
  Результаты: complexFlag – признак комплексных корней;
              r1, r2 – вещественные корни, если complexFlag = False и
                       вещественная и мнимая части двух корней, если complexFlag = True
}
procedure squareRoots (a, b, c : Real ; var complexFlag : Boolean; var r1, r2 : Real);
    function discriminant (a, b, c : Real) : Real;
    begin
        discriminant := sqr(b) – 4 * a * c
    end;

    var discr : Real;                 { Значение дискриминанта }
begin
    discr := discriminant (a, b, c);  { Вычисляем дискриминант }
    complexFlag := discr < 0;         { Определяем, вещественные или мнимые корни }
    if complexFlag then begin
        r1 := (-b) / (2*a);           { Вещественная часть корней }
        r2 := sqrt(-discr) / (2*a)    { Мнимая часть корней }
    end else begin
        r1 := (-b + sqrt(discr)) / (2*a);
        r2 := (-b – sqrt(discr)) / (2*a)
    end
end;
```

Та же программа, написанная в функциональном стиле программирования (на псевдоязыке с паскалеобразным синтаксисом)

```
{ Функция вычисляет вещественные или комплексные корни квадратного трехчлена,
  в предположении, что первый коэффициент (a) отличен от нуля.
  Аргументы:  a, b, c – коэффициенты квадратного трехчлена;
  Результаты: признак комплексных корней;
              вещественные корни, если они вещественные и
                      вещественная и мнимая части двух корней, если мнимые
}
function squareRoots (a, b, c : Real) : (Boolean, Real, Real);
    function discriminant (a, b, c : Real) : Real;
    begin
        return sqr(b) – 4 * a * c
    end;

    const discr = discriminant(a, b, c);    { Значение дискриминанта }
    const complexFlag = discr < 0;          { Определяем, вещественные или мнимые корни }
begin
    return (complexFlag,
            if complexFlag then ((-b) / (2*a), sqrt(-discr) / (2*a))
                           else ((-b + sqrt(discr)) / (2*a), (-b – sqrt(discr)) / (2*a))
           )
end;
```

Особенности этой программы



Функциональное представление множеств

```
type intSet = function (Integer) : Boolean;      { описание функционального типа данных }

function emptySet (n : Integer) : Boolean;       { пустое множество }
begin return False end;

function oddSet (n : Integer) : Boolean;         { множество нечетных чисел }
begin return n mod 2 = 1 end;
```

Несколько полезных операций над множествами

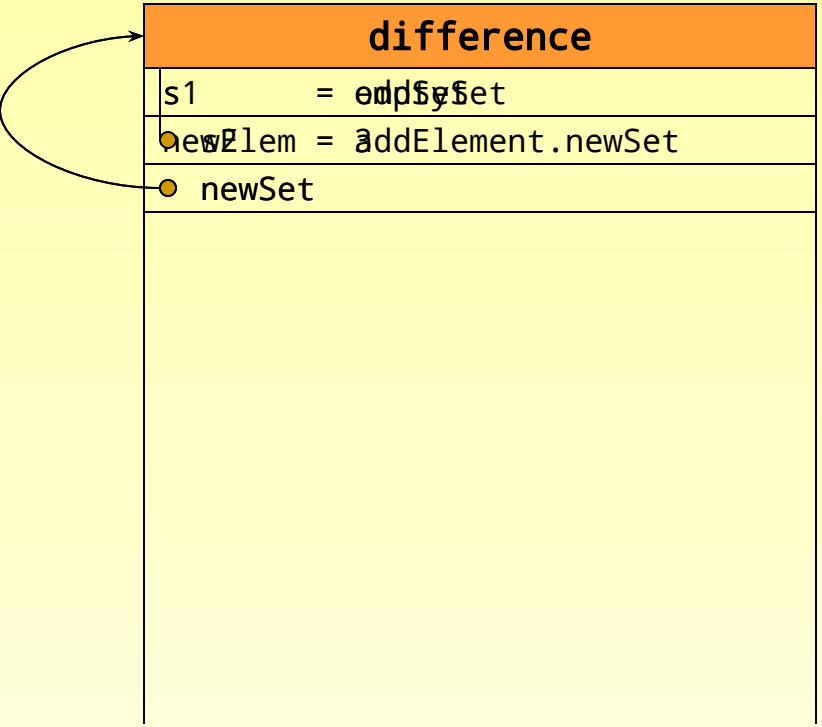
```
function addElement (s : intSet; newElem : Integer) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return s(n) or (n = newElem) end;
begin return newSet end;

function buildInterval (min, max : Integer) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return (n >= min) and (n <= max) end;
begin return newSet end;


function difference (s1, s2 : intSet) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return s1(n) and not s2(n) end;
begin return newSet end;
```

Будут ли работать эти операции?

Попробуем вычислить выражение

```
difference (oddSet, addElement (emptySet, 3)) (7)  { Принадлежит ли 7 множеству [odds] \ [3] }
```

Стек вычислений

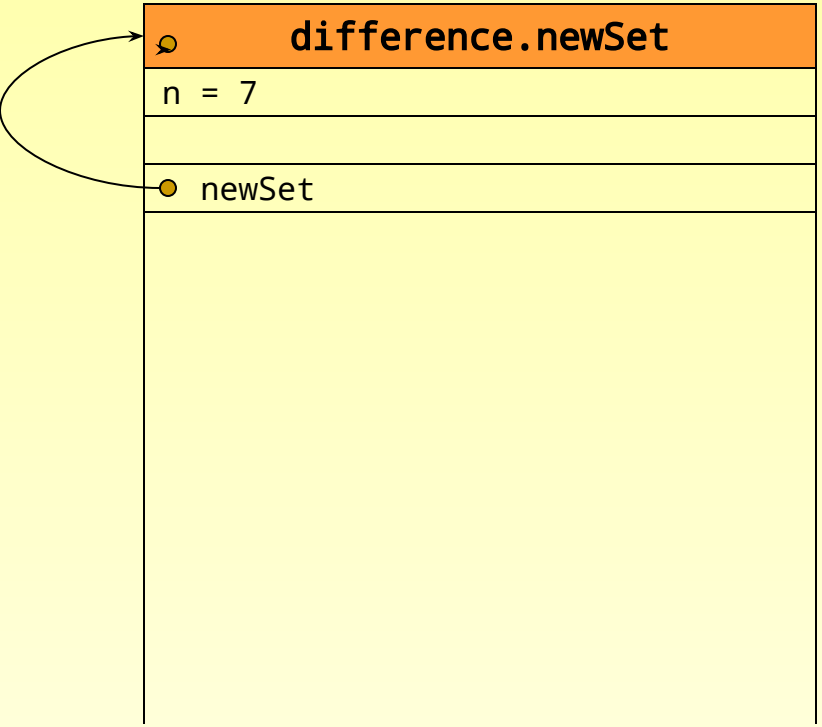



```
function emptySet (n : Integer) : Boolean;
begin return False end;

function oddSet (n : Integer) : Boolean; begin
return n mod 2 = 1 end;

function addElement
        (s : intSet; newElem : Integer) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return s(n) or (n = newElem) end;
begin return newSet end;

function difference (s1, s2 : intSet) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return s1(n) and not s2(n) end;
begin return newSet end;
```

Попробуем вычислить выражение

```
difference (oddSet, addElement (emptySet, 3)) (7)  { Принадлежит ли 7 множеству [odds] \ [3] }
```

Стек вычислений




```
function emptySet (n : Integer) : Boolean;
begin return False end;

function oddSet (n : Integer) : Boolean; begin
return n mod 2 = 1 end;

function addElement
        (s : intSet; newElem : Integer) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return s(n) or (n = newElem) end;
begin return newSet end;

function difference (s1, s2 : intSet) : intSet;
    function newSet (n : Integer) : Boolean;
    begin return s1(n) and not s2(n) end;
begin return newSet end;
```

Подведение итогов

- Императивные языки служат для описания процессов; функциональные – для описания функций, вычисляющих результат по исходным данным.
- На традиционных языках можно писать в функциональном стиле, однако средств работы с функциями в традиционных языках недостаточно.
- Традиционные способы реализации языков программирования плохо подходят для программ, написанных в функциональном стиле.
- Традиционные языки не могут обеспечить удобных средств для распараллеливания вычислений: последовательное выполнение команд – узкое место традиционной архитектуры компьютеров («фон-Неймановское горлышко»).
- **Для функционального программирования требуются специализированные языки**

Литература

1. А.Филд, П.Харрисон. Функциональное программирование. «Мир», Москва, 1993. – 637 с.
2. П.Хендерсон. Функциональное программирование. Применение и реализация. «Мир», Москва, 1983. – 349 с.
3. Р.В.Душкин. Функциональное программирование на языке Haskell. «ДМК Пресс», Москва, 2007. – 608 с.
4. IBM developerWorks. Beginning Haskell. http://www-106.ibm.com/developerworks/edu/os-dw-linuxhask-i.html
5. P.Hudak, J.Peterson, J.Fasel. Gentle Introduction to Haskell. http://haskell.org/tutorial