6- Abstração

- Abstração é o processo de identificar as qualidades ou propriedades importantes do fenômeno que está sendo modelado
- Em programação, a abstração faz referência à distinção entre O QUE uma parte do programa faz e COMO ela é implementada
- Exemplo: as construções de uma LP são abstrações do código de máquina
- A abstração é importante para a construção de programas grandes. Por exemplo, o programador pode introduzir vários níveis de abstração, através da implementação de procedimentos

6- Abstração...

- Tipos de abstrações
  - Uma abstração é uma entidade que incorpora (personifica) uma computação
  - Exemplo: uma abstração de função incorpora uma expressão a ser avaliada

  Abstrações de função

  - Uma abstração de função incorpora uma expressão a ser avaliada e, quando chamada, produz um valor como resultado
  - Visão do usuário: uma chamada da função irá mapear os argumentos a um resultado
  - Visão do programador: uma chamada da função irá avaliar o corpo da função, tendo os parâmetros formais amarrados aos argumentos correspondentes

6- Abstração...

Def. de função em Pascal	Def. de função em ML
function pot (x: Real; n: Integer): Real; begin if (n = 1) then pot := x; else pot := x * pot (x, n-1); end;	**fun pot (x: real, n: int) = if n = 1 then x else x * pot (x, n-1)**

- Críticas à notação de Pascal:
  - O corpo da função sempre possui comandos
  - O retorno da função é definido através de uma atribuição a uma pseudo-variável $\Rightarrow$ é possível chegar ao fim do corpo de uma função, sem executar uma atribuição
  - O identificador de função denota duas entidades diferentes em um mesmo escopo
  - O corpo da função é, sintaticamente, um comando, mas, semanticamente, é um tipo de expressão

6- Abstração...

- Qual o significado de uma definição de função do tipo **function I (FP_1, …, FP_n) is E**?

⇓

O identificador **I** é amarrado a uma determinada abstração de função, que é uma entidade que possui a propriedade de retornar um resultado sempre que for chamada com argumentos apropriados

- É perfeitamente possível para uma LP separar os conceitos de abstração e amarração. Por exemplo, em ML, podemos escrever: **fn (x:real) ⇒ x * x * x;**

6- Abstração...

Abstrações de procedimento

- Uma abstração de procedimento incorpora um comando a ser executado e, quando chamado, irá atualizar o valor de variáveis
- Visão de usuário: uma chamada de procedimento irá atualizar variáveis, de uma maneira que será influenciada por seus argumentos
- Visão do programador: uma chamada de procedimento irá executar o corpo do procedimento, tendo os parâmetros formais amarrados aos argumentos correspondentes

6- Abstração...

- Qual o significado de uma definição de procedimento do tipo **procedure I (FP_1, …, FP_n) is C**?

⇓

O identificador **I** é amarrado a uma determinada abstração de procedimento, que é uma entidade que possui a propriedade de atualizar variáveis sempre que for chamada com argumentos apropriados

Princípio da Abstração

- Princípio da abstração = é possível construir abstrações sobre qualquer classe sintática, desde que as sentenças dessa classe especifiquem algum tipo de computação

6- Abstração...

- Uma abstração de seletor é uma abstração sobre um acesso a variável. Em outras palavras, uma abstração de seletor possui um corpo, que é um acesso a variável, e uma chamada de seletor é um acesso a variável que retorna uma referência a uma variável através da avaliação do corpo de uma abstração de seletor
- Exemplo: supondo que Pascal permitisse a definição de abstrações de seletor, então poderíamos definir o seguinte:

```
selector primeiro (var f: fila): integer is
...{retorna uma referência ao primeiro item em f}
```

E escrever as seguintes chamadas:

```
I := primeiro (filaA);
primeiro (filaA) := primeiro (filaA) – 1;
```

6- Abstração...

- Como muitas linguagens imperativas, Pascal possui seletores pré-definidos, mas não oferece recursos para aumentar este repertório
- O Princípio da Abstração ajuda a identificar formas de tornar a linguagem mais expressiva e mais regular

6- Abstração...

- Parâmetros
  - Parâmetro formal = um identificador utilizado dentro de uma função para denotar um argumento
  - Parâmetro real = uma expressão (ou outra sentença) que produz um argumento
  - Argumento = um valor que pode ser passado para uma abstração. Cada LP possui o seu próprio conjunto de valores que podem ser passados como argumentos
  - Mecanismo de passagem de parâmetro = método de associação entre parâmetros formais e reais. Há vários tipos

6- Abstração...

Mecanismos de cópia

- Um mecanismo de cópia permite que valores sejam copiados para dentro ou para fora de uma abstração, quando chamada
- Os parâmetros formais denotam variáveis locais $\Rightarrow$ criação na entrada da abstração e destruição na saída
- Passagem de valor = na entrada da abstração, uma variável local X é criada e recebe como valor inicial o valor do argumento. Qualquer alteração de X não afeta outra variável não local
- Passagem de resultado = o argumento deverá ser (uma referência a) uma variável. Uma variável X é criada, sem valor inicial. Na saída da abstração, o valor final de X é atribuído à variável que é o argumento
- Passagem de valor-resultado = combinação dos dois mecanismos anteriores

6- Abstração...

- Exemplo: Ada

```
procedure Inclui (i: in ITEM; f: in out FILA) is
  ...{retorna uma fila com mais um item}
end;

procedure Retira (i: out ITEM; f: in out FILA) is
  ...{retorna um item e a fila com menos um item}
end;
```

- Desvantagem: a cópia de valores compostos pode ter custo alto

6- Abstração...

Mecanismos de definição

- Um mecanismo de definição permite que um parâmetro formal X seja amarrado diretamente ao argumento
- Parâmetro constante = o argumento é um valor (de primeira classe). X é amarrado ao valor do argumento durante a ativação da abstração chamada
- Parâmetro variável (ou de referência) = o argumento é uma referência a uma variável $\Rightarrow$ qualquer utilização de X é, na verdade, uma utilização indireta do argumento
- Parâmetro procedimental $\Rightarrow$ o argumento é uma abstração de procedimento $\Rightarrow$ qualquer chamada a X é, na verdade, uma chamada indireta ao argumento (que é um procedimento)
- Parâmetro funcional $\Rightarrow$ o argumento é uma abstração de função $\Rightarrow$ qualquer chamada a X é, na verdade, uma chamada indireta ao argumento (que é uma função)
- A passagem de subprogramas como parâmetros tem a vantagem de permitir que um mesmo subprograma possa executar funções diferentes. Mas pode ser obscuro

6- Abstração...

- Exemplo

```
program param (input, output)

  procedure b (function h (n: integer): integer);
  begin
    writeln(h(2));
  end;

  procedure c;
   var m: integer;

   function f(n: integer): integer;
   begin
     f := m + n;
   end;

  begin
    m := 0;
    b (f);
  end;

begin
  c;
end.
```

- Os parâmetros constantes e variável possibilitam um poder de expressão similar aos dos mecanismos de cópia. A escolha é uma decisão importante para o projetista da linguagem

6- Abstração...

- Exemplo: que valor será impresso por esse programa, supondo: a) passagem de valor-resultado, e; b) passagem de variável?

```
program doido (input, output)
var A: integer;

  procedure sei_la (var x: integer);
  begin
    x := 2; A := 0;
  end;

begin
  A := 1;
  sei_la (A);
  writeln(A);
end.
```

6- Abstração...

- Mecanismos de definição X mecanismos de cópia
  - O mecanismo de definição possui uma semântica mais simples
  - O mecanismo de definição, em geral, é mais eficiente
  - Uma desvantagem dos parâmetros variáveis é a possibilidade de utilização de apelidos (“aliasing”), o que torna o programa mais difícil de entender
- Exemplo:

```
procedure confuso (var m, n: integer);
begin
  n := 1;
  n := m + n;
end;
```

- Supondo **i = 4**, qual será o valor de **i** após a chamada **confuso (i, i)**?

6- Abstração...

O Princípio da Correspondência

- Pode-se perceber uma correspondência entre alguns mecanismos de passagem de parâmetro e alguns tipos de declarações. A diferença é que uma declaração especifica o identificador e a entidade a qual será amarrada. Uma especificação de parâmetro formal especifica somente identificador (às vezes, o seu tipo) e o argumento vem de alguma outra parte do programa (parâmetro real). Exemplo:

const x = E;	**procedure P (const X: T); ...;** **…** **P(E)**

6- Abstração...

- Por questões de simplicidade e regularidade, um projetista de LP pode optar por eliminar todas as diferenças entre declarações e mecanismos de passagem de parâmetros
- Princípio da correspondência: para cada forma de declaração, existe um mecanismo de passagem de parâmetro e vice-versa

6- Abstração...

- Ordem de avaliação
  - Quando uma abstração é chamada, em que momento cada parâmetro real é avaliado?
    - Avaliação prévia (ou de ordem aplicativa) – no ponto da chamada. O parâmetro real é avaliado uma única vez e o seu valor substitui cada ocorrência do parâmetro formal
    - Avaliação de ordem normal – no momento em que o argumento for realmente utilizado. O parâmetro formal é substituído pelo parâmetro real
      - Avaliação tardia – o argumento será avaliado somente na primeira vez que for utilizado

6- Abstração...

- O resultado pode variar dependendo da ordem de avaliação
- Exemplo:

```
fun cand (b1: bool, b2: bool) =
  if b1 then b2 else false
```

  Supondo **n = 0**, qual será o resultado da chamada **cand (n>0, t/n > 0.5)**?

- Função *strict* (rigorosa) = uma chamada à função só pode ser avaliada se todos os seus argumentos puderem ser avaliados
- Função *nonstrict* = uma função é dita *nonstrict* com relação a um argumento n, se uma chamada à função puder ser avaliada, mesmo que o n-ésimo argumento não possa ser avaliado
- A avaliação de ordem normal é claramente ineficiente, quando faz o mesmo argumento ser avaliado várias vezes

6- Abstração...

- Propriedade de Church-Rosser = se uma expressão pode ser avaliada, então ela pode ser avaliada através da utilização consistente da avaliação de ordem normal. Se uma avaliação pode ser realizada em ordens diferentes (misturando ordem normal e aplicativa), então todas as ordens de avaliação produzirão o mesmo resultado
- Qualquer LP que permita efeitos colaterais não possui a propriedade de Church-Rosser
- Algol-60 permite que o programador faça uma escolha entre a avaliação prévia (passagem de valor) e avaliação de ordem normal (passagem de nome)
- Basicamente, no mecanismo de passagem de nome:
  - Cada ocorrência do parâmetro é considerada como sendo textualmente substituída pelo argumento. Isso pode levar a complicações
  - A passagem de nome é poderosa, porque permite passar funções e procedimentos, além das variáveis simples e estruturadas
  - A passagem de nome pode levar a programas que são difíceis de ler