Valores

- O que é um valor?

  Qualquer entidade que existe durante uma computação, ou seja, qualquer coisa que pode ser avaliada, armazenada, passada como parâmetro para uma função, etc.

- Por que estudar valores?

  Dados são a matéria-prima da computação.

Valores...

- Como estudar valores?
  - Agrupa-los em tipos.
- Tipos: especificação da classe de valores que podem ser associados à variável, bem como das operações que podem ser usadas para criar, acessar e modificar estes valores.
- LPs, em geral, proveem um conjunto de tipos primitivos e mecanismos para estruturar tipos compostos. Algumas oferecem a opção de tipos recursivos.

Valores...

- Tipos primitivos (tipos embutidos)
  - Valores atômicos
  - Indicam a área de aplicação da LP
  - LPs costumam dar nomes diferentes a seus tipos primitivos
  - Tipos primitivos mais comuns:
    - Lógico = {*false, true*}
    - Inteiro = {...,-2,-1,0,+1,+2,...}
    - Real = {...,-1.0,...,0,...,+1.0,...}
    - Caractere = {...,‘a’,‘b’,...,‘z’,...}
    - Obs: os tipos inteiro, real e caractere são definidos pela implementação da LP.

Valores...

- A representação subjacente fica invisível ao programador $\Rightarrow$ maior legibilidade e portabilidade dos programas
- Algumas LPs permitem a definição de um novo tipo primitivo, através da enumeração de seus valores $\Rightarrow$ *tipos enumerados*

Valores...

- Exemplo 1: PASCAL

```
type dia = (domingo, segunda, terca, quarta, quinta, sexta, sabado);
var hoje, amanha, aniversario: dia;

hoje := segunda;
amanha := succ(hoje);
aniversario := amanha;
```

- Exemplo 2: C

```
enum dia_semana (domingo = 1, segunda, terca, quarta, quinta, sexta, sabado);
```

- Algumas LPs também permitem a definição de um subconjunto de um tipo já existente, através da definição de um intervalo.
- Exemplo: **type** indice = 1..100;

Valores...

- Tipos compostos (tipos estruturados)
  - São compostos a partir de tipos simples
  - Grande variedade: tuplas, registros, variantes, uniões, arranjos, strings, listas, árvores, arquivos, relações, etc
  - Mecanismos básicos de estruturação:

  A) Produto cartesiano
    - O produto cartesiano de *n* conjuntos $S_1,S_2,...,S_n$, denotado por $S_1 \times S_2 \times ... \times S_n$, é um conjunto cujos elementos são *n*-tuplas ordenadas $(s_1,s_2,...,s_n)$, onde $s_i \in S_i$.
    - Nomes simbólicos: tuplas, registros, estruturas.
    - $\#(S_1 \times S_2 \times ... \times S_n) = \#S_1 \times ... \times \#S_n$

Valores...

- Exemplo 1: Registro em Pascal

```
type pessoa = record
                nome: string[20];
                idade: integer;
                altura: real;
              end;
```

- Exemplo 2: Tupla em ML

  type pessoa = string * int * real OU

  type pessoa = {nome: string, idade: int, altura: real }

- Caso especial: *n* = 0
  - Não é um conjunto vazio, mas um conjunto contendo uma tupla sem elementos.
  - Ex.: **unit** em ML e **void** em Algol-68 e C.

Valores...

B) União discriminada

- É um mecanismo de estruturação que especifica que uma escolha será feita dentre diferentes estruturas alternativas. Cada estrutura alternativa é chamada *variante*.
- É diferente da união de conjuntos

  Seja T = {a, b}

  $T \cup T = \{a, b\} = T$

  $T + T = \{esq\ a, esq\ b, dir\ a, dir\ b\} \neq T$
- Constitui a base de registros variantes, uniões, construções de ML e tipos algébricos de Miranda
- $\#(S_1 + S_2 + ... + S_n) = \#S_1 + ... + \#S_n$

Valores...

- Exemplo: Pascal

```
type item = record
                preco: real;
                case disponivel: boolean of
                    true: (quantidade: integer; local: string);
                    false: (mes_entrega:1..12)
            end;
```

⇓

Tags(discriminantes) são valores e podem gerar insegurança

⇒ Permite acessar um campo mesmo que ele não exista ⇒ erro de execução

⇒ Atribuição de um novo valor ao tag possui o efeito colateral de destruir um campo e criar um novo com valor indefinido. Logo, sugere-se utilizar comando **case**.

Valores...

C) Mapeamento

- É uma função de um conjunto finito de valores S em valores de um tipo T
- Um arranjo representa um mapeamento finito
- Muitas LPs possuem arranjos multidimensionais
- O conjunto do índice deve ser discreto
- $\#(S \rightarrow T) = (\#T)^{\#S}$
- Exemplo (Pascal): `type matriz = array [1..10, 0..20] of real;`
- Abstrações de funções representam um outro tipo de mapeamento
- Abstração de função $\neq$ função matemática
  - Abstração de função implementa uma função através de um algoritmo
  - Ela pode acessar e alterar valores de variáveis não-locais
- Exemplo (Pascal):

```
function quadrado (x: real): real
begin quadrado := x * x; end;
```

Valores...

D) Sequência

- Uma sequência consiste em um número arbitrário de ocorrências de itens de dados de um determinado tipo T
- Propriedade: deixa em aberto o número de ocorrências de um componente
- Exemplos: strings (que também podem ser implementados como arranjos de caracteres) e arquivos sequenciais
- Não há um consenso com relação à classificação de strings: é um valor primitivo ou composto?
- Operações usuais: concatenação, seleção do primeiro elemento, fatia (substring), ordenação lexicográfica, etc

Valores...

E) Conjunto potência

- É o tipo de variáveis cujo valor pode ser qualquer subconjunto de um conjunto de elementos de um determinado tipo T, o qual é chamado tipo base
- Representam conjuntos $\Rightarrow$ operações típicas de conjuntos
- $\#(\wp T) = 2^{\#T}$
- Exemplo (Pascal):

```
type ingrediente = (feijao, arroz, alface,
                                cenoura, couve, cebola);
var salada, sobras: set of ingrediente;
      sobras := [alface];
      salada := [cenoura..cebola];
      if not feijao in sobras
        then salada := salada + sobras;
```

Valores...

F) Recursão

- Um tipo de dados recursivo T pode ter componentes que pertençam ao próprio tipo T
- Permite definir agregados cujo tamanho pode crescer arbitrariamente e cuja estrutura pode ter complexidade arbitrária $\Rightarrow$ pode ser implementado por ponteiros
- Algumas LPs permitem a definição de tipos recursivos diretamente
- A cardinalidade de um tipo recursivo é infinita
- Exemplo (Pascal):

```
type ref_arvore = ^nodo_arvore;
     nodo_arvore = record
                     info: char;
                     esq, dir: ref_arvore:
                   end;
```

Valores...

- Sistema de tipos
  - Tipagem estática e dinâmica
    - Para evitar operações sem sentido, uma implementação de LP deve realizar uma verificação de tipos sobre os operandos
    - Quando realizar essa verificação?

$$\Downarrow$$

    - Linguagem tipada estaticamente: toda variável e parâmetro possui um tipo fixo determinado pelo programador $\Rightarrow$ verificação em tempo de compilação

Valores...

- Linguagem tipada dinamicamente: somente os valores possuem um tipo fixo, ou seja, variáveis e parâmetros podem assumir valores de tipos diferentes, durante a execução $\Rightarrow$ verificação em tempo de execução

- Tipagem dinâmica X estática
  - A tipagem dinâmica torna mais lenta a execução de um programa
  - A tipagem estática é mais segura
  - A tipagem dinâmica é mais flexível. Exemplo:

Valores...

- Exemplo:

```
procedimento leliteral (var: item);
  inicio
    ler um string;
    se o string representa um literal inteiro
    entao item := valor numerico do string
    senao item := proprio string
  fim
```

Obs: o código acima permitiria, por exemplo, a leitura de um mês como 2 ou FEV.

Valores...

- Equivalência de Tipos
  - **Equivalência estrutural**: duas variáveis têm tipos compatíveis se possuem a mesma estrutura
  - **Equivalência de nomes**: duas variáveis têm tipos compatíveis se possuem o mesmo nome de tipo, definido pelo usuário ou primitivo, ou se aparecem na mesma declaração

  - Equivalência estrutural X equivalência de nomes
    - Equivalência de nomes se aproxima mais ao conceito de tipos abstratos de dados
    - Equivalência de nomes é mais fácil de implementar

Valores...

- Exemplo:

```
type t = array [1..20] of integer;
var a, b: array [1..20] of integer;
      c    : array [1..20] of integer;
      d    : t;
      e, f : record a: integer; b: t; end;
```

Pela equivalência estrutural, a, b, c, d, e.b e f.b têm tipos compatíveis.

Pela equivalência de nomes, a e b, d, e.b e f.b têm tipos compatíveis, mas a e c não!

Valores...

- Princípio da Completeza de Tipo
  - “Nenhuma operação deve ser arbitrariamente restringida sobre os tipos de valores envolvidos”
  - Justificativa: restrições tendem a reduzir o poder de expressão de uma LP
  - Utiliza-se os termos valores de primeira classe e de segunda classe para diferenciar valores que podem ser utilizados de todas as formas possíveis (avaliados, atribuídos, passados como argumentos, etc), de valores que sofrem alguma restrição

Valores...

- Expressões
  - São frases de programa que podem ser avaliadas a fim de fornecer um valor
  - A) Literal
    - Forma mais simples de expressão
    - Representa um valor fixo
    - Ex. em Pascal: 1234, 1.5, ‘b’
  - B) Agregado
    - Expressão que constrói um valor composto a partir dos valores de seus componentes
    - Ex. 1 (ML): (a*2.0,b/2.0)
    - Ex. 2 (Ada): anonovo := (y => ano + 1, m => jan, d => 1);
    - Pascal oferece agregados apenas para conjuntos, mas para arranjos e registros a linguagem exige que se faça atribuição a cada um de seus componentes

Valores...

- C) Chamada de função
  - Calcula um resultado através da aplicação de uma abstração de função a um argumento
  - Ex. 1 (ML): (if cond then sin else cos) (x)
  - Ex. 2 (C): f (x)
  - Um operador também pode ser visto como uma função
  - Várias LPs reconhecem essa semelhança entre operadores e funções e, portanto, permitem a definição de operadores da mesma forma utilizada para se definir funções

Valores...

- D) Expressão condicional
  - Possui várias subexpressões, dentre as quais somente uma é escolhida para ser avaliada
  - Nem toda LP oferece esse recurso
  - Ex. (C): (a>b) ? a : c
  - Ex. (ML): if a > b then a else b
- E) Acesso a variáveis e constantes
  - Produz os valores das variáveis e/ou constantes denotados pelos respectivos identificadores