Paradigmas de Linguagem de Programação – Tipos de Dados

Profª Maria Adelina Raupp Sganzerla

Ulbra – Gravataí – 2016/2

Conceituação

- **Valor**

3 2.5 ‘a’ “Maria” 0X1F 026

- **Tipo**

{25, ‘b’, “vermelho”} não corresponde a um tipo

{true, false} corresponde a um tipo

Introdução

- Em LP, os tipos de dados são de fundamental importância;

- Processar dados consiste basicamente:
  - Determinar quais são os possíveis dados (entradas);
  - Determinar quais são os resultados (saídas);
  - Determinar quais são as transformações (ou operação) para transformar entradas em saídas.

Tipos de Dados

- Função de organizar dados em um programa;

- Características importantes:
  - Determina a classe de valores que podem ser armazenados em uma variável;
  - Informação do tipo é usada para prevenir ou detectar construções incorretas;
  - Determinar métodos de representação;
  - Manipulação de dados no computador.

Tipos de Dados

- A declaração explicita dos objetos possui vantagens sob o aspecto de expressividade, legibilidade e confiabilidade, como:
  - Conhecimento dos possíveis valores;
  - Saber quais operações são suportadas pelos tipos;
  - Tradutor disponibilizar espaço para os dados e operações;
  - Manipular exceções.

Tipos de Dados x Domínios

- Tipo de Dados:
  - Pode envolver diversos domínios
  - Por exemplo: domínio do tipo **inteiro**, consiste em valores “inteiros”, ou seja, valores numéricos sem a parte decimal.
  - O valor 13 pode perterncer ao domínio dos “naturais” ou “inteiros”
  - Em C pode ser:

```
int valor; //valores inteiros
short int valor; //possui metade da capacidade de
                 armazenamento
unsigned int valor; //valores inteiros em sinal
long int valor; //valor inteiro “longo”
```

Tipos de Dados x Domínios

- Um tipo de dados é associado a um conjunto de operações para manipular seus valores enquanto que um domínio é apenas um conjunto de valores.

Tipos de Dados

- São classificados como:
  - Primitivos: não necessitam de definição explícita. Exemplo: domínio dos números inteiros;
  - Definidos: seus componentes devem ser especificados, por enumeração (cria novo domínio) ou restrição (específica um subdomínio). Exemplos:
    - Enumeração: domínio estação = (primavera, verão, outono, inverno)
    - Restrição: domínio mandato: 2000..2005

Tipos Primitivos

- Costumam ser definidos na implementação da LP;

- Pode-se ter:
  - Inteiro;
  - Caractere (um ou uma sequência de caracteres);
  - Booleano (lógico);
  - Ponto Flutuante (real).

Tipo Inteiro

- Corresponde a um intervalo do conjunto dos números inteiros;

- Vários tipos inteiros em uma mesma LP:
  - Normalmente, intervalos são definidos na implementação do compilador.

Tipo Caractere/String

- Armazenados como códigos numéricos:
  - EBCDIC
  - ASCII
  - UNICODE

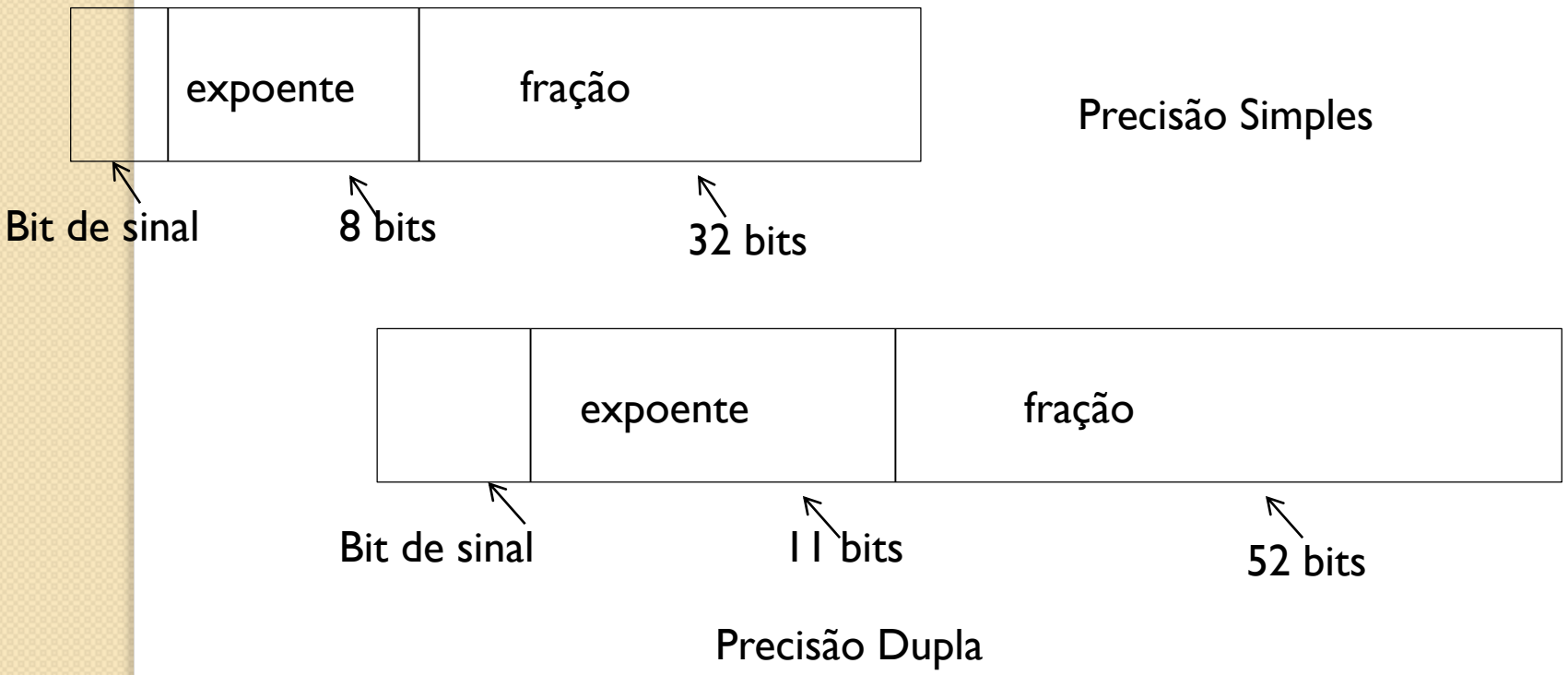
Tipo Booleano

- Tipo mais simples, oferece apenas dois valores: Verdadeiro ou Falso;

- C não possui o tipo de dado Booleano, mas qualquer expressão numérica pode ser usada como condicional:
  - Valores != zero -> Verdadeiro
  - Valores == zero -> Falso

Tipo Ponto Flutuante

- Compreende os números reais;
- LPs normalmente incluem dois tipos de ponto flutuante: float e double



Domínios Estruturados

- Produto Cartesiano;
- Mapeamento Finito;
- Sequência;
- União;
- Conjunto Potência.

Produto Cartesiano

- O produto cartesiano de *n* domínios A_1, A_2,…, A_n, denotado por A_1 x A_2 x…. x A_n, fornece conjuntos de tuplas ordenadas (a_1, a_2,…, a_n), onde cada a_k, pertence a A_k.

- **Exemplo**: Sendo A: inteiro e B: real, o produto cartesiano A x B fornece uma dupla (*a,b*) sendo que *a* pertence ao domínio A e *b* pertence ao domínio B, mais propriamente representável pela dupla (inteiro, real).

Produto Cartesiano

- Na representação deste tipo em linguagens de programação, o produto cartesiano corresponde a uma estrutura de registro;
- Exemplo de associação de nomes de campos a uma dupla (inteiro, real):

pessoa (idade: inteiro, peso: real)

nota_fiscal (quant: inteiro, valor: real)

referencia: pessoa.idade, pessoa.peso

```
struct Pessoa{
   int idade;
   float peso;
   };
```

Mapeamento Finito

- Seja *m* uma função de mapeamento e sejam A e B domínios. Aplicando a função *m* em um elemento *a* do domínio A obtém-se o valor *b* do contra-domínio B correspondente, também representado por *b = m(a)*.

- **Exemplo**: Seja A={1, 2} e B={V, F}. A partir de um valor do domínio A, aplica-se o mapeamento e obtém-se um valor do domínio B. Os possíveis valores resultantes do mapeamento são:
- V= m(1) V= m(2) ou
- V= m(1) F= m(2) ou
- F= m(1) V= m(2) ou
- F= m(1) F= m(2)

Mapeamento Finito

- Na representação deste tipo em linguagens de programação, o mapeamento finito corresponde à estrutura de array (matrizes unidimensionais - vetor), um tipo estruturado homogêneo quanto a tipo;

Sequência

- A sequência define um domínio cujos objetos são sequências de tamanho indeterminado;
- Uma sequência consiste de ocorrências de elementos em ordem arbitrária, permitindo repetições.

- **Exemplo**: Verbo = seq <'s', 'e' , 'r' , 'i', 'a'>
- A Linguagem C apresenta como uma string (sequência de caracteres)
- Arquivos também são sequências

União

- Constrói novos domínios a partir da união de outros domínios, fornecendo alternativas.
- **Exemplo**:

  dia_util = (segunda, terça, quarta, quinta, sexta)
  fim_de_semana = (sabado, domingo)
  dias_da_semana= união (trabalho: dia_util | descanso:fim_de_semana).

- Este modelo de construção corresponde ao tipo *union* em C e *variant record* em Pascal.

Conjunto Potência

- Sendo S um domínio, o conjunto de todos os subconjuntos dos valores de S é denominado de conjunto potência de S.

- **Exemplo**: Sendo S = {chá, café}, o conjunto de todos os subconjuntos de S seria:
  - { }
  - {chá}
  - {café}
  - {chá, café}

Conjunto Potência

- Observar que, ao contrário do método de sequência, a ordem dos elementos não define novos subconjuntos: o subconjunto. {chá, café} equivale a {café, chá}

- Poucas linguagens de programação oferecem mecanismos para implementar este tipo de modelo de construção. Pascal oferece a representação através do tipo *set*.

Representação de Tipos Estruturados

- As LP's oferecem as mais variadas estruturas de dados, tais como:
  - Vetores
  - Matrizes
  - Registros
  - Pilhas
  - Listas lineares
  - Strings, que raramente, poderão ser diretamente representadas por hardware convencional;

- Sua representação, portanto, geralmente será feita com auxílio de descritores (vetores, registros,...)

Vetores

- São arrays de tamanho fixo;
- São representados diretamente pelo hardware, através de registradores de idexação.

- Exemplo em C:

```
int vet[10]; //vetor de 10 elementos inteiros
```

Exemplo em Pascal:

```
vet = array[1..10] of integer; //vetor de 10 elementos inteiros
```

Vetores Dinâmicos

- Podem ser implementados em Pascal, C, C++ e Java;
- É necessário alocar nova memória e copiar conteúdo quando o vetor aumenta de tamanho;
- É encargo do programador controlar alocação e cópia. Em C e C++, o programador deve controlar a desalocação (liberação de memória) também. Isso torna a programação mais complexa e suscetível a erros.

Vetores/Matrizes Multidimensionais

- Também são conhecidos como matrizes;

- Em linguagens que não possuem o conceito de matrizes, como JAVA, vetores multidimensionais são obtidos com o uso de vetores unidimensionais cujos elementos são outros vetores

```
int [ ] [ ] a = new int [5] [ ];
```

- O mesmo efeito pode ser obtido em C com o uso de ponteiros para ponteiros

Tipos Ponteiros

- Ponteiro é um conceito de baixo nível relacionado com a arquitetura dos computadores;
- O conjunto de valores de um tipo ponteiro são os endereços de memória e o valor Null;
- Muito utilizados nas estruturas de Listas, Pilhas, Filas Encaeadas e Grafos.

Registro

- Dado que os elementos são heterogêneos, torna-se necessária a utilização de descritores para cada um dos campos do registro;

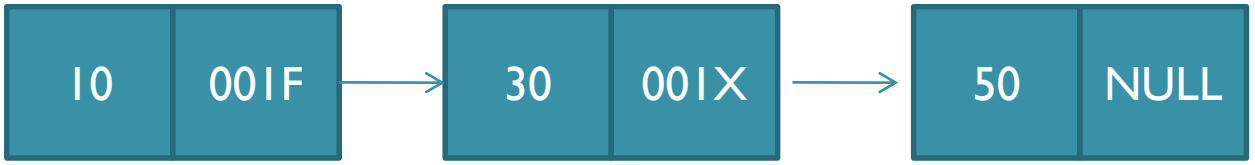
- Caso o campo do registro seja, por sua vez, um tipo estruturado, será utilizado um descritor correspondente ao tipo estruturado.

- Exemplo em C:

```
strcut cadastro{
    char nome[30];
    int idade;
    int sexo;
    };
```

Listas

- A representação de listas pode ser feita de diferentes maneiras;
- Lista é uma estrutura de dados dinâmica, mas que seus elementos são homogêneos.



Definição de Tipos de Dados

- A possibilidade de definir tipos de dados em Linguagens de Programação (*typedef* em C ou *type* em Pascal), não implica a criação de novos tipos e sim a possibilidade de associar um nome (sinônimo) a um tipo simples ou estruturado;
- Dois mecanismos devem ser oferecidos pela linguagem de programação: a definição do tipo e a instanciação da variável correspondente.
- Os principais objetivos deste mecanismo são o aumento de expressividade e da legibilidade e a redução do esforço de programação.

Hierarquia de Tipos