

Linguagens de Programação

Tipos

Carlos Bazilio

bazilio@ic.uff.br

<http://www.ic.uff.br/~bazilio/cursos/lp>

Definições

- Tipos de dados primitivos: tipos de dados que **não são definidos** em termos de outros tipos
- Exemplos:
 - Tipo Numérico (Inteiro, Ponto Flutuante, Decimal, ..)
 - Tipo Booleano (valores verdade)
 - Tipo Caracter

Cadeia de Caracteres ou String

- O valor usualmente é uma sequência de caracteres
- Apesar de usualmente não ser um tipo primitivo, é considerado um tipo básico no projeto de linguagens
- Operações comuns:
 - Extração de parte da cadeia, Concatenação, Cálculo do tamanho, Busca e/ou substituição de parte da cadeia, Comparação, etc.

Cadeia de Caracteres ou String

- Uma outra operação comum em cadeias é a busca por padrões
- A linguagem SNOBOL foi precursora nesse tipo de operação
- Uma outra linguagem importante nesse aspecto é a linguagem Perl
- Nela foram incorporados recursos, os quais são comumente chamados expressões regulares
 - Exemplo: `/[A-Za-z][A-Za-z\d]+/`
- Acima define-se regras para identificadores

Cadeia de Caracteres ou String

- Quanto ao tamanho das cadeias, definido no projeto da linguagem, podemos ter:
 - Tamanho estático: o tamanho da cadeia não se modifica ao longo da execução do programa (Fortran 90, COBOL, Pascal, Java e Ada, ..)
 - Tamanho dinâmico limitado: tamanho variável até um máximo especificado (C, C++, ..)
 - Tamanho dinâmico: aumento/diminuição das cadeias ilimitado (SNOBOL, JavaScript, Perl, ..)

Cadeia de Caracteres ou String

- Quanto à implementação, tanto a manipulação de cadeias de tamanho estático, quanto as de tamanho dinâmico limitado, não requerem cuidados especiais
- A manipulação de cadeias de tamanho dinâmico exige cuidado especial
- 2 abordagens comuns de manipulação:
 - Usando listas encadeadas
 - Usando células de armazenagem adjacentes
- Vantagens e desvantagens dessas técnicas?

Tipos Ordinais

Enumeração

- Define faixa de valores, a qual usualmente é associada a um conjunto de inteiros
 - Ex (ADA): type DIAS is {Seg, Ter, Qua, Qui, Sex};
- Uma questão de projeto é quanto a ocorrência de um valor em mais de uma enumeração (literal sobrecarregado)
 - type LETRAS is {'A', 'B', 'C', 'Z'}
 - type VOGAIS is {'A', 'E', 'I', 'O', 'U'}
- As linguagens normalmente proíbem estas ocorrências, ou forçam uma diferenciação explícita

Tipos Ordinais

Sub-faixa

- Um tipo similar à enumeração, normalmente uma sub-sequência
 - Ex (Pascal): **type** maiuscula = 'A' .. 'Z'; indice = 1..100;
- No caso de Ada, esses tipos são sub-tipos de tipos já existentes
 - Ex: **subtype** FINAL is DIAS range Qua .. Sex;
- Neste caso, as operações definidas para o tipo pai (DIAS, no exemplo) também valem para o filho (FINAL); ou seja, o filho herda as operações do pai

Coleções

- Permite a manipulação de conjunto de valores
- A linguagem pode permitir o acesso por índices ou de forma iterativa, como numa lista encadeada
- Coleções com acesso por índice:
 - Vetores, Matrizes, etc
- Coleções acessadas de forma iterativa:
 - Listas, Pilhas, Filas, Mapas, Conjuntos, etc
- Há linguagens que oferecem tipos mistos (Java, por exemplo)

Coleções

- Algumas possibilidades de implementação:
 - Estática: alocada antes da execução e com tamanho bem definido (Fortran 77)
 - Fixa e Dinâmica na Pilha: alocação feita durante a execução, mas também com um tamanho já definido (Pascal e C)
 - Dinâmica na Pilha: mesmo que o anterior, mas com o tamanho sendo definido durante a execução (Fortran 90)
 - Dinâmica no Heap: totalmente flexível, podendo ser alocada a qualquer momento e tendo seu tamanho modificado ao longo da execução (C, C++, ...)

Coleções Indexadas

Características

- Usualmente, o número de índices suportados por uma linguagem é indeterminado
- Operações comuns:
 - Inicialização múltipla
 - Ex (C): `char *nomes[] = {"fulano", "ciclano", "beltrano"};`
 - Ex (ADA-like): `nomes : array (1..3) of STRING := (1 => "fulano", 2 => "ciclano", others => "");`
 - Atribuição, Comparação, Concatenação, Operações Aritméticas (soma, média, ..), Mín/Max, Ordenação ..

Coleções Indexadas

Características

- Matrizes e suas operações são a parte principal da linguagem APL
- Nela, a operação $A + B$ funciona, sejam A e B escalares, vetores ou matrizes
- Além disso, a linguagem possui alguns operadores comuns para esse tipo de coleção:
 - Inversão de elementos, linhas, colunas, transposição, etc
 - Outros especiais: por exemplo, $A +.x B$ faz a multiplicação das matrizes A por B

Coleções Indexadas

Características

- Quanto a implementação, o esforço computacional para uso destas estruturas reside na manipulação de memória
 - Por exemplo, em C, o acesso $vet[indice]$ representa a expressão $*(vet + indice)$, onde o $*$ é o operador que dereferencia um endereço de memória
- Esta complexidade aumenta com o aumento no número de índices, por exemplo, na manipulação de matrizes

Matrizes Associativas

- Diferentemente das matrizes comuns, essas matrizes associativas (mapas, dicionários, tabelas ...) utilizam índices diferentes de números (chaves)
- A cada chave utilizada, que pode ser uma string por exemplo, é associado um valor
- Por exemplo, em Perl:
 - `%salarios = ("fulano" => 10000, "ciclano" => 5000);`
 - `$salarios{"beltrano"} = 8000;`
 - `delete $salarios{"ciclano"};`
- Essas estruturas são bastante dinâmicas e elegantes

Tipo Registro

- É uma composição de tipos primitivos e/ou derivados, denominados campos do registro
- Por exemplo, em COBOL temos:

```
01 REGISTRO-EMPREGADO.  
  02 NOME-EMPREGADO.  
    05 PRIMEIRO          PICTURE IS X(20).  
    05 MEIO              PICTURE IS X(10).  
    05 ULTIMO           PICTURE IS X(20).  
  02 TAXA-HORARIA      PICTURE IS 99V99.
```

- As linguagens também possuem formas distintas de referência a campos num registro
 - No exemplo, MEIO of NOME-EMPREGADO of REGISTRO-EMPREGADO

Tipo Registro

- Pascal, assim como outras linguagens, procura simplificar o acesso a registros:

```
empregado.nome := 'fulano';  
empregado.idade := 55;  
empregado.sexo := 'M';
```

```
with empregado do  
  begin  
    nome := 'fulano';  
    idade := 55;  
    sexo := 'M';  
  end;
```

- Além das operações sobre cada campo do registro (de acordo com o seu tipo), a atribuição entre registros (cópia campo a campo) é comumente suportada

Tipo Registro

- Na implementação o acesso a cada campo é feito da seguinte forma:
 - Obtem-se o endereço inicial do registro (base)
 - Soma-se o tamanho dos campos localizados anteriormente ao campo desejado (deslocamento)

Tipo União ou Registro com Variante

- Suponha que queremos criar um único tipo registro para manipular diferentes informações
- Estas informações possuem campos em comum e campos exclusivos
- O tipo união possui essa flexibilidade

```
type
  forma = (circulo, triangulo, retangulo);
  cores = (vermelho, verde, azul);
  figura = record
    preenchido: boolean;
    cor: cores;
    case aspecto: forma of
      circulo: (diametro: real);
      triangulo: (le, ld: integer; angulo: real);
      retangulo: (l1, l2: integer);
    end;
var minhafigura: figura;
```

Tipo União

- O exemplo em Pascal fornecido anteriormente pode ser utilizado da seguinte forma:

```
var minhafigura: figura;  
{Inicialização dos campos comuns da variável minhafigura}  
minhafigura.aspecto := triangulo;  
case minhafigura.aspecto of  
    circulo: writeln('É um circulo com diametro: ', minhafigura.diametro);  
    triangulo: ...;  
    retangulo: ...;  
end;
```

- Ou seja, precisamos definir qual o tipo de figura manipulada
- Observe que erros podem ocorrer caso se defina um figura de um aspecto e utilize campos de outro

Tipo União

- Em termos de implementação, é alocado para o tipo a área de memória correspondente à maior variante
- Isto propicia um uso mais eficiente da memória
- Em C, esta funcionalidade é obtida através do recurso denominado *union*
- Em linguagens OO (C++, Java, etc), estes recursos são elegantemente substituídos

Tipo Conjunto

- Tipo que armazena uma coleção de valores distintos e não-ordenados
- As operações comuns deste tipo são as operações comuns de conjuntos na matemática:
 - União, Interseção, Igualdade, Pertinência, ...
- Apesar de não haver ordem, algumas linguagens (Java, por exemplo) oferecem recursos para varrer um conjunto (chamados iteradores)

Ponteiros e Tipos Recursivos

Conversão de Tipos

- Suponha a atribuição:

```
int a = 10;
```

```
float b = 7.5;
```

```
a = b;
```

- Nesta atribuição teremos perda de valor, uma vez que o tipo *int* só armazena inteiros
- Algumas linguagens/compiladores aceitam naturalmente esta atribuição, outros emitirão um alerta, e também existem os que proibirão

Conversão de Tipos

- Na situação em que há um alerta, este pode ser eliminado através das operações de conversão explícita (*casting*)

```
int a = 10;
```

```
float b = 7.5;
```

```
a = (int) b;
```

- Desta forma, o programador indica para o compilador que está ciente da possível perda de valor

Classificação de Linguagens com respeito a Tipos

- As linguagens podem ser classificadas em:
 - Tipagem Forte
 - Tipagem Fraca
 - Atipadas
- Quanto ao momento em que as verificações são feitas, podemos ter:
 - Tipagem Estática
 - Tipagem Dinâmica