PARADIGMAS E LINGUAGENS DE PROGRAMAÇÃO

ENGENHARIA DA COMPUTAÇÃO – UFC/SOBRAL

Prof. Danilo Alves

danilo.alves@alu.ufc.br

TIPOS DE DADOS COMPOSTOS

UNIVERSIDADE FEDERAL DO CEARÁ

- O que são tipos de dados compostos?
- Aqueles que podem ser criados a partir de tipos mais simples
  - Matrizes
  - Registros
  - Uniões

TIPOS MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma matriz é um agregado **homogêneo** de elementos de dados no qual um elemento individual é identificado por sua **posição** na agregação, **relativamente ao primeiro elemento**
- Homogêneo: Mesmo tipo
- Posição: Através de [índices]

QUESTÕES DE PROJETO DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Que tipos são permitidos para índices?
- As expressões de índices em referências a elementos são verificadas?
- As matrizes multidimensionais irregulares ou retangulares são permitidas?

MATRIZES E ÍNDICES

UNIVERSIDADE FEDERAL DO CEARÁ

- Indexar (ou subscrever) é um mapeamento de índices para elementos
  - nome_matriz (lista_valores_índices) → elemento

- Sintaxe de índices
  - FORTRAN, PL/I e Ada usam parênteses
    - Ada explicitamente usa parênteses para mostrar a uniformidade entre as referências matriz e chamadas de função, pois ambos são mapeamentos
  - A maioria das outras linguagens usa colchetes

TIPOS DE ÍNDICES DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- FORTRAN, C: apenas inteiros
- Ada: inteiro ou enumeração
- Java: apenas tipos inteiros
- Verificação de faixas de índices
  - C, C++, Perl e Fortran não especificam faixas de índices
  - Java, ML e C# especificam faixas de índices
  - Em Ada, o padrão é exigir a verificação de faixas de índice, mas pode ser desligada

TIPOS DE ÍNDICES DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

C:
```
int mat[5][4];
float a[10];
```

Pascal :
```
var a: array[1 .. 10] of real;
    b: array[1 .. 10, 1..10] of real;
```

Fortran 90:
```
integer vetor1(1:10), vetor2(1:10)
                  ...
vetor1 = vetor1 + vetor2
```

VINCULAÇÕES DE ÍNDICES E CATEGORIAS DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- **Estática**: é uma na qual as faixas de índices são vinculadas estaticamente e a alocação de armazenamento é estática (antes do tempo de execução)
  - **Vantagem**: eficiência (sem alocação dinâmica)

- **Dinâmica da pilha fixa**: é uma na qual as faixas de índices são vinculadas estaticamente, mas a alocação é feita em tempo de elaboração da declaração
  - **Vantagem**: eficiência de espaço

- **Dinâmica do monte fixa** é uma na qual tanto as faixas de índices quanto a alocação de armazenamento são vinculadas dinamicamente em tempo de elaboração
  - **Vantagem**: flexibilidade (o tamanho da matriz não precisa ser conhecido até ela ser usada)

VINCULAÇÕES DE ÍNDICES E CATEGORIAS DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- **Dinâmica do monte**: é uma na qual a vinculação das faixas de índices e da alocação de armazenamento é dinâmica e pode mudar qualquer número de vezes durante seu tempo de vida
  - **Vantagem**: flexibilidade (matrizes podem crescer e encolher durante a execução de um programa)

VINCULAÇÕES DE ÍNDICES E CATEGORIAS DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Matrizes C e C++ que incluem o modificador static são estáticas
- Matrizes C e C++ sem o modificador static são dinâmicas da pilha fixas
- C e C++ também fornecem matrizes dinâmicas do monte fixa
  - Malloc e free ou new e delete
- C# inclui matrizes dinâmicas da pilha
  - List<String> stringlist = new List<String>();
- Perl, JavaScript, Python e Ruby suportam matrizes dinâmicas do monte
  - Push e unshift
  - Em javascript as matrizes podem ser esparsas

INICIALIZAÇÃO DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Algumas linguagens fornecem os meios para inicializar matrizes no momento em que seu armazenamento é alocado

– Exemplo em C, C++, Java, C#

```
int list [] = {4, 5, 7, 83}
```

– Cadeias de caracteres em C e C++

```
char name [] = "freddie";
```

– Matrizes de cadeias em C e C++

```
char *names [] = {"Bob", "Jake", "Joe"];
```

– Inicialização de objetos String em Java

```
String[] names = {"Bob", "Jake", "Joe"};
```

MATRIZES HETEROGÊNEAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma matriz heterogênea é uma em que os elementos não precisam ser do mesmo tipo

- Suportadas por Perl, Python, JavaScript e Ruby

INICIALIZAÇÃO DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Linguagens baseadas em C
  - int list [] = {1, 3, 5, 7}

    char *names [] = {“Mike”,“Fred”,“Mary Lou”};
- Python
  - Inicialização de lista

    lista = [1, 2, 3]

INICIALIZAÇÃO DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Linguagens baseadas em C

  – int list [] = {1, 3, 5, 7}

  char *names [] = {“Mike”,“Fred”,“Mary Lou”};

- Python possui um recurso poderoso chamado: Compreensões de lista

  lista = [x ** 2 for x in range(12) if x % 3 == 0]

  Coloca [0, 9, 36, 81] em lista

OPERAÇÕES DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- APL é a linguagem de processamento de matrizes mais poderosa já desenvolvida.
- As quatro operações aritméticas básicas são definidas para vetores (matrizes de dimensão única) e para matrizes, bem como operadores escalares
- Ada permite atribuição de matrizes, mas também concatenação
- **Python fornece atribuição de matrizes**
- **Python também suporta operações para concatenação de matrizes e para verificar se um elemento pertence à matriz**
- Ruby também fornece concatenação de matrizes
- Fortran inclui operações elementais porque elas ocorrem entre pares de elementos de matrizes

MATRIZES RETANGULARES E IRREGULARES

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma matriz retangular é uma multidimensional na qual todas as linhas e colunas têm o mesmo número de elementos

- Na matriz irregular, o tamanho das linhas não precisa ser o mesmo
  - Possíveis quando as multidimensionais são matrizes de matrizes
- C, C++ e Java suportam matrizes irregulares
- Fortran, Ada e C# suportam matrizes retangulares (C# também suporta irregulares)

MATRIZ COM PYTHON

- Implementado como lista de listas

```
# Matriz identidade
mat = [
    [1, 0, 0],
    [0, 1, 0],
    [0, 0, 1]
]
```

- Possui biblioteca para manipulação de matrizes – numpy

EXEMPLO DE FATIAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Python

```
A = range(10) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
A[5:]         # [5, 6, 7, 8, 9]
A[:5]         # [0, 1, 2, 3, 4]
A[2:7]        # [2, 3, 4, 5, 6]
A[2:7:2]      # [2, 4, 6]
A[-2:]        # [8, 9]
A[-2:4:-1]    # [8, 7, 6, 5]
mat[1][1:3]   # [1, 0]
```

- Ruby suporta fatias com o método slice

  list.slice(2, 2) retorna o terceiro e o quarto elementos de list

IMPLEMENTAÇÃO DE MATRIZES

UNIVERSIDADE FEDERAL DO CEARÁ

- Função de acesso mapeia expressões subscritas para um endereço na matriz

- Função de acesso para list:

  endereço(list[k]) = endereço (list[0])

  + k * tamanho_do_elemento)

  tamanho_do_elemento = 4 para inteiros, em C em algumas implementações

MATRIZES ASSOCIATIVAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma matriz associativa é uma coleção não ordenada de elementos de dados indexados por um número igual de valores chamados de chaves
  - Chaves definidas pelo usuário devem ser armazenadas

- Questões de projeto:
  - Qual é o formato das referências aos seus elementos?
  - O tamanho é estático ou dinâmico?

- Suportadas diretamente em Perl, Python, Ruby e Lua
  - Em Lua, suportadas por tabelas

MATRIZES ASSOCIATIVAS EM PYTHON

UNIVERSIDADE FEDERAL DO CEARÁ

```
eng2port = {}                   # cria uma matriz (dict) vazia
eng2port['one'] = 'um'          # define um item único
eng2port['two'] = 'dois'        # define outro item
eng2port['three'] = 'três'      # define um terceiro item

for key, value in eng2port.items():  # (key, value)
    print(key, ' => ', value)
```

```
one  =>  um
two  =>  dois
three  =>  três
```

MATRIZES ASSOCIATIVAS EM JAVASCRIPT

UNIVERSIDADE FEDERAL DO CEARÁ

```
var cars        = [];
cars['cor']     = 'vermelho';
cars['marca']   = 'Ferrari';
cars['potencia']= '700';
cars['blindado']= false;
cars['aro']     = 21;
```

- E Java?

REGISTROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Um registro é um agregado de elementos de dados no qual os elementos individuais são identificados por **nomes**

- Questões de projeto:
  - Qual é a forma sintática das referências a campos?

DEFINIÇÃO DE REGISTROS EM COBOL

UNIVERSIDADE FEDERAL DO CEARÁ

- COBOL usa números de nível para montar uma estrutura hierárquica de registros

```
01 EMP-REC.
   02 EMP-NAME.
      05 FIRST PIC X(20).
      05 MID   PIC X(10).
      05 LAST  PIC X(20).
   02 HOURLY-RATE PIC 99V99.
```

- Outras linguagens usam definições recursivas

DEFINIÇÃO DE REGISTROS EM ADA

UNIVERSIDADE FEDERAL DO CEARÁ

- Estruturas de registro em Ada

```
type Emp_Rec_Type is record
   First: String (1..20);
   Mid: String (1..10);
   Last: String (1..20);
   Hourly_Rate: Float;
end record;
Emp_Rec: Emp_Rec_Type;
```

DEFINIÇÃO DE REGISTROS PASCAL E C

UNIVERSIDADE FEDERAL DO CEARÁ

```
type
  reg_pessoa = record
    nome: string;
    idade: integer;
    cpf: string[11];
    end;
```

```
typedef struct {
    char nome[100];
    int idade;
    char cpf[12];
} s_pessoa;
```

REFERÊNCIAS A REGISTROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Referências a campos de registros

1) COBOL
   - nome_campo OF nome_registro_1 OF ... OF nome_registro_n

2) Outros (notação por pontos)
   - nome_registro_1.nome_registro_2. ... nome_registro_n.nome_campo

OPERAÇÕES EM REGISTROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Em C, é possível:
- Inicializar

```
typedef struct {
    char nome[100];
    int dia;
    int mes;
    int ano;
} Pessoa;


Pessoa p1 = {"Smith", 01, 10, 1976};
```

Atribuir

```
typedef struct {
    char nome[100];
    int dia;
    int mes;
    int ano;
} Pessoa;


Pessoa p1 = {"Smith", 01, 10, 1976};
Pessoa p2 = p1;
```

UNIÕES

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma união é um tipo cujas variáveis podem armazenar diferentes valores de tipos em diferentes momentos durante a execução de um programa

- Questões de projeto
  - A verificação de tipos deve ser obrigatória?
  - As uniões devem ser embutidas em registros?

UNIÕES DISCRIMINADAS X UNIÕES LIVRES

UNIVERSIDADE FEDERAL DO CEARÁ

- Fortran, C e C++ fornecem construções para representar uniões nas quais não existe um suporte da linguagem para a verificação de tipos; as uniões nessas linguagens são chamadas de uniões livres

- A verificação de tipos união requer que cada construção de união inclua um indicador de tipo, chamado de discriminante
  - Uniões discriminadas são suportadas por Ada

UNIÕES EM ADA

UNIVERSIDADE FEDERAL DO CEARÁ

```
type Shape is (Circle, Triangle, Rectangle);
type Colors is (Red, Green, Blue);
type Figure (Form: Shape) is record
    Filled: Boolean;
    Color: Colors;
    case Form is
        when Circle => Diameter: Float;
        when Triangle =>
            Leftside, Rightside: Integer;
            Angle: Float;
        when Rectangle => Side1, Side2: Integer;
    end case;
end record;
```

UNIÕES EM C

UNIVERSIDADE FEDERAL DO CEARÁ

```
union u_nomeUniao{
   tipoDadoMembro_1 nomeMembro_1;
   tipoDadoMembro_2 nomeMembro_2;
   ...
   tipoDadoMembro_n nomeMembro_n;
}; //end union


union u_type {
   int i;
   char ch;
};//end union


union u_type cnvt;
```

UNIÕES EM C

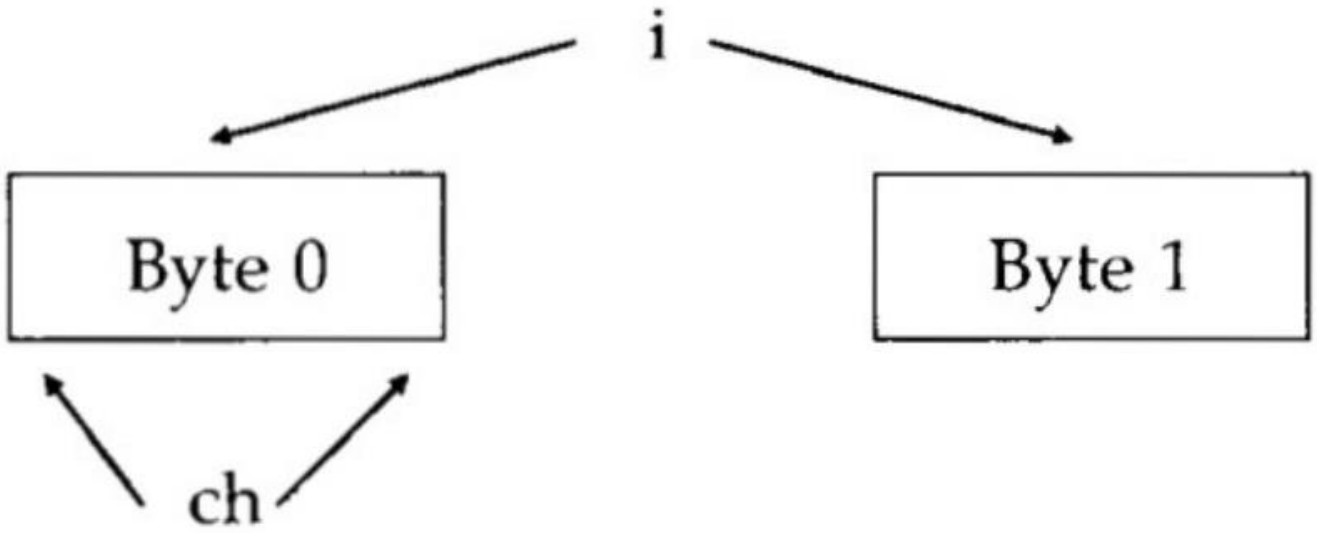
UNIVERSIDADE FEDERAL DO CEARÁ

```c
// estrutura usando "regular union"
struct {
  char title[50];
  char author[50];
  union {
    float dollars;
    int yens;
  } price;
} book;
```

```c
// estrutura usando "anonymous union"
struct {
  char title[50];
  char author[50];
  union {
    float dollars;
    int yens;
  };
} book;
```

UNIÕES EM C

- Na **union cnvt**, tanto inteiro **i** quanto o caractere **ch** compartilham a mesma posição de memória
  - **i** ocupa 2 (ou 4) bytes
  - **ch** ocupa 1 byte



UNIÕES EM C

UNIVERSIDADE FEDERAL DO CEARÁ

- Quando uma **union** é definida, o compilador cria aloca um espaço na memória grande o bastante para conter o maior tipo dentro da **union**
  - Para o exemplo, **cnvt** possui 2 bytes (ou 4)

- Para acessar um elemento da **union**, usa-se a mesma sintaxe das **structs**
  - Operador ponto (.) e operador seta (->)

AVALIAÇÃO DE UNIÕES

UNIVERSIDADE FEDERAL DO CEARÁ

- Uniões são construções potencialmente inseguras
  - Não permite verificação de tipos
- Java e C# não suportam uniões
  - Reflexo da crescente preocupação com a segurança em linguagens de programação
- Em Ada, podem ser usadas com segurança

PONTEIROS E REFERÊNCIAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Um tipo **ponteiro** é um tipo no qual as variáveis possuem uma faixa de valores que consistem em endereços de memória
- Fornecem alguns dos poderes do endereçamento indireto
- Fornecem uma maneira de gerenciar o armazenamento dinâmico
- Um ponteiro pode ser usado para acessar uma posição na área onde o armazenamento é dinamicamente alocado, o qual é chamado de monte (*heap*)

QUESTÕES DE PROJETO

UNIVERSIDADE FEDERAL DO CEARÁ

- Os ponteiros especificam os tipos de valores aos quais eles podem apontar?
- Por que algumas linguagens não tem ponteiro?

OPERAÇÕES DE PONTEIROS

- Duas operações de ponteiros fundamentais:

  – atribuição e desreferenciamento

- **Atribuição**: modifica o valor de uma variável de ponteiro para algum endereço útil.

- **Desreferenciamento**: leva uma referência por meio de um nível de indireção

– Pode ser explícito ou implícito

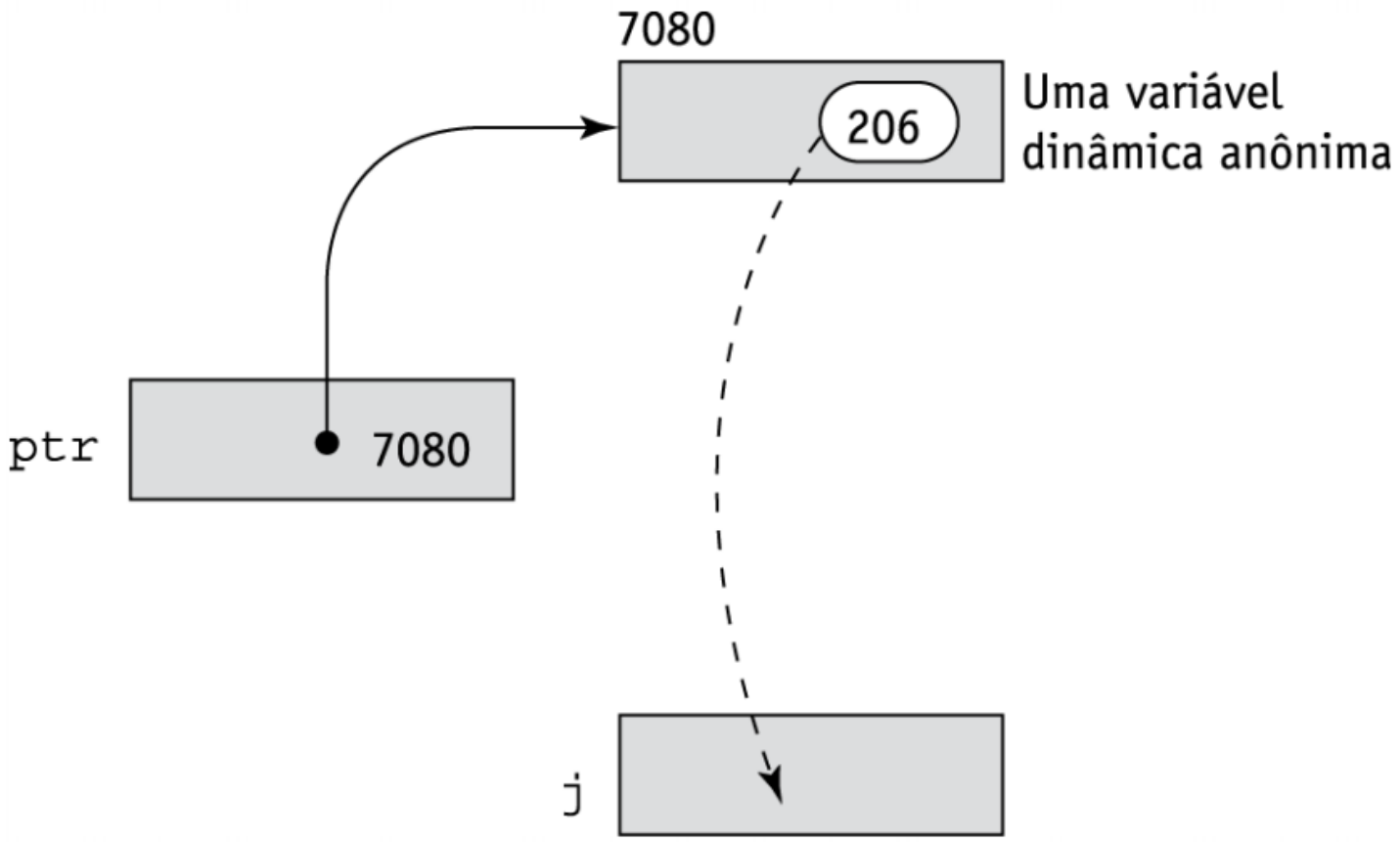
- Em C++, é explicitamente especificado com o asterisco (*)

  j = *ptr

  modifica j para o valor de ptr

ATRIBUIÇÃO DE PONTEIRO

- A operação de atribuição j = *ptr



PROBLEMAS COM PONTEIROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Ponteiros soltos (perigoso)

– É um ponteiro que contém o endereço de uma **variável dinâmica do monte que já foi liberada**

- Variáveis dinâmicas do monte perdidas

  – É uma variável dinâmica alocada do monte que não está mais acessível para os programas de usuário (geralmente chamadas de lixo)

  - O ponteiro p1 é configurado para apontar para uma variável dinâmica do monte recém criada
  - p1 posteriormente é configurado para apontar para outra variável dinâmica do monte recém criada
  - A primeira variável dinâmica do monte é agora inacessível, ou perdida. Isso às vezes é chamado de **vazamento de memória**

AVALIAÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Ponteiros soltos e lixo são problemas, tanto quanto o gerenciamento do monte

- Ponteiros e referências são necessários para estruturas de dados dinâmicas – não podemos projetar uma linguagem sem eles