Introdução à Ciência da Computação II

Revisão de Linguagem C – Pt. I: Arranjos, Registros e Funções

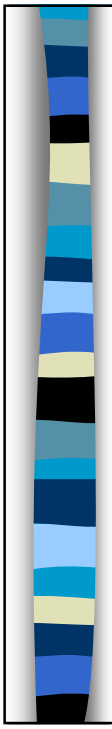
Prof. Ricardo J. G. B. Campello

Agradecimentos

- Parte dos slides a seguir são adaptações dos originais gentilmente cedidos por:
  - Prof. Rudinei Goularte
  - Prof. André C. P. L. F. Carvalho

Sumário

- Vetores, Matrizes e Strings
- Registros (structs)
- Funções



Vetor

- **Definição: variável composta** dada por uma coleção de elementos individuais com as seguintes características:
  - É **ordenado**: os elementos de um vetor são indexados de forma ordenada
  - É **homogêneo**: Todo valor armazenado em um mesmo vetor deve ser do mesmo tipo
    - Ex.: vetor de inteiros só pode ter elementos do tipo inteiro

4

Vetor

- Pode-se pensar em um vetor como uma seqüência de células, uma para cada elemento:

- Também denominado **arranjo** 1D
- Possui duas propriedades fundamentais:
  - Tipo de elemento
  - Tamanho do vetor

5

Vetor

- Declaração
  - **tipo** *identificador*[tamanho]
    - Ex.: **int** vet[10];
  - Tamanho do vetor pode ser especificado como uma constante
    - Facilita mudança do tamanho
    - Exemplo: ***#define*** *NElementos 10*
      int *vet[NElementos];*

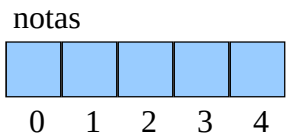
6

Vetor

- Declaração (cont.)
  - Cada elemento de um vetor é identificado por um índice
    - Em C começa com o valor 0 e termina com um valor igual ao número de elementos menos 1
    - Ex. vetor de 4 elementos possui os índices 0, 1, 2, 3

7

Vetor



- Para se referir a um elemento específico de um vetor, devem ser fornecidos:
  - nome do vetor e o índice correspondente
- Exemplo:

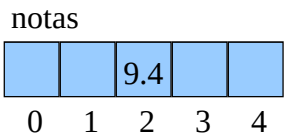
```
#define NJuizes 5
double notas[NJuizes];
```

  - A nota do 2o juiz é dada por notas[1]

8

Vetor

- Células são variáveis
  - simples ou compostas
    - Ex.: notas[2] = 9.4;
    - É importante distinguir entre índice de um elemento e valor de um elemento



Índice = 2
Valor = 9.4

9

Vetor

- Inicialização de vetores
  - Valores iniciais podem ser atribuídos a uma variável do tipo vetor quando da sua declaração
    - Ex.: ***int*** *digitos[10] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };*
    - Neste caso, o tamanho do vetor pode ser omitido
      - Ex.: ***int*** *digitos[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };*
      - Compilador conta o número de inicializadores e reserva a mesma quantidade de elementos
      - Programador não precisa contar (útil em strings)

10

Vetor

- Tamanho real do vetor em bytes:
  - ***sizeof****(tipo base) * tamanho do vetor*
  - Exemplo Anterior: **sizeof**(**int**)*10

11

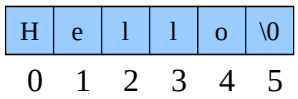
Vetor

- A Linguagem C não checa se você passou dos limites de um vetor !
  - Se passar do fim de um vetor, você pode:
    - Escrever no espaço reservado para outras variáveis...
    - Escrever no espaço reservado para outros programas...
    - ...

12

Strings

- *Strings* são vetores de caracteres
  - Cada célula = 1 caractere = 1 byte
  - Final de *string* em C é indicado por '\0'
  - Declarações com inicialização:
    - **char** hello[] = {'H', 'e', 'l', 'l', 'o', '\0'}; ou
    - **char** hello[6] = {'H', 'e', 'l', 'l', 'o', '\0'}; ou
    - **char** hello[] = "Hello";
    - **char** hello[6] = "Hello";



Strings

- Bibliotecas de operações sobre strings
  - Biblioteca ANSI *string.h* para manipular strings
    - Biblioteca padrão da linguagem C
    - Fornece um conjunto de operações avançadas
      - Permite trabalhar com uma ou mais strings inteiras utilizando uma simples chamada de função

Strings

Funções mais comuns de string.h

Nome	Função
strncpy (s1, s2, ...)	Copia s2 em s1
strncat (s1, s2, ...)	Concatena s2 ao final de s1
strlen (s1)	Retorna o tamanho de s1
strncmp (s1, s2, ...)	Retorna 0 se s1 ==s2; menor que 0 se s1<s2; maior que 0 se s1>s2
strstr (s1, s2)	Retorna um ponteiro para a primeira ocorrência de s2 em s1

15

Exemplo

```
#include <stdio.h>
#include <string.h>
void main(void){
  char nome1[12] = "Jose", nome2[] = "Maria";
  strncpy(nome1, nome2, 12);
  printf("%s\n", nome1);
}
```

16

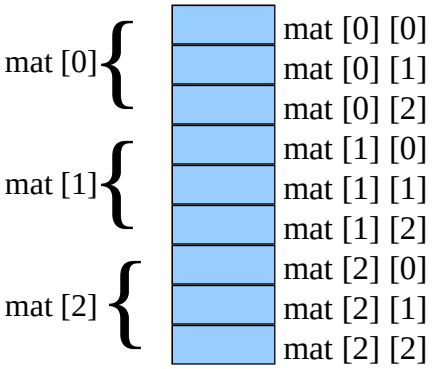
Arranjos Multi-Dimensionais

- Quando os elementos de um arranjo são arranjos
  - Arranjos bidimensionais (**matrizes**) são a forma mais comum
    - Vetor de vetores
  - Ex.: ***double*** *mat [3][3];*

mat[0][0]	mat[0][1]	mat[0][2]
mat[1][0]	mat[1][1]	mat[1][2]
mat[2][0]	mat[2][1]	mat[2][2]

17

Matrizes



C trata *mat* como um vetor de três elementos

Cada elemento é, por sua vez, um vetor de três elementos

Na memória, estes nove valores formam uma lista unidimensional

18

Exemplo

```
#include <stdio.h>
void main(void){
   char str[3][80];
   for(i=0; i<=2; i++){
      printf("Entre com 1 string (max. 79 carac.): ");
      scanf("%s", str[i]);
   }
}
```

19

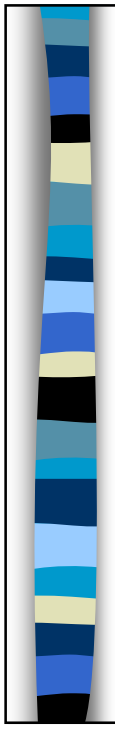
Matrizes

- Podem ser inicializadas na declaração
  - Para enfatizar a estrutura geral, valores de cada vetor interno são inicializados entre chaves

double *ident [3][3] = {*
{1.0, 0.0, 0.0 },
{0.0, 1.0, 0.0 },
{0.0, 0.0, 1.0 }
};

1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

20



Exercício

- Faça um programa que declare 3 matrizes 4 x 4 de inteiros, inicialize 2 delas arbitrariamente na declaração e use laços (loops) para calcular a 3a como a soma das duas anteriores

21

Registros

- **Definição**:
  - Um registro é uma **variável composta heterogênea**
    - É um conjunto de dados estruturados, os quais podem ser de tipos diferentes
  - Os dados em um registro são representados através de variáveis ou constantes, normalmente chamadas de campos

22

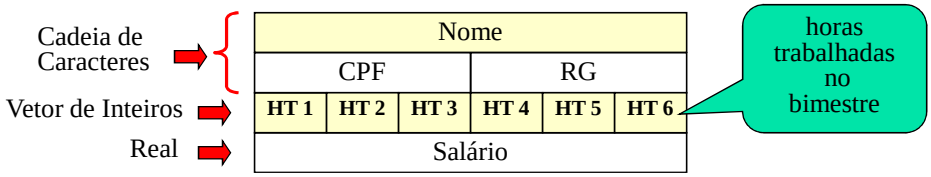
Registros

- Exemplo: Registro de Pagamento
  - Dados:
    - Nome, RG, CPF, salário, horas trabalhadas, etc...
  - Estrutura de Dados:
    - Uma variável (campo) para cada dado, mas...
    - Um identificador comum para o conjunto (registro)

23

Registros

- Exemplo: Registro de Pagamento



24

Declaração de Registros

Declaração de Variável Única:

```
struct {
        tipo_1 campo(s);
        tipo_2 campo(s);
                        ...
        tipo_n campo(s);
                }; variável;
```

25

Declaração de Registros

- **Exemplo:**

```
struct {
        char NOME [30];
        char CPF[12];
        char RG[10];
        int HT[6];
        float SALARIO;
                }; Reg_Pag1;
```

26

Declaração de Registros

Declaração de Múltiplas Variáveis (Forma 1):

```
struct tipo_registro {
        tipo_1 campo(s);
        tipo_2 campo(s);
            ...
        tipo_n campo(s);
        }; variável(is);
```

27

Declaração de Registros

- **Exemplo:**

```
struct Registro_Pagamento {
        char NOME [30];
        char CPF[12];
        char RG[10];
        int HT[6];
        float SALARIO;
    }; Reg_Pag1, Reg_Pag2, Reg_Pag3;
```

28

Declaração de Registros

Declaração de Múltiplas Variáveis (Forma 2):

```
struct  tipo_registro  {
                tipo_1 campo(s);
                tipo_2 campo(s);
                    ...
                tipo_n campo(s);
                    };

struct tipo_registro variáveis;
```

29

Declaração de Registros

- **Exemplo:**

```
struct Registro_Pagamento {
        char NOME [30];
        char CPF[12];
        char RG[10];
        int HT[6];
        float SALARIO;
        };
struct Registro_Pagamento  Reg_Pag1,  Reg_Pag2,  Reg_Pag3;
```

30

Declaração de Registros

Declaração de Múltiplas Variáveis (Forma 3):

```
typedef  struct {
                tipo_1 campo(s);
                tipo_2 campo(s);
                    ...
                tipo_n campo(s);
                    } tipo_registro;

tipo_registro  variáveis;
```

31

Declaração de Registros

- **Exemplo:**

```
typedef struct {
        char NOME [30];
        char CPF[12];
        char RG[10];
        int HT[6];
        float SALARIO;
          } Registro_Pagamento;
Registro_Pagamento  Reg_Pag1,  Reg_Pag2,  Reg_Pag3;
```

32

Exercícios

- Use **typedef** para declarar um tipo de registro *Endereço* com os campos Rua, No, Cidade, UF, CEP.
- Use o novo tipo declarado no exercício anterior para incrementar o tipo de registro *Registro_Pagamento* declarado nos exemplos anteriores para que este contenha um campo do tipo Endereço

33

Manipulação de Registros

- Como acessar os campos de um registro?
  - *nome_do_registro* **.** *nome_do_campo*
- Exemplos de Atribuição:

```
Reg_Pag1.HT[0] = 163 ;
Reg_Pag1.SALARIO = 852.7 ;
```

Vetor de Inteiros

Real

34

Manipulação de Registros

- Exemplo de Manipulação:

```
printf("Entre o nome do empregado: ");
scanf("%s", Reg_Pag1.NOME);
printf("Entre as horas trabalhadas no bimestre: ");
for (i=0; i<=5; i++) scanf("%d", &Reg_Pag1.HT[i]);
printf("Entre o salario: ");
scanf("%f", &Reg_Pag1.SALARIO);
```

- **Nota:** C ANSI admite a atribuição direta entre estruturas de um mesmo tipo. Por exemplo, Reg_Pag2 = Reg_Pag1

35

Manipulação de Registros

- Quando um dos campos é outro registro:

```
struct  {
    char NOME[30], CPF[12], RG[10];
    int HT[6];
    float SALARIO;
    struct {
        char Rua[20], Cidade[15], UF[3];
        int No, CEP;
        } ENDERECO;
    } Reg_Pag1;
```

Atribuições

```
Reg_Pag1.SALARIO = 1200;
...
Reg_Pag1.ENDERECO.No = 230;
Reg_Pag1.ENDERECO.CEP = 15980;
...
```

36

Manipulação de Registros

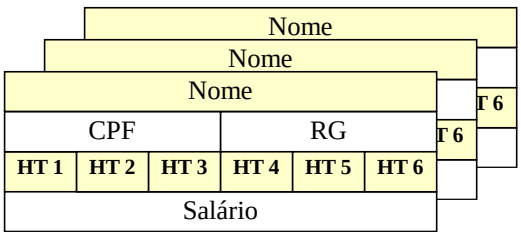
- E se tenho 500 empregados?
  - Declaro 500 variáveis do tipo registro???

37

Arranjos de Registros

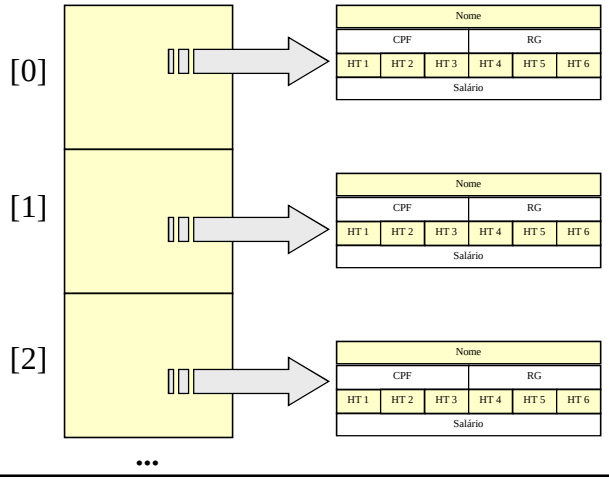
Se, em vez de um <u>única</u> ficha do empregado, quisermos cadastrar <u>várias</u> fichas?

SOLUÇÃO
criar um <u>vetor</u> de <u>registros</u> !



38

Arranjos de Registros



39

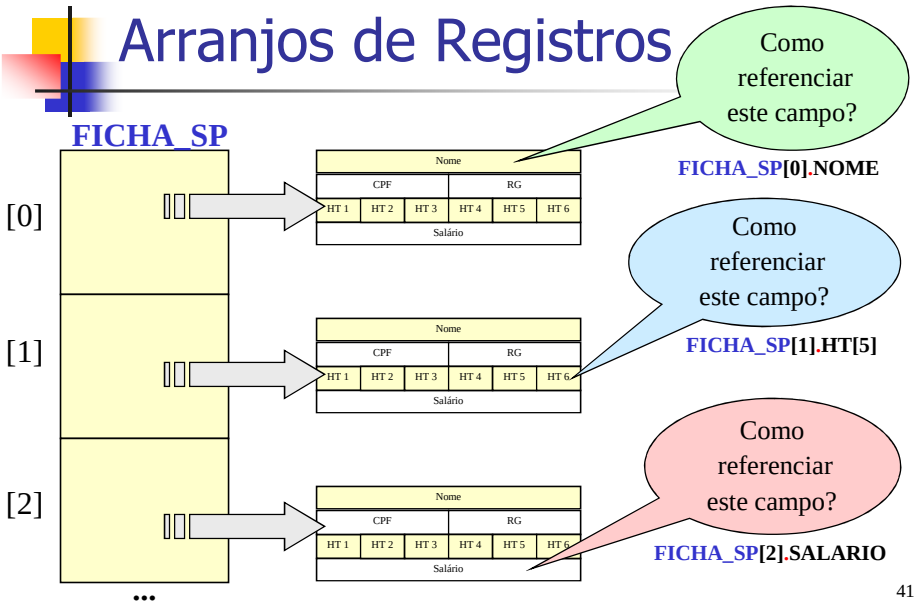
Declaração de Registros

- **Exemplo:**

```
typedef struct {
        char NOME [30];
        char CPF[12];
        char RG[10];
        int HT[6];
        float SALARIO;
                } Registro_Pagamento;
Registro_Pagamento  Ficha_SP[500], Ficha_RJ[300], Ficha_MG[100];
```

40

Arranjos de Registros



41

Arranjos de Registros

Exemplos de Atribuição:

- FICHA_SP[2].SALARIO = 243.45;
- FICHA_SP[6].HT[3] = 228;
- scanf("%s", FICHA_SP[1].NOME);

...

42

Funções

- Podemos ver uma função em C como uma rotina que recebe ou não valores como argumentos e retorna ou não explicitamente um valor
- **Declaração:**

  tipo *nome* (*lista de parâmetros*) **{**
  corpo da função (incluindo declarações de vars. locais)
  }

- Exemplo:

```
double quadrado (double x) {
  return x*x; }
```

43

Funções

- Funções que não retornam valores são do tipo **void**
  - **void** também é usado para indicar a inexistência de parâmetros
- Exemplo:

```
void hello(void){
     printf("Hello World!");
}
```

- O retorno de valor é feito através do comando **return**
  - **return** interrompe imediatamente o fluxo de execução
    - e retorna o valor especificado

44

Funções

- **Escopo:**
  - As variáveis declaradas dentro de uma função possuem **escopo local**, ou seja, provisório e interno à função
  - Variáveis **globais** são declaradas no início de qualquer módulo (arquivo) de programa, fora de qualquer função (incluindo **main**)
- **Passagem de Parâmetros:**
  - Em geral, **por valor** (uma **cópia** do parâmetro é feita)
    - Em C ANSI, isso inclui *structs*
  - Para forçar passagem por **referência**, é preciso utilizar ponteiros
    - É o que ocorre necessariamente com arranjos (vetores/matrizes)

45

Funções Pré-Definidas

- As linguagens de programação têm à sua disposição várias funções pré-definidas
  - Em C, essas funções são organizadas em bibliotecas
- Exemplo (Biblio. Matemática C ANSI – #include <math.h>):
  - pow(b,e) base *b* elevada ao expoente *e*
  - sqrt(x) raiz quadrada de x

Parâmetro formal

Identificador (nome)

46

Ativação de Funções

- Ativação e captura do resultado de uma função pode ser:
  - Por atribuição direta do retorno a uma variável do mesmo tipo
  - Por uso do retorno dentro de uma expressão
- Exemplo (H é **real**, A e Y são **inteiros** ou **reais**):

```
H = sqrt(A + pow(Y,2) + 2 * sin(Y));
```

atribuição direta — uso em expressão

47

Exemplo

- Dados dois números *N* e *K*, calcular a Combinação:

$$C(N,K) = \frac{N!}{K!(N-K)!}$$

- Se existisse uma função **fat**(X) que calculasse o fatorial de um dado X, o cálculo acima ficaria:

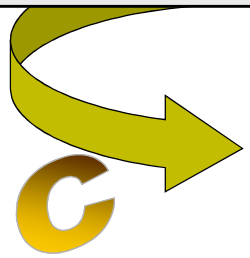
```
C[N,K] = fat(N) / (fat(K) * fat(N-K))
```

- Se não existe, podemos defini-la

48

Exemplo

```
função FAT(inteiro: X): retorna inteiro;
início
    inteiro: P, I;
    P ← 1;
    para I de 1 até X faça P ← P * I;
    retorne P;
fim
```



```
long int FAT(long int X){
      long int I, P;
      P = 1;
      for(I=1; I<=X; I++) P=P*I;
      return P;
}
```

49

Exercício

- Faça um programa em C que receba do usuário dois inteiros, N e K, e utilize a função definida anteriormente para calcular (e depois exibir) a combinação:

$$C[N,K] = \mathbf{fat}(N) / (\mathbf{fat}(K) * \mathbf{fat}(N-K))$$

50

Bibliografia

- Schildt, H. "C Completo e Total", 3a. Edição, Pearson, 1997.
- Damas, L. “Linguagem C”, 10a. Edição, LTC, 2007