Estrutura de Dados

Aula 2 : Estruturas Estáticas

Prof. MSc. Fausto Sampaio

https://github.com/Fausto14/estrutura_de_dados

Centro Universitário UniFanor - Wyden

5 de novembro de 2019

Sumário

Tipos Básicos

Tipos Básicos

Tipo	Tamanho (bytes)	Valor
char	1	Um caractere (ou, reciprocamente, um inteiro de -128 a 127)
int	2 (padrão ANSI) 4 (atualmente)	Número inteiro (ANSI: de -32768 a 32767 (atualmente: de -2147483648 a 2147483647)
float	4	Número real (ponto flutuante com precisão simples: 7 dígitos, limitados a: $+/-3.4*10^{+/-38}$)
double	8	Número real (ponto flutuante com precisão dupla: 15 dígitos, limitados a: $+/-1.7*10^{+/-308}$)

Operadores de Incremento e Decremento

Incremento

```
 4  int main(){
 5      int contador = 3;
 6      printf("%d\n",contador);
 7
 8      contador++;
 9      printf("%d\n",contador);
10
11      contador = contador + 1;
12      printf("%d\n",contador);
13
14      contador += 1;
15      printf("%d\n",contador);
16
17      contador += 5;
18      printf("%d\n",++contador);
19
20      return 0;
21  }
```

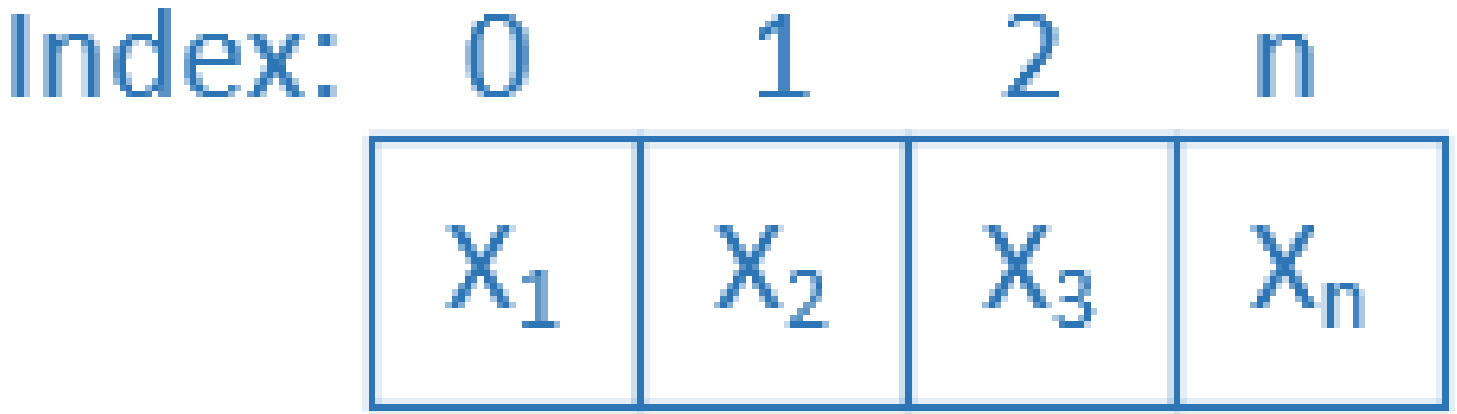
Decremento

```
 4  int main(){
 5      int contador = 20;
 6      printf("%d\n",contador);
 7
 8      contador--;
 9      printf("%d\n",contador);
10
11      contador = contador - 1;
12      printf("%d\n",contador);
13
14      contador -= 1;
15      printf("%d\n",contador);
16
17      contador += 6;
18      printf("%d\n",--contador);
19
20      return 0;
21  }
```

Vetores e Matrizes

Vetores ou Arrays

- **Vetor** ou **Array** é uma estrutura de dados estática.
- Vetores são estruturas de dados consistindo em itens de dados do mesmo tipo relacionados.
- Vetores tornam conveniente processar grupos relacionados de valores.
- O acesso aos elementos de um vetor é por meio de índices únicos que vão de 0 (zero) à "n" posições.
- Em resumo, é um tipo de armazenamento sequencial.
- Exemplo: O índice 1, no exemplo logo abaixo, aponta para X_2.



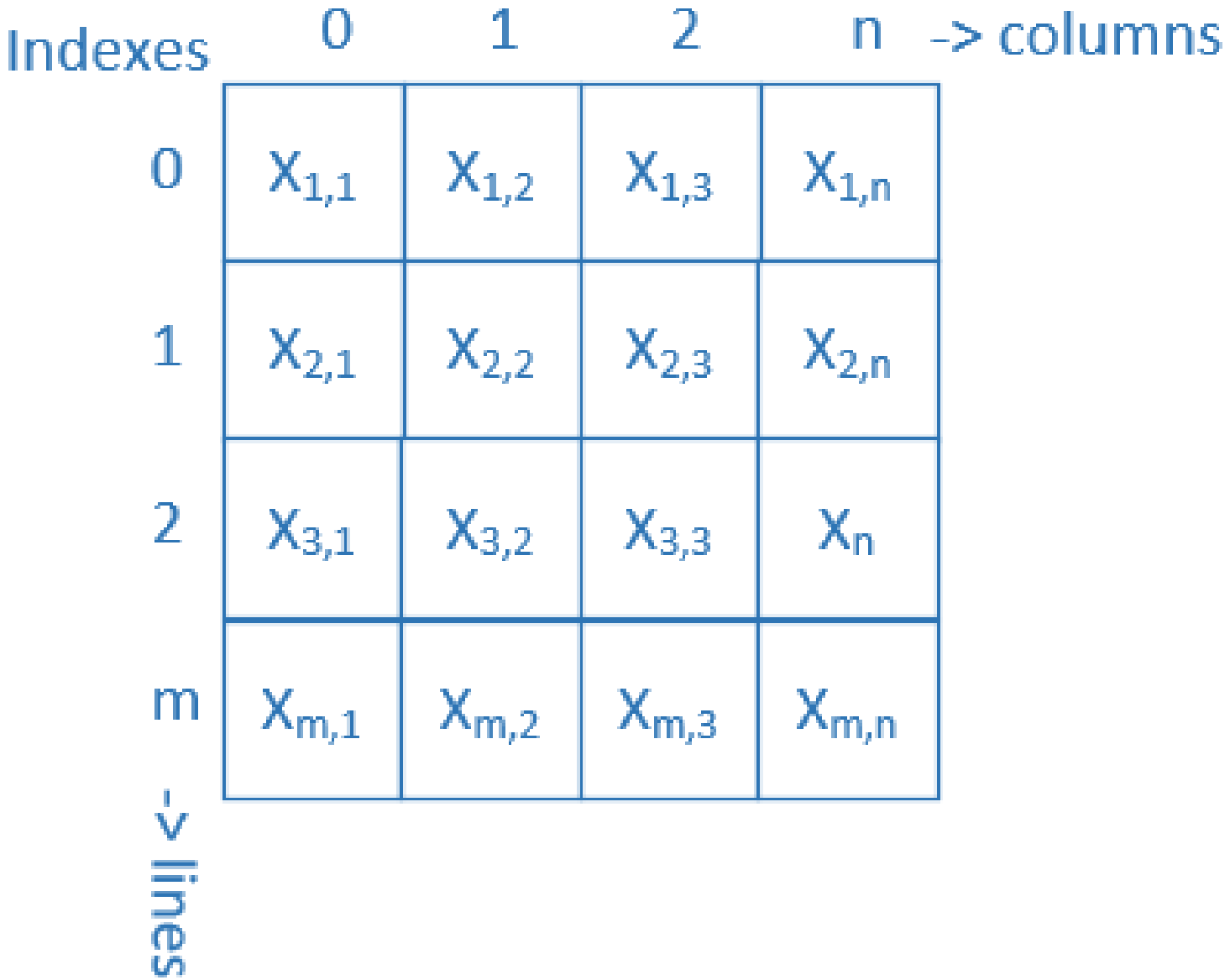
Exemplo em C

```c
int main(){
    int indice = 2;
    int codigo_produto[] = {56, 78, 32, 99, 112};
    float precos[5];
    precos[0] = 10.4;
    precos[1] = 1.0;
    precos[2] = 2.5;
    precos[3] = 10.0;
    precos[4] = 7.6;
    printf("Produto %d : %d  | %.2f",indice, codigo_produto[indice],precos[indice]);
    // saída: Produto 2 : 32  | 2.50
    return 0;
}
```

Matrizes

- **Matrizes** (Array multi-dimensional).
- Trata-se de um vetor de vetores.
- Os vetores ou arrays com duas dimensões costumam ser utilizados para representar tabelas de valores que consistem nas informações dispostas em linhas e colunas.
- Identificamos os elementos da tabela através de 2 índices.
- Por convenção, primeiro identificamos a linha e em seguida a coluna.

Matrizes


Indexes
0
1
2
n
-> columns
0
1
2
m
v lines
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,n}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,n}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ x_n
$x_{m,1}$ $x_{m,2}$ $x_{m,3}$ $x_{m,n}$

Exemplo em C

```c
int main(){
    int qtd_alunos = 4;
    int qtd_notas = 3;
    float notas[qtd_alunos][qtd_notas] = {
                                        {9.0, 5.6, 3.0},
                                        {6.0, 5.0, 7.0},
                                        {8.0, 8.0, 6.0},
                                        {7.0, 5.6, 9.0},
                                    };
    printf("%.2f \n",notas[2][2]);
    return 0;
}
```

Operador sizeof

sizeof

- Permite saber o número de bytes ocupado por um determinado tipo de variável.
- É muito usado na alocação dinâmica de memória.

sizeof

```
 3  int main(void)
 4  {
 5    float nota;
 6    printf(" --- TIPO ---|--- BYTES ---\n");
 7    printf(" char .......: %d bytes\n", sizeof(char));
 8    printf(" short.......: %d bytes\n", sizeof(short));
 9    printf(" int.........: %d bytes\n", sizeof(int));
10    printf(" long........: %d bytes\n", sizeof(long));
11    printf(" float ......: %d bytes\n", sizeof(float));
12    printf(" double......: %d bytes\n", sizeof(double));
13    printf(" long double.: %d bytes\n\n", sizeof(long double));
14    printf("\nO tamanho de nota eh...: %d \n\n",sizeof nota);
15
16    return 0;
17  }
```

sizeof

```
--- TIPO ---|--- BYTES ---
char ........: 1 bytes
short........: 2 bytes
int..........: 4 bytes
long.........: 4 bytes
float .......: 4 bytes
double.......: 8 bytes
long double.: 16 bytes


O tamanho de nota eh...: 4
```

Conversaõ de Tipos

Cast

- C tem um operador para alterar o tipo de um valor explicitamente.
- Este operador é chamado de **cast**.
- Executando um cast de tipos, o valor da expressão é forçado a ser de um tipo particular, não importando a regra de conversão de tipos.
- (*nomedotipo*) expressao

```
printf("%.2f \n", 10/2);
printf("%.2f \n", 10/2.0);
printf("%d \n", 10/2);
printf("%.2f \n", (float)10/2);
```

Função e Bloco

Função e Bloco

```c
int maiorDeIdade(int idade);
int main(){
    int idade;
    printf("Digite sua idade:\n");
    scanf("%d",&idade);
    if(maiorDeIdade(idade) == 1){
        printf("BLOCO DO IF \n");
        printf("Aluno maior de 18 anos.");
    }
    else{
        printf("BLOCO DO ELSE \n");
        printf("Aluno menor de 18 anos.");
    }
    return 0;
}

int maiorDeIdade(int idade){
    if(idade >= 18)
        return 1;
    else
        return 0;
}
```

Variáveis Global e Local

Variáveis Global e Local

```c
#include <stdlib.h>
#include <stdio.h>

#define PI 3.14

float calc_area(float r);

float area;

int main(){
    float raio; // raio é variável local
    printf("Digite o raio da circunferencia:\n");
    scanf("%f",&raio);
    area = calc_area(raio); // area é variável global
    printf("A area eh: %.2f", area);
}

float calc_area(float r){
    return PI * r * r; // r é variável local
}
```

Passagem por referência

Passagem por referência

```c
#include <stdio.h>
//função que soma 10 ao valor recebido
void soma10(int x)
{
  x = x + 10;
  printf("Valor de x apos a soma = %d \n",x);
  return;
}
void soma10p(int *x)
{
  *x = *x + 10;
  printf("Valor de x apos a soma = %d \n",*x);
  return;
}
int main(void)
{
  int numero;
  printf("Digite um numero: ");
  scanf("%d", &numero);
  printf("O numero digitado foi: %d \n",numero);
  soma10(numero);  //chamada da função
  printf("Agora o numero vale: %d \n",numero);
  soma10p(&numero);  //chamada da função com ponteiro como parâmetro
  printf("Agora o numero vale: %d \n",numero);
  return 0;
}
```