PROGRAMAÇÃO DE COMPUTADORES V - TCC- 00.323

Modulo 5: Array

(complemento de dois)

Aura - Erick
aconci@ic.uff.br, erickr@id.uff.br

Roteiro

- Vetores
- Matrizes
- Como são tratados os números negativos
- Operação complemento de dois

Exemplo

Ler as notas de 3 alunos. Calcular a média da turma. Informar quantos alunos tiveram nota acima da média.

```
#include <stdio.h>

main() {
float N1,N2,N3, media;
      int maior;
      printf("Digite as notas dos 3 alunos\n");
      scanf("%f %f %f",&N1, &N2, &N3);
      maior = 0;
      media = (N1 + N2 + N3)/3;
      if (N1 > media)
         maior = maior + 1;
      if (N2 > media)
         maior = maior + 1;
      if (N3 > media)
         maior = maior + 1;
      printf("A media da turma eh %f\n",  media);
      printf("%d alunos tiveram nota maior que a media",  maior);
      system("pause");
}
```

Mas...

- E se fossem 50 alunos?
- Usar 50 variáveis seria muito trabalhoso

- Existe alguma forma de guardar vários valores que seja menos trabalhosa?

...

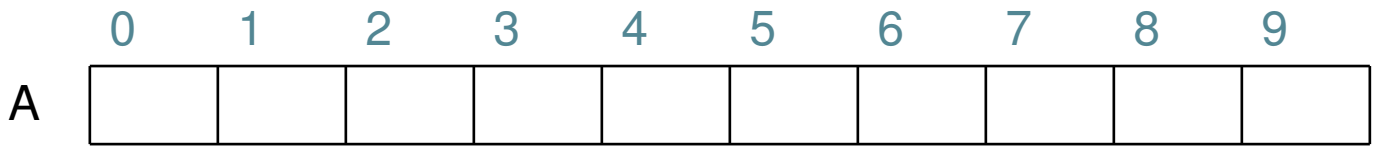
- Sim, **vetores**!!

Vetores

- Conjuntos ordenados de elementos do mesmo tipo

- Exemplo:

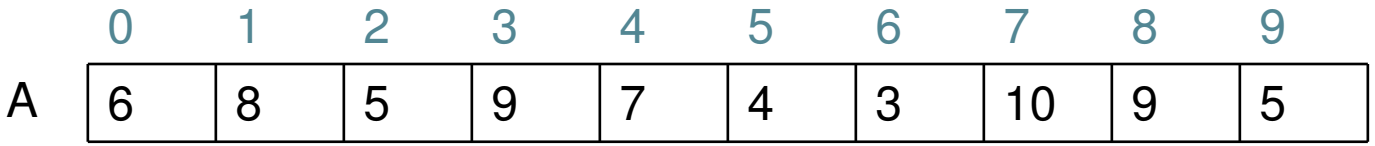
```
int A[10];
```



Vetores

- Exemplo:

int A[10];



- Posições são acessadas pelo índice
  - Índice começa em ZERO
  - A[0] = 6
  - A[1] = 8

Retornando ao exemplo para 50 alunos...

```
#include <stdio.h>

main() {
      float nota[50];
      float soma, media;
      int i, maior = 0;
      //lê as notas
      for (i=0;i < 50; i++) {
          printf("Digite a nota do aluno: ");
          scanf("%f",&nota[i]);
      }
      soma = 0;
      //Calcula a média
      for (i=0;i < 50; i++)
          soma = soma + nota[i];
      media = soma / 50;
      //Conta quantas notas são maiores que a média
      for (i=0;i < 50; i++)
          if (nota[i] > media)
              maior = maior + 1;
      printf("A media da turma eh %f\n",  media);
      printf("%d alunos tiveram nota maior que a media",  maior);
      system("pause");
}
```

Retornando ao exemplo para 50 alunos...

```
#include <stdio.h>

main() {
      float nota[50];
      float soma, media;
      int i, maior = 0;
      //lê as notas
      for (i=0;i < 50; i++) {
          printf("Digite a nota do aluno: ");
          scanf("%f",&nota[i]);
      }
      soma = 0;
      //Calcula a média
      for (i=0;i < 50; i++)
          soma = soma + nota[i];
      media = soma / 50;
      //Conta quantas notas são maiores que a média
      for (i=0;i < 50; i++)
          if (nota[i] > media)
              maior = maior + 1;
      printf("A media da turma eh %f\n",  media);
      printf("%d alunos tiveram nota maior que a media",  maior);
      system("pause");
}
```

Todas as manipulações são feitas com FOR

A média é calculada através do acumulador SOMA

Exemplo

- Preencher por leitura 2 vetores de 10 elementos inteiros (A e B).
- Somar os elementos de igual índice e colocar os resultados em um terceiro vetor (C).

```
#include <stdio.h>

main() {

      int A[10], B[10], C[10];
      int i;
      //lê os dois vetores
      for (i=0;i < 10; i++) {
          printf("\nDigite 2 numeros inteiros:\n ");
          scanf("%d%d",&A[i],&B[i]);
          C[i] = A[i] + B[i];
      }
      //Exibe o vetor resultado (C)
      for (i=0;i < 10; i++) {
          printf("%d ",C[i]);
      }
      system("pause");
}
```

Exemplo

- Guardar 3 produtos, seu estoque e preço e realizar uma entrada no estoque.

- Utilizar um vetor para guardar cada informação: produto, estoque e preço. As informações serão ligadas pelo índice do vetor. Podemos usar o índice como o código do produto.

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

main() {
      char produto[3][20];  //vetor de 3 posições, de strings de tamanho 20
       int estoque[3];
       float preco[3];
      int cod, quant;

       strcpy(produto[0], "caneta");
       strcpy(produto[1], "lapis");
       strcpy(produto[2], "borracha");

       estoque[0] = 10;
       preco[0] = 2;
       estoque[1] = 20;
       preco[1] = 3;
       estoque[2] = 20;
       preco[2] = 4;
       printf("Digite o codigo do produto: ");
      scanf("%d",&cod);
      printf("\nDigite a quantidade: ");
       scanf("%d",&quant);
       estoque[cod] = estoque[cod] + quant;
       printf("O nome estoque de %s eh %d.", produto[cod], estoque[cod]);
       system("pause");
}
```

```
#include <stdio.h>
#include <string.h>
```

→ **Biblioteca para trabalhar com strings**

```
main() {
      char produto[3][20];  //vetor de 3 posições, de strings de tamanho 20
       int estoque[3];
       float preco[3];
      int cod, quant;

       strcpy(produto[0], "caneta");
       strcpy(produto[1], "lapis");
       strcpy(produto[2], "borracha");

       estoque[0] = 10;
       preco[0] = 2;
       estoque[1] = 20;
       preco[1] = 3;
       estoque[2] = 20;
       preco[2] = 4;
       printf("Digite o codigo do produto: ");
      scanf("%d",&cod);
      printf("\nDigite a quantidade: ");
       scanf("%d",&quant);
       estoque[cod] = estoque[cod] + quant;
       printf("O nome estoque de %s eh %d.",produto[cod],estoque[cod]);
       system("pause");
}
```

→**Atribuição de strings tem que ser feita através da função strcpy**

Matrizes

- Armazenar as 5 notas de trabalhos de um aluno e sua média

- Como armazenar a média de todos os alunos da turma?

Matrizes

- Supondo uma turma de 10 alunos

```
float NOTAS[6][10];

//Primeira nota do sexto aluno foi 4
NOTAS[0][5] = 4;
```

NOTAS:

	0	1	2	3	4	5
0	5	8	7	10	9	7.8
1	5					
2	6					
3	7					
4	9					
5	**4**					
6						
7						
8						
9						

Vetores X Matrizes

- Cada linha da matriz é um vetor de notas de um aluno

```
float NOTAS[6][10];
float NOTASAluno[6];
int i;

//Preencher o vetor NOTASAluno com as notas do primeiro aluno
for (i=0;i<6;i++)
    NOTASAluno[i] = NOTAS[i][0];
```

Inicialização de vetores

- Inicialização de um vetor de inteiros

  ```
  int A[5] = {1, 2, 3, 2, 1};
  ```

- Inicialização de um vetor de floats

  ```
  float B[4] = {1.3, 1.5, 50.5, 30.4};
  ```

- Inicialização de um vetor de strings

  ```
  char nomes [3][10] = { "Joao", "Maria", "Jose" };
  ```

Inicialização de matrizes

- **matrx** está sendo inicializada com 1, 2, 3 e 4 em sua primeira linha, 5, 6, 7 e 8 na segunda linha e 9, 10, 11 e 12 na última linha.

```
int matrx [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

Matrizes multidimensionais

- As matrizes podem ter mais de 2 dimensões. Basta para isso declarar mais índices.

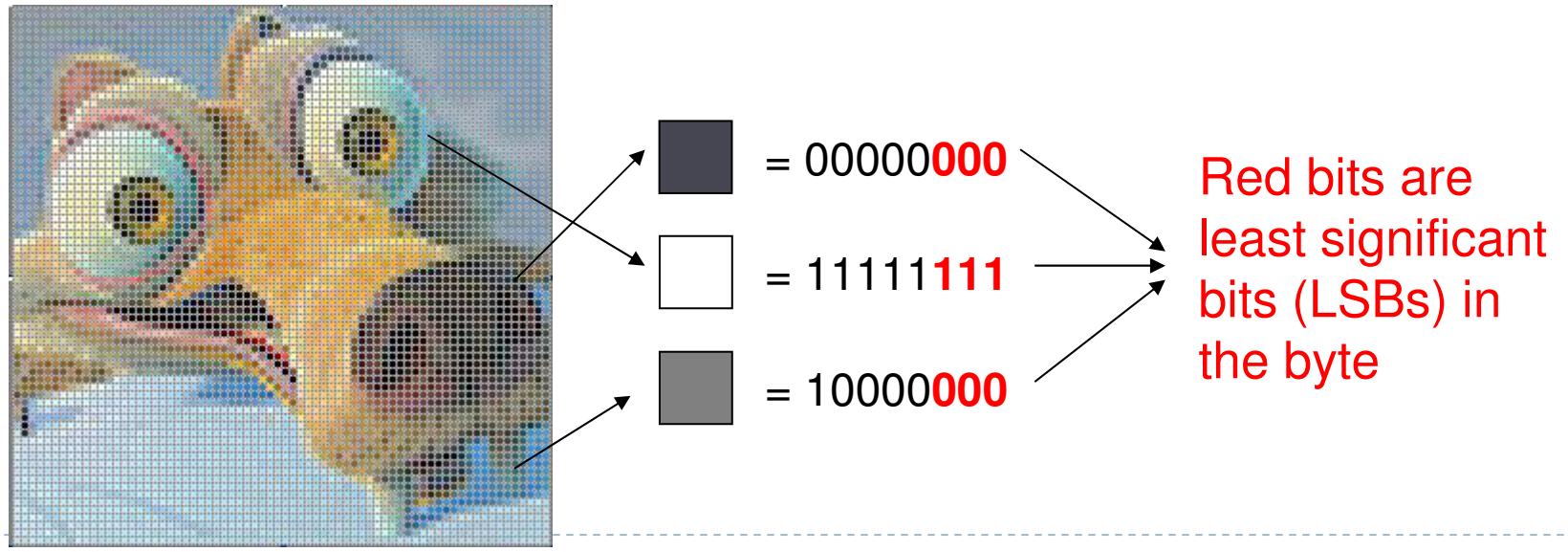
- Exemplo:

coordenadas dos vértices de um cubo

```
int M[10][10][10];
M[1][2][2] = 20;
```

Noções de Estruturas de dados

- Tudo em computação é apenas representado em zeros e uns
- Mesmo imagens



Números

- Em zeros e uns
- Base binária
- Base octal
- Base hexadecimal

valor	bits
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Números Negativos

- Números inteiros (positivos e negativos) ou os números inteiros da Matemática, em C, são chamados INTEIROS COM SINAL

- E DELARADOS COMO int i;

Ou em outras palavras para criar uma variável **numeroInteiro** deste tipo, basta dizer

int numeroInteiro;

limites dos inteiros

- Cada **int** é armazenado em ***s*** bytes consecutivos, sendo ***s*** o valor da expressão **sizeof (int)**
- Assim, cada **int** é representado por uma seqüência de **8*s*** bits e portanto tem $\mathbf{2}^{8s}$ possíveis valores. O conjunto desses valores é

$$-2^{8s-1}, \ldots, -1, 0, 1, \ldots, 2^{8s-1}-1$$

Se ***s* = 2**, por exemplo, o conjunto de valores vai de $\mathbf{-2^{15}}$ a $\mathbf{2^{15}-1}$, ou seja, de –32768 a 32767.

Se ***s* = 4**, o conjunto de valores vai de -2^{31} a $2^{31}-1$, ou seja, de –2147483648 a 2147483647.

limites dos inteiros

- Os números -2^{8s-1} e $2^{8s-1}-1$ são os valores das constantes

INT_MIN e **INT_MAX**,

definidas na interface **limits.h**

O bit mais significativo da seqüência de **s bytes** é reservado para **indicar o sinal.**

Cada seqüência de 8*s* bits que começa com **0** representa, em notação binária, um **int positivo**.

Cada seqüência de 8*s* bits que começa com **1** representa o inteiro estritamente negativo $k - 2^{8s}$, sendo k o valor da seqüência em binários.

exemplificando

- Se tivéssemos um total de 4 bits.
- Uma variável inteira poderia assumir $2^4 = 16$ valores diferentes.

A convenção **complemento-de-dois** usa essas 16 possibilidades para representar os números:

valor	bits
−8	1000
−7	1001
−6	1010
−5	1011
−4	1100
−3	1101
−2	1110
−1	1111
+0	0000
+1	0001
+2	0010
+3	0011
+4	0100
+5	0101
+6	0110
+7	0111

–8, –7, –6, –5, –4, –3, –2, –1, +0, +1, +2, +3, +4, +5, +6, +7

limites dos inteiros

- **short int** e **int**: -32,767 to 32,767
- **unsigned short int** e **unsigned int**: 0 to 65,535
- **long int**: -2,147,483,647 até 2,147,483,647
- **unsigned long int**: 0 to 4,294,967,295
- **long long int**: -9,223,372,036,854,775,807 até 9,223,372,036,854,775,807
- **unsigned long long int**:
  - 0 até 18,446,744,073,709,551,615

Complemento-de-dois

- Esta maneira de representar os inteiros estritamente negativos é conhecida como *complemento-de-dois* (ou *two's-complement*).

- Para descobrir a seqüência de bits de um inteiro estritamente negativo, basta tomar a seqüência de bits do correspondente inteiro estritamente positivo, inverter todos os bits — ou seja,
trocar 0 por 1 e 1 por 0 — e somar 1, em binário, ao resultado.

Quanto vale ?

- 1 1 1 1 1 0 1 0

- É um valor estritamente negativo

- Invertendo os bits teremos:
- 0 0 0 0 1 0 1
- 101 representa o número 5 , pois
- $1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$
- 4 + 1 = 5
- Somando 1: 6 logo - 6 !

- Ou $k - 2^{8s}$

- $K = 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0 =$

  $= 1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = =$

  $= 128 + 64 + 32 + 16 + 8 + 2 =$

  $= 250$

Para esse tamanho em bits $2^8 = 256$

Logo: 250 – 256 = -6 !!!!