Algoritmos II

VETORES

Vetores

- Imagine que fosse necessário criar um programa para armazenar 2000 salários dos funcionários de uma empresa.
- Caso só se quisesse somar os salários poderia se criar um laço de repetição, ler 2000 vezes a mesma variável, e ir acumulando os valores.
- Mas, e se fosse necessário acessar esses salários novamente para outras funções? Os valores estariam perdidos. Então, neste caso, seria necessário criar 2000 variáveis.
- Criar 2000 variáveis se torna inviável. Para resolver isso usam-se arrays, que podem ser tanto vetores quanto matrizes.

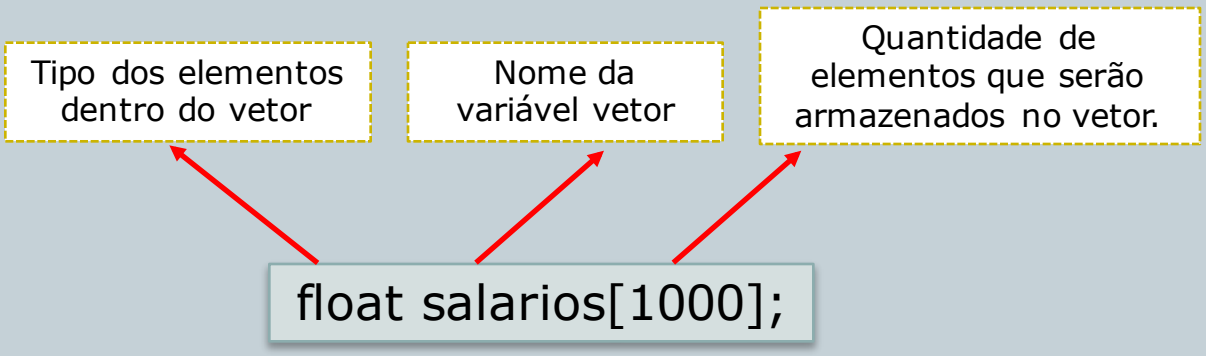
Vetores

- Mas, *o que é um vetor?*
- Uma sequência de vários valores de mesmo tipo, armazenados sequencialmente na memória, e fazendo uso de um mesmo nome de variável para acessar esses valores.
- Vetores são variáveis que servem para guardar vários valores do mesmo tipo, de forma uniforme, na memória.
- E como se declara um vetor? Basicamente, como uma variável normal.
  - Primeiro, coloca-se o tipo de dados que serão armazenados no vetor, ou seja, se o vetor conterá valores int, float, string, …

Vetores

- Logo após, se dá um nome para o vetor (assim como as variáveis). Vale lembrar que as regras para nomear uma variável também são válidas para um vetor.
- E finalmente, a quantidade de elementos que serão armazenados pelo vetor, entre [].

- Exemplos:
  - int vet[10];
  - float salarios[100], x, aumento[100];
  - char nome[30];



Vetores

- A variável vet possui 4 espaços de memória onde podem ser guardados 4 números inteiros.
- Para acessar o valor interno de uma variável é só chamar pelo nome da mesma. Por exemplo: printf("%d",idade).
- Mas o vetor possui mais que uma posição. Como acessar um determinado elemento?

- Após declarado, como se manipular um vetor?

Vetores

- A variável vet possui 4 espaços de memória onde podem ser guardados 4 números inteiros.
- Para acessar o valor interno de uma variável é só chamar pelo nome da mesma. Por exemplo: printf("%d",idade);.
- Mas o vetor possui mais que uma posição. Como acessar um determinado elemento?
- O vetor trabalha com índices, ou posições, que no C/C++ iniciam de 0 (sempre).

vet			
0	1	2	3

Vetores

- Para inserir o valor 76 na posição 2 seria necessário o seguinte código:
  - `vet[2] = 76;`
- Ou seja, para atribuir um valor a uma posição do vetor é como uma variável, com o acréscimo do índice.
- É como um carteiro. Para entregar a carta ele precisa saber o nome da rua e o número da residência.
- O nome da rua não deixa de ser o nome do vetor, e o número da residência é o índice do vetor.

Vetores

- *Lembre-se:* o limite do vetor é sempre seu tamanho menos 1. Usando o exemplo: vetor de tamanho 4, posição máxima é 3 (pois 4-1=3). Então, ao atribuir um valor a posição 4 ocorrerá um erro. Resumidamente, jamais se poderia fazer vet[4]=200.
- Exemplos:

Vetores

- Supondo a declaração de um vetor de 6 números inteiros chamado vetor
  - **int vetor[6];**

Vetor [0]	**Vetor [1]**	**Vetor [2]**	**Vetor [3]**	**Vetor [4]**	**Vetor [5]**

- Supondo a colocação do valor 123 na primeira posição:
  - vetor[0] = 123

123					
Vetor [0]	**Vetor [1]**	**Vetor [2]**	**Vetor [3]**	**Vetor [4]**	**Vetor [5]**

Vetores

- Preenchendo a última posição do vetor com o dobro do valor do primeiro elemento
  - vetor[5] = vetor[0]*2;

123					**246**
Vetor [0]	**Vetor [1]**	**Vetor [2]**	**Vetor [3]**	**Vetor [4]**	**Vetor [5]**

- Preenchendo o terceiro elemento do vetor com a soma do primeiro com o último elemento:
  - vetor[2] = vetor[0] + vetor [5];

123		**369**			**246**
Vetor [0]	**Vetor [1]**	**Vetor [2]**	**Vetor [3]**	**Vetor [4]**	**Vetor [5]**

Carga inicial automática de vetores

- É possível iniciar todos os elementos do vetor automaticamente através da seguinte sintaxe:
  - Tipo var[n] = {valor1, valor2, ..., valor n};
    - Exemplo:
      - char vogal[5] = {‘a’, ‘e’ , ‘i’, ‘o’, ‘u’};
    - Evitando a escrita do código:
      - Char vogal[5];
        - Vogal[0] = ‘a’;
        - Vogal[1] = ‘e’;
        - Vogal[2] = ‘i’;
        - Vogal[3] = ‘o’;
        - Vogal[4] = ‘u’;

Exemplos de declaração

- Vetor com 10 elementos não iniciados
  - int v[10];
- Vetor com 3 elementos automaticamente iniciados com os valores 5, 10 e 15
  - int v[3] = {5, 10, 15};
- Vetor com 3 elementos iniciados com os valores 5, 10 e 15
  - int v[] = {5, 10, 15};

Vetores

- Faça um programa que preencha um vetor de 10 posições e exiba o mesmo na tela.

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

#define TAM 10

int main() {
    setlocale(LC_ALL,"Portuguese");

    int numeros[TAM], i;

    for (i = 0; i < TAM; i++)  {
        printf("Posição %d do vetor: ", i);
        scanf("%d",&numeros[i]);
    }

    printf("\n\n");

    for (i = 0; i < TAM; i++)  {
        printf("%d\t", numeros[i]);
    }

    return 0;
}
```

Vetores

- Ao ler uma variável, acessamos somente uma posição. No caso do vetor é mais que uma posição acessada pela mesma variável. Imagine um vetor de 1000 posições, sendo lido posição a posição:
  - scanf("%d%d%d%d%d…",&vet[0],&vet[1],&vet[2],&vet[3],&vet[4]…); até a milésima posição.
- Dessa forma, fica melhor se utilizar um laço de repetição para acessar as posições.
- Seguem outros exemplos.

Vetores

- Faça um programa que preencha um vetor de 20 posições e exiba todos os valores pares e suas respectivas posições.

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

#define TAM 20

int main() {
    setlocale(LC_ALL,"Portuguese");

    int numeros[TAM], i;

    for (i = 0; i < TAM; i++)  {
        printf("Posição %d do vetor: ", i);
        scanf("%d",&numeros[i]);
    }

    printf("\n\n");

    for (i = 0; i < TAM; i++)  {
        if (numeros[i] % 2 == 0) {
            printf("\nElemento %d na posição %d\n", numeros[i],i);
        }
    }

    return 0;
}
```

Vetores

- Existe, normalmente, confusão ao que se refere a acessar o elemento dentro do vetor e o índice.
- Toda vez que é necessário acessar o elemento dentro do vetor é necessário a composição nome do vetor e posição a ser acessada (seja a posição fixa ou representada por um índice).
- No exemplo anterior era necessário verificar os valores, ou seja, o que foi inserido pelo usuário no vetor. Então, para acessar é utilizado numeros[i] (nome do vetor e posição representada por índice). Mas ao se mostrar somente a posição, é a variável i (neste caso) que é exibida, pois ela representa a posição.

Vetores

- Elabore um algoritmo que preencha um vetor de 15 posições e o imprima de forma invertida.

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

#define TAM 15

int main() {
    setlocale(LC_ALL,"Portuguese");

    int vet[TAM], posicao;

    for (posicao = 0; posicao < TAM; posicao++) {
        printf("Posição %d do vetor: ", posicao);
        scanf("%d",&vet[posicao]);
    }

    printf("\n\n");

    for (posicao = TAM - 1; posicao>= 0; posicao--) {
        printf("%d\t", vet[posicao]);
    }

    return 0;
}
```

Vetores

- O vetor foi preenchido de forma normal.
- A grande diferença é que o vetor precisa ser impresso de forma invertida, ou seja, iniciar da última posição. Foi criado um laço que iniciou do tamanho – 1, ou seja, se o vetor possui 15 posições precisa iniciar da posição 14 e ir decrementando até a posição 0.
- Este é um erro efetuado constantemente. Quando se precisa acessar a última posição se utilizar o tamanho do vetor. Lembre-se: a quantidade total de elementos é usado na declaração do vetor. Para acessar a última posição do vetor, como o índice inicia de 0, é sempre tamanho – 1, ou seja, se tiver 100 posições declaradas a última posição é 99.

Passagem de vetores para funções

- Suponha a seguintes declarações:
  - Int V[10]
  - Int X[20]
- Suponha que a intenção seja iniciar os vetores V e X com zero em todas as posições utilizando uma função para realização da carga inicial
- Como os vetores tem dimensões diferentes, teríamos que definir duas funções distintas. Uma para o vetor com 10 inteiros e outra com o de 20 inteiros

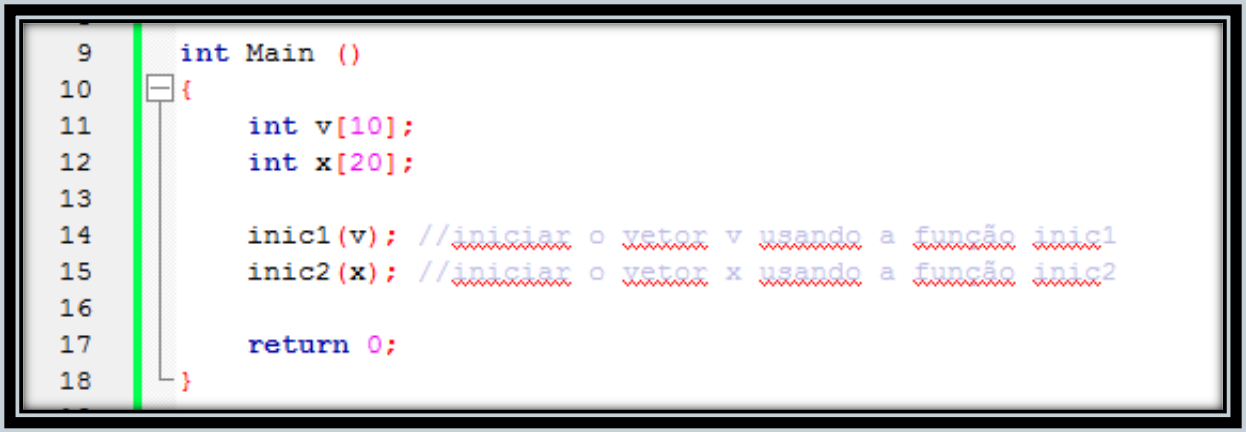
Passagem de vetores para funções

- O código ficaria da seguinte forma para cada uma das funções:

```
18  void inic1(int s[10])
19  {
20      int i;
21      for(i=0;i<10;i++)
22          s[i]=0;
23  }
24  void inic2(int s[20])
25  {
26      int i;
27      for(i=0;i<20;i++)
28          s[i]=0;
29  }
30
```

Passagem de vetores para funções

- Prontas as funções devemos realizar o seguinte procedimento para invocação dessas funções na main:



```
 9  int Main ()
10  {
11      int v[10];
12      int x[20];
13
14      inic1(v); //iniciar o vetor v usando a função inic1
15      inic2(x); //iniciar o vetor x usando a função inic2
16
17      return 0;
18  }
```

  - Lembrando que para enviar um vetor com 10 inteiros a função inic1 esta deverá ter um parâmetro do mesmo tipo de variável.

Passagem de vetores para funções

- As duas funções de carga inicial podem ser substituídas por uma única que inicie qualquer vetor de inteiros com valor 0.
- Isso é possível porque em C não interessa qual a dimensão do vetor que é passado a uma função, mas sim qual o tipo dos seus elementos, e, nesse caso ambos os vetores são constituídos por inteiros.

Passagem de vetores para funções

- No entanto é necessário indicar a essa função o número de elementos que o vetor contém:

```
1   #include <iostream>
2   #include <conio.h>
3
4   void inic(int s[], int n)
5   {
6       int i;
7       for(i=0;i<n;i++)
8           s[i]=0;
9   }
10
11  int Main ()
12  {
13      int v[10];
14      int x[20];
15
16      inic(v,10);
17      inic(x,20);
18
19      return 0;
20  }
```

  - Agora a função void inic(int s[], int n) recebe um vetor de inteiros (sem indicar qual a sua dimensão) e um inteiro que indica o numero de elementos a iniciar