9002 — Aula 17 Algoritmos e Programação de Computadores

Instituto de Engenharia – UFMT

Segundo Semestre de 2014

24 de novembro de 2014

Roteiro

- Revisão
- Vetores em funções
- 3 Exercícios
- Informações Extras: Inicialização de um vetor
- O Problema da Busca
- 6 Busca Sequencial

Recapitulando....

- O que é um vetor?
- Como declarar?
- Como acessar suas posições?
- Mãos a obra..

Recapitulando....

- O que é um vetor?
- Como declarar?
- Como acessar suas posições?
- Mãos a obra...

Exercício: Produto Interno de dois vetores

- Ler dois vetores de dimensão 5 e computar o produto interno destes.
- Quais tipos de variáveis usar?

Exercício: Produto Interno de dois vetores

Ler dois vetores de dimensão 5 e computar o produto interno destes.

```
#include <stdio.h>
int main(){
  double vetor1[5], vetor2[5], resultado;
  int i;
  for(i=0: i<5: i++){
    printf("Entre com valor %d para vetor 1:",i+1);
    scanf("%lf",&vetor1[i]);
  for(i=0: i<5: i++){
    printf("Entre com valor %d para vetor 2:",i+1);
    scanf("%lf", &vetor2[i]);
  //calculando o produto interno
  resultado = 0.0:
  for(i=0; i < 5; i++){
    resultado = resultado + ( vetor1[i]*vetor2[i] ):
  printf("\n\n0 produto interno e: %lf\n",resultado);
  return 0;
}
```

- Vetores também podem ser passados como parâmetros em funções.
- Ao contrário dos tipos simples, vetores têm um comportamento diferente quando usados como parâmetros de funções.
- Quando uma variável simples é passada como parâmetro, seu valor é atribuído para uma nova variável local da função.
- No caso de vetores, não é criado um novo vetor!
- Isto significa que os valores de um vetor são alterados dentro de uma função!

```
#include <stdio.h>
void fun1(int vet[], int tam){
  int i:
  for(i=0;i<tam;i++)
     vet[i]=5;
}
int main(){
  int x[10];
  int i;
  for(i=0;i<10;i++)
    x[i]=8;
  fun1(x,10);
  for(i=0;i<10;i++)
    printf("%d\n",x[i]);
  return 0;
```

- Vetores não podem ser devolvidos por funções.
- Mas mesmo assim, podemos obter um resultado parecido com isso, usando o fato de que vetores são alterados dentro de funções.

```
#include <stdio.h>
int[] leVetor() {
   int i, vet[100];
   for (i = 0; i < 100; i++) {
      printf("Digite um numero:");
      scanf("%d", &vet[i]);
   }
   return vet;
}</pre>
```

O código acima não compila, pois não podemos retornar um int[] .

 Mas como um vetor é alterado dentro de uma função, podemos criar a seguinte função:

```
#include <stdio.h>
void leVetor(int vet []. int tam) {
  int i:
  for(i = 0; i < tam; i++){}
    printf("Digite numero:");
    scanf("%d", &vet[i]);
void escreveVetor(int vet[], int tam){
  int i:
  for(i=0; i< tam; i++)
    printf("vet[%d] = %d\n",i,vet[i]);
}
```

```
int main(){
 int vet1[10], vet2[20];
 printf(" ----- Vetor 1 -----\n");
 leVet(vet1, 10);
 printf(" ----- Vetor 2 ----\n");
 leVet(vet2, 20);
 printf(" ----- Vetor 1 ----\n");
 escreveVet(vet1, 10);
 printf(" ----- Vetor 2 ----\n"):
 escreveVet(vet2, 20);
 return 0;
```

Exercício

Crie uma função
 int maiorValor(int vet[], int tam)
 que recebe como parâmetros um vetor e seu tamanho e devolve o
 maior valor armazenado no vetor.

Exercício

 Crie uma função double media(int vet[], int tam) que recebe como parâmetros um vetor e seu tamanho e devolve a média dos valores armazenados no vetor.

Informações Extras: Inicialização de um vetor

- Em algumas situações é necessário declarar e já atribuir um conjunto de valores contantes para um vetor.
- Em C, isto é feito atribuindo-se uma lista de elementos para o vetor na sua criação da seguinte forma:

```
\begin{tabular}{ll} $<$tipo}>$ identificador [] = \{elementos separados por vírgula\} ; \end{tabular}
```

• Exemplos:

```
double vet1[] = \{2.3, 3.4, 4.5, 5.6\};
int vet2[] = \{5, 4, 3, 10, -1, 0\};
```

Informações Extras: Inicialização de um vetor

```
#include <stdio.h>
int main(){
  double vet1[] = \{2.3, 3.4, 4.5, 5.6\};
  int vet2[] = \{5, 4, 3, 10, -1, 0\};
  int i;
  for(i=0; i<4; i++)
    printf("%lf\n", vet1[i]);
  for(i=0; i<6; i++)
    printf("%d\n", vet2[i]);
```

Considere o seguinte problema:

Há coleção de elementos (cada qual com um identificador único) e uma chave de busca. Deseja-se verificar se algum elemento desta coleção possui um identificador como o mesmo valor da chave de busca.

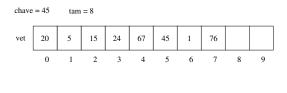
- Em nossos exemplos, usaremos um vetor de inteiros para representar os identificadores da coleção.
- Note que apesar de usarmos inteiros, os algoritmos servem para buscar elementos em qualquer coleção de elementos que possuam chaves que possam ser comparadas.

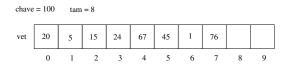
O problema da busca é um dos mais básicos em Computação e possui diversas aplicações.

 E, por exemplo, pode ser utilizado para encontrar um aluno com um determinado número de carteira de motorista, RG, CPF ou coisa que o valha

- Neste curso, veremos algoritmos simples para realizar a busca assumindo que dados estão em um vetor.
- Em cursos mais avançados (estruturas de dados, projeto e análise de algoritmos...) são estudados outros algoritmos e estruturas (que não um vetor) para armazenar e buscar elementos.

- Nos nossos exemplos vamos criar a função:
 - ▶ int busca(int vet[], int tam, int chave), que recebe um vetor com um determinado tamanho, e uma chave para busca.
 - ► A função deve retornar o índice do vetor que contém a chave ou -1 caso a chave não esteja no vetor.





Pode-se perceber que o valor (esperado) a ser devolvido pela função é 5 e -1, respectivamente.

Busca Sequencial

- O algoritmo de busca sequencial pode ser enunciado em três simples passos:
 - Percorrer vetor comparando a chave com cada elemento.
 - Se a chave for entrada, então devolva o índice deste elemento.
 - Caso contrário (todo o vetor foi percorrido), devolva -1.

Busca Sequencial

```
int buscaSequencial(int vet[], int tam, int chave){
  int i;
  for(i=0; i<tam; i++){
    if(vet[i] == chave)
      return i;
  }
  return -1;
}</pre>
```

Busca Sequencial

```
#include <stdio.h>
int buscaSequencial(int vet[], int tam, int chave);
int main(){
  int pos, vet[] = {20, 5, 15, 24, 67, 45, 1, 76, -1, -1}; //-1 indica
                                                           //posição não usada
  pos = buscaSequencial(vet, 8, 45);
  if(pos != -1)
    printf("A posicao da chave 45 no vetor é: %d\n", pos);
  else
    printf("A chave 45 não está no vetor! \n");
  pos = buscaSequencial(vet, 8, 100);
  if(pos != -1)
    printf("A posicao da chave 100 no vetor é: %d\n", pos);
  else
    printf("A chave 100 n\u00e3o est\u00e1 no vetor! \n");
}
int buscaSequencial(int vet[], int tam, int chave){
  int i:
  for(i=0: i<tam: i++){
    if(vet[i] == chave)
      return i;
  return -1;
```

Exercício

- Crie uma função
 int checa(int vet[], int tam, int C)
 que recebe como parâmetros um vetor, seu tamanho e um inteiro C.
 A função deve retornar 1 caso existam dois elementos distintos do vetor tal que a multiplicação destes é C.
- Exemplo: Se vet = (2, 4, 5, -10, 7) e C = 35 então a função deve devolver 1. Mas se C = -1 então a função deve devolver 0.

FIM

Nas próximas aulas, estudaremos o problema da ordenação...