Funções: vetores , busca e ordenação

Instituto de Computação – Unicamp

24 de Abril de 2014

Roteiro

Vetores em funções

- 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!
- Entretanto, passar como parâmetro uma posição indexada do vetor não altera seu valor.

Vetores em funções

- Passar como parâmetro uma posição indexada do vetor não altera seu valor.

```
#include <stdio.h>

void naoAltera(int a){
  a = a + 10;
}

int main(){
  int x[10], i;

  for(i=0;i<10;i++)
    x[i]=8;

  naoAltera(x[0]);
  for(i=0;i<10;i++)
    printf("%d\n",x[i]);
}
```

Vetores em funções

- Passar como parâmetro o nome do vetor permite alterar os valores armazenados no vetor.

```
#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]);
}
```

Vetores em funções

- 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[] leVet() {
  int i, vet[100];
  for (i = 0; i < 100; i++) {
    printf("Digite um numero:");
    scanf("%d", &vet[i]);
  }
}
```

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

Vetores em funções

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

```
#include <stdio.h>

void leVet(int vet[], int tam){
  int i;
  for(i = 0; i < tam; i++){
    printf("Digite numero:");
    scanf("%d",&vet[i]);
  }
}

void escreveVet(int vet[], int tam){
  int i;
  for(i=0; i< tam; i++)
    printf("vet[%d] = %d\n",i,vet[i]);
}
```

Vetores em funções

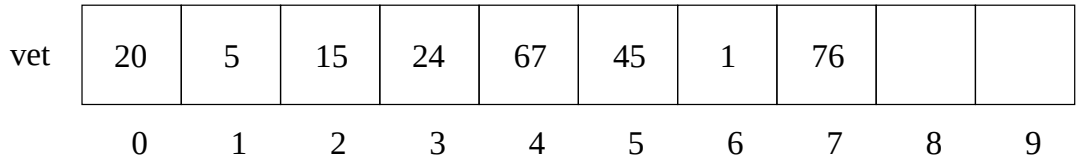
```
int main(){
  int vet1[10], vet2[20];

  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);

}
```

O Problema da Busca

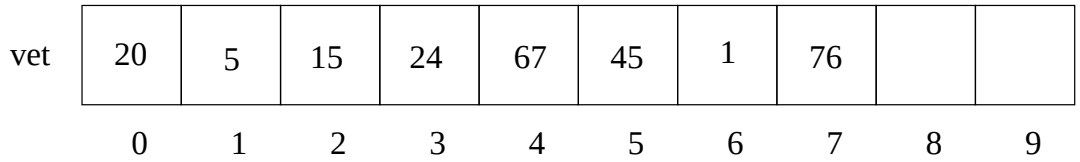
- Nos nossos exemplos vamos criar a função:
  - **int busca(int vet[], int tam, int chave)**, que recebe um vetor com um determinado tamanho, e uma chave para busca.
  - A função deve retornar o índice do vetor que contém a chave ou -1 caso a chave não esteja no vetor.

O Problema da Busca

chave = 45 tam = 8



chave = 100 tam = 8



No primeiro exemplo a função deve retornar 5, enquanto no segundo a função deve retornar -1.

Busca Sequencial

- A busca sequencial é o algoritmo mais simples de busca:
  - Percorra todo o vetor comparando a chave com o valor de cada posição.
  - Se for igual para alguma posição, então devolva esta posição.
  - Se o vetor todo foi percorrido então devolva -1.
- O que é necessário como entrada para a busca sequencial?
- Como é o protótipo da função de busca sequencial?

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;
}
```

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 não está 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;
}
```

Busca Binária

- O que é necessário como entrada para a busca binária?
- Como é o protótipo da função de busca sequencial?

Busca Binária

```
int buscaBinaria(int vet[], int tam, int chave){
  int posIni=0, posFim=tam-1, posMeio;

  while(posIni <= posFim){ //enquanto o vetor tiver pelo menos 1 elemento
    posMeio = (posIni+posFim)/2;

    if(vet[posMeio] == chave)
      return posMeio;
    else if(vet[posMeio] > chave)
      posFim = posMeio - 1;
    else
      posIni = posMeio + 1;
  }

  return -1;
}
```

Busca Binária

```
int main(){
  int vet[] = {20, 5, 15, 24, 67, 45, 1, 76, 78, 100};
  int pos, i;

  //antes de usar a busca devemos ordenar o vetor
  insertionSort(vet,10); //veremos em seguida
  printf("Vetor Ordenado:");
  for(i =0; i<10; i++){
    printf("%d, ", vet[i]);
  }
  printf("\n");
  pos = buscaBinaria(vet, 10, 15);
  if(pos != -1)
    printf("A posicao da chave 15 no vetor é: %d\n", pos);
  else
    printf("A chave 15 não está no vetor! \n");


}
```

- Como seria o protótipo do InsertionSort?

```
void insertionSort(int vet[], int tam){
    int i,j, aux;
    for(i=1; i<tam; i++){
        aux = vet[i];
        j=i-1;
        while( (j>=0) && (vet[j] > aux)  ){
            vet[j+1] = vet[j];
            j--;
        }
        vet[j+1] = aux;
    }
}
```

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.

Exercício

- Crie uma função
  int verifica(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.

Exercícios

- Refaça as funções de busca sequencial e busca binária assumindo que o vetor possui chaves que podem aparecer repetidas. Neste caso, você deve retornar em um outro vetor todas as posições onde a chave foi encontrada.
  Protótipo: **int busca(int vet[], int tam, int chave, int posicoes[]);**
- Você deve devolver em **posicoes[]** as posições de **vet** que possuem a **chave**, e o retorno da função é o número de ocorrências da chave.
  - **OBS:** Na chamada desta função, o vetor **posições** deve ter espaço suficiente (tam) para guardar todas as possíveis ocorrências.