ICC - ORDENAÇÃO DE VETORES E PESQUISA BINÁRIA

PROF. FERNANDO W CRUZ

Roteiro da aula

- PESQUISA SEQUENCIAL EM VETORES
- ORDENAÇÃO DE VETORES (MÉTODO DA BOLHA)
- PESQUISA BINÁRIA

Discussão inicial (I)

- Em linguagem C os vetores e matrizes são estruturas de dados homogêneas importantes porque ajudam a solucionar problemas típicos de programação.
- No entanto, é preciso saber manipular os índices de um vetor/matriz, assim como o seu conteúdo. Vejamos um exemplo. Suponha o vetor CRR abaixo:

CRR	!	U	O	T	R	E	C	A
	1	2	3	4	5	6	7	8

```
i=2;
while (i<5) {
      aux = CRR[i];
      CRR[i] = CRR[8-i+1];
      CRR[8-i+1] = aux;
       i = i+1;
} /* fim-while */
aux = CRR[1];
CRR[i] = CRR[8];
CRR[8] = aux;
```

- Após a execução do trecho de código acima, qual a nova configuração de CRR?

CRR	A	C	E	R	T	O	U	!
	1	2	3	4	5	6	7	8

Discussão inicial (II)

- Outro ponto importante é a pesquisa de valores em um determinado vetor ou matriz.
- Vejamos um exemplo de busca sequencial de uma chave K num vetor desordenado. Se a chave K for encontrada nas primeiras posições, o programa apresenta bom desempenho. No entanto, o caso pior é quando a chave K se encontra na última posição do vetor. Portanto, em pesquisas sequenciais, podem ser feitas até N comparações para identificar se uma chave K está ou não presente no vetor. Esse caso fica mais evidente quando o vetor é de tamanho razoável (grande).
- Por outro lado, se o vetor estiver ordenado, a pesquisa pela chave K fica muito mais rápida, como poderá ser percebido nos exemplos adiante.
- Vejamos então três exercícios: *(i)* Busca de uma chave K num algoritmo desordenado, *(ii)* Ordenação de vetores (método da bolha), e *(iii)* Pesquisa binária em vetores ordenados.

Pesquisa sequencial em um vetor:

1) Elaborar um programa para encontrar a primeira ocorrência da chave k num vetor de 128 posições. Se não encontrar, imprimir que não encontrou.

```c
#include <stdio.h>
#define TAMANHO 128
#define FALSO 0
#define VERDADEIRO 1
main () {
   int vetor[TAMANHO], i;
   int achou_k;
   int k; // chave K a ser pesquisada.
   for (i=0; i<TAMANHO; i=i+1) {
       printf("Digite vetor[%d] : ", i);
       scanf("%d", &vetor[i]);
   } /* fim-for */
   printf("\nDigite o valor da chave K: ");
   scanf("%d", &k);
   achou_k = FALSO;
   i = 0;
   while (i<TAMANHO) {
       if (vetor[i] == k) {
          printf("A chave %d estah na posicao %d do vetor\n", k, i);
          achou_k = VERDADEIRO;
          i = TAMANHO;
       } /* fim-if */
       i++;
   } /* fim-while */
   if (achou_k == FALSO) {
       printf("A chave %d nao estah no vetor\n", k);
   } /* fim-se */
} /* fim-programa */
```

Essas são **constantes:** não alteram valor durante a execução do código.
São utilizadas para facilitar a compreensão e manutenção do programa.

Esse comando faz com que a variável de controle **i** ultrapasse o valor limite e saia do laço mais rapidamente.

Ordenação de vetores:

2) Elaborar um programa para ordenar um vetor de 128 posições em ordem crescente.

```c
#include <stdio.h>
#define TAMANHO 128
main () {
   int vetor[TAMANHO], i, aux;
   int limite_superior, bolha;

   for (i=0; i<TAMANHO; i=i+1) {
       printf("Digite vetor[%d] : ", i);
       scanf("%d", &vetor[i]);
   } /* fim-for */
   limite_superior = TAMANHO - 1;
   while (limite_superior > 0) {
       bolha = -1;
       for (i=0; i <= (limite_superior - 1); i=i+1) {
          printf("i = %d \n", i);
          if (vetor[i] > vetor[i+1]) {
             aux       = vetor[i];
             vetor[i]  = vetor[i+1];
             vetor[i+1]   = aux;
             bolha = i;
          } /* fim-if */
       } /* fim-for */
       limite_superior = bolha;
   } /* fim-while */
   for (i=0; i<TAMANHO; i=i+1) {
       printf("vetor[%d]=%d \n", i, vetor[i]);
   } /* fim-for */
} /* fim-programa */
```

Essa variável está igual a TAMANHO-1 já que o vetor vai de 0 a 127 posições.

Essa comparação vai até limite_superior -1 para garantir que não haverá comparações com posições fora do vetor.

Essa variável aponta para a posição onde houve a última troca. Se não houver troca essa variável sai do escopo do **for** com valor -1 e isso acelera a saída do laço **while** mais externo.

Pesquisa binária em vetores:

3) Elaborar um programa para pesquisar a chave K num vetor de 128 posições que está ordenado em ordem crescente.

```c
#include <stdio.h>
#define TAMANHO 128

main () {
   int comeco, fim, meio, k;
   int i, vetor[TAMANHO];

   for (i=0; i<TAMANHO; i=i+1) {
       printf("Digite vetor[%d] ordenado: ", i);
       scanf("%d", &vetor[i]);
   } /* fim-for */
   printf("\nDigite o valor da chave K: ");
   scanf("%d", &k);
   comeco = 0;
   fim = TAMANHO - 1;
   do {
       meio = (comeco + fim)/2;
       if (k < vetor[meio]) {
          fim = meio - 1;
       } else {
          comeco = meio + 1;
       } /* fim-if */
   } while ( (vetor[meio] != k) && (comeco <= fim) );
   if (vetor[meio] != k) {
       printf("Nao existe o elemento %d no vetor\n", k);
   } else {
       printf("A chave %d estah na posicao %d\n", k, meio);
   } /* fim-se */
} /* fim-programa */
```

O usuário deve digitar um vetor ordenado para o programa dar certo.

Estrutura de repetição com teste no fim do laço. A única diferença em relação ao while é que esse comando executa pelo menos uma vez, mesmo quando a condição não é satisfeita. No caso do **while** isso não ocorre.