INTRODUÇÃO À PROGRAMAÇÃO ESTRUTURADA EM LINGUAGEM C

Conteúdo desta aula:

- Revisão sobre tópicos da aula anterior e solução de dúvidas

- Vetores (arrays)

- Buscar informação no vetor

  -- Busca linear

  -- Busca binária

Referências:

Capítulo 10 da apostila

O que é um vetor (array)?

Um vetor permite associar um identificador a um conjunto de variáveis de um mesmo tipo

Exemplo: tipo_do_identificador nome_do_identificador[quantidade de elementos];

```
int main() {

  char palavra[25]; // palavra com até 25 letras (caracteres)

  int numeros[5] = { 7, 2, 1, 4, 10 }; // vetor inicializado com cinco números quaisquer

  return 0;
}
```

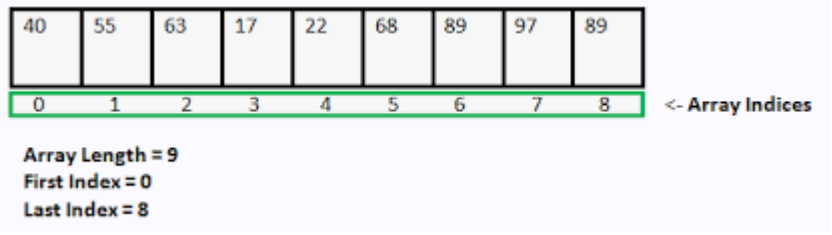
Vetor (array): Exemplo

```c
#include <stdio.h>
/* Programa: Melhores notas
   Recebe dez notas de alunos, calcula a média
   da sala e destaca as melhores notas
*/
int main() {
  float nota[10],media=0.0;
  int i;
  printf("Entre com as notas dos alunos\n");
  for (i=0; i<10; i++) {
    printf("Digite a nota %2d: ",i+1);
    scanf("%f", &nota[i]);
    media += nota[i];
  }
  media /= 10.0;
  printf("\nMedia da classe: %.2f\n\n",media);
  for (i=0; i<10; i++) {
    if (nota[i] > media)
      printf("Nota[%d]=%.2f\n",i+1,nota[i]);
  }
  return 0;
}
```

```
Run                                   Ask AI   26s

Entre com as notas dos alunos
Digite a nota  1: 5
Digite a nota  2: 6
Digite a nota  3: 2
Digite a nota  4: 8
Digite a nota  5: 4
Digite a nota  6: 7
Digite a nota  7: 7
Digite a nota  8: 9
Digite a nota  9: 0
Digite a nota 10: 1

Media da classe: 4.90

Nota[1]=5.00
Nota[2]=6.00
Nota[4]=8.00
Nota[6]=7.00
Nota[7]=7.00
Nota[8]=9.00
```

Vetor (array): Acesso aos elementos

O armazenamento e a leitura dos valores funciona de maneira similar a um identificador simples. Na memória, os elementos do vetor são armazenados um após o outro. O endereço de v[i] pode ser obtido pro &v[i] ou (v+i), já que v guarda o endereço de v[0] e, portanto, (v+i) guarda o endereço de v[i]. Ao somarmos (v+i), pulamos na memória i*sizeof(int) bytes a partir do endereço de v[0].



Vetor (array): Testando o acesso aos elementos

main.c > ... Format

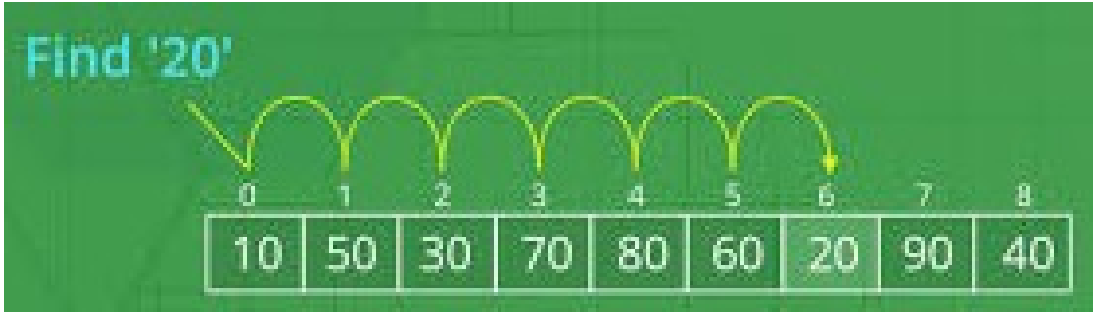
```c
#include <stdio.h>
// vetor com 5 posicoes do tipo inteiro preenchidos
int main()
{
int v[5]={7,2,1,4,10};
printf("Endereço de v[0]: %d = %d\n", &v[0], v);
printf("Conteudo de v[0]: %d = %d\n", v[0], *v);
printf("Conteudo de v[1]: %d = %d\n", v[1], *(v+1));
printf("Conteudo de v[4]: %d = %d\n", v[4], *(v+4));
printf("Endereço de v[4]: %d = %d\n", &v[4], (v+4));
return 0;
}
```

Run Ask AI

```
Endereço de v[0]: 875285104 = 875285104
Conteudo de v[0]: 7 = 7
Conteudo de v[1]: 2 = 2
Conteudo de v[4]: 10 = 10
Endereço de v[4]: 875285120 = 875285120
```

Vetor (array): Busca em vetores – Busca Linear (sequencial)

Um problema comum quando se manipula vetores é a necessidade de encontrar um elemento com um dado valor. Uma forma trivial de fazer este acesso é percorrer do índice inicial ao final todos os elementos do vetor até achar o elemento desejado. Esta forma de busca é chamada linear, pois no pior caso o número de comparações necessárias é igual ao número de elementos no vetor. No pior caso, em O(n).



INTRODUÇÃO À PROGRAMAÇÃO ESTRUTURADA EM LINGUAGEM C

Vetor (array):

Busca Linear

```c
#include <stdio.h>
/* Exemplo: efetua busca linear
   Verifica se existe uma nota X no vetor lido com 10 elementos
*/
int main() {
  float nota[11], x; /* vetor criado com uma posição a mais */
  // Verificando quantas posições o vetor nota possui
  int length = sizeof(nota) / sizeof(nota[0]);
  printf("O vetor NOTA possui %d posições\n\n", length);
  // indice auxiliar
  int i;
  // Lendo as notas
  printf("Entre com 10 notas\n");
  for (i=0; i<10; i++) {
    printf("Nota[%i]: ", i+1);
    scanf("%f", &nota[i]);
  }
  // Busca elemento no vetor
  while(1) {
    printf("\nDigite a nota procurada ou -1 para sair do programa:
");
    scanf("%f", &x);
    if (x==-1)
      break;
    /* busca linear */
    nota[10]=x; /* elemento sentinela */
    i=0;
    while(nota[i] != x) /* busca com sentinela */
      i++;
    if (i <10)
      printf("Nota %.2f encontrada na posição %d\n", nota[i], i);
    else
      printf("Nota %.2f não encontrada\n", x);
  }
  return 0;
}
```

```
O vetor NOTA possui 11 posições

Entre com 10 notas
Nota[1]: 1
Nota[2]: 2
Nota[3]: 3
Nota[4]: 4
Nota[5]: 5
Nota[6]: 6
Nota[7]: 7
Nota[8]: 8
Nota[9]: 9
Nota[10]: 10

Digite a nota procurada ou -1 para sair do programa: 9.3
Nota 9.30 não encontrada

Digite a nota procurada ou -1 para sair do programa: 10
Nota 10.00 encontrada na posição 9

Digite a nota procurada ou -1 para sair do programa: -1
```

Vetor (array): Busca em vetores – Busca Binária

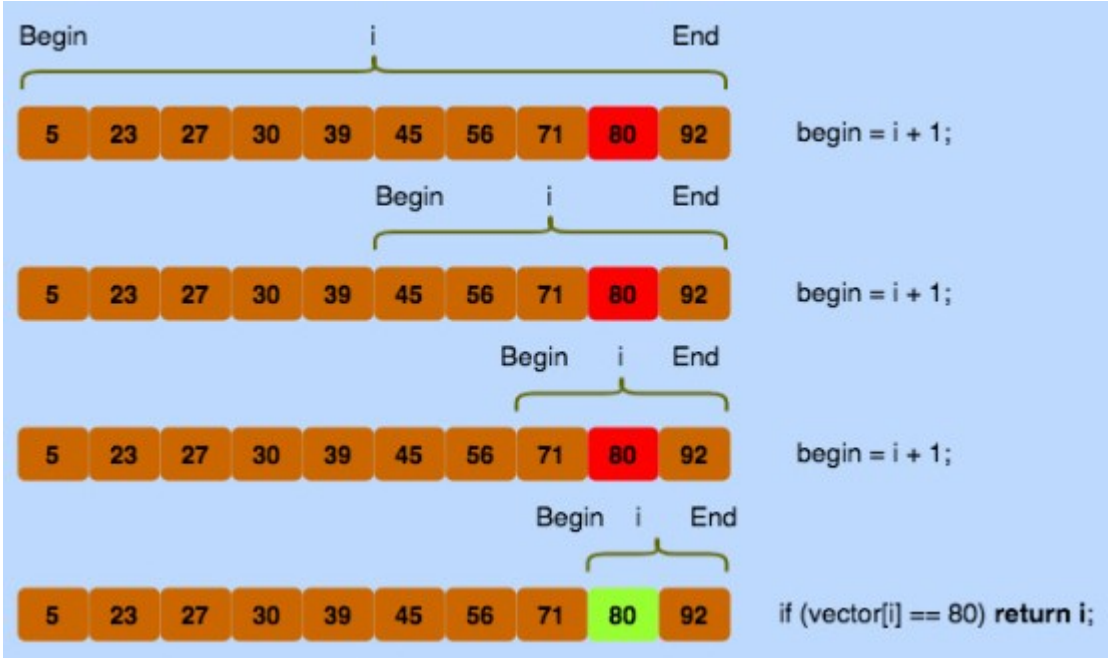
A busca binária reduz o número de comparações de n para $\log_2(n)$ no pior caso, onde n é o tamanho do vetor.

Considerando um vetor com 1024 posições ($1024 = 2^{10}$) notaremos que no pior caso serão realizadas dez comparações. No entanto, a busca binária precisa que o vetor esteja ordenado. Vale lembrar que essa ordenação também tem um custo a ser considerado, mas caso sejam feiras várias buscas, este custo computacional certamente pode valer a pena. Temos o tempo de execução em O(log n).

A ideia básica é que a cada iteração do algoritmo, podemos eliminar a metade dos elementos no processo de busca.

Vetor (array):

Busca Binária



Vetor (array):

Busca Binária

```c
#include <stdio.h>
typedef enum {false,true} bool;

int main() {
  float nota[10],x;
  int i,pos,inicio,fim;
  bool achou;

  printf("Entre com 10 notas em ordem crescente\n");
  for (i=0; i<10; i++) {
    printf("Nota[%d]: ",i+1);
    scanf("%f", &nota[i]);
  }
  while(1) {
    printf("\nDigite a nota procurada ou -1 para sair do programa: ");
    scanf("%f", &x);
    if (x==-1.0)
      break;
    /* busca binária */
    inicio=0;
    fim=9;
    achou=false;
    while((inicio<=fim)&&(!achou)) {
      pos=(inicio+fim)/2;
      if(x < nota[pos])
        fim=pos-1;
      else
        if(x > nota[pos])
          inicio=pos+1;
        else
          achou=true;
    }
    if(achou)
      printf("Nota %.2f encontrada na posição %d\n", nota[pos],pos);
    else
      printf("Nota %.2f nao encontrada\n",x);
  }
  return 0;
}
```

```
Entre com 10 notas em ordem crescente
Nota[1]: 1
Nota[2]: 2
Nota[3]: 3
Nota[4]: 4
Nota[5]: 5
Nota[6]: 6
Nota[7]: 7
Nota[8]: 8
Nota[9]: 9
Nota[10]: 10

Digite a nota procurada ou -1 para sair do programa: 8.3
Nota 8.30 nao encontrada

Digite a nota procurada ou -1 para sair do programa: 9
Nota 9.00 encontrada na posição 8

Digite a nota procurada ou -1 para sair do programa: -1
```

Exercícios:

10.1. Faça um programa que permita a um usuário digitar e armazenar dois vetores com 10 números inteiros em cada. Depois de digitado todos os valores, calcular a soma dos elementos de mesma posição dos vetores (elemento a elemento) armazenando em um terceiro vetor. Exibir no monitor os elementos do terceiro vetor.

10.2. Faça um programa que leia 20 números reais e permita que o usuário decida qual operação realizar: soma, subtração, multiplicação ou divisão. O vetor resultante de ser R = A <operação> B, onde os valores lidos para o segundo vetor devem ser diferentes de zero.

10.3. Com base no exercício 10.2, faça um programa, onde dado um valor digitado pelo usuário, pesquise no terceiro vetor (R), utilizando busca linear.

10.4. Com base no exercício 10.2, faça um programa, onde dado um valor digitado pelo usuário, pesquise no terceiro vetor (R), utilizando busca binária.