Introdução à Programação

1.Ano LCC-MIERSI
DCC - FCUP

Nelma Moreira

Aula 8

Manipulação de valores guardados numa variável indexada

* Contar quantos dos valores guardados são múltiplos de um inteiro não nulo m, para vários valores de m lidos sucessivamente
* Verificar se a sequência de valores guardados se encontra ordenada por ordem não decrescente

```c
#include <stdio.h>
#define MAXN 100
int main() {
  int x[MAXN], i, n, m, conta;
  scanf("%d",&n);
  for(i=0; i<n; i++)
    scanf("%d",&x[i]);
  scanf("%d",&m);
  while(m != 0) {
    for(conta=0,i=0; i < n; i++) if (x[i]%m == 0) conta++;
    printf("Numero de multiplos de %d = %d\n",m,conta);
    scanf("%d",&m);
  }
  return 0;
}
```

```c
#include <stdio.h>
#define MAXN 100
int main() {
  int x[MAXN], i, n, m, conta;
  scanf("%d",&n);
  for(i=0; i<n; i++)
    scanf("%d",&x[i]);
  for(i=0; i < n-1; i++)
    if (x[i] > x[i+1]) break;
  if (i >= n-1)    // i == n-1 se n nao for zero
    printf("Ordenada por ordem nao decrescente\n");
  else
    printf("Nao ordenada por ordem nao decrescente\n");
  return 0;
}
```

Torneios

* Analisar um conjunto de sequências de inteiros não negativos, todas com o mesmo número de elementos, e identificar aquelas cuja soma dos elementos é máxima.
* Modelo de um jogo em que cada sequência representa as pontuações de um jogador nas várias partidas e se pretende determinar os jogadores vencedores.

* Número sequências, `njogadores`
* Tamanho de cada sequência, `njogos`
* Cada valor, `pontos`
* Maior soma, `maxpontos`
* Para cada sequência:
  * determinar a soma, `soma`
  * comparar com o máximo
* Se só se quisser saber o máximo (os pontos dos vencedores) não é necessário usar variáveis indexadas.

```c
#include <stdio.h>
int main() {
  int njogadores, njogos, maxpontos, soma, pontos;
  int i, j;
  maxpontos = -1;
  scanf("%d %d",&njogadores,&njogos);
  for(i=1; i <= njogadores; i++) {
    // determina total de pontos do jogador i
    for(soma=0, j = 0; j < njogos; j++) {
      scanf("%d",&pontos);
      soma += pontos;
    }
    // compara com melhor encontrado antes
    if (soma > maxpontos)
      maxpontos = soma;
  }

  // escreve resultado
  printf("Pontos do primeiro lugar: %d\n",maxpontos);

  return 0;
}
```

* Mas se se quisser saber quais são os vencedores, é necessário saber as sequencias que atingiram a soma máxima:

  * `int vencedores[]` indica quais os vencedores

  * `nvencedores` quantos os vencedores

  * `MAXJ` máximo de sequencias/jogadores

```c
#include <stdio.h>
#define MAXJ 200
int main() {
  int njogadores, njogos, maxpontos, soma, pontos;
  int nvencedores, i, j, vencedores[MAXJ];
  maxpontos = -1;
  nvencedores = 0;
  scanf("%d %d",&njogadores,&njogos);
  for(i=1; i <= njogadores; i++) {
     for(soma=0, j = 0; j < njogos; j++) {
      scanf("%d",&pontos);
      soma += pontos;
     }
     // determina lugar de i face aos analisados antes
     if (soma > maxpontos) {
       maxpontos = soma;
       vencedores[0] = i;  // identifica vencedor
       nvencedores = 1;
     } else if (soma == maxpontos) {
       vencedores[nvencedores] = i; // jogador empatado
       nvencedores++;
     }
}
```

```
  // escreve resultado

  printf("Pontos do primeiro lugar: %d\n",maxpontos);
  if (nvencedores == 1)
    printf("O vencedor foi o concorrente %d\n", vencedores[0]);
  else {
    printf("Os vencedores foram os concorrentes:\n");
    for(i=0; i < nvencedores; i++)
      printf("%d\n",vencedores[i]);
  }

  return 0;
}
```

Manipulação dos elementos duma variável indexada

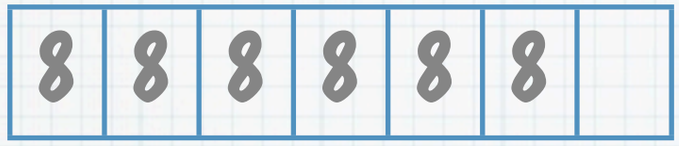
* Deslocar os `n-1` primeiros elementos, de uma variável indexada `a[]`, uma posição para a direita (supondo que se reservou espaço).
* Exemplo:

a | 8 | 5 | 6 | 1 | 3 | | | para

a | 8 | 8 | 5 | 6 | 1 | 3 | |

* Primeira tentativa:

```
for(i = 0; i < n - 2;i++) a[i+1] = a[i];
```



* Resultado: ERRADO!!

* Resolução correcta:

```
for(i = n - 2;i > 0;i--) a[i+1] = a[i]
```

Aplicação

* Inserir um elemento `x` na posição `j < n`, deslocando os elementos para a direita... se estiver reservado espaço.
* Por exemplo, se

8	5	6	1	3		

* e `x = 12` e `j = 3` deve resultar:

8	5	6	12	1	3	

```
for(i = n - 1; i >= j; i--)
        a[i+1] = a[i];
a[j] = x;
```

Inversão da ordem dos elementos

* Trocar simetricamente os `n` elementos de uma variável indexada `a[]`, isto é, `a[0]` troca com `a[n-1]`, `a[1]` troca com `a[n-2]`, etc. Por exemplo,

8	5	6	1	3		

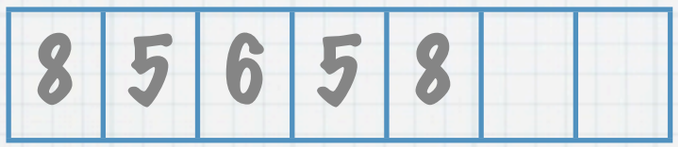
deve ficar

3	1	6	5	8		

* Ideia: para `i=0..n-1` trocar `a[n-1-i]` com `a[i]`. Como?
* Primeira tentativa:

```
for(i = 0;i < n;i++) {
    a[n-1-i] = a[i];
    a[i] = a[n-1-i];
}
```

* Resultado: ERRADO!! Faz duas passagens



* Correcto: com uma variável auxiliar

```
for(i = 0;i < n/2; i++) {
    aux = a[n-1-i];
    a[n-1-i] = a[i];
    a[i] = aux;}
```

Intervalo

5 minutos

Operações sobre conjuntos

* As variáveis indexadas representam conjuntos finitos de dados em que cada elemento tem uma posição determinada e pode ocorrer mais que uma vez.
* No entanto podem-se definir algoritmos para implementar as operações básicas sobre conjuntos:
  * um elemento pertence a um conjunto
  * reunião, intersecção, diferença de conjunto
  * a relação estar contido

Pesquisa

* Procurar se um valor `x` existe numa variável indexada `a[]`:

  `a[0],..., a[n-1].`

* A resposta deve ser:

  * Um índice `i` tal que `x = a[i]`.

  * `-1` se não houver nenhum `i` nessas condições.

i	0	1	2	3	4	5	6
a[i]	8	5	6	1	3	9	4

* Se `x = 9` a resposta deve ser `5` pois `a[5] = 9`.

* Quais deveriam ser as respostas para os seguintes valores de `x`: 0, 8 e 1?

Pesquisa Linear

* Algoritmo:

Para `i = 0, 1,...,n - 1` comparamos `a[i]` com `x`. Se forem iguais, o resultado é `i`. Se nunca forem iguais, o resultado é `-1`.

```
#include<stdio.h>
int main(){
 int x, pos;
 int a[]={8,5,6,1,3,9,4},
        i, n = 7;
 scanf("%d",&x);
 for(i = 0;i < n;i++)
    if(a[i] == x) break;
 if (i < n) pos = i;
 else pos = -1;
 if(pos >= 0)
   printf("Posicao = %2d
\n",pos);
 else
   printf("Nao ocorre em
a[]\n");
return 0;
}
```

Execução:

```
$proc
3
Posicao =  4
$proc
4
Posicao =  6
$proc
12
Nao ocorre em a[]
```

- Alterações ao programa anterior:
  - Usando a instrução de ciclo `while` sem utilizar `break`. Uma solução:

```
i=0;  while(i<n && a[i]!=x) i++;
```

    Pela semântica do `&&`, `a[n] != x` não é executado
  - Melhorar a eficiência do programa anterior, colocando `x` em `a[n]`, supondo-se que esta posição existe. Uma solução:

```
 i = 1; a[n] = x;
while(a[i] != x) i++; /* termina sempre! */
pos = i<n? i: -1;
```

Pesquisa e Ordenação

* Se os valores da sequência estiverem ordenados a pesquisa pode ser efectuada de forma mais eficiente.
* Como se procura um número de telefone numa lista telefónica (ordenada por nome)? E se não estivesse ordenada?
  * O que acontece numa pesquisa linear? Seja `x=30`.

    `-1  8  14  25  45  78  90`
  * Não é necessário continuar se `a[i] > x`
* Mas se `n` grande e/ou o `x` não estiver lá, tem de se percorrer os `n` elementos

- Vamos supor os valores ordenados por ordem crescente.
- Suponhamos `v[0],..., v[n-1]` uma sequência ordenada. Comparar `x` com o elemento do meio da sequência: seja `m=(n-1)/2` o seu índice:
  - Se `x = v[m]`, eureka, a resposta é `m`.
  - Se `x < v[m]`, o valor `x` só pode estar nos índices compreendidos entre `0` e `m-1`.
  - Se `x > v[m]`, o valor `x` só pode estar nos índices compreendidos entre `m+1` e `n-1`.
  - Só com uma comparação ou encontramos `x` ou eliminamos cerca de `n/2` elementos!

Pesquisa binária

Algoritmo

* Sejam em cada passo `a` e `b` os índices entre os quais pode estar `x`
  * Inicialmente `a=0` e `b=n-1`
  * Se `a > b` o intervalo é vazio, `x` não está em `v[]`
  * Seja `m = (a + b)/2`. Ao comparar `x` com `v[m]`
    * Se `x < v[m]`, passa `b` a `m - 1` (`a` mantém-se)
    * Se `x > v[m]`, `a` passa a `m+1` (`b` mantém-se)
    * Se `x == v[m]`, o índice foi encontrado: `m`.

* Vamos escrever uma função que retorna o índice de `x` ou `-1`, caso `x` não ocorra na variável indexada.

```
int v[]={-1,2,4,6,7,9,15};
int n=7;
int pb(int x, int a, int b)
{
  int m;
  while(a <= b) {
   m = (a + b) / 2;
   if (x == v[m]) return m;
   if (x < v[m]) b = m - 1;
   else a = m + 1;
  }
     return -1;
}

int main() {
 int  x;
 scanf("%d",&x);
 printf("%d", pb(x,0,n-1));
 return 0;
}
```

i	0	1	2	3	4	5	6
v[i]	-1	2	4	6	7	9	15

* Procurar x=8:

```
a=0, b=6, m=3
a=4, b=6, m=5
a=4, b=4, m=4
-1
```

* Procurar x=15:

```
a=0, b=6, m=3
a=4, b=6, m=5
a=6, b=6, m=6
6
```

* Procurar x=0:

```
a=0, b=6, m=3
a=0, b=2, m=1
a=0, b=0, m=0
-1
```

Número máximo de iterações: log_2(n)