Departamento de Matemática — Universidade de Coimbra

Ano Lectivo de 2015/2016 Métodos de Programação I Frequência 20/5/2016

Leia atentamente o enunciado de cada pergunta antes de iniciar a sua resolução.

Duração da prova: 1h30min Os programas devem ser escritos em *C*

---

1. (a) Na linguagem de programação *C*, o que é se entende por avaliação inteligente, *lazy evaluation*, de expressões lógicas.

   (b) Considere o seguinte problema: dado um vector de inteiros quer-se somar os elementos do vector até que: para um dado par de elementos consecutivos se tenha $v_i > v_{i+1}$, ou se atinja o fim do vector.

   Escreva o excerto do programa referente ao ciclo de soma dos elementos.

2. **Torre de Hanói** é um «quebra-cabeças» em que se têm três pinos e n discos que encaixam nos pinos. Os discos estão num dos pinos uns sobre os outros, em ordem crescente de diâmetro.

   Pretende-se mover n discos de um dado pino de origem, para um dado pino de destino, tendo um outro pino como auxiliar.

   É necessário cumprir duas regras simples: só se pode mover um disco de cada vez; um disco nunca pode ficar por cima de um de menor diâmetro.

   Escreva uma função recursiva que, dado um n (número de discos) e três caracteres (nomes dos pinos), escreva a sequência de passos necessários para a resolução do problema. Algo como:

   ```
   Move um disco do pino A para o pino B
   Move um disco do pino A para o pino C (...)
   ```

   escreva a respectiva instrução de chamada.

3. Construa um sub-programa que, dado um vector de elementos inteiros inverta a ordem dos seus elementos, sem usar elementos auxiliares para efectuar a inversão.

4. Dadas dois conjuntos (ordenáveis) A e B, a ordem lexicográfica sobre o produto cartesiano $A \times B$ é definida como:

   $$(a, b) \leq (a', b') \quad \text{se e somente se} \quad a < a' \vee (a = a' \wedge b \leq b').$$

   Dado a seguinte estruturas de dados:

   ```
   typedef struct par {
     int peso;
     float elem;
   } Pares;
   ```

   Construa um sub-programa (e eventualmente outros auxiliares) para ordenar um vectores de pares, `Pares v[100];`

Resolução

1. Lazy Evaluation

   (a) É de notar que o "C" adopta um avaliação inteligente, "lazy evaluation", das expressões lógicas de forma a evitar efectuar cálculos desnecessários, ou seja: a avaliação de uma conjunção pára assim que uma das proposições tenha o valor de verdade "Falso". De forma correspondente a avaliação de uma disjunção pára assim que o valor de uma das proposições tenha o valor de verdade "Verdade". Isto é a avaliação de uma proposição pára assim que o seu valor final é conhecido por se ter obtido o valor absorvente da conectiva lógica que se está a considerar.

   (b)

```
soma = 0;
do {
  soma=soma+v[i];
} while (i<dim && v[i] <= v[i+1])
```

2. Torre de Hanói

```
void hanoi(int n,char inicio, char fim, char auxiliar){
  if (n==1) {
    printf("Move um disco do pino %c para o pino %c\n",inicio,fim);
  }
  else {
    hanoi(n-1,inicio,auxiliar,fim);
    printf("Move um disco do pino %c para o pino %c\n",inicio,fim);
    hanoi(n-1,auxiliar,fim,inicio);
  }
};

hanoi(numeroDiscos,'A','B','C');
```

3. Inversão de um vector.

```
void inverte(int v[], int dim) {
  int i,meio;

  meio = dim/2;
  for (i=0;i<meio;i++) {
    v[i] = v[i]*v[dim-1-i];
    v[dim-1-i] = v[i]/v[dim-1-i];
    v[i] = v[i]/v[dim-1-i];
  }
}
```

4. Ordem Lexicográfica

```
int borbulhagem(int dim, Pares v[]) {
  int i;
  Pares aux;
  int  ordenado ;

  do {
    ordenado = 1;
    for (i=0; i<dim-1 ; i++) {
      if (v[i].peso > v[i+1].peso ||
          (v[i].peso == v[i+1].peso && v[i].elem > v[i+1].elem)) {
        aux = v[i];
        v[i] = v[i+1];
        v[i+1] = aux ;
        ordenado = 0;
      }
    }
  } while (!ordenado) ; // repete até não haver mais trocas
  return(0);
}
```