Análise e Desenvolvimento de Algoritmos (2006/2007)

Alguns desenvolvimentos em série de Taylor:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots, \ \forall x \in \mathbb{R}$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots, \ \forall x \in \mathbb{R}$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots, \ \forall x \in \mathbb{R}$$

$$\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots, \ -1 < x \leq 1$$

$$a^x = e^{x \ln a} = 1 + x \ln a + \frac{(x \ln a)^2}{2!} + \frac{(x \ln a)^3}{3!} + \cdots + \frac{(x \ln a)^n}{n!} + \cdots$$

$$\sinh x = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \cdots$$

$$\cosh x = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \cdots$$

Elabore Algoritmos para:

1. Calcular o valor de uma Soma Parcial do desenvolvimento em Série de Taylor da Função Exponencial para um dado $x \in \mathbb{R}$, de modo a que o valor absoluto do último termo somado seja inferior a um dado $\varepsilon \in \mathbb{R}^+$.

2. Calcular o valor de uma Soma Parcial do desenvolvimento em Série de Taylor da Função Exponencial para um dado $x \in \mathbb{R}$, de modo a que o valor absoluto do primeiro termo desprezado seja inferior a um dado $\varepsilon \in \mathbb{R}^+$.

3. Determinar quantos termos do desenvolvimento em Série de Taylor são necessários para calcular a Exponencial para um dado $x \in \mathbb{R}$, de modo a que o valor absoluto do primeiro termo desprezado seja inferior a um dado $\varepsilon \in \mathbb{R}^+$.

4. Calcular o valor de uma Soma Parcial do desenvolvimento em Série de Taylor da Função Seno para um dado $x \in \mathbb{R}$, de modo a que o valor absoluto do último termo somado seja inferior a um dado $\varepsilon \in \mathbb{R}^+$.

5. Modifique o Algoritmo do problema 1. de forma a que, com um só ciclo, sejam calculados o Seno e também o Cosseno.

6. Teste o programa anterior para valores elevados de x.
   O que acontece? Como resolver o problema?

Utilizando a Axiomática de Hoare,Verifique formalmente a correcção parcial dos seguintes algoritmos:

1. $\{x \in \mathbb{N}_0 \ \wedge \ y \in \mathbb{N}\}$

```
q := 0; r := x;
while r >= y  do
      begin r := r - y;
            q := q + 1
      end;
```

$\{x = q \times y + r \ \wedge \ 0 \leq r < y\}$

2. $\{n \in \mathbb{N}_0 \ \wedge \ x \in \mathbb{R}^+\}$

```
k := 0; y := 1;
while k < n  do
      begin k := k + 1;
            y := y * x
      end;
```

$\{y = x^n\}$

3. Repita o problema anterior, mas decrementando o contador k.

4. Repita o mesmo problema, mas sem o contador k.

5. O contador k calcula quantas vezes a divide b:

$\{a, b \in \mathbb{N}\}$

```
c := b; k:= 0;
divide := c mod a = 0;
while divide do
      begin k:= k + 1;
            c:= c div a;
            divide := c mod a = 0
      end;
```

$\{b = a^k * c \ \wedge \ c \, mod \, a \neq 0\}$

6. $\{a, b \in \mathbb{N}\}$

```
x := 0; c := 0;
while c < a do
      begin c := c + 1;
            x := x + b;
      end;
```

$\{x = a \times b\}$

7. $\{a, b \in \mathbb{N}\}$

```
x := 0; c := a;
while c > 0 do
      begin c := c - 1;
            x := x + b;
      end;
```

$\{x = a \times b\}$

8. $\{a, b \in \mathbb{N}\}$

```
x := a; y := b; z := 0;
while x <> 0 do
      begin if odd(x)
            then z := z + y;
            x := x div 2;
            y := 2 * y
      end;
```

$\{z = a \times b\}$

9. $\{n \in \mathbb{N}_0 \ \wedge \ x \in \mathbb{R}^+ \ \wedge \ n = n_0\}$

```
p := 1; f := x;
while n <> 0 do
      begin if odd(n)
            then p := p * f;
            n := n div 2;
            f := f * f
      end;
```

$\{p = x^{n_0}\}$

10. Máximo Divisor Comum e Menor Múltiplo Comum

(a) Na sua forma original, o Algoritmo de Euclides era descrito:

> *Para calcular o máximo divisor comum de dois números inteiros e positivos, vá subtraindo o que for menor ao que for maior, até ficarem iguais.*

(b) Em 10 de Fevereiro de 1975, Edsger W. Dijkstra colocou a seguinte questão:

$\{a, b \in \mathbb{N}\}$

```
c:= a; d:= b;
x:= b; y:= a;
while   c <> d  do
        if c > d
        then begin c:= c - d;
                   y:= y + x
             end
        else begin d:= d - c;
                   x:= x + y
             end;
```

$\{\ mdc(a,b) = (c+d)\ div\ 2 \ \wedge\ mmc(a,b) = (x+y)\ div\ 2\ \}$

11. Seja `a[1..n]` um vector cujos elementos estão dispostos por ordem não-decrescente.

    (a) Construa um procedimento para a **Inserção Ordenada** de um `novo` elemento no vector. O algoritmo utilizado deve conter apenas um ciclo.

    (b) Calcule o número de comparações (envolvendo elementos do vector) realizadas por esse algoritmo nos seus casos: Melhor, Pior e Médio.

    (c) Utilizando o Algoritmo anterior, construa um procedimento para o **Ordenamento por Inserção Sequencial** de um dado vector, com base na seguinte Estratégia:

        Se um sub-vector (primeiros elementos) já estiver ordenado, inserir ordenadamente o elemento seguinte nesse sub-vector.

    (d) Faça a análise do número de comparações (envolvendo elementos do vector) realizadas por esse algoritmo.

12. Elabore sub-programas Recorrentes para escrever:

(a) A Palavra $a^n b^n$ com $n \in \mathbb{N}$.
Exemplo para $n = 4$,

```
aaaabbbb
```

(b) Um Triângulo de asteriscos, de dimensão $n \in \mathbb{N}$.
Exemplo para $n = 4$,

```
   *
  ***
 *****
*******
```

(c) Um Losango de asteriscos, de dimensão $n \in \mathbb{N}$.
Exemplo para $n = 3$,

```
  *
 ***
*****
 ***
  *
```

(d) Um Triângulo de números, de dimensão $n \in \mathbb{N}$.
Exemplo para $n = 5$,

```
    1
   121
  12321
 1234321
123454321
```

(e) Um Losango de números, de dimensão $n \in \mathbb{N}$.
Exemplo para $n = 3$,

```
  1
 121
12321
 121
  1
```

13. Pretendemos determinar o índice do maior elemento de um vector $x[1..n]$ de elementos inteiros.

    (a) Escreva o algoritmo iterativo clássico e diga qual o número de comparações efectuadas com elementos do vector.

    (b) Elabore a versão recorrente do algoritmo anterior e calcule o número de comparações efectuadas.

    (c) Construa uma versão recorrente baseada em sucessivas partições binárias do vector e consequente pesquisa em ambas as partes. Calcule também o número de comparações.

    (d) Escreva um algoritmo iterativo para a determinação conjunta dos índices dos elementos máximo e mínimo do vector, calculando também o número de comparações realizadas.

    (e) Para o mesmo efeito e utilizando a estratégia de **Separação** de C. A. R. Hoare construa um algoritmo que, começando por separar os elementos *grandes* dos elementos *pequenos*, pesquisa sequencialmente o máximo de entre os *grandes* e o mínimo de entre os *pequenos*. Calcule o número de comparações.

    (f) Elabore um algoritmo recorrente para **colocar** o valor máximo no último elemento do vector e calcule também o número de comparações.

    (g) Generalize o problema anterior, por forma a colocar o *k-ésimo* menor elemento na posição de ordem k do vector.

14. A estratégia utilizada em cada passo do Método de Ordenamento habitualmente conhecido por **Selecção Linear**, consiste basicamente na pesquisa do elemento máximo do (sub)vector e consequente troca com o último.

    (a) Construa um procedimento iterativo para ordenar um dado vector $x[1..n]$, por Selecção Linear.

    (b) Calcule o número de comparações, envolvendo elementos do vector, realizadas por este algoritmo.

    (c) Escreva agora um procedimento recorrente para o mesmo efeito.

    (d) Faça a análise do número de comparações, envolvendo elementos do vector, realizadas pelo algoritmo recorrente.

    (e) Um possível melhoramento deste método consiste na pesquisa conjunta do máximo e do mínimo, seguida das correspondentes trocas. Elabore um procedimento recorrente baseado nesta ideia.

    Nota: Existe um caso particular de trocas, que deve ser ponderado ...

    (f) Por fim calcule as comparações, envolvendo elementos do vector, realizadas por esta versão.

15. Considere o tipo Lista Ligada Linear, definido por:

```
type ponteiro = ^elemento;
     elemento = record num : integer;
                       prox : ponteiro
                end;
```

e um Ficheiro `numeros.txt` que contém números inteiros, um por cada linha. Construa os procedimentos:

(a) `Registar`: para registar no Ficheiro os números contido numa Lista, pela mesma ordem.

(b) `RegistarR`: para registar no Ficheiro os números contido numa Lista, por ordem inversa.

(c) `RecuperarR`: para construir uma Lista com os números registados no Ficheiro, por ordem inversa.

(d) `Recuperar`: para construir uma Lista com os números registados no Ficheiro, pela mesma ordem.

(e) `DarMax`: que devolve os ponteiros `pmax` para o nó que contém o elemento máximo da Lista e `pantmax` para o nó anterior.

(f) `RemoverMax`: remove e destrói o nó com o elemento máximo da Lista, numa só passagem e sem utilizar o procedimento anterior.

(g) `RemoverDarMax`: remove e devolve o ponteiro `pmax` para o nó com o elemento máximo da Lista.

(h) `OrdenarLista`: utilizando o procedimento anterior ordenar a Lista dada, sem criar espaço de memória adicional.

16. São bem conhecidas as limitações do tipo `integer` quanto à grandeza dos seus valores. Para permitir a manipulação de números inteiros de *grandeza astronómica*, podemos utilizar uma representação na forma de Lista Ligada Linear,

```
type tipodigito = 0..9;
     astronomico = ^elemento;
     elemento = record digito : tipodigito;
                       prox : astronomico
                end;
```

onde o primeiro elemento da lista contém o algarismo das unidades do inteiro representado.

Comecemos por construir os módulos:

(a) `function representacao(n : integer) : astronomico;`

(b) `procedure LereConstruir(var a : astronomico);`

(c) `procedure Escrever(a : astronomico);`

(d) `function soma(a, b : astronomico) : astronomico;`

(e) `function diferenca(a, b : astronomico) : astronomico;`

(f) `function iguais(a, b : astronomico) : boolean;`

(g) `function maior(a, b : astronomico) : boolean;`

(h) `function copia(a : astronomico) : astronomico;`

(i) `function zero : astronomico;`

(j) ...