UNIVERSIDADE DE AVEIRO

Análise e Desenvolvimento de Algoritmos

2004/2005

Folha 1

1. Escreva uma instrução de atribuição para cada uma das seguintes acções:

   (a) O contador c é incrementado de uma unidade;

   (b) a é uma cópia do valor de b;

   (c) q é o valor da divisão inteira dos inteiros i e j;

   (d) x é o valor da divisão real dos inteiros i e j;

   (e) i é o valor arredondado do real x;

   (f) i é o maior inteiro inferior ou igual a x, positivo;

   (g) m é o inteiro mais próximo da média dos reais $r1$ e $r2$;

   (h) A variável $tan20$ toma o valor da tangente de 20 graus;

   (i) Dado n, inteiro não nulo, a variável inteira *sinal* toma o valor 1 se n for positivo e -1 se n for negativo;

   (j) u toma o valor do algarismo das unidades do real x;

   (k) y toma o valor de $\sqrt{x}$ se x for não negativo, ou de $\sqrt{-x}$, caso contrário;

   (l) p toma o valor de módulo de y elevado a x;

   (m) $alfa$ é o ângulo (em graus) cuja tangente é x;

   (n) $y \leftarrow y + 4x + 3x^2 + 2x^3 + x^4$ (procure minimizar o numero de operações);

   (o) A área de um polígono regular de n lados de comprimento b é dada por:

   $$area \leftarrow \frac{1}{4} n b^2 \text{cotg} \frac{\pi}{n}$$

   (p) A variável lógica L é verdadeira se e só se $L1$ e $L2$ forem ambas falsas;

   (q) A variável lógica L é verdadeira se e só se $L1$ é verdadeira mas não $L2$;

   (r) $BOOL$ é verdadeira se e só se os inteiros A e B forem iguais em valor absoluto;

   (s) A variável $VOGAL$ é verdadeira se e só se a variável caracter $LETRA$ for uma vogal minúscula;

   (t) A variável $MINUSCULA$ é verdadeira se e só se a variável caracter $LETRA$ for uma letra minúscula;

   (u) A variável lógica $BISSEXTO$ é verdadeira se e só se a variável inteira ANO for divisível por 4 mas não por 100 ou então por 400;

(v) A variável lógica $MULT$ é verdadeira se e só se C for múltiplo de D (ambos inteiros);

(w) Sendo C uma variável do tipo caracter, o booleano $DIGITO$ é verdadeiro se e só se C representar um algarismo decimal;

(x) A variável inteira PAR vale 1 se n for par e 2 se n for ímpar;

(y) A variável inteira $ALTER$ toma o valor de $(-1)^n$ com n inteiro;

(z) Sendo c um caracter que representa um dígito, atribuir a N o valor numérico desse dígito (N pertence a $\{0, \ldots, 9\}$).

2. Desenvolva programas em Pascal para:

   (a) Converter euros e cêntimos em escudos e reciprocamente.

   (b) Converter graus em radianos e reciprocamente.

3. Desenvolva um programa em Pascal que:

   - Leia um número inteiro;
   - Calcule $n!$;
   - Escreva no ecrã o valor calculado.

4. Desenvolva um programa em Pascal que:

   - Leia um número inteiro $0 < n \leq 20$;
   - Escreva no ecrã os números $0, 1, \ldots, n$ (numa única linha);
   - Escreva no ecrã os números $n, n-1, \ldots, 1, 0$ (numa outra linha).

5. Desenvolva um programa em Pascal que:

   - Leia um número inteiro $0 < n \leq 20$;
   - Escreva no ecrã n linhas iguais a $1, \ldots, n$ (ou seja os elementos da matriz $[a_{i,j}]_{n \times n}$, em que $a_{i,j} = j$).

6. Modifique o programa anterior de modo que sejam apresentados no ecrã os elementos da matriz:

   (a) $[a_{i,j}]_{n \times n}$, em que $a_{i,j} = i + j$.

   (b) identidade de ordem n.

   (c) $[a_{i,j}]_{n \times n}$, em que $a_{i,j} = \begin{cases} 1 & \text{se } i \text{ e } j \text{ são pares} \\ -1 & \text{se } i \text{ e } j \text{ são ímpares} \\ 0 & \text{caso contrário.} \end{cases}$

UNIVERSIDADE DE AVEIRO

Análise e Desenvolvimento de Algoritmos

2004/2005

Folha 2

1. Desenvolva programas em Pascal que realizem as seguintes funções:

   (a) Ler dois números inteiros positivos e escrever no écran o maior.

   (b) Ler uma sequência de números inteiros positivos e escrever no écran o maior (a sequência termina com um número não positivo).

   (c) Calcular e escrever a soma de uma sequência de números inteiros pares dados pelo utilizador (a sequência termina com um número ímpar).

   (d) Calcular e escrever a média de uma sequência de números inteiros não nulos dados pelo utilizador (a sequência termina com o zero).

2. Escreva uma função em Pascal que calcule o valor de e^x por desenvolvimento em série, até a parcela somada ser, em valor absoluto, inferior a um certo valor positivo dado pelo utilizador.

3. Escreva uma função em Pascal que calcule o valor de $cos(x)$, desprezando parcelas de valor absoluto inferior a 10^{-6}, sabendo que:

$$cos(x) = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

4. Implemente em Pascal o Método da Bissecção.

5. O algoritmo de Euclides para o cálculo do maior divisor comum de dois números inteiros positivos, baseia-se na seguinte propriedade:

$$\text{mdc}(a,b) = \begin{cases} a & \text{se } b = 0 \\ \text{mdc}(b, a \bmod b) & \text{se } b \neq 0 \end{cases}$$

   Elabore um programa utilizando os seguintes passos:

   - peça ao utilizador dois números inteiros positivos e leia esses inteiros (a e b);
   - enquanto $b \neq 0$:
     - atribuir a *resto* o valor de a mod b;
     - faça a tomar o valor de b;
     - faça b tomar o valor de *resto*;
   - escreva os valores dados e o respectivo máximo divisor comum.

6. Elabore um programa que leia um número inteiro e verifique se é uma capicua, utilizando os seguintes passos:

   - Peça ao utilizador que forneça um número inteiro;
   - Leia esse número inteiro, n;
   - Atribua a *ncópia* o valor de n;
   - Inicialize a variável *inverso* a zero;
   - Enquanto *ncópia* diferente de zero:
     - atribua a *dígito* o valor do algarismo das unidades de *ncópia*;
     - atribua a *inverso* o seu valor anterior multiplicado por 10 mais o valor de *dígito*;
     - atribua a *ncópia* o valor da sua divisão inteira por 10;
   - Se o número dado for igual a *inverso* então escreve "é capicua"
     senão escreve "não é capicua".

7. (a) Desenvolva um programa que:
   - leia um número natural n;
   - leia uma sequência de n números reais para um array;
   - calcule o maior dos elementos do array.

   (b) Determine o número de comparações com elementos do vector realizadas pelo programa anterior (como função de n).

   (c) O que pode dizer quanto ao número de atribuições?

8. (a) Modifique o programa anterior de modo a que este calcule simultaneamente o maior e o menor dos elementos do array.

   (b) Determine o número de comparações realizadas por este programa.

   (c) O que pode dizer quanto ao número de atribuições?

9. Suponha que nos dois problemas anteriores os números dados pelo utilizador são positivos (o utilizador introduz um número negativo para indicar o fim da sequência). Resolva-os sem utilizar arrays.

10. (a) Implemente em Pascal o seguinte algoritmo que determina o máximo e o mínimo de uma sequência:
    - Ler n (o número de elementos da sequência);
    - Ler os n elementos para um vector x;
    - Para i desde 1 até $n/2$ se x_i é maior que o elemento correspondente a contar do fim do vector então trocar os elementos;
    - Procurar o menor elemento apenas até ao meio do vector;
    - Procurar o maior elemento a partir do meio do vector.

(b) Calcule o número de comparações realizadas pelo algoritmo como função do número de elementos da sequência.

(c) Quantas trocas são realizadas no melhor caso? E no pior caso? Identifique esses casos.

(d) Suponha que uma troca de elementos corresponde (em termos de tempo) a uma comparação. O que conclui em termos de eficiência deste algoritmo quando comparado com o algoritmo do exercício 8.

11. Diz-se que dois números inteiros positivos são amigos se cada um deles é igual à soma dos divisores próprios do outro.

(a) Desenvolva um subprograma em Pascal que verifique se dois dados números são ou não amigos.

(b) Utilizando o subprograma anterior desenvolva um programa que escreva no ecrã todos os pares de números amigos até 1000.

UNIVERSIDADE DE AVEIRO

Análise e Desenvolvimento de Algoritmos

2004/2005

Folha 3

1. Elabore subprogramas recursivos que realizem as seguintes funções:

   (a) dado um número inteiro positivo n, calcular $1 + 2 + \ldots + n$.

   (b) dado um número inteiro não negativo n, calcular o factorial de n, $n!$.

   (c) dados dois números inteiros positivos, a e b, calcular m.m.c.(a, b) baseando-se nas seguintes propriedades:

$$\text{m.m.c.}(a, b) = \frac{a \times b}{\text{m.d.c}(a, b)}$$

$$\text{m.d.c.}(a, b) = \begin{cases} a & \text{se } b = 0 \\ \text{m.d.c.}(b, a \bmod b) & \text{se } b \neq 0 \end{cases}$$

2. O subfactorial ($!n$) de um número inteiro não negativo , n, é definido por:

$$!n = \begin{cases} 1 & \text{se } n = 0 \\ !(n-1) \times n + (-1)^n & \text{se } n > 0 \end{cases}$$

   (a) Escreva uma função recursiva para calcular $!n$.

   (b) Implemente a correspondente versão iterativa.

3. Dados um número real x e um inteiro n,

$$x^n = \begin{cases} 1 & \text{se } n = 0 \\ x^{n-1} \cdot x & \text{se } n > 0 \\ x^{n+1}/x & \text{se } n < 0 \end{cases}$$

   (a) Implemente uma função recursiva em Pascal para calcular x^n.

   (b) Escreva a correspondente versão iterativa.

4. Implemente um procedimento recursivo que permita escrever, por ordem inversa, os dígitos de um número inteiro positivo.

5. Escreva um procedimento que converta recursivamente um dado inteiro sem sinal na respectiva string de dígitos.

6. Escreva uma função recursiva que converta uma string de dígitos no número inteiro por ela representado.

7. Escreva, usando um algoritmo recursivo, um subprograma que, dados um elemento k e um vector de n elementos inteiros, determine o número de vezes que k ocorre no vector.

8. Desenvolva um programa em Pascal que indique quais os movimentos necessários para resolver o "puzzle"das Torres de Hanoi.

9. Elabore um procedimento recursivo para escrever no ecrã a expressão $a^n b^n$ que representa a concatenação de uma sequência de n *a's* com uma sequência de n *b's* (a e b são caracteres e $n \in \mathbb{Z}_0^+$). Por exemplo, $a^2b^2 \iff aabb$.

10. Escreva um subprograma que implemente a *Função de Ackermann*:

$$A(m,n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1,1) & \text{se } n = 0 \\ A(m-1, A(m,n-1)) & \text{caso contrário} \end{cases}$$

com m , $n \in \mathbb{Z}_0^+$.

11. Elabore um procedimento recursivo que inverta um vector de comprimento $n \in \mathbb{N}$.

12. Elabore uma função que, dados m , $n \in \mathbb{Z}_0^+$, implemente:

$$\begin{pmatrix} n \\ m \end{pmatrix} = \begin{pmatrix} n-1 \\ m \end{pmatrix} + \begin{pmatrix} n-1 \\ m-1 \end{pmatrix}$$

sabendo que $\begin{pmatrix} n \\ 0 \end{pmatrix} = 1$.

13. Escreva um programa recursivo que determine se um número natural $n \in \mathbb{N}$ é capicua.

14. Usando um sub-programa recursivo, escreva um programa que leia uma palavra (sequência de letras terminada por '.') e crie um palíndromo cuja primeira palavra seja a palavra lida.

15. Seja $n \in \mathbb{N}$. Dados um real x e um vector de coeficientes $a[0..n]$:

    (a) Construa um procedimento recursivo que avalie o valor de um polinómio

$$P_n(x) = \sum_{i=0}^{n} (a_i x^i)$$

    usando o método de *Horner*.

    (b) Calcule o número de somas e multiplicações que o algoritmo efectua e compare-os com os resultantes de um procedimento iterativo.

16. Seja n um número inteiro e v um vector de inteiros.

    (a) Escreva uma função recursiva que detecte a posição de n em v. No caso de n não pertencer a v deverá devolver o valor *zero*.

    (b) Repita o problema da alínea (a) mas supondo agora que v é um vector ordenado e utilizando *Pesquisa Binária*.

(c) Ainda o mesmo problema mas usando agora *Pesquisa Ternária*.

(d) Faça a análise de complexidade para o *pior dos casos* do algoritmo da alínea anterior.

17. Escreva um procedimento que, dado $n \in \{0, 1, \ldots, 9\}$, desenhe no écran um losango como o da figura seguinte ($n = 3$):

```
    1
   121
  12321
   121
    1
```

Sugestão: Comece por determinar como pode desenhar recursivamente linhas de dígitos como as utilizadas na construção do losango acima.

UNIVERSIDADE DE AVEIRO

Análise e Desenvolvimento de Algoritmos

2004/2005

Folha 4

1. Seja A uma matriz real de ordem $n \times m$. Elabore subprogramas em Pascal que realizem as seguintes funções:

   (a) Calcular e escrever a soma dos elementos em cada linha da matriz.

   (b) Calcular e escrever a soma dos elementos em cada coluna da matriz.

   (c) Determinar a posição (linha e coluna) do maior elemento da matriz.

   (d) Verificar se A é a matriz identidade.

   (e) Verificar se A é simétrica.

   (f) Determinar a posição (linha e coluna) do elemento que mais se aproxima da média dos elementos da matriz.

2. Seja A uma matriz real, quadrada, de ordem n. O valor do determinante de A, calculado a partir do Teorema de Laplace, por desenvolvimento ao longo da primeira coluna, é dado por:

$$\det A = \sum_{i=1}^{n} (-1)^{i+1} a_{i1} \det(A_{i1}),$$

   onde A_{i1} é a matriz obtida de A por eliminação da linha i e da coluna 1.

   (a) Escreva a função recursiva:

```
function Laplace( A: Matriz ; n : Indice ): real;
```

   (b) Analise a ordem de complexidade do algoritmo recursivo de Laplace para o cálculo de determinantes.

3. Seja A uma matriz de ordem $m \times n$ de elementos inteiros ordenados de modo que

$$A[i,j] < A[i,j+1], \;\; i = 1, \ldots, m \;\; \text{e} \;\; j = 1, \ldots, n-1$$

$$A[i,n] < A[i+1,1], \;\; i = 1, \ldots, m-1.$$

   (a) Elabore um algoritmo que indique se o inteiro x pertence a A e que efectue, no pior dos casos, $m + n$ comparações .

   (b) Elabore um algoritmo que efectue menos comparações que o anterior.

4. Seja $x[1..n]$, $n \in \mathbb{N}$, um vector de elementos inteiros dispostos por ordem não decrescente.

   (a) Elabore um procedimento para inserir ordenadamente um novo elemento no vector, utilizando apenas um ciclo.

   (b) Faça uma análise do número de comparações, entre elementos do vector, realizadas pelo algoritmo.

5. A estratégia utilizada em cada passo do método de ordenamento habitualmente conhecido por selecção linear consiste basicamente na pesquisa do elemento máximo do vector e consequente troca com o último.

   (a) Escreva um procedimento recursivo que implemente esse algoritmo.

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

   (c) Um possível melhoramento do algoritmo anterior consiste na pesquisa conjunta do máximo e do mínimo seguida das necessárias trocas. Elabore um procedimento usando esta ideia.

   (d) Qual o número de comparações realizadas no caso mais favorável deste novo algoritmo? Identifique a instância do vector que origina este caso.

   (e) Idem para o pior caso.

   (f) Faça um estudo comparativo das complexidades dos dois algoritmos considerados.

6. Para colocar por ordem não-decrescente os elementos inteiros do vector $a[1..n]$, o algoritmo da Inserção Sequencial consiste em:

   - Procurar o menor elemento e trocá-lo com o primeiro;
   - Para cada elemento de ordem $i = 3..n$:
     { a[1..i-1] ordenado}
     - guardar a[i];
     - inseri-lo no sub-vector parcialmente ordenado;

     { a[1..i] ordenado}.

   (a) Elabore um procedimento que ordene um dado vector pelo método da Inserção Sequencial.

   (b) Faça uma análise do número de comparações efectuadas entre elementos do vector.

7. Considere o vector $x[1..n]$ de elementos inteiros. Pretendemos determinar o seu maior elemento.

   (a) Escreva um procedimento para esse efeito, utilizando o algoritmo trivial, e diga qual o número de comparações efectuadas com elementos do vector.

   (b) Elabore uma versão recursiva do mesmo, por sucessivas partições binárias do vector e consequente pesquisa em ambas as partes.

   (c) Calcule a complexidade da versão recursiva, em termos do número de comparações efectuadas com elementos do vector.

8. Com base na estratégia de Separação utilizada por Hoare no método de ordenamento Quicksort, construa a seguinte versão do problema:

   Procurar o Máximo num vector:

   - Separar os "maiores"dos "menores";
   - Procurar Máximo entre os "maiores".

   (a) Calcule a complexidade desta versão, em termos do número de comparações efectuadas com elementos do vector, no seu Melhor Caso.

   (b) Qual a complexidade do Pior Caso deste algoritmo?

   (c) Assumindo que o elemento arbitrado para a separação é o elemento médio do vector, indique qual a instância do problema que conduz ao Pior Caso do algoritmo.

UNIVERSIDADE DE AVEIRO

Análise e Desenvolvimento de Algoritmos

2004/2005

Folha 5

1. Considere a seguinte definição:

```
type Fraccao = record
                 num, den :integer
               end;
```

   (a) Elabore subprogramas em Pascal que lhe permitam operar com fracções (leitura, escrita, simplificação, soma, subtracção, divisão, multiplicação de fracções).

   (b) Utilizando os subprogramas anteriores escreva um subprograma para calcular f^n em que f é uma fracção e n um número inteiro.

   (c) Escreva um subprograma que calcule e escreva, na forma de fracção, a soma dos primeiros n termos da série:

$$1 + 1/2 + 1/3 + \ldots + 1/n + \ldots$$

2. Pretende-se implementar em Pascal a aritmética de complexos na forma cartesiana.

   (a) Defina o tipo de dados **complexo** como um record constituído por dois números reais.

   (b) Implemente as operações de adição, subtracção, multiplicação e divisão de complexos.

   (c) Utilizando os subprogramas anteriores, escreva um programa em Pascal que calcule z^n, em que $z \in \mathbb{C}$ é um número complexo e n um inteiro, ambos dados pelo utilizador.

3. Pretende-se determinar as raízes índice n de um dado número complexo.

   (a) Defina o tipo de dados **complex** que lhe permita representar complexos na forma polar.

   (b) Implemente as operações de adição, subtracção, multiplicação e divisão de complexos na forma polar.

   (c) Utilizando os subprogramas anteriores, escreva um programa em Pascal que calcule z^n, em que $z \in \mathbb{C}$ é um número complexo na forma polar e n um inteiro, ambos dados pelo utilizador.

   (d) Escreva um programa em Pascal que calcule $\sqrt[n]{z}$, em que $z \in \mathbb{C}$ é um número complexo na forma polar e n um inteiro, ambos dados pelo utilizador.

4. Os seguintes tipos de dados em Pascal permitem definir sequências de números reais, como uma lista ligada simples.

```
type elemento = real;
     sequencia = ^termo;
     termo = record
                 val: elemento;
                 prox : sequencia
             end;
```

   Escreva subprogramas em Pascal que implementem as seguintes funções:

   (a) criar uma sequência vazia;

   (b) juntar um elemento no início de uma dada sequência;

   (c) escrever no ecrã todos os elementos de uma sequência;

   (d) juntar um elemento no fim de uma sequência;

   (e) remover o primeiro elemento de uma sequência;

   (f) remover o último elemento de uma sequência;

   (g) procurar um dado elemento numa sequência;

   (h) inserir em ordem um dado elemento numa sequência;

   (i) remover um dado elemento de uma sequência;

   (j) realizar uma cópia de uma sequência;

   (k) destruir uma sequência.

5. Considere que as listas do problema 4 são listas ligadas simples circulares, com uma posição inicial vazia. Implemente todas as funções referidas nesse problema e indique as vantagens / desvantagens desta representação.

6. Defina em Pascal sequências de números reais, como listas circulares duplamente ligadas, com uma posição inicial vazia. Implemente as funções referidas no problema 4 e indique as vantagens / desvantagens desta nova representação.

7. Pelo Teorema Fundamental da Aritmética qualquer número inteiro positivo pode ser decomposto como produto de potências não negativas de números primos, $n = p_1^{n_1} \times p_2^{n_2} \ldots \times p_k^{n_k}$.

   Considere os tipos de dados em Pascal:

```
type canonico =^termo;
     termo= record
                primo, expoente:integer;
                prox: canonico
     end;
```

(a) Faça um esboço da representação do número 3528 segundo a definição anterior.

(b) Elabore subprogramas em Pascal para:

i. obter a representação canónica de um dado inteiro positivo x.

ii. verificar se um dado inteiro na forma canónica é ou não um número primo.

iii. calcular o produto de dois inteiros na forma canónica.

iv. calcular o máximo divisor comum de dois números inteiros na forma canónica.

v. calcular o mínimo múltiplo comum de dois números inteiros na forma canónica.

8. Escreva uma função que:

(a) procure um dado elemento numa lista e devolva nil caso não encontre esse elemento.

(b) retire e devolva um ponteiro para o nó com o "maior"elemento de uma dada lista.

9. Escreva um procedimento para:

(a) ordenar por ordem ascendente uma dada lista.

(b) ordenar por ordem descendente uma dada lista.

10. Elabore uma função que faça a fusão ordenada de duas listas previamente ordenadas:

(a) devolvendo uma nova lista.

(b) devolvendo uma das listas dadas actualizada.

UNIVERSIDADE DE AVEIRO

Análise e Desenvolvimento de Algoritmos

2004/2005

Folha 6

1. Defina o Tipo Abstracto de Dados POLINOMIO de uma variável e coeficientes reais, por intermédio de uma escolha adequada de operações.

   (a) Proponha uma estrutura de dados adequada para a representação desse tipo.

   (b) Implemente a operação $DIVIDIR(pol1, pol2) \longrightarrow (quociente, resto)$.

   (c) Elabore um subprograma para calcular o máximo divisor comum de dois polinómios.

2. O ficheiro de texto Casos contém um conjunto de nomes, sem nenhuma ordem em especial, aos quais está associada informação de um determinado tipo.

   (a) Elabore um procedimento para ler o ficheiro Casos e construir uma lista ligada onde os nomes aparecem pela mesma ordem.

   (b) Com vista à optimização de repetidas consultas a esta lista, e tendo-se verificado que certos nomes requerem consultas muito mais frequentes, foi decidido que estes deveriam manter-se no início da lista. Para esse efeito elabore o procedimento :

   ```
   procedure consulta-altera(var ponta: lista; estenome: nome; var val: info);
   ```

   que, para além de consultar `estenome`, deverá alterar a lista de modo a obter o resultado pretendido.

   (c) Como terá certamente reparado, uma versão mais eficiente do procedimento anterior consiste em, após cada consulta, recuar o respectivo nó uma posição na lista. Implemente esta versão.

3. Uma Lista de Cartas de Jogar pode ser representada de acordo com a seguinte declaração :

```
type naipes = ( paus, ouros, copas, espadas );
     valores = (dois, tres, ... , dez, valete, dama, rei, as);
     lista = ^carta;
     carta = record
               naipe : naipes;
               valor : valores;
               prox  : lista
             end;
```

   (a) Sendo dada uma Lista de Cartas - `cartas` - e a indicação de `este-naipe`, elabore um subprograma para construir a lista ordenada das cartas de `este-naipe` que nela ocorrem.

(b) Proponha uma estrutura de dados adequada à representatação de uma "Mão" (conjunto de cartas que um jogador possui). Indique as vantagens da estrutura que propôs.

(c) Elabore um procedimento para converter uma dada Lista de cartas à representação definida na alínea anterior.

4. Seja `P` uma estrutura de lista ligada do tipo Pilha. Elabore sub-programas que efectuem as seguintes operações sobre `P`:

   (a) Listar o conteúdo de `P` por ordem inversa.

   (b) Contar o número de elementos de `P`.

   (c) Remover um elemento específico de `P`.

5. Existe uma variante particular da estrutura de dados do tipo Fila que permite inserir ou remover elementos por qualquer um dos seus extremos e que vamos denominar por FILA DUPLA (DEQUE em inglês). Proponha uma representação e um conjunto mínimo de operações que permita implementar em Pascal o Tipo Abstracto de Dados FILA DUPLA.

6. Sabendo que, para $n \geq k \geq 0$, $C_k^n = C_k^{n-1} + C_{k-1}^{n-1}$ com $C_n^n = C_0^n = 1$, elabore um procedimento iterativo para calcular o número de C_k^n simulando uma estratégia recursiva. Este procedimento, para além do valor C_k^n, deve ainda devolver o número de simulações de chamadas recursivas feitas.

   Sugestão: Utilize uma estrutura de Pilha para registrar os sucessivos parâmetros das simulações das chamadas recursivas.

7. Seja ABP o tipo abstracto das Árvores Binárias de Pesquisa, onde a informação associada a cada nó é identificada por um campo chamado chave. Existe, naturalmente, uma relação de ordem total definida no conjunto das chaves.

   (a) Escreva uma definição formal do tipo ABP.

   (b) Directamente a partir da definição anterior, demonstre (por indução) que a pesquisa de um elemento numa árvore total do tipo ABP é de ordem de complexidade logarítmica.

   (c) Elabore um procedimento para a Remoção de eventuais repetições numa árvore do tipo ABP.

8. A informação referente ao aproveitamento dos 100 alunos de um dado curso encontra-se armazenada de acordo com as seguintes declarações:

```
type  ponteiro  =  ^cadeira;
      cadeira  =  record  nome-de-cadeira : nome;
                    nota : 0..20;
                    prox : ponteiro;
                  end;
      arvbin  =  ^aluno;
      aluno  =  record
                  nome-de-aluno : nome;
                  nome-de-curso : nome;
                  lista-de-cadeiras : ponteiro;
                  esq, dir : arvbin;
                end;
```

Assumindo que:

- a arvbin definida é uma árvore binária de pesquisa onde os vértices estão ordenados alfabeticamente em-ordem por nome-de-aluno;
- cada lista-de-cadeiras está ordenada por ordem alfabética de nome-de-cadeira;

elabore sub-programas para:

(a) Listar por ordem alfabética os nomes dos alunos de este-curso.

(b) Actualizar a estrutura de dados de acordo com os resultados obtidos em esta-cadeira.

A pauta respectiva está registada num ficheiro onde, em cada linha, se encontra o nome de um aluno (50 caracteres) seguido de ":" e uma letra (D ou R) ou de um valor inteiro entre 10 e 20. Se um determinado aluno já tiver aprovação em esta-cadeira, deverá ser-lhe atribuída a melhor das duas notas.

Note que cada vértice da árvore só deve ser visitado uma única vez.