



Programação 2

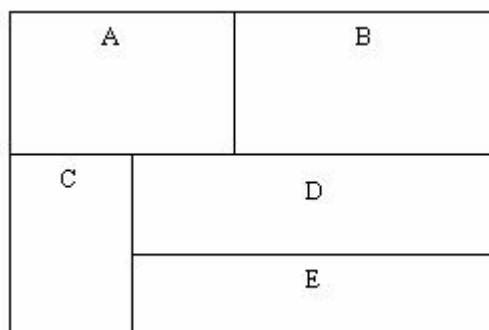
2º Semestre

Folha de Exercícios 4A

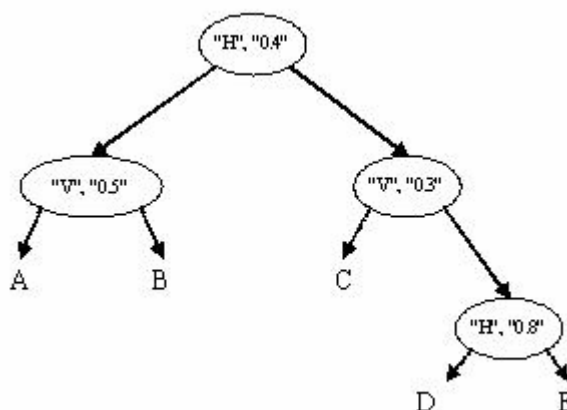
Efectue as tarefas de programação descritas abaixo, usando a linguagem C++ em ambiente Linux.

Exercício 1

O desenho da interface com o utilizador passa, muitas vezes, pelo agrupamento das opções de acordo com as suas funções na aplicação, ou, no caso de páginas HTML, do seu conteúdo. A selecção de uma opção geralmente é realizada com auxílio de um dispositivo apontador, como o rato. No caso das páginas HTML é frequente a organização de diferentes assuntos em forma de frames. A selecção de um frame para torná-lo activo pode ser realizada com o rato, por exemplo. A questão, entretanto, é saber, dadas as coordenadas (x, y) do dispositivo apontador, qual dos frames se está a seleccionar. Uma página organizada em frames pode ser representada a partir de uma árvore binária. Considere uma webpage com a seguinte topologia:



Essa estrutura pode ser representada pela seguinte árvore binária:



em que cada nó da árvore contém duas informações básicas: se a divisão é vertical (“V”) ou horizontal (“H”), e em que proporções a divisão acontece. O lado esquerdo de um nó representa a primeira parte da divisão (parte de cima, no caso de divisão horizontal, ou parte da esquerda, no caso de divisão vertical) enquanto o lado direito a parte complementar. Considere que a razão da divisão é uma proporção de um valor absoluto para a largura e a altura da janela da interface.

Utilize uma árvore binária (**BinaryTree**) apresentada nas aulas, disponível no ficheiro [BinaryTree.h](#).

```

class FrameNode
{
public:
    char isVH;
    float ratio;
    ...
}

class FrameSet
{
    int largura, altura;
    BinaryTree<FrameNode> frames;
public:
    FrameSet();
    void iniciar(char *file);
}

```

a) Implemente o método `iniciar` da classe `FrameSet`.

```
Void FrameSet::iniciar(char *file)
```

O ficheiro `file` contém informação sobre a topologia de webpage. As duas primeiras linhas contém a altura e a largura da frame mais exterior. As linhas seguintes do ficheiro são do formato:

```

string (nome da frame)
char ('n' se frame simples; 'v' se divisão vertical em 2 subframes; 'h' se
    divisão horizontal em 2 subframes. Se !='n' tem mais 3 linhas: )
float (razão)
string (nome da 1ª subframe)
string (nome da 2ª subframe)

```

Suponha que no ficheiro as frames mais interiores estão indicadas antes das frames exteriores.

b) Implemente o membro-função

```
int FrameSet::numFrames ()
```

que retorna o número de frames simples (não compostas) da webpage armazenada na árvore.

c) Implemente o membro-função

```
string FrameSet::encontrarFrame(int x, int y)
```

que, a partir das coordenadas `x` e `y` do dispositivo apontador, identifica o nome do frame simples seleccionado.

Sugestão: use um iterador por nível. Um nó da árvore tem sempre 0 ou 2 filhos. Os filhos de um nó na posição `pos` estão em `posF1=2*pos` e `posF2=2*pos+1` (se no caminho por nível até lá encontrar nós sem filhos decrementar `-2` a `posF1` e `posF2`).



Programação 2

2º Semestre

Folha de Exercícios 4A

Efectue as tarefas de programação descritas abaixo, usando a linguagem C++ em ambiente Linux.

Exercício 2

Considere que a análise de um qualquer texto consiste em determinar todas as palavras que nele ocorrem e a sua frequência. Considere a classe **PalavraNVeces**, que representa informação sobre a ocorrência de determinada palavra no texto.

```
class PalavraNVeces {
    string palavra;
    int nVeces;
public:
    PalavraNVeces(string pal, int n): palavra(pal), nVeces(n) {};
};
```

Todo o código pedido para resolver o problema deve ser colocado na directoria **texto**, criada na sua área de trabalho. O código fonte do programa deve ser escrito num ficheiro com o nome “Texto.cpp”.

Nota: deve usar a implementação de árvores binárias apresentada nas aulas (não alterar este código): [BST.h](#)

- a) Implemente a classe **AnaliseTexto**, que irá disponibilizar métodos úteis para a análise de um texto. Esta classe usa uma árvore de pesquisa binária (BST) para armazenar objectos da classe **PalavraNVeces**. Os elementos da árvore de pesquisa binária devem estar ordenados alfabeticamente.

```
class AnaliseTexto {
    BST<PalavraNVeces> palavras;
public:
    AnaliseTexto();
    void trataTexto(string texto);
    string palavraOrdemK(int k);
    string maisVeces(string pref);
};
```

Implemente:

- o método construtor
- o método void *trataTexto(string texto)*. Este método cria a árvore de pesquisa binária que contém as palavras existentes em texto e respectivo número de ocorrências.

Teste o código implementado.

- b) Na classe **AnaliseTexto** implemente o método

```
string palavraOrdemK(int k)
```

que retorna a k -ésima maior palavra da árvore, de acordo com a ordenação desta. Recordar-se que a árvore de pesquisa binária está ordenada por ordem crescente de palavras (ordem alfabética).

- c) Na classe **AnaliseTexto** implemente o método

```
string maisVeces(string pref)
```

que determina qual a palavra com maior número de ocorrências no texto inicial (presente na árvore), de prefixo *pref*.