13 Ficheiros

Os ficheiros *CSV* (*Comma separated values*) são um padrão informal, muito usado, para trocas informação entre aplicações distintas (por exemplo, todos os programas o tipo «folha de cálculo», processam este tipo de ficheiro.

É importante então saber como ler e escrever neste formato. Resolva os exercícios seguintes usando o formato *CSV* nos ficheiros.

74 Dado uma lista de valores (do tipo inteiro) contidos no ficheiro `listaNaoOrdenada.dados`, leia os valores, ordene-os e escreva-os no ficheiro `listaOrdenada.dados`.

75 Dado um ficheiro contendo uma série de expressões em NPI (notação polaca inversa), proceda ao seu cálculo e escreva o resultado num outro ficheiro.

O formato de saída, por linha do ficheiro, deverá ser `expressão_em_NPI = resultado`.

14 Biblioteca Padrão do C++

Usando os diferentes «livros» da «biblioteca» padrão do *C++*, construa programas para:

76 Construa um programa capaz de obter (e mostrar) o tempo de execução (tempo de CPU) de uma dada função.

77 Para operações geométricas básicas é possível definir ponto através de números complexos. Para um complexo $z = (x + \mathrm{i}y)$ podemos vê-lo como um ponto com coordenadas (x, y).

Temos então as seguintes operações:

- Adição de vectores: $a + b$;
- Multiplicação escalar: $r * a$;
- Produto interno: $(\mathrm{conj}(a) * b).\mathrm{real}()$;
- Producto vectorial: $(\mathrm{conj}(a) * b).\mathrm{imag}()$;
- Distância Euclideana: $\mathrm{abs}(a - b)$

Usando a classe `complex` da STL defina ponto, as funções acima e a função que nos dá a intersecção de duas rectas (não paralelas). Construa um pequeno programa que permita testar as funções que definiu.

15 Standard Template Library (STL)

78 Implemente o algoritmo de ordenação Borbulhagem («Bubble Sort») usando para tal «contentores» apropriados:

1. utilizando a classe `Lista` (exercício 12.3).

2. utilizando os métodos próprios da classe `vector` (sem usar iteradores);

3. utilizando a classe `list` e os «iteradores» correspondentes.

Calcule as diferenças de tempo de execução entre implementações.

79 Calculadora em Notação Polaca Inversa, usando para tal o «contentor» `stack`, com os «iteradores» correspondentes.

80 Implemente o algoritmo de ordenação «Quick Sort» usando para tal o «contentor» `list`, com os «iteradores» correspondentes.

16 Gestão de Erros

O *C++*, através do mecanismo `raise` e `catch` permite uma gestão dos erros, sinalizando-os e gerindo-os aquando da sua ocorrência.

Volte a considerar os seguintes TAD mas agora tenha o cuidado de gerir os casos de erro.

81 Pilhas=({pilhaVazia,Inteiro:Pilha},{cria, push, pop, top, vazia?});
Os métodos `top` e `pop` são ambos problemáticos quando aplicados a uma pilha vazia.

82 Filas=({filaVazia,Inteiro:Fila},{cria, insere, retira, topo, vazia?});
Os métodos `retira` e `topo` são ambos problemáticos quando aplicados a uma fila vazia.

83 Listas=({listaVazia,Inteiro:Lista},{cria, insereI, retiraI, veI, vazia?})
Os métodos `insereI`, `retiraI` e `veI` são todos problemáticos para valores do i fora da número de elementos na lista, Além disso os métodos `retiraI` e `veI` são ambos problemáticos quando aplicados a uma lista vazia.