PROJECTO 1
CLASSIFY: GESTÃO DE CLASSIFICAÇÕES
(ßeta)

INTRODUÇÃO E OBJECTIVOS

A fnalidade deste projecto passa por aplicar os conhecimentos adquiridos até agora (sobre programação em geral, e sobre C/C++ em particular) no desenvolvimento de aplicações úteis para manutenção e administração de sistemas. Em particular, pretende-se que desenvolva codigo modular, decomposto em funções, utilizando os tipos de dados apropriados e, sempre que possível, recorrendo a bibliotecas que implementem as funcionalidades de que precisa.

Pretende-se que desenvolva uma mini-aplicação interactiva para gestão das notas de uma turma. Como **extra**, pode desenvolver um comando para detectar semelhanças entre fcheiros. Todas as partes do projecto a desenvolver são mencionadas na secção **AVALIAÇÃO**. Nesta secção encontra também uma explicação sobre o que se entende por "extra" no contexto deste projecto.

PARTE I – GESTÃO DAS CLASSIFICAÇÕES DE UMA TURMA

Apos corrigir os testes de uma turma, e de lhes atribuir uma pontuação de 0 a 200, um determinado professor deseja classifcá-los qualitativamente, um a um, utilizando a seguinte escala:

PONTUAÇÃO	CLASSIFICAÇÃO QUALITATIVA
180 - 200	Excelente
150 - 180	Bom
120 - 150	Sufciente
< 120	Insufciente

As classifcações do texto são obtidas a partir de um fcheiro indicado na linha de comandos. Por exemplo, se o nome do fcheiro for turma01.csv, o programa deve ser invocado na linha de comandos

da seguinte forma:

```
$ classify turma01.csv
```

Eis um exemplo de tal fcheiro:

```
Alberto Antunes,alb@mail.com,1990-07-23,150,REG
Armando Alves,arm@mail.com,1990-07-23,78,BOLS Pedro
Pereira,pp@mail.com,1994-07-23,162,REG
Bruno Bastos,brn@mail.com,1991-02-23,190,TRAB
Arnaldo Almeida,arn@mail.com,1987-07-23,128,REG
```

Cada linha contém a seguinte informação sobre um estudante: nome completo, endereço de email, data de nascimento, classifcação actual e estatuto (valores possíveis: BOLSeiro, REGular, estudante TRABalhador).

O programa deve ler o fcheiro para um estrutura de dados, sendo que as operações em baixo indicadastrabalham sobre esta estrutura de dados. Por exemplo, pode recorrer a uma lista, a um tuplo, a um dicionário, ou a qualquer combinação destas estruturas. Não utilize classes.

Apos ler as classifcações, o seu programa exibe o seguinte menu:

```
GESTÃO DE CLASSIFICAÇÕES
L - Listar classificações
P - Pesquisar classificações
A - Actualizar informação sobre estudante
E - Exportar classificações
R – Reler classificações
0 - Terminar
09:52:19 >
```

A linha de comandos exibe a hora actual no formato em cima indicado. Apos cada uma das opções ter sido executada, o programa aguarda que o utilizador pressione qualquer tecla, apos o que apaga o ecrã e volta a exibir as opções do menu principal. Para implementar estas funcionalidades, invoque os comandos do SO apropriados usando para tal o modulo subprocess. A tabela seguinte indica possíveis comandos a utilizar.

DESCRIÇÃO DO COMANDO	WINDOWS/CMD	UNIX/BASH
Apagar ecrã	cls	clear
Aguardar que tecla seja pressionada	pause>nul|set/p='Pressione...'	read -s -n 1 -p \ 'Pressione...'

A opção **L** permite ver uma listagem igual à indicada a seguir:

```
NOME            | Classificação | Classificação |        Email
                | Quantitativa  | Qualititativa |
----------------+---------------+---------------+-------------------------
Alberto Antunes |         150 |      Bom      |      alb@mail.com
```

Armando Alves | 78 | Insuficiente | arm@mail.com

Arnaldo Almeida | 128 | Suficiente | arn@blo.co.uk
Bruno Bastos | 190 | Excelente | bruno.bastos@mail.org Pedro Pereira | 162 | Bom | pp@email.org

MÉDIA: 141.6 SUFICIENTE

Número De Alunos : 5
Melhor Classificação : 190 (Excelente) | Bruno Bastos | bruno.bastos@mail.org
Pior Classificação : 78 (Insuficiente) | Armando Alves | arm@mail.com

Através da opção **P** acede ao menu de pesquisas. Neste menu, o utilizador escolhe o tipo de pesquisa – por nome, por email ou por intervalo de classifcações – e depois o programa exibe uma tabela igual à anterior, mas apenas para os alunos seleccionados.

Na pesquisa por nome, o utilizador introduz o nome (ou parte do nome). As pesquisas por nome e por email devem ignorar a capitalização. Na pesquisa por email o utilizador deve introduzir um email completo, ao passo que na pesquisa por intervalo, o utilizador deve introduzir os limites inferior e superior do intervalo. Se o utilizador não introduzir um dos limites (pressionando apenas ENTER quando solicitado a introduzir um destes valores), a pesquisa assume 0 para o limite inferior e 200 para o limite superior.

A opção **A** permite actualizar parte da informação de um estudante. O estudante é localizado através do email, e depois é possível alterar um dos seguintes parâmetros: email, classifcação ou estatuto.

Através da opção **E**, o programa solicita o nome de um fcheiro a criar, fcheiro esse que será criado com seguinte conteúdo:

MÉDIA: 141.6 SUFICIENTE

Alberto Antunes,alb@mail.com,150,Bom
Armando Alves,arm@mail.com,78,Insuficiente
Arnaldo Almeida,arn@blo.co.uk,128,Suficiente
Bruno Bastos,brn@mail.com,190,Excelente
Pedro Pereira,pp@email.org,162,Bom

Com a opção **R**, o programa relê o fcheiro de classifcações, descartando quaisquer alterações que tenham sido efectuadas durante a execução do programa até então.

Com a opção **0** (zero), o programa termina, não sem antes gravar a informação sobre os estudantes no fcheiro de entrada (turmas01.csv, no exemplo).

[PARTE II - SYMS]

Vamos agora fazer um programa **extra** para detectar fcheiros com "semelhanças". Através da linha de comandos, o seu programa recebe um caminho para uma directoria e "desce" essa directoria procurando por fcheiros que apresentem uma determinada semelhança. No fnal exibe uma listagem com os grupos de fcheiros semelhantes.

De seguida, precedidas da respectiva opção da linha de comandos, indicamos as semelhanças a detectar:

- -c, --contents: detecta fcheiros com conteúdo binário igual
- -n, --name: detecta fcheiros com o mesmo nome (incluindo a extensão)
- -e, --extension: detecta fcheiros com a mesma extensão
- -r PATTERN, --regex==PATTERN: detecta fcheiros cujo o nome verifca a expressão regular dada por PATTERN; os fcheiros são depois agrupados pela ocorrência concreta do padrão

A sintaxe do comando deve ser a seguinte:

$ syms [-c] [-n] [-e] [-r PATTERN] [caminho_dir]

Utilize a biblioteca **docopt** para leitura das opções da linha de comandos. Se mais do que uma opção for especifcada, então o seu programa deve exibir uma listagem por opção. Se nenhuma opção for especifcada, o programa assume a opção -n. O valor por omissão para caminho_dir é a directoria corrente.

Sugestões:

1. Utilize boost::filesystem::recursive_directory_iterator para percorrer a árvore de directorias dentro do caminho fornecido

2. Para evitar ter que comparar fcheiros, processo dispendioso em termos de CPU e de utilização dememoria, utilize uma função de hashing para obter de forma efciente um resumo de cada fcheiro. Este resumo é, para todos os efeitos, unívoco, isto é, dois fcheiros com conteúdo igual produzem um resumo igual; dois fcheiros com conteúdo diferente (mesmo que ligeiramente diferente) produzem um resumo que será, com toda a probabilidade, diferente. Pode utilizar um dos algoritmos de hashing da biblioteca da suite de criptografa OpenSSL ou então pode recorrer à biblioteca Crypto++ (tcc, Cryptocpp).

3. Pode utilizar um mapa para guardar estes resumos e listar, por cada resumo, o nome dos fcheiros (numvector de strings) que produziram esse resumo. Quando encontra um fcheiro, obtém o resumo e verifca-se já existe um resumo igual no dicionário. Se for o caso, então o fcheiro é duplicado e pode acrescentar o nome do fcheiro à lista correspondente no mapa.

AVALIAÇÃO

No contexto deste projecto, um elemento **extra** é um componente ou funcionalidade cuja cotação (ver tabela em baixo) é substancialmente inferior à de outros componentes ou funcionalidades de difculdade semelhante. É possível ter uma boa classifcação neste projecto não realizando nenhum dos elementos extras.

Como incentivo, a classifcação das funcionalidades **extra** pode contribuir até 2 valor(es) para a nota do proximo teste escrito e até 1 valor(es) para a nota do proximo projecto. O projecto deve ser resolvido em grupos de dois formandos. Excepcionalmente, poderá ser realizado por grupos com outras dimensões.

ELEMENTO	**COTAÇÃO MÁXIMA (0..20)**

Classify (Restantes opções)	10
Classify (Opões Actualizar, Exportar)	8
Syms	2

Como incentivo, a classifcação das funcionalidades **extra** pode contribuir até 2 valor(es) para a nota do proximo teste escrito e até 1 valor(es) para a nota do proximo projecto. O projecto deve ser resolvido em grupos de dois formandos. Excepcionalmente, poderá ser realizado por grupos com outras dimensões.

Deve também elaborar um relatorio (ver secção **ENTREGAS**) que valerá 20% da cotação do projecto.
Aconselha-se que a primeira parte do relatorio, **INTRODUÇÃO E OBJECTIVOS** e **DESENHO E ESTRUTURA**, seja realizada em primeiro lugar, antes mesmo de avançar para uma implementação.

ENTREGAS

O projecto deve ser entregue até às **23h59m** do dia **03/04/2020**. Um atraso de *N* dias na entrega levará a uma penalização dada pela formula $0.5\times2^{(N-1)}$ ($N > 0$).

Deverá ser entregue um ZIP com o seguinte:

1. Ficheiros **classify.cpp** e **syms.cpp** com o codigo produzido. Cada fcheiro deve estar documentada com o nome dos elementos do grupo e com a data de entrega.

   Finalmente, este ZIP deve também incluir o fcheiro **Relatorio.pdf** (ver a seguir).

2. Relatorio em PDF que deve seguir o modelo fornecido em anexo. Em termos de formatação, adapte apenas a designação da acção e o nome dos modulos. Siga as recomendações relacionadas com a elaboração de um relatorio dadas pelo formador Fernando Ruela. O relatorio deve incluir uma capa simples com o símbolo do IEFP, referência ao Centro de Formação de Alcântara, data, indicação do curso e da acção (eg, *Tecnico de Informatica - Sistemas 07*) e dos elementos que elaboraram o trabalho.

   Em termos de conteúdo o seu relatorio deve possuir as seguintes secções e anexos:

   2.1 **Introdução e Objectivos:** Por palavras suas, descreva o proposito do projecto e quais os principais objectivos a atingir. Não plagie a introdução deste enunciado.

   2.2 **Desenho e Estrutura:** Para este projecto, é sufciente elaborar um fuxograma a descrever o algoritmo/processo principal de cada um dos programas realizados.

   No caso do **classify**, deve elaborar um fuxograma para

   1. Leitura do fcheiro inicial de classifcações e importação dos dados para a estrutura de dadosem memoria.

2. Ciclo do menu principal

3. Pesquisa por nome e intervalo de notas

Para o **syms**, apenas necessita de considerar as opções -c e -n. Deve também considerar a possibilidade de ambas terem sido seleccionadas.

Os fuxogramas devem incidir nos algoritmos e nos processos, e na logica que lhes é subjacente. Não devem incluir instruções de programação, nem incluir detalhes de visualização (eg, “apaga ecrã” ou “formata a 20 colunas”).

2.3 **Implementação:** Deve descrever brevemente a solução implementada para cada um dos programas. Em particular, os aspectos mais importantes a mencionar para cada programa são:

- As funções principais que defniu.
- As estruturas de dados principais utilizadas e sua fnalidade
- Os principais modulos utilizados, qual a fnalidade de cada um e, dentro destes, quais osmecanismos efectivamente utilizados.

Sempre que achar necessário pode incluir pedaços de codigo ilustrativos.

2.4 **Conclusão:** Além de seguir as recomendações indicadas no modelo a respeito da elaboração da conclusão de um relatorio, deve também listar o que foi implementado e o que fcou por implementar, indicando, neste último caso, o porquê de não ter sido implementado.