DEEC
DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

PROGRAMAÇÃO

(LEEC 22/23)

Enunciado do Projecto

Concurso Nacional de Acesso ao Ensino Superior

Colocação de Candidatos

1. Introdução

O trabalho que se descreve corresponde ao projecto da UC de Programação para o MEEC em 2022/2023. O projecto a realizar corresponde à implementação de uma versão simplificada para colocação de candidatos ao ensino superior. O objetivo do projeto é promover e avaliar a prática dos conceitos e técnicas de programação estudados no âmbito desta UC. O projecto deverá ser realizado ao longo do período acompanhando a matéria teórica lecionada nesta UC. Conforme descrito na secção de avaliação, haverá demonstrações parciais a realizar numa aula de laboratório e uma entrega final, que corresponde à entrega do projecto completo.

2. Descrição do Problema a Resolver

A colocação de candidatos ao ensino superior é um problema que se coloca anualmente e que envolve mais de 60000 candidatos e mais de 1000 cursos em instituições de ensino superior. Será um problema complexo? será um problema de elevado custo computacional? … como se comprovará, será um problema de fácil resolução podendo o resultado das colocações ser obtido instantaneamente num computador portátil de uso geral.

2.1. Dados de Entrada

Os dados de entrada correspondem ao conjunto de cursos e à lista de candidatos a esses cursos.
Os cursos caracterizam-se por: (1) identificador de Instituição; (2) identificador de curso; (3) nome de instituição; (4) nome do curso; (5) grau conferido pelo curso; (6) número de vagas do curso. Existem 1103 cursos, com um total 54641 vagas, aos quais os candidatos se podem candidatar. Os candidatos caracterizam-se por: (1) identificador do candidato; (2) nota na prova de ingresso; (3) média do secundário; (4) nota de candidatura; (5) entre 1 e 5 pares de indentificadores de instituição de ensino e identificador de curso. Existem 60000 candidatos que pretendem ser colocados.
Nos testes de validação do projeto serão considerados vários exemplos com diferentes subconjuntos de cursos e de candidatos.
Consulte o formato dos dados de entrada na secção 3.1. Note que, para simplificar, algumas informações dos dados de entrada não são usadas. Por exemplo, não são utilizadas a (2) nota na prova de ingresso e (3) média do secundário, assumindo-se que a (4) nota de candidatura é igual para todos os cursos em que um aluno se candidate.

2.2. Algoritmo de Colocação de Candidatos

Considere o algoritmo descrito de seguida para colocação dos candidatos ao ensino superior. Assuma que cada candidato tem entre 1 e 5 opções de curso sendo a opção 1 (OP1) a mais prioritária e a opção 5 (OP5) a menos prioritária. A colocação de candidatos é feita sequencialmente começando no candidato 1 (C1) e seguindo até ao último. Os candidatos não colocados deverão ser guardados numa estrutura de dados a definir e por ordem do seu identificador (ID). Por simplicidade considera-se que os cursos indicados pelo candidato são cursos aos quais se pode candidatar com base na sua formação e nota de candidatura, isto é, podem surgir candidatos com opções em áreas científicas diferentes. Em caso de candidatos colocados com a mesma nota de candidatura no mesmo curso deve ser considerado empate, não havendo lugar a desempate (ainda assim devem ser listados por ordem de ID, ver exemplos); isto pode dar origem a que o número de vagas seja excedido se os últimos candidatos tiverem a mesma nota de candidatura.

P1: Ler a informação dos cursos e representar cada curso na estrutura de dados definida para representar cursos.

P11: Abrir o ficheiro de cursos
P12: Verificar se ainda existem cursos não lidos no ficheiro

P121: Se existirem cursos por ler: Ler linha de ficheiro com informação para estrutura de dados correspondente. Regressa ao passo P12
P122: Se não existirem mais cursos por ler, concluir a tarefa e avançar para colocação de candidatos (P2).

P2: Ler a informação dos candidatos, representar cada candidato na estrutura de dados definida para representar candidatos e executar colocação de candidatos.

P21: Abrir o ficheiro de candidatos
P22: Verificar se ainda existem candidatos não lidos no ficheiro

P221: Se existirem candidatos por ler: Ler linha de ficheiro com informação para estrutura de dados correspondente. Executa colocação de candidato em P3
P222: Se não existirem mais candidatos por ler, a **colocação de candidatos está concluída**!

P3: Coloca candidato na opção de curso mais prioritária que tenha vaga ou guarda candidato como não colocado.

P31: Seleciona, para o candidato, qual a opção de curso mais prioritária ainda não testada (OPi)

P311: Caso exista opção, avança para P32
P312: Caso não exista mais opções de colocação, **guarda candidato como não colocado** e regressa a P22.

P32: Verifica se existe vaga para o candidato na opção OPi, isto é, se a nota de acesso é maior ou igual à nota mínima.

P321: Se sim, avança para P33
P322: Se não, avança para P31

P33: Inserção de candidato no curso correspondente à opção OPi

P331: Insere candidato, por ordem de nota de candidatura, no curso OPi
P332: Atualiza número de vagas do curso OPi
P333: Atualiza nota mínima de acesso ao curso OPi
P334: Verifica se com a colocação excede o número de vagas do curso

P3341: Se não excede o número de vagas, concluiu a colocação e passa para P22.
P3342: Se excede, verificar se precisa de retirar candidatos e tenta colocá-los nas opções seguintes **repetindo o processo de colocação para cada candidato retirado do curso**. Depois regressa a P22.

Fig. 1: Algoritmo de Colocação de Candidatos

No caso do passo P3342, só se retiram candidatos se o nº de vagas for n e se tiver um número de colocados maior ou igual a n+1 e a nota do candidato colocado na posição n+1 for inferior à nota do candidato colocado na posição n.

Nota: Aceitam-se implementações com outras variantes do algoritmo desde que respeitem as regras descritas e que produzam o mesmo output.

2.3. Dados de Saída

O programa deverá produzir uma listagem dos candidatos colocados nos vários cursos e indicar para cada curso a nota mínima de acesso e o nº de vagas por preencher. Deverá ainda ser produzida uma listagem dos candidatos não colocados. Consulte formato de saídas na secção 3.3.

3. Implementação

3.1. Ficheiros de Entrada

Os dados de entrada Cursos e Candidatos serão fornecidos em ficheiros CSV (Valores Separados por Vírgula) com os parâmetros descritos na secção 2.1. Note-se, figs. 2 e 3, que a primeira linha de ambos os ficheiros tem a designação dos vários parâmetros das linhas abaixo.

```
Codigo Instit.,Codigo Curso,Nome da Instituicao,Nome do Curso,Grau,Vagas Iniciais
...
1518,L209,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Eletrotécnica e de Computadores,L1,220
1518,L221,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial,L1,130
1518,L233,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Física Tecnológica,L1,102
1518,L239,Universidade de Lisboa - Instituto Superior Técnico,Engenharia de Minas e Recursos Energéticos,L1,20
...
```

(a)

1	Codigo Inst	Codigo Cur	Nome da Instituicao	Nome do Curso	Grau	Vagas Iniciais
506	1518	L209	Universidade de Lisboa - Instituto Superior Técnico	Engenharia Eletrotécnica e de Computadores	L1	220
507	1518	L221	Universidade de Lisboa - Instituto Superior Técnico	Engenharia Aeroespacial	L1	130
508	1518	L233	Universidade de Lisboa - Instituto Superior Técnico	Engenharia Física Tecnológica	L1	102
509	1518	L239	Universidade de Lisboa - Instituto Superior Técnico	Engenharia de Minas e Recursos Energéticos	L1	20

(b)

Fig. 2: Extrato de um Ficheiro de Entrada com a Descrição dos Cursos: (a) visualização em ficheiro de texto; (b) visualização em Excel.

```
ID Candidato,PI,12,Nota Candidatura,E1,C1,E2,C2,E3,C3,E4,C4,E5,C5
1,18.5,15.6,17.05,7230,L101,0400,9351,0400,9740,1110,9847,3064,9104
2,20,16.5,18.25,0903,9015,3022,9010,6800,9189,3092,9147,0505,9139
...
59999,13.6,17.1,15.35,0300,9113,0602,8262,3101,L099,0400,9740,1103,L227
60000,14.8,17.1,15.95,1101,9554,3117,9870,3013,9890,6810,L282,3103,9457
```

(a)

1	ID Candida	PI	12	Nota Candi	E1	C1	E2	C2	E3	C3	E4	C4	E5	C5
2	1	18.5	15.6	17.05	7230	L101	400	9351	400	9740	1110	9847	3064	9104
3	2	20	16.5	18.25	903	9015	3022	9010	6800	9189	3092	9147	505	9139
60000	59999	13.6	17.1	15.35	300	9113	602	8262	3101	L099	400	9740	1103	L227
60001	60000	14.8	17.1	15.95	1101	9554	3117	9870	3013	9890	6810	L282	3103	9457

(b)

Fig. 3: Extrato de um Ficheiro de Entrada com a Descrição dos Candidatos: (a) visualização em ficheiro de texto; (b) visualização em Excel.

3.2. Estruturas de Dados e Algoritmo de Colocação

A estruturas de dados são uma opção de projeto. Contudo deixam-se aqui algumas sugestões:

- Defina uma estrutura que lhe permita representar os cursos
- Defina uma estrutura que lhe permita representar os candidatos
- Implemente o algoritmo considerando alocação estática
    - Valide o algoritmo com os exemplos disponibilizados, para o efeito, na *webpage* do projecto.
- Altere a implementação para considerar alocação dinâmica.

3.3. Ficheiros de Saída

Os resultados serão guardados em 4 ficheiros CSV com as seguintes designações: CNAES_Colocacoes.csv, CNAES_Completo.csv, CNAES_Cursos.csv e CNAES_NC.csv.
O ficheiro CNAES_Colocacoes.csv, fig. 4, deve conter a linha de cabeçalho indicada e a lista de todos os candidatos colocados por ordem ascendente de nota e pela mesma ordem de cursos do ficheiro de entrada Cursos.csv. Em caso de empate mantém-se a ordem dos candidatos no ficheiro de entrada. Para cada candidato colocado, escreve-se o identificador de candidato, nota de candidatura, opção em que foi colocado (1-5), instituição e curso onde foi colocado.

O ficheiro CNAES_Completo.csv, fig. 5, não deve conter linha de cabeçalho, mas deve ter uma linha por curso antes da lista de todos candidatos colocados nesse curso. A sequência de cursos deve ser a mesma que a apresentada no ficheiro de entrada Cursos.csv. A linha inicial de cada curso tem a instituição, nome do curso, número inicial de vagas, número de candidatos colocados, nota do último colocado. A linha de cada candidato colocado é igual à do ficheiro CNAES_Colocacoes.csv. A ordenação dos candidatos é a mesma que no ficheiro CNAES_Colocacoes.csv.

O ficheiro CNAES_Cursos.csv, fig. 6, deve conter a linha de cabeçalho indicada e deve conter apenas uma linha por curso, ou seja, não deve incluir os candidatos colocados. A sequência de cursos deve ser a mesma que a apresentada no ficheiro de entrada Cursos.csv. A linha de cada curso é igual à do ficheiro CNAES_Completo.csv.

O ficheiro CNAES_NC.csv, fig. 7, deve conter a linha de cabeçalho indicada e deve conter a lista dos candidatos não colocados por ordem crescente de indentificador.

```
Candidato,Nota,Opçao,Instituição,Curso
…
35812,18.799999,1,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Eletrotécnica e de Computadores
12283,18.850000,1,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Eletrotécnica e de Computadores
7594,19.000000,1,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Eletrotécnica e de Computadores
2600,19.400000,1,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Eletrotécnica e de Computadores
49427,15.500000,4,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial
27405,15.500000,3,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial
204,15.500000,1,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial
…
```

Fig. 4: Extrato de um Ficheiro de Saída com Lista de Candidatos Colocados.

```
…
Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial,130,132,15.500000
49427,15.500000,4,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial
27405,15.500000,3,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial
204,15.500000,1,Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial
…
```

Fig. 5: Extrato de um Ficheiro de Saída com informação sobre os Cursos e Descrição dos Candidatos.

```
Instituição,Curso,Vagas,Colocacoes,Nota_Mínima
…
Universidade de Lisboa - Instituto Superior Técnico,Engenharia Eletrotécnica e de Computadores,220,226,15.350000
Universidade de Lisboa - Instituto Superior Técnico,Engenharia Aeroespacial,130,132,15.500000
…
```

Fig. 6: Extrato de um Ficheiro de Saída com Informação sobre os Cursos.

```
Candidato,Nota
1,14.050000
2,14.050000
7,13.450000
12,13.750000
```

Fig. 7: Extrato de um Ficheiro de Saída com Lista de Candidatos não colocados, ordenados por ID crescente.

4. Interface de entrada/saída e parametrização do programa

De seguida descrevem-se os parâmetros de linha de comando que deverão ser interpretados pelo programa a desenvolver.

4.1. Parametrização do programa

O programa deverá ser invocado na linha de comando da seguinte forma:

$ ./CNAES [OPTIONS]

'**$**' é a prompt do Linux e "**./**" representa a directoria corrente
CNAES designa o nome do ficheiro executável contendo o programa desenvolvido.

[OPTIONS] designa a possibilidade de o programa ser invocado com diferentes opções de funcionamento

As **opções** de funcionamento são identificadas sempre como strings começadas com o caractere '-', e podem aparecer por qualquer ordem. De seguida, descrevem-se as várias opções disponíveis:

-h ajuda para o utilizador
-v valor número de vagas a considerar em cada curso
-n valor número de candidatos a considerar da lista de candidatos
-i filename nome do ficheiro entrada com lista de cursos
-c filename nome do ficheiro entrada com lista de candidatos
-o filename nome do ficheiro de saída com lista de colocações
-u filename nome do ficheiro de saída com lista de universidades e colocados
-m filenamenome do ficheiro de saída com lista de cursos
-x filename nome do ficheiro de saída com lista de não colocados

- A opção -h, quando invocada, deverá imprimir para stdout uma mensagem de ajuda de execução da linha de comandos. Para ver um exemplo deste tipo de mensagens, executar no terminal: "gedit -h".
- A opção -v indica o número máximo de vagas a considerar em cada curso. Quando se utiliza esta opção, não se consideram as vagas indicadas no ficheiro de cursos.
- A opção -n indica o número de candidatos a ler do ficheiro de candidatos (sempre a contar do começo do ficheiro)
- A opção -i indica o ficheiro de cursos a usar em alternativa a "cursos.csv".

- A opção -c indica o ficheiro de candidatos a usar em alternativa a “candidatos.csv”.
- A opção -o indica o ficheiro a usar em alternativa a “CNAES_Colocacoes.csv”.
- A opção -u indica o ficheiro a usar em alternativa a “CNAES_Completo.csv”.
- A opção -m indica o ficheiro a usar em alternativa a “CNAES_Cursos.csv”.
- A opção -x indica o ficheiro a usar em alternativa a “CNAES_NC.csv”.

4.2. Exemplos de Invocação do Programa

De seguida apresentam-se alguns exemplos válidos e inválidos de invocação do programa com especificação de parâmetros em linha de comando.

Exemplos válidos de invocação do programa:

Exemplo 1: ./CNAES
Exemplo 2: ./CNAES -v 20
Exemplo 3: ./CNAES -n 100
Exemplo 4: ./CNAES -v 10 -n 500
Exemplo 5: ./CNAES -v 10 -n 500 -i c.csv -c cand.csv
Exemplo 6: ./CNAES -i c.csv -c cand.csv
Exemplo 7: ./CNAES -v 10 -n 500 -i cs.csv -o lcol.csv -x lncol.csv

Exemplos inválidos de invocação do programa:

Exemplo 1: ./CNAES -v -20
Exemplo 2: ./CNAES -v -n
Exemplo 3: ./CNAES -v 10 -n 500 -i -c cand.csv

Nota: Pode recorrer à função de biblioteca getopt() para efetuar a validação dos parâmetros de entrada ou implementar uma função para realizar essa validação. Consulte a página do manual com o comando “man 3 getopt” num terminal Linux ou numa pesquisa no Google.

No caso de invocação inválida do programa, o programa deve escrever uma mensagem de erro e terminar.

5. Processo de Submissão

Os trabalhos submetidos deverão obrigatoriamente ser **compilados com as seguintes opções**:

-Wall -O3 -g

e correr na máquina virtual fornecida.

O projecto deve ser submetido no Fénix um **ficheiro ZIP** com:

(1) código comentado com as funcionalidades indicadas no enunciado (ficheiros .h e .c);
(2) Makefile para gerar o executável;

Para além disso, será necessário preencher uma **ficha de auto-avaliação no GoogleForms**.

Por fim, o código deve ser estruturado de forma lógica em vários ficheiros (*.c e *.h). As funções devem ter um cabeçalho curto, mas explicativo e o código deve estar corretamente indentado e com comentários que facilitem a sua legibilidade.

É importante reforçar que certos modos de operação do programa serão avaliados de forma automática, pelo que é imperativo que o programa respeite a impressão para stdout/ficheiro, a leitura de stdin/ficheiro. A falha na execução dos mesmos poderá levar a uma penalização na nota final.

6. Processo de Avaliação

A avaliação do trabalho terá a seguinte distribuição de cotações:

- Verificação da Solução com Alocação Estática (**Lab4**)
- Leitura dos ficheiros de entrada para estruturas de dados (20%)
- Colocação de Alunos
    - 40% se utiliza alocação dinâmica, sendo 10% para verificação com VALGRIND ou 20% se utiliza apenas alocação estática
- Escrita das estruturas de dados nos ficheiros de saída (20%)
- Opções de Linha de Comando (5%)
- Comentários (5%)
- Qualidade do código (10%)

Avaliação oral (**realizada após a entrega do projecto**):

- Grupos com **nota superior a 17** realizarão obrigatoriamente oral para defender a nota (a nota atribuída antes da oral será o ponto de partida e limite máximo da nota após a discussão oral)
- Grupos com **nota entre 8 e 17** só realização oral se o docente de laboratório ou o avaliador de projecto considerar necessário.
- Grupos com **nota inferior a 8** podem fazer oral para obter nota mínima de projecto.

7. Código de Honestidade Académica

Espera-se que os alunos conheçam e respeitem o Código de Honestidade Académica que rege esta disciplina e que pode ser consultado na página da cadeira. O projeto é para ser planeado e executado por grupos de dois alunos e é nessa base que será avaliado. Quaisquer associações de grupos ou outras, que eventualmente venham a ocorrer, serão obviamente interpretadas como violação do Código de Honestidade Académica e terão como consequência a anulação do projeto aos elementos envolvidos.

Lembramos igualmente que a verificação de potenciais violações a este código é feita de forma automática com recurso a sofisticados métodos de comparação de código, que envolvem não apenas a comparação direta do código, mas também da estrutura do mesmo.