Projeto Prático

Tratamento de Arquivos e Ordenação em Memória Secundária

1. Descrição e Objetivo

A finalidade deste trabalho é implementar um sistema que converta um arquivo CSV em binário e, posteriormente, trabalhe com ordenação em memória secundária. O projeto vale 10 pontos. O projeto deverá ser realizado em equipe de dois a três alunos, sendo possível montar equipes com membros de turmas diferentes. A equipe deve ser formada até a data limite especificada no Campus Virtual. Após a data limite as pessoas que não tiverem grupos serão agrupadas pelos professores de forma aleatória. A avaliação do projeto será realizada preferencialmente sobre o código e relatório entregues, mas os docentes podem solicitar a realização de entrevista com os membros de cada grupo, caso seja necessário.

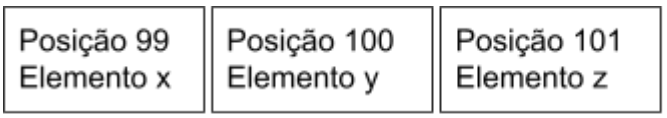
Para desenvolvimento deste trabalho, cada grupo utilizará uma base de dados em formato .csv sorteada entre seis possíveis. As bases foram construídas a partir de dados publicados na internet, e estão disponíveis para a realização do projeto prático no Google Drive. Exemplo dos dados em um dos arquivos disponíveis:

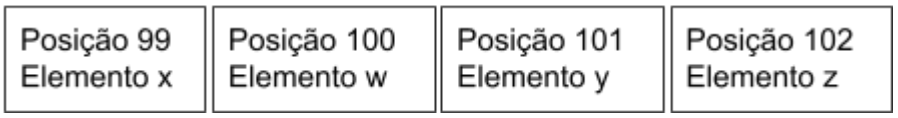
```
ID,Name,Team,Games,Year,Season
0,A Dijiang,China,1992 Summer,1992,Summer
1,A Lamusi,China,2012 Summer,2012,Summer
2,Gunnar Nielsen Aaby,Denmark,1920 Summer,1920,Summer
3,Edgar Lindenau Aabye,Denmark/Sweden,1900 Summer,1900,Summer
4,Christine Jacoba Aaftink,Netherlands,1988 Winter,1988,Winter
5,Christine Jacoba Aaftink,Netherlands,1988 Winter,1988,Winter
6,Christine Jacoba Aaftink,Netherlands,1992 Winter,1992,Winter
7,Christine Jacoba Aaftink,Netherlands,1992 Winter,1992,Winter
```

2. Desenvolvimento do projeto

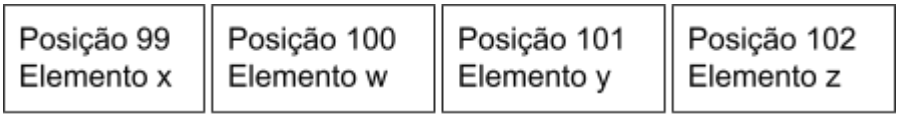
A primeira parte do projeto consiste no desenvolvimento de um sistema que converta a base sorteada para o grupo no Campus Virtual em arquivo binário, e também realiza algumas operações no arquivo. O grupo pode desenvolver um único sistema que pode gerar o arquivo binário e posteriormente trabalhar com esse arquivo binário nas operações citadas a seguir. Entre os tratamentos do arquivo binário, é esperado que a aplicação permita:

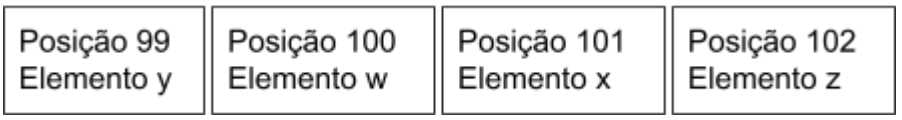
- **Adicionar um elemento no arquivo binário em uma posição específica qualquer.**
  Por exemplo, inserir um elemento w na posição relativa número 100 do arquivo:





- **Visualizar os registros entre a posição x e y.**
  Por exemplo, pode ser solicitado para visualizar os registros entre a posição 100 e 1234.
- **Alterar os dados de um registro em uma posição específica.**
- **Trocar dois registros de posição no arquivo.**
  Por exemplo, trocar os elementos x e y de posição:





- **Imprimir todos os registros do arquivo binário, na ordem em que estão armazenados.**

Observação: *as operações devem ser realizadas no arquivo binário, sendo permitido apenas carregar registros individuais para a memória. Ou seja, não é permitido carregar o arquivo inteiro na memória para realizar qualquer das operações solicitadas.*

Com o arquivo binário pronto, este deve ser ordenado. Cada grupo será sorteado com a base a ser utilizada, o método de ordenação a ser implementado e o tipo de ordenação (se crescente ou decrescente). O grupo deverá ler os dados do arquivo e o processo de armazenamento e ordenação deverá utilizar arquivos em formato binário. **O método de ordenação utilizado deve ter foco para uso em memória secundária** (em arquivos, portanto). Certifiquem-se de que exista uma forma de avaliação de ordenação (exibição dos dados ordenados, por exemplo).

Apenas UM dos membros do grupo deve acessar o sorteio. Mesmo que mais de um membro acesse o sorteio, o projeto do grupo será corrigido considerando as informações do primeiro sorteio realizado.

3. Instruções gerais para desenvolvimento e entrega do projeto prático

Recomendações gerais:
Deverão ser entregues o código fonte e um readme com informações importantes, como detalhes de compilação, entre outros. Além disso, deve ser entregue **um relatório** que deverá conter obrigatoriamente:

- ☐ Uma descrição de todas as estruturas de dados utilizadas;
- ☐ Uma descrição **em alto nível** (isto é, sem código) explicando a lógica do programa.

O trabalho deverá ser implementado utilizando C++ e ser passível de compilação e execução em um Linux genérico qualquer. Não é permitido o uso da STL ou bibliotecas similares sem a autorização dos professores. Isso é válido, principalmente, para elementos considerados centrais no trabalho. Também não se deve fazer uso de programação não-estruturada (como uso de "goto" ou mesmo "break" fora de "switch", uso de return para interromper laços de repetição, etc).

Boa organização, comentários bem feitos e indentação do arquivo contam pontos na nota final. Má organização e indentação retiram pontos, assim como comentários inúteis e sem sentido. Vocês têm liberdade para escolherem o estilo de indentação que julgarem mais adequado, mas respeitem o estilo escolhido. Recomendamos o uso de espaços para indentação, mas você pode usar tabulações, desde que não misture os dois tipos.

Uma implementação que apenas cumpre a obrigação recebe nota 80. Se você quer mais que 80, precisa avançar do que foi pedido, por exemplo:

- ❖ boa apresentação do projeto no geral (código e relatório);
- ❖ fazendo uma aplicação em que realmente as escolhas mostram-se bastante adequadas (ex: escolha das estruturas, uso adequado de entrada e saída de dados, etc.);
- ❖ utilizando soluções mais eficientes ou estruturas que já mantenham os dados ordenados;
- ❖ utilizando recursos de programação não solicitados, como controle de exceção, etc.
- ❖ entrega antecipada;
- ❖ implementação de recursos não solicitados, que mostrem esforço da equipe;
- ❖ relatório bem escrito, formal e **em formato pdf**;
- ❖ código bem organizado (ex: separação adequada do código em arquivos, cabeçalho e/ou uso de compilação separada com makefiles);

A entrega do projeto deve ser feita sempre através de um único arquivo compactado. (preferência por zip, 7z, tar.gz ou tar.bz -- **não usar rar**), sendo que o nome do arquivo deve conter os nomes dos alunos e o trabalho, por exemplo: joao_silva_maria-souza_jose_pereira-projeto_suplementar.tar.gz. Arquivos fora do formato não serão corrigidos. A atividade deverá ser enviada em espaço específico, criado para este fim no campus virtual.

Observação importante: Utilizaremos ferramentas de detecção automática de plágios nos trabalhos enviados. Trabalhos com suspeitas de plágio e uso de inteligência artificial serão zerados e os responsáveis poderão sofrer processo disciplinar junto à CSI-ICET.

Prazo de entrega: 18 de junho de 2025

4. Roteiro para correção

O projeto prático será avaliado de acordo com os critérios estabelecidos em sua descrição mediante avaliação de código e entrevista com os participantes. Seguem os elementos para vocês ficarem atentos durante o processo de correção.

Avaliação da Entrevista

A entrevista terá por objetivo não apenas esclarecer eventuais dúvidas, mas também conferir a participação efetiva de cada membro na realização do trabalho. Apesar disso, a correção será feita para o grupo e não individualmente. Ou seja, uma participação inadequada de um dos membros irá penalizar não apenas a si, mas todo o grupo envolvido.

A entrevista será realizada caso os professores entendam ser necessário. Nesse caso, será feito o pré-agendamento do dia e horário em que a mesma será realizada e todos os membros do grupo deverão estar presentes.

Durante a entrevista, os alunos serão indagados sobre trechos do código, sendo esperado que todos consigam responder sobre perguntas realizadas em trechos específicos. Os questionamentos buscarão avaliar o domínio de cada membro sobre o trecho destacado, envolvendo itens como objetivo, funcionalidade e alternativas para o código apresentado. Não será permitido que um aluno ou aluna responda a pergunta direcionada a outro membro do grupo.

De antemão, alertamos que usaremos a entrevista como elemento central do processo avaliativo. Ou seja, não é possível obter boa pontuação com resultado ruim durante a entrevista.

Avaliação do Código

A primeira avaliação sobre o código é realizada em sua execução **em Linux.** Portanto, o primeiro e maior critério é a correta execução do projeto nesse ambiente. Problemas como funcionamento inadequado e falhas de segmentação impactam negativamente, e de maneira significativa, a avaliação.

Espera-se ainda que a aplicação ordene corretamente os dados, de acordo com o que foi sorteado para o grupo. A interface da aplicação deverá permitir a execução dessas tarefas, sem dificuldades.

No que diz respeito ao código em si, serão verificados um conjunto de itens que garanta sua qualidade. Entre os itens analisados, verificamos se o código não infringe a programação estruturada e não faz uso de artifícios que tornam o código menos

legível ou mais difícil de manter. Como já informado, não queremos ver um único “goto” no trabalho. E “break” só é aceito em “switch”, sendo também não recomendado o uso de “continue”.

A mesma recomendação se aplica ao uso de repetições não-contadas usando for, por exemplo “for (int i = 0; i < 100 and naoTerminou; i++)”, que deveriam ser substituídas por “while” ou “do... while”, já que não se sabe quantas vezes o laço será executado. O uso desse tipo de código não só pesa negativamente no trabalho, como pode levar à sua anulação (nota zero ou simbólica). Em tempo, a mesma recomendação se aplica ao uso de return “vazio” para interromper um laço de repetição.

No que diz respeito à organização do código, esperamos que o código esteja distribuído em classes ou estruturas, ficando a critério do grupo sobre como definir qual o melhor conjunto de classes, atributos e métodos. É critério do grupo avaliar quais atributos são ou não públicos.

Dependendo da forma de organização das classes, recomenda-se que estas estejam em arquivos separados. Nesse sentido, o uso de compilação separada, usando Makefiles, é avaliada positivamente. Também é avaliado positivamente se o código está comentado adequadamente. Para isso, é importante que todo arquivo tenha um cabeçalho, contendo as informações do grupo e projeto. Arquivos sem esse cabeçalho pontuam negativamente no trabalho. Se o arquivo enviado também estiver fora do formato informado, isso também é avaliado negativamente.

Ainda no que diz respeito à organização do código, os comentários devem ser utilizados para informar para que servem os métodos principais das classes, caso o nome utilizado não deixe isso bem claro. Além disso, comentários devem ser utilizados para esclarecer trechos mais confusos do código. No que diz respeito à indentação, é exigido apenas que o grupo siga um padrão, sem misturar espaço e tabulação, por exemplo. Além dos elementos já comentados, e que podem gerar bonificação adicional, destacamos também o uso de controle de exceção de forma adequada, evitando que a aplicação gere problemas ao tratar situações inusitadas.

Avaliação do Relatório

Esperamos que o relatório complete o código. Não esperamos encontrar nele trechos de código, printscreen da interface ou qualquer cópia de informações de arquivos suplementares, a não ser que seja **extremamente** necessário para explicar uma dada solução. Assim, o relatório deve apresentar uma descrição em alto nível de todas as estruturas de dados utilizadas, bem como uma apresentação da lógica do programa, sendo recomendado o uso de diagramas e/ou fluxogramas para essa tarefa. **O texto precisa ser formal, com introdução, desenvolvimento e conclusão**.

Além do que foi citado anteriormente, o documento pode conter informações relevantes ao processo de compilação ou execução. Por ser considerado uma questão mais técnica, o grupo pode colocar essas informações em um arquivo readme.