UNIVERSIDADE FEDERAL DE LAVRAS
DEPARTAMENTO DE COMPUTAÇÃO APLICADA

Bárbara Ferreira Rodrigues **Matrícula: 202120532**
Lucas Gabriel Pereira Moreira **Matrícula: 202110677**
Michel Alexandrino de Souza **Matrícula: 202111084**

PRIMEIRA PARTE DO PROJETO PRÁTICO DA DISCIPLINA ESTRUTURA DE DADOS (GAC108): CONVERSÃO E MANIPULAÇÃO DE DADOS DE UM ARQUIVO CSV PARA BINÁRIO

O seguinte relatório tem como objetivo descrever e instruir a utilização do projeto que teve como objetivo a criação de um programa responsável pela conversão e manipulação de um arquivo .csv em um arquivo binário. Dentre seus arquivos e funções, ele é capaz de realizar a conversão e, após ela, inserir novos dados, trocar dados de posição, alterar dados em uma posição específica, visualizar os dados presentes entre duas posições e, por fim, mostrar ao usuário todos os dados presentes no arquivo binário. A BASE DE DADOS É **DATA_ATHLETE_EVENT**.

1 - MAIOR DADO DE CADA CAMPO

Para trabalhar com arquivos binários, é necessário trabalhar com vetores de caracteres em vez de strings e, para não alocar espaços desnecessários, primeiramente foi rodado um programa que passa por todo o arquivo binário, separa cada campo em um registro e confere qual o maior dado de cada campo em todas as linhas. Os detalhes da separação de cada campo serão demonstrados posteriormente nesse documento. O programa descrito encontra-se como `"encontra_maior_dado_cada_campo.cpp"`.

2 - PROGRAMA PRINCIPAL

O programa principal, nomeado “main.cpp”, chama as classes, as quais estão separadas em três arquivos. Os arquivos “leCSV.cpp” e “leCSV.h” fazem a operação de converter o arquivo CSV para binário; “operacoes.cpp” e “operacoes.h” fazem as operações requisitadas pelo programa main com o arquivo binário. E, por fim, existe o arquivo “estrutura.h”, que não possui métodos e é apenas uma definição global de um registro que será necessário a todos arquivos descritos.

Como são três arquivos separados, é necessário fazer a compilação manualmente; abaixo está descrita a diretiva para facilitar a compilação:

```
g++ main.cpp leCSV.cpp operacoes.cpp -o executavel
```

Assim que se dá a execução, o programa tentará converter o arquivo CSV para binário; caso o arquivo binário com o nome "csv_em_binario.dat" já exista, essa tentativa não ocorrerá, para que as modificações no binário não sejam perdidas a cada execução do programa. Caso seja a primeira vez que o programa é usado ou o usuário deletou o arquivo binário para resetar tudo, será chamada a classe "leCSV" para fazer então fazer as conversões.

3 - CLASSE leCSV

Aqui, o CSV é aberto e é criado o arquivo binário com o nome "csv_em_binario.dat". Caso ocorra algum erro em uma dessas duas tentativas, uma mensagem de erro será impressa e a execução será abortada. No caso de sucesso, é lida uma linha para descartar as informações das colunas do csv e, então, enquanto o arquivo não terminar cada linha, ela é lida e tem seus campos separados da seguinte maneira:

```
0,A Dijiang,Barcelona,Basketball,Basketball Men's Basketball,CHN
```

(encontra-se a posição do primeiro delimitador)

```
0,A Dijiang,Barcelona,Basketball,Basketball Men's Basketball,CHN
```

(é criada uma substring que vai da posição 0 até a posição do delimitador)

```
0,A Dijiang,Barcelona,Basketball,Basketball Men's Basketball,CHN
```

(deleta-se da posição 0 até a posição do delimitador mais um)

```
A Dijiang,Barcelona,Basketball,Basketball Men's Basketball,CHN
```

(resultado)

Esse processo ocorre 6 vezes (cada campo) e a próxima linha é lida.

Foi necessário colocar um controle para casos que a vírgula aparece no meio do campo, da seguinte maneira.

```
12969,Andrey Pavlovich Bakhvalov,Calgary,Speed Skating,"Speed Skating
Men's 1,000 metres",URS
```

Assim, caso o primeiro caracter dos campos nome ou evento sejam aspas duplas

```
"Speed Skating Men's 1,000 metres",URS
```

(exemplo após as separações e exclusões da linha anterior)

Deverá ser encontrado como delimitador um ‘",’ e não somente uma aspas duplas

```
"Speed Skating Men's 1,000 metres",URS
```

Após cada separação e preenchimento do registro, este então é inserido no arquivo binário.

4 - CLASSE OPERAÇÕES

Após tudo estar devidamente preparado, podem ser feitas as operações.

4.1 - INSERÇÃO EM POSIÇÃO:

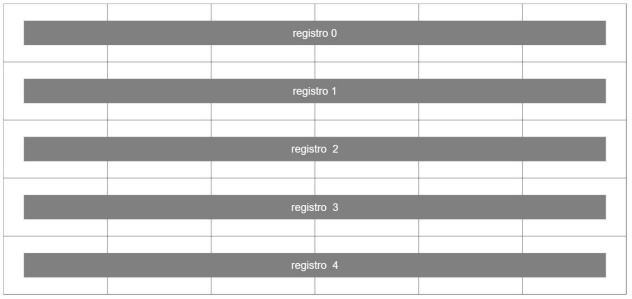
A tentativa foi de criar um arquivo auxiliar que recebe os dados até a posição do arquivo que queremos inserir, depois insere-se os novos dados nesse mesmo arquivo temporário e continua a copiar do arquivo original para o temporário a partir da posição que o elemento foi inserido até o final do original. Após isso é copiado do arquivo temporário para o original tudo novamente.

4.2 - OPERAÇÃO DE VISUALIZAÇÃO ENTRE DUAS POSIÇÕES ESPECÍFICAS:

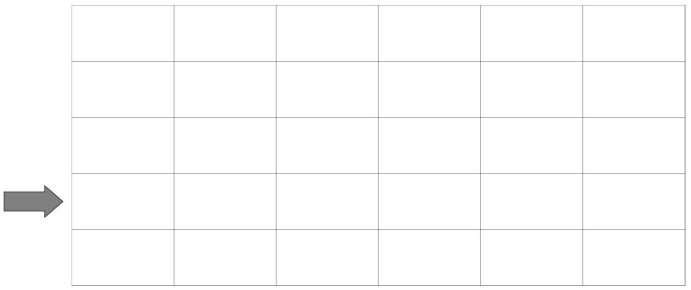
O usuário deverá entrar com duas posições e foi feito somente um método para fazer exibições. Existe um controle para o método exibir, ou todos os registros ou entre posições e essa decisão se dá durante a chamada do programa principal, mas não pelo usuário mas diretamente em código.

O que interessa nesse tópico é a exibição entre duas posições, para fazer isso o código primeiro posiciona a cabeça de leitura na posição desejada no binário. Isso é possível pois podemos saber o tamanho de um registro em bytes, então basta multiplicar a posição desejada pelo tamanho do registro em questão.

Para exemplificar, caso tivéssemos uma estrutura que possui um inteiro e um char de dois bytes (6 bytes) que foi registrada 5 vezes em um binário:



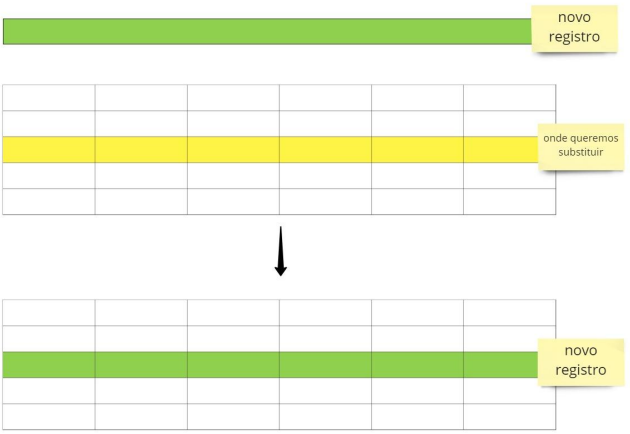
Aqui cada quadrado branco representa um byte no arquivo, e as tarjas cinzas são referentes aos tamanhos do registro de exemplo, caso quiséssemos acessar o 3 registro basta multiplicar 3 pelo tamanho da estrutura, 6. E então posiciona-se a cabeça de leitura na 18ª posição:



Voltando ao nosso programa, após feito o posicionamento da cabeça de leitura, o usuário é questionado se quer mostrar os registros em um arquivo (este arquivo será salvo contendo em seu nome o primeiro e ultimo id) ou na tela, é recomendado no arquivo pois podem ter visualizações grandes que podem não caber no prompt, no entanto existe a opção da tela para pequenas visualizações. A impressão ou inserção no arquivo ocorre até que o ID lido seja igual ao requisitado ou o arquivo acabe.

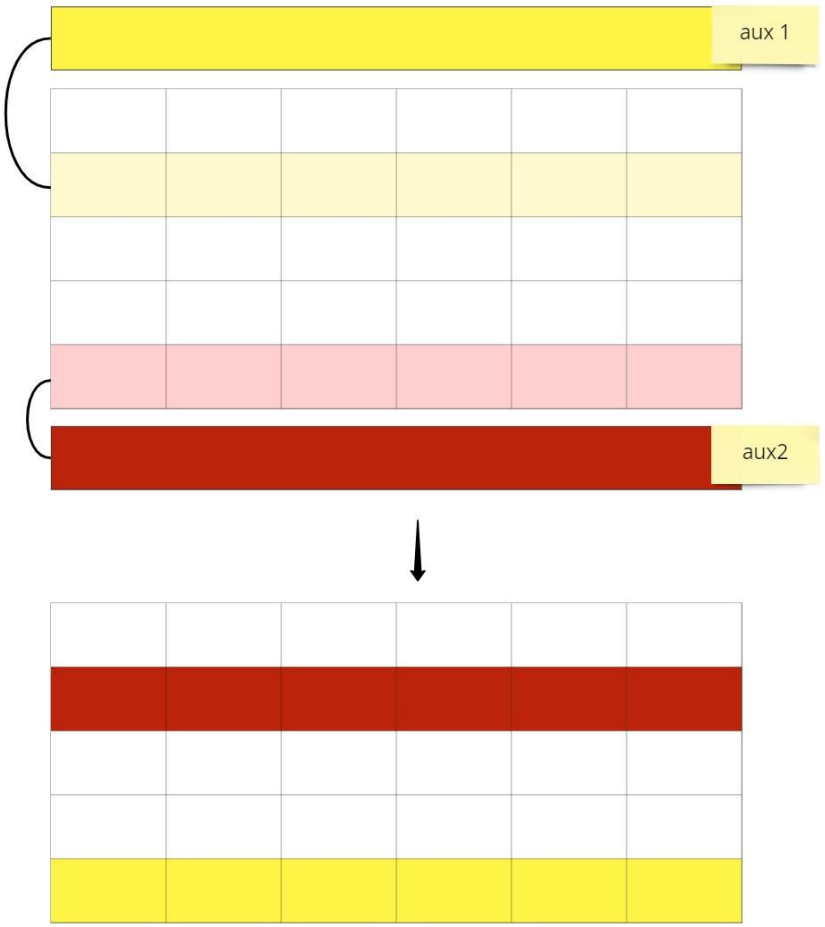
4.3 - ALTERAR OS DADOS DE UMA POSIÇÃO ESPECÍFICA:

Para fazer uma alteração em uma posição o usuário entra com a posição em questão e é feito o mesmo procedimento explicado anteriormente de posicionar a cabeça de leitura na posição desejada e simplesmente usamos a função que a própria biblioteca fstream disponibiliza para fazer a escrita em arquivo binário, essa função sobrescreve o que está escrito em uma certa quantidade de bytes, exemplo:



4.4 - TROCAR DOIS REGISTROS DE POSIÇÃO NO ARQUIVO

Para trocar os registros em duas posições é preciso fazer a operação de posicionamento da cabeça de leitura na primeira posição que o usuário digitou e salvar aqueles dados em um registro auxiliar, depois posicionar a cabeça de leitura em um segundo registro auxiliar e depois escrever o registro auxiliar 2 na posição 1 e o registro auxiliar 1 na posição 2:



4.5 - ESCREVER TODOS OS REGISTROS:

Este método é muito parecido com o descrito no item 4.2, tanto que é usada a mesma função na classe. A cabeça de leitura é posicionada em 0 e é inserido no arquivo (que agora será chamado de todos-registros) ou impresso na tela os registros, até que o binário chegue ao fim. Aqui é ainda mais recomendada a inserção em arquivo, mas ainda assim o usuário pode mostrar na tela.