Relatório Dynamic Spreadsheets

Projeto Final de Programação de Computadores II

João Victor Lopez Pereira,
William Victor Quintela Paixão,
Gustavo de Mendonça Freire,
Yuri Rocha de Albuquerque

8 de dezembro de 2023

Rio de Janeiro - RJ

Sumário

Autores e suas Responsabilidades

Decidimos que, para acelerar e facilitar o processo de criação de nosso trabalho como um todo, dividiríamos as tarefas de forma que todos contribuíssem com aquilo que tiverem maior interesse ou facilidade. Além disso, estabelecemos que todos deveriam contribuir com o código, mesmo que alguns integrantes mais que outros. Esses que não dessem tanta ênfase na criação do programa, ficariam responsáveis por tratar do relatório e de suas partes, como o fluxograma, casos de teste, dissertação sobre as dificuldades encontradas, entre outros.

Em primeira instância, pensamos em fazer um programa que gerenciasse um estoque, de tal forma que cada item tenha apenas nome e quantidade, mas decidimos posteriormente por expandir ainda mais essa ideia e fazer uma planilha, que ao abstrair, é praticamente um estoque, porém, com uma quantidade dinâmica de campos. Decidimos por implementar diversas funções de tal forma que nós abrangêssemos todos os conteúdos vistos nesse período. Sendo assim, segue abaixo o nome e *DRE* dos membros do grupo que realizaram esse trabalho e suas respectivas funções e responsabilidades na equipe.

João Victor Lopez Pereira, de *DRE 123317370*, ficou responsável por fazer o relatório, implementar a impressão das planilhas no terminal, para melhor visualização por parte do usuário, além de tratar dos casos de teste e seus resultados obtidos. Além disso, teve algumas tarefas que decidiu por dividir com William Victor Quintela Paixão, de *DRE 123089993*, que seria dissertar sobre as dificuldades encontradas, montar o fluxograma que representa as possibilidades de escolhas do usuário, e escrever a conclusão do trabalho. Além disso, William também ficou responsável por escrever a descrição do problema, implementar no programa a interface de usuário e criar a funcionalidade de criar e apagar planilhas.

Gustavo de Mendonça Freire, de *DRE 123102270*, ficou responsável primordialmente com a parte do código, assim como Yuri de Rocha Albuquerque, de *DRE 123166143*. Gustavo ficou responsável por implementar as funções de filtro, incorporar a migração de planilhas para listas, e montar o esqueleto inicial que, posteriormente, se tornaria a base para programa. Yuri ficou responsável por implementar um método do usuário adicionar, remover, e editar itens de determinada linha, além de permitir o usuário a adicionar e/ou remover determinada coluna por inteiro.

Descrição do Problema

No início, nós, como um grupo, tivemos a ideia de elaborar um programa de controle de estoque, mais voltado para a conjuntura comercial. Contudo, depois de algumas negociações, concordamos que poderíamos criar algo de escopo mais geral e que atendesse ao mesmo propósito da proposta anterior: o de organização de dados. Nesse sentido, uma aplicação para criação de planilhas de maneira dinâmica seria uma forma perfeita de concretizarmos nossos anseios.

Planilhas são usadas em todos os lugares, seja para montar uma mera lista de compras, seja como base para um banco de dados interativo complexo feito para fins de suporte à decisão em contextos empresariais. Todo mundo precisa, em algum momento, tratar de certo volume de informações de forma esquematizada e de fácil visualização, pontos esses que as planilhas melhor se aplicam. Entretanto, apesar da abundância de softwares especializados para criação e edição de planilhas, pouco se vê pessoas que os utilizem com proficiência. Consideramos que isso ocorra, principalmente, por causa da fraca intuição que se pode ter ao lançar mão desses programas, demandando aprendizado das funcionalidades inerentes a cada plataforma. Identificamos, pois, que o grande prejuízo que torna alguns indivíduos alheios aos benefícios das planilhas seja a dificuldade de usufruto dos famosos editores.

Sob essa perspectiva, tentamos bolar um jeito eficaz e prático que tornasse qualquer um apto a produzir planilhas, sem necessidade de conhecimentos técnicos. A solução que achamos foi o desenvolvimento de um programa, no próprio terminal, que apresentasse uma interface amigável para os leigos, interface essa atingida por meio de menus interativos que fossem guiando o usuário nas mais variadas possibilidades referentes ao gerenciamento de planilhas. Tudo isso seria acompanhado de um forte aparato lógico e computacional, através da implementação de um tipo abstrato de dado correspondente às planilhas em si, relacionado com outras estruturas auxiliares que operam nos mais variados aspectos de nosso projeto.

Unimo-nos como equipe para tornarmos essa ideia algo concreto. Esperamos que, com essa simples solução, possamos atingir resultados positivos e, o mais importante, agregar em algo na vida das pessoas, ajudando-as a se organizarem e trilharem seus caminhos rumo a uma realidade descomplicada e feliz.

Fluxograma

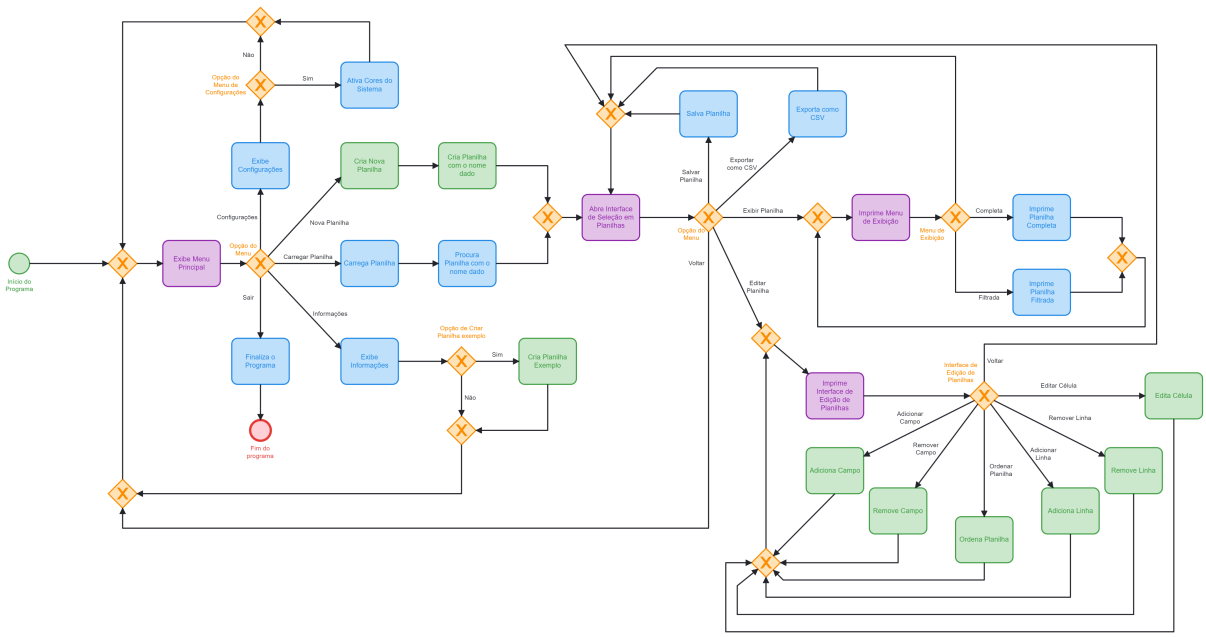

Figura 1: Fluxograma do Programa

Clique Aqui para abrir o Fluxograma no site do Cawemo. A senha é helloworld.

Recomendamos que veja o fluxograma a partir do site, já que é de manipulação melhor do que um documento PDF estático. Decidimos por adotar o seguinte sistema de cores para melhor visualização do fluxograma:

- Roxo: Apresentação de Interfaces de Escolha para o usuário.
- Azul: Operações que não modificam o estado da planilha.
- Verde: Operações que modificam o estado da planilha
- Amarelo: Tomadas de decisão.
- Vermelho: Finalização do processo.

Funções e suas Descrições

Buscando adotar uma abordagem mais estruturada e clara, optamos por dividir o código de forma modular, organizando-o em conjuntos distintos de funções, nomeadamente: *"filter.c"*, *"spreadsheets.c"*, *"interface.c"*, e *"main.c"*. Adicionalmente, incorporamos três bibliotecas de suporte: *"filter.h"*, *"spreadsheets.h"* e *"interface.h"*. Explicaremos sobre suas funções a seguir:

filter.h

Em filtro.h, declaramos alguns dos tipos necessários para o funcionamento das funções citadas abaixo, que são:

- Comparison

  Define um tipo enumerado para representar diferentes tipos de comparação que podem ser realizadas em consultas de filtragem na planilha

- Literal

  Representa um valor literal que pode ser de diferentes tipos, assim, sendo uma maneira flexível de representar valores de diferentes tipos na aplicação.

- Term

  Representa termos em uma expressão, podendo ser uma referência a uma coluna ou um valor constante, dependendo de seu valor booleano.

- Filter

  Proporciona uma maneira organizada de representar as condições do filtro do sistema, facilita a manipulação e avaliação das expressões de filtragem.

Além disso, também assinamos algumas das funções, presentes em *filter.c*, necessárias para que o código funcione normalmente.

filter.c

As funções contidas nesse trecho de código têm como função fornecer uma base sólida para que o usuário possa filtrar suas planilhas e abstrair informações nas quais não são

de vital importância para ele naquele determinado momento.

- `Literal evaluate(Spreadsheet s, char *entries, Term t);`

  Avalia um termo, consultando, se necessário, sua linha '`entries`' na planilha '`s`'.

- `double literalAsDouble(Literal a);`

  converte um tipo de literal, desde que não seja `string`, para um `double`.

- `bool compare(Literal a, Literal b, Comparison op);`

  Compara dois literais e retorna verdadeiro se ambos são iguais, se o primeiro é maior do que o segundo ou se o segundo é maior do que o primeiro, dependendo do operador de comparação 'op'. Caso contrário, retorna falso.

- `bool checkCondition(Spreadsheet s, char *entries, Filter cond);`

  Checa se uma linha satisfaz a condição cond do filtro ao comparar o termo com o seu adjacente a esquerda e a direita.

- `bool inputToFilter(Spreadsheet s, char *buffer, Filter *cond);`

  Transforma uma expressão textual para uma variável que o filtro aceite.

- `void filterSpreadsheet(Spreadsheet *s, Filter cond);`

  Remove as linhas que não satisfazem um filtro.

Interface.h

Em *interface.h*, não tivemos a declaração de novos tipos, somente a assinatura de algumas funções necessárias para que o código funcionasse.

Interface.c

As funções contidas nesse trecho de código garantem que o usuário tenha um contato mais prazeroso com o programa, já que essas servem o propósito de deixar o visual do terminal mais atraente, simples, intuitivo e interativo.

- `void clearTerminal();`

  Limpa o terminal usando comandos dependendo da verificação de sistema operacional do usuário.

- `void fill(char c, int n);`

  Imprime o caractere `c` `n` vezes, usado no escopo de outras funções.

- `void title(const char title[], const wchar_t wtitle[]);`

  Imprime um título `title` no terminal com uma borda estilizada ao redor, aceita strings que contém acentos.

- `int menu(const char options[][81], const wchar_t woptions[][81], int quant);`

  Imprime opções para que o usuário possa escolher ao digitar o respectivo número de sua escolha no terminal. A própria função já defende a entrada do usuário. Aceita acentos.

- `int titleMenu(const char msg[], const wchar_t wmsg[], const char options[][81], const wchar_t woptions[][81], int quant);`

  Uma junção das funções `title` e `menu`, ela imprime na tela um título `msg` e logo após, dá ao usuário opções para que ele possa escolher, de tal forma que tudo esteja estilizado, com tamanhos e defesas dinâmicas, e também aceita acentos.

spreadsheets.h

Nessa biblioteca, assinamos diversas funções contidas em *spreadsheets.c* definimos três novos tipos:

- `Type`

  Define um tipo enumerado para representar os tipos suportados pelas planilhas, que, no caso, são: `bool`, `string`, `int`, e `double`;

- `Row`

  Representa as linhas da planilha, é composto de dois ponteiros, um para o primeiro elemento da linha, e outro para a próxima linha. Quando a próxima linha for `NULL`, significa que a atual é a última.

- `Spreadsheet`

  Representa as planilhas, contém a quantidade de linhas e colunas, os nomes e tipos das colunas, e um ponteiro pra primeira coluna.

spreadsheets.c

As funções contidas nesse trecho de código são focadas na manipulação da planilha, acesso às colunas, linhas, células, exibição e etc.

- `__attribute__ ((noreturn)) void badType(Type t, char *functionName);`

  Fecha o programa quando se encontra um tipo em um lugar inesperado, de tal forma que nós, desenvolvedores, possamos saber em que função foi encontrado e também qual foi o tipo encontrado.

- `int sizeOfType(Type t);`

  Retorna a quantidade de bytes de um tipo recebido. Decidimos por representar a `string` usando 81 bytes (80 caracteres mais o caractere de terminação).

- `int columnIndex(Spreadsheet s, char *name);`

  Retorna o índice de uma coluna, dado seu nome.

- `int columnOffset(Spreadsheet s, int col);`

  Obtém o índice dos dados da coluna `'col'` em uma linha qualquer.

- `Row *getRow(Spreadsheet s, int rowIndex);`

  Retorna um ponteiro do tipo Row de uma linha, dado o índice recebido.

- `char *rowEntries(Spreadsheet s, int rowIndex);`

  Retorna um ponteiro para os dados de uma linha, dado o índice recebido.

- `int rowSize(Spreadsheet s);`

  Retorna o tamanho em bytes de uma linha da planilha.

- `int dataSize(Spreadsheet s);`

  Retorna o tamanho em bytes de todos os dados de uma dada planilha.

- `char *getCell(Spreadsheet s, int row, int col);`

  Retorna um ponteiro para uma célula, dado o índice da linha e da coluna.

- `void printCell(Type t, char *cell, FILE *file)`

  Imprime o conteúdo de uma célula em um arquivo de acordo com seu tipo.

- `void printCellByIndex(Spreadsheet s, int row, int col, FILE *file);`

  Imprime o conteúdo de uma célula em um arquivo dado seus índices.

- `void checkOpen(FILE *file, char *file_name);`

  Garante que um arquivo foi aberto com sucesso, caso não tenha sido, fecha o programa.

- `void initializeSpreadsheet(Spreadsheet *s);`

  Inicializa a planilha ao atribuir valores às suas variáveis.

- `void writeToFile(Spreadsheet s, char *file_name)`

  Imprime uma planilha em um arquivo em formato binário.

- `bool readFromFile(Spreadsheet *s, char *file_name);`

  Lê uma planilha de um arquivo binário.

- `void freeRow(Row *row);`

  Libera o espaço de uma linha.

- `void freeRows(Row *row);`

  Libera o espaço de todas as linhas da planilha

- `void freeSpreadsheet(Spreadsheet s);`

  Libera os espaços da planilha aberta pela função `readFromFile`.

- `void deleteRow(Spreadsheet *s, Row **prev, Row *curr);`

  Remove uma linha da planilha.

- `void deleteRowByIndex(Spreadsheet *s, int pos);`

  Remove uma linha da planilha dado seu índice.

- `void addRow(Spreadsheet *s, int row1);`

  Adiciona uma linha após outra dada seu índice.

- `void updateCellValue(Spreadsheet s, int row, int col);`

  Atualiza valor de uma célula específica dado seus índices.

- `void addColumn(Spreadsheet *s, char *colName, Type type);`

  Adiciona uma coluna à planilha.

- `void removeColumn(Spreadsheet *s, int col);`

  Remove uma coluna da planilha.

- `void ascendingSortByValue(Spreadsheet *s, int col);`

  Ordena as linhas de uma planilha pelo valor das células em ordem não-decrescente.

- `void descendingSortByValue(Spreadsheet *s, int col);`

  Ordena as linhas de uma planilha pelo valor das células em ordem não-crescente.

- `void sortByAlphabet(Spreadsheet *s, int col);`

  Ordena as linhas de uma planilha pelo valor das células em ordem alfabética.

- `void exportAsCsv(Spreadsheet s, char *file_name);`

  Exporta a planilha para formato *CSV* (Comma Separated Values) para poder ser acessada por outros programas

- `Spreadsheet example();`

  Cria uma planilha aleatória na qual o usuário pode ter contato pelos menu de informações. Serviu de grande ajuda enquanto programávamos o programa, já que precisamos testar casos e procurar por erros.

- `void displaySpreadsheet(Spreadsheet s);`

  Imprime a planilha no terminal.

main.c

- `int main();` O arquivo *main.c* é onde nossa função principal main está contida. Na main, fizemos com que ocorre o contato do usuário com as funções as quais ele tem acesso.

- `void newSpreadsheet(Spreadsheet *mainSpreadsheet, char *name, char *fileName, bool *createdNow);`

  Mostra ao usuário a opção de criar uma nota planilha.

- `void openExistentSpreadsheet(Spreadsheet *mainSpreadsheet, char name, char *fileName, bool *backToMenu);`

  Possibilita ao usuário escolher qual planilha acessar.

- `void addField(Spreadsheet *mainSpreadsheet, char *myColumnName, Type *myColumnType, char *name);`

  Imprime pro usuário a opção para usar a função de “Adicionar Campo”.

- `void removeField(Spreadsheet *mainSpreadsheet, char *myColumnName, int *columnPosition);`

  Imprime pro usuário a opção para usar a função de “Remover Campo”.

- `void addLine(Spreadsheet *mainSpreadsheet, char *buffer, int rowPosition);`

  Imprime pro usuário a opção para usar a função de “Adicionar Linha”.

- `void deleteLine(Spreadsheet *mainSpreadsheet, char *buffer, int *rowPosition);`

  Imprime pro usuário a opção para usar a função de “Remover Linha”.

- `void editCell(Spreadsheet *mainSpreadsheet, char *buffer, int columnPosition, int *rowPosition);`

  Imprime pro usuário a opção para usar a função de “Editar Célula”.

- `void sortSpreadsheet(Spreadsheet *mainSpreadsheet, char *name, char *myColumnName, int *columnPosition);`

  Imprime pro usuário a opção para usar a função de “Ordenar Planilha”.

- `void completeDisplay(Spreadsheet *mainSpreadsheet, char *name);`

  Imprime pro usuário a opção para usar a função de “Imprimir Planilha Completa”.

- `void filteredDisplay(Spreadsheet *mainSpreadsheet, char *name, char *fileName);`

  Imprime pro usuário a opção para usar a função de “Imprimir Planilha Filtrada”.

- `void showSpreadsheet(Spreadsheet *mainSpreadsheet, char *name, char *fileName);`

  Imprime pro usuário o menu de exibição de Planilhas.

- `void exportSpreadsheet(Spreadsheet *mainSpreadsheet, char *name);`

  Imprime pro usuário a opção para usar a função “Exportar Planilha como CSV”.

- `void saveSpreadsheet(Spreadsheet *mainSpreadsheet, char *name, char *fileName);`

  Imprime pro usuário a opção para usar a função de “Salvar Planilha”.

- `void information(Spreadsheet *mainSpreadsheet);`

  Imprime pro usuário o menu de informações.

- `void configurations(bool *colored);`

  Imprime pro usuário o menu de troca de cores.

Bibliotecas Externas

Além das funções que criamos, fizemos uso de algumas funções de bibliotecas padrões da linguagem C, que são:

- `stdio.h`
- `stdlib.h`
- `stdbool.h`
- `string.h`
- `wchar.h`

Casos de Teste Realizados e seus Resultados

Menu Inicial

Ao entrar no programa, o usuário é apresentado com 4 opções diferentes, a primeira, “Nova Planilha”, a segunda, “Carregar Planilha”, a terceira, “Informações”, a quarta, “Configurações”, e a quinta, “Sair”. Trataremos das cinco opções a seguir, em ordem decrescente.

Antes disso, tentaremos digitar alguma coisa que não esteja contida no intervalo esperado pelo desenvolvedor, ou seja, algum número ou caractere que seja diferente de 1, 2, 3, 4 e 5. Ao digitar 76, o programa imprimiu na tela “Escolha inválida!”, ao digitar “teste de entrada”, o programa se comportou da exata maneira, assim, mostrando estar defendido.

Sair

Ao selecionar a Quinta opção (Sair), o programa se encerra, assim como é de se esperar.

Configurações

Ao selecionar a Quarta opção (Configurações), o usuário é apresentado com a opção de Ativar a interface colorida do programa. Caso selecione que sim, o programa ativará as cores, caso selecione que não, o programa permanecerá com as cores padrões. Caso o usuário digite qualquer outra coisa que não seja “1”ou “2”(que correspondem a sim e não, respectivamente), o programa dará ao usuário outra chance de escolher. Após isso, o programa retornará ao menu.

Informações

Caso se selecione a terceira opção (Informações), o programa exibe ao usuário uma descrição de suas funcionalidades, uma explicação a respeito de como usá-lo, a intenção por trás de sua criação e também o nome dos desenvolvedores. Além disso, aparece um pequeno menu oferecendo ao usuário se ele deseja criar uma planilha de exemplo, ou não. Caso a resposta seja que não, o usuário retornará ao menu inicial, caso selecione que sim, aparecerá uma mensagem dizendo que uma planilha com o nome “exemplo”foi criada e guardada em “exemplo.data”, em seguida, o usuário retorna ao menu inicial.

Carregar Planilha

Ao selecionar a segunda opção (Carregar Planilha), é solicitado ao usuário o nome da planilha que ele deseja acessar. Caso o usuário não tenha criado uma planilha até aquele momento e tente digitar três nomes que não são encontrados na base de dados do sistema, aparece uma opção pro usuário se deseja retornar ao menu inicial ou continuar tentando acessar alguma planilha pelo seu nome. Caso selecione que deseja retornar, o programa se comporta como esperado. Mas caso o usuário deseje continuar tentando, o programa volta para a opção do usuário continuar digitando o nome da planilha que deseja acessar. Caso ele digite algum nome que finalmente foi encontrado no sistema, como "exemplo", caso ele tenha desejado criá-la no menu de informações, o sistema levará o usuário a uma tela que chamaremos de "Interface de Seleção em Planilhas", que trataremos em breve nessa mesma seção do documento.

Nova Planilha

Caso o usuário selecione a primeira opção (Nova Planilha), é pedido para que ele insira o nome da planilha que deseja criar. Também é solicitado ao usuário que digite o nome sem acentos, já que o comportamento do *UTF-8* utilizado em nossos compiladores, não se comporta de maneira previsível em diferentes sistemas operacionais. Ao digitar um nome aleatório, como "teste", o programa avança para a próxima tela de interação, que chamaremos de "Interface de Seleção em Planilhas", a mesma a qual o usuário foi levado após inserir o nome de sua planilha.

Interface de Seleção em Planilhas

Nesse momento, o usuário é apresentado com uma tela na qual há cinco opções para se escolher, sendo elas, enumeradas de um a cinco, a primeira, "Editar Planilha", a segunda "Exibir Planilha", a terceira, "Exportar Planilha como CSV", a quarta, "Salvar Planilha"e por último, a quinta, "Voltar". Assim como no Menu Inicial, trataremos das opções possíveis em ordem decrescente.

Assim como no menu anterior, o código está defendido, tal que se o usuário digitar uma frase, ou um número ou algum outro caractere que não está compreendido no intervalo solicitado pelo programa, recebe a mensagem "Escolha Inválida!".

Voltar

Começando pela quinta opção (Voltar), se o usuário selecionar essa opção, ele é redirecionado ao menu no início do programa, sobre o qual já falamos.

Salvar Planilha

Caso o usuário escolha a quarta opção (Salvar Planilha), supondo que o nome de nossa planilha seja "teste", o programa diz que o arquivo "teste.data"foi salvo com sucesso. Nesse momento, um arquivo com esse nome aparece na pasta na qual o código do programa está inserido, e o usuário é retornado ao menu que estava anteriormente.

Exportar Planilha como CSV

Caso selecione a terceira opção (Exportar Planilha como CSV), uma mensagem dizendo que o o arquivo "teste.csv"foi criado com sucesso é impressa no terminal. Nesse momento, um arquivo com esse mesmo nome surge na pasta na qual o código está inserido. Ao abrir esse arquivo, de fato, trata de uma planilha em formato CSV. Após isso, o usuário retorna ao menu em que estava anteriormente.

Exibir Planilha

Ao selecionar a segunda opção (Exibir Planilha), nos é dito que não é possível que nossa planilha seja exibida pois não há dados inseridos nela. Caso o usuário esteja com a planilha "exemplo", gerada no menu informações, ou com uma outra planilha na qual ele criou anteriormente, desde que haja conteúdo nela, ao usuário solicitar para que a planilha seja exibida, aparece um pequeno menu perguntando se o usuário deseja voltar, ver a planilha filtrada ou ver a planilha completa. Caso o usuário selecione voltar, ele retorna ao menu em que estava anteriormente. Caso selecione "Planilha Completa", a planilha é impressa no terminal de forma concisa e organizada, com colunas de tamanho igual, em seguida, o usuário volta ao pequeno menu onde tem acesso às três possíveis opções novamente. Ao usuário selecionar a segunda opção(Planilha Filtrada), é impresso na tela instruções de como usar o filtro. Ao tentar digitar algo aleatório para o programa interpretar como filtro, é dito que ao menos um campo deve ser selecionado. Os possíveis campos são os mesmos tipos que a planilha aceita, que são: "Bool", "String", "Int"e "Double". Ao digitar "Bool", por exemplo, nos diz que o valor do literal deveria ser um booleano, em outras palavras, ele está nos dando esse aviso pois deixamos esse valor vazio. Segundo as orientações nas informações, devemos seguir o formato "Campo Operador Literal/Campo", nessa ordem. E como operadores, temos as opções: "Maior que, Maior ou igual a, Menor que, Menor ou igual a, igual a". Após tentarmos digitar coisas que não fazem sentido e o programa não ser prosseguido, digitamos "bool = verdadeiro", o programa nos imprime a planilha tal que somente as linhas em que bool tem o valor de verdadeiro aparecem, enquanto as linhas em que o bool com valor falso não são exibidas. Ao digitarmos informações coerentes mas que são casos que não aparecem na planilha que selecionamos, por exemplo, ao digitar "int = 10"e "bool = falso", é exibido o cabeçalho da planilha mas nenhuma linha, já que esse caso não existe em nosso programa. Ou seja, é possível que se adicione quantos filtros quiser, sejam esses envolvendo int, double, string ou bool.

Editar Planilha

Ao usuário selecionar a primeira opção (Editar Planilha), o usuário é apresentado a mais um menu, o qual chamaremos de "Interface de Edição de Planilhas".

Interface de Edição de Planilhas

Nesse menu, são apresentadas sete opções diferentes para o indivíduo, que são: A primeira, "Adicionar Campo", a segunda, "Remover Campo", a terceira, "Adicionar Linha", a quarta, "Remover Linha", a quinta, "Editar Célula", a sexta, "Ordenar Planilha", e a sétima, "Voltar". Veremos cada uma delas em ordem decrescente.

Assim como em todos os menus anteriores, caso o usuário digite qualquer coisa que não esteja compreendida no intervalo de 1 a 7, aparece a mensagem de erro e é dado mais uma oportunidade para que ele escolha uma opção.

Voltar

Ao selecionar a sétima opção (Voltar), o usuário volta ao menu que falamos na seção anterior.

Ordenar Planilha

Ao selecionar a sexta opção (Ordenar Planilha), é solicitado que o usuário digite qual o campo que ele deseja ordenar. Ao escrever algo aleatório, o programa diz que não há campo com esse nome, e dá mais uma possibilidade para que se possa escrever o campo desejado. Usando a planilha de exemplo, decidi por ordenar o campo "int", após isso, o programa dá a opção de escolher ordenar de forma crescente ou decrescente. Após selecionar qualquer uma das duas, o programa imprime corretamente a planilha, como se tivesse trocado as linhas de lugar. Caso eu decida por querer ordenar o campo "str"ao invés de "int", o programa imprimirá a planilha ordenada em ordem alfabética. Ao digitar "double"no lugar do campo, é dado as mesmas opções que ao digitar "int", e ao digitar "bool", o programa diz que não é possível ordenar a partir do campo booleano. Após isso, o usuário retorna ao menu.

Editar Célula

Ao selecionar a quinta opção (Editar Célula), o programa diz qual a posição da célula que eu desejo modificar (linha e coluna). Caso eu digite 5 e 8, que são índices não presentes na planilha, o programa pede para que eu digite coordenadas existentes, o mesmo ocorre se eu digitar um texto aleatório. Finalmente, ao digitar índices existentes, como 1 e 1, o programa diz o valor atual que está nessa célula e pede para digitarmos qual o novo valor que queremos inserir. Ele também diz o tipo desse campo, e, consequentemente, demanda que respeitemos essa limitação. Na planilha de exemplo, o campo 1 1 é um inteiro, portanto, ao tentar digitar um texto, o programa pede para que digitemos um valor inteiro válido. As mesmas defesas ocorrem ao tentarmos digitar textos na coluna de tipo Double. Na coluna de tipo booleano, o programa defende todas as entradas que não sejam string que diz"verdadeiro"ou "falso". Após a alteração, o usuário retorna ao menu.

Remover Linha

Ao selecionar a quarta opção (Remover Linha), o programa pede qual o índice da linha a ser removida, inclusive, ele diz as opções que temos de 0 até n linhas disponíveis para serem removidas. Se digitarmos algo que não esteja nesse intervalo numérico, o programa pede para que se digite um número válido e dá outra oportunidade ao usuário. Finalmente, ao digitarmos um número nesse intervalo, o programa irá apagar essa linha de índice escolhido e o usuário irá retornar ao menu.

Adicionar Linha

Ao selecionar a terceira opção (Adicionar Linha), o programa pede o índice da linha a ser adicionada. Caso o usuário selecione um índice 'n' no qual uma linha já existe, essa nova linha será inserida na posição 'n' solicitada e a linha que estava ali antes irá se tornar a linha do índice 'n+1', e o mesmo ocorrerá para as próximas linhas com índice maior que 'n+1'. Após isso, o programa pede para que o usuário insira os valores que devem estar naquela linha dado seus campos, todos muito bem definidos e, assim como anteriormente no código, a defesa de só poder escrever algo correspondendo ao seu tipo é respeitado. Após isso, o usuário retorna ao menu.

Remover Campo

Ao selecionar a segunda opção (Remover Campo), o programa pede o nome do campo a ser excluído. Já que cada coluna é representada pelo seu nome, nós, desenvolvedores, achamos que seria a forma mais fácil do usuário saber o que gostaria de remover. Ao inserir um nome aleatório, o programa diz que não há campo com esse nome na planilha. Após digitar um nome válido, o programa diz que o campo foi removido, e o usuário retorna ao menu.

Adicionar Campo

Ao selecionar a primeira opção (Adicionar Campo), o programa pergunta qual será o nome desse novo campo, e pede ao usuário para que não use acentos. Ao digitar qualquer coisa aleatória, o programa pegará esse nome e o colocará como o nome do campo, seja uma string ou um número. Essa escolha garante versatilidade e maior liberdade para o usuário poder nomear seus campos como preferir. Após isso, o programa pergunta qual o tipo do campo, dando ao usuário escolhas de um a quatro, variando entre os quatro tipos implementados em nosso programa. Ao digitar algo aleatório ou um número não contido no intervalo, o programa diz que a escolha é inválida e aguarda uma nova entrada correta. Ao inserir um número válido, o programa começa a percorrer as posições do campo e pede para o usuário digitar o que deve estar naquele campo, defendendo as entradas de acordo com seus tipos. Após isso, o usuário retornará ao menu.

Esses são todos os menus e todas as opções que o usuário pode escolher no código. Como vimos, todos os campos estão muito bem defendidos, fazendo com que o usuário não fique preso em um *bug* por conta de ter digitado algo errado. Além disso, vimos também que o programa conta com menus bem explicados e informativos, facilitando a navegação do usuário. Todos os casos de teste funcionaram e o programa se comportou conforme esperado.

Dificuldades Encontradas

Não encontramos dificuldades expressivas que fossem atrapalhar o rendimento do trabalho. Logo quando o projeto fora anunciado, já decidimos quais seriam os integrantes do grupo e vislumbramos um pouco sobre qual problema trataríamos. Depois, fixamos uma meta e dividimos as tarefas entre os membros, podendo, logicamente, haver a colaboração mútua interoperacional. Gustavo e Yuri ficaram encarregados do esqueleto do código, enquanto João e William trataram da documentação do projeto, como explicitado no começo deste relatório.

Um dos obstáculos que encontramos foi a escolha dos tópicos da disciplina que deveríamos incluir no programa. Da premissa de construção de planilhas internamente e externamente à aplicação, vêm, diretamente, os conceitos de manipulação de dados, em listas ligadas, e arquivos, binários e de texto, além da alocação dinâmica de memória e uso de ponteiros. No entanto, tivemos de nos desdobrar para encontrarmos usos para algoritmos de busca e ordenação em nosso escopo. No fim, decidimos que seriam implementadas a pesquisa sequencial (a estrutura de lista ligada nos impediria de montarmos um algoritmo mais eficiente de busca, como a binária) e o ordenamento em bolha (por motivo parecido, temos maior alcance dos elementos próximos no *Bubble Sort* do que nas demais formas estudadas).

Outra questão vivida pelo time de desenvolvimento do *"back-end"* do programa foi a exigência de tratamento do mais baixo nível para o funcionamento de algumas funções associadas a algumas estruturas avançadas, visto que a linguagem C não oferece um suporte tão direto para a sua manipulação. Isso fez com que os colegas tivessem que procurar novos procedimentos e especificidades do C para que pudessem lograr seus objetivos. Com efeito, essa perseguição do aprendizado trouxe uma evolução bastante interessante para os nossos integrantes.

Quanto ao documento, os membros do grupo já apresentavam experiência com o uso do LaTeX, porém, uma dificuldade encontrada foi na hora de apresentar as funções. Não encontramos uma maneira fácil de colorir os tipos das funções e das variáveis. Achamos que seria de grande importância por melhorar a legibilidade do documento. Existe um pacote de nome "minted"que auto-detecta os elementos das linguagens de programação, porém, por termos criado diferentes tipos usando o typedef, a maioria deles não foi detectada, então optamos por incluir as cores de forma manual, o que causou cansaço e um esforço que poderia facilmente ser contornado a partir da introdução de uma possível função do pacote "minted"que deixasse que os usuários declarassem tipos.

O momento da programação da interface de usuário, apesar de envolver uma codificação de nível mais alto que a desenvolvida no *"back-end"*, foi desafiadora porque demandou integração das inúmeras funções montadas em uma forma palpável ao usuário. Assim, foi necessária, por vezes, uma adaptação dos códigos das bibliotecas mais basilares do programa de acordo com as possíveis demandas do utilizador levantadas por nós. Foi nessa etapa, também, que ficaram evidentes algumas incorreções no que já tinha sido escrito, levando em conta que é no processo de *"front-end"* que o programa em si se materializa visualmente. A constante preocupação com a defesa das entradas do usuário configurou-se, juntamente com os pontos citados, uma importante questão. Afinal, era preciso que adequássemos a experiência de quem fosse usar o programa com tudo o que fora construído internamente, sem, no entanto, que ela se tornasse complicada demais. Para tal, foi preciso tratar todos os *inputs* do programa como cadeias de caracteres para que, em seguida, fossem executadas as conversões e operações necessárias. A defesa, inclusive, teve de ser constantemente atestada através de dezenas de testes.

Uma última situação que trouxe relativa atenção à equipe, mesmo que não tenha sido tão difícil, foi a conciliação dos trabalhos empenhados por cada membro. Como as atividades foram segmentadas entre nós, tivemos que decidir uma maneira eficiente de atualizar e acompanhar o projeto. Cada um deveria poder contribuir com o código em seu tempo disponível e, simultaneamente, todos deveriam ter acesso ao conteúdo atualizado. Observando essa realidade, optamos por lançar mão da plataforma de versionamento de código *Git*, principalmente em sua versão online: o *GitHub*. Por meio dele, fomos capazes de ajudar da melhor maneira na evolução do programa, com cada um podendo estar ciente do que o outro fez de modo detalhado e completo. Esse fator contribuiu bastante para o sucesso do trabalho.

Conclusão

Trabalhamos duro para a incorporação de nossa ideia de acordo com os mecanismos da linguagem C. Foi uma maneira divertida e intrigante de aplicar os conhecimentos adquiridos ao longo de nossa trajetória acadêmica, tanto na disciplina de Programação de Computadores I quanto na atual, para satisfação não só dos interesses próprios, como também dos da sociedade. É bacana imaginar como seria se fôssemos um time de desenvolvedores em uma empresa real trabalhando em uma solução inovadora.

Com esse trabalho, tivemos a oportunidade de aprimorar habilidades técnicas e de organização em equipe, além de podermos estreitar nossos laços como grandes amigos. Pudemos conciliar os desejos de cada um com suas funções no projeto, e, dessa forma, o programa foi se encaminhando cada vez mais ao objetivo final. Foi um esforço pelo código e pela evolução pessoal de cada um.

Agradecemos à professora Giseli Rabelo Lopes e ao professor Ronald Chiesse de Souza por nos introduzirem ao universo da programação e por fornecerem todas as ferramentas necessárias para nossa formação como desenvolvedores do futuro. É com profundo sentimento de gratidão que expressamos nossos sinceros agradecimentos ao estimado Docente Daniel Chicayban Bastos. Sua dedicação incansável e paixão pelo ensino transcendem as fronteiras do convencional.

"To think you have to write,
to really think you have to write,
if you are thinking without writing,
chances are you are fooling yourself"

Leslie Lamport, 2016, Microsoft Facility