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

6 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

Aqui faremos o fluxograma, também acredito que seja mais fácil implementar depois que nosso código estiver pronto.

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.

- `voidfilterSpreadsheet(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.

- `void wait(float TimeInSeconds);`

  Aguarda `TimeInSeconds` unidades de tempo, em segundos, antes do programa continuar sendo executado. Decidimos por criar essa função ao invés de utilizar o `sleep` pois esse só aceita como entrada valores inteiros, além disso, também preferimos implementar nossa própria função ao invés de usar o `nanosleep`, pois essa aceita como entrada o tempo em nanosegundos, o que prejudicaria a legibilidade do nosso código.

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.

- `void 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.

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
- time.h

Casos de Teste Realizados e seus Resultados

Esperarei o código estar pronto para poder implementar essa parte do documento.

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.

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 aprimorarmos 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.

"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