ESCOLA SUPERIOR DE TECNOLOGIA E EDUCAÇÃO DE RIO CLARO
ASSOCIAÇÃO DAS ESCOLAS REUNIDAS

MELHORIAS PARA O FLUXO DE TRABALHO DO PORTAL TRANSPARÊNCIA DA CAMARA MUNICIPAL DE RIO CLARO

Nome: Alisson Daniel Rodrigues dos Santos RA: 8100213

Nome: Amanda dos Santos RA: 8100223

Nome: Andrew Vianna Carrazzone RA: 8100217

Nome: Danilo Alves da Silva RA: 8100211

Nome: Gismar Pereira Barbosa RA: 8100215

Nome: Mileide Cristina Loureiro RA: 8100209

RIO CLARO

DEZEMBRO / 2018

ESCOLA SUPERIOR DE TECNOLOGIA E EDUCAÇÃO DE RIO CLARO

ASSOCIAÇÃO DAS ESCOLAS REUNIDAS

Nome: Alisson Daniel Rodrigues dos Santos	RA: 8100213
Nome: Amanda dos Santos	RA: 8100223
Nome: Andrew Vianna Carrazzone	RA: 8100217
Nome: Danilo Alves da Silva	RA: 8100211
Nome: Gismar Pereira Barbosa	RA: 8100215
Nome: Mileide Cristina Loureiro	RA: 8100209

MELHORIAS PARA O FLUXO DE TRABALHO DO PORTAL TRANSPARÊNCIA DA CAMARA MUNICIPAL DE RIO CLARO

Trabalho da disciplina de Programação de Computadores I, ministrada na Escola Superior de Tecnologia e Educação de Rio Claro – ASSER entregue para avaliação.

Orientador: Prof. Dr. Erik Aceiro Antonio

RIO CLARO

DEZEMBRO / 2018

RESUMO

Utilizando-se a integração prática das disciplinas do semestre atual do curso de Sistemas de Informação: Cálculo Diferencial e Integral; (ii) Geometria Analítica e Vetores; (iii) Programação de Computadores I; (iv) Teoria Geral da Administração; (v) Lógica Matemática e com base na Atividade Multidisciplinar proposta, foi desenvolvido um sistema que visa uma melhoria sistemática para o Fluxo de Trabalho (workflow) do Portal Transparência da Câmara Municipal de Rio Claro.

SUMÁRIO

INTRODUÇÃO

Conforme Lei nº 12.527/2011, que regulamenta o direito constitucional de acesso às informações públicas, O Portal da Transparência da Câmara Municipal de Rio Claro, disponibiliza mecanismos que possibilitam, a qualquer pessoa, física ou jurídica, sem necessidade de apresentar motivo, o recebimento de informações públicas dos órgãos e entidades.

Porém, o usuário ao acessar o mesmo e realizar o download de um arquivo com lançamentos detalhados das despesas de um período informado, recebe um arquivo com dados desorganizados e de difícil compreensão.

O programa desenvolvido em Linguagem C pelo curso de Sistemas de Informação visa trazer facilidade na interpretação, avaliação e conclusão.

Para isso foi desenvolvido um sumarizador para:

- Calcular a média geral do Valor Empenhado;
- Calcular a média geral do Valor Pago;
- Calcular a média geral do Valor Líquido; e
- Calcular a média agrupada por item de serviço.

1. DESENVOLVIMENTO

A atividade foi desenvolvida utilizando-se a linguagem de programação C, e baseando-se nos moldes de uma Webquest, (atividade investigativa onde as informações com as quais os alunos interagem provêm da internet). Para controle de versão de arquivos do código utilizamos o sistema GitHub.

Para a organização e desenvolvimento da atividade foi utilizado o método Kanban, que é um sistema de gerenciamento de tarefas que mostra em um quadro o fluxo de trabalho onde é possível visualizar quais tarefas estão em desenvolvimento (doing), quais já foram finalizadas (done) e as que ainda não foram realizadas (to do). O método foi adaptado ao “Issues” no GitHub, onde é possível realizar o controle das tarefas estipuladas pelo grupo:

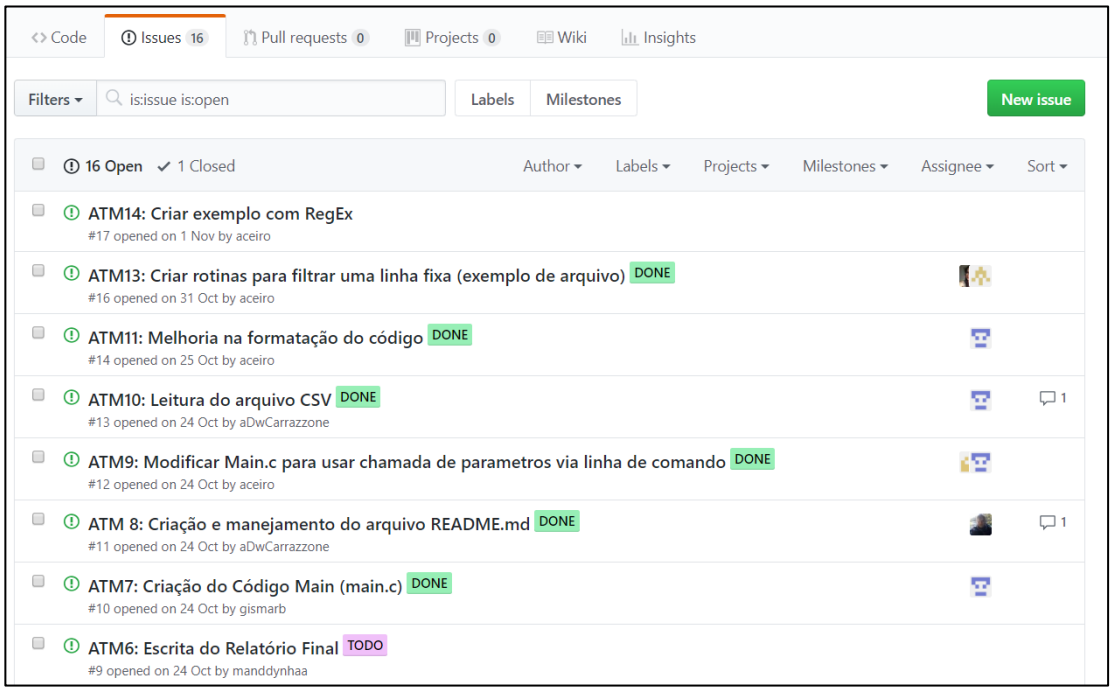
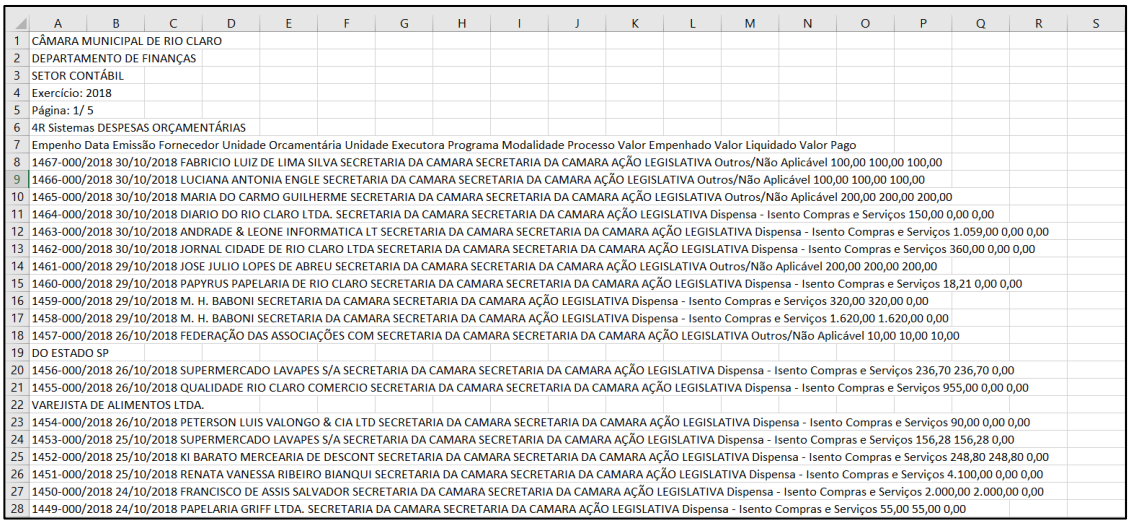


Figura 1: Issues - Lista de status das tarefas.

1.1 Coleta do arquivo xls gerado no site

Para o início do projeto primeiramente foi necessário realizar a coleta do arquivo que é gerado no site da transparência, o site apresenta quatro possíveis formatos para o download, para o projeto foi escolhido o formato xls.

Ao analisar o arquivo gerado o grupo notou que era necessário o desenvolvimento de rotinas dentro do código para separar as informações importantes que seriam utilizadas posteriormente para outras rotinas principais do código que são: o cálculo das médias (valor empenhado; valor pago; valor líquido e média agrupada por item de serviço). Abaixo temos uma visualização do arquivo que é gerado em xls:



	A
1	CÂMARA MUNICIPAL DE RIO CLARO
2	DEPARTAMENTO DE FINANÇAS
3	SETOR CONTÁBIL
4	Exercício: 2018
5	Página: 1/ 5
6	4R Sistemas DESPESAS ORÇAMENTÁRIAS
7	Empenho Data Emissão Fornecedor Unidade Orcamentária Unidade Executora Programa Modalidade Processo Valor Empenhado Valor Liquidado Valor Pago
8	1467-000/2018 30/10/2018 FABRICIO LUIZ DE LIMA SILVA SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Outros/Não Aplicável 100,00 100,00 100,00
9	1466-000/2018 30/10/2018 LUCIANA ANTONIA ENGLE SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Outros/Não Aplicável 100,00 100,00 100,00
10	1465-000/2018 30/10/2018 MARIA DO CARMO GUILHERME SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Outros/Não Aplicável 200,00 200,00 200,00
11	1464-000/2018 30/10/2018 DIARIO DO RIO CLARO LTDA. SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 150,00 0,00 0,00
12	1463-000/2018 30/10/2018 ANDRADE & LEONE INFORMATICA LT SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 1.059,00 0,00 0,00
13	1462-000/2018 30/10/2018 JORNAL CIDADE DE RIO CLARO LTDA SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 360,00 0,00 0,00
14	1461-000/2018 29/10/2018 JOSE JULIO LOPES DE ABREU SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Outros/Não Aplicável 200,00 200,00 200,00
15	1460-000/2018 29/10/2018 PAPYRUS PAPELARIA DE RIO CLARO SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 18,21 0,00 0,00
16	1459-000/2018 29/10/2018 M. H. BABONI SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 320,00 320,00 0,00
17	1458-000/2018 29/10/2018 M. H. BABONI SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 1.620,00 1.620,00 0,00
18	1457-000/2018 26/10/2018 FEDERAÇÃO DAS ASSOCIAÇÕES COM SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Outros/Não Aplicável 10,00 10,00 10,00
19	DO ESTADO SP
20	1456-000/2018 26/10/2018 SUPERMERCADO LAVAPES S/A SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 236,70 236,70 0,00
21	1455-000/2018 26/10/2018 QUALIDADE RIO CLARO COMERCIO SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 955,00 0,00 0,00
22	VAREJISTA DE ALIMENTOS LTDA.
23	1454-000/2018 26/10/2018 PETERSON LUIS VALONGO & CIA LTD SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 90,00 0,00 0,00
24	1453-000/2018 25/10/2018 SUPERMERCADO LAVAPES S/A SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 156,28 156,28 0,00
25	1452-000/2018 25/10/2018 KI BARATO MERCEARIA DE DESCONT SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 248,80 248,80 0,00
26	1451-000/2018 25/10/2018 RENATA VANESSA RIBEIRO BIANQUI SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 4.100,00 0,00 0,00
27	1450-000/2018 24/10/2018 FRANCISCO DE ASSIS SALVADOR SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 2.000,00 2.000,00 0,00
28	1449-000/2018 24/10/2018 PAPELARIA GRIFF LTDA. SECRETARIA DA CAMARA SECRETARIA DA CAMARA AÇÃO LEGISLATIVA Dispensa - Isento Compras e Serviços 55,00 55,00 0,00

Figura 2: Arquivo gerado no site da transparência no formato xls.

O grupo concluiu os seguintes aspectos relevantes do formato do arquivo baixado:

- Os valores para os cálculos das médias estão sempre no final de cada linha. Exceto algumas poucas linhas que terminam com caracteres, mas em sua maioria seguem este padrão;
- O arquivo gerado não vem com as informações organizadas por células, as informações de cada linha estão contidas todas juntas dentro de uma única célula;
- Há um padrão no início de cada linha (sequência numérica e data);
- Há palavras que são dispensáveis para a execução do programa (Ex; “SECRETARIA DA CAMARA”);

Após esta análise, foi possível verificar quais seriam os primeiros códigos a serem desenvolvidos pelo grupo.

1.2 Leitura dos dados do arquivo e armazenamento em memória

Num primeiro momento, foi realizada a implementação de uma forma de leitura do arquivo, onde, todo o conteúdo existente no mesmo, fosse transportado para memória, para o trabalho da coleta de tratamento das informações necessárias.

Para tal, construímos no código "*main*", um bloco de código, onde com uso das funções internas da própria linguagem C (*fopen, fgets*, etc), onde a cada linha lida do arquivo, os dados são armazenados em uma estrutura temporária (um vetor de *strings*, no código chamado de ***buffer***), e posteriormente armazenado em uma matriz de *strings* (um ponteiro de *strings*, no código chamado de ***read_data***). E assim, já teríamos cada linha do arquivo armazenada na memória (durante o tempo que o programa estive em execução) para trabalhar no tratamento e construção dos dados necessários.

1.3 Criar rotinas para filtrar uma linha fixa

Como no arquivo gerado as informações não vêm divididas por células, foi necessário encontrar outra forma para localizar e armazenar em cada linha os dados relevantes.

Notou-se que as informações eram separadas por espaços, a partir deste padrão, para a leitura e armazenamento dos valores para o cálculo das medias, foi utilizado a função "*strtok*" da qual é possível definir um delimitador para a separação de cada string.

O espaço foi definido como delimitador do final de cada bloco de caracteres. Esta função devolve um ponteiro e a cada leitura de linha foi criado a rotina para armazenar cada "bloco de caracteres" em uma variável, no total foram 3 variáveis.

Como os 3 números que precisamos para os cálculos da média estão sempre no final da linha, foi criada a rotina para o armazenamento em variáveis seguindo a posição de onde elas se encontram, ou seja, o tamanho total da linha subtraindo a quantidade de posição necessária para chegar no número.

Após conseguir ler e armazenar os 3 valores foi necessário transformar a *string* [1]para um *double*[2], ou seja, transformar o bloco de caracteres que filtramos num para um número real, para conseguirmos fazer as operações matemáticas.

Primeiramente, foi necessário retirar os pontos dos números e no lugar da vírgula substituir por ponto, pois no formato que o número vem o compilador não consegue realizar as contas. Para fazer esta correção foi implementado dentro da função ***strtodouble*** uma laço de repetição para ler todas as posição do *array* dos caracteres encontrados na linha anteriormente, e, nas posições em que fossem igual a virgula, substituir por ponto, e onde tivesse ponto, substituísse pelo próximo caractere da sequência do *array,* e após essas conversões, é chamado o retorno com a função ***atof*** para transformar a *string* em *double*.

Em resumo, foi utilizado filtros para a leitura e armazenamento seletivo em memória (transição de dados em formato de ***string, não ordenados***, para dados em formato ***numérico ordenados***) que, no caso dos valores, foi usado o ***padrão de final de linha***. Basicamente, usando uma matriz e um vetor auxiliar (matriz nomeada no programa de ***filter***) aplicando as funções ***strtodouble*** (função capacitada para converter os valores em formato numérico, tipo double) e ***get_values*** (função que aplicava a conversão de *strtodouble* linha a linha, para as 3 colunas de valores dispostos: ***valor empenhado, valor liquidado e valor gasto***) foi possível recuperar somente os valores, já convertidos. A lógica idealizada no processo, foi usar uma função própria da linguagem C (o ***strtok***), onde (por meio das funções acima, que implementaram o *strtok*) onde linha a linha era analisada, buscando inicialmente, **"espaços"**, sendo a terminação de todas linhas (quase todas) ocorriam nos valores, estes eram sequencias de *string* separadas por espaços. Logo, como tínhamos 3 valores, simplesmente, armazenávamos as *strings* dos últimos 3 espaços encontrados, e posteriormente, realizávamos a conversão para formato numérico, e logo após armazenamos na matriz, já citada acima, à estrutura ***values***.

[1] String: Vetor de caracteres; http://linguagemc.com.br/string-em-c-vetor-de-caracteres/

[2] Double: Variável de números reais. https://www.cprogressivo.net/2012/12/Os-tipos-float-e-double-numeros-decimais-reais-em-C.html

1.4 Transformar dados (Matriz de String) em informações (Médias)

Para tal, o trabalho foi divido em funções (funções implementadas no código para cada tipo de trabalho). Como já posicionado, o arquivo de entrada que deveria estar no padrão CSV, não estava formatado da forma correta (arquivo CSV são uma forma compatível com arquivos do tipo planilha, onde os campos são separados, arbitrariamente, por “,”). Logo, iniciamos uma busca por ***“***padrões***”***, onde para cada tipo de informação necessária, tínhamos que fazer nosso programa ler padrões de dados, para gerar as saídas corretas. Baseado na necessidade de ter dados de valores, e também informações de nomes (os nomes dos fornecedores), pensamos em usar estruturas auxiliares, para sempre ler dados em memória e trabalhar com o mesmo. Sendo assim, criamos 2 estruturas principais (Matrizes Globais) para armazenar dados do tipo ***“char”*** em uma estrutura, e dados do tipo ***“double / float”*** em outra. Sendo assim, a estrutura global para armazenar valores, foi chamada de ***“values”***, sendo uma matriz de valores no formato ***“double”***.

Uma outra ação que foi inicialmente necessária, foi para resolver um problema no arquivo CSV gerado no portal (site). O arquivo, a cada quebra de página, emendava no final da última linha o cabeçalho da próxima página. Logo, a cada página, tínhamos uma linha que não terminava em caracteres que representavam números, mas em ***“CAMÂRA MUNICIPAL DE RIO CLARO”***. Solução aí, foi simples e objetiva: Em cada linha lida, ocorria a verificação se o tamanho ***(quantidade de caracteres) – 31 = “C”*** (se era igual ao caractere C). Se isso ***fosse verdadeiro***, simplesmente ***atribuíamos a esta posição (resultante a diferença acima) o carácter de terminador “\0”*** (bloco de código implementada no código main). Isso, por que, desta forma, resumidamente, desprezamos o restante da linha, ficando somente com a nossa necessidade.

Com todos valores já ordenados em uma tabela de valores, já era possível submeter os dados gravados e disponíveis na ***matriz global values*** (dados em memória) para funções de calculo de médias ***average_overall***, ou simplesmente imprimir os valores com a função ***print_screen_values*** (impressão em tela).

1.5 Realizar rotina de filtro de Fornecedores

Para desenvolvermos o filtro de fornecedores utilizamos a segunda estrutura global (matriz) criada. Nesse caso, trata-se de uma Matriz de *Strings*, denominada em nosso programa de ***"suppliers"***. Para esta, o grande desafio seria armazenar somente os nomes dos fornecedores. Levando em consideração que o nome tinha um padrão, existia nos mais diversos tamanhos, e poderia ser repetido várias vezes nas linhas de *strings* presente no arquivo, o problema era grande e demandava uma abordagem diferenciadas da composição da matriz de valores.

Partindo da necessidade de captar dados em cuja qual não se tinha um padrão, iniciamos o desenvolvimento de filtros, que removeriam tudo aquilo que tinha um padrão. Desta forma, no próprio código *main*, identificamos ***o padrão de início de linha*** (basicamente, todas as linhas necessárias, ***iniciavam pela numeração 1***). Desta forma, o primeiro filtro: criar condicionais para ler somente linhas que satisfaziam este padrão. A este, adicionamos um parâmetro inicial para leitura do início da linha ***(constante com valor 25, sendo esta a posição inicial a ser realizada a leitura da linha) o START_OFF_LINE***. Assim, a linha iniciava a leitura já no início do nome do fornecedor. Logo, este foi mais um padrão criado para captação dos dados.

Para finalizar a leitura dos nomes do fornecedor, verificamos, que após o termino dos nomes, existia um padrão que era ***"SECRETARIA DA CAMARA"***. Aqui, a solução foi criar uma função, a ***clean_line***. A mesma era aplicada em todas as linhas, e usando uma função específica da linguagem C a ***strstr*** (que aponta o início e o fim das posições de uma *string*), para localizar este padrão, simplesmente, usamos o tamanho gerado pelo retorno desta função, e aplicamos em uma subtração entre o tamanho lido da linha – tamanho de retorno da função, que, nos dava justamente a *string* com o nome do fornecedor. Neste ponto, apenas fizemos o a inserção posição a posição (usando estruturas auxiliares, como a variável ***"ch"***) dos nomes do fornecedor, linha a linha matriz ***suppliers***.

2. ROTINAS DE SAÍDA (UTILIZAÇÃO DO PROGRAMA)

Com todos os dados já tratados e devidamente ordenados ou armazenados em memória (no formato mais amigável, para o trabalho), partimos para gerar as solicitações de interação ***usuário x programa*** (passagem de parâmetros e/ou opções, via CLI).

Neste ponto, de acordo com entrada, via CLI, informada pelo usuário, o programa geraria uma saída específica, que poderia ocorrer direto na tela do monitor, ou em um arquivo (gerando dados no arquivo em questão).

E, é claro, neste momento, a cada opção repassada, o programa iria fazer todo o fluxo necessário: verificar a existência do arquivo, se possível abri-lo, alocá-lo em memória, trabalhar outras estruturas, usando a memória (como demonstrados em capítulos anteriores), e de acordo com os parâmetros ou opções repassados, tratar os dados para cálculos de média, ou simplesmente exibir de forma ordenada, os valores e fornecedores.

2.1. Rotinas de Validação (Entrada de Parâmetros)

Para cada parâmetro e/ou opção repassada por parâmetro, existirá uma validação desta entrada. Isto ocorre, por exemplo, primariamente para seguir as premissas das funções idealizadas (ou seja, a cada parâmetro repassado, uma saída programada), e após aplicado, para satisfazer a utilização do programa. As rotinas de validação (basicamente *if, else*, com usos de alguns laços de repetição, como *while*, *for*, etc.) farão a validação das entradas via parâmetros na linha de comando.

Sendo assim, algumas das validações:

- Verificar ordem dos parâmetros: a passagem de parâmetros segue uma lógica de encadeamento: ***1°(./sumarizador) , 2°(--ajuda ou –entrada), 3°([arquivo e/ou caminho]), 4°(--media-empenhado, –media-liquidado, –media-valor-pago, –media-geral ou –media-fornecedor), 5°(--saida), 6°(--formato-csv ou formato-txt);***

- Verificar o própria parâmetro: por exemplo, se o arquivo repassado para leitura (entrada de dados) é existente ou não;

- Verificar a composição do parâmetro (obrigatório ou não): por exemplo, ao entrar no encadeamento o parâmetro ***"--saida"***, é arbitrária, logo em seguida, repassar os parâmetros ***--formato-csv*** ou –***formato-txt***;

Resumindo, as validações de entrada sempre estarão ou executando uma função, por meio do parâmetro de entrada, para gerar uma saída (médias, etc.) ou, retornando uma mensagem de notificação/atenção, reportando alguma entrada errada (mensagens de ajuda ou orientação).

2.2. Parâmetros de Entrada

Os parâmetros de entrada válidos, com funções internas dentro do programa são os seguintes:

- ***./sumarizador***: nome do programa ***(usamos este nome no instalador, mas poderá ser compilado com outro nome, não gerando problemas para seu funcionamento)***, e o programa propriamente dito. ***Sempre é o argumento inicial na CLI***, e sua utilização é arbitrária;

- ***--ajuda***: pode ser usado logo após o nome do programa (após um espaço), para imprimir em tela, instruções de uso do programa;

- ***--entrada***: argumento secundário, sempre usado antes da passagem do arquivo a ser lido (arquivo.csv). Caso não seja passado o arquivo, ele retorna mensagem de notificação;

- ***[arquivo/caminho]***: simbolicamente, seria o arquivo a ser lido. Exemplo de aplicação: ./sumarizador --entrada ***arquivo.csv,*** ou ./sumarizador --entrada ***./temp/file.csv***;

- ***--media-empenhado***: imprime em tela a média dos valores empenhados;

- ***--media-liquidado***: imprimi em tela a média dos valores liquidados;

- ***--media-valor-pago***: imprimi em tela a média dos valores gastos;

- ***--media-geral***: imprimi em tela as médias de valores empenhados, liquidados e gastos, todas juntas;

- ***--media-fornecedor***: imprimi em tela as médias de todos os valores, agregadas por fornecedores;

- ***--saida***: sinaliza que a saída das médias não será em tela. Logo após sua utilização, é arbitrária a utilização dos parâmetros ***--formato-csv*** ou –***formato-txt***;

- ***--formato-csv***: de acordo com parâmetro de média repassando no encadeamento, gera o ***arquivo.csv*** da média solicitada;

- ***--formato-txt***: de acordo com parâmetro de média repassando no encadeamento, gera o ***arquivo.txt*** da média solicitada.

3. RESUMO DAS TÉCNICAS UTILIZADAS

Quando a linguagem C, utilizamos várias técnicas para elaboração e desenvolvimento do projeto, dente elas modularização por funções, uso de constantes, variáveis e vetores globais, desmembramento do código *main*, criando uma biblioteca de funções, uso de funções específicas de *strings* (*strlen, strcpy, strcmp,* etc), dentre outras. Isso foi necessário, para facilitar o trabalho e/ou desenvolvimento em equipe do código.

3.1. Criação da Biblioteca atm_2s_2018.h

Utilizamos várias bibliotecas padrões da linguagem C (*stdio.h, stdlib.h, string.h, strings.h,* etc), mas para facilitar o trabalho em equipe (vários desenvolvedores trabalhando no mesmo código), criamos uma biblioteca própria (ao menos para nosso projeto), a ***atm_2s_2018.h***.

Esta biblioteca contém todas as partes globais do código:

- ***Funções***:
  - ***get_values*** (já citada acima);
  - ***strtodouble*** (já citada acima);
  - ***print_screen_values*** (imprime em tela a listagem de fornecedor e valores);
  - ***average_overall*** (aplica cálculos das médias);
  - ***write_to_file_all*** (imprime em arquivo médias de valores empenhado + liquidado + gasto);
  - ***write_to_file*** (imprime em arquivo médias de valores empenhado ou liquidado ou gasto);
  - ***help_message*** (imprime em tela, ajuda e orientações sobre o programa);
  - ***clean_line*** (já citada acima);

- ***average_per_supplier*** (organiza e calcula a média por fornecedores. Aqui, são usadas estruturas auxiliares de vetores e matrizes, para alocar os dados agregados e concatenar matrizes de valores e fornecedores. No código, nos comentários, é possível verificar qual foi a tática de ordenação para agregar fornecedores, e depois concatenar com os valores.);
- ***write_to_file_avg_supplier*** (imprime em arquivo a médias por fornecedores).

- ***Constantes (#define):*** foram utilizadas para delimitar tamanho de strings, criar rótulos para os parâmetros de entrada, rótulos para validação de padrões, etc.
  - ***BUFFER_SIZE*** 1024
  - ***MATRIX_SIZE_C*** 3
  - ***START_OF_LINE*** 25
  - ***CLEAN_LINE_FROM*** "SECRETARIA DA CAMARA"
  - ***TAG_GO_TO_NEXT*** "DONE"
  - ***CLI_INPUT*** "--entrada"
  - ***CLI_OUTPUT*** "--saida"
  - ***CLI_FILE_TYPE_CSV*** "--formato-csv"
  - ***CLI_FILE_TYPE_TXT*** "--formato-txt"
  - ***CLI_AVG_ENGAGED*** "--media-empenhado"
  - ***CLI_AVG_SETTLED*** "--media-liquidado"
  - ***CLI_AVG_PAID*** "--media-valor-pago"
  - ***CLI_AVG_OVERALL*** "--media-geral"
  - ***CLI_HELP*** "--ajuda"

- ***CLI_AVG_SUPPLIER*** "--media-fornecedor"
- ***VERSION*** "Sumarizador ATM2s2018"
- ***PROGRAM_NAME*** “. /sumarizador"
- ***EXAMPLE_PATH_OR_FILE*** "./Downloads/consolidado_out2018.csv"

- ***Variáveis e Vetores Globais:***
  - ***values:*** (já citado acima)
  - ***suppliers:*** (já citado acima)
  - ***aggregate_suppliers:*** matriz de strings, agregadas e separadas por fornecedores;
  - ***aggregate_values***: matriz de valores de médias, separadas por fornecedores;
  - ***type_file:*** váriavel que determina o tipo de arquivo (csv ou txt)
  - ***number_lines:*** variável que guarda o número de linhas finais da estrutura.

Em tese a grande massa do código, que controi as saidas de dados são realizadas por meio das funções implementadas na biblioteca atm_2s_2018.h.

Para sua utilização, basta adicionar, como demais bibliotecas usando ***#include “atm_2s_2018.h”***, no código main.

3.2. Código main.c

O código ***main.c***, praticamente executa ações de entrada e saída, ou seja, no código ***main.c*** realizamos o tratamento das entradas, para validação de parâmetros e outros, e de acordo com as entradas, retorna mensagem notificativas, ou executa as funções para gerar os resultados. Com exceção da validação de alguns padrões (para leitura das linhas), sua estrutura baseia-se em uma “espinha dorsal”, relacionando cada entrada de parâmetros, com sua respectiva saída. Trabalhamos fortemente no mesmo, para as validações de entrada, com comparação, cópia, tamanho de *strings*

(usando funções próprias da linguagem C, como *strlen, stcmp*, etc.) fazendo as comparações dos argumentos de entrada com a *tags* criadas (constantes). Resumindo, ele concebe as entradas e saída, recebendo cada rota de entrada para uma rota de saída. Ele executa quase todas as funções construída na biblioteca atm_2s_2018.h (algumas funções são chamadas por outras funções dentro da biblioteca). Isso foi feito, como já dito, para dar flexibilidade na hora do desenvolvimento, e agilizar o *deploy* do programa.

3.3. Estrutura de Organização GitHub (diretórios)

Para organização do repositório no GitHub, criamos a seguinte estrutura de pastas:

./atm_2018: "*README.md", "atm_2s_2018.h", "install_sumarizador.sh", "main.c";*
./examples: *arquivos ou código de apoio, exemplos;*
./resources: *pasta "documentation" e "Outubro.csv" (arquivo de entrada)*
./documentation: *"matriz_logica_atm2s_2018.pdf", "relatorio_final_atm2s_2018"*

3.4 Instalador install_sumarizador.sh

Apenas para facilitar a compilação, logo após baixar o repositório do GitHub, criamos um instalador (para uso em Linux, ambiente onde foi desenvolvido).

O ***install_sumarizado.sh*** apenas executa a linha de comando ***gcc main.c -o sumarizador***, fato que apenas força a usar o nome do programa ./sumarizador. Após o seu uso ***(./install_sumarizador.sh, e se der problemas, antes use chmod +x install_sumarizador.sh, para dar permissão e execução ao mesmo.)***, ele irá gerar o executável ./sumarizador, sendo possível então realizar os testes.

CONCLUSÃO

Os Sistemas de Informação são considerados ferramentas indispensáveis para a avaliação de resultados e tomada de decisões, é através das informações oferecidas ao sistema que é possível verificar a eficiência da operação de uma empresa pública ou privada, baseando-se no ciclo PDCA: planejar, dirigir, controlar e avaliar.

Com base na ATM2s2018 proposta foi possível elaborar um projeto envolvendo a integralização dos conceitos tratados nas disciplinas do semestre. Ressalta-se também que a ATM2s2018 foi colaborativa entre todos os alunos integrantes da equipe.

O Portal da Transparência estudado neste projeto, já oferece informações em forma de dados, porem de forma desorganizada, dificultando a leitura e compreensão dos dados do arquivo.

De fato, o site disponibiliza as informações devem ser divulgadas de forma obrigatória, porém o arquivo que é gerado para o usuário é confuso. Se o usuário desejar manipular os dados para fazer contas, realizar comparativos, fazer relatórios e entre outras ações, ele precisa num primeiro momento compreender a lógica e ordem do arquivo, localizar e pegar as informações que deseja anotando em um papel ou em outro arquivo, para depois realizar o seu estudo, ou seja, o arquivo gerado que em tese seria para facilitar esse processo de estudo, acaba na verdade, dificultando-o.

Desta maneira, o programa desenvolvido realiza a preparação de dados, deixando-os prontos para análise, após organização e limpeza de dados desnecessários, facilitando assim, a experiência do usuário.

REFERÊNCIAS

MINISTÉRIO DA EDUCAÇÃO. Disponível via URL: <http://webeduc.mec.gov.br/webquest/>. Acesso 30 de Novembro de 2018

THE C PROGRAMMING LANGUAGE. Disponivel via URL: <http://www.dipmat.univpm.it/~demeio/public/the_c_programming_language_2.pdf> Acesso 30 de Novembro de 2018.

OFICINA DA NET. Disponível via URL: <https://www.oficinadanet.com.br/post/14791-o-que-github> Acesso 20 de Novembro 2018.

ACESSO A INFORMAÇÃO. Disponível via URL: http://www.acessoainformacao.gov.br/assuntos/conheca-seu-direito/a-lei-de-acesso-a-informacao Acesso de 30 de Novembro 2018.