Problema 02: Sistema de Gerenciamento de informações estratégicas

Uellington Da Conceição Damasceno

[1]Curso de Engenharia de Computação – Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n - Novo Horizonte, Feira de Santana - BA, 44036-900

uellington99@gmail.com

Abstract. *This report will show the steps required to implement software that can calculate the amount generated from a compound interest calculation. Basically, the software should be able to receive an input file composed of various information. The program must then receive this information, process it, and generate an output file with as much relevant information about the input file.*

Resumo. *Neste relatório serão mostrados, as etapas necessárias para a implementação de um software capaz de calcular o montante gerado a partir de um cálculo de juros composto. Basicamente, o software deve ser capaz de receber um arquivo de entrada composto por diversas informações. Diante disso o programa deve receber estas informações, processá-las e gerar um arquivo de saída com o máximo de informações relevantes sobre o arquivo de entrada.*

1. Introdução

No Brasil, a cada dia, vemos muitos casos de corrupção, os quais tem tomando as manchetes dos meios de comunicação. Além de se tornar um dos assuntos mais discutidos no pais, atualmente. É evidente que as autoridades estão tomando providencias para a punição daqueles que praticam a corrupção.

Com o grande número de casos de corrupção sendo descobertos por nossas autoridades, nesse caso a Policia Federal, assim descobre-se grandes quantidades de informações. Essas informações estão se acumulando e dificultando a organização, separação e armazenamento das informações que são importantes para a investigação.

Afim de melhorar essa situação foi solicitado pelo diretor de inteligência da Policia Federal, um *software* que possa organizar e armazenar essas informações importantes. Pois, um *software* torna mais fácil e rápida esse processamento, organização e armazenamento do grande volume de informações.

Segundo a Polícia federal essas informações se tratam de propinas que eram pagas em parcelas aos políticos corruptos por empresas, afim de receberem vantagens em processos de licitação de obras. Assim o *software* deve calcular e organizar as informações a volta das propinas pagas pelas empresa aos políticos corruptos.

Neste relatório, será apresentado como foi feito, e quais os critérios foram considerados para a implementação deste produto, para um melhor entendimento de como o *software* foi desenvolvido, este relatório foi dividido em quatro partes, sendo elas: introdução, metodologia, resultados e discussões, conclusão e por fim apresentar as referências que foram utilizadas ao longo do projeto.

2. Metodologia

Visando alcançar o máximo desempenho do *software* ocorreram algumas reuniões, onde algumas ideias e questões importantes foram debatidas. Como por exemplo: O layout do arquivo, tamanho das matrizes, validação do arquivo, formulas e cálculos.

A figura 1 mostra uma representação gráfica de como o software foi estruturado durante as sessões.

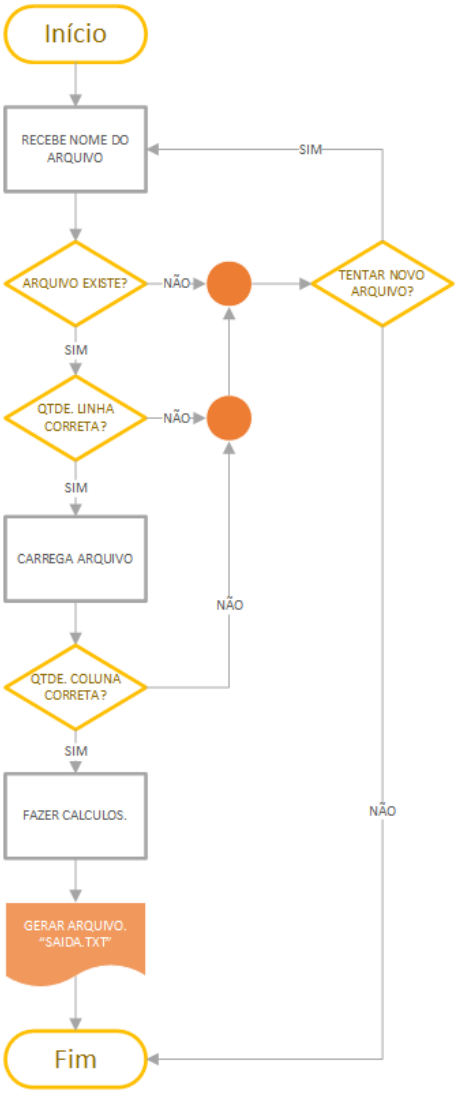

Figura 1. Fonte: Próprio autor

2.1. Layout do arquivo de entrada

Foi pré-determinado que o arquivo de entrada deve seguir o seguinte padrão:

- Na primeira linha deve conter o valor correspondente a taxa de juros (formato de valor decimal);
- Na segunda deve ter o número de político que receberam propina (formato inteiro);
- Na terceira deve ter o número de empresas (formato inteiro);

A partir da terceira linha, deve conter um matiz com valores que corresponde ao valor da propina paga pelas empresas. Essa matriz deve seguir os seguintes requisitos:

- Primeira deve corresponder a identificação do político (formato inteiro);
- Próximas colunas devem conter os valores das propinas a ser paga (formato decimal);

Já na última matriz deve conter valores que representam a quantidade de parcelas Pagas por empresa (em formato inteiro).

Vale também ressaltar que todas as matrizes devem conter a uma quantidade de elementos condizentes com os valores descritos na segunda e terceira linha do arquivo.

Veja na figura 02 o exemplo de layout genérico.

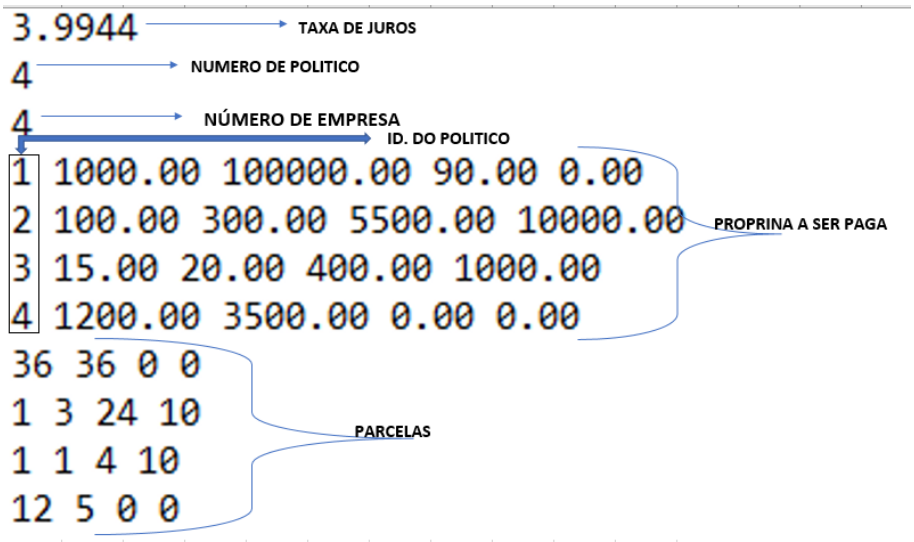

Figura 2. Fonte: Próprio autor

2.2. Tamanho das matrizes

Para evitar possíveis conflitos que poderiam ser gerados pela falta de comunicação, ficou decidido que a quantidade de político, empresa e as parcelas não deveriam passar de 50. Consequentemente, ficou definido que as matrizes utilizadas para armazenar tais valores seriam estáticas com 50 linhas e 50 colunas ou seja: uma matriz 50x50.

2.3. Validação do arquivo

Antes de executar todos os cálculos o arquivo passar por um rigoroso processo de validação baseados em quatro vertentes sendo elas:

- verificação de linhas;
- verificação de letras (string);
- verificação de pontos (’.’);
- verificação de colunas;

Note que além das verificações propostas pelo grupo (figura 1) houve mais duas verificações foram implementadas.

2.3.1. Verificação de linhas

Inicialmente, houve um tanto de dificuldade com relação a como saber se quantidade de linhas do arquivo era correta. Porém, depois de algumas observações foi descoberto que o arquivo de sempre deve ter um número ímpar de linhas.

Sabendo que um arquivo (valido) sempre deve ter um número ímpar de linhas. Tornou-se simples elaborar uma solução.

Veja na figura 03 o algoritmo usado para validar o número de linhas do arquivo.

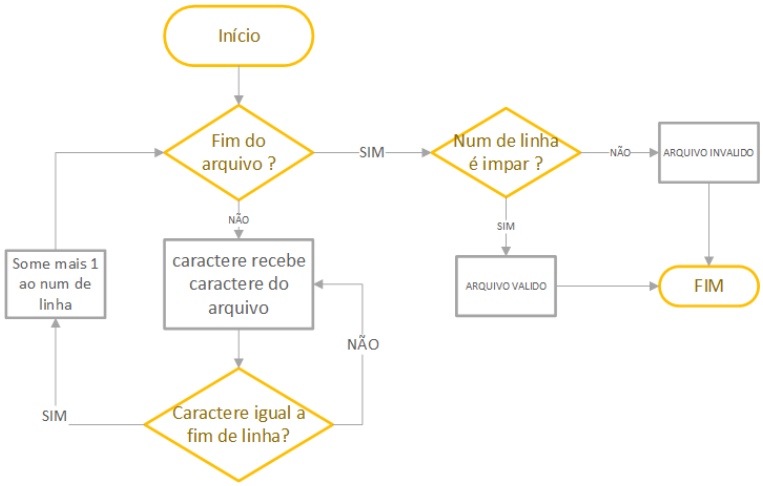

Figura 3. Fonte: Próprio autor

2.3.2. Verificação de letras

O algoritmo utilizado para verificar se existe alguma letra no arquivo, é bem semelhante com o utilizado na verificação de linha. O diferencial dessa verificação é que a função ”isalpha”da biblioteca ”ctype.h”foi utilizada.

2.3.3. Verificação de pontos

Pode-se dizer que, assim como as duas verificações anteriores foram utilizados algoritmos semelhantes, a verificação de pontos também continuou seguindo a mesma linha de raciocínio.

Entretanto, o diferencial dessa verificação é que, além de usar a função ”ispunct”para verificar se o carácter analisado é um ponto. Ela também compara como o carácter anterior. E assim, consequentemente impedindo que dois pontos seguidos sejam colocados, ou até mesmo por um ponto sozinho em meio aos elementos da matriz.

2.3.4. Verificação de colunas

A verificação de colunas surgiu a partir da necessidade de saber se a quantidade de elementos no arquivo condizia com os valores descritos na linha de político e empresa. Entretanto, não existia uma maneira especifica para resolver tal problema.

Diante disso, foram levantadas algumas questões sobre como resolver esse problema. E então surgiram algumas propostas como por exemplo: ler o número de políticos e empresas e ao final verificar se a quantidade de elementos na matriz condizia com esses valores.

No entanto, não foi possível implementar essa solução. Pois, demandava mais tempo do que o disponível.

Depois de uma análise do arquivo notou se que, a quantidade correta de elementos que o arquivo deve ter para ser valido equivale a seguinte expressão algébrica:

- empresa * politico + ((empresa -1) * politico);

Assim como as linhas poderiam ser contadas com base na identificação do caractere especial ”new line”. Notou-se que era possível contar a quantidade de elementos do arquivo a partir da quantidade de espaços em branco.

2.4. Formulas e cálculos

Durante o desenvolvimento do programa algumas fórmulas matemáticas foram utilizadas para que fosse possível resolver determinadas contas.

Diante todas as formulas utilizadas. Pode-se afirmar que, a formula de juro composto foi de grande valia para o desenvolvimento do *software*. Pois, foi a partir dela que os cálculos empregados nas outras formulas obtiveram o resultado esperado.

Observe na figura 04 a formula do juro composto.

$$M = C*(1+i)\hat{}t$$

$$J = M - C$$

M = Montante
C = Capital Aplicado
J = Juros
i = Taxa de juros
t = Tempo de aplicação

Figura 4. Fonte: Próprio autor

Vale também ressaltar que o *software* em questão foi desenvolvido em linguagem

de programação C tendo em vista o suporte para a plataforma Windows. Utilizando o DEV-C++ e o Code::Blocks com ambiente de desenvolvimento.

3. Resultados e discussões

O *software* em questão pode ser resumido em três partes, sendo elas:

1. Validação do arquivo;
2. Processamento dos dados;
3. Emissão do arquivo de saída;

3.1. Validação Do arquivo

Para que todos os cálculos sejam executados com excelência, o arquivo de entrada deve passar por algumas validações de *layout*. O arquivo deve obrigatoriamente cumprir todos os requisitos descritos na sessão 3 desse relatório.

Se por algum motivo o arquivo de entrada não for ”aprovado”nos testes, uma mensagem de erro juntamente com a opção inserir novo arquivo irá aparecer. Possibilitando ao usuário finalizar a execução do *software* ou fazer a inserção de um novo arquivo.

No entanto, se o arquivo for devidamente validado, irá para a fase de processamento de dados. E ao final irá gerar um arquivo de saída contém o resultado de todos os cálculos.

3.2. Processamento dos dados

Após passar com sucesso em todas as validações, o arquivo tem os dados carregados para as variáveis. Para que futuramente possa ser utilizada para fazer todos os cálculos necessários. E assim, transformando dados em informações relevantes.

Esse processamento passa por diversas etapas e procedimentos diferentes. Inicialmente, o procedimento ”calculaJuro”recebe como parâmetro de execução a taxa de juros (i) e a quantidade de parcela (t). Em seguida executa a primeira parte do cálculo do juro composto e ao final, retorna diversos valores (que foram chamados de ”juroCalculado”) que armazenados em uma arranjo (matriz).

Com a taxa de juros calculada um segundo procedimento chamado ”calcularPropina”entra em ação e multiplica o ”juroCalculado”pelo capital inicial (C) e ao final retorna uma matriz com os valores do montante gerado (A matriz foi chamada de ”propinaCalculada”).

Vale também ressaltar que, por questão de eficiência o procedimento responsável por calcular a propina também é responsável por somar e armazenar os valores que correspondem ao total pago com e sem juro da propina paga a cada político.

Com a propina calculada um terceiro procedimento que recebe como parâmetro o montante é inicializado. Esse procedimento tem como função descobrir qual é o valor total com e sem juro pago por empresa. Para efetuar esse cálculo, foi necessário somar os valores de todas as colunas exceto a primeira. Pois, a primeira coluna corresponde o número de identificação do político.

Ao final da execução do procedimento anterior, um novo procedimento que tem como função descobrir qual é o político que, mais ou menos receberam propina. Após

concluir esse passo o procedimento retorna um vetor com a identificação do político a identificação, valor com e sem juros, e o valor total recebido.

O procedimento responsável por identificar os valores descritos acima foi chamado de ”maisEmenos”. Ele recebe como parâmetro de funcionamento dois vetores. Sendo eles chamados de ”PCJ”(propina Com juro calculado) e ”PSJ”(propina Sem juro calculado). Além desses dois vetores ele também recebe uma variável de controle que utiliza com base os valores de político ou empresa coletados no arquivo de entrada e informa quantas vezes o laço de repetição deve ser executado

Durante a estruturação do procedimento ”maisEmenos”houve a implementação proposital de uma verificação redundante. Essa verificação foi implementada visando solucionar uma falha que se dava no momento da identificação do político que menos recebeu propina. Pois, a metodologia utilizada para descobrir tais valores não funcionava corretamente quando o político que menos recebeu ocupava a primeira posição do vetor.

O último procedimento que os dados coletados do arquivo de entrada passa antes de finalizar a parte de processamento e iniciar a emissão do arquivo de saída é o cálculo da média aritmética. Para efetuar o cálculo da média esse procedimento recebe com parâmetro de funcionamento o valor pago ou recebido com juros um vetor chamado de JNDPE (Juros Numero De Político Empresa) que contém armazenado os valores que é utilizado para controlar o laço de repetição informando quantas vezes ele deve repetir utilizando como base a quantidade de político ou empresa.

3.3. Gerando arquivo

Por questão de organização a parte de emissão de arquivo foi separado em três procedimentos. Sendo eles:

- ”Gerar arquivo P1”
- ”Gerar arquivo P2”
- ”Gerar arquivo P3”

Apesar de ter sido dividido em três procedimento a metodologia empregada segue o mesmo padrão de estruturação. O diferencial dos procedimentos é que no primeiro um ponteiro do tipo FILE é aberto em modo de escrita ”w”. Já nos outros dois procedimentos de geração de arquivo, o ponteiro é inicializado em modo “a+”, o que corresponde a verificar se o arquivo existe e escrever ao final do arquivo.

Gerar arquivo P1 é responsável por imprimir os valores que encabeça, a saída do arquivo. É através dele os valores que correspondem a identificação do arquivo, propina com juro, total pago de propina e total pago por empresa, são impressos.

Gerar arquivo P2 é responsável por imprimir os valores correspondente ao político e a empresa que mais e menos pagou e ou recebeu propina e com os seus respectivos valores com e sem juros.

Gerar arquivo P3 tem tão somente como função imprimir os valores que correspondem a média de propina paga e recebida. Veja na figura 05 a representação gráfica de um arquivo.

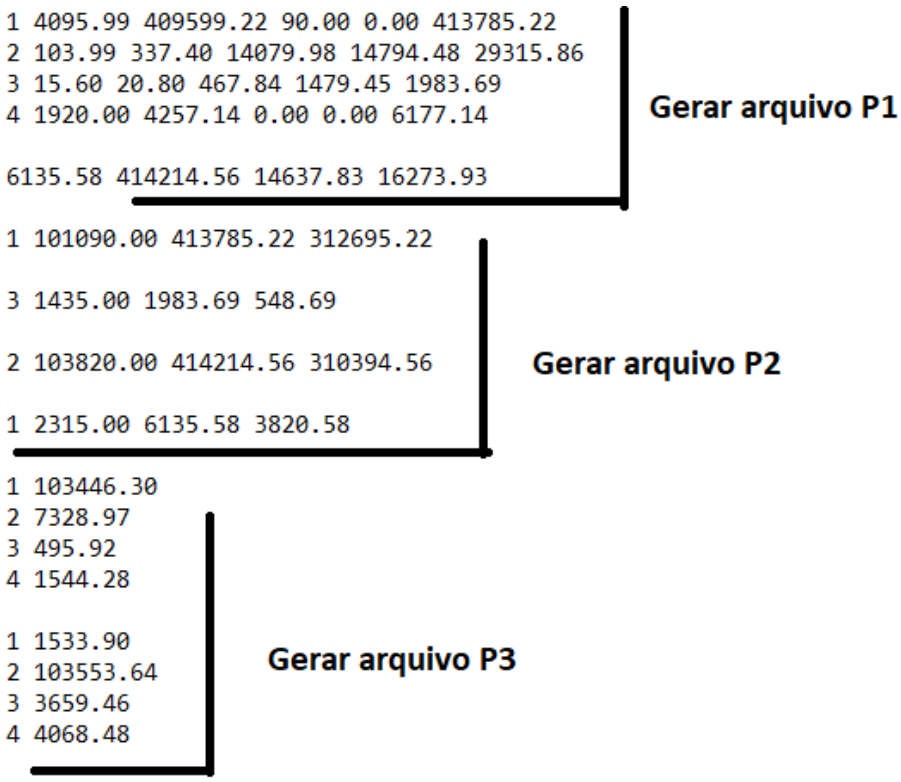

Figura 5. Fonte: Próprio autor

3.4. Teste efetuados e resultados obtidos

A inserção de arquivos inválidos foi o principal teste utilizado para reduzir significativamente a quantidade de *bug's* que poderiam ocorrer durante a execução do código.

Para garantir que o máximo de falhas fossem eliminadas, 12 arquivos de teste foram utilizados. Os arquivos utilizados para teste podem ser divididos em duas categorias, sendo elas:

- Arquivo de erros sintáticos;
- Arquivos de erros semânticos;

Os principais arquivos de erros sintáticos utilizados, tinham como função ”ferir”a integridade do layout, utilizando como artificio a inserção de colunas, linhas, string e pontos em lugares indevidos do arquivo.

Dois arquivos com erros semânticos foram inserido. Visando, verificar a eficácia do programa, com relação ao nível de rigorosidade no tratamento de inserção de arquivos inválidos semanticamente.

O resultado dos testes foram extremamente satisfatório. Pois o *software* conseguiu impedir a inserção de todos arquivos inválidos tanto semanticamente quanto sintaticamente.

4. Conclusão

Após todo o trabalho de desenvolvimento, o programa chegou em um ponto do qual consegue executar com excelência todos os requisitos mínimos necessários. Requisitos esses que variam desde um simples carregamento do arquivo, cálculo com matrizes até gerar um arquivo de saída com todas as informações pré-definidas.

Na atual versão do programa não existe nenhum *bug* encontrado. Entretanto, é possível que exista algum problema ainda não descoberto, que pode ser desencadeado depois e alguma sequência de inserção de entradas ou arquivos inválidos.

Para um melhor desempenho do programa, poderia ser implementado o uso de gerenciamento de informação por índice, onde o usuário através da identificação do político poderia encontrar apenas as informações relevante do mesmo. Além de poder incrementar um banco de dados para que fosse possível fazer uma comparação mensal, trimestral, semestral, e ou anual da propina paga.

5. Referências

Nenhuma fonte foi consultada.