



**CENTRO UNIVERSITÁRIO DA GRANDE DOURADOS**

---

**Jhonathan Paulo Banczek**

## **RELATÓRIO**

---

Dourados  
**2010**



CENTRO UNIVERSITÁRIO DA GRANDE DOURADOS

---

Jhonathan Paulo Banczek  
RGM: 122.845

### **Relatório da criação da Agenda.**

Relatório apresentado na Disciplina de Estrutura de Dados II do 2º ano, Curso de ciência da computação Faculdade de Ciências exatas. Unigran.

Professor: Ademir M. Sanches.

---

Dourados

---

## Relatório.

Foi implementado uma Agenda Eletrônica em que o usuário informa o nome e data de nascimento. Usou-se a estrutura de dados – Lista Ordenada – Encadeada.

Como o meu RGM é ímpar, tive que usar lista ligada ordenada no meu programa.

### Seqüência das instruções:

Ao executar o programa, a função **abrirArquivo** carrega os dados direto do arquivo para a lista.

Nesta função se não existir o arquivo “agenda.csv” o algoritmo cria um novo.

### Código:

```
if(aux == NULL) {//senao existir, cria um novo  
  
    aux.open("agenda.csv",ios::out);  
    aux.close();
```

Em **abrirArquivo**, utilizo uma string como buffer, então abro o arquivo “*agenda.csv*” e dentro de um loop que pára só quando chega o fim do arquivo, vou recebendo sempre linha por linha do arquivo. (para cada laço), recebendo a linha que seria como “*nome ; data*” utilizo outras duas strings que recebem os valores de nome e da data.

Utilizei uma função que pega o nome da posição 0 da string(bufferArquivo) até quando achar “;”. Pois seria o campo do nome no arquivo. Logo depois pega a data do “;” até fim do buffer, e passo o nome e a data para a lista (inserindo) chamando a função **insereOrdenado**.

### Código:

```
//grava o nome da variavel da posição 0 até achar ';' ;  
nome = bufferArquivo.substr(0, bufferArquivo.find_first_of(';'));  
//grava a data de nascimento partindo do ';' até final da string "bufferArquivo"  
data = bufferArquivo.substr(bufferArquivo.find_first_of(';')+1 );
```

Isso se repete até o fim do arquivo.

Depois de ler o arquivo, é chamado a função **menu**, que constrói o menu de opções do programa. (Na função **menu** é utilizado *system ("cls")*; para limpar a tela. )

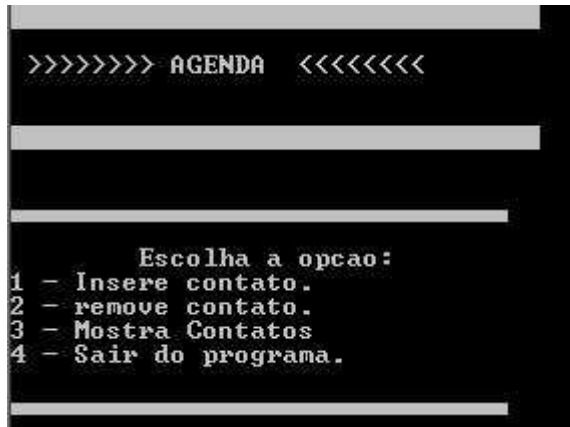


imagem da tela.

Escolhendo a opção, vai para um CASE em que chama as próximas funções.

**1 - Insere contato** // recebe o *NOME* (sendo menor de 30 caracteres e não possuindo ‘;’) e da *DATA*(sendo menor que 10 caracteres “DD/MM/AAA” e não possuindo “;” ), antes de passar o nome e a data, criei um código para converter a *STRING* nome para minúscula. Pois não existe dentro de *STRING* uma função para isso:

**Código:**

```
//CONVERTE a string para minúscula
for( int i = 0; i < int(add.nome.length()); i ++)
    add.nome[i] = tolower(add.nome[i]);
```

Depois chama a função **insereOrdenado**, que por sua vez chama a função **insereNo**, que insere na lista.

**Código:**

```
do{  
    cout <<"Informe o nome: " << endl;  
    cin.sync(); // equivalente ao flushall() do C  
    getline(cin,add.nome); //pega o nome  
}while ( ( int(add.nome.length()) > 30) || add.nome.find(';') != string::npos)
```

**2 - Remove contato** // remove o contato de maior prioridade (ordenado por nome)

**3 - Mostra Contatos** // mostra os contatos que estão na lista, é passado por parâmetro 0, para mostrar

**4 - Sair do programa** // termina o programa

Estas opções ficam dentro de um LOOP principal que só termina quando digitado 4 (sair do programa).

Ao terminar o programa, chama-se novamente a função **mostra**, passando agora por parâmetro **1**, que exclui o arquivo “agenda.csv” e passa os dados dos contatos atualizados para “agenda.csv”.

Depois chama a função **libera**, que libera toda a memória alocada que a lista usou no programa.

A minha maior dificuldade foi na implementação da lista, que muitas vezes se mostrou confusa toda aquela série de ponteiros pra ponteiros e operadores de endereço.

## **Funções:**

**void menu**                    *//função que monta as opções.*

**int main**                    *//função principal.*

**void abrirArquivo**                    *// abre os dados do arquivo "agenda.csv".*

**void libera**                    *//libera toda a memória alocada no programa.*

**void mostra**                    *//mostra os contatos da lista, parâmetro 0 =mostra / 1 = grava e atualiza dados no arquivo.*

**void insereOrdenado**                    *//insere os dados ordenado pelo nome (campo chave)*

**void insereNo**                    *//insere o nó na lista com os dados = nome e data*

**void desenha**                    *//desenha as linhas parâmetros 0 e 1 < formatos da linha*

## Conclusão

*A criação da Agenda eletrônica foi fácil, em alguns momentos existiam alguns problemas na solução de determinada rotina, porem, uma olhada na referencia da linguagem C++, conseguia-se obter funções que resolviam o caso, como por exemplo, as funções subst e find.*

*Na manipulação do Arquivo também foram utilizadas simples funções descritas na biblioteca “fstream”.*

*O meu maior problema foi com a Lista, pois teoricamente aparenta ser fácil, as analogias usadas em sala para explicar o conceito ajudam bastante, só que, quando implementado em código se mostra confusa e muitas vezes por ter um código extenso e demasia de ponteiros se torna sujeita a erros.*

## Referência

- Apostila de Programação Orientada a Objeto em C++ - A André Duarte Bueno, UFSC-LMPT-NPC
- Estrutura de dados – Lista Linear – Prof. Ademir M. Sanches.
- <http://www.cppreference.com/wiki/> ( onde aprendi algumas funções de string)
- <http://www.cplusplus.com/> (onde aprendi algumas funções de Arquivos em c++)