UNIVERSIDADE ESTADUAL DE MONTES CLAROS

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS – CCET

Patrick Pierre Fernandes Ferreira

Messias Ferreira Muniz

**VESTIBULAR**

Montes Claros, Minas Gerais

Junho de 2011

Patrick Pierre Fernandes Ferreira

Messias Ferreira Muniz

**VESTIBULAR**

Trabalho apresentado ao professor Renato, como parte das exigências para avaliação da disciplina de Algoritmos e programação I, 1º período.

Montes Claros – MG

Junho/2011

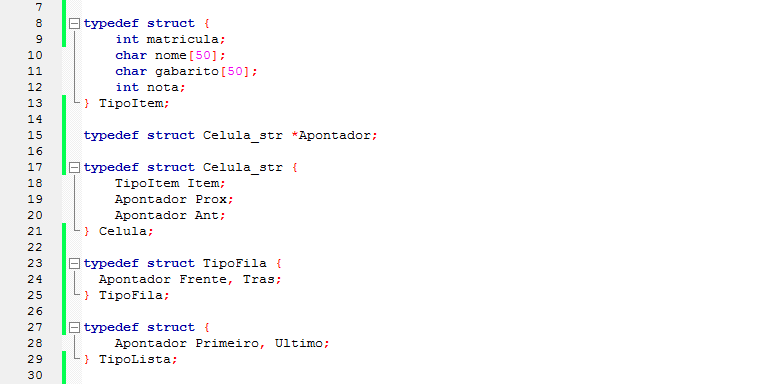
**Introdução**

Em 12 de junho de 2011 acontece em Montes Claros o vestibular da UNIMONTES – Universidade Estadual de Montes Claros. Tendo em vista que, sem a utilização da computação o tempo demandado para correção de todas essas provas seria enorme, o uso de softwares capazes de analisar e calcular as notas se faz extremamente necessário.

O objetivo deste trabalho é desenvolver um software para correção de provas de vestibular. Nosso modelo se baseia nos apresentados pelo professor orientador em sala de aula, utilizando estruturas de dados, lista duplamente encadeada e fila, o objetivo deve ser construir um programa que receba um gabarito oficial, matricula, nome e gabarito de cada candidato, calcule a nota e imprima uma lista contendo os classificados, a lista de espera, pontuação e gabarito e tenha opções para matricular os candidatos e transferi-los da lista de chamada para a lista de classificados.

**Implementação**

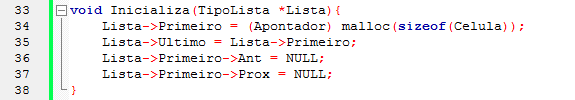
Como já foi dito na introdução, nosso modelo se baseia nos apresentados pelo professor orientador em sala. Uma das primeiras coisas a se observar são as definições das estruturas, ou tipos.



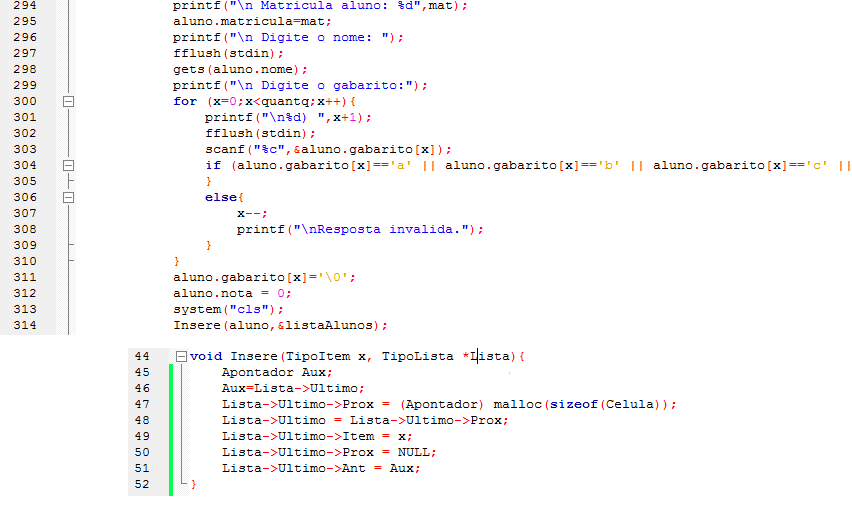
Na figura acima são mostradas as estruturas utilizadas no nosso modelo de programa. Primeiramente é criado uma tipo denominado TipoItem, para conter os dados, ou seja, onde vão ser salvos as informações necessárias ao programa (matricula, nome, gabarito do candidato e sua nota). Após isso, é definido um tipo ponteiro chamado Apontador para Celula\_str. Para este Celula\_str são criados um TipoItem chamado Item e dois tipos “Apontador”, um chamado Prox e outro Ant. Logo após isso são definidos o que poderíamos caracterizar como sendo os locais onde os dados vão ser armazenados, ou seja, a própria fila e lista. Para isso são definidos um TipoFila e um TipoLista, cada um contendo dois ponteiros do tipo Apontador – Frente e Tras para TipoFila e Primeiro e Ultimo para TipoLista.

Fazendo uma comparação mais grosseira, poderíamos comparar uma lista ou fila a uma agenda, cada página possui dados que podem ser acessados, apagados ou modificados. Neste caso, poderíamos considerar que cada página desta “agenda” seria uma célula, e os dados contidos nele um TipoItem. Como foi visto na imagem, a célula recebe também dois ponteiros do tipo Apontador, sua função é de permitir “navegar” entre as células, voltando a comparação, mudar as páginas para poder encontrar o que deseja, fazendo com que elas tenham uma ligação entre si. O Prox, neste caso, é criado para ligar essa alocação de memória a que viria a ser a próxima, e Ant, a anterior.

Antes de mais nada é preciso criar uma função que de inicio a uma lista. Uma estrutura serve como um modelo já pronto necessitando somente da inserção dos dados. Iniciar uma lista, basicamente, seria definir para onde a primeira célula e a ultima vão apontar inicialmente. Como mostrado na imagem abaixo, primeiramente Primeiro e Ultimo da lista recebe uma alocação do tamanho de “Celula”. Após isto, como ainda não existem outras células, Ant e Prox recebem um valor nulo temporariamente.

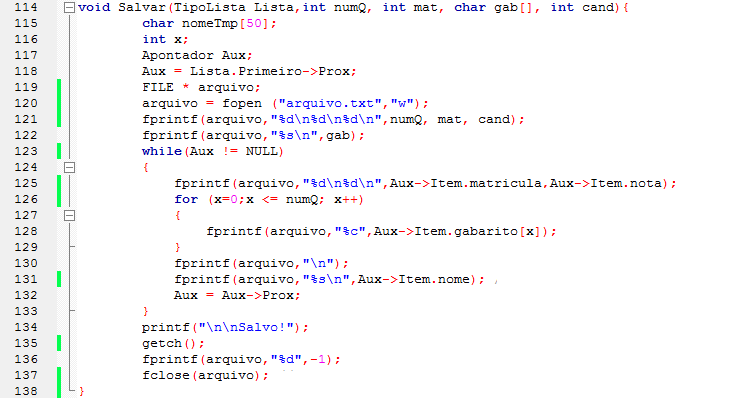


A função de inserção de dados na estrutura segue uma lógica aproximada da função acima. Neste caso, precisamos primeiro dar entrada com os dados necessário. Matricula, no caso do nosso programa, está começando de 1000, e a é incrementado sempre que inserir um candidato. Essa matricula é gravado primeiramente em uma variável temporariamente (mat), para depois ser colocado em matricula do TipoItem definido aluno (aluno.nome). O mesmo se repete com o gabarito, e a nota recebe inicialmente o valor “0”. Após todos estes dados serem colocados no TipoItem aluno, é dado inicio a função de inserção em si.

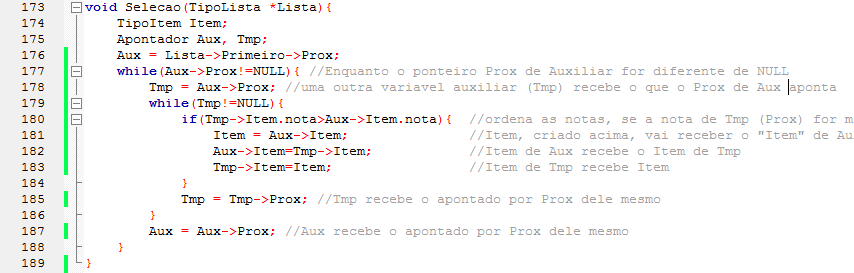


A função de inserção funciona da seguinte maneira: primeiramente ele recebe o TipoItem aluno (aqui armazenado em um TipoItem x) e um TipoLista listaAlunos (nesta função chamada de \*Lista), e é criado também uma variável auxiliar do tipo Apontador. Essa variável Auxiliar recebe primeiro o espaço alocado denominado por Ultimo em Lista. Se lembrarmos, veremos que inicialmente o Prox recebia um valor nulo, aqui começamos a ver como ele realmente vai funcionar. Após o Ultimo ser armazenado na variável auxiliar, Prox recebe uma alocação de memória do tamanho de Celula também, deixando de ser assim o ultimo da lista e Ultimo de Lista recebe então Prox dele mesmo. Ultimo recebe então o valor de entrada x (“aluno” na função principal) e dos ponteiros do tipo Apontador, Prox passa a apontar um valor nulo e Ant de ultimo passa a apontar para Aux.

Para salvar em arquivo foi preciso de uns conhecimentos a mais sobre o conteúdo, pois alguns comandos não conhecíamos. A idéia básica do que é feito aqui é muito simples: primeiramente é criado o arquivo de texto (no caso do nosso programa, arquivo.txt). O comando “fopen” serve para que o programa crie o arquivo, podendo assim “alterá-lo”. O lugar onde se encontra o “w” define o modo de acesso do arquivo. No caso do “w”, é criado um arquivo vazio onde possam ser salvas as informações. Se acaso já existir um arquivo com o mesmo nome, este será apagado, permanecendo somente o mais recente. O comando fprintf é usado para salvar no arquivo criado os dados desejados. Basicamente seria como um printf comum, mas ao invés de imprimir na tela do computador, imprime em um arquivo, e depois de tudo feito é necessário fechar este arquivo, onde entra o fclose.



Uma outra função importante também é a para criar a lista de aprovados (Selecao). Primeiramente é criado uma variável do tipo TipoItem chamada Item, e dois apontadores auxiliares (Aux e Tmp). Serão criados dois loops nesta função, um para verificar as possibilidades do com o Aux e outra para verificar as do Tmp. Essas duas funções vão ser executadas ate que chegue ao final da lista (indicado com o um valor nulo). As notas dos alunos então são comparadas, se a nota do apontado agora denominado Tmp for maior que a do Aux, o TipoItem Item irá receber o Item apontado de Aux, que por sua vez receberá o de Tmp, e este o de Item.



**Conclusão**

Este trabalho acabou se revelando uma experiência muito interessante, apesar de termos encontrado certas dificuldades em algumas coisas. A falta de tempo e de horários acessíveis aos dois membros da equipe, principalmente, foram os fatores que mais nos prejudicaram e atrasaram na conclusão do mesmo. A idéia do programa já havia nos sido apresentada, basicamente já sabíamos o modelo e o que fazer, mas não estávamos encontrando tempo suficiente para implementá-lo.

Na parte das descrições de funcionamento das funções não encontrei necessidade de falar sobre todas elas, citando somente as mais importantes para o funcionamento de alguns comandos do programa.