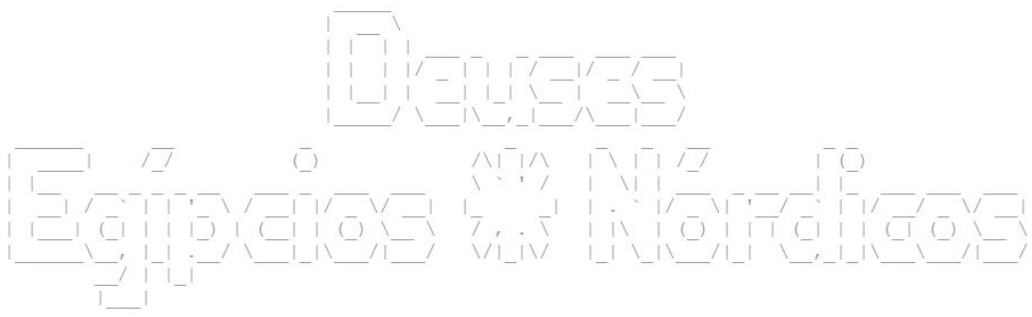
Relatório de Desenvolvimento de Software - Cadastro de Deuses Egípcios e Nórdicos

Gabriel Marques de Melo
Murilo Carmagnani Lopes



Resumo

Manipulações em arquivos são operações comuns em um *SO*, por exemplo. O domínio da implementação de operações com arquivos binários, bem como a habilidade de desenvolvimento de softwares baseados em paradigmas de orientação a objetos, são pré-requisitos para ser um **bom programador**. Pensando nisso, fora proposto pela disciplina *DCC216* (Estruturas de Dados) o desenvolvimento de um software que gerencie as principais operações em um arquivo binário (**inserção ordenada**, **remoção**, **busca** e **impressão**) manipulando objetos de determinadas classes dispostos em ***Sequence Sets***. O trabalho fora dividido em grupos com temas distintos entre si. O tema designado ao nosso grupo fora "**Deuses Egípcios e Nórdicos**"

1 O Código

1.1 Bibliotecas

Na implementação foram utilizadas as seguintes bibliotecas padrões do *C/C++*:

- ***<iostream>***
  Biblioteca padrão de entrada/saída de fluxos(*streams*);
- ***<fstream>***
  Biblioteca padrão usada para entrada/saída em arquivos;
- ***<cstring>***
  Biblioteca para manipulação de cadeias de caracteres e vetores. Utilizamos a função ***strcmp()***;
- ***<ctype.h>***
  Biblioteca referenciada para tratamento de *char* através das funções ***toupper()*** e ***isdigit()***;
- ***<cstdlib>***
  Biblioteca padrão de utilidades gerais. Utilizamos seu metodo ***exit()***;
- ***<sstream>***
  Biblioteca para o uso das classes de *stringstreams* (fluxo de cadeias de caracteres).

1.2 Estrutura e modelagem

Embasados nas aulas da disciplina e em estudos individuais, utilizamos o conceito de orientação a objetos para modelagem do software proposto.

O software é composto por 3 classes: ***deus***, ***arquivo*** e ***excecao***.

- ***deus*** → Classe que define os atributos de um objeto *deus* e o constrói. É *friend* da classe *arquivo*;
- ***arquivo*** → Classe que define todas as operações a serem realizadas no arquivo (*inserção*, *ordenação*, *remoção*, *busca*), além das operações de interface com o usuário[1] (*menu*);
- ***sequence*** → Classe que contém os dados mínimos (*dados do cabeçalho*) de um *sequence* para uso auxiliar a fim de melhorar a legibilidade;
- ***excecao*** → Classe auxiliar que define e constrói o objeto excecao que é criado para formalizar exceções prédefinidas encontradas durante a execução do programa;

Abaixo verifica-se o resultado da modelagem ilustrado por um diagrama de classes.

1.3 O Algoritmo

Inserção

→ Receber o ***valor*** do usuário. Inicia a leitura do cabeçalho geral, Confere se ***qntSeq*** > 0

Se **FALSE**, incrementa ***numSeq***, ***posSeq***, ***numReg***. Escreve o cabeçalho do primeiro ***sequence*** na posição atual(Cabeçalho Sequence → ***proxSeq*** = -1, ***qntDados*** = 1. Escreve o ***valor***.

Se **TRUE**, ir para a posição ***firstSeq***, e compara se ***valor.nome*** > ***Dado.nome*** na primeira posicao

(A) Se **TRUE**,Atualiza as variáveis do sequence com os valores do cabeçalho e confere se ***proxSeq*** != -1

Se ***TRUE***, cria uma variavel ***auxPos*** = tellg()→ ***posAtual*** (long posAtual((auxPos-(5*sizeof(int)+sizeof(Dado)))/(TAM_CAB_SEQ + TAM_SEQ))) e usa-la para a nova variável ***percurso*** (long percurso = abs(proxSeq - posAtual)*(TAM_SEQ + TAM_CAB_SEQ)) que recebera o valor da quantidade que ira andar para o primeiro ***Dado*** do proximo sequence. Pula para o proximo sequence, no primeiro ***Dado*** e confere se ***valor.nome*** > ***Dado.nome*** na posicao atual

Se ***TRUE***, Volta em **(A)**

Se ***FALSE***, usa -***percurso*** (menos ***percurso***) para voltar para o primeiro ***Dado*** do sequence anterior. Confere se ***qntDados*** == 5

Se ***TRUE***, ***percurso*** (long percurso = abs(***NextFreeSeq*** - ***posAtual***)*(TAM_SEQ + TAM_CAB_SEQ)-2*sizeof(int)) salva o 4º e 5º Dado do Sequence em duas variaveis auxiliares (Dado aux1, Dado aux2), volta para o cabeçalho grava ***proxSeq*** em uma variável ***auxProxSeq***, atualiza ***proxSeq*** com o valor de ***nextFreeSeq***, atualiza o tamanho com 3 e pula até o proximo sequence vazio(usando percurso), escreve seu cabeçalho com os valores (***proxSeq*** e 2, nesta ordem), escreve ***aux1*** na primeira posição do Dado do sequence atual, ***aux2*** na segunda posição do sequence atual, volta para a primeira posicao e compara se ***valor.nome*** > ***Dado.nome***

Se ***TRUE***, função insere ordenado, volta para o cabeçalho e atualiza o tamanho;

Se ***FALSE***, volta para o sequence passado com -***percurso*** e insere ordenado, volta para o cabeçalho e atualiza o cabeçalho;

Se ***FALSE***, insere ordenado e atualiza o tamanho;

Se ***FALSE***, confere se ***qntDados*** == 5

Se ***TRUE***, ***percurso*** (long percurso = abs(NextSeq - posAtual)*(TAM_SEQ + TAM_CAB_SEQ)-2*sizeof(int)) salva o 4º e 5º Dado do Sequence em duas variaveis auxiliares (Dado aux1, Dado aux2) e dpois sobresescreve com o campo disponível em true, volta para o cabeçalho grava ***proxSeq*** em uma variável ***auxProxSeq***, atualiza ***proxSeq*** com o valor de @nextSeq, atualiza o tamanho com 3 e pula até o proximo sequence vazio, escreve seu cabeçaho com os valores (***proxSeq*** e 2, nesta ordem), escreve aux1 na primeira posição do Dado do sequence atual, aux2 na segunda posição do sequence atual, volta para a primeira posicao e compara se ***valor.nome*** > ***Dado.nome***

Se ***TRUE***, função insere ordenado, volta para o cabeçalho e atualiza o tamanho;

Se ***FALSE***, volta para o sequence passado com -***percurso*** e insere ordenado, volta para o cabeçalho e atualiza o cabeçalho;

Se ***FALSE***, insere ordenado, atualiza o cabeçalho

[1] não identificamos a necessidade da criação de uma nova classe para realizar somente essa operação (*menu*)

Se ***FALSE***,Atualiza as variáveis do sequence com os valores do cabeçalho e confere se ***qntDados*** == 5

Se ***TRUE***, ***percurso*** (long percurso = abs(NextSeq - posAtual)*(TAM_SEQ + TAM_CAB_SEQ)-2*sizeof(int)) salva o 4º e 5º Dado do Sequence em duas variaveis auxiliares (Dado aux1, Dado aux2) e depois sobrescreve com o campo disponível em true, volta para o cabeçalho grava ***proxSeq*** em uma variável ***auxProxSeq***, atualiza ***proxSeq*** com o valor de ***nextSeq***, atualiza o tamanho com 3 e pula até o proximo sequence vazio, escreve seu cabeçaho com os valores (***proxSeq*** e 2, nesta ordem), escreve ***aux1*** na primeira posição do Dado do sequence atual, ***aux2*** na segunda posição do sequence atual, volta para a primeira posicao e compara se ***valor.nome*** ¿ ***Dado.nome***

Se ***TRUE***, função insere ordenado, volta para o cabeçalho e atualiza o tamanho;

Se ***FALSE***, volta para o sequence passado com -***percurso*** e insere ordenado, volta para o cabeçalho e atualiza o cabeçalho;

Busca

Inicia a leitura do cabeçalho geral, confere se ***qntSeq*** > 0

Se ***FALSE***, exception(arquivoVazio)

Se ***TRUE***, ir para a posição ***firstSeq***, Atualiza as variáveis do sequence com os valores do cabeçalho e compara se ***nome*** > Dado.nome na primeira posicao

(B) Se ***TRUE***,e Atualiza as variáveis do sequence com os valores do cabeçalho e confere se @proxSeq != -1

Se ***TRUE***, cria uma variavel ***auxPos*** = tellg()→ @posAtual (long posAtual((auxPos-(5*sizeof(int)+sizeof(Dado)))/(TAM_CAB_SEQ + TAM_SEQ))) e usa-la para a nova variável @percurso (long percurso = abs(proxSeq - posAtual)*(TAM_SEQ + TAM_CAB_SEQ)) que recebera o valor da quantidade que ira andar para o primeiro Dado do proximo sequence. Pula para o proximo sequence, no primeiro Dado e confere se ***valor.nome*** > ***Dado.nome*** na posicao atual

Se ***TRUE***, volta em **(B)**

Se ***FALSE***, usa -***percurso*** para voltar no sequence anterior e procuraIgual();

Se ***FALSE***, percorre o sequence buscando o valor;

Se ***FALSE***, percorre o sequence atual buscando o valor;

Remoção

Inicia a leitura do cabeçalho geral, confere se ***qntSeq*** > 0;

Se ***FALSE***, exception(arquivoVazio)

Se ***TRUE***, realiza a busca do ***valor***, confere se encontrou

(B) Se ***TRUE***, vai até o sequence, vai no segundo campo do cabeçalho do sequence (***qntDados*** e o decrementa.

(B) Se ***FALSE***, nao possui o dado para remover.

2 A interface

A interface foi projetada de forma minimalista, porém intuitiva e funcional. Consiste em menus com entradas numéricas com demais operações informadas ao usuário.

Figura 1: Interface com software em execução

UNIVERSIDADE FEDERAL DE LAVRAS