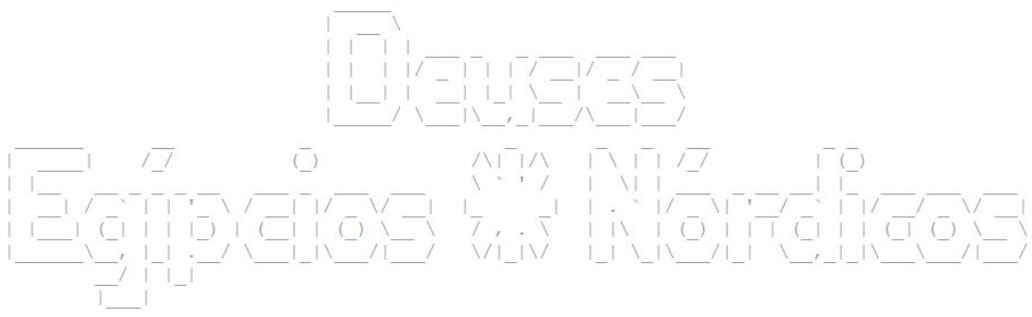
Relatório de Desenvolvimento de Software - Cadastro de Deuses Egípcios e Nórdicos

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 (**inserção ordenada**, **remoção**, **busca** e **impressão**) manipulando objetos de determinadas classes. 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*);
- ***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.

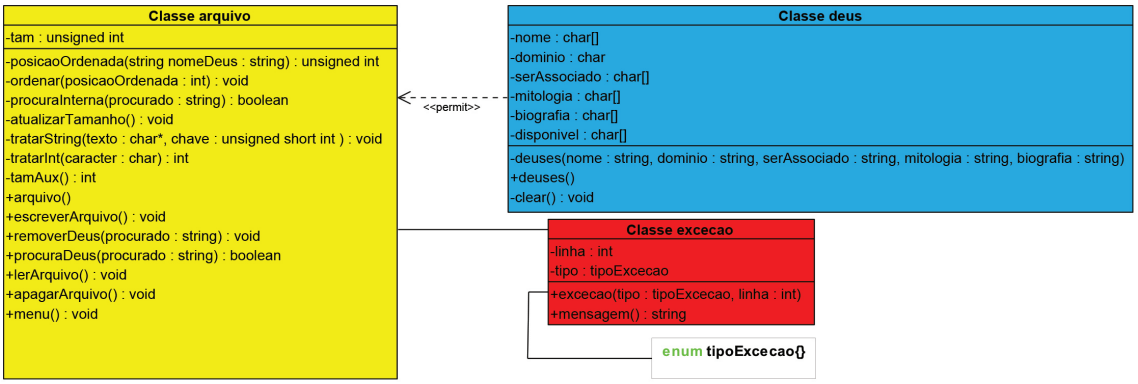


Figura 1: Diagrama de classes do projeto

1.3 Arquivos (Códigos-fonte)

Os códigos-fonte do projeto foram divididos de acordo com as classes, acrescentando um novo que contém a função básica *main()*. Cada classe possui dois arquivos: um com a extensão *.hpp* e outro com *.cpp*. Nos arquivos *.hpp* são contidas as instanciações das classes, assim como a declaração de seus métodos e atributos. Já nos *.cpp* é feita a implementação de todos os métodos de cada classe. Foi criado também, como facilitador de construção e compilação do código, um *Makefile*:

- *"deus.hpp" & "deus.cpp"*

  Atributos

  Privados

  1. ***char nome[]*** → Vetor de caracteres que armazena o nome do Deus;
  2. ***char dominio[]*** → Vetor de caracteres que armazena o domínio do Deus;
  3. ***char serAssociado[]*** → Vetor de caracteres que armazena o ser associado do Deus;
  4. ***char mitologia[]*** → Vetor de caracteres que armazena a mitologia na qual o Deus pertence;
  5. ***char biografia[]*** → Vetor de caracteres que armazena a biografia do Deus;
  6. ***bool disponivel*** → Variável booleana que define se um Deus possui espaço válido no arquivo para sobrescrita.

  Métodos

  Privados

  1. ***deus(string, string, string, string, string)*** → Construtor da classe *deus* que recebe os 5 vetores de caracteres de um Deus e o cria com o campo *disponivel* já em *false*;
  2. ***void clear()*** → Função que "limpa"os valores de todos os vetores de caracteres de um objeto *deus*.

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

- *"arquivo.hpp" & "arquivo.cpp"*

Atributos

Privados

1. ***unsigned int tam*** $\longrightarrow$ Variável inteira positiva que armazena a números de Deuses contidos no arquivo binário.

Métodos

Privados

1. ***unsigned int posicaoOrdenada (string texto)*** $\longrightarrow$ O objetivo da função é retornar a posição da *string texto* (passada por parâmetro) a ser inserida no arquivo para mante-lo ordenado. Ela lê o arquivo comparando *texto* com cada nome de deus no arquivo, retornando a primeira posição na qual texto é "menor" do que o Deus na posição do arquivo;
2. ***void ordenar(int posicaoBase)*** $\longrightarrow$ Função que recebe o valor retornado em *posicaoOrdenada(string)* e realiza o ordenamento dos Deuses antes da inserção. Usada pela função *escreverArquivo()*. A variável *posicaoBase* recebida como parâmetro (retornada de *posicaoOrdenada(string)*) é a posição na qual o Deus a ser inserido deve ser escrito para manter a ordenação da lista. A função analisa, primeiramente, se o Deus da posição a ser escrita possui o campo *disponível* como *true* (um Deus removido e disponível para sobrescrita) e, caso isso ocorra, ele "sai" da função. Após isso, verifica se o arquivo está vazio e, em caso afirmativo, também "sai" da função pois a inserção será feita de qualquer forma na posição 0 (primeira posição, que será retornada pela *posicaoOrdenada(string)*). A ordenação funciona na seguinte forma: Um contador (variável *pos*) recebe o índice da última posição do arquivo, faz a leitura do Deus da posição, o escreve na posição seguinte e, em seguida, o decrementa, repetindo isso até que o valor de pos seja menor que *posicaoBase* e o programa saia do laço de repetição e da função;
3. ***bool procuraInterna (string procurado)*** $\longrightarrow$ Função idêntica à *procuraDeus(string)* porém não imprime os dados do Deus, mesmo que a busca seja bem sucedida;
4. ***void atualizarTamanho ()*** $\longrightarrow$ Função que altera (atualiza) o atributo *tam* da classe *arquivo* considerando apenas os Deuses com campo *disponivel* em *false*;
5. ***void tratarString(char texto)*** $\longrightarrow$ Função usada para tratar as entradas de caracteres dos campos de cada Deus, padronizando a primeira letra de cada palavra dos campos *nome* e *mitologia* em caixa alta, iniciando os demais campos (*dominio*, *serAssociado* e *biografia*) também em caixa alta e finalizando o campo *biografia* com ponto final;
6. ***int tratarInt(char caracter)*** $\longrightarrow$ Função que trata o caso do input de dados inválidos para entradas de menus e/ou confirmações de ações da programa. Dados **válidos** para entradas de menus e/ou confirmações: 1 algarismo. Caso a entrada seja um dado inválido, será repetida a solicitação de entrada;
7. ***int tamAux()*** $\longrightarrow$ Função interna (É chamada como informação auxiliar nas funções *remover* e *posicaoOrdenada*) que retorna um inteiro que representa o tamanho total do arquivo (Considera, também, os Deuses que possuem o campo booleano *disponivel* com valor *true*, ou seja, "deuses removidos"). A função é parecida com a *atualizarTamanho()*, porem não altera nada.

Públicos

1. ***void escreverArquivo()*** $\longrightarrow$ Função que recebe como entrada do usuário os campos (exceto o *disponivel*) de um novo Deus a ser inserido, confere se já não existe um Deus com o mesmo nome, faz a chamada das funções *posicaoOrdenada(string)* e *ordenar(int)* e escreve o novo Deus na posição retornada por *posicaoOrdenada(string)*;
2. ***void removerDeus(string procurado)*** $\longrightarrow$ Função para remover um Deus do banco de dados. É utilizada a estratégia de marcar o espaço do arquivo a ser removido como DISPONÍVEL para sobrescrita. Como o propósito da aplicação não é realizar muitas remoções, utilizamos o método de sobrescrita de dados sem tratar o lixo (posições com campo *disponível* em *true*);
3. ***bool procuraDeus(string procurado)*** $\longrightarrow$ Função que recebe string *procurado* (um possível nome de Deus), lê o arquivo e retorna *true* (e imprime os dados do Deus) caso encontre no arquivo algum Deus com a mesma string que *procurado*;
4. ***bool lerArquivo ()*** $\longrightarrow$ Função que lê o arquivo (desde o início) e imprime todos os Deuses com campo *disponivel* em *false*;

5. ***void apagarArquivo()*** $\longrightarrow$ Função para apagar o Banco de Dados dos Deuses (deleta todos os dados contidos no arquivo!)
6. ***void menu()*** $\longrightarrow$ Função que imprime a interface para o usuário com as possíveis operações no arquivo.

- *"excecao.hpp" & "excecao.cpp"*

Atributos

Privados

1. ***int linha*** $\longrightarrow$ Variável inteira que armazena a linha do código onde foi chamada a exceção;
2. ***tipoExcecao tipo*** $\longrightarrow$ Variável enum que armazena o tipo de exceção chamada e o associa à um número inteiro.

```
enum tipoExcecao {digitoInvalido, dadoRepetido, arquivoProblema,
    excedeTamanho};
```

Métodos

Públicos

1. ***excecao (tipoExcecao tipo, int linha)*** $\longrightarrow$ Construtor da classe *excecao* que recebe um tipo de exceção e um inteiro correspondente à linha do código onde o objeto foi construído;
2. ***string mensagem()*** $\longrightarrow$ Retorna uma string de acordo com o *tipo* na qual o objeto *excecao* foi construído.

- *"main.cpp"*

```
#include "arquivo.hpp"
#include "deus.hpp"

using namespace std;

int main() {
        arquivo arquivoDeus;
        arquivoDeus.menu();
        return 0;
}
```

- *"Makefile"*

```
/* Makefile */
CPP = g++
CPPFLAGS = -Wall

OBJ = main.o deus.o arquivo.o excecao.o

main: $(OBJ)
        $(CPP) $(OBJ) -o main

main.o: main.cpp
        $(CPP) -c main.cpp -o main.o

deus.o: deus.cpp deus.hpp
        $(CPP) -c deus.cpp -o deus.o

arquivo.o: arquivo.cpp arquivo.hpp
        $(CPP) -c arquivo.cpp -o arquivo.o

excecao.o: excecao.cpp excecao.hpp
        $(CPP) -c excecao.cpp -o excecao.o

all: main

clean:
        find . -type f | xargs touch
        rm -f *.o
```

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 2: Interface com software em execução

UNIVERSIDADE FEDERAL DE LAVRAS