UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
CURSO DE BTI - ENGENHARIA DE SOFTWARE

RELATÓRIO DO LABORATÓRIO 10

Tratamento de Exceções

LABORATÓRIO DE LINGUAGENS DE PROGRAMAÇÃO I

Dionísio Dias Aires de Carvalho

NATAL

JUNHO / 2017

Dionísio Dias Aires de Carvalho

RELATÓRIO DO LABORATÓRIO 10

Relatório apresentado à disciplina de Linguagens de Programação I, correspondente ao laboratório 10 sobre a implementação do tratamento de exceções em C++ do curso de Bacharel em Tecnologia da Informação (BTI) da Universidade Federal do Rio Grande do Norte, sob orientação do **Prof. Dr. Sílvio Sampaio e Prof. Dr. Ewerton Cavalcante.**

NATAL

JUNHO / 2017

RESUMO

Projeto de implementação de uma biblioteca organizados em um namespace próprio. A biblioteca contém classes como pilha e fila, bem como, funções de ordenação e busca, todos genéricos. Para englobar todas estas funcionalidades, foi criado um namespace edb1. A biblioteca também de dispõe de tratamento para exceções.

SUMÁRIO

1. INTRODUÇÃO

Uma das principais razões para o uso de bibliotecas é a reutilização de código, permitindo que outros programas utilizem partes de códigos comuns. Para isto, foi apresentado em sala de aula um modelo de criação e implementação destes. Para aplicar isto, o laboratório 9 exigiu que fosse criada uma biblioteca contendo os algoritmos e estruturas de dados que foram vistas ao longo do semestre na disciplina IMD0029 – Estruturas de Dados Básicas I ou equivalente. Os componentes da biblioteca deveriam ser organizados em um namespace nomeado edb1.

2. DESENVOLVIMENTO

Para implementar a solução solicitada, foi criado um arquivo “dionisio.h” contendo os “#includes” para todos os outros arquivos de definição (classes e funções). Cada arquivo também recebeu a definição de namespace “edb1”. Estes outros arquivos foram:

- buscas.h → contendo todas as funções de busca (sequencial, binária e ternária) com versões recursiva e iterativas;
- ordenacoes.h → com funções de insertionsort, selectionsort, bubblesort, mergesort e quicksort;
- fila.h → classe que implementa uma fila;
- lista.h → classe que implementa uma lista ligada;
- pilha.h → classe que implementa uma pilha.

Um pequeno arquivo makefile foi escrito com o propósito de gerar a biblioteca descrita em suas versões estática e dinâmica (para Windows e Linux). Este script também gera dois programas que testam as funcionalidades da biblioteca (também com versões Windows e Linux).

3. DIFICULDADES

Para a implementação das funções genéricas, o uso do “extern 'C'” foi utilizado (conforme exposto em sala) porém, isso ocasionava erro de compilação visto que o 'C' não reconhece templates. Após uma pesquisa nas referências bibliográficas, o comando mudou para “extern 'C++'”.

4. ERROS

O tratamento de erros foi montado para as seguintes situações:

- Buscas – elemento não encontrado: ocorre quando é solicitada uma busca em um vetor e este não encontra o elemento;
- Ordenação – vetor vazio: ocorre na tentativa de ordenar um vetor que está vazio;
- Pilha – vazia: ocorre na tentativa de pop em uma pilha que está vazia;
- Pilha – cheia: fazer push em uma pilha cheia;
- Fila – vazia: ocorre na tentativa de pop em uma fila que está vazia;
- Fila – cheia: fazer push em uma fila cheia;
- Lista – insere: ocorre quando não há disponibilidade de endereço de memória para um novo objeto na lista;
- Lista – remove: erro na tentativa de remover o objeto de uma posição que não faz parte da lista.

5. CONCLUSÃO

Para testar todas as funcionalidades, criou-se um pequeno programa que testas todas as funcionalidades da biblioteca criada. O programa, utilizando-se das funcionalidades da biblioteca, executa os seguintes testes:

- cria uma variável dinâmica do tipo Lista, insere 3 valores, exibe-os na tela, remove todos e mais um para testar o erro de remoção inexistente;
- cria uma variável estática do tipo Pilha com 3 posições disponíveis, insere 3+1 valores (o 4º valor é o teste do erro de pilha cheia), exibe-os na tela e remove todos mais 1 para testar o erro de remoção em pilha vazia;
- cria uma variável estática do tipo Fila com 3 posições disponíveis, insere 3+1 valores (0 4º é o teste do erro de fila cheia) exibe-os na tela e remove todos mais 1 para testar o erro de remoção em fila vazia;
- cria um vetor, preenche-o e executa as funções de busca. Procura um valor que sabe-se que não existe para testar a mensagem de erro. Depois, procura um valor que sabe-se que existe, exibe a posição do valor dentro do vetor.Todas as buscas são, respectivamente:
  - busca sequencial iterativa;
  - busca sequencial recursiva;
  - busca binária iterativa;
  - busca binária recursiva;
  - busca ternária iterativa;
  - busca ternária recursiva.
- Cria um vetor e executa todas as funções de ordenação (insertionsort, selectionsort, mergesort, quicksort e bubblesort) sempre fazendo a seguinte sequência:
  - Preenche o vetor com valores aleatórios;
  - Exibe o vetor para mostrar a desordenação;
  - Chama a função de ordenação com um vetor vazio para testar a exceção;
  - Ordena o vetor através da função;
  - Exibe o vetor para mostrar a ordenação.

Portanto, o projeto foi concluído gerando arquivos de bibliotecas dinâmica e estática e binários que utilizam estas bibliotecas. A documentação sobre todo o código foi gerada através do Doxygen e está na pasta “doc”.

REFERÊNCIAS BIBLIOGRÁFICAS

Fórum Stack Overflow. Disponível em:
https://stackoverflow.com/questions/24229814/how-does-extern-c-work/24229844
Acesso em 18 de junho de 2017.

Site C++ Reference. Disponível em:
http://en.cppreference.com/w/cpp/language/class_template
Acesso em 18 de junho de 2017.