AED II - Trabalho I

Bruno Tomé - 0011254[1], **Cláudio Menezes - 0011255**[1]

[1]Instituto Federal de Minas Gerais (IFMG)
São Luiz Gonzaga, s/nº - Formiga / MG - Brasil

ibrunotome@gmail.com, claudiomenezio@gmail.com

Abstract. *Report about the first job of AED II, working with tree and expressions.*

Resumo. *Relatório sobre o primeiro trabalho de AED II, trabalhando com árvores e expressões.*

1. Introdução

O objetivo deste trabalho era a avaliação de uma expressão lida via console ou via arquivo, esta expressão é lida na forma infixa e além de avaliá-la, o programa transforma a expressão em prefixada e pós-fixada, a partir de pilhas e árvores.

2. Implementação

Decidimos seguir a maioria das dicas passadas em sala de aula, usar pilha e pilha de árvores. Temos um TAD pilha de char e um TAD pilha de árvore, talvez se tivéssemos pensado desde o início em usar union um desses TADs seria economizado.

Nossa função main() apenas lê as opções do menu de entrada e saída de dados e chama as funções contidas no arquivo funcoes.c

Fizemos uma função que lê via console e outra que lê via arquivo texto, a primeira retorna uma string expressão, a segunda chama a função separa_string em um while até terminar o arquivo.

A função separa_string recebe a expressão, e a opção de saída do usuário, nela são chamadas funções como: criatronco, percorre_avalia, criar inserir e remover em pilhas, inserir em pilhas de árvore, criar e inserir elementos em árvores. É a função que trata a expressão para que ela seja inserida na árvore de forma correta, respeitando as precedências e retorna as impressões no console, ou em arquivo, de acordo com o menu_saida que foi passado através do main.

Nosso TAD pilha foi traduzido do TAD pilha em pascal que usamos em AED 1, adaptamos ele para trabalhar com árvores, tornando assim uma pilha de árvores.

O TAD arvore_binaria fazia apenas as funções básicas de uma árvore, conseguimos ele através de uma equipe www.geeksbr.com. Nele criamos as funções criaFolha, criaTronco e percorre_avalia, esta última tivemos uma dica do professor Diego em como avaliar a árvore apenas com recursão, sem a necessidade de pilha.

Mais detalhes estão comentados nos códigos fonte.

2.1. Como executar o programa

Abra o Terminal e digite:

cd <DIRETÓRIO>

gcc main.c funcoes.c pilha.c arvore_binaria.c pilha_arvore.c -omain.bin -Wall -pedantic -ansi

./main.bin

No windows, você pode rodar via netbeans ou se tiver um terminal batch basta seguir os comandos acima.

3. Descrição dos testes realizados

Fizemos testes seguindo o que foi proposto no enunciado, usando expressões com apenas um dígito, tratamos as ordens prioritárias de execução, e testamos com as seguintes expressões:

2 + 2 * 5

2 * 2 + 5

2 * (2 + 5)

((5 + 5) * 2) + 8

((5)) + 5 * 2

(((5 + 5)) * 2) + 8

O teste foi bem sucedido em todas esses exemplos.

4. Estudo da Complexidade

4.1. Leitura e impressão do arquivo:

A complexidade para ler e imprimir o arquivo varia com o número de expressões que ele possui. Isso nos leva a uma complexidade de O(n).

As funções de empilhar, e desempilhar utilizadas nas pilhas de árvore e de caractere, são de complexidade O(1).

Para a árvore as funções de inserção, e percorrer árvore, a complexidade é O(log n).

Portanto temos que a soma das complexidades das funções realizadas pelo programa acima:

O(n) + O(n) + O(1) + O(log n) = O(n)

Enfim temos que a complexidade do programa como um todo é da ordem de O(n).

4.2. Programa rodando a partir da leitura via console:

Guardar a expressão digitada pelo usuário é O(1).

As funções de empilhar, e desempilhar utilizadas nas pilhas de árvore e de caractere, são de complexidade O(1).

Para a árvore as funções de inserção, e percorrer árvore, a complexidade é O(log n).

Portanto temos que a soma das complexidades das funções realizadas pelo programa acima:

O(1) + O(1) + O(log n) = O(log n)

Enfim temos que a complexidade do programa como um todo é da ordem de O(log n).

5. Conclusão

O trabalho nos proporcionou um imenso aprendizado sobre a estrutura árvore, suas vantagens e desvantagens e como usá-la da forma adequada aos nossos problemas. Adquirimos conhecimentos em recursão e no uso de ponteiros, estes dois foram os pontos mais difíceis de se abstrair. Aproveitamos a oportunidade para pedir quem sabe, em uma semana livre que tivermos, um aprofundamento em recursão.

6. Bibliografia

TAD arvore_binaria = www.geeksbr.com TAD pilha = TAD pilha traduzido do pascal Professores Diego e Everthon nos ajudaram a abstrair o funcionamento da recursão e como ela poderia nos ajudar a avaliar a expressão.