Arvore de Expressão
IFMG - Campus Formiga

Guilherme Cardoso Silva

[1] Algoritmos e Estrutura de Dados II – Instituto Federal de Minas Gerais (IFMG)
Caixa Postal 35570-000 – Formiga – MG – Brasil

godgcs@gmail.com.br

1. Introdução

O objetivo geral deste trabalho é o projeto e a implementação de um sistema que construa uma arvore de expressão. Sendo que para implementação deste método deve ser utilizado estrutura de dados de arvores binarias.

2. Implementação

Para compilar é necessario apenas utilizar *gcc *.c -o "NomeExecutavel"* para gerar o executavel.

A implementação deste projeto foi codificada em uma TAD chamada ArvoreExpressoes para que futuramente possa ser reaproveitada em outros projetos.

2.1. Estrutura

A estrutura de dados implementada para este metodo foi a seguinte:

```
struct arvore{
        int Tipo;
        int Operando;
        int Operador;
        struct arvore *Esq;
        struct arvore *Dir;
};
struct nodoPilha{
        struct nodoPilha *Prox;
        struct arvore *Arvore;
};
struct pilha{
        struct nodoPilha *Inicio;
};
struct info{
        int op1;
        int op2;
        char arqEnt[255];
```

char arqSai[255];
};
*typedef struct nodoPilha *NodoPilha;*
typedef struct info Info;
*typedef struct arvore *Arv;*
*typedef struct pilha *Pilha;*

A estrutura da arvore seria definida por um nó de *struct arvore*,ou Arv (Definição publica), onde cada indice do vetor seria composto pelas seguintes propriedades:

- **Tipo:** Tipo do nó da arvore, podendo ser operando ou operador;
- **Operando:** Valor do nó da arvore caso seja do tipo operando;
- **Operador:** Valor do nó da arvore caso seja do tipo operador;
- **Esq:** Ponteiro para sub arvore Esquerda;
- **Dir:** Ponteiro para sub arvore Direta.

Além da Arvore foi implementada uma pilha de arvores que seria utilizada nos algoritimos para construção dela a medida que o arquivo fosse sendo lido.

A struct info, ou Info (Definição publica), foi criada para que fosse criado um menu inicial que armazena as opções de entrada para execução do algoritimo.

2.2. Principais Funções

A TAD *ArvoreExpressoes.c* foi implementada de acordo que pudesse armazenar uma expressão numerica em uma arvore para que possa ser calculado o valor, para isto foram implementadas as seguintes funções:

- **void menu(Info* Entradas):** Esta função é responsavel por definir como será executado o programa, se vai ser utilizado um arquivo de entrada ou o proprio terminal, se vai ser utilizado um arquivo de saida ou o proprio terminal para gerar os resultados.
- **void LeExpressao(char Expre[], FILE *arquivo):** Esta função recebe como parametro uma expressão e um possivel arquivo de saida.
  Primeiramente é criado uma arvore referente a expressão, para que ela seja criada é utilizada a pilha de arvores, onde a cada valor numero lido é armazenado na pilha e a cada operando lido, são retirados dois valores da pilha e criado uma arvore com ambos.
  Para a construção da arvore com os valores retirados da pilha é seguido o seguinte padrão: O primeiro valor retirado é inserido na direita da nova arvore, já o segundo valor retirado é inserido na esquerda da arvore. Isto faz com que seja gerado uma arvore invertida, porém todos os algoritimos foram implementados desta forma então os resultados são gerados corretamentes.
  Após isto ela imprime os resultados:
  - Resultado da operação
  - Impressão em preOrdem
  - Impressão em posOrdem
  - Impressão em inOrdem

- **void LeExpressoesArquivo(char Arq[], FILE *arquivo):** Esta função recebe o nome de um arquivo de entrada e um arquivo de saida.
  Basicamente ela irá realizar um loop no arquivo de entrada lendo todas as espressões em uma string e então chamando a função LeExpressão que irá gerar a arvore e os resultados para cada expressão do arquivo.
- **float ArvoreCalcula(Arv Arvore):** faz um calculo do resultado da arvore de expressão.
- **void ArvoreImprimePosOrdem(Arv Arvore, FILE *arquivo):** Faz a impressão da arvore de expressão em formato posOrdem.
- **void ArvoreImprimePreOrdem(Arv Arvore, FILE *arquivo):** Faz a impressão da arvore de expressão em formato preOrdem.
- **void ArvoreImprimeInOrdem(Arv Arvore, FILE *arquivo):** Faz a impressão da arvore de expressão em formato InOrdem.
  Este formato é o padrão utilizado pra escrita e leitura de humanos, e para que seja visiveis as prioridades nas somas são impressos parenteses que detalham esta ordem, para isto foram criadaos duas funções que tratam estes casos, elas serão detalhadas a seguir.
- **void ImprimeParenteseEsquerdo(Arv Arvore, char parentese, FILE *arquivo):** Caso exista uma soma ou uma subtração que devem ser realizados antes de uma multiplicação ou divisão é necessario uma parentização para deixar isto explicito. Uma arvore de expressão pode ser construida sendo ramificada para a esquerda ou para a direita, está função trata o caso da esquerda, se um nó de multiplicação ou divisão contem uma ramificação para o lado esquerdo, que é referente a um nó de soma ou subtração é necessario que exista um parentese para demonstrar a prioridade na soma.
  Nestes casos é impresso o parentese passado por parametro.
- **void ImprimeParenteseDireito(Arv Arvore, char parentese, FILE *arquivo):** Está função faz a mesma coisa que a ImprimeParenteseEsquerdo porém para os casos em que a ramificação da arvore é feita pelo lado direito.

3. Descrição dos Testes

Para que fosse possivel avaliar os metodos implementados, foram realizados alguns testes tanto por terminal quanto por arquivo de texto.

4. Resultados

Os resultados foram satisfatorios, onde todos os calculos coincidiram com os desejados, e todas as saidas referentes a cada expressão foram corretas.

5. Conclusão

Este trabalho resultou em uma grande experiencia e conhecimento com relação a estrutura de dados em forma de arvore. Além disto pode-se exercitar com um problema pratico de como utilizar arvores para armazenar expressões numericas, percorrendo seus nós para calcular e imprimir os resultados