Estudo sobre árvores de expressão

Arthur Alexsander Martins Teodoro[1]

[1]Instituto Federal de Minas Gerias - Campus Formiga

arthurmteodor@gmail.com

Introdução

Uma expressão aritmética pode ser representado usando uma árvore binária. Nesta árvore, os nós folhas são os operandos e os nós internos são associados aos operadores. Este tipo de árvore possui a denominação de árvore de expressão. Quando se possui uma árvore de expressão é possível obter todos os tipos de notação usadas, a Notação Polonesa que consiste nos operandos antes dos operadores, a Notação Infixa que é os operandos entre os operadores e a Notação Polonesa Inversa que é os operandos depois dos operadores. Também é possível calcular o valor de uma expressão usando uma árvore de expressão.

O objetivo deste trabalho era o projeto e implementação de um sistema interpretador e conversor de expressões aritméticas usando o conceito de árvore de expressão. Neste trabalho a entrada *software* consiste em uma notação pósfixa (Polonesa Inversa), que, a partir desta, é gerado a árvore de expressão e as outras formas de notação.

Implementação

Nesta seção deste documento será mostrada as decisões de implementação, considerações, demonstração da estrutura de dados criada e modo de compilação e execução.

Descrição sobre as decisões de projetos e implementação

Para a resolução do problema foi usada a linguagem de programação C. O trabalho foi codificado usando o editor de texto Sublime Text 3[1], compilador GCC versão 5.4.0, sistema operacional GNU/Linux Ubuntu 16.04 e controle de versão Git.

Para a resolução do problema foi necessária a criação de três TAD's, árvore binária(*arvore.c, arvore.h*), uma pilha que armazena valores do tipo ponto flutuante(*pilha.c, pilha.h*) e uma pilha que armazena ponteiros para árvores(*pilhaArv.c, pilhaArv.h*). Além destas três TAD's, foi criado arquivos para a criação da expressão e avaliação da expressão(*expressao.c, expressao.h*). Foi criado também um arquivo que contem a função main(*main.c*), que neste é realizada somente a verificação de arquivos e chamadas para funções das TAD's e do Avaliador de Expressão.

Compilação e Execução

Para a compilação do sistema desenvolvido, foi criado um arquivo *Makefile*, logo para a compilação do sistema deve-se usar somente o comando make. Para a execução do sistema, existe algumas diretivas que devem ser levadas em consideração. O sistema usa argumentos para identificar o tipo de entrada e saída do sistema. Caso o sistema possua entrada e saída via arquivo texto, o modo de execução deverá ser:

[1]www.sublimetext.com

```
$./main.out -i <nome_da_entrada> -o <nome_da_saida>
```

Caso seja desejada entrada ou saída padrão, a diretiva -i ou -o deve ser omitida. Caso o desejo é o uso de entrada e saída padrão, o sistema deverá sem executado sem ambas as diretivas.

Estruturas Criadas

Para a resolução do problema, além das pilhas foi criada uma estrutura de árvore binária. Como um nó da árvore pode possuir tanto valores numéricos ou o tipo de operação a ser feita, a estrutura criada possui ambos os campos, sendo um campo do tipo ponto flutuante para armazenamento do valor numérico, um campo do tipo char para armazenar o tipo de operação a realizar entre os filhos além de um campo do tipo inteiro que armazena que tipo de valor aquele nó da árvore possui, além dos ponteiros para os próximos nós.

```
struct arv
{
  int tipoDado;
  char operador;
  float operando;
  struct arv* esq;
  struct arv* dir;
};
```

Testes Realizados

Para ser verificado o correto funcionamento do sistema foi usada uma bateria de testes criada pelo próprio autor do sistema. Tal bateria possuía algumas expressões aritméticas. Destas expressões, algumas possuíam as operações de subtração e divisão, que são problemáticas do ponto de visto do sistema, uma vez que a ordem dos fatores alteram o resultado. Para verificar se o sistema realizava o cálculo correto do valor da expressão, foi usada o comando dc existente em distribuições GNU/Linux. DC nada mais é do que uma uma calculado de pilha que pode receber como entrada uma expressão em notação pósfixa.

Complexidade do sistema

Como o sistema usa basicamente uma árvore binária e seus tipos de caminhamento, a complexidade assintótica do sistema é *O(n)*, uma vez que que esta é a complexidade de caminhar entre os nós da árvore.

Conclusão

Foi concluído que a árvore de expressão é um bom método para avaliação de expressões. Além disso, foi reforçada a programação de árvores, além do uso de recursão para a resolução de problemas.