Relatório Notação Polonesa

Rodrigo Theodoro Rocha

15/0157967

Introdução

A notação polonesa reversa, refere-se a forma de escrever expressões algébricas na qual a posição relativa dos operadores é após dois operandos (posfixa). Por exemplo, a expressão A+B*C, escrita na forma usual (infixa) pode ser reescrita na forma posfixa respeitando-se a precedência dos operadores. Neste caso, a multiplicação precede a operação de adição, na ausência de parênteses que indiquem o contrário. Portanto a expressão A+B*C pode ser interpretada como A+(B*C) e convertida para a forma posfixa através das 4 etapas a seguir:

i. A + (B * C) parêntese para obter ênfase;
ii. A + (BC*) conversão da multiplicação;
iii. A(BC*)+ conversão da adição;
iv. ABC*+ forma posfixa

Para a conversão somente é necessário lembrar a ordem de precedência dos operadores e as restrições impostas pelos parênteses na expressão. A ordem de precedência é descrita a seguir:

1. Exponenciação
2. Multiplicação/Divisão
3. Adição/Subtração

Abaixo listou-se alguns exemplos de expressões aritméticas convertidas da forma infixa (coluna esquerda) para a posfixa (coluna direita):

```
A+B-C       AB+C-
(A+B)*(C-D) AB+CD-*
(A+B)*C     AB+C*   ①
A+(B*C)     ABC*+   ②
```

① Apesar da forma infixa requerer o uso de parênteses para alterar a ordem de precedência da adição sobre a multiplicação, na forma posfixa o parêntese não é necessário pois a ordem dos operandos determina as etapas de resolução da expressão.

② Como a escrita envolvendo a forma posfixa muitas vezes não é trivial, sua avaliação envolve uma forma alternativa de realizar as operações. Imagine que os valores dos operandos nesta expressão seja A=3, B=2 e C=4. A avaliação do resultado da expressão <2> na forma infixa é trivial e na forma posfixa não, portanto, o algoritmo para sua resolução será explicado na próxima seção.

Algoritmo para avaliação das expressões na forma posfixa

Cada vez que um operador é encontrado numa expressão posfixa ele refere-se aos 2 operandos imediatamente anteriores. Por exemplo, em <2> percorre-se a expressão até encontrar o primeiro operador (multiplicação) e efetua-se a multiplicação dos operandos B e C. O resultado é, então armazenado e prosseguimos até encontrar o próximo operador (adição). Adiciona-se A ao resultado

da multiplicação de B e C. Este é o resultado final da avaliação da expressão em questão.

A implementação desse procedimento envolve a utilização de pilhas, no qual, empilham-se os operandos até deparar-se com um operador e, então, efetua-se a aplicação deste operador contra os 2 últimos operandos empilhados.

```
stk = a pilha vazia;
while (nao terminar de percorrer a expressão posfixa){
  c = próximo caractere da entrada;
  if (c é um operando){
    push (stk, c);
  }
  else {
    /* c é um operador */
    opnd2 = pop (stk);
    opnd1 = pop (stk);
    value = resultado da aplicação do operador aos 2 operandos anteriores;
    push (stk, value);
  } /* fim else */
} /* fim while */
```

Note que para avaliação de uma expressão posfixa os termos dela podem ser qualquer número inteiro, fracionado, negativo ou positivo. Portanto, na implementação considerou-se os operandos da expressão como qualquer número pertencente ao conjunto dos Reais.

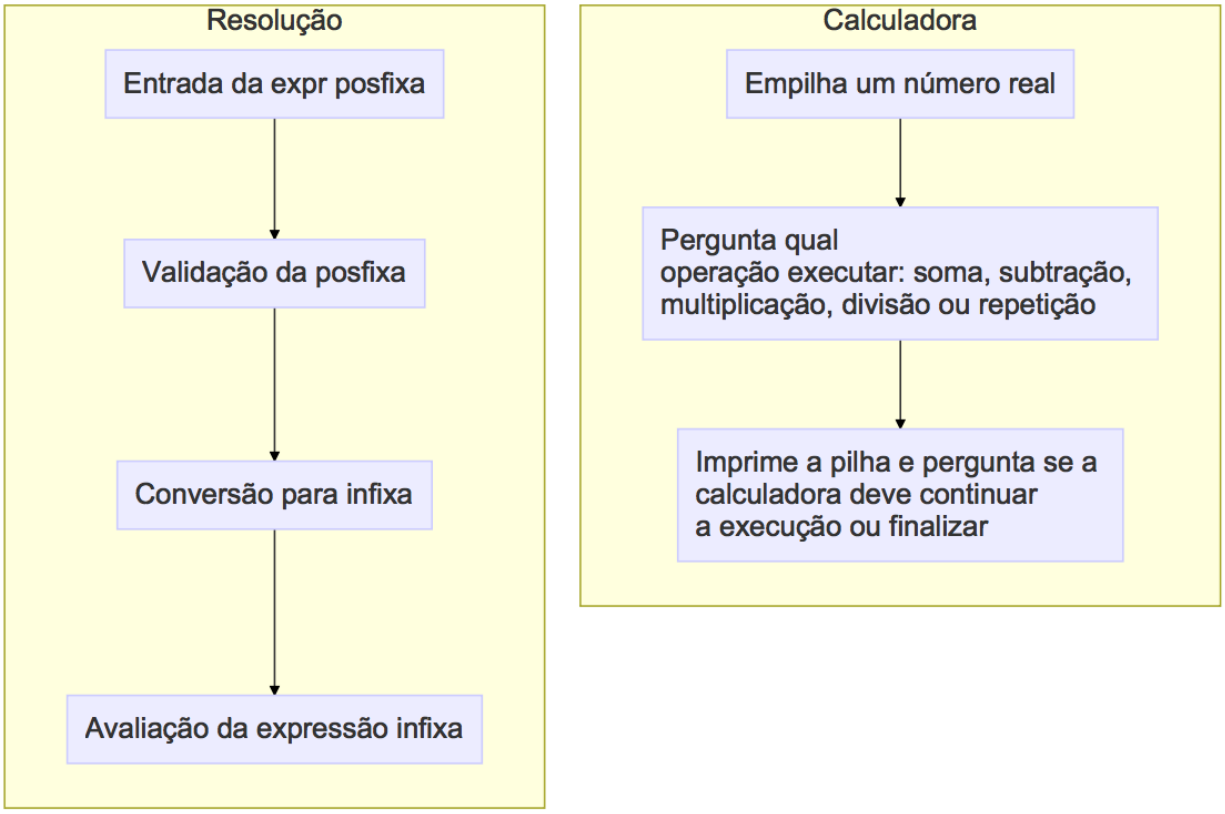
Como a implementação considera qualquer número real, adotou-se duas notações especiais quanto a forma de escrita da expressão de entrada a ser avaliada. Primeiro, separam-se os termos da expressão via espaços. Segundo, o sinal negativo dos termos negativos não podem estar separados da parte numérica do termo. Por exemplo, a expressão posfixa `2+(-3*4)` deve ser editada para a forma `2 + ( -3 * 4 )`, isto é, respeitando-se a notação explicada acima para uma correta transformação e avaliação da expressão infixa.

Implementação da conversão da forma posfixa para infixa

A função `infixa_to_posfixa ()` usa como entrada uma expressão posfixa válida respeitando a notação e depois retorna sua forma infixa. Um ponto importante na implementação dessa função é lidar com os caracteres especiais (, { e [que servem como separadores e mudam a importância dos operadores.

Arquitetura

O diagrama abaixo representa os dois modos de opção do arquivo binário `bin/trabalho1`.



Instalação e execução dos Testes

Execute as seguintes etapas para instalar o binário e executar os testes:

1. Dentro do diretório principal execute `mkdir build`;
2. Entre no subdiretório com `cd build`;
3. Execute `cmake ..`;
4. Execute o comando `make` para compilar as bibliotecas na pasta `bin/include` e o binário `trabalho1` no diretório `bin`;
5. Para executar os testes: `make test` no diretório `build`.

Smoke Tests

Tópico	Etapa	Arquivo	Ação	Expectativa	Taxa de Sucesso
Pilha Genérica	Estrutura de dado primária	`src/pilha.c`	Implementar funções básicas (push, pop, isEmpty...)	Implementar uma estrutura genérica de pilhas na qual os elementos podem ser do tipo `char`, `int` ou `float`.	Unit tests passaram com 100% de sucesso (`tests/check_pilha.c`)
Validação da Expressão	Resolução de Expressão	Função `check_infixa()` (`src/posfixa.c`)	Checar se a entrada na forma infixa é válida	Dada uma expressão infixa (usual) tanto na forma numérica quanto com variáveis alfabéticas, avalia a correta colocação dos parênteses	Unit tests passaram com 100% de sucesso (`tests/check_posfixa.c` linhas 9-20)
Validação da Expressão	Resolução de Expressão	Função `check_infixa()` (`src/posfixa.c`)	Checar se a entrada na forma infixa é válida	Dada uma expressão infixa (usual) tanto na forma numérica quanto com variáveis alfabéticas, avalia a correta colocação dos parênteses	Unit tests passaram com 100% de sucesso (`tests/check_posfixa.c` linhas 9-20)

Tópico	Etapa	Arquivo	Ação	Expectativa	Taxa de Sucesso
Entrada e saída de Dados	Inicio do Programa	Função `main()`	Construir o Menu e avaliar se diferentes expressões funcionam	Considerar parênteses, colchetes e chaves bem como qualquer número pertencente aos reais.	Testes na execução do programa
Validação da Expressão	Resolução de Expressão	Função `check_infixa()` (`src/posfixa.c`)	Checar se a entrada na forma infixa é válida	Dada uma expressão infixa (usual) tanto na forma numérica quanto com variáveis alfabéticas, avalia a correta colocação dos parênteses	Unit tests passaram com 100% de sucesso (`tests/check_posfixa.c` linhas 9-20)
Transformação da forma infixa para posfixa	Resolução de Expressão	Função `avalia_posfixa()` (`src/posfixa.c`)	Transformar a expressão na sua forma posfixa	Dada uma expressão válida na forma infixa (usual), transforma-a para a forma posfixa	Unit tests passaram com 100% de sucesso (`tests/check_posfixa.c`)

Tópico	Etapa	Arquivo	Ação	Expectativa	Taxa de Sucesso
Avaliação da expressão na forma posfixa	Resolução de Expressão	Função `avalia_posfixa()` (`src/posfixa.c`)	Calcula o valor final de uma expressão posfixa	Entrando com uma expressão na forma posfixa, na qual os operadores encontram-se separados dos operandos por um ou mais espaços, a função realiza as operações e calcula o valor final. Quando a entrada possui operandos negativos, o sinal deve se encontrar adjacente ao número (sem separação)	Unit tests passaram com 100% de sucesso (`tests/check_posfixa.c`)
Construção da Calculadora	Calculadora	Funções implementadas no arquivo `src/calculadora.c`	Executa as operações básicas especificadas	Usar uma pilha como estrutura de dados e considerar números inteiros, fracionados, positivos e negativos	Testes através da execução e avaliação dos resultados