Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20092
Professor: David Menotti (menottid@gmail.com)

Trabalho Prático 3 – Expressões Aritméticas
Recursividade e Pilhas

Valor: 0,5 pontos (5% da nota total) **Data de entrega: 20/10/2009**
Documentação não-Latex: -0,1 pontos
Impressão não-frente-verso: -0,05 pontos

Objetivos

Consiste em utilizar os conceitos de pilhas e recursividade para a solução de um problema, o cálculo de expressões aritméticas. Esse trabalho é baseado em um dos problemas da Olímpiada Brasileira de Informática do Instituto de Computação da UNICAMP [1].

Descrição

A disseminação dos computadores se deve principalmente à capacidade de eles se comportarem como outras máquinas, vindo a substituir muitas destas. Esta flexibilidade é possível porque podemos alterar a funcionalidade de um computador, de modo que ele opere da forma que desejarmos: essa é a base do que chamamos programação.

Sua tarefa é escrever um programa que faça com que o computador opere como uma calculadora simples. O seu programa deve ler expressões aritméticas e produzir como saída o valor dessas expressões, como uma calculadora faria. O programa deve implementar apenas um subconjunto reduzido das operações disponíveis em uma calculadora: somas, subtrações, multiplicações e divisões.

Entrada

A entrada deve ser lida de um arquivo a ser fornecido como parâmetro em linha de comando (`int main(int argc, char**argv)`) composta de vários conjuntos de testes, onde cada linha do arquivo contém uma expressão aritmética a ser avaliada, no seguinte formato:

$$X_1\ s_1\ X_2\ s_2\ X_3\ s_3 \ldots X_{m-1}\ s_{m-1}\ X_m$$

onde

- X_i , $1 \leq i \leq m$, é um operando ou uma nova expressão que pode conter parentesis de abertura `'('` e fechamento `')'`;
- s_j, $1 \leq j < m$, é um operador, representado pelos símbolos `'+'`, `'-'`, `'*'` ou `'/'` ;
- Não há espaços em branco entre operandos e operadores.

O final da entrada é indicado pelo final do arquivo.

Lembre-se de respeitar a ordem de precedência entre os operadores. Isto é, a multiplicação (*) e divisão (/) tem precedência sobre a adição (+) e subtração (–).

Exemplo de Entrada

```
4
3+5*2
3*5+2
(3+5)*2
(20/(5*2)+7-(10*5))+2
1+2+3+4+5+6+7+8+9+10
```

Saída

Para cada expresão de teste do arquivo de entrada, seu programa deve produzir três linhas. A primeira linha deve conter um identificador da expressão, no formato "Teste n", onde n é numerado a partir de 1. Na segunda linha deve aparecer o resultado encontrado pelo seu programa. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
4

Teste 2
13

Teste 3
17

Teste 4
-39

Teste 5
55
```

(esta saída corresponde ao exemplo de entrada acima)

Restrições

Os operandos são valores inteiros limitados entre 0 e 100

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20092
Professor: David Menotti (menottid@gmail.com)

O que deve ser entregue

- Código fonte do programa em C ou C++ (bem identada e comentada).
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:
  1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Muito importante: os códigos utilizados na implementação devem ser inseridos na documentação.
  3. Análise de Complexidade: estudo da complexidade de tempo e espaço das funções implementados e do programa como um todo (notação O).
  4. Listagem de testes executados: os testes executados devem ser simplesmente apresentados.
  5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto quando da sua utilização
  7. Em Latex: Caso o trabalho não seja elaborado/escrito em latex, perde-se 0,1 pontos.
  8. Impressão: Caso o trabalho não seja impresso usando modo frente-verso, perde-se 0,05 pontos.
  9. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

  Obs: Veja modelo de como fazer o trabalho em latex:
  http://www.decom.ufop.br/prof/menotti/aedI092/tps/modelo.zip

Como deve ser feita a entrega:

A entrega DEVE ser feita via Moodle (www.decom.ufop.br/moodle) na forma de um **único** arquivo zipado, contendo o código, os arquivos e a documentação. Também deve ser entregue a documentação impressa na próxima aula teórica após a data de entrega do trabalho.

Comentários Gerais:

- Clareza, identação e comentários no programa também vão valer pontos;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero;
- Trabalhos entregues em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.

Referências

[1] *Olimpíada Brasileira de Informática* - http://olimpiada.ic.unicamp.br/