CIV 2802 – Sistemas Gráficos para Engenharia – 2011.1

4º Trabalho: Programação básica em C++: Classes

Implementação de uma calculadora RPN (Reversed Polish Notation)

Entrega: 05/abril/2011

Pede-se complementar um programa que implementa uma calculadora RPN. O código incompleto do programa e os arquivos de projeto para o Visual C++ 2008 (VC9) estão disponíveis através do *link* na internet: http://www.tecgraf.puc-rio.br/ftp_pub/lfm/civ2802-111-trab4-vc9.zip. Para fazer as complementações necessárias é preciso que se entenda completamente o programa fornecido. **Este entendimento faz parte do trabalho**.

Pede-se um executável do programa final e os arquivos que foram modificados para implementar o trabalho. Pede-se também, em papel, somente as linhas que foram adicionadas nos arquivos. Estas linhas devem ser indicadas da seguinte forma:

```
COMPLETE AQUI: XX
LINHAS ADICIONADAS
```

Arquivo "main.cpp"

```
#include "rpn.h"
#include "iocalc.h"

// Rotina principal do programa para calculadora RPN (Reversed Polish Notation).
int main()
{
  RPN *hp = new RPN;

  IOcalc io;

  while(1)
  {
    io.prompt();
    io.get();

    float val;
    if(io.isNumber(&val))
    {
      hp->enter(val);
    }
    else
    {
      char c;
      io.getChar(&c);
      switch(c)
      {
        case '+': hp->sum(); break;
        case '-': hp->sub(); break;
        case '/': hp->div(); break;
        case '*': hp->mul(); break;
        case 'q': delete hp; return 0;
        default:  io.message(IOcalc::MSG_UNKNOWN_OP);
      }
    }
    hp->show();
  }

  return 1;
}
```

Arquivo “stack.h”

```
#ifndef _STACK_H
#define _STACK_H

const unsigned int MAX = 50;

class StackIterator;

// Classe para pilha de numeros.
class Stack {
   // Indica classe "friend" para percorrer (interar) elementos da pilha.
   friend class StackIterator;

 private:
   // Atributo: indice do elemento do topo da pilha.
   int top;
   // Atributo: pilha de numeros e' armazenada como um vetor.
   float *elems;

 public:
   // Construtor com parametro default "size" definido pela constante "MAX".
   // Inicializa "top" com valor nulo, indicando que a pilha esta' vazia.
   // Aloca dinamicamente vetor de numeros com tamanho "size".
   Stack(unsigned int size = MAX) { /*** COMPLETE AQUI: 01 ***/ }

   // Destrutor: elimina vetor de numeros.
   ~Stack()                       { /*** COMPLETE AQUI: 02 ***/ }

   // Metodo para inserir um valor dado no topo da pilha de numeros.
   // Depois de inserir, incrementa o indice "top".
   void push(float val)           { /*** COMPLETE AQUI: 03 ***/ }

   // Metodo para retornar o valor do topo da pilha de numeros.
   // Para ter acesso ao valor do topo, e' necessario decrementar antes o
   // indice "top".
   float pop()                    { /*** COMPLETE AQUI: 04 ***/ }

   // Metodo para verificar se a pilha esta' vazia.  Retorna "true" se a
   // pilha de numeros estiver vazia; senão retorna "false".
   bool empty()                   { /*** COMPLETE AQUI: 05 ***/ }
};

// Classe "friend" que cria um objeto para percorrer (iterar) os elementos
// de uma pilha da classe "Stack".
class StackIterator {
 private:
   int current;
   Stack *st;
 public:
   StackIterator(Stack *s) { st = s; current = 0;         }
   bool end()              { return st->top == current;   }
   float next()            { return st->elems[current++]; }
};

#endif
```

Arquivo “rpn.h”

```
#ifndef _RPN_H
#define _RPN_H

#include "stack.h"

// Classe para calculadora "Reversed Polish Notation" - RPN.
class RPN
{
 private:
   // Atributo: pointeiro para um objeto da classe Stack.
   Stack *stk;

 public:
   RPN();
   RPN(unsigned int size);
   ~RPN();

 private:
   bool getop(float *val1, float *val2);

 public:
   void enter(float val);
   void sum();
   void sub();
   void mul();
   void div();
   void show();
};

#endif
```

Arquivo “rpn.cpp”

```
#include "rpn.h"
#include "iocalc.h"

// Construtor "default" que cria uma calculadora com a pilha com tamanho
// "default".
RPN :: RPN()
{
/*** COMPLETE AQUI: 06 ***/
}

// Construtor que cria uma calculadora com a pilha com tamanho dado pelo
// parametro "size".
RPN :: RPN(unsigned int size)
{
/*** COMPLETE AQUI: 07 ***/
}

// Destrutor de um objeto calculadora
RPN :: ~RPN()
{
/*** COMPLETE AQUI: 08 ***/
}

// Metodo privado "getop".
// Se a pilha contiver pelo menos dois numeros no topo, retira esses dois
// numeros da pilha, retorna esses dois numeros para serem utilizados como
// operandos em alguma das operacoes da calculadora.  Nesse caso, retorna
// "true" como valor de retorno.  Se a pilha estiver vazia, ou so' tiver
// um elemento, retorna "false".  Nesses casos, imprime uma mensagem
// indicativa do erro, mantendo a pilha com seu estado inalterado.
bool RPN :: getop(float *a, float *b)
{
/*** COMPLETE AQUI: 09 ***/
}

// Metodo publico "enter".
// Insere o valor fornecido no topo da pilha.
void RPN :: enter(float val)
{
/*** COMPLETE AQUI: 10 ***/
}
```

```
// Metodo publico "sum".
// Se existirem pelo menos dois numeros no topo da pilha, faz a soma desses
// dois numeros, retira os dois numeros do topo da pilha e coloca o resultado
// da soma no topo da pilha.
void RPN :: sum()
{
/*** COMPLETE AQUI: 11 ***/
}

// Metodo publico "sub".
// Se existirem pelo menos dois numeros no topo da pilha, faz a subtracao do
// numero do topo pelo segundo, retira os dois numeros do topo da pilha e
// coloca o resultado da subtracao no topo da pilha.
void RPN :: sub()
{
/*** COMPLETE AQUI: 12 ***/
}

// Metodo publico "mul".
// Se existirem pelo menos dois numeros no topo da pilha, faz a multiplicacao
// desses dois numeros, retira os dois numeros do topo da pilha e coloca o
// resultado da multiplicacao no topo da pilha.
void RPN :: mul()
{
/*** COMPLETE AQUI: 13 ***/
}

// Metodo publico "div".
// Se existirem pelo menos dois numeros no topo da pilha, faz a divisao do
// numero do topo pelo segundo, retira os dois numeros do topo da pilha e
// coloca o resultado da divisao no topo da pilha.
void RPN :: div()
{
/*** COMPLETE AQUI: 14 ***/
}

// Metodo publico "show".
// Imprime o indice e o valor de todos os elementos da pilha.
void RPN :: show()
{
  StackIterator si(this->stk);

  IOcalc io;
  int i=0;

  while(!si.end())
  {
  /*** COMPLETE AQUI: 15 ***/
  }
}
```

Arquivo “iocalc.h”

```
#ifndef _IOCALC_H
#define _IOCALC_H

#include <iostream>
#include <sstream>
#include <string>

using namespace std;

// Classe para entrada e saida (Input/Output) de dados para calculadora.
class IOcalc
{
 public:
   enum ErrorMsg { MSG_EMPTY, MSG_TWO_OP, MSG_UNKNOWN_OP };
 private:
   string  _line;
 public:
   void get() { cin >> _line; }
   bool isNumber(float *val);
   void getChar(char *c);
   void prompt()  { cout << "> "; }
   void message(ErrorMsg status);
   void stackItem(int i, float val);
};

#endif
```

Arquivo “iocalc.cpp”

```
#include "iocalc.h"

bool IOcalc :: isNumber(float *val)
{
   istringstream buf_num(_line, istringstream::in);

   if(buf_num >> *val)
      return true;

   return false;
}

void IOcalc :: getChar(char *c)
{
   istringstream buf_num(_line, istringstream::in);

   buf_num >> *c;
}

void IOcalc :: stackItem(int i, float val)
{
  cout << i << ": ";
  cout.precision( 2 );
  cout << fixed << val << endl;
}

void IOcalc :: message(ErrorMsg status)
{
  prompt();

  switch(status)
  {
    case MSG_EMPTY:
      cout << "Empty Stack !" << endl;
      break;
    case MSG_TWO_OP:
      cout << "Two Operands Needed !" << endl;
      break;
    case MSG_UNKNOWN_OP:
      cout << "Invalid Operator !" << endl;
      break;
  }
}
```