**Relatório**

**Cauã Ribas, Nilson Andrade**

Universidade do Vale do Itajaí - Univali

Escola do Mar, Ciência e Tecnologia

Ciência da Computação

(cauaribas, nilson.neto) @edu.univali.br

**Estruturas de Dados**

Avaliação 01 - Programação em linguagem de montagem

**Marcos Cesar Cardoso Carrard**

08/09/2023

**1. Introdução:**

Este relatório descreve a implementação de um programa usando a Linguagem de Alto Nível C/C++, este código implementa um programa que realiza a conversão de expressões matemáticas em notação infixa (a forma tradicional com operadores entre operandos) para notação polonesa reversa (RPN) e, em seguida, calcula o resultado da expressão em notação polonesa reversa. O programa também verifica a correspondência correta de parênteses na expressão de entrada.

**2. Programa:**

**2.1. Enunciado:**

O objetivo deste trabalho é a escrita de uma aplicação na qual o usuário pode digitar uma expressão matemática, esta aplicação deverá fazer a análise sintática dessa expressão e a sua resolução (se possível). Para isso, a expressão original (em notação normal – húngara) deve ser convertida para notação polonesa inversa e, uma vez nesse formato, resolvida. Quando dos processos de conversão para notação polonesa inversa e da resolução, os erros possíveis (listados a seguir) devem ser verificados e notificados ao usuário.

Entre os dois processos, os valores para as incógnitas devem ser solicitados ao usuário. Os erros a serem detectados são a paridade entre os parênteses, falta de operadores ou de operandos (se for realizada a resolução), nada além disso. Além disso, para o processo de conversão e solução, os dados devem utilizar de forma obrigatória as estruturas de dados tipo Pilha e Fila.

**2.2. Explicação Lógica do Código:**

**Bibliotecas e Macros:** O código inclui as bibliotecas iostream e math.h, além de arquivos de cabeçalho personalizados fila.hpp e pilha.hpp. Usa “using namespace std” para evitar a necessidade de prefixar objetos e funções padrão com “std::”.

**Constantes e Função “isOperator”:** Define uma constante TAM com valor 50. Implementa a função “isOperator” que verifica se um caractere é um operador (+, -, \*, /, ^).

**Função “calcularNPR”:** Esta função realiza o cálculo da notação polonesa reversa (RPN) com base em uma fila de entrada e uma pilha. Processa a fila de entrada, operando em dígitos, variáveis e operadores. Realiza as operações matemáticas quando encontra um operador, empilhando o resultado. Lida com resultados de uma maneira que considere números negativos e números com mais de um dígito. **Função “verificarParenteses”:** Verifica se os parênteses estão corretos, comparando dois caracteres: abertura (e fechamento).

**Função “ordenarOp”:** Essa função é usada para determinar a precedência de operadores. Ela recebe um operador b e verifica se sua precedência é maior ou igual à precedência do operador no topo da pilha. A precedência é determinada pelos valores atribuídos a cada operador.

**Função “converterNPR”:** Esta função realiza a conversão de uma expressão infixa para notação polonesa reversa (RPN). Usa uma fila de entrada original e uma pilha auxiliar para processar a conversão. Lida com parênteses, operadores e operandos de acordo com as regras da notação polonesa reversa. Exibe a notação polonesa reversa resultante e o resultado da expressão.

**Função “main”:** A função principal lê expressões matemáticas da entrada padrão até que o usuário digite "fim" para sair. Para cada expressão, ela converte e calcula seu resultado chamando “converterNPR”. A expressão é lida como uma string e depois dividida em caracteres individuais que são inseridos em uma fila. Após cada conversão e cálculo, o programa pausa e limpa a tela antes de continuar.

**2.3. Código em C/C++:**

#include <iostream>

#include <math.h>

#include "fila.hpp"

#include "pilha.hpp"

using namespace std;

#define TAM 50

// Função para verificar se um caractere é um operador

bool isOperator(char c) {

return (c == '+' || c == '-' || c== '^' || c == '\*' || c == '/');

}

//Função para calcular a notação polonesa

template <typename T>

bool calcularNPR(Fila <T> &f, Pilha <T> &p) {

char aux;

retirarFila(f, aux);

// if para insira na pilha diretamente caso for um digito

if (isdigit(aux)) {

inserirPilha(p, aux);

return true;

}

// else if caso for uma variável (letra), insira seu valor na pilha

else if (isalpha(aux)) {

char valor;

cout << "\nAdicione o valor de " << aux << ": ";

cin >> valor;

inserirPilha(p, valor);

return true;

}

//else if para calcular e transforma char em int

else if (isOperator(aux)) {

char auxOperating2;

char auxOperating1;

retirarPilha(p, auxOperating2);

retirarPilha(p, auxOperating1);

// Converte os operandos de char para int

auxOperating2 = auxOperating2 - '0';

auxOperating1 = auxOperating1 - '0';

int resultado;

// Realiza a operação de acordo com o operador

if (aux == '+')

resultado = auxOperating1 + auxOperating2;

else if (aux == '-')

resultado = auxOperating1 - auxOperating2;

else if (aux == '\*')

resultado = auxOperating1 \* auxOperating2;

else if (aux == '/')

resultado = auxOperating1 / auxOperating2;

else if (aux == '^')

resultado = pow(auxOperating1, auxOperating2);

// if se o resultado tiver mais de dois digitos

if (resultado >= 10) {

string resultadoStr = to\_string(resultado);

for (char c : resultadoStr)

inserirPilha(p, c);

}

// else if Se o resultado for negativo

else if (resultado < 0) {

string resultadoStr = to\_string(resultado);

for (char c : resultadoStr)

inserirPilha(p, c);

}

//se o resultado tiver apenas um digito

else {

char converter = resultado + '0';

inserirPilha(p, converter);

}

return true;

}

return false;

}

//Função bool para verificar se os parenteses estão corretos.

bool verificarParenteses(char a, char b){

if(a == '(' && b == ')') return true;

return false;

}

//Função para ordenar os caracteres e verificar os operadores

template <typename T>

int ordenarOp(T b, Pilha<T> p) {

char a;

retirarPilha(p, a);

int operadorA = 0;

int operadorB = 0;

// Determina o valor do operador A

if (a == '^')

operadorA = 3;

else if (a == '\*' || a == '/')

operadorA = 2;

else if (a == '+' || a == '-')

operadorA = 1;

// Determina o valor do operador B (o operador atual)

if (b == '^')

operadorB = 3;

else if (b == '\*' || b == '/')

operadorB = 2;

else if (b == '+' || b == '-')

operadorB = 1;

// Retorna true se o operador A for maior ou igual o operador B

return operadorA >= operadorB;

}

//Função para converção de equação para anotação polonesa.

template <typename T>

T converterNPR(Fila <T> &f){

Pilha <T> p;

Fila <T> f\_auxiliar;

char aux, aux\_2;

inicializarPilha(p);

inicializarFila(f\_auxiliar);

// Loop para percorrer a fila de entrada

while(retirarFila(f, aux)){

// Se o caractere atual for '(', insira-o na pilha

if(aux == '(')

inserirPilha(p, aux);

else if(aux == ')'){

//if para Verificar se há um parêntese aberto correspondente na pilha

if(pilhaVazia(p)){

cout << "Ha mais parenteses fechados do que abertos \n";

return 0;

}

else{

//while para transferir elementos da pilha para a fila auxiliar até encontrar '('

while(!pilhaVazia(p)){

retirarPilha(p, aux\_2);

inserirPilha(p, aux\_2);

//if para encontrar '(' e retirar da pilha e sai do loop.

if(aux\_2 == '('){

if(!verificarParenteses(aux\_2, aux)){

cout << "Parenteses nao combinam. \n";

return 0;

}

else{

retirarPilha(p, aux\_2);

break;

}

}

retirarPilha(p, aux\_2);

inserirFila(f\_auxiliar, aux\_2);

}

}

}

else if(aux == '^' || aux == '\*' || aux == '/' || aux == '+' || aux == '-'){

// Loop Enquanto a pilha não estiver vazia e o operador for maior ou igual ao topo da pilha

while(!pilhaVazia(p) && ordenarOp(aux, p)){

char aux\_3;

retirarPilha(p, aux\_3);

inserirFila(f\_auxiliar, aux\_3);

}

inserirPilha(p, aux);

}// else for um operando, insira na fila auxiliar

else{

inserirFila(f\_auxiliar, aux);

}

}

// while para percorrer toda a fila e tranferir os operadores da pilha para a fila auxiliar

while(!pilhaVazia(p)){

char topo;

retirarPilha(p, topo);

if(topo != '(')

inserirFila(f\_auxiliar, topo);

else{

cout << "Ha mais parenteses abertos do que fechados. \n";

return 0;

}

}

// Exibe a notação polonesa reversa resultante

cout << endl << "Notacao Polonesa: ";

mostrarFila(f\_auxiliar);

cout << endl;

// Inicialização de uma pilha para calcular o resultado

Pilha <char> resultadoFinal;

inicializarPilha(resultadoFinal);

// Realiza o cálculo com a notação polonesa reversa

while (!filaVazia(f\_auxiliar)){

calcularNPR(f\_auxiliar, resultadoFinal);

}

// Exibe o resultado final

cout << "\nResultado final: \n";

mostrarPilha(resultadoFinal);

return 1;

}

int main(){

string equacao;

string opcao;

Fila <char> f;

do{

inicializarFila(f);

cout<<"Digite a expressao matematica ou (fim) para sair: \n";

cout<<">>";getline(cin, equacao);cout<<"\n";

opcao = equacao;

if(opcao == "fim" || opcao == "FIM")break;

//For para separar a veriavel escrita em string para char +

for( int i=0; i < equacao.length(); i++ ){

inserirFila(f, equacao[i]);

}

cout <<"Fila Original: ";

mostrarFila(f);

cout << endl;

converterNPR(f);

system("pause");

system("cls");

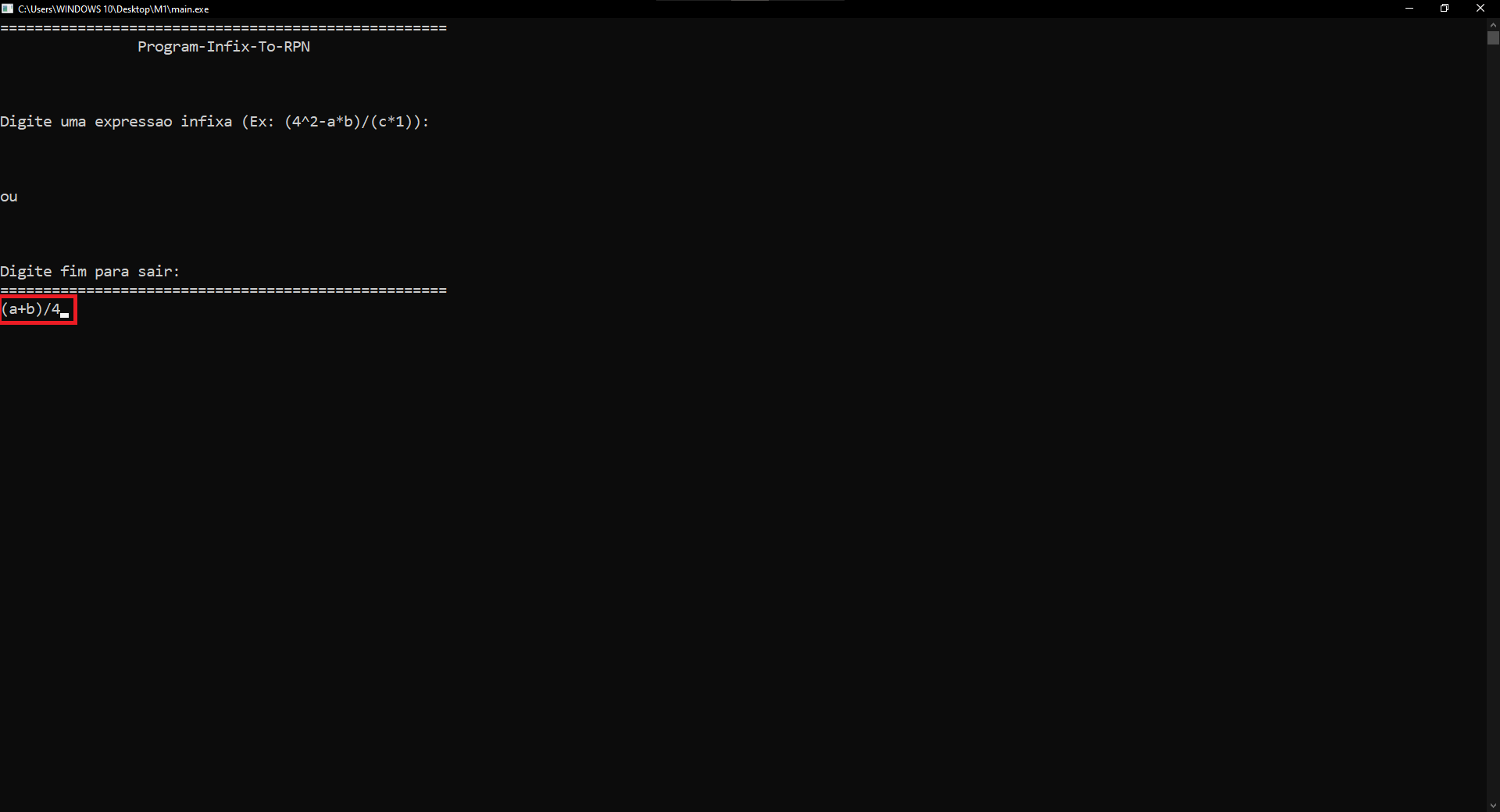
}while(opcao != "fim" && opcao != "FIM");

return 0;

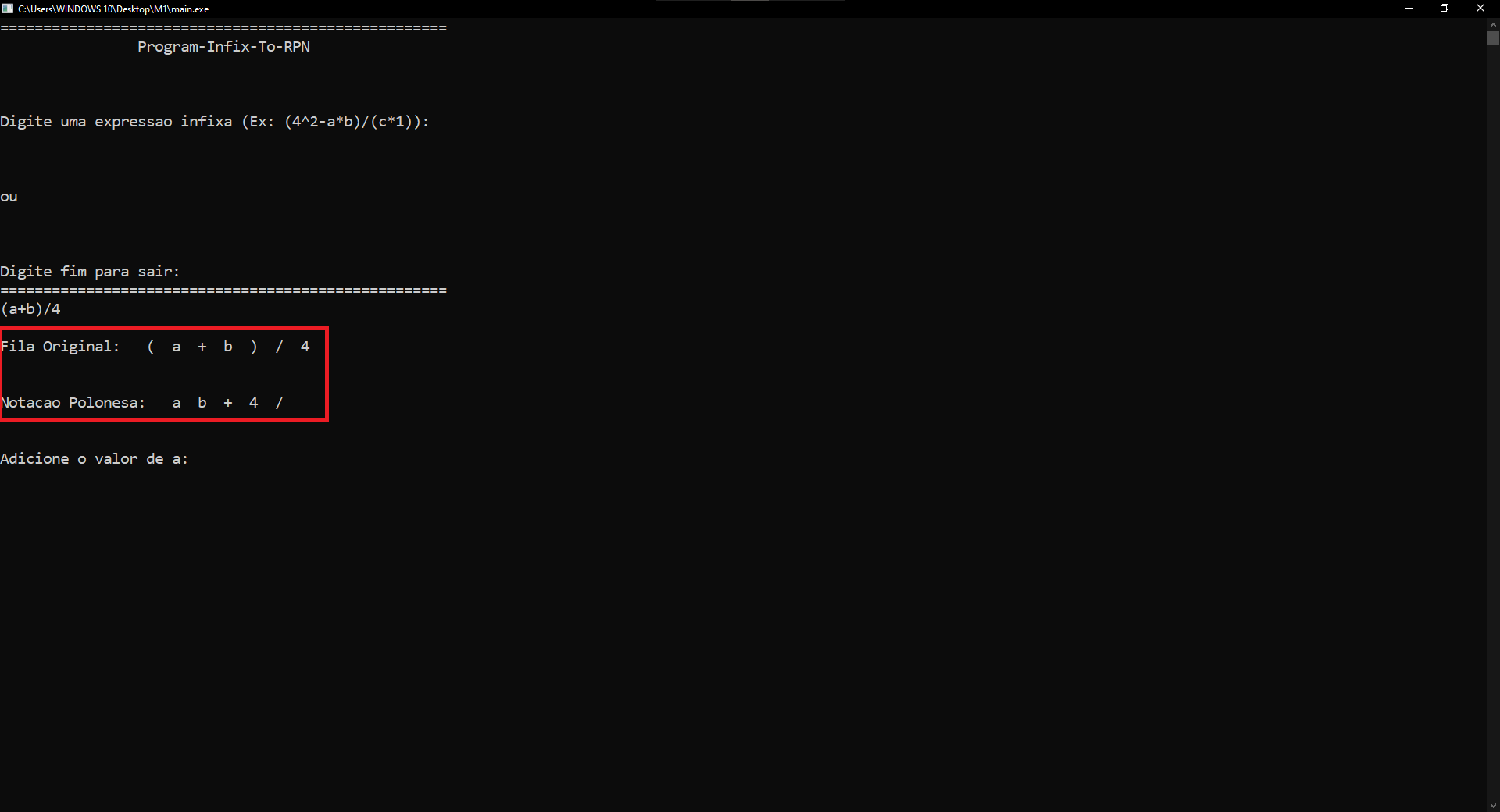
}

**3. Execução do Código:**

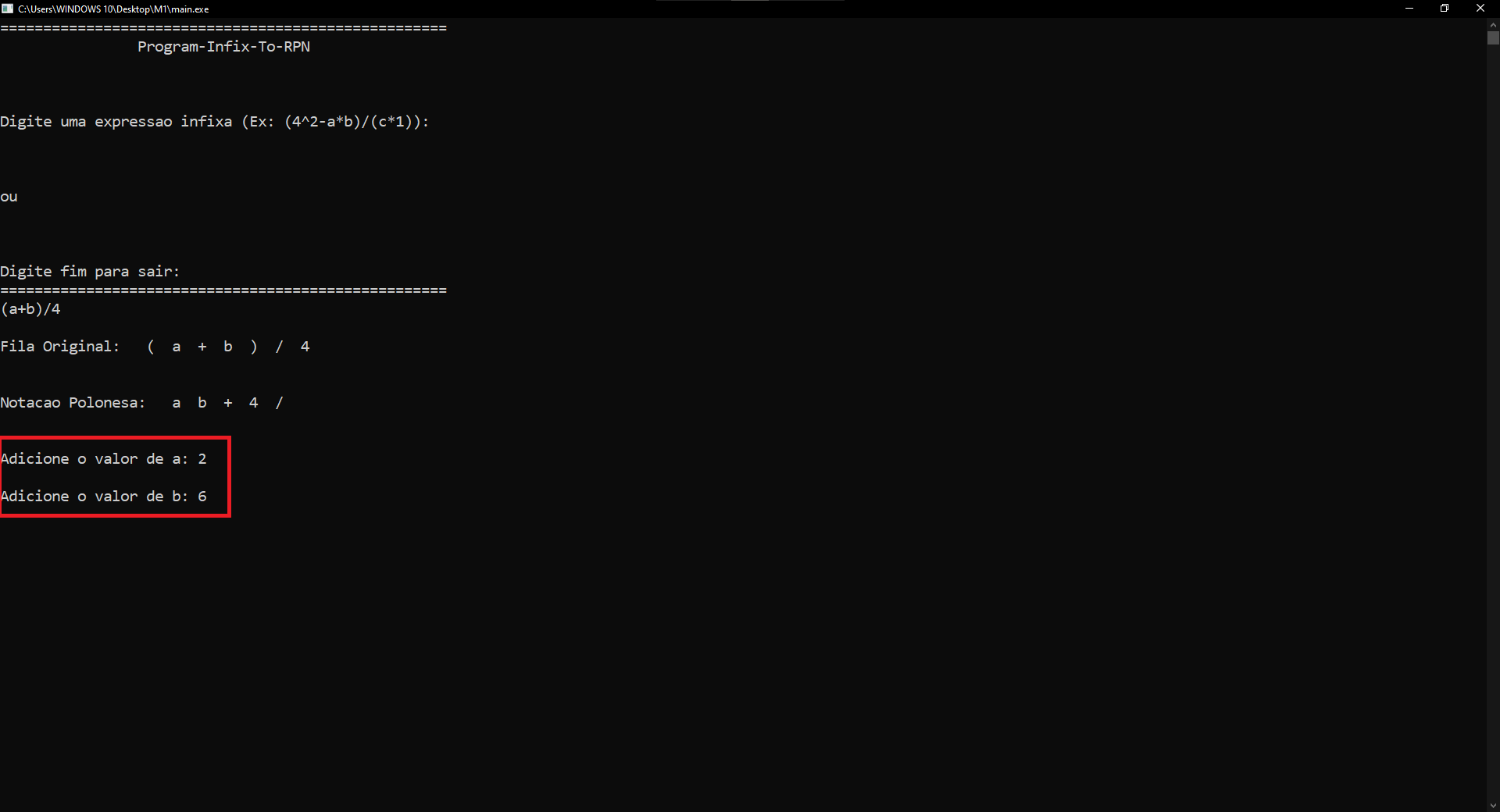
Após executar o programa, o usuário deve informar uma expressão matemática em notação infixa (Ex: 5+((1+2)\*4)-3), ou ((a\*b)-(c\*d))/(e\*f)), após o usuário informar a expressão, o programa mostra na tela a notação infixa convertida para notação polonesa reversa, após a conversão o programa irá calcular o resultado da expressão, se o usuário digitou uma expressão com variáveis, o programa pedirá o valor das variáveis, e em seguida mostrará na tela o resultado final da expressão.



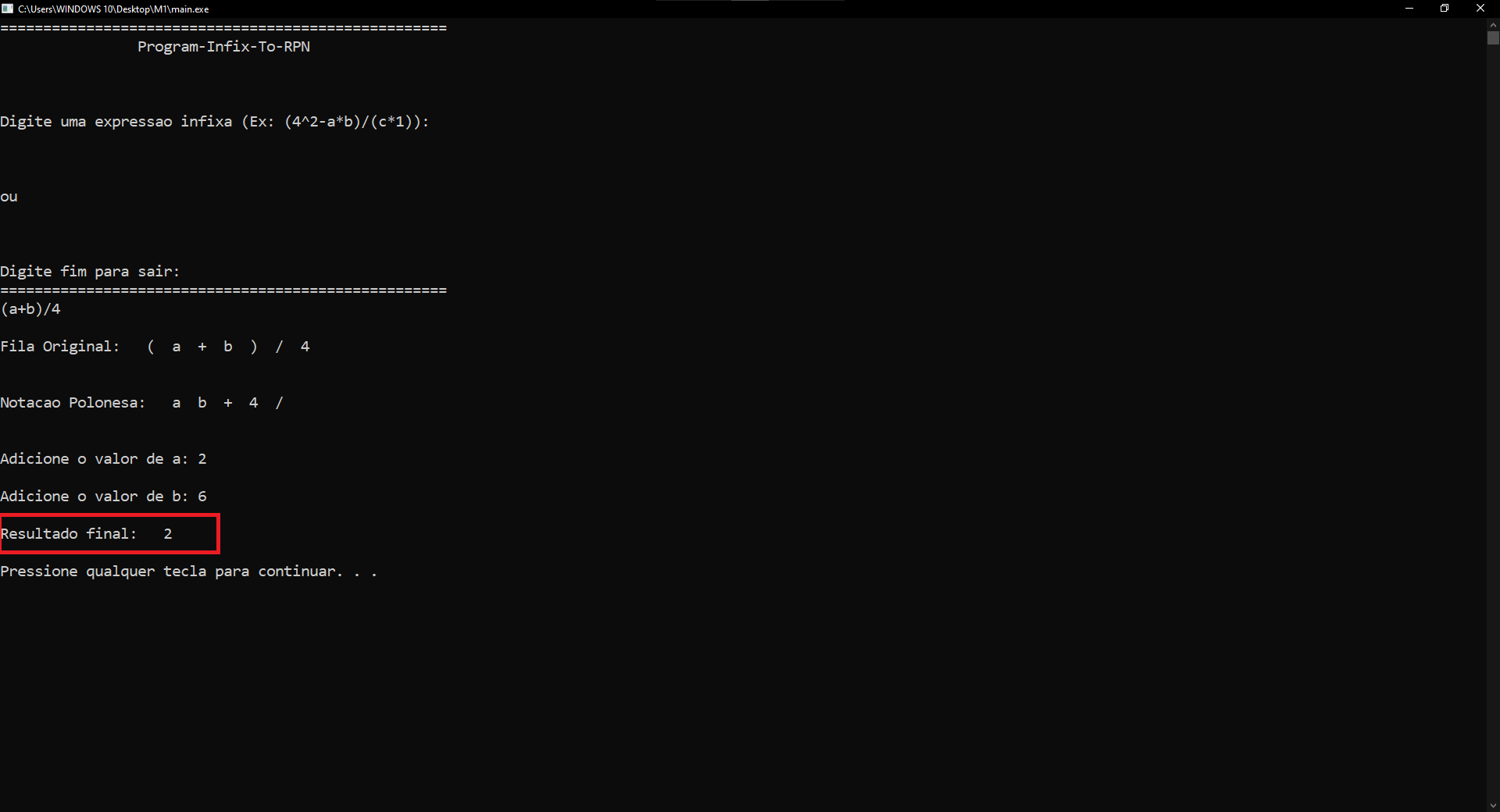
Expressão pós conversão.



Caso a expressão contenha variáveis, o programa solicita seus valores.



Aqui o programa apresenta o resultado final da expressão.



**4. Conclusão:**

Portanto, compreendemos que o programa em questão, que utiliza pilhas e filas é extremamente eficaz e performático no que lhe é proposto, as lógicas são básicas e de fácil entendimento, possuem uma variedade de implementações em programas reais, como uma calculadora financeira.

Graças às estruturas como pilhas e filas, podemos desenvolver programas mais complexos limitando as ações para somente aquilo que queremos permitir na estrutura.

**5. Referências:**

Wikipedia, Notação Polonesa Reversa, disponível em:

https://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o\_polonesa\_inversa

Panda Ime Usp, Notação Polonesa, disponível em:

https://panda.ime.usp.br/algoritmos/static/eps/ep6/parted/parted-polonesa.html