Universidade de Brasília (Unb)
Departamento da Ciência da Computação (CiC)

Estrutura de Dados

TRABALHO 1

Projeto

Avaliação de Expressões Aritméticas e Calculadora

Aluno
Luis Vinicius Capelletto

27 de Maio de 2018

1. Descrição da Notação Polonesa Reversa (NPR)

Notação polonesa reversa (RPN), também conhecida como notação pós-fixada é uma notação matemática na qual os operadores vem após seus operandos, ao contrário da notação polonesa (PN) ou forma prefixa ,de onde derivou, na qual os operadores precedem seus operandos. A descrição “polonês” refere-se à nacionalidade do lógico Jan Łukasiewicz, que inventou a notação polonesa em 1924.

O esquema polonês reverso foi proposto para reduzir o acesso à memória do computador e utilizar a pilha para avaliar expressões. Os algoritmos e a notação para este esquema foram estendidos pelo filósofo australiano e cientista da computação Charles L. Hamblin em meados da década de 1950.

Na ciência da computação, a notação polonesa reversa é usada em linguagens de programação orientadas a pilha, como Forth e PostScript. No nosso caso estaremos utilizando uma estrutura do tipo pilha na linguagem de programação C para sua implementação.

Exemplo:

- **Tradicional (infixa): A * (B -C) / D**
- **Polonesa Reversa (posfixa) : A B C - * D /**

2. Arquitetura e funcionalidades do sistema desenvolvido:

1. MODO CALCULADORA:

- ARQUIVOS

Calculadora.h: Arquivo header contendo as diretivas usadas, as estruturas utilizadas pela implementação da pilha de dados tipo float e todos os protótipos de funções utilizadas no modo calculadora encontrados em **Pilha_func.c / Valida_func.c / Suporte_func.c;**

Menu.c: Arquivo principal (main) de todo código contendo todas funcionalidades do Menu e seus 2 modos:

- Modo Resolução de Expressões (1);
- Modo Calculadora (2);

Valida_func.c: Arquivo que contém todas funções utilizadas com o fim de validar ou verificar dados;

- Funções em Valida_func.c:

-bool valida_str (char c): Retorna true apenas se char de entrada estiver entre os caracteres listados;
-bool tem_char (char c): Retorna true apenas se char de entrada for um entre os listados;
-bool tem_operador (char c): Retorna true apenas se char de entrada for um entre os listados (+ - * / ! c);
-int check (char* str): Função que verifica diversos casos inválidos de entrada utilizando as funções booleanas mencionadas e retorna true caso entrada seja válida e false caso contrário;

Pilha_func.c: Arquivo que contém todas funções relacionadas às estruturas usadas pela pilha de dados tipo float, utilizadas tanto para o modo calculadora quanto para a resolução de expressão (Algoritmo_3.c);

- Funções em Pilha_func.c:

. Alocação:

-t_pilha* aloca_pilha (): Aloca e retorna a estrutura do tipo ponteiro t_pilha com os campos primeiro e ultimo Nulos, e quantidade 0;
-t_no* aloca_no (float dado): Aloca e retorna estrutura do tipo ponteiro t_no atribuindo a entrada tipo float ao campo dado de t_no e Nulo para o ponteiro de t_no próximo;

. Funcionalidades:

-void imprime_pilha (t_pilha* pilha): Imprime conteúdo da pilha conforme indicado na especificação do projeto;
-int esta_vazia (t_pilha* pilha): Retorna true se estiver vazia e false caso contrário;
-void apaga_pilha (t_pilha* pilha): Apaga a pilha liberando sistematicamente, partindo do primeiro nó até o último e em seguida libera a estrutura pilha;

. Operações:

-int empilha (float dado, t_pilha* pilha): Insere no topo da pilha o valor float de entrada e retorna true caso bem sucedida e false caso contrário;

-int verifica_pilha (t_pilha* pilha): Retorna true se quantidade de nós da pilha for igual ou maior que 2 e false caso contrário;
-float desempilha_e_calcula_1 (char op, t_pilha* pilha): Desempilha e faz a operação dada entre os valores tipo float dos 2 nós do topo da pilha e retorna seu resultado;
-float desempilha_e_calcula_2 (char op, t_pilha* pilha): Função semelhante a utilizada no modo calculadora, que desempilha os 2 valores do topo da pilha de dados tipo float e realiza a operação dada conforme o char de operação recebido pela função, a diferença entre essa função e sua semelhante é que a ordem dos operandos muda, sendo o valor anterior ao do topo o primeiro operando na operação a ser realizada, ao contrário do utilizado no modo calculadora, o valor float retornado é o resultado da operação feita;
-int remove_ultimo (t_pilha* pilha): Remove o nó do topo da pilha (último) e retorna true caso bem sucedida e false caso contrário;

Suporte_func.c: Contem as funções responsáveis pela operação especial cópia de elemento (c) , pela liberação de caracteres remanescentes no “Buffer” e a função que transforma string em um valor do tipo float;

- Funções em Suporte_func.c:

-int copia_elemento (t_pilha* pilha): Desempilha o valor do nó do topo da pilha (N) , o valor anterior a esse (k) e empilha o número de vezes o valor “k” igual ao valor “N”, retorna true caso bem sucedida e false caso contrário;
-void limpabuffer (): Libera os caracteres remanescentes na área de armazenamento temporário;
-float str_em_float (char* str): transforma string de entrada recebida e retorna seu valor em float;

2. MODO RESOLUÇÃO DE EXPRESSÃO:

Res_exp.h: Arquivo header contendo as diretivas, as estruturas usadass pela implementação da pilha de dados tipo char e todos protótipos de funções utilizados pelo Modo de Resolução de Expressão;

Pilha_c_func.c: Contém todas funções relacionadas com as estruturas da pilha de dados tipo char e suas funcionalidades/operações;

- Funções em Pilha_c_func.c:

. Alocação:

-t_pilha_c* aloca_pilha_c (): Aloca e retorna a estrutura do tipo ponteiro t_pilha_c com os campos primeiro e ultimo Nulos, e quantidade 0;
-t_no_c* aloca_no_c (char dado): Aloca e retorna estrutura do tipo ponteiro t_no_c atribuindo a entrada tipo char ao campo dado de t_no_c e Nulo para o ponteiro de t_no_c próximo;

. Funcionalidades:

-void imprime_pilha_c (t_pilha_c* pilha_c): Imprime os dados tipo char sistematicamente do primeiro ao ultimo dado (topo) da pilha de chars;
-int esta_vazia (t_pilha_c* pilha_c): Retorna true caso pilha esteja vazia e false caso contrário;
-void apaga_pilha_c (t_pilha_c* pilha_c): Apaga a pilha_c liberando sistematicamente, partindo do primeiro nó até o último e em seguida libera a estrutura pilha_c;

.Operações:

-int empilha_c (char dado, t_pilha_c* pilha_c): Insere no topo da pilha o dado tipo char de entrada e retorna true caso bem sucedida e false caso contrário;
-char desempilha_c (char dado, t_pilha_c* pilha_c): Desempilha o dado tipo char do topo da pilha e retorna-o;

Valida_exp_func.c: Arquivo que contém todas funções utilizadas com o fim de validar ou verificar dados;

- Funções em Valida_exp_func.c:

-bool valida_str_1 (char c): Retorna true apenas se char de entrada conter apenas os caracteres listados e false caso contrário;
-bool tem_numero (char c): Retorna true apenas se char de entrada conter apenas os caracteres listados e false caso contrário;
-bool tem_escopo (char c): Retorna true apenas se char de entrada conter apenas os caracteres listados e false caso contrário;
-bool tem_op (char c): Retorna true apenas se char de entrada conter apenas os caracteres listados e false caso contrário;
-int check_exp (char* str): Verifica todos os caracteres da string de entrada e caso algum deles seja inválido retorna false, caso contrário retorna true;

Algoritmo_1.c: Contém função com o algoritmo 1, conforme especificado no trabalho, que verifica a validação dos escopos da expressão de entrada;

- Função:

-int valida_escopo (char* str, t_pilha_c* pilha_c): Recebe string de entrada contendo a expressão dada pelo usuário e verifica cada caractere da string em busca de separadores (() , [] , {}), validando-os utilizando a pilha conforme especificado na descrição do trabalho para o algortimo 1, retorna true caso expressão seja válida e false caso contrário;

Algoritmo_2.c: Arquivo que contém as funções de precedencia e algoritmo 2, especificado no trabalho

- Funções

-int precedencia (char c): Recebe dado tipo char de entrada e conforme seu tipo retorna o valor 2, 1 ou 0 de acordo com sua precedência de operação;
-void transforma (char* exp_infixa, char* exp_posfixa): Recebe as strings alocadas infixa e posfixa, realizando conforme especificado no algoritmo 2 do trabalho, a transformação da expressão infixa de entrada dada pelo usuário para a forma posfixa, inserindo espaços entres os operandos e operadores, a string na forma posfixa é inserida na string posfixa recebida de entrada pela função tipo void;

Algoritmo_3.c: Arquivo que contém as funções responsáveis por realizar a resolução da expressão posfixa, conforme especificado no algoritmo 3 do trabalho, **único** arquivo do modo Resolução de Expressão que utiliza as estruturas da pilha de dados tipo float do modo calculadora (2), localizadas no arquivo **Calculadora.h**, por isso o include do header neste código;

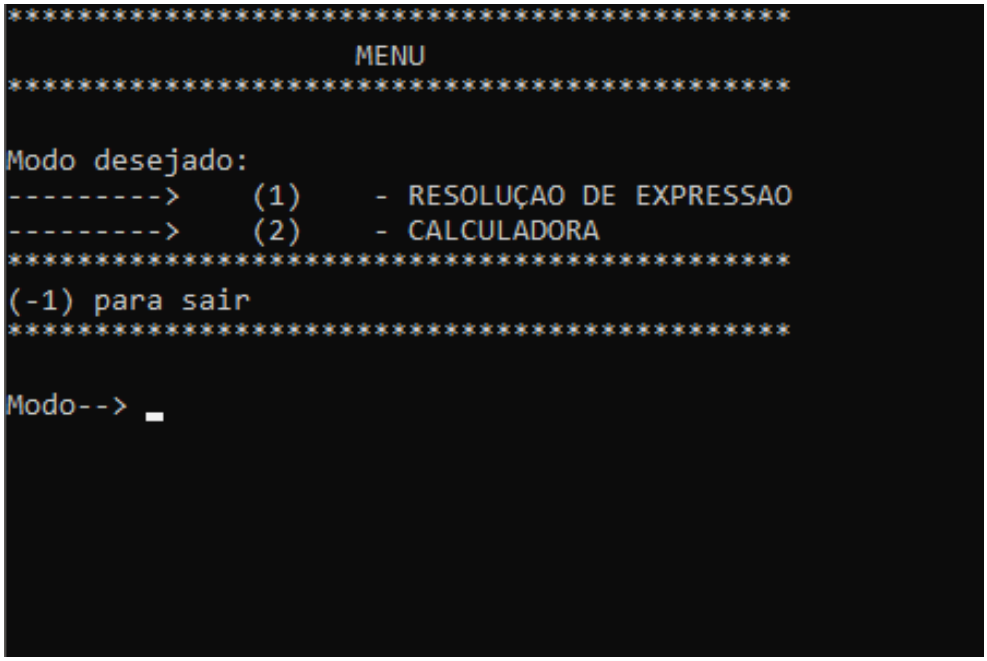
- Função:

-float resolve_posfixa (char* str): Recebe string contendo a expressão posfixa e conforme especificado na descrição do algoritmo 3, realiza sua resolução ao empilhar os operandos conforme os encontra e desempilhando ao realizar as devidas operações ao encontrar os operadores, verificando sempre se a pilha contém o número de operandos necessários e empilhando seu resultado novamente na pilha, sendo assim sobrando apenas 1 dado tipo float contendo o valor do resultado final das operações, o qual é retornado pela função;

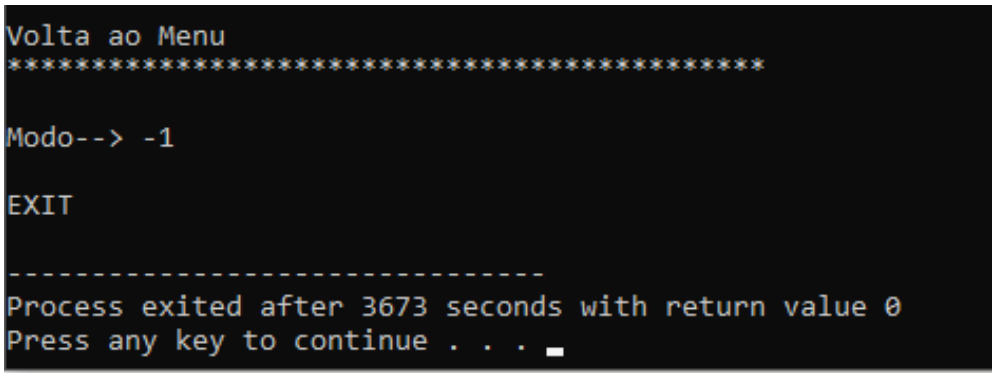
3. Screenshots das funcionalidades implementadas

MENU:

Apresentação do Menu;

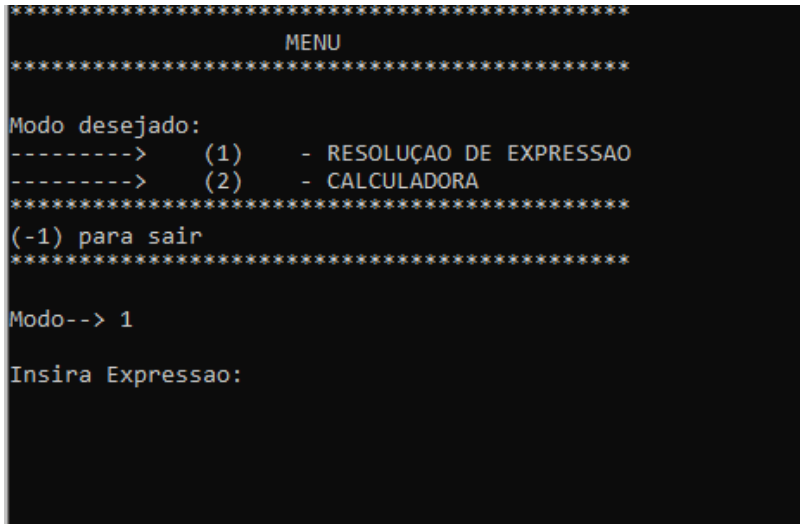


Sair do programa (-1);

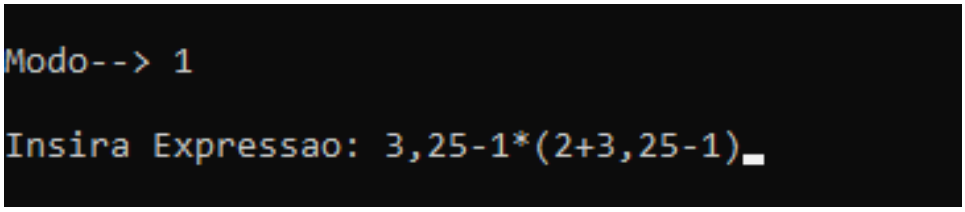


MODO RESOLUÇÃO DE EXPRESSÃO (1):

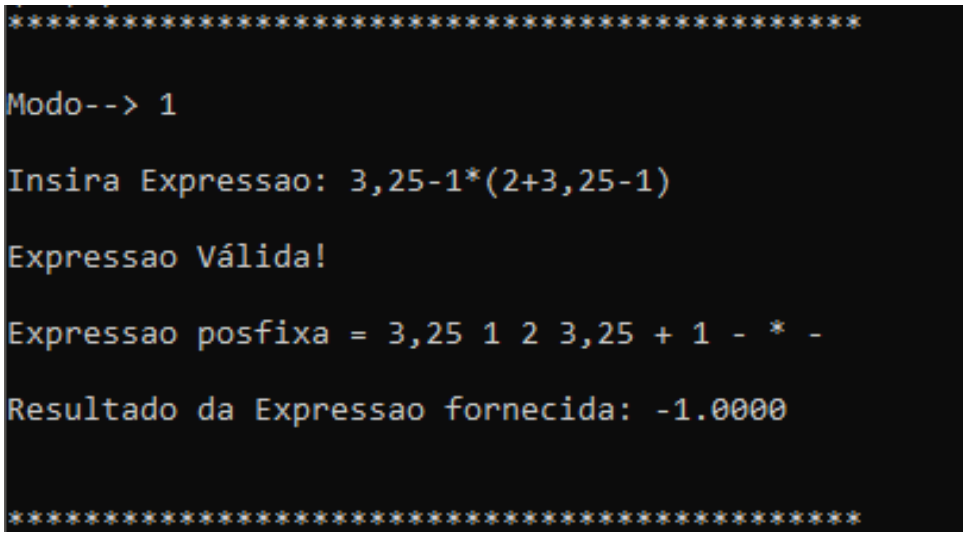
Entra no modo Resolução de expressão (1);



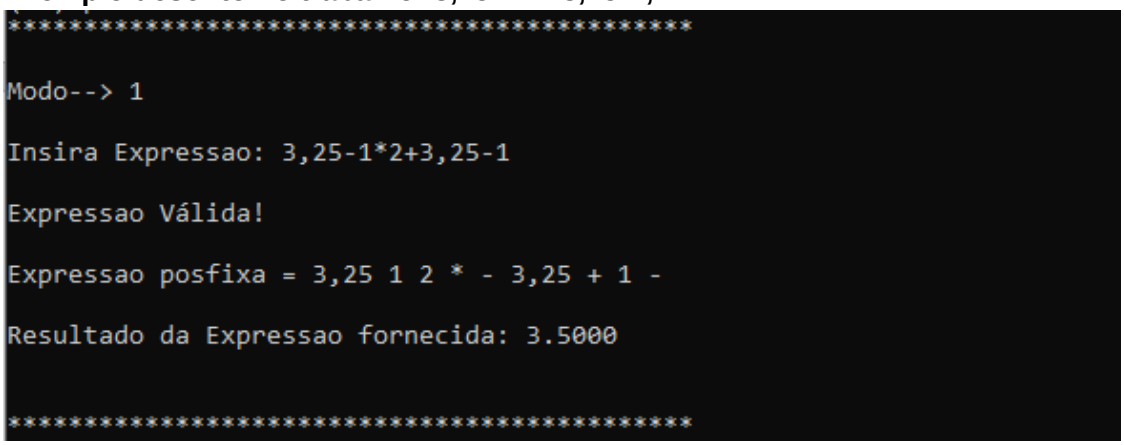
Exemplo de entrada de expressão infixa válida;



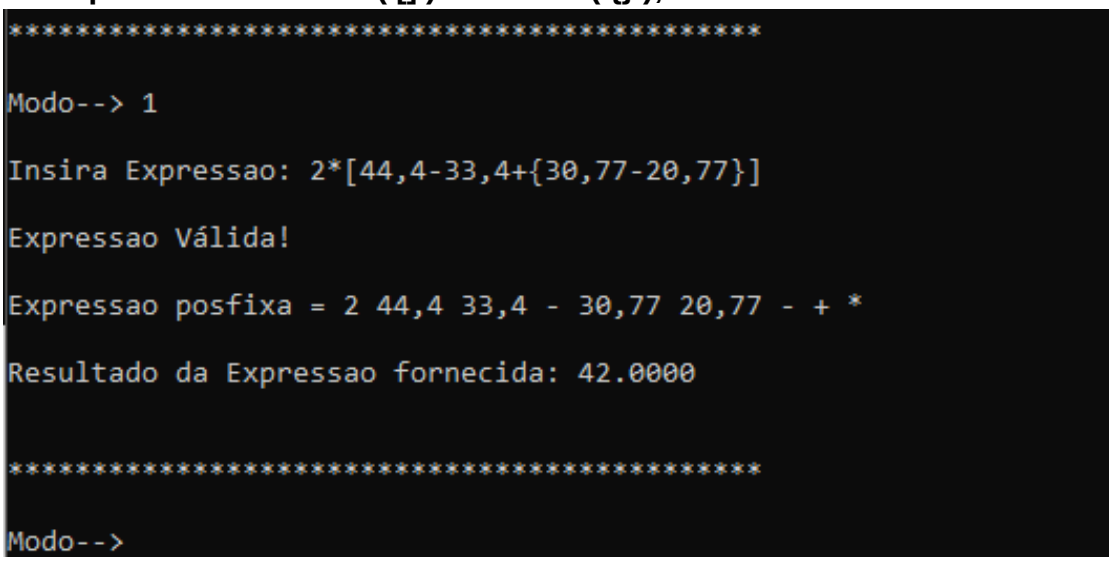
Transformação e resolução da expressão inserida;



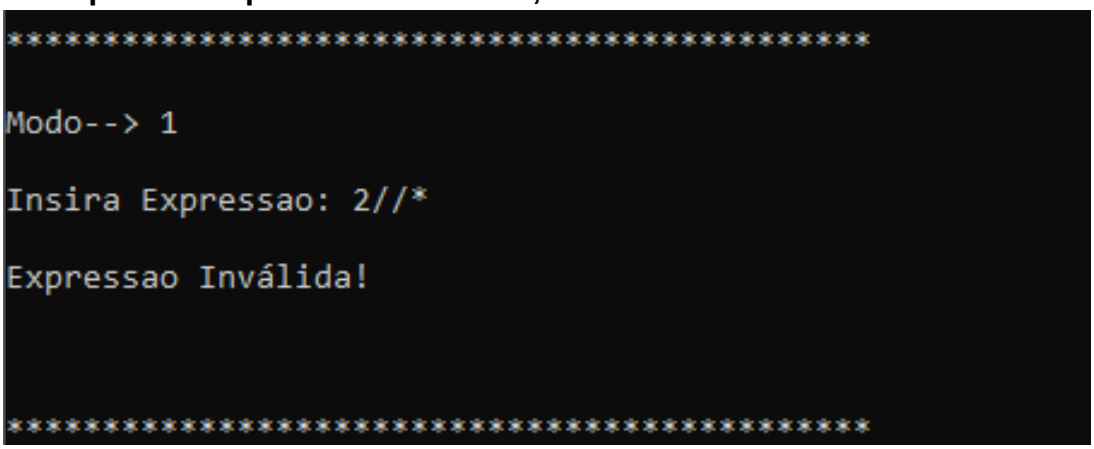
Exemplo descrito no trabalho: 3,25-1*2+3,25-1;



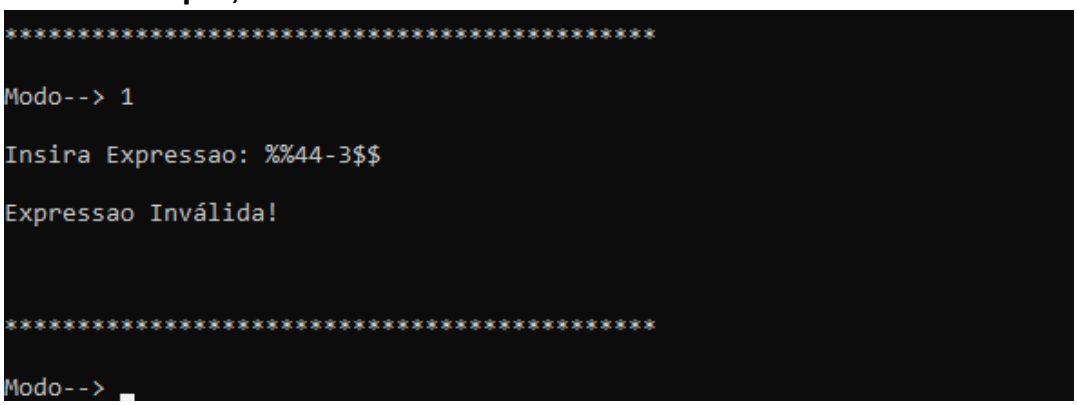
Exemplo com colchetes ([]) e chaves ({});



Exemplos de expressões inválidas;



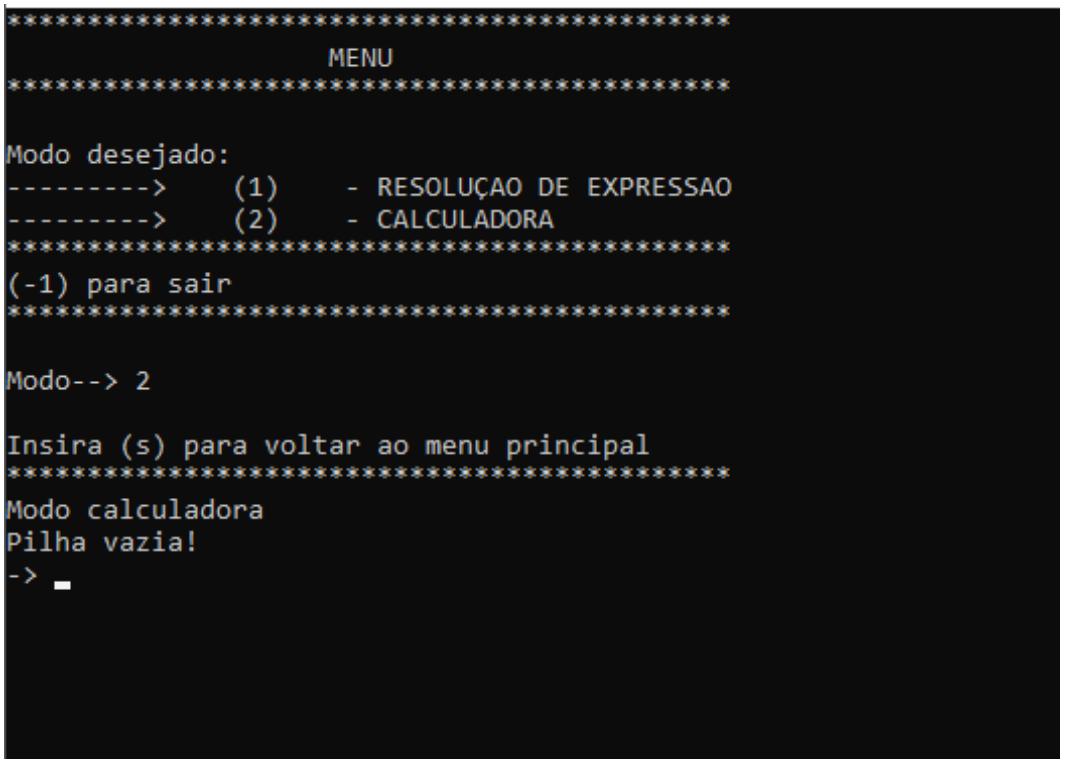
Outro exemplo;



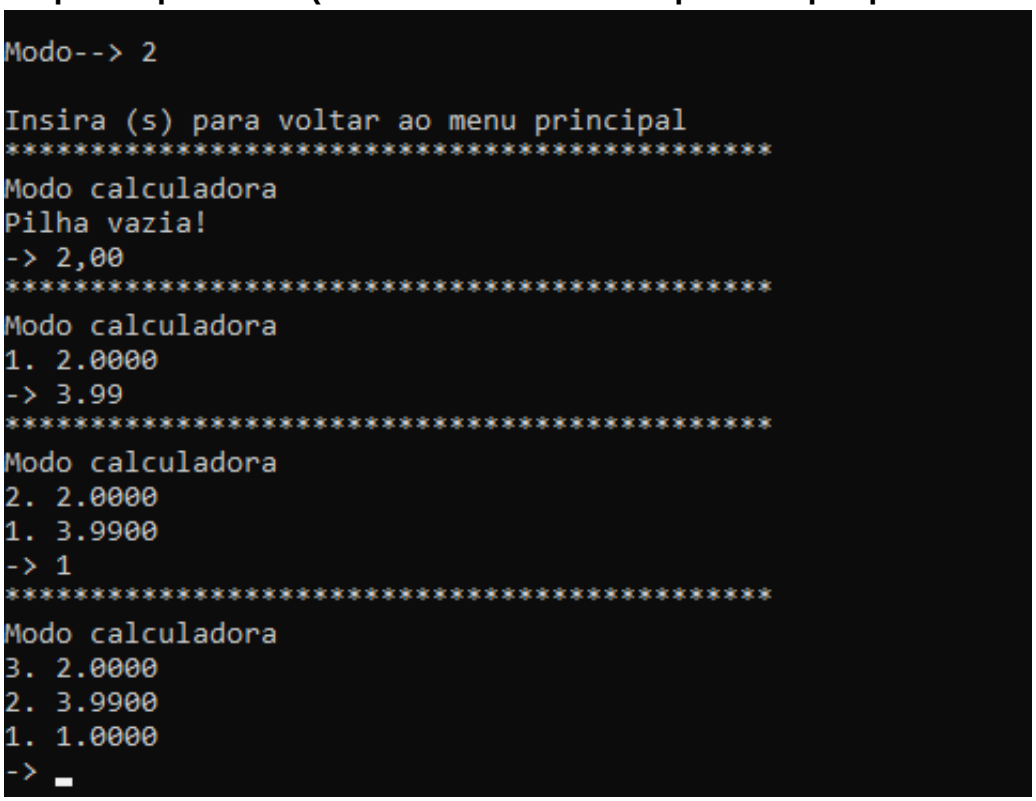
Obs: Após entrada de expressão válida ou inválida, o programa sempre retorna ao Menu, esperando entrada do modo desejado;

MODO CALCULADORA (2):

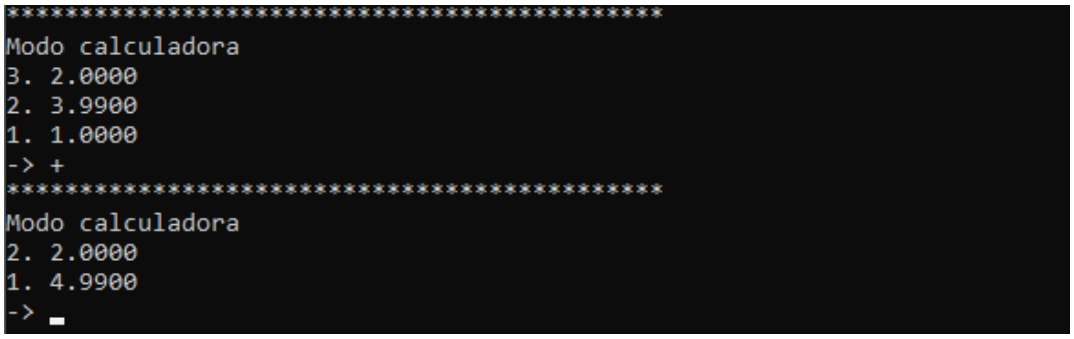
Entra no modo Calculadora (2);



Empilha operandos (inteiros ou decimais separados por ponto ou vírgula);

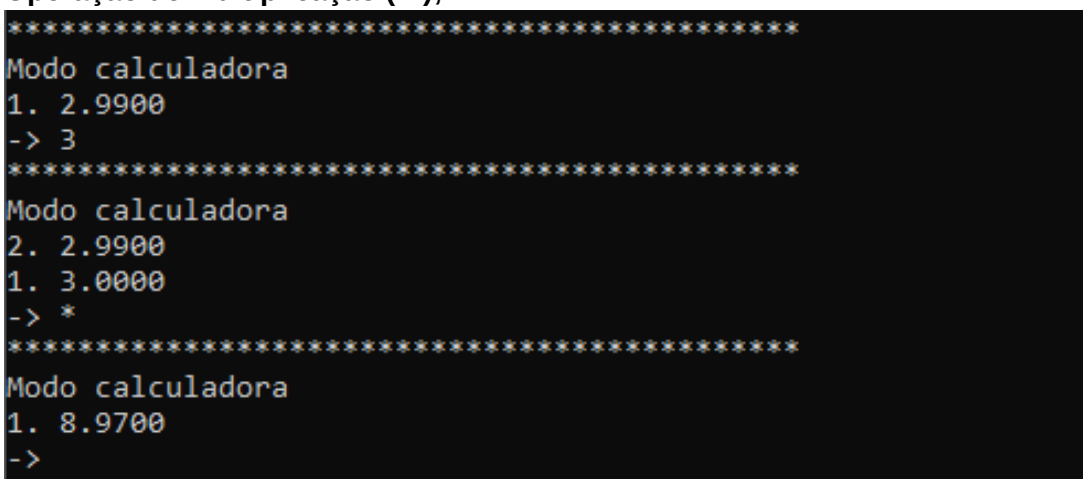


Operação de soma (+);



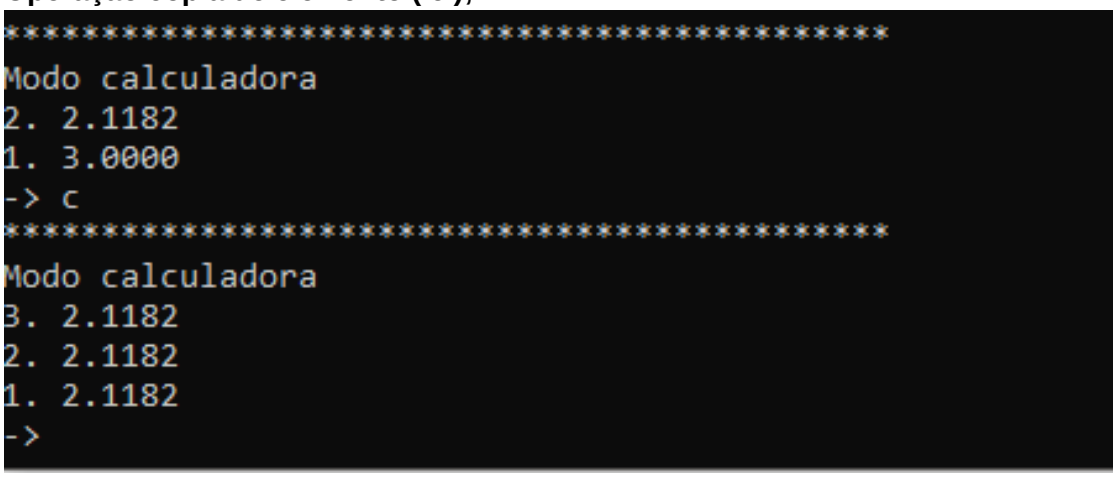
Operação de subtração (-);

Operação de multiplicação (*);



Operação de divisão (/);

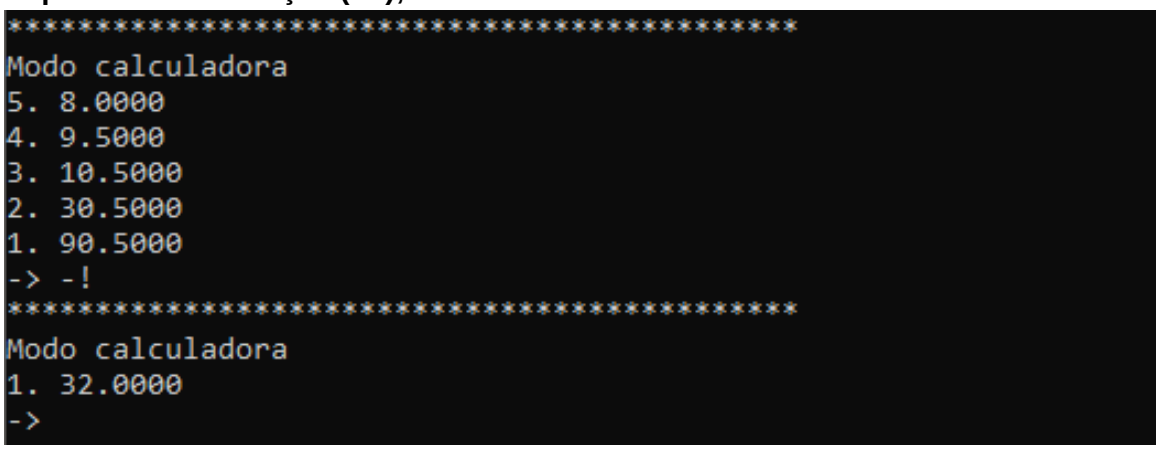
Operação cópia de elemento (c);



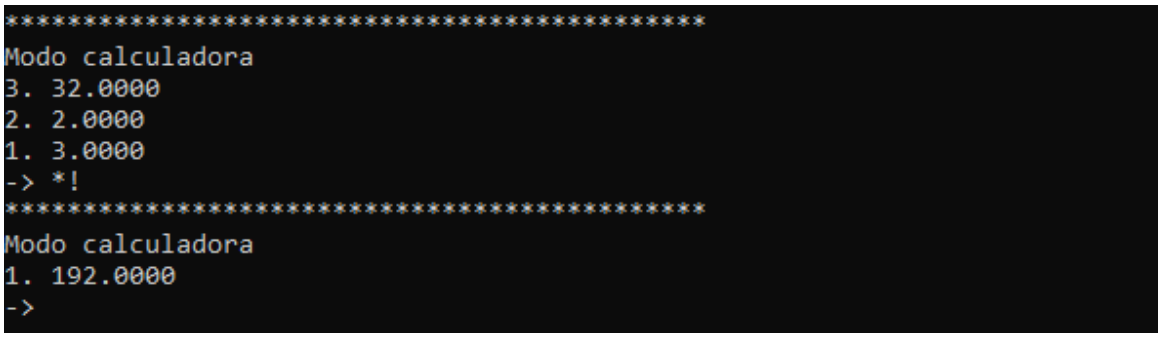
Operação repetidor de operação (+!, -!, *!, /!):

Repetidor de soma (+!);

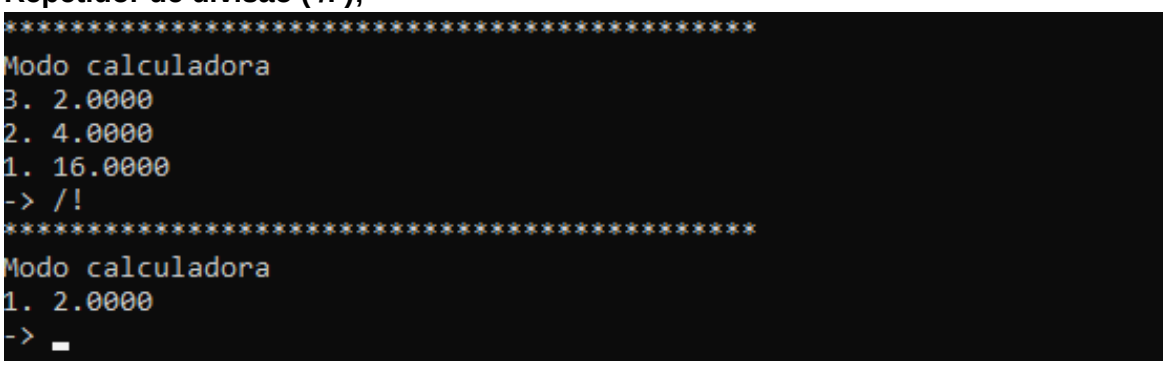
Repetidor de subtração (-!);



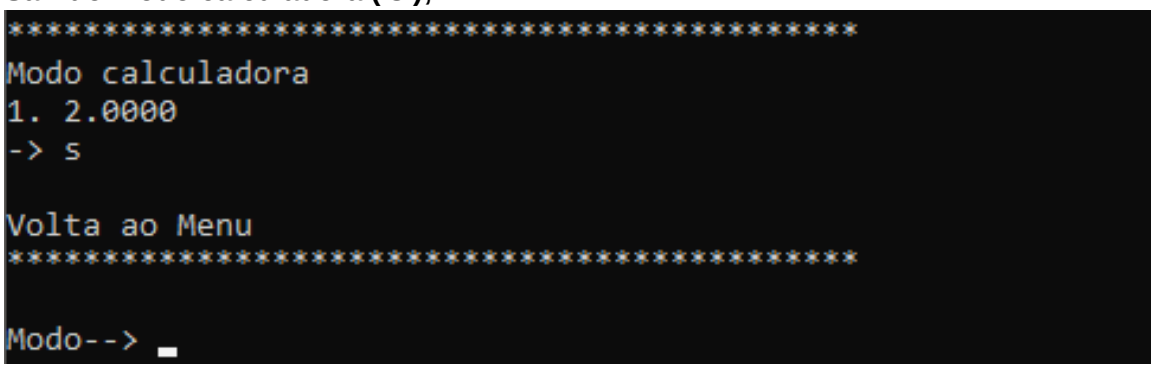
Repetidor de multiplicação (*!);



Repetidor de divisão (/!);



Sair do modo calculadora (s);



FIM.