Ex: 001

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "pilha.h"

char \*posfixa(char \*e) {

   static char s[256];

   int j = 0;

   Pilha P = pilha(256);

   for(int i=0; e[i]; i++)

      if( isdigit(e[i]) ) s[j++] = e[i];

      else if( strchr("+\*-/",e[i]) ) empilha(e[i],P);

      else if( e[i]==')' ) s[j++] = desempilha(P);

   s[j] = '\0';

   destroip(&P);

   return s;

}

int main(void) {

   char e[513];

   printf("Infixa? ");

   gets(e);

   printf("Posfixa: %s\n", posfixa(e));

   return 0;

}

/\*

EX001:   a) 23\*84/+

         b) 95-1-

         c) 234\*5-

         d) 34+86-\*2/

\*/

Ex: 002

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "pilha.h"

int prio(char o) {

   switch( o ) {

      case '(': return 0;

      case '+':

      case '-': return 1;

      case '\*':

      case '/': return 2;

   }

   return -1; // operador inválido!

}

char \*posfixa(char \*e) {

   static char s[256];

   int j = 0;

   Pilha P = pilha(256);

   for(int i=0; e[i]; i++)

      if( e[i]=='(' ) empilha('(',P);

      else if( isdigit(e[i]) ) s[j++] = e[i];

      else if( strchr("+-/\*",e[i]) ) {

         while( !vaziap(P) && prio(topo(P))>=prio(e[i]) )

            s[j++] = desempilha(P);

         empilha(e[i],P);

      }

      else if( e[i] == ')' ) {

         while( topo(P)!='(' )

           s[j++] = desempilha(P);

         desempilha(P);

      }

        while( !vaziap(P) )

            s[j++] = desempilha(P);

        s[j] = '\0';

        destroip(&P);

    return s;

}

int main(void) {

   char e[513];

   printf("Infixa? ");

   gets(e);

   printf("Posfixa: %s\n", posfixa(e));

   return 0;

}

/\*EX002: a) 23\*84/+

         b) 95-1-

         c) 234\*+5-

         d) 34+86-\*2/

\*/

Ex: 003

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "pilha.h"

int valor(char \*e) {

   Pilha P = pilha(256);

   for(int i=0; e[i]; i++)

      if( isdigit(e[i]) ) empilha(e[i]-'0',P);

      else {

         int y = desempilha(P);

         int x = desempilha(P);

         switch( e[i] ) {

            case '+': empilha(x+y,P); break;

            case '-': empilha(x-y,P); break;

            case '\*': empilha(x\*y,P); break;

            case '/': empilha(x/y,P); break;

         }

   }

   int z = desempilha(P);

   destroip(&P);

   return z;

}

int prio(char o) {

   switch( o ) {

      case '(': return 0;

      case '+':

      case '-': return 1;

      case '\*':

      case '/': return 2;

   }

   return -1; // operador inválido!

}

char \*posfixa(char \*e) {

   static char s[256];

   int j = 0;

   Pilha P = pilha(256);

   for(int i=0; e[i]; i++)

      if( e[i]=='(' ) empilha('(',P);

      else if( isdigit(e[i]) ) s[j++] = e[i];

      else if( strchr("+-/\*",e[i]) ) {

         while( !vaziap(P) && prio(topo(P))>=prio(e[i]) )

            s[j++] = desempilha(P);

         empilha(e[i],P);

      }

      else if( e[i] == ')' ) {

         while( topo(P)!='(' )

           s[j++] = desempilha(P);

         desempilha(P);

      }

        while( !vaziap(P) )

            s[j++] = desempilha(P);

        s[j] = '\0';

        destroip(&P);

    return s;

}

int main(void) {

   char e[513];

   printf("Infixa? ");

   gets(e);

   printf("Posfixa: %s\n", posfixa(e));

   printf("Resultado: %d", valor(posfixa(e)));

   return 0;

}



Ex:004

/\*

   Com base nos algoritmos descritos, crie um programa para ler uma expressão booleana comple-tamente parentesiada e exibir sua forma posfixa correspondente, bem como seu valor numérico.

   Considere que as expressões são compostas por:

   Operandos: letras maiúsculas F e V, com valores numéricos 0 e 1, respectivamente.

   Operadores: ! (não), & (e) e | (ou), da maior para a menor prioridade.

   Delimitadores: parênteses de abertura e fechamento.

   Por exemplo, para a expressão booleana infixa parentesiada "((!F)|(F&V))", o programa deve apresentar como saída a forma posfixa "F!FV&|" e o valor numérico 1.

   (0 -> 1) ou (0 & 1 -> 0) -> 1

\*/

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "pilha.h"

int Opou(char x, char y){

   if(x=='V')x=1; else x=0;

   if(y=='V')y=1; else y=0;

   if(x==1 || y==1 ){

      return 1;

   }

   else{

      return 0;

   }

   printf("%d",x);

   printf("%d\n",y);

}

int Ope(char x, char y){

   if(x=='V')x=1; else x=0;

   if(y=='V')y=1; else y=0;

   if(x==1 && y ==1){

      return 1;

   }

   else{

      return 0;

   }

   printf("%d",x);

   printf("%d\n",y);

}

int OpInv(char x){

   if(x=='V')x=0;

   if(x=='F')x=1;

   printf("%d",x);

   return x;

}

int valor(char \*e) {

   Pilha P = pilha(256);

   for(int i=0; e[i]; i++){

      if( e[i]=='V' || e[i]=='F' ) {

        //  printf("%c",e[i]);

         empilha(e[i],P);

      }

      else {

         if(e[i]=='!'){

            int x = desempilha(P);

            empilha(OpInv(x),P);

         }

         int y = desempilha(P);

         int x = desempilha(P);

         switch( e[i] ) {

            case '|': empilha(Opou(x,y),P); break;

            case '&': empilha(Ope(x,y),P); break;

         }

      }

   }

   int z = desempilha(P);

   destroip(&P);

   return z;

}

int prio(char o) {

   switch( o ) {

      case '|': return 0;

      case '&': return 1;

      case '!': return 2;

   }

   return -1; // operador inválido!

}

char \*posfixa(char \*e) {

   static char s[256];

   int j = 0;

   Pilha P = pilha(256);

   for(int i=0; e[i]; i++)

      if( e[i]=='(' ) empilha('(',P);

      else if( e[i]=='V' || e[i]=='F' ) s[j++] = e[i];

      else if( strchr("!&|",e[i]) ) {

         while( !vaziap(P) && prio(topo(P))>=prio(e[i]) )

            s[j++] = desempilha(P);

         empilha(e[i],P);

      }

      else if( e[i] == ')' ) {

         while( topo(P)!='(' )

           s[j++] = desempilha(P);

         desempilha(P);

      }

      while( !vaziap(P) )

         s[j++] = desempilha(P);

      s[j] = '\0';

      destroip(&P);

    return s;

}

int main(void) {

   char e[513];

   printf("Infixa? ");

   gets(e);

   printf("Posfixa: %s\n", posfixa(e));

   printf("Resultado: %d", valor(posfixa(e)));

   return 0;

}



Ex: 005

/\*

   Crie a função prefixa(e), que devolve a forma prefixa da expressão aritmética completa-mente parentesiada e. Em seguida, faça um programa para testar a função.

   Dicas:

   Percorra a expressão infixa da direita para a esquerda.

   Use strlen() para determinar o tamanho da expressão infixa.

   Use \_strrev() para inverter uma cadeia de caracteres

\*/

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "pilha.h"

char \*prefixa(char \*e) {

   static char s[256];

   int j = strlen(e);

   int i=0;

   // printf("%d", j);

   Pilha P = pilha(256);

                                 //((2\*3)+(8/4))

   for(j; j>0; j--){

      // printf("%c",e[j]);      //   ))

      if( isdigit(e[j]) ) {

         s[i++] = e[j];       //   s[4,8,/,3,2,\*,+]

      }

      else if( strchr("+\*-/",e[j]) ){

         empilha(e[j],P);         //P []

      }

      else if( e[j]=='(' ) {

         s[i++] = desempilha(P);

      }

   }

   s[i++] = desempilha(P);

   // s[j] = '\0';

   // destroip(&P);

   return s;

}

int main(void) {

   char e[513];

   printf("Infixa? ");

   gets(e);

   printf("Prefixa: %s\n", strrev(prefixa(e)));

   // printf("Resultado: %d\n", valor2(prefixa(e)));

   return 0;

}



Ex: 006

/\*

  Crie a função valpre(e), que devolve o valor da expressão aritmética prefixa e. Em seguida, faça um programa para testar a função.

\*/

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include "pilha.h"

int charInt(int x)

{

   switch (x)

   {

   case 48:

      return x = 0;

      break;

   case 49:

      return x = 1;

      break;

   case 50:

      return x = 2;

      break;

   case 51:

      return x = 3;

      break;

   case 52:

      return x = 4;

      break;

   case 53:

      return x = 5;

      break;

   case 54:

      return x = 6;

      break;

   case 55:

      return x = 7;

      break;

   case 56:

      return x = 8;

      break;

   case 57:

      return x = 9;

      break;

   }

}

int valpre(char \*e)

{

   // printf("\n%s\n", e);

   // char s[256];

   Pilha R = pilha(256);

   int j = strlen(e);

   for (int i = j - 1; i >= 0; i--)

   { // 8

      if (isdigit(e[i]))

      {

         empilha(e[i], R);                      //      R[4,8]

         // printf("Estou empilhando %c\n", e[i]); // R["4","8"]

      }

      else if (strchr("+\*-/", e[i]))

      {

         // printf("O operador eh: %c\n", e[i]);

         int a = desempilha(R);

         int b = desempilha(R);

         int x = charInt(a);

         int y = charInt(b);

         // printf("X : %d\n", x);

         // printf("Y : %d\n", y);

         int c=0;

         switch (e[i])

         {

         case '+':

            c = y+x;

            empilha(c+'0', R);

            break;

         case '-':

            c = y-x;

            empilha(c+'0', R);

            break;

         case '\*':

            c = y\*x;

            empilha(c+'0', R);

            break;

         case '/':

            c = x/y;

            empilha(c+'0', R);

            break;

         }

         // printf("O topo de R: %c\n", topo(R));

      }

   }

   int z = desempilha(R);

   destroip(&R);

   return z;

}

char \*prefixa(char \*e)

{

   static char s[256];

   int j = strlen(e);

   int i = 0;

   // printf("%d", j);

   Pilha P = pilha(256);

   //((2\*3)+(8/4))

   for (j; j > 0; j--)

   {

      // printf("%c",e[j]);      //   ))

      if (isdigit(e[j]))

      {

         s[i++] = e[j]; //   s[4,8,/,3,2,\*,+]

      }

      else if (strchr("+\*-/", e[j]))

      {

         empilha(e[j], P); // P []

      }

      else if (e[j] == '(')

      {

         s[i++] = desempilha(P);

      }

   }

   s[i++] = desempilha(P);

   // s[j] = '\0';

   // destroip(&P);

   return s;

}

int main(void)

{

   char e[513];

   printf("Infixa? ");

   gets(e);

   printf("Prefixa: %s\n", strrev(prefixa(e)));

   printf("Resultado: %d\n", charInt(valpre(strrev(prefixa(e)))));

   return 0;

}

