/\*

EX001

    Crie um programa que usa duas pilhas A e B para ordenar uma sequ�ncia de n n�meros dados

    pelo usu�rio. A ideia � organizar a pilha A de modo que nenhum item seja empilhado sobre outro

    menor (use a pilha B apenas para manobra) e, depois, descarregar e exibir os itens da pilha A.

\*/

#include <stdio.h>

#include "pilha.h"

void bubbleSort(int veta[], int n){

    int k, j, aux;

    for (k = 1; k < n; k++)

    {

        for (j = 0; j < n - 1; j++)

        {

            if (veta[j] > veta[j + 1])

            {

                aux = veta[j];

                veta[j] = veta[j + 1];

                veta[j + 1] = aux;

            }

        }

    }

}

int main(void)

{

    int tamanhoPilha;

    int n;

    int i;

    printf("Entre com a quantidade total de numeros: \n");

    scanf("%d", &tamanhoPilha);

    Pilha A = pilha(tamanhoPilha \* sizeof(int));

    Pilha B = pilha(tamanhoPilha\*sizeof(int));

    int veta[tamanhoPilha];

    //limpa vetor

    for(i = 0; i < tamanhoPilha; i++){

        veta[i]=0;

    }

    for (i = 0; i < tamanhoPilha; i++)

    {

        printf("Insira o numero: \n");

        scanf("%d", &n);

        empilha(n, A);

        veta[i] = n;

    }

    // imprime "pilha"

    printf("imprime pilha : \n");

    for (i = 0; i < tamanhoPilha; i++)

    {

        printf("%d\n", veta[i]);

        desempilha(A);

    }

    bubbleSort(veta, tamanhoPilha);

    // imprime "pilha" ordenada

    printf("\n\n\nimprime pilha ordenada: \n");

    for (i = tamanhoPilha-1; i >= 0; i--)

    {

        printf("%d\n", veta[i]);

        empilha(veta[i],B);

        empilha(topo(B),A);

    }

    printf("\n\n\nImprimindo pilha A: \n");

    for (i = 0; i < tamanhoPilha; i++)

    {

        int x;

        x=topo(A);

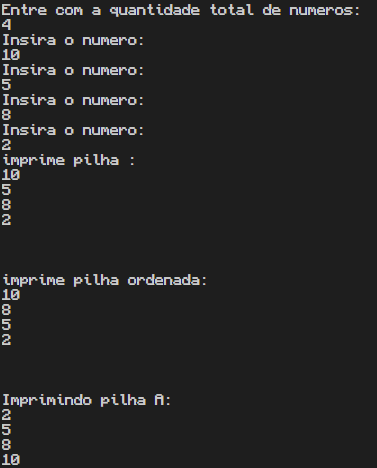
        printf("%d\n", x);

        desempilha(A);

    }

    return 0;

}



/\*

EX002

    Faça a alteração mínima necessária para que o programa do exercício anterior ordene os números

    em ordem decrescente, eliminando números repetidos.

\*/

#include <stdio.h>

#include "pilha.h"

void bubbleSort(int veta[], int n){

    int k, j, aux;

    for (k = 1; k < n; k++)

    {

        for (j = 0; j < n - 1; j++)

        {

            if (veta[j] > veta[j + 1])

            {

                aux = veta[j];

                veta[j] = veta[j + 1];

                veta[j + 1] = aux;

            }

        }

    }

}

int main(void)

{

    int tamanhoPilha;

    int n;

    int i;

    printf("Entre com a quantidade total de numeros: \n");

    scanf("%d", &tamanhoPilha);

    Pilha A = pilha(tamanhoPilha \* sizeof(int));

    Pilha B = pilha(tamanhoPilha \* sizeof(int));

    int veta[tamanhoPilha];

    //limpa vetor

    for(i = 0; i < tamanhoPilha; i++){

        veta[i]=0;

    }

    for (i = 0; i < tamanhoPilha; i++)

    {

        printf("Insira o numero: \n");

        scanf("%d", &n);

        empilha(n, A);

        veta[i] = n;

    }

    // imprime "pilha"

    printf("imprime pilha : \n");

    for (i = 0; i < tamanhoPilha; i++)

    {

        printf("%d\n", veta[i]);

        desempilha(A);

    }

    bubbleSort(veta, tamanhoPilha);

    // imprime "pilha" ordenada

    printf("\n\n\nimprime pilha ordenada: \n");

    for (i = 0; i < tamanhoPilha; i++)

    {

        printf("%d\n", veta[i]);

        empilha(veta[i],B);

        empilha(topo(B),A);

    }

    printf("\n\n\nImprimindo pilha A: \n");

    for (i = 0; i < tamanhoPilha; i++)

    {

        int x;

        x=topo(A);

        if(veta[i]!=topo(A)){

            printf("%d\n", x);

        }

        desempilha(A);

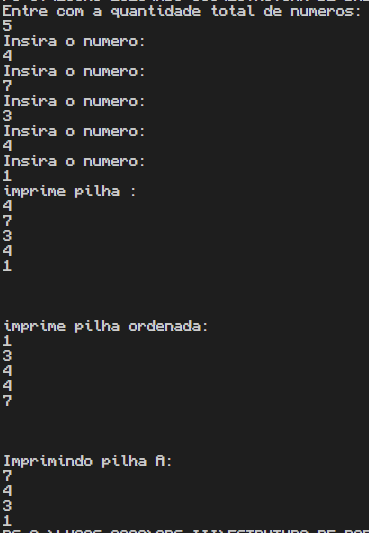
    }

    return 0;

}

;

}



/\*

EX003

    Usando uma pilha, crie um programa para inverter a ordem das letras nas palavras de uma frase,

    sem inverter a ordem das palavras na frase. Por exemplo, se for     digitada a frase "apenas um

    teste", o programa deverá produzir a seguinte saída: sanepa mu etset.

\*/

#include <stdio.h>

#include "pilha.h"

#include <string.h>

void reverse(char \*palavra){

    int x = strlen(palavra);

    Pilha X = pilha(x \* sizeof(int));

    for(int i = 0; i < x; i++){

        empilha(palavra[i],X);

    }

    for(int i = 0; i < x; i++){

        palavra[i]=desempilha(X);

        printf("%c", palavra[i]);

    }

    printf(" ");

}

int main()

{

    char txt[100];

    char \*palavra;

    printf("Digite uma frase: \n");

    gets(txt);

    int x = strlen(txt);

    char frase[x];

    strcpy(frase,txt);

    palavra = strtok(frase, " ");

    while(palavra != NULL){

        reverse(palavra);

        palavra = strtok(NULL, " ");

    }

    return 0;

}



Ex:004

/\*

Usando pilha, crie uma função para verificar se uma expressão composta apenas por chaves,

colchetes e parênteses, representada por uma cadeia de caracteres, está ou não balanceada. Por

exemplo, as expressões "[{()()}{}]" e "{[([{}])]}" estão balanceadas, mas as

expressões "{[(}])" e "{[)()(]}" não estão.

P = [ {[(}]) ]

c = [ {[(}]) ]

por espelhamento n deu certo, passou nos testes {[([{}])]}, {[(}]) e {[)()(]}, porem com o [{()()}{}] não funcionou.

acho que por varredura pode dar certo, mas não sei como fazer isso.

\*/

#include <stdio.h>

#include "pilha.h"

#include <string.h>

/\* CODIGO POR ESPELHAMENTO

int main() {

    Pilha P = pilha(256);

    char c[256];

    char aux;

    printf("Entre com a expressão: ");

    scanf("%s", c);

    //Inserindo todos os elementos de c em P

    for(int i =0; i < strlen(c); i++){

        empilha(c[i], P);

    }

    //Verificando se a expressão está balanceada

    for(int i = 0; i < strlen(c); i++){     //c[i] = {

        aux = desempilha(P);                //aux = }

        // printf("%c\n", aux);

        if(aux == '{' || aux =='[' || aux == '('){

            if(aux == '{'){

                if(c[i]!='}'){

                    printf("Expressao nao esta balanceada\n");

                    return 0;

                }

            }

            else if(aux == '['){

                if(c[i]!=']'){

                    printf("Expressao nao esta balanceada\n");

                    return 0;

                }

            }

            else if(aux == '('){

                if(c[i]!=')'){

                    printf("Expressao nao esta balanceada\n");

                    return 0;

                }

            }

        }

    }

    if(vaziap(P) && aux == c[0]){

        printf("Expressao esta balanceada\n");

    }

    return 0;

}

\*/

//CODIGO POR CONTAGEM

int main(){

    Pilha P = pilha(256);

    Pilha P2 = pilha(256);

    char c[256];

    printf("Entre com a expressao: ");

    scanf("%s", &c);                    //c = [ {[(}]) ]

    int tam= strlen(c);

    if(tam%2!=0){                       //se o tamanho da string for impar, a expressão não está balanceada

        printf("Expressao nao esta balanceada\n");

        return 0;

    }

    else{

         int countOC=0, countOP=0, countOCh=0, countCC=0, countCP=0, countCCh=0;

        for(int i=0; i<tam; i++){

            empilha(c[i], P);               //P = [ [{}] ]

        }

        char target = desempilha(P);        //target = ]

        printf("%c\n", target);             //target = ]

        int ok = 0;

        while(!vaziap(P)){                  //P = [  ]

                                            //target = [

            for(int i=0; i < tam; i++){

                if(target==']'){

                    if(c[i]=='['){

                        countOC++;          //1

                        countCC++;          //1

                        empilha(target, P2);//P2 = [ ] ]

                        i=tam+1;

                    }

                }

                if(target==')'){

                    if(c[i]=='('){

                        countOP++;

                        countCP++;

                        empilha(target, P2);

                        i=tam+1;

                    }

                }

                if(target=='}'){

                    if(c[i]=='{'){

                        countOCh++;         //1

                        countCCh++;         //1

                        empilha(target, P2);//P2 = [  ]

                        i=tam+1;

                    }

                }

                if(target=='[' && topo(P2)==']' || target=='{' && topo(P2)=='}' || target=='(' && topo(P2)==')'){

                    desempilha(P2);

                    ok++;

                    i=tam+1;

                }

            }

            target = desempilha(P);

        }

        ok++;

        int metade = tam/2;

        if(countOC == countCC && countOP == countCP && countOCh == countCCh && ok == metade){

            printf("Expressao esta balanceada\n");

        }

        else{

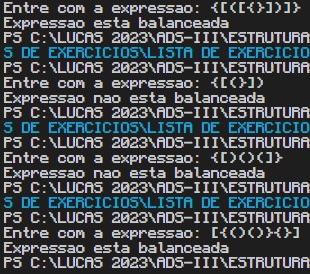
            printf("Expressao nao esta balanceada\n");

        }

    }

    return 0;

}



//EX005

Não terá saida nenhuma, pois a Pilha não está recebendo valor nenhum. O empilha(s,P) não está funcionando corretamente, visto que a função deve ser executada para cada caractere da cadeia "s[x]" e não para uma string inteira "s". Outro erro que está acontecendo é a forma de impressão usando o while.

//A função puts() não está reconhecendo o caractere desempilhado pela função desempilha(P)

//O tamanho da pilha não suporta todos os caracteres inseridos

//EX006

#include <stdio.h>

#include "pilha.h"

#include <string.h>

int main(void)

{

    Pilha P = pilha(12);

    char s[11];

    for (int i = 1; i <= 3; i++)

    {

        printf("? ");

        gets(s);

        char \*palavra = strdup(s);

        // printf("%x", palavra);

        for (int j = 0; s[j]; j++)

        {

            empilha(palavra[j], P);

        }

        free(palavra);

    }

    while (vaziap(P) == 0)

        printf("%c", desempilha(P));

    return 0;

}

