**UNIVERSIDAD AUTONOMA** **DE BAJA CALIFORNIA**

**Algoritmos y estructura de datos**

**Practica 5. Pilas estáticas**

Alumno: Caudillo Sánchez Diego

Matricula: 1249199

Grupo: 551

Docente: Alma Leticia Palacios Guerrero

Fecha de entrega: 29/Marzo/2019

**Introducción**

**Competencia**

**Problema**

**Código**

/\*

Alumno: Caudillo Sánchez Diego

Matricula: 1249199

Grupo: 551

Asignatura: Algoritmos y estructuras de datos

Docente: Alma Leticia Palacios Guerrero

Fecha de entrega: 29-marzo-2019

DESCRIPCION DEL PROGRAMA

Se pide diseñar e implementar un algoritmo basado en pilas estáticas que

reciba una y evalúe una expresión dada por el usuario y determine si es

un palíndromo o no.

\*/

/\*Librerías utilizadas\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*Headers\*/

void deleteSpace(char\* array);

int palindrome(char\* array);

void push(char\* array, char\* stack);

char pop(char\* stack, int index);

void printStack(int size, char\* stack);

/\*Función main, donde se ejecutan todos los métodos creados\*/

int main(int argc, char const \*argv[])

{

char array[20] = {0};

printf("Ingrese palabra: ");

gets(array);

deleteSpace(array);

// si devuelve -1 no es palíndromo, caso contrario si lo es.

if(palindrome(array) == -1) printf("==========================\n==== No es palindrome ====\n==========================\n");

else printf("=========================\n===== Es palindrome =====\n=========================\n");

return 0;

}

/\*

Descripción

Función que recorre un vector buscando espacios para borrarlos.

Parámetros

-array: arreglo de caracteres el cual se van a borrar los espacios.

\*/

void deleteSpace(char\* array)

{

for(int i=0; array[i] != 0; i++ )

{

if(array[i] == ' ')

{

for(int j = i; array[j] != 0; j++ )

array[j] = array[j+1];

}

}

}

/\*

Descripción

Función que realiza un push de un vector hacia la pila. La función mete el

vector entero a la pila.

Parámetros

-array: es el arreglo el cual se va introducir a la pila.

-stack: un arreglo el cual se reserva como la pila y donde los datos que

recibe son del "array"

\*/

void push(char\* array, char\* stack)

{

//array[position] = 0;

for(int i=strlen(array)-1, j=0; i >= 0; i--,j++)

stack[j] = array[i];

}

/\*

Descripción

Función que realiza un pop a la pila, quiere decir que extrae un dato de ella.

la función solo va removiendo datos de uno en uno. Devuelve el dato extraído.

Parámetros

-stack: array el cual actúa como la pila y es la cual de donde se extraen los

datos.

-index: la posición actual donde se la pila quedo apuntada.

\*/

char pop(char\* stack, int index)

{

char data\_out;

data\_out = stack[index]; // guarda el dato que salió

stack[index] = 0; // coloca un cero en la posición del dato que salió, indicando

// que esa posición se ha vaciado.

return data\_out; // regresa el dato sacado de la pila.

}

/\*

Descripción

Función que imprime los datos actuales en la pila.

Parámetros

-size: entero el cual tiene el tamaño de la pila

-stack: la pila, de la cual se van a extraer los datos

\*/

void printStack(int size, char\* stack)

{

for(int i = 0; i < size; i++)

printf("|\_%c\_|\n", stack[i]);

}

/\*

Descripción

Función que verifica un arreglo si son palíndromos con la ayuda de pila.

Primeramente, el arreglo se carga en la pila (LIFO) y se van comparando

los caracteres de la pila y del arreglo, si todos coinciden, quiere decir

que la palabra es palíndroma y se retorna un valor de 1, caso contrario se

devuelve un -1 indicando que la palabra o frase no es palíndromo.

Parámetros

-array: arreglo que contiene la palabra o frase que se quiere verificar si

es palíndromo o no.

\*/

int palindrome(char\* array)

{

char stack[20], letra;

int index=0;

push(array,stack); // se realiza un push del arreglo.

puts("STACK");

while(index <= strlen(array))

{

printStack(strlen(array),stack); // impresion del stack con datos actualizados

if(index < strlen(array))

{

printf("\n==>Actualizado<==\n");

letra = pop(stack, index); // letra contiene el dato de cuando se realiza un pop.

printf("Dato que salio: [%c]\n", letra);

// Si el dato extraído de la pila no coincide con el dato del arreglo

// entonces se devuelve un -1 indicando que la palabra no es palíndroma.

if(letra != array[index])

{

printf("[%c] es diferente de [%c]\n", letra, array[index]);

return -1;

}

}index++;

}return 1; // si al recorrer todo el arreglo y las letras coinciden, entonces la palabra en palíndromo.

}

**Desarrollo**

La frase debe empezar a evaluarse después de haber sido capturada.

El programa debe mostrar el estado de la(s) pila(s) durante el proceso de evaluación de cada caracter de la expresión.

El programa debe presentar cada pareja de caracteres que se esté comparando durante el proceso de evaluación de la expresión.

El tamaño máximo de la expresión será preestablecido por el programador, pero el usuario podrá introducir expresiones de menor tamaño.

El lenguaje de programación a utilizar es libre, C o Java.

El programa debe repetirse N veces, hasta que el usuario elija la opción de salida.

Deben eliminarse los espacios de las frases.

Las expresiones deben tener al menos un carácter.

La interfaz del programa debe estar bien organizada.

**Evidencia**

**Conclusión**

**Bibliografías**