

Objetivo: Aplicar los conocimientos de pilas en lenguaje C.

Pseudo codigo

void create\_DS(Node\_DS \*\*ds)

    \*ds=NULL;

push\_DS(Node\_DS \*\*ds,element e)

    Node\_DS \*new

    new=(Node\_DS\*)malloc(sizeof(Node\_DS))

    new->e=e

    new->next=\*ds

    \*ds=new

is\_Empty\_DS(Node\_DS \*ds)

    if (ds==NULL)

        return 1;

    else

        return 0;

pop\_DS(Node\_DS \*\*ds)

    Node\_DS \*node\_aux

    element elem\_aux

    if(is\_Empty\_DS(\*ds))

    printf("ERROR DE UNDERFLOW”)

    else

        node\_aux=\*ds

        \*ds=node\_aux->next

        elem\_aux=node\_aux->e

        free(node\_aux)

        return elem\_aux

printDS(Node\_DS \*ds)

    Node\_DS \*aux

    aux = ds

    while (aux != NULL)

        printf("%d ", aux->e)

        aux = aux->next

    printf("\n")

Codigo

DStack.h

#ifndef \_DSTACK

#define \_DSTACK

typedef char element;

typedef struct Node\_DS{

    element e;

    struct Node\_DS \*next;

}Node\_DS;

void create\_DS(Node\_DS \*\*ds);

void push\_DS(Node\_DS \*\*ds,element e);

int is\_Empty\_DS(Node\_DS \*ds);

element pop\_DS(Node\_DS \*\*ds);

void printDS(Node\_DS \*ds);

void llenarPila(Node\_DS \*\*P);

#endif

DStack.c

#include <stdio.h>

#include <stdlib.h>

#include "DStack.h"

void create\_DS(Node\_DS \*\*ds){

    \*ds=NULL;

}

void push\_DS(Node\_DS \*\*ds,element e){

    Node\_DS \*new;

    new=(Node\_DS\*)malloc(sizeof(Node\_DS));

    new->e=e;

    new->next=\*ds;

    \*ds=new;

}

int is\_Empty\_DS(Node\_DS \*ds){

    if (ds==NULL)

        return 1;

    else

        return 0;

}

element pop\_DS(Node\_DS \*\*ds){

    Node\_DS \*node\_aux;

    element elem\_aux;

    if(is\_Empty\_DS(\*ds)){

    printf("\nERROR DE UNDERFLOW\n");

    } else {

        node\_aux=\*ds;

        \*ds=node\_aux->next;

        elem\_aux=node\_aux->e;

        free(node\_aux);

        return elem\_aux;

    }

}

void printDS2(Node\_DS \*ds){

    Node\_DS \*aux;

    aux=ds;

    while(aux!=NULL){

        printf("%d",aux->e);

        aux=aux->next;

    }

}

void deleteDS(Node\_DS \*\*ds) {

    element e;

    while (!is\_Empty\_DS(\*ds)) {

        e = pop\_DS(ds);

        printf("Elemento eliminado: %d\n", e);

    }

}

void printDS(Node\_DS \*ds) {

    Node\_DS \*aux;

    aux = ds;

    while (aux != NULL) {

        printf("%d ", aux->e);

        aux = aux->next;

    }

    printf("\n");

}

void llenarPila(Node\_DS \*\*P){

    printf("Desea agregar un elemento a la pila? 1=Si, 0=No\n");

    int n;

    scanf("%d",&n);

    while(n!=0){

        element e;

        printf("Ingrese el elemento a agregar: ");

        scanf("%d",&e);

        push\_DS(P,e);

        printf("Desea agregar un elemento a la pila? 1=Si, 0=No\n");

        scanf("%d",&n);

    }

}

Main.c

EJERCICIO 1 Y 2

#include <stdio.h>

#include "DStack.h"

//jazmin montserrath velazquez trinidad

//Rivera Plascencia Bryan

void copyStack(Node\_DS \*, Node\_DS \*\*);

void Suma(Node\_DS \*, Node\_DS \*, Node\_DS \*\*);

void main() {

    Node\_DS \*P1, \*P2, \*P3, \*P4, \*P5;

    create\_DS(&P1);

    create\_DS(&P2);

    create\_DS(&P3);

    create\_DS(&P4);

    create\_DS(&P5);

    //ejericio 1

    llenarPila(&P1);

    printf("\nPILA Original:\n\n");

    printDS(P1);

    copyStack(P1, &P2);

    printf("\nCopia de la PILA:\n");

    printDS(P2);

    //ejercicio 2

    printf("\n--------------------------------\n");

    llenarPila(&P3);

    printf("\n\n");

    llenarPila(&P4);

    Suma(P3, P4, &P5);

    printf("\n");

    printDS(P5);

}

void copyStack(Node\_DS \*P1, Node\_DS \*\*P2) {

    if (is\_Empty\_DS(P1)) {

        printf("La pila esta vacia\n");

        return;

    }

    Node\_DS \*aux = NULL;

    element e;

    while (!is\_Empty\_DS(P1)) {

        e = pop\_DS(&P1);

        push\_DS(&aux, e);

    }

    while(!is\_Empty\_DS(aux)){

        e = pop\_DS(&aux);

        push\_DS(P2, e);

    }

}

void Suma(Node\_DS \*P1, Node\_DS \*P2, Node\_DS \*\*P3) {

    if (is\_Empty\_DS(P1) || is\_Empty\_DS(P2)) {

        printf("Al menos una de las pilas está vacía\n");

        return;

    }

    Node\_DS \*aux = NULL; // Pila auxiliar para almacenar los acarreos

    while (!is\_Empty\_DS(P1) || !is\_Empty\_DS(P2) || !is\_Empty\_DS(aux)) {

        element e1 = 0, e2 = 0, auxi = 0;

        if (!is\_Empty\_DS(P1))

            e1 = pop\_DS(&P1);

        if (!is\_Empty\_DS(P2))

            e2 = pop\_DS(&P2);

        if (!is\_Empty\_DS(aux))

            auxi = pop\_DS(&aux);

        element suma = e1 + e2 + auxi;

        if (suma > 9) {

            push\_DS(&aux, 1); // Agregar el acarreo a la pila auxiliar

            suma -= 10;

        }

        push\_DS(P3, suma);

    }

}

EJERCICIO 3

#include <stdio.h>

#include <stdlib.h>

#include "DStack.h"

//jazmin montserrath velazquez trinidad

//Rivera Plascencia Bryan

void equilibrado(Node\_DS \*\*);

void main() {

    char \*A;

    A = (char\*)malloc(100\*sizeof(char));

    printf("Introduce una cadena: ");

    scanf("%s", A);

    printf("\n");

    Node\_DS \*P1;

    create\_DS(&P1);

    char a;

    int i;

    for(i = 0; A[i] != '\0'; i++){

        a = A[i];

        push\_DS(&P1, a);

    }

    equilibrado(&P1);

}

void equilibrado(Node\_DS \*\*P1){

    int contPA = 0;

    int contPC = 0;

    int contCA = 0;

    int contCC = 0;

    int contLA = 0;

    int contLC = 0;

    int P,C,L;

    element e;

    while(!is\_Empty\_DS(\*P1)){

        e = pop\_DS(P1);

        if(e == '('){

            contPA++;

        } else if(e == ')'){

            contPC++;

        } else if(e == '['){

            contCA++;

        } else if(e == ']'){

            contCC++;

        } else if(e == '{'){

            contLA++;

        } else if(e == '}'){

            contLC++;

        }

    }

    if(contPA == contPC && contCA == contCC && contLA == contLC){

        printf("La cadena es equilibrada\n");

    }else{

        printf("La cadena no es equilibrada\n");

        if(contPA > contPC){

            P = contPA - contPC;

            printf("Faltan %d (\n", P);

        }else if(contPA < contPC){

            P = contPC - contPA;

            printf("Faltan %d )\n", P);

        }

        if(contCA > contCC){

            C = contCA - contCC;

            printf("Faltan %d [\n", C);

        }else if(contCA < contCC){

            C = contCC - contCA;

            printf("Faltan %d ]\n", C);

        }

        if(contLA > contLC){

            L = contLA - contLC;

            printf("Faltan %d {\n", L);

        }else if(contLA < contLC){

            L = contLC - contLA;

            printf("Faltan %d }\n", L);

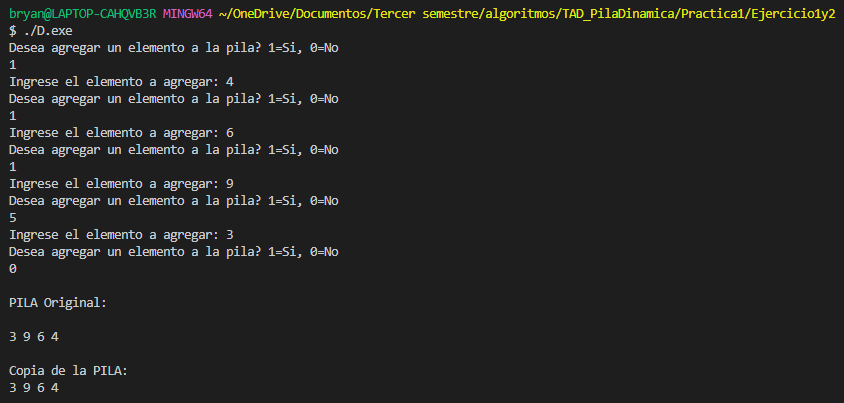
        }

    }

}

RESULTADOS

Ejercicio 1:



Ejercicio 2:

Texto

Descripción generada automáticamente

Ejercicio 3:

Texto

Descripción generada automáticamente

CONCLUSIONES INDIVIDUALES

Rivera Plascencia Bryan

1. ¿Cuál fue su experiencia con la práctica?

La practica no me resulto muy dificil ya que era algo un poco intuitivo, lo único que me resulto un poco complicado fue el uso de los apuntadores

1. Si tuvieron errores ¿Cómo lo solucionaron?

Investigando como era el manejo de los apuntadores

1. Si aprendieron algo nuevo ¿Qué fue?

Aprendí sobre el uso de apuntadores en las pilas

Velazquez Trinidad Jazmin Montserrath

1. ¿Cuál fue su experiencia con la práctica?

Personalmente considero que fue una practica sencilla ya que anteriormente ya lo había visto, sin embargo, las funciones definidas fueron un poco diferentes a las que en su momento llegue a usar

1. Si tuvieron errores ¿Cómo lo solucionaron?

En el tercer ejercicio personalmente se me complico un poco el definir que no estuviera balanceada indicar el faltante; sin embargo, ya investigando y preguntando a mi compañero pudimos resolverlo

1. Si aprendieron algo nuevo ¿Qué fue?

Aprendí el uso de ambas pilas y diferenciar bien una de otra

Referencias

<https://learn.microsoft.com/es-es/cpp/cpp/pointers-cpp?view=msvc-170>