



UNIVERSIDAD CATÓLICA ANDRÉS BELLO
CIUDAD GUAYANA
FACULTAD DE CIENCIAS Y TECNOLOGÍA
INGENIERÍA INFORMÁTICA
ALGORITMOS Y PROGRAMACIÓN II

FORERO, MARÍA PAULA
FREITAS, VÍCTOR
MORALES, CRISTINA
RODRÍGUEZ, JOSÉ ANDRÉS

PROYECTO 1:
CALCULADORA DE NOTACIÓN POLACA INVERSA

DOCENTE: LÁREZ, JESÚS

CIUDAD GUAYANA, NOVIEMBRE DE 2021

ANÁLISIS DEL PROBLEMA

El presente proyecto es una calculadora que funciona en su totalidad con Notación Polaca Inversa (RPN). Este contiene una entrada, la cual es llamada "input", y funcionará para recibir los datos ingresados por el usuario. Una vez pedida la entrada se procederá a evaluar si esta es un número, un símbolo de una operación matemática, una función de memoria, una función matemática, una función miscelánea, o una base numérica.

Esto servirá para analizar qué funciones se deben implementar para cumplir con el comando indicado. Finalmente, dependiendo del comando, se mostrará por pantalla el resultado de la operación o se guardará dicho valor en la pila, para ser operado nuevamente o simplemente almacenado y mostrado al imprimirla.

La calculadora recibe parámetros para implementar:

- Funciones con memoria, como: STO (Store), MRCL (Memory Recall), S+ (Sumar a la memoria), y S- (restar a la memoria).
- Funciones matemáticas, como: Seno, Coseno, Tangente, Raíces, Potencias, Logaritmos, Inversas, y Factoriales.
- Funciones misceláneas, como: RCL (Recall), CLR (Clear), DSP (Display), y SWP (Swap).
- Funciones para bases numéricas, como: Binary, Octal, Decimal, Hexa.

DISEÑO DE LA SOLUCIÓN

Algoritmo de la librería "stack.h":

```
typedef struct node { //Se define la estructura de un nodo con valor flotante
    flotante value
    struct node *next
} Node

función push(Node **stack, flotante x) { //Se crea la función push
    Node *p = (Node*)malloc(sizeof(Node)) //Se define el espacio de memoria del nodo
    p->value = x
    p->next = *stack
    *stack = p //Se iguala la pila al nodo creado
```

```
}
```

```
función pop(Node **head) {           //Se crea la función pop
    Node *p
    flotante v
    v = (*head)->value
    p = (*head)->next
    free(*head)                       //Borra el elemento
    *head = p
    return v                          //Retorna el valor al programa
}
```

```
función display(Node *stack) {        //Se crea la función display
    Para (;stack != NULL; stack = stack->next) {
        Escribir("%.2f ",stack->value) //Imprime la pila
    }
    Fin_para}
}
```

```
función *liberar(Node *raiz){         //Se crea la función liberar
    Node *reco = raiz
    Node *bor
    Mientras (reco != NULL)
    {
        bor = reco
        reco = reco->next
        free(bor)                     //Borra el elemento
    }
    Fin_mientras}
return reco                          //Devuelve la pila vacía
}
```

```
función radianes(flote valor){        //Se crea la función para pasar a radianes
    constante double PI=3.141592653589
    double rad
    rad=valor*PI/180                 //Calcula el valor en radianes
    return rad                      //Retorna el valor
}
```

```
función grados(flote valor){          //Se crea la función para pasar a grados
    constante double PI=3.141592653589
    double grad
    grad=valor*180/PI                //Calcula el valor en grados
}
```

```
return grad      //Retorna el valor
}
```

```
función binario(flotante n){  //Se crea la función para pasar a binario
    flotante binary=0
    entero n1=n
    entero a[n1], i=0
    Mientras (n1>0){
        a[i]=n1%2      //Calcula los restos del número entre 2
        i=i+1
        n1=n1/2
    Fin_mientras}
    Mientras (0<=i){      //Concatena los restos desde el último al primero
        Si (a[i]==1 && i ==0){
            binary ++
        }Sino
        Si (a[i]==0)
            binary*= 10
        Sino
        Si (a[i]==1 && i !=0){
            binary++
            binary*=10
        }Fin_si}
        i--
    Fin_mientras}
    return binary      //Retorna el valor binario
}
```

```
función octadecimal(flotante v1){  //Se crea la función para pasar a octal
    flotante NumOctal = 0, i = 1
    entero a= v1

    Mientras (a != 0)
    {
        NumOctal += (a % 8) * i      //Calcula los restos del número entre 8
        a /= 8
        i *= 10
    }Fin_mientras}
    return NumOctal      //Retorna el número en binario
}
```

```

función Hexa(floatante v1) { //Se crea la función para pasar a hexal
    long entero quotient
    entero i=1,j,temp
    char hexadecimalNumber[100]
    quotient = v1
    Mientras(quotient!=0) {
        temp = quotient % 16 //Calcula los restos del número entre 16
        //Para convertir el entero a char
        Si ( temp < 10)
            temp =temp + 48
        Sino
            temp = temp + 55
        Fin_si
        hexadecimalNumber[i++]= temp
        quotient = quotient / 16
    Fin_mientras}
    Para (j = i -1 ;j> 0;j--)
        Escribir ("%c",hexadecimalNumber[j]) //Muestra por pantalla el resultado
    Fin_para
}

función print(entero valor2, floatante valor){ //Se crea la función para verificar
    Si (valor-valor2 !=0) //si el número tiene decimales
        Escribir ("%0.2f\n", valor) //Imprime el resultado con decimales
    Sino
        Escribir("%d\n",valor2) //Imprime el resultado sin decimales
    Fin_si
}

```

Algoritmo del programa "main.c":

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "stack.h"

función main(int argc, char *argv[]) {

    //Pila de números
    Node* stack=(Node*)malloc(sizeof(Node))

    //Modo de operación en terminal UNIX

```

```

flotante memory=0
entero i, chara = 0
entero flag = 0 // 0=Menú interactivo // 1=Operando desde el terminal//
Si (argc>1) {
    flag = 1
    Para (i=1;argv[i] != NULL && atof(argv[i]) != 0; i++) {
        push(&stack,atof(argv[i]))
    }
    Fin_para}
    i--
    Fin_si}

```

// Bucle de input de usuario

```

Mientras(1) {
    char input[10]
    entero valor2
    flotante v1,v2,valor
    Si (flag==1) {
        i++
        Si (argv[i]==NULL)
            exit(0)
        Fin_si
        strcpy(input, argv[i])
    }
    Fin_si}
    Si (flag==0) {
        Escribir ("CALC>")
        Leer ("%s", &input)
    }
    Fin_si}

```

//Redirije el flujo del código según el input del usuario

//SUMA//

```

Si (strcmp(input, "+") == 0) {
    v1 = pop(&stack)
    v2 = pop(&stack)
    valor = v1 + v2
    push(&stack, valor)
    valor2=valor
    print (valor2, valor)
}

```

}Sino

//RESTA//

```

Si (strcmp(input, "-") == 0) {
    v1 = pop(&stack)
    v2 = pop(&stack)
    valor = v2 - v1
    push(&stack, valor)
}

```

```
    valor2=valor
    print (valor2, valor)
}Sino
```

```
//MULTIPLICACIÓN//
```

```
Si (strcmp(input, "x") == 0) {
    v1 = pop(&stack)
    v2 = pop(&stack)
    valor = v1 * v2;
    push(&stack, valor)
    valor2=valor
    print (valor2, valor)
}
```

```
}Sino
```

```
//DIVISIÓN//
```

```
Si (strcmp(input, "/") == 0) {
    v1 = pop(&stack)
    v2 = pop(&stack)
    valor = v2 / v1
    push(&stack, valor)
    valor2=valor
    print (valor2, valor)
}
```

```
}Sino
```

```
//RAÍZ CUADRADA//
```

```
Si (strcmp(input , "RA") == 0){
    valor=sqrt(v1=pop(&stack))
    push(&stack, valor)
    valor2=valor
    print (valor2, valor)
}
```

```
}Sino
```

```
//POTENCIA//
```

```
Si (strcmp(input, "PO")==0){
    v1= pop(&stack)
    v2= pop(&stack)
    valor = pow(v2,v1)
    push(&stack, valor)
    valor2=valor
    print (valor2, valor)
}
```

```
}Sino
```

```
//FACTORIAL//
```

```
Si (strcmp(input , "FACT") == 0){
    valor=1
    v1 = pop(&stack)
```

```
Para (; v1>1; v1--)  
    valor *= v1  
Fin_para  
valor2=valor  
print (valor2, valor)  
push(&stack, valor)  
}Sino
```

```
//LOGARITMO//
```

```
Si (strcmp(input , "LOG")==0){  
    v1= pop(&stack)  
    valor=(log(v1))  
    valor2=valor  
    print (valor2, valor)  
    push(&stack, valor)  
}Sino
```

```
//COSENO//
```

```
Si (strcmp(input , "COS")==0){  
    v1=pop(&stack)  
    valor=cos(radianes(v1))  
    valor2=valor  
    print (valor2, valor)  
    push(&stack,valor)  
}Sino
```

```
//SENO//
```

```
Si (strcmp(input , "SEN")==0){  
    v1=pop(&stack)  
    valor=sin(radianes(v1))  
    valor2=valor  
    print (valor2, valor)  
    push(&stack,valor)  
}Sino
```

```
//TANGENTE//
```

```
Si (strcmp(input , "TAN")==0){  
    v1=pop(&stack)  
    valor=tan(radianes(v1))  
    valor2=valor  
    print (valor2, valor)  
    push(&stack,valor)  
}Sino
```

```
//ARCOSENO//
```

```
Si (strcmp(input , "ASEN")==0){  
    v1= pop(&stack)
```



```

Si ((v1>=-1) && (v1<=1)){
    valor=(grados(asin(v1)))
    valor2=valor
    print (valor2, valor)
    push(&stack, valor)
}Sino
    Escribir ("ERROR DE DOMINIO")
    push(&stack, v1)
Fin_si
}Sino

```

//ARCOCOSENO//

```

Si (strcmp(input , "ACOS")==0){
    v1= pop(&stack)
    Si ((v1>=-1) && (v1<=1)){
        valor=(grados(acos(v1)))
        valor2=valor
        print (valor2, valor)
        push(&stack, valor)
    }Sino
        Escribir ("ERROR DE DOMINIO")
        push(&stack, v1)
    Fin_si
}Sino

```

//ARCOTANGENTE//

```

Si (strcmp(input , "ATAN")==0){
    v1=pop(&stack)
    valor=grados((atan(v1)))
    valor2=valor
    print (valor2, valor)
    push(&stack,valor)
}Sino

```

//FUNCIÓN DISPLAY//

```

Si (strcmp(input, "DSP") == 0) {
    display(stack)
}Sino

```

//FUNCIÓN RECALL//

```

Si (strcmp(input , "RCL") == 0){
    Si (stack != NULL){
        valor=pop(&stack)
        valor2=valor
        print(valor2, valor)
        push(&stack,valor)
    }Sino{

```

```
        Escribir ("La lista esta vacia\n")
    } Sino
```

```
//FUNCIÓN CLEAR//
```

```
Si (strcmp(input, "CLR") == 0) {
    stack=liberar(stack)
} Sino
```

```
//FUNCIÓN SWAP//
```

```
Si (strcmp(input, "SWAP")==0){
    v1=pop(&stack)
    v2=pop(&stack)
    push(&stack,v1)
    push(&stack,v2)
    Escribir ("Cambiados con exito\n")
}Sino
```

```
//FUNCIÓN BINARIO//
```

```
Si (strcmp(input, "BINARY")==0){
    v1=pop(&stack)
    v1=v1/1
    valor=binario(v1)
    push(&stack,valor)
    Si (flag == 1)
        Escribir("%.0f\n", valor)
    Fin_si
}Sino
```

```
//FUNCIÓN OCTADECIMAL//
```

```
Si (strcmp(input, "OCTAL")==0){
    v1=pop(&stack)
    valor=octadecimal(v1)
    push(&stack,valor)
    Si (flag == 1)
        Escribir("%.0f\n", valor)
    Fin_si
}Sino
```

```
//FUNCIÓN HEXADECIMAL//
```

```
Si (strcmp(input, "HEXAL")==0){
    v1=pop(&stack)
    Hexa(v1)
    push(&stack,v1)
}Sino
```

```
//FUNCIÓN ALMACENAR EN MEMORIA
```

```
Si (strcmp(input, "STO")==0){
```

```

        v1=pop(&stack)
        memory=v1
        Escribir ("Se ha realizado el almacenado con exito\n")
        push(&stack,memory)
    }Sino

    //FUNCIÓN RECODAR VALOR DE MEMORIA
    Si (strcmp(input, "MRCL")==0){
        valor2=memory
        print(valor2, memory)
    }Sino

    //FUNCIÓN SUMAR A LA MEMORIA
    Si (strcmp(input, "S+")==0){
        v1=pop(&stack)
        memory+=v1
        valor2=memory
        print(valor2, memory)
    }Sino

    //FUNCIÓN RESTAR A LA MEMORIA
    Si (strcmp(input, "S-")==0){
        v1=pop(&stack)
        memory-=v1
        valor2=memory
        print(valor2, memory)
    }Sino

    //FUNCIÓN QUIT//
    Si (strcmp(input, "QUIT") == 0) {
        exit(0)
    }Sino

    Si (input != '\0'){
        push(&stack,atof(input))
    }
    Fin_si}
Fin_mientras}
return 0
}

```

DETALLES DE LA IMPLEMENTACIÓN

Detalles de la implementación en la librería “stack.h”:

Función Push: Esta función permite insertar un nuevo elemento en el tope de la pila. Recibe la pila y el nuevo valor que se colocará en el tope, se crea un nodo

auxiliar que su valor apunte a el valor recién ingresado como parámetro, y que ese nodo auxiliar apunte a la pila. Se iguala la pila a ese nodo, quedando así el valor en el tope.

Función Pop: Esta función permite sacar un elemento de la pila, y después lo elimina de esta. Recibe la pila, y esa pila se iguala a un auxiliar, se empieza recorrer el auxiliar mientras que el auxiliar apuntando a siguiente sea diferente a nulo, entonces se iguala el valor del auxiliar a auxiliar apuntando a siguiente. Cuando llega al final de la condición toma el valor actual del auxiliar apuntando a valor, ese valor se retorna y se borra de la lista.

Función Display: Esta función permite imprimir por pantalla todos los elementos que conforman una pila. Recorre toda la pila, imprimiendo cada valor hasta que encuentre NULL, que le indicará el final de la pila.

Función Clear: Esta función elimina cada uno de los elementos que contiene la pila. Recorre toda la pila liberando cada uno de los elementos, hasta que llegue a NULL.

Función Radianes: Esta función permite convertir un valor numérico a radianes, ya que las funciones trigonométricas únicamente reciben parámetros en radianes.

Función Grados: Esta función permite convertir un valor dado en radianes a un valor numérico, ya que las funciones trigonométricas inversas únicamente reciben parámetros en grados.

Función Binario: Esta función permite convertir un valor que se encuentra en una base decimal a una base binaria. Divide el número ingresado entre dos para tomar el resto y dividirlo nuevamente hasta que este sea menor a 0. Luego va concatenando los restos de las divisiones, desde el último hasta el primero obtenido.

Función Octadecimal: Esta función permite convertir un valor que se encuentra en una base decimal a una base octadecimal. Divide el número ingresado entre 8, toma el resto divide nuevamente hasta que el número sea 0. Luego va concatenando los restos de las divisiones, desde el último hasta el primero obtenido.

Función Hexadecimal: Esta función permite convertir un valor que se encuentra en una base decimal a una base hexadecimal. Divide el número entre 16 y toma el resto, si este es menor a 10 se le suma 48, y si es mayor se le suma

55, luego vuelve a dividirse entre 16 hasta que el cociente sea menor a 16, donde se le asignará la letra correspondiente al resto.

Función print: Esta función permite imprimir por pantalla los valores enteros sin decimales. Se toma una variable “valor2” que es entera, y otra variable “valor” que es un flotante, luego se restan para evaluar si tiene decimales, si el resultado es distinto a 0 se imprimirá el valor flotante, y si es igual a 0 se imprimirá el valor entero.

Detalles de la implementación en el programa “main.c”:

Se hará uso de las siguientes librerías: “stdio.h”, “stdlib.h”, “string.h”, “math.h”, y “stack.h”.

Pila de números: Se le asigna con la función “malloc” un espacio de memoria a la pila “stack” y a la pila auxiliar “memory”.

Modo de operación en terminal UNIX: Se crea una bandera, donde si su valor es 0 se usará el menú interactivo y si es 1 se operará desde la terminal.

Bucle de input del usuario: Se declara una variable tipo char llamada “input”, la cual obtendrá el valor que ingrese el usuario por pantalla, además se asignarán otras variables auxiliares a esta, para utilizarlas en otras funciones (v1, v2, valor, valor2).

Si el valor de input es “+”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila, luego se sumarán, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “-”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila, luego se restarán, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “X”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila, luego se multiplicarán, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “/”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila, luego se dividirán, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “RA”, se utilizará la función “sqrt” para calcular la raíz del último valor obtenido, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “PO”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila, luego se elevará el segundo valor al primero, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “FACT”, se utilizará la función “pop” para sacar el último valor ingresado a la pila, luego se calculará su factorial, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “LOG”, se utilizará la función “pop” para sacar el último valor ingresado a la pila, luego se sacará su logaritmo natural con la función “log”, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “COS”, se utilizará la función “pop” para sacar el último valor ingresado a la pila, luego se pasará dicho valor como parámetro a la función “radianes”, con la función “cos” se calculará el resultado, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “SEN”, se utilizará la función “pop” para sacar el último valor ingresado a la pila, luego se pasará dicho valor como parámetro a la función “radianes”, con la función “sen” se calculará el resultado, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es “TAN”, se utilizará la función “pop” para sacar el último valor ingresado a la pila, luego se pasará dicho valor como parámetro a la función “radianes”, con la función “tan” se calculará el resultado, y con la función “push”, se ingresará el resultado a la pila. También será mostrado por pantalla con la función “print”.

Si el valor de input es "ASEN", se utilizará la función "pop" para sacar el último valor ingresado a la pila, luego se pasará dicho valor como parámetro a la función "grados", con la función "asin" se calculará el resultado, y con la función "push", se ingresará el resultado a la pila. También será mostrado por pantalla con la función "print".

Si el valor de input es "ACOS", se utilizará la función "pop" para sacar el último valor ingresado a la pila, luego se pasará dicho valor como parámetro a la función "grados", con la función "acos" se calculará el resultado, y con la función "push", se ingresará el resultado a la pila. También será mostrado por pantalla con la función "print".

Si el valor de input es "ATAN", se utilizará la función "pop" para sacar el último valor ingresado a la pila, luego se pasará dicho valor como parámetro a la función "grados", con la función "atan" se calculará el resultado, y con la función "push", se ingresará el resultado a la pila. También será mostrado por pantalla con la función "print".

Si el valor de input es "DSP", se imprimirá por pantalla toda la pila con la función "display".

Si el valor de input es "RCL", se recorrerá toda la pila y con la función "pop" se sacará el último valor ingresado, se imprimirá por pantalla con la función "print", y con la función "push", se ingresará el valor devuelta a la pila.

Si el valor de input es "CLR", se borrará toda la pila utilizando la función "liberar".

Si el valor de input es "SWAP", se utilizará la función "pop" para sacar los últimos dos valores ingresados a la pila, y con la función "push", se ingresarán los valores intercambiados devuelta a la pila.

Si el valor de input es "BINARY", se utilizará la función "pop" para sacar el último valor ingresado a la pila, con la función "binario" se calculará su resultado, y con la función "push", ingresará el resultado a la pila. También será mostrado por pantalla.

Si el valor de input es "OCTAL", se utilizará la función "pop" para sacar el último valor ingresado a la pila, con la función "octadecimal" se calculará su

resultado, y con la función “push”, ingresará el resultado a la pila. También será mostrado por pantalla.

Si el valor de input es “HEXAL”, se utilizará la función “pop” para sacar el último valor ingresado a la pila, con la función “Hexa” se calculará su resultado, y con la función “push”, ingresará el resultado a la pila. También será mostrado por pantalla.

Si el valor de input es “STO”, se pasará la pila a una variable auxiliar, que actuará de pila auxiliar.

Si el valor de input es “MRCL”, se imprimirá por pantalla toda la pila auxiliar, con la función “print”.

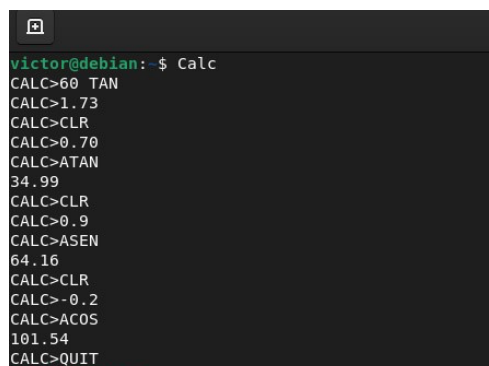
Si el valor de input es “S+”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila auxiliar, luego se sumarán, y se mostrará el resultado por pantalla con la función “print”.

Si el valor de input es “S-”, se utilizará la función “pop” para sacar los últimos dos valores ingresados a la pila auxiliar, luego se restarán, y se mostrará el resultado por pantalla con la función “print”.

Si el valor de input es “QUIT”, se detendrá la ejecución del programa.

CORRIDAS DE EJEMPLOS

Modo interactivo desde el intérprete de comandos del Sistema de Linux:



```
victor@debian:~$ Calc
CALC>60 TAN
CALC>1.73
CALC>CLR
CALC>0.70
CALC>ATAN
34.99
CALC>CLR
CALC>0.9
CALC>ASEN
64.16
CALC>CLR
CALC>-0.2
CALC>ACOS
101.54
CALC>QUIT
```



```
Actividades Terminal
victor@debian:~/Proyecto1$ cd ..
victor@debian:~$ Calc
CALC>1
CALC>2
CALC>+
3
CALC>1
CALC>-
2
CALC>15
CALC>*
30
CALC>5
CALC>/
6
CALC>2
CALC>PO
36
CALC>30
CALC>-
6
CALC>FACT
720
CALC>RA
26.83
CALC>CLR
CALC>1
CALC>RCL
1
CALC>9
CALC>+
10
CALC>LOG
2.30
CALC>2
CALC>SWAP
Cambiados con exito
CALC>DSP
2.30 2.00
CALC>DSP
2.30 2.00
CALC>SWAP
Cambiados con exito
CALC>DSP
2.00 2.30
CALC>BINARY
CALC>RCL
100
CALC>CLR
CALC>5
CALC>OCTAL
CALC>RCL
```

```
Actividades Terminal
victor@debian:~$ Calc
CALC>5
CALC>4
CALC>*
20
CALC>12
CALC>+
32
CALC>CLR
CALC>4
CALC>5
CALC>6
CALC>*
30
CALC>+
34
CALC>RCL
34
CALC>CLR
CALC>5
CALC>RCL
5
CALC>BINARY
CALC>RCL
101
CALC>QUIT
```

```
Actividades Terminal
victor@debian:~$ Calc
CALC>60 TAN
CALC>1.73
CALC>CLR
CALC>0.70
CALC>ATAN
34.99
CALC>CLR
CALC>0.9
CALC>ASEN
64.16
CALC>CLR
CALC>-0.2
CALC>ACOS
101.54
CALC>QUIT
```

Uso desde la línea de comando de Linux:

```
victor@debian:~$ Calc 1 2 +
3
victor@debian:~$ Calc 1 15 -
-14
victor@debian:~$ Calc 3 50 x
150
victor@debian:~$ Calc 1232 10 /
123.20
victor@debian:~$ Calc 4 FACT
24
victor@debian:~$ Calc 4 RA
2
victor@debian:~$ Calc 4 2 P0
16
victor@debian:~$ Calc 1223 LOG
7.11
```

```
victor@debian:~$ Calc 5 4 x
20
victor@debian:~$ Calc 5 BINARY
101
victor@debian:~$
```

```
Actividades Terminal
victor@debian:~$ Calc 15 COS
0.97
victor@debian:~$ Calc 30 SEN
0.50
victor@debian:~$ Calc 60 TAN
1.73
victor@debian:~$ Calc 0.15 ATAN
0.53
victor@debian:~$ Calc 0.55 ASEN
33.37
victor@debian:~$ Calc 1 ACOS
0
victor@debian:~$ Calc 12 BINARY
11000
victor@debian:~$ Calc 27 BINARY
11011
victor@debian:~$ Calc 15 OCTAL
17
victor@debian:~$ Calc 30 HEXAL
1E
victor@debian:~$
```

```
victor@debian:~$ Calc 1 2 +
3
victor@debian:~$ Calc 1 15 -
-14
victor@debian:~$ Calc 3 50 x
150
victor@debian:~$ Calc 1232 10 /
123.20
victor@debian:~$ Calc 4 FACT
24
victor@debian:~$ Calc 4 RA
2
victor@debian:~$ Calc 4 2 P0
16
victor@debian:~$ Calc 1223 LOG
7.11
```

PROGRAMA FUENTE

Código fuente de la librería “stack.h”:

```
typedef struct node {
    float value;
```

```
    struct node *next;  
} Node;
```

```
void push(Node **stack, float x) {  
    Node *p = (Node*)malloc(sizeof(Node));  
    p->value = x;  
    p->next = *stack;  
    *stack = p;  
}
```

```
float pop(Node **head) {  
    Node *p;  
    float v;  
    v = (*head)->value;  
    p = (*head)->next;  
    free(*head);  
    *head = p;  
    return v;  
}
```

```
//FUNCION DISPLAY//  
void display(Node *stack) {  
    for(;stack != NULL; stack = stack->next) {  
        printf("%.2f  ",stack->value);  
    }  
    printf("\n");  
}
```

```
//FUNCION CLEAR//  
Node *liberar(Node *raiz){  
    Node *reco = raiz;  
    Node *bor;  
    while (reco != NULL)  
    {  
        bor = reco;  
        reco = reco->next;  
        free(bor);  
    }  
    return reco;  
}
```

```
//FUNCION PARA CONVERTIR A RADIANTES//
```

```
float radianes(float valor){  
    const double PI=3.141592653589;  
    double rad;  
    rad=valor*PI/180;  
    return rad;  
}
```

```
//FUNCION PARA CONVERTIR DE RADIANTES A GRADOS//
```

```
float grados(float valor){  
    const double PI=3.141592653589;  
    double grad;  
    grad=valor*180/PI;  
    return grad;  
}
```

```
//FUNCION CONVERTIR A BINARIO//
```

```
float binario(float n){  
    float binary=0;  
    int n1=n;  
    int a[n1], i=0;  
  
    while (n1>0){  
        a[i]=n1%2;  
        i=i+1;  
        n1=n1/2;  
    }  
    while(0<=i){  
        if (a[i]==1 && i==0){  
            binary ++;  
        }  
        else if(a[i]==0)  
            binary*= 10;  
        else if(a[i]==1 && i !=0){  
            binary++;  
            binary*=10;  
        }  
        i--;  
    }  
    return binary;  
}
```

```
//FUNCION CONVERTIR A OCTADECIMAL//
```

```
float octadecimal(float v1){  
    float NumOctal = 0, i = 1;  
    int a= v1;  
  
    while (a != 0)  
    {  
        NumOctal += (a % 8) * i;  
        a /= 8;  
        i *= 10;  
    }  
    return NumOctal;  
}
```

```
//FUNCION CONVERTIR A HEXADECIMAL//
```

```
void Hexa(float v1) {  
    long int quotient;  
    int i=1,j,temp;  
    char hexadecimalNumber[100];  
    quotient = v1;  
  
    while(quotient!=0) {  
        temp = quotient % 16;  
        //Para convertir el entero a char  
        if ( temp < 10)  
            temp =temp + 48;  
        else  
            temp = temp + 55;  
        hexadecimalNumber[i++]= temp;  
        quotient = quotient / 16;  
    }  
    for (j = i -1 ;j> 0;j--)  
        printf("%c",hexadecimalNumber[j]);  
    printf("\n");  
}
```

```
//FUNCION IMPRIMIR LOS VALORES//
```

```
void print(int valor2, float valor){  
  
    if (valor-valor2 !=0)
```

```

        printf("%.2f\n", valor);
    else
        printf("%d\n", valor2);
}

```

Código fuente del programa "main.c":

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "stack.h"

//----PROGRAMA PRINCIPAL----//
int main(int argc, char *argv[]) {

    //Pila de numeros
    Node* stack=(Node*)malloc(sizeof(Node));

    //Modo de operacion en terminal UNIX
    float memory=0;
    int i, chara = 0;
    int flag = 0; // 0=Menu interactivo // 1=Operando desde el terminal//
    if(argc>1) {
        flag = 1;
        for(i=1;argv[i] != NULL && atof(argv[i]) != 0; i++) {
            push(&stack,atof(argv[i]));
        }
        i--;
    }

    // Bucle de input de usuario
    while(1) {
        char input[10];
        int valor2;
        float v1,v2,valor;
        if(flag==1) {
            i++;
            if(argv[i]==NULL)
                exit(0);
            strcpy(input, argv[i]);

```

```

    }
    if(flag==0) {
        printf("CALC>");
        scanf("%s", &input);
    }

```

//Redirije el flujo del codigo segun el input del usuario

```

//SUMA//
if(strcmp(input, "+") == 0) {
    v1 = pop(&stack);
    v2 = pop(&stack);
    valor = v1 + v2;
    push(&stack, valor);
    valor2=valor;
    print(valor2,valor);
}else

```

```

//RESTA//
if(strcmp(input, "-") == 0) {
    v1 = pop(&stack);
    v2 = pop(&stack);
    valor = v2 - v1;
    push(&stack, valor);
    valor2=valor;
    print(valor2,valor);
}else

```

```

//MULTIPLICACION//
if(strcmp(input, "x") == 0) {
    v1 = pop(&stack);
    v2 = pop(&stack);
    valor = v1 * v2;
    push(&stack, valor);
    valor2=valor;
    print(valor2,valor);
}else

```

```

//DIVISION//
if(strcmp(input, "/") == 0) {
    v1 = pop(&stack);

```

```

    v2 = pop(&stack);
    valor = v2 / v1;
    push(&stack, valor);
    valor2=valor;
    print(valor2,valor);
}else

//RAIZ CUADRADA//
if(strcmp(input , "RA") == 0){
    valor=sqrt(v1=pop(&stack));
    push(&stack, valor);
    valor2=valor;
    print(valor2,valor);
}else

//POTENCIA//
if(strcmp(input, "PO")==0){
    v1= pop(&stack);
    v2= pop(&stack);
    valor = pow(v2,v1);
    push(&stack, valor);
    valor2=valor;
    print(valor2,valor);
}else

//FACTORIAL//
if(strcmp(input , "FACT") == 0){
    valor=1;
    v1 = pop(&stack);
    for (; v1>1; v1--)
        valor *= v1;
    valor2=valor;
    print(valor2,valor);
    push(&stack, valor);
}else

//LOGARITMO//
if(strcmp(input , "LOG")==0){
    v1= pop(&stack);
    valor=(log(v1));
    valor2=valor;

```



```

        print(valor2,valor);
        push(&stack, valor);
    }else

//COSENO//
    if(strcmp(input , "COS")==0){
        v1=pop(&stack);
        valor=cos(radianes(v1));
        valor2=valor;
        print(valor2,valor);
        push(&stack,valor);
    }else

//SENO//
    if(strcmp(input , "SEN")==0){
        v1=pop(&stack);
        valor=sin(radianes(v1));
        valor2=valor;
        print(valor2,valor);
        push(&stack,valor);
    }else

//TANGENTE//
    if(strcmp(input , "TAN")==0){
        v1=pop(&stack);
        valor=tan(radianes(v1));
        valor2=valor;
        print(valor2,valor);
        push(&stack,valor);
    }else

//ARCOSENO//
    if(strcmp(input , "ASEN")==0){
        v1= pop(&stack);
        if ((v1>=-1) && (v1<=1)){
            valor=(grados(asin(v1)));
            valor2=valor;
            print(valor2,valor);
        } else{
            printf("ERROR DE DOMINIO");
            push(&stack, v1);}
    }

```

```
}else
```

```
//ARCOCOSENO//
```

```
if (strcmp(input , "ACOS")==0){  
    v1= pop(&stack);  
    if ((v1>=-1) && (v1<=1)){  
        valor=(grados(acos(v1)));  
        valor2=valor;  
        print(valor2,valor);  
        push(&stack, valor);  
    } else{  
        printf("ERROR DE DOMINIO");  
        push(&stack, v1);}  
}else
```

```
//ARCOTANGENTE//
```

```
if (strcmp(input , "ATAN")==0){  
    v1=pop(&stack);  
    valor=grados((atan(v1)));  
    valor2=valor;  
    print(valor2,valor);  
    push(&stack,valor);  
}else
```

```
//FUNCION DISPLAY//
```

```
if(strcmp(input, "DSP") == 0) {  
    display(stack);  
}else
```

```
//FUNCION RECALL//
```

```
if(strcmp(input , "RCL") == 0){  
    if (stack != NULL){  
        valor=pop(&stack);  
        valor2=valor;  
        print(valor2,valor);  
        push(&stack,valor);  
    }  
    else  
        printf("La lista esta vacia\n");  
} else
```

```

//FUNCION CLEAR//
if(strcmp(input, "CLR") == 0) {
    stack=liberar(stack);
} else

//FUNCION SWAP//
if (strcmp(input, "SWAP")==0){
    v1=pop(&stack);
    v2=pop(&stack);
    push(&stack,v1);
    push(&stack,v2);
    printf("Cambiados con exito\n");
}else

//FUNCION BINARIO//
if (strcmp(input, "BINARY")==0){
    v1=pop(&stack);
    v1=v1/1;
    valor=binario(v1);
    push(&stack,valor);
    if(flag == 1)
        printf("%.0f\n", valor);
}else

//FUNCION OCTADECIMAL//
if (strcmp(input, "OCTAL")==0){
    v1=pop(&stack);
    valor=octadecimal(v1);
    push(&stack,valor);
    if(flag == 1)
        printf("%.0f\n", valor);
}else

//FUNCION HEXADECIMAL//
if (strcmp(input, "HEXAL")==0){
    v1=pop(&stack);
    Hexa(v1);
    push(&stack,v1);
}else

//FUNCIÓN ALACENAR EL MEMORIA//

```

```

if(strcmp(input, "STO")==0){
    v1=pop(&stack);
    memory=v1;
    printf("Se ha realizado el almacenado con exito\n");
    push(&stack,memory);
}else

//FUNCIÓN RECORDAR VALOR DE MEMORIA//
if(strcmp(input, "MRCL")==0){
    valor2=memory;
    print(valor2,memory);
}else

//SUMAR A LA MEMORIA//
if (strcmp(input, "S+")==0){
    v1=pop(&stack);
    memory+=v1;
    valor2=memory;
    print(valor2,memory);
}else

//RESTAR A LA MEMORIA//
if (strcmp(input, "S-")==0){
    v1=pop(&stack);
    memory-=v1;
    valor2=memory;
    print(valor2,memory);
}else

//FUNCION QUIT//
if(strcmp(input, "QUIT") == 0) {
    exit(0);
}else

if(input != '\0'){
    push(&stack,atof(input));
}
}
return 0;
}

```

ENLACE AL REPOSITORIO

https://github.com/Noralgorithm/AyP2_PR0Y3CT0_1