**Sergio Moncada Muñoz**

**1089380570**

**Taller punteros**

**Programación 2**

**Grupo 4**

// Taller Apuntadores

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\* 1) Realice una función sqr(&n) que reciba una dirección a un numero entero,

eleve el numero al cuadrado y deje el resultado en el mismo número. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: own\_sqrt

\* tipo: Función

\* Objetivo: Calcular la raíz cuadrada de un número utilizando el método de aproximación iterativa.

\* Parametros:

\* - num (float\*): Puntero al número del cual se desea calcular la raíz cuadrada.

\* Retorno:

\* - float: Aproximación de la raíz cuadrada del número.

\* Ejemplo de uso:

\* Ejemplo 1:

\* float num = 16.0;

\* float resultado = own\_sqrt(&num);

\* printf("%f\n", resultado);

\* Salida esperada: 4.0

\*/

float own\_sqrt (float \*num)

{

if (\*num == 0 || \*num == 1)

{

return \*num;

}

float aprox = \*num;

float epsilon = 0.000001; // Precisión deseada

while ((aprox - \*num / aprox) > epsilon)

{

aprox = (aprox + \*num / aprox) / 2;

}

return aprox;

}

//-----------------------------------------------------------------------

/\* 2) Realice una función printstr(str) que reciba un apuntador a una cadena

de caracteres y la imprima en la salida estándar usando la función putchar y

manejo de apuntadores. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: printstr

\* tipo: Procedimiento

\* Objetivo: Imprimir una cadena de caracteres en la consola.

\* Parametros:

\* - str (char\*): Puntero a la cadena de caracteres a imprimir.

\* Retorno: void

\* Ejemplo de uso:

\* Ejemplo:

\* char cadena[] = "Sergio";

\* printstr(cadena);

\* Salida esperada: Sergio

\*/

void printstr (char \*str)

{

int i;

for (i=0; \*(str+i)!='\0'; i++)

{

putchar (\*(str+i));

}

}

//-----------------------------------------------------------------------

/\* 3) Escriba una función str\_busca(str, c) que reciba un apuntador a una

cadena de caracteres str, y un carácter c. Busque el carácter en la cadena

y regrese la posición en que se encuentra el carácter en la cadena o un -1

si no se encuentra utilizando apuntadores. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: str\_busca

\* Tipo: Función

\* Objetivo: Buscar la primera ocurrencia de un carácter en una cadena de caracteres.

\* Parámetros:

\* - str (char\*): Puntero a la cadena de caracteres en la que se realizará la búsqueda.

\* - c (char): Carácter a buscar en la cadena.

\* Retorno: El índice de la primera ocurrencia del carácter en la cadena. Si no se encuentra, retorna -1.

\* Ejemplo de uso:

\* char cadena[] = "Hola!";

\* int indice = str\_busca(cadena, 'l');

\* // indice tendrá el valor de 2, que es la posición de la primera 'l' en la cadena.

\*/

int str\_busca (char \*str, char c)

{

int i, cont=0;

for (i=0; \*(str+i)!='\0'; i++)

{

if (\*(str+i)==c)

{

cont++;

return i;

break;

}

}

if (cont==0)

{

return (-1);

}

}

//-----------------------------------------------------------------------

/\* 4) Realice una función concatena(str1, str2)que agregue la cadena de

caracteres str2 al final de la cadena str1. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: concatena

\* Tipo: Procedimiento (void)

\* Objetivo: Concatenar dos cadenas de caracteres.

\* Parámetros:

\* - str1 (char\*): Puntero a la primera cadena de caracteres, que será modificada para contener la concatenación.

\* - str2 (char\*): Puntero a la segunda cadena de caracteres, que se agregará al final de la primera cadena.

\* Retorno: void (ningún valor de retorno)

\* Ejemplo de uso:

\* char cadena1[20] = "Hola";

\* char cadena2[] = " mundo!";

\* concatena(cadena1, cadena2);

\* // cadena1 contendrá "Hola mundo!"

\*/

void concatena (char \*str1, char \*str2)

{

int i, t1, t2;

t1=strlen(str1);

t2=strlen(str2);

for (i=0; i<t2; i++)

{

\*(str1+t1+i)=\*(str2+i);

}

printf ("%s\n\n", str1);

}

//-----------------------------------------------------------------------

/\* 5) Realice una función reversa(str) que reciba un apuntador a una

cadena de caracteres e invierta la cadena. Utilizando apuntadores. \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: reversa

\* Tipo: Procedimiento (void)

\* Objetivo: Invertir una cadena de caracteres.

\* Parámetros:

\* - str (char\*): Puntero a la cadena de caracteres que se invertirá.

\* Retorno: void (ningún valor de retorno)

\* Ejemplo de uso:

\* char cadena[] = "Hola";

\* reversa(cadena);

\* // cadena contendrá "aloH"

\*/

void reversa (char \*str)

{

int i, t;

char l;

t=strlen (str);

for (i=0; i<(t/2); i++)

{

l=\*(str+i);

\*(str+i)=\*(str+t-i-1);

\*(str+t-i-1)=l;

}

printf ("%s\n\n", str);

}

//-----------------------------------------------------------------------

/\* 6) Realice un programa que:

a) Pida n números enteros y los almacene en un arreglo de n elementos

b) Busque el número menor y el numero mayor

c) Despliegue el número menor y el numero mayor \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: mayormenor

\* Tipo: Procedimiento (void)

\* Objetivo: Encontrar el número mayor y menor en un arreglo de enteros.

\* Parámetros:

\* - n (int): Número de elementos en el arreglo.

\* Retorno: void (ningún valor de retorno)

\* Ejemplo de uso:

\* int cantidad = 5;

\* mayormenor(cantidad);

\* // Solicitará al usuario ingresar 5 números, y luego mostrará el número mayor y menor.

\* Notas: Esta función utiliza memoria dinámica para almacenar el arreglo de enteros.

\*/

void mayormenor (int n)

{

int i, \*v, mayor, menor;

v=(int\*)malloc(sizeof(int) \* n);

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

{

printf ("Ingrese un numero: ");

scanf ("%d", v+i);

}

putchar ('[');

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

{

printf ("%d ", \*(v+i));

}

printf ("]\n");

mayor=\*v;

menor=\*v;

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

{

if (\*(v+i)>mayor)

{

mayor=\*(v+i);

}

if (\*(v+i)<menor)

{

menor=\*(v+i);

}

}

printf ("El numero mayor es %d y el menor es %d\n\n", mayor, menor);

free (v);

}

//-----------------------------------------------------------------------

/\* 7) Hacer un programa que:

a) Pida n números enteros y los almacene en un arreglo de n elementos

b) Los ordene de menor a mayor

c) Despliegue el arreglo \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: orden

\* Tipo: Procedimiento (void)

\* Objetivo: Ordenar un arreglo de enteros de forma ascendente.

\* Parámetros:

\* - n (int): Número de elementos en el arreglo.

\* Retorno: void (ningún valor de retorno)

\* Ejemplo de uso:

\* int cantidad = 5;

\* orden(cantidad);

\* // Solicitará al usuario ingresar 5 números, y luego mostrará el arreglo ordenado.

\* Notas: Esta función utiliza memoria dinámica para almacenar el arreglo de enteros.

\* Asegúrate de liberar la memoria utilizando la función free después de utilizar esta función.

\*/

void orden (int n)

{

int i, j, \*v, menor, aux, cont;

v = (int\*)malloc(sizeof(int) \* n);

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

{

printf ("Ingrese un numero: ");

scanf ("%d", v+i);

}

putchar ('[');

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

{

printf ("%d ", \*(v+i));

}

printf ("]\n");

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

{

menor = \*(v+i);

cont=i;

for (j=i; j<n; j++)

{

if (menor>\*(v+j))

{

menor=\*(v+j);

cont=j;

}

}

aux=\*(v+i);

\*(v+i)=menor;

\*(v+cont)=aux;

}

putchar ('[');

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

{

printf ("%d ", \*(v+i));

}

printf ("]\n");

free (v);

}

//-----------------------------------------------------------------------

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Nombre de la subrutina: menu

\* Tipo: void

\* Objetivo: Mostrar un menú de opciones y ejecutar las funciones correspondientes según la opción seleccionada por el usuario.

\* Parámetros: Ninguno.

\* Retorno: Ninguno.

\* Ejemplo de uso:

\* menu();

\* Notas:

\* - La función `menu` muestra un menú en la consola con diferentes opciones numéricas.

\* - El usuario puede seleccionar una opción ingresando el número correspondiente.

\* - Cada opción ejecuta una función específica y luego vuelve a mostrar el menú.

\* - Las funciones asociadas a cada opción del menú son:

\* - Opción 1: Calcula la raíz cuadrada de un número.

\* - Opción 2: Imprime una palabra.

\* - Opción 3: Busca la posición de una letra en una palabra.

\* - Opción 4: Concatena dos cadenas de texto.

\* - Opción 5: Revierte una cadena de caracteres.

\* - Opción 6: Encuentra el número mayor y menor en un vector.

\* - Opción 7: Ordena un vector de menor a mayor.

\* - Opción 8: Sale del programa.

\* - Después de ejecutar una opción, se muestra un mensaje y se pausa la ejecución antes de volver a mostrar el menú.

\*/

void menu ()

{

int op;

system ("cls");

printf ("Ingrese 1 para calcular la raiz cuadrada de un numero\n");

printf ("Ingrese 2 para imprimir una palabra\n");

printf ("Ingrese 3 para buscar la posicion de una letra en una palabra\n");

printf ("Ingrese 4 para concatenar dos cadenas de texto\n");

printf ("Ingrese 5 para revertir una cadena de caracteres\n");

printf ("Ingrese 6 para saber el numero mayor y menor de un vector\n");

printf ("Ingrese 7 para ordenar un vector de menor a mayor\n");

printf ("Ingrese 8 para salir\n");

printf ("Ingrese la opcion que desea (1-8): ");

scanf ("%d", &op);

if (op==1)

{

float x, y;

printf ("Ingrese un numero: ");

scanf ("%f", &x);

y=own\_sqrt (&x);

printf ("La raiz cuadrada de %.2f es %.2f\n\n", x, y);

system ("pause");

menu ();

}

if (op==2)

{

char str [5];

printf ("Ingrese una palabra: ");

fflush (stdin);

scanf ("%s", &str);

printstr (str);

putchar ('\n');

putchar ('\n');

system ("pause");

menu ();

}

if (op==3)

{

int n;

char str [5];

printf ("Ingrese una palabra: ");

fflush (stdin);

scanf ("%s", &str);

char l;

printf ("Ingrese una letra: ");

fflush (stdin);

scanf ("%c", &l);

n=str\_busca (str, l);

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

system ("pause");

menu ();

}

if (op==4)

{

char str1 [5], str2 [5];

printf ("Ingrese la primera palabra: ");

fflush (stdin);

scanf ("%s", &str1);

printf ("Ingrese la segunda palabra: ");

fflush (stdin);

scanf ("%s", &str2);

concatena (str1, str2);

system ("pause");

menu ();

}

if (op==5)

{

char str[5];

printf ("Ingrese una cadena de caracteres: ");

fflush (stdin);

scanf ("%s", &str);

reversa (str);

system ("pause");

menu ();

}

if (op==6)

{

int n;

printf ("Ingrese la cantidad de numeros del vector: ");

scanf ("%d", &n);

mayormenor (n);

system ("pause");

menu ();

}

if (op==7)

{

int n;

printf ("Ingrese la cantidad de numeros del vector: ");

scanf ("%d", &n);

orden (n);

system ("pause");

menu ();

}

}

//------------------------------------------------------------------------

main ()

{

menu ();

}