**Sergio Moncada Muñoz**

**1089380570**

**Programación 2 – Grupo 4**

**Proyecto final**

//Proyecto final

//Sergio Moncada Muñoz - 1089380570

//Programacion 2 - Grupo 4

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#include <stdbool.h>

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

//Estructura Recibos

struct Recibo

{

long valor;

char numero [5], fecha[11], nombre[40], ciudad[25], concepto[50], numero\_en\_letras [100];

};

struct Recibo recibos [50];

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

//Variables globales

int numero\_recibos;

char linea[150];

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

/\*

Nombre subrutina: gotoxy

Tipo: Procedimiento (void)

Objetivo: Mover el cursor de la consola a una posición específica en las coordenadas (x, y).

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) a la que se desea mover el cursor.

posicion\_y: Entero que indica la posición vertical (coordenada y) a la que se desea mover el cursor.

Retorno: Ninguno.

Ejemplo de uso:

int x = 10;

int y = 5;

gotoxy(x, y);

Notas: Esta función utiliza funciones de la biblioteca de Windows para manipular la posición del cursor

en la consola. Asegúrate de compilar y ejecutar el código en un entorno que admita estas funciones, como

Microsoft Visual Studio en Windows.

\*/

void gotoxy(int posicion\_x, int posicion\_y){

COORD coord;

coord.X = posicion\_x;

coord.Y = posicion\_y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

}

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

/\*

Nombre subrutina: esquina\_1

Tipo: void

Objetivo: Dibujar la esquina superior izquierda en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) de la esquina.

posicion\_y: Entero que indica la posición vertical (coordenada y) de la esquina.

Retorno: Ninguno.

Ejemplo de uso:

int x = 10;

int y = 5;

esquina\_1(x, y);

Notas: Esta función utiliza la función "gotoxy" para mover el cursor a la posición especificada y

luego imprime el carácter ASCII 218, que representa la esquina superior izquierda en la tabla de

caracteres.

\*/

void esquina\_1 (int posicion\_x, int posicion\_y)

{

gotoxy (posicion\_x, posicion\_y);

printf ("%c", 218);

}

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

/\*

Nombre subrutina: esquina\_2

Tipo: void

Objetivo: Dibujar la esquina superior derecha en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) de la esquina.

posicion\_y: Entero que indica la posición vertical (coordenada y) de la esquina.

Retorno: Ninguno.

Ejemplo de uso:

int x = 20;

int y = 5;

esquina\_2(x, y);

Notas: Esta función utiliza la función "gotoxy" para mover el cursor a la posición especificada y

luego imprime el carácter ASCII 191, que representa la esquina superior derecha en la tabla de

caracteres.

\*/

void esquina\_2 (int posicion\_x, int posicion\_y)

{

gotoxy (posicion\_x, posicion\_y);

printf ("%c", 191);

}

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

/\*

Nombre subrutina: esquina\_3

Tipo: void

Objetivo: Dibujar la esquina inferior izquierda en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) de la esquina.

posicion\_y: Entero que indica la posición vertical (coordenada y) de la esquina.

Retorno: Ninguno.

Ejemplo de uso:

int x = 30;

int y = 10;

esquina\_3(x, y);

Notas: Esta función utiliza la función "gotoxy" para mover el cursor a la posición especificada y

luego imprime el carácter ASCII 192, que representa la esquina inferior izquierda en la tabla de

caracteres.

\*/

void esquina\_3 (int posicion\_x, int posicion\_y)

{

gotoxy (posicion\_x, posicion\_y);

printf ("%c", 192);

}

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

/\*

Nombre subrutina: esquina\_4

Tipo: void

Objetivo: Dibujar la esquina inferior derecha en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) de la esquina.

posicion\_y: Entero que indica la posición vertical (coordenada y) de la esquina.

Retorno: Ninguno.

Ejemplo de uso:

int x = 40;

int y = 15;

esquina\_4(x, y);

Notas: Esta función utiliza la función "gotoxy" para mover el cursor a la posición especificada y

luego imprime el carácter ASCII 217, que representa la esquina inferior derecha en la tabla de

caracteres.

\*/

void esquina\_4 (int posicion\_x, int posicion\_y)

{

gotoxy (posicion\_x, posicion\_y);

printf ("%c", 217);

}

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

/\*

Nombre subrutina: linea\_horizontal

Tipo: void

Objetivo: Dibujar una línea horizontal en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) de inicio de la línea.

posicion\_y: Entero que indica la posición vertical (coordenada y) de la línea.

longitud: Entero que indica la longitud de la línea horizontal.

Retorno: Ninguno.

Ejemplo de uso:

int x = 5;

int y = 10;

int longitud = 15;

linea\_horizontal(x, y, longitud);

Notas: Esta función utiliza la función "gotoxy" para mover el cursor a la posición especificada y

luego imprime el carácter ASCII 196 repetidamente para dibujar la línea horizontal. La línea se

extenderá desde la posición indicada hasta la posición indicada más la longitud especificada.

\*/

void linea\_horizontal (int posicion\_x, int posicion\_y, int longitud)

{

int i;

i=posicion\_x;

while (i<=posicion\_x+longitud)

{

gotoxy(i, posicion\_y);

printf("%c",196);

i++;

}

}

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

/\*

Nombre subrutina: linea\_vertical

Tipo: void

Objetivo: Dibujar una línea vertical en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) de la línea.

posicion\_y: Entero que indica la posición vertical (coordenada y) de inicio de la línea.

longitud: Entero que indica la longitud de la línea vertical.

Retorno: Ninguno.

Ejemplo de uso:

int x = 10;

int y = 5;

int longitud = 10;

linea\_vertical(x, y, longitud);

Notas: Esta función utiliza la función "gotoxy" para mover el cursor a la posición especificada y luego imprime el carácter ASCII 179 repetidamente para dibujar la línea vertical. La línea se extenderá desde la posición indicada hasta la posición indicada más la longitud especificada.

\*/

void linea\_vertical (int posicion\_x, int posicion\_y, int longitud)

{

int i;

i=posicion\_y;

while (i<=posicion\_y+longitud)

{

gotoxy(posicion\_x, i);

printf("%c",179);

i++;

}

}

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

/\*

Nombre subrutina: cuadro

Tipo: void

Objetivo: Dibujar un cuadro en la consola delimitado por las coordenadas de inicio y fin.

Parámetros:

x\_inicial: Entero que indica la posición horizontal (coordenada x) de inicio del cuadro.

y\_inicial: Entero que indica la posición vertical (coordenada y) de inicio del cuadro.

x\_final: Entero que indica la posición horizontal (coordenada x) de fin del cuadro.

y\_final: Entero que indica la posición vertical (coordenada y) de fin del cuadro.

Retorno: Ninguno.

Ejemplo de uso:

int xInicial = 5;

int yInicial = 5;

int xFinal = 15;

int yFinal = 10;

cuadro(xInicial, yInicial, xFinal, yFinal);

Notas: Esta función utiliza las funciones "linea\_horizontal", "linea\_vertical", "esquina\_1",

"esquina\_2", "esquina\_3" y "esquina\_4" para dibujar un cuadro en la consola. Se dibujan las

líneas horizontales y verticales en las posiciones correspondientes, y se dibujan las esquinas

utilizando los caracteres ASCII 218, 191, 192 y 217. Los parámetros x\_inicial, y\_inicial,

x\_final e y\_final definen las coordenadas de inicio y fin del cuadro.

\*/

void cuadro (int x\_inicial, int y\_inicial, int x\_final, int y\_final)

{

linea\_horizontal (x\_inicial+1, y\_inicial, x\_final-x\_inicial-2);

linea\_horizontal (x\_inicial+1, y\_final, x\_final-x\_inicial-2);

linea\_vertical (x\_inicial, y\_inicial+1, y\_final-y\_inicial-2);

linea\_vertical (x\_final, y\_inicial+1, y\_final-y\_inicial-2);

esquina\_1 (x\_inicial, y\_inicial);

esquina\_2 (x\_final, y\_inicial);

esquina\_3 (x\_inicial, y\_final);

esquina\_4 (x\_final, y\_final);

}

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

/\*

Nombre subrutina: estilo\_recibos

Tipo: void

Objetivo: Dibujar un estilo de recibo en una posición específica de la consola.

Parámetros:

posicion\_x: Entero que indica la posición horizontal (coordenada x) del recibo.

posicion\_y: Entero que indica la posición vertical (coordenada y) del recibo.

Retorno: Ninguno.

Ejemplo de uso:

int x = 10;

int y = 5;

estilo\_recibos(x, y);

Notas: Esta función utiliza las funciones "cuadro", "linea\_horizontal", "linea\_vertical",

"esquina\_1" y "esquina\_3" para dibujar un estilo de recibo en la consola. Se dibuja un

cuadro en la posición indicada y se añaden líneas horizontales y verticales para dividir

el recibo en secciones. Se colocan etiquetas para indicar los campos correspondientes

dentro del recibo.

\*/

void estilo\_recibos (int posicion\_x, int posicion\_y)

{

int i;

cuadro (posicion\_x, posicion\_y, posicion\_x+55, posicion\_y+21);

for (i=1; i<7; i++)

{

linea\_horizontal (posicion\_x+1, posicion\_y+(3\*i), 53);

}

linea\_vertical (posicion\_x+26, posicion\_y+1, 1);

esquina\_1 (posicion\_x+26, posicion\_y);

esquina\_3 (posicion\_x+26, posicion\_y+3);

linea\_vertical (posicion\_x+37, posicion\_y+1, 4);

esquina\_1 (posicion\_x+37, posicion\_y);

esquina\_3 (posicion\_x+37, posicion\_y+6);

gotoxy (posicion\_x+1, posicion\_y+1);

printf ("Ciudad");

gotoxy (posicion\_x+27, posicion\_y+1);

printf ("DD-MM-AAAA");

gotoxy (posicion\_x+29, posicion\_y+2);

printf ("Fecha");

gotoxy (posicion\_x+38, posicion\_y+1);

printf ("No. XXXX");

gotoxy (posicion\_x+38, posicion\_y+4);

printf ("Valor $");

gotoxy (posicion\_x+1, posicion\_y+4);

printf ("Pagado a");

gotoxy (posicion\_x+1, posicion\_y+7);

printf ("Concepto");

gotoxy (posicion\_x+1, posicion\_y+16);

printf ("Valor (en letras)");

}

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

/\*

Nombre subrutina: ubicacion\_datos

Tipo: void

Objetivo: Mostrar los datos de un recibo en una ubicación específica de la consola.

Parámetros:

recibos: Arreglo de estructuras Recibo que contiene los datos de los recibos.

posicion\_x: Entero que indica la posición horizontal (coordenada x) de los datos del recibo.

posicion\_y: Entero que indica la posición vertical (coordenada y) de los datos del recibo.

numero\_estructura: Entero que indica el índice de la estructura de recibo a mostrar.

Retorno: Ninguno.

Ejemplo de uso:

struct Recibo recibos[10]; // Suponiendo un arreglo de estructuras de recibo

int x = 10;

int y = 5;

int numeroEstructura = 0;

ubicacion\_datos(recibos, x, y, numeroEstructura);

Notas: Esta función utiliza la función "gotoxy" para posicionar el cursor en las ubicaciones

específicas de la consola y muestra los datos correspondientes de la estructura de recibo

indicada. Se muestra el número, la fecha, el nombre, el valor, la ciudad, el concepto y el valor

en letras del recibo en las ubicaciones especificadas.

\*/

void ubicacion\_datos (struct Recibo recibos[], int posicion\_x, int posicion\_y, int numero\_estructura)

{

gotoxy (posicion\_x+42, posicion\_y+1);

printf ("%s\n", recibos[numero\_estructura].numero);

gotoxy (posicion\_x+27, posicion\_y+1);

printf ("%s\n", recibos[numero\_estructura].fecha);

gotoxy (posicion\_x+10, posicion\_y+4);

printf ("%s\n", recibos[numero\_estructura].nombre);

gotoxy (posicion\_x+45, posicion\_y+4);

printf ("%ld\n", recibos[numero\_estructura].valor);

gotoxy (posicion\_x+8, posicion\_y+1);

printf ("%s\n", recibos[numero\_estructura].ciudad);

gotoxy (posicion\_x+10, posicion\_y+7);

printf ("%s\n", recibos[numero\_estructura].concepto);

if (strlen (recibos[numero\_estructura].numero\_en\_letras) <= 53)

{

gotoxy (posicion\_x+1, posicion\_y+17);

printf ("%s", recibos[numero\_estructura].numero\_en\_letras);

}

else if (strlen (recibos[numero\_estructura].numero\_en\_letras) > 53)

{

int i, ubicacion\_espacio;

for (i=53; i>0; i--)

{

if (recibos[numero\_estructura].numero\_en\_letras[i] == ' ')

{

ubicacion\_espacio = i;

break;

}

}

for (i=0; i<ubicacion\_espacio; i++)

{

gotoxy (posicion\_x+1+i, posicion\_y+17);

printf ("%c", recibos[numero\_estructura].numero\_en\_letras [i]);

}

for (i=ubicacion\_espacio+1; i<strlen(recibos[numero\_estructura].numero\_en\_letras); i++)

{

gotoxy (posicion\_x+i-ubicacion\_espacio, posicion\_y+19);

printf ("%c", recibos[numero\_estructura].numero\_en\_letras [i]);

}

}

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

}

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

/\*

Nombre subrutina: estilo\_menu

Tipo: void

Objetivo: Dibujar un estilo de menú en la parte superior de la consola.

Parámetros: Ninguno.

Retorno: Ninguno.

Ejemplo de uso:

estilo\_menu();

Notas: Esta función utiliza las funciones "cuadro" y "linea\_horizontal" para dibujar un estilo

de menú en la parte superior de la consola. Se dibuja un cuadro con dimensiones predefinidas y

se añade una línea horizontal para separar el título del menú del resto de la consola.

\*/

void estilo\_menu ()

{

cuadro (0, 0, 33, 9);

linea\_horizontal (1, 2, 31);

}

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

/\*

Nombre subrutina: estilo\_lista

Tipo: void

Objetivo: Dibujar un estilo de lista en la consola.

Parámetros: Ninguno.

Retorno: Ninguno.

Ejemplo de uso:

estilo\_lista();

Notas: Esta función utiliza las funciones "cuadro" y "linea\_horizontal" para dibujar un estilo

de lista en la consola. Se dibuja un cuadro con dimensiones predefinidas y se añade una línea

horizontal para separar la lista del resto de la consola. El número de filas en la lista se

determina por la variable "numero\_recibos".

\*/

void estilo\_lista ()

{

cuadro (0, 0, 103, numero\_recibos+3);

linea\_horizontal (1, 2, 101);

}

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

/\*

Nombre subrutina: escribir\_numero\_en\_letras

Tipo: void

Objetivo: Convertir un número entero en su representación en letras.

Parámetros:

numero: Número entero a convertir.

resultado: Puntero a un arreglo de caracteres donde se almacenará el resultado.

tamano: Entero que indica el tamaño máximo del arreglo "resultado".

Retorno: Ninguno.

Ejemplo de uso:

long num = 12345;

char resultado[100];

escribir\_numero\_en\_letras(num, resultado, sizeof(resultado));

Notas: Esta función recibe un número entero y lo convierte en su representación en letras en

idioma español. El resultado se almacena en el arreglo de caracteres "resultado". El tamaño

máximo del arreglo se especifica en el parámetro "tamano". En caso de que el número esté fuera

del rango válido (0 a 999,999,999), se almacenará un mensaje indicando que el número está fuera

de rango en el arreglo "resultado".

\*/

void escribir\_numero\_en\_letras(long numero, char\* resultado, int tamano) {

const char\* unidades[] = {"cero", "uno", "dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho", "nueve"};

const char\* especiales[] = {"diez", "once", "doce", "trece", "catorce", "quince", "dieciséis", "diecisiete", "dieciocho", "diecinueve"};

const char\* decenas[] = {"", "", "veinte", "treinta", "cuarenta", "cincuenta", "sesenta", "setenta", "ochenta", "noventa"};

const char\* centenas[] = {"", "ciento", "doscientos", "trescientos", "cuatrocientos", "quinientos", "seiscientos", "setecientos", "ochocientos", "novecientos"};

const char\* miles[] = {"", "mil", "millon", "millones"};

if (numero < 0 || numero > 999999999) {

strncpy(resultado, "Numero fuera de rango", tamano);

return;

}

if (numero < 10) {

strncpy(resultado, unidades[numero], tamano);

} else if (numero < 20) {

strncpy(resultado, especiales[numero - 10], tamano);

} else if (numero < 100) {

int decena = numero / 10;

int unidad = numero % 10;

if (unidad == 0) {

strncpy(resultado, decenas[decena], tamano);

} else {

snprintf(resultado, tamano, "%s y %s", decenas[decena], unidades[unidad]);

}

} else if (numero < 1000) {

int centena = numero / 100;

int decena = (numero % 100) / 10;

int unidad = numero % 10;

if (decena == 0 && unidad == 0) {

strncpy(resultado, centenas[centena], tamano);

} else {

char temp\_resultado[100];

escribir\_numero\_en\_letras(numero % 100, temp\_resultado, sizeof(temp\_resultado));

snprintf(resultado, tamano, "%s %s", centenas[centena], temp\_resultado);

}

} else if (numero < 1000000) {

int mil = numero / 1000;

int centena = (numero % 1000) / 100;

int decena = (numero % 100) / 10;

int unidad = numero % 10;

if (mil == 1 && centena == 0 && decena == 0 && unidad == 0) {

strncpy(resultado, "mil", tamano);

} else {

char temp\_resultado1[100];

char temp\_resultado2[100];

escribir\_numero\_en\_letras(mil, temp\_resultado1, sizeof(temp\_resultado1));

if (numero % 1000 == 0) {

snprintf(resultado, tamano, "%s mil", temp\_resultado1);

} else {

escribir\_numero\_en\_letras(numero % 1000, temp\_resultado2, sizeof(temp\_resultado2));

snprintf(resultado, tamano, "%s mil %s", temp\_resultado1, temp\_resultado2);

}

}

} else if (numero < 1000000000) {

int millon = numero / 1000000;

int mil = (numero % 1000000) / 1000;

int centena = (numero % 1000) / 100;

int decena = (numero % 100) / 10;

int unidad = numero % 10;

if (millon == 1 && mil == 0 && centena == 0 && decena == 0 && unidad == 0) {

strncpy(resultado, "un millon", tamano);

} else {

char temp\_resultado1[100];

char temp\_resultado2[100];

escribir\_numero\_en\_letras(millon, temp\_resultado1, sizeof(temp\_resultado1));

if (numero % 1000000 == 0) {

snprintf(resultado, tamano, "%s millones", temp\_resultado1);

} else {

escribir\_numero\_en\_letras(numero % 1000000, temp\_resultado2, sizeof(temp\_resultado2));

snprintf(resultado, tamano, "%s millones %s", temp\_resultado1, temp\_resultado2);

}

}

} else {

strncpy(resultado, "Numero fuera de rango", tamano);

}

}

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

/\*

Nombre subrutina: leer\_archivo

Tipo: void

Objetivo: Leer un archivo de texto que contiene información de recibos y almacenar los datos en

una estructura.

Parámetros: Ninguno.

Retorno: Ninguno.

Ejemplo de uso: leer\_archivo();

Notas: Esta función lee el archivo de texto "Recibos.txt" y extrae la información de cada línea

para almacenarla en la estructura "recibos". El archivo debe tener un formato específico, donde

cada línea representa un recibo y los datos están separados por el delimitador ";". Los datos se

asignan a los campos correspondientes en la estructura "recibos". Además, se utiliza la función

"escribir\_numero\_en\_letras" para convertir el valor del recibo en letras y almacenarlo en el campo

"numero\_en\_letras" de la estructura. Si el archivo no se puede abrir, se muestra un mensaje de error.

\*/

void leer\_archivo ()

{

FILE \*archivo = fopen ("Recibos.txt", "r");

if (archivo == NULL)

{

printf ("No se pudo abrir el archivo\n");

}

else

{

numero\_recibos=0;

while (fgets (linea, 100, archivo) != NULL)

{

char \*delimitador;

delimitador = strtok (linea, ";");

strcpy (recibos[numero\_recibos].numero, delimitador);

delimitador=strtok(NULL, ";");

strcpy (recibos[numero\_recibos].fecha, delimitador);

delimitador=strtok(NULL, ";");

strcpy (recibos[numero\_recibos].nombre, delimitador);

delimitador=strtok(NULL, ";");

recibos[numero\_recibos].valor = atol(delimitador);

delimitador=strtok(NULL, ";");

strcpy (recibos[numero\_recibos].ciudad, delimitador);

delimitador=strtok(NULL, ";");

strcpy (recibos[numero\_recibos].concepto, delimitador);

escribir\_numero\_en\_letras (recibos[numero\_recibos].valor, recibos[numero\_recibos].numero\_en\_letras, sizeof(recibos[numero\_recibos].numero\_en\_letras));

numero\_recibos++;

}

fclose (archivo);

}

}

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

/\*

Nombre subrutina: fecha\_valida

Tipo: bool

Objetivo: Verificar si una fecha es válida.

Parámetros:

- fecha: cadena de caracteres que representa la fecha en formato "DD-MM-AAAA".

Retorno: Valor booleano que indica si la fecha es válida (true) o no (false).

Ejemplo de uso: bool es\_valida = fecha\_valida("25-06-2023");

Notas: Esta función verifica si una fecha es válida comprobando que el día, mes y año se encuentren

dentro de los rangos permitidos. Además, se tienen en cuenta las reglas de los meses con 30 o 31 días,

y la verificación de los años bisiestos en el mes de febrero. Si la fecha es válida, la función

devuelve true; de lo contrario, devuelve false.

\*/

bool fecha\_valida (const char fecha [])

{

int dia = (fecha[0] - '0') \* 10 + (fecha[1] - '0');

int mes = (fecha[3] - '0') \* 10 + (fecha[4] - '0');

int year = (fecha[6] - '0') \* 1000 + (fecha[7] - '0') \* 100 + (fecha[8] - '0') \* 10 + (fecha[9] - '0');

if (dia < 1 || dia > 31 || mes < 1 || mes > 12 || year < 0 || year >2023) {

return false;

}

if ((mes == 4 || mes == 6 || mes == 9 || mes == 11) && dia > 30) {

return false;

}

if (mes == 2) {

bool bisiesto = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

if (bisiesto && dia > 29) {

return false;

} else if (!bisiesto && dia > 28) {

return false;

}

}

return true;

}

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

/\*

Nombre subrutina: comparar\_numero

Tipo: int

Objetivo: Comparar un número ingresado con los números de los recibos existentes.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

- numero\_ingresado: número ingresado a comparar.

Retorno:

- Entero que representa la posición del recibo en el arreglo recibos si se encuentra una coincidencia.

- -1 si no se encuentra ninguna coincidencia.

Ejemplo de uso: int posicion = comparar\_numero(recibos, numero\_ingresado);

Notas: Esta función recorre el arreglo de recibos y compara el número ingresado con los números de los

recibos existentes. Si encuentra una coincidencia, retorna la posición del recibo en el arreglo. En caso

contrario, retorna -1.

\*/

int comparar\_numero (struct Recibo recibos [], char numero\_ingresado[5])

{

int i;

for (i=0; i<numero\_recibos; i++)

{

if (strcmp (numero\_ingresado, recibos[i].numero) == 0)

{

return i;

}

}

return -1;

}

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

/\*

Nombre subrutina: asignar\_datos

Tipo: void

Objetivo: Asignar datos a una estructura de tipo Recibo.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

- numero\_a\_asignar: entero que indica la posición en el arreglo donde se asignarán los datos.

- posicion\_x: entero que indica la posición horizontal en la pantalla donde se mostrarán los campos.

- posicion\_y: entero que indica la posición vertical en la pantalla donde se mostrarán los campos.

Retorno: Ninguno.

Ejemplo de uso: asignar\_datos(recibos, 0, 10, 20);

Notas: Esta función permite ingresar datos para asignarlos a una estructura de tipo Recibo en la posición

especificada del arreglo. Los campos que se solicitan son: número, fecha, nombre, valor, ciudad y concepto.

Además, se verifica que la fecha ingresada sea válida utilizando la función fecha\_valida. La función también

utiliza la función escribir\_numero\_en\_letras para convertir el valor numérico a su representación en letras.

\*/

void asignar\_datos (struct Recibo recibos [], int numero\_a\_asignar, int posicion\_x, int posicion\_y)

{

if (numero\_a\_asignar == numero\_recibos)

{

do

{

gotoxy (posicion\_x+42, posicion\_y+1);

printf ("XXXX");

gotoxy (posicion\_x+42, posicion\_y+1);

printf ("");

fflush (stdin);

gets (recibos[numero\_a\_asignar].numero);

if (strlen(recibos[numero\_a\_asignar].numero)>4)

{

recibos[numero\_a\_asignar].numero [4] = '\0';

}

gotoxy (posicion\_x+1, posicion\_y+22);

printf ("El numero que ingreso ya se encuentra registrado");

}while (comparar\_numero(recibos, recibos[numero\_a\_asignar].numero) != -1);

gotoxy (posicion\_x+1, posicion\_y+22);

printf (" ");

}

else if (numero\_a\_asignar<numero\_recibos)

{

char numero\_actual [5];

strcpy (numero\_actual, recibos[numero\_a\_asignar].numero);

do

{

gotoxy (posicion\_x+42, posicion\_y+1);

printf ("XXXX");

gotoxy (posicion\_x+42, posicion\_y+1);

printf ("");

fflush (stdin);

gets (recibos[numero\_a\_asignar].numero);

if (strlen(recibos[numero\_a\_asignar].numero)>4)

{

recibos[numero\_a\_asignar].numero [4] = '\0';

}

if (strcmp (numero\_actual, recibos[numero\_a\_asignar].numero) == 0)

{

break;

}

gotoxy (posicion\_x+1, posicion\_y+22);

printf ("El numero que ingreso ya se encuentra registrado");

}while (comparar\_numero(recibos, recibos[numero\_a\_asignar].numero) != -1);

gotoxy (posicion\_x+1, posicion\_y+22);

printf (" ");

}

do

{

gotoxy (posicion\_x+27, posicion\_y+1);

printf ("DD-MM-AAAA");

gotoxy (posicion\_x+27, posicion\_y+1);

printf ("");

fflush (stdin);

gets (recibos[numero\_a\_asignar].fecha);

if (strlen(recibos[numero\_a\_asignar].fecha)>10)

{

recibos[numero\_a\_asignar].fecha [10] = '\0';

}

if (!fecha\_valida (recibos[numero\_a\_asignar].fecha))

{

gotoxy (posicion\_x+1, posicion\_y+22);

printf ("La fecha que ingreso no es valida (DD-MM-AAAA)");

}

}while (!fecha\_valida (recibos[numero\_a\_asignar].fecha));

gotoxy (posicion\_x+1, posicion\_y+22);

printf (" ");

gotoxy (posicion\_x+10, posicion\_y+4);

printf ("");

fflush (stdin);

gets (recibos[numero\_a\_asignar].nombre);

if (strlen(recibos[numero\_a\_asignar].nombre)>28)

{

recibos[numero\_a\_asignar].nombre [28] = '\0';

}

gotoxy (posicion\_x+45, posicion\_y+4);

printf ("");

scanf ("%ld", &recibos[numero\_a\_asignar].valor);

escribir\_numero\_en\_letras (recibos[numero\_a\_asignar].valor, recibos[numero\_a\_asignar].numero\_en\_letras, sizeof(recibos[numero\_a\_asignar].numero\_en\_letras));

if (strlen (recibos[numero\_a\_asignar].numero\_en\_letras) <= 53)

{

gotoxy (posicion\_x+1, posicion\_y+17);

printf ("%s", recibos[numero\_a\_asignar].numero\_en\_letras);

}

else if (strlen (recibos[numero\_a\_asignar].numero\_en\_letras) > 53)

{

int i, ubicacion\_espacio;

for (i=53; i>0; i--)

{

if (recibos[numero\_a\_asignar].numero\_en\_letras[i] == ' ')

{

ubicacion\_espacio = i;

break;

}

}

for (i=0; i<ubicacion\_espacio; i++)

{

gotoxy (posicion\_x+1+i, posicion\_y+17);

printf ("%c", recibos[numero\_a\_asignar].numero\_en\_letras [i]);

}

for (i=ubicacion\_espacio+1; i<strlen(recibos[numero\_a\_asignar].numero\_en\_letras); i++)

{

gotoxy (posicion\_x+i-ubicacion\_espacio, posicion\_y+19);

printf ("%c", recibos[numero\_a\_asignar].numero\_en\_letras [i]);

}

}

gotoxy (posicion\_x+7, posicion\_y+1);

printf (" ");

fflush (stdin);

gets (recibos[numero\_a\_asignar].ciudad);

if (strlen(recibos[numero\_a\_asignar].ciudad)>24)

{

recibos[numero\_a\_asignar].ciudad [24] = '\0';

}

gotoxy (posicion\_x+10, posicion\_y+7);

printf ("");

fflush (stdin);

gets (recibos[numero\_a\_asignar].concepto);

if (strlen(recibos[numero\_a\_asignar].concepto)>49)

{

recibos[numero\_a\_asignar].concepto [49] = '\0';

}

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

}

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

/\*

Nombre subrutina: agregar\_recibo

Tipo: void

Objetivo: Agregar un nuevo recibo al arreglo de recibos.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

Retorno: Ninguno.

Ejemplo de uso: agregar\_recibo(recibos);

Notas: Esta función permite agregar un nuevo recibo al arreglo de recibos. Primero se limpia la pantalla,

luego se aplica el estilo de los recibos utilizando la función estilo\_recibos. A continuación, se utiliza

la función asignar\_datos para ingresar los datos del nuevo recibo en la posición indicada por el

número\_recibos actual. Finalmente, se incrementa el número\_recibos en 1.

\*/

void agregar\_recibo (struct Recibo recibos [])

{

system ("cls");

estilo\_recibos (2, 1);

asignar\_datos (recibos, numero\_recibos, 2, 1);

numero\_recibos++;

}

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

/\*

Nombre subrutina: consultar\_recibo

Tipo: void

Objetivo: Consultar un recibo a partir de un número ingresado.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

- numero\_ingresado: número del recibo a consultar.

Retorno: Ninguno.

Ejemplo de uso: consultar\_recibo(recibos, numero\_ingresado);

Notas: Esta función consulta un recibo a partir del número ingresado. Primero verifica si el número existe

en los recibos registrados utilizando la función comparar\_numero. Si el número existe, muestra los datos del

recibo utilizando la función ubicacion\_datos. En caso contrario, muestra un mensaje indicando que el número

no está registrado.

\*/

void consultar\_recibo (struct Recibo recibos [], char numero\_ingresado [5])

{

system ("cls");

int numero\_estructura;

numero\_estructura = comparar\_numero (recibos, numero\_ingresado);

if (numero\_estructura == -1)

{

printf ("El numero que ingreso no esta registrado\n\n");

}

else

{

estilo\_recibos (2, 1);

ubicacion\_datos (recibos, 2, 1, numero\_estructura);

}

}

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

/\*

Nombre subrutina: modificar\_recibo

Tipo: void

Objetivo: Modificar los datos de un recibo a partir de un número ingresado.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

- numero\_ingresado: número del recibo a modificar.

Retorno: Ninguno.

Ejemplo de uso: modificar\_recibo(recibos, numero\_ingresado);

Notas: Esta función modifica los datos de un recibo a partir del número ingresado. Primero verifica si el

número existe en los recibos registrados utilizando la función comparar\_numero. Si el número existe,

muestra los datos actuales del recibo y luego solicita los nuevos datos utilizando la función asignar\_datos.

Los nuevos datos se asignan a la misma posición en el arreglo de recibos.

\*/

void modificar\_recibo (struct Recibo recibos [], char numero\_ingresado [5])

{

system ("cls");

int numero\_estructura;

numero\_estructura = comparar\_numero (recibos, numero\_ingresado);

if (numero\_estructura == -1)

{

printf ("El numero que ingreso no esta registrado\n\n");

}

else

{

gotoxy (1, 0);

printf ("Datos actuales del recibo:");

estilo\_recibos (1, 1);

ubicacion\_datos (recibos, 1, 1, numero\_estructura);

gotoxy (63, 0);

printf ("Datos nuevos del recibo:");

estilo\_recibos (63, 1);

asignar\_datos (recibos, numero\_estructura, 63, 1);

}

}

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

/\*

Nombre subrutina: total\_valores

Tipo: long

Objetivo: Calcular el total de los valores de los recibos registrados.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

Retorno: El total de los valores de los recibos registrados.

Ejemplo de uso: long total = total\_valores(recibos);

Notas: Esta función calcula el total de los valores de los recibos registrados. Itera sobre el arreglo de

recibos y suma los valores de cada recibo. Retorna el resultado total.

\*/

long total\_valores (struct Recibo recibos [])

{

long total;

int i;

for (i=0; i<numero\_recibos; i++)

{

total = total + recibos[i].valor;

}

return total;

}

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

/\*

Nombre subrutina: valor\_maximo

Tipo: int

Objetivo: Obtener la ubicación del recibo con el valor máximo.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

Retorno: La ubicación del recibo con el valor máximo en el arreglo.

Ejemplo de uso: int ubicacion = valor\_maximo(recibos);

Notas: Esta función recorre el arreglo de recibos y busca el recibo con el valor máximo. Retorna la

ubicación (índice) de ese recibo en el arreglo. En caso de empate, retorna la primera ubicación encontrada.

\*/

int valor\_maximo (struct Recibo recibos [])

{

int i=1, ubicacion\_parcial=0;

long maximo\_parcial = recibos[0].valor;

while (i<numero\_recibos)

{

if (recibos[i].valor > maximo\_parcial)

{

maximo\_parcial = recibos[i].valor;

ubicacion\_parcial = i;

}

i++;

}

return ubicacion\_parcial;

}

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

/\*

Nombre subrutina: valor\_minimo

Tipo: int

Objetivo: Obtener la ubicación del recibo con el valor mínimo.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

Retorno: La ubicación del recibo con el valor mínimo en el arreglo.

Ejemplo de uso: int ubicacion = valor\_minimo(recibos);

Notas: Esta función recorre el arreglo de recibos y busca el recibo con el valor mínimo. Retorna la

ubicación (índice) de ese recibo en el arreglo. En caso de empate, retorna la primera ubicación encontrada.

\*/

int valor\_minimo (struct Recibo recibos [])

{

int i=1, ubicacion\_parcial=0;

long minimo\_parcial = recibos[0].valor;

while (i<numero\_recibos)

{

if (recibos[i].valor < minimo\_parcial)

{

minimo\_parcial = recibos[i].valor;

ubicacion\_parcial = i;

}

i++;

}

return ubicacion\_parcial;

}

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

/\*

Nombre subrutina: mostrar\_lista\_recibos

Tipo: void

Objetivo: Mostrar en pantalla la lista de recibos registrados.

Parámetros: Ninguno.

Retorno: Ninguno.

Ejemplo de uso: mostrar\_lista\_recibos();

Notas: Esta función muestra en pantalla la lista de recibos registrados. Imprime cada uno de los campos

de los recibos, incluyendo el número, la fecha, el beneficiario, el valor y la ciudad. Además, calcula y

muestra el total de los valores de los recibos, el recibo con el valor máximo y el recibo con el valor mínimo.

\*/

void mostrar\_lista\_recibos ()

{

system ("cls");

estilo\_lista ();

gotoxy (1, 1);

printf ("# RC");

gotoxy (14, 1);

printf ("Fecha");

gotoxy (33, 1);

printf ("Pagado a");

gotoxy (65, 1);

printf ("Valor");

gotoxy (83, 1);

printf ("Ciudad");

int i;

for (i=0; i<numero\_recibos; i++)

{

gotoxy (1, 3+i);

printf ("%s", recibos[i].numero);

gotoxy (14, 3+i);

printf ("%s", recibos[i].fecha);

gotoxy (33, 3+i);

printf ("%s", recibos[i].nombre);

gotoxy (65, 3+i);

printf ("$%ld", recibos[i].valor);

gotoxy (83, 3+i);

printf ("%s", recibos[i].ciudad);

}

long total;

total = total\_valores (recibos);

gotoxy (1, numero\_recibos+4);

printf ("Total: $%ld", total);

int ubicacion\_maximo;

ubicacion\_maximo = valor\_maximo (recibos);

gotoxy (1, numero\_recibos+5);

printf ("Valor m%cximo: No. %s Valor $%ld", 160, recibos[ubicacion\_maximo].numero, recibos[ubicacion\_maximo].valor);

int ubicacion\_minimo;

ubicacion\_minimo = valor\_minimo (recibos);

gotoxy (1, numero\_recibos+6);

printf ("Valor m%cnimo: No. %s Valor $%ld", 161, recibos[ubicacion\_minimo].numero, recibos[ubicacion\_minimo].valor);

gotoxy (1, numero\_recibos+8);

system ("pause");

}

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

/\*

Nombre subrutina: guardar\_datos

Tipo: void

Objetivo: Guardar los datos de los recibos en un archivo de texto.

Parámetros:

- recibos: arreglo de estructuras de tipo Recibo.

Retorno: Ninguno.

Ejemplo de uso: guardar\_datos(recibos);

Notas: Esta función guarda los datos de los recibos en un archivo de texto llamado "Recibos.txt". Abre el archivo

en modo escritura y, si la apertura es exitosa, itera sobre el arreglo de recibos y escribe cada campo de los recibos

en el archivo, separados por punto y coma. Cada recibo se escribe en una nueva línea. Finalmente, cierra el archivo.

\*/

void guardar\_datos (struct Recibo recibos [])

{

FILE \*archivo = fopen ("Recibos.txt", "w");

if (archivo == NULL)

{

printf ("No se pudo abrir el archivo\n");

}

else

{

int i;

for (i=0; i<numero\_recibos; i++)

{

fputs (recibos[i].numero, archivo);

fputc (';', archivo);

fputs (recibos[i].fecha, archivo);

fputc (';', archivo);

fputs (recibos[i].nombre, archivo);

fputc (';', archivo);

fprintf (archivo, "%ld", recibos[i].valor);

fputc (';', archivo);

fputs (recibos[i].ciudad, archivo);

fputc (';', archivo);

fputs (recibos[i].concepto, archivo);

fputc (';', archivo);

fputc ('\n', archivo);

}

fclose (archivo);

}

}

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

/\*

Nombre subrutina: menu

Tipo: void

Objetivo: Mostrar el menú principal y gestionar las opciones seleccionadas por el usuario.

Parámetros: Ninguno.

Retorno: Ninguno.

Ejemplo de uso: menu();

Notas: Esta función muestra un menú principal con diferentes opciones y solicita al usuario que ingrese una opción.

Luego, según la opción seleccionada, ejecuta la función correspondiente o muestra un mensaje de error en caso de opción

inválida. Las opciones disponibles son:

1. Capturar Documento: Permite agregar un nuevo recibo llamando a la función agregar\_recibo.

2. Consultar Documento: Solicita al usuario el número de recibo a consultar y llama a la función consultar\_recibo.

3. Listar Documentos: Muestra la lista de todos los recibos registrados llamando a la función mostrar\_lista\_recibos.

4. Modificar Documento: Solicita al usuario el número de recibo a modificar y llama a la función modificar\_recibo.

5. Salir: Guarda los datos de los recibos en un archivo de texto llamando a la función guardar\_datos y finaliza la

ejecución del programa.

Si se ingresa una opción inválida, se muestra un mensaje de error y se vuelve a mostrar el menú principal.

\*/

void menu ()

{

int opcion;

system ("cls");

estilo\_menu ();

gotoxy (1, 1);

printf ("Men%c principal\n", 163);

gotoxy (1, 3);

printf ("1. Capturar Documento\n");

gotoxy (1, 4);

printf ("2. Consultar Documento\n");

gotoxy (1, 5);

printf ("3. Listar Documentos\n");

gotoxy (1, 6);

printf ("4. Modificar Documento\n");

gotoxy (1, 7);

printf ("5. Salir\n");

gotoxy (1, 8);

printf ("Ingrese la opci%cn que desea: ", 162);

scanf ("%d", &opcion);

if (opcion>5 || opcion<1)

{

printf ("\nLa opci%cn que ingres%c no es v%clida\n\n", 162, 162, 160);

system ("pause");

menu ();

}

if (opcion == 1)

{

agregar\_recibo (recibos);

system ("pause");

menu ();

}

if (opcion == 2)

{

system ("cls");

char buscar\_numero [5];

printf ("Ingrese el n%cmero del recibo que desea consultar (XXXX): ", 163);

fflush (stdin);

gets (buscar\_numero);

consultar\_recibo (recibos, buscar\_numero);

system ("pause");

menu ();

}

if (opcion == 3)

{

mostrar\_lista\_recibos ();

menu ();

}

if (opcion == 4)

{

system ("cls");

char buscar\_numero [5];

printf ("Ingrese el n%cmero del recibo que desea modificar (XXXX): ", 163);

fflush (stdin);

gets (buscar\_numero);

modificar\_recibo (recibos, buscar\_numero);

system ("pause");

menu ();

}

if (opcion == 5)

{

guardar\_datos (recibos);

}

}

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

/\*

Nombre función: main

Tipo: int

Objetivo: Punto de entrada del programa.

Retorno: Valor entero que indica el estado de finalización del programa.

Notas: Esta función es el punto de entrada del programa. Llama a la función leer\_archivo() para cargar los datos

existentes desde un archivo. Luego, muestra el menú principal llamando a la función menu() para que el usuario

pueda interactuar con las opciones. Finalmente, retorna 0 para indicar que el programa finalizó sin errores.

\*/

int main ()

{

leer\_archivo ();

menu ();

return 0;

}