A gold and blue coat of arms

Description automatically generated

ESTRUCTURA DE DATOS

TAREA INDIVIDUAL N.º 1

FECHA DE ENTREGA: 26-03-24

CODIGO DE MATRICULA: 22200007

APELLIDOS Y NOMBRES: BENITES PARDAVE, EDER GUSTAVO

TEMA: ARCHIVOS SECUENCIALES

DESARROLLO DE PROBLEMAS:

1 – 3

#include <conio.h>

#include <stdio.h>

#include <stdlib.h>

#include <fstream>

#include <iostream>

#include <string>

#include <cstring> *// Add this line*

using namespace std;

struct *Producto*

{

    char codigo[10];

    char nombre[10];

    int cantidad;

};

class *Fila*

{

*FILE* \*F;

    int N;

*Producto* P;

public:

    void *crear*();

    void *recorrer*();

    void *buscar*();

    void *eliminar*();

};

void *Fila*::*crear*()

{

    int i, n;

    F = *fopen*("productos.txt", "w");

    if (F == *NULL*)

    {

        cout *<<* "Error al abrir el archivo" *<<* *endl*;

*exit*(1);

    }

    cout *<<* "Ingrese la cantidad de productos: ";

    cin *>>* n;

    i = 0;

    while (i < n)

    {

        cout *<<* "Producto " *<<* i + 1 *<<* *endl*;

        cout *<<* "Codigo: ";

        cin *>>* P.codigo;

        cout *<<* "Nombre: ";

        cin *>>* P.nombre;

        cout *<<* "Cantidad: ";

        cin *>>* P.cantidad;

*fwrite*(&P, sizeof(P), 1, F);

        i++;

    }

*fclose*(F);

}

void *Fila*::*recorrer*()

{

    F = *fopen*("productos.txt", "r");

    if (F == *NULL*)

    {

        cout *<<* "Error al abrir el archivo" *<<* *endl*;

*exit*(1);

    }

*fread*(&P, sizeof(P), 1, F);

    int bol = 0;

    cout *<<* "Reporte de productos" *<<* *endl*;

    while (!*feof*(F))

    {

        bol++;

        cout *<<* *endl*

*<<* *endl*

*<<* bol *<<* " Codigo " *<<* P.codigo;

        cout *<<* "\nNombre: " *<<* P.nombre *<<* *endl*;

        cout *<<* *endl*

*<<* *endl*

*<<* "Cantidad: " *<<* P.cantidad;

        if (bol % 2 == 0)

        {

            cout *<<* "Presione una tecla para continuar" *<<* *endl*;

        }

*fread*(&P, sizeof(P), 1, F);

    }

*fclose*(F);

*getch*();

}

void *Fila*::*buscar*()

{

    F = *fopen*("productos.txt", "r");

    if (F == *NULL*)

    {

        cout *<<* "Error al abrir el archivo" *<<* *endl*;

*exit*(1);

    }

    char codigo[10];

    cout *<<* "Ingrese el codigo del producto a buscar: ";

    cin *>>* codigo;

*fread*(&P, sizeof(P), 1, F);

    int bol = 0;

    bool encontrado = false;

    while (!*feof*(F))

    {

        bol++;

        if (*strcmp*(codigo, P.codigo) == 0)

        {

            encontrado = true;

            cout *<<* *endl*

*<<* "El producto con el codigo " *<<* codigo *<<* " se ha encontrado" *<<* *endl*;

            cout *<<* "Nombre: " *<<* P.nombre *<<* *endl*;

            cout *<<* "Cantidad: " *<<* P.cantidad *<<* *endl*;

            break;

        }

*fread*(&P, sizeof(P), 1, F);

    }

    if (!encontrado)

    {

        cout *<<* "No se ha encontrado el producto con el codigo " *<<* codigo *<<* *endl*;

    }

*fclose*(F);

*getch*();

}

*// 7AB01*

*// 8AB01*

void *Fila*::*eliminar*()

{

*FILE* \*G;

    F = *fopen*("productos.txt", "r");

    if (F == *NULL*)

    {

        cout *<<* "Error al abrir el archivo" *<<* *endl*;

*exit*(1);

    }

    G = *fopen*("PTemp.txt", "w");

    if (G == *NULL*)

    {

        cout *<<* "Error al abrir el archivo temporal" *<<* *endl*;

*exit*(1);

    }

    char codigo[10] = "8AB01";

*fread*(&P, sizeof(P), 1, F);

    int bol = 0;

    bool encontrado = false;

    while (!*feof*(F))

    {

        bol++;

        if (*strcmp*(codigo, P.codigo) != 0)

        {

*fwrite*(&P, sizeof(P), 1, G);

        }

        else

        {

            encontrado = true;

        }

*fread*(&P, sizeof(P), 1, F);

    }

*fclose*(F);

*fclose*(G);

    int rem;

    rem = *remove*("productos.txt");

    cout *<<* *endl*

*<<* "Se elimino el registro pedido";

    int ren;

    ren = *rename*("PTemp.txt", "productos.txt");

    cout *<<* *endl*

*<<* "Se renombro el archivo";

    if (!encontrado)

    {

        cout *<<* *endl*

*<<* "No existe el codigo 8AB01" *<<* *endl*;

    }

*getch*();

}

int *menu*()

{

    int opc;

    cout *<<* "Menu" *<<* *endl*;

    cout *<<* "1. Crear archivo" *<<* *endl*;

    cout *<<* "2. Recorrer archivo" *<<* *endl*;

    cout *<<* "3. Buscar producto" *<<* *endl*;

    cout *<<* "4. Eliminar producto" *<<* *endl*;

    cout *<<* "5. Salir" *<<* *endl*;

    cout *<<* "Opcion: ";

    cin *>>* opc;

    return opc;

}

*main*()

{

    int opc;

*Fila* fil;

    int i, n;

    opc = *menu*();

    while (opc != 5)

    {

        switch (opc)

        {

        case 1:

            fil.*crear*();

            break;

        case 2:

            fil.*recorrer*();

            break;

        case 3:

            fil.*buscar*();

            break;

        case 4:

            fil.*eliminar*();

            break;

        case 5:

            cout *<<* "Saliendo..." *<<* *endl*;

            break;

        default:

            cout *<<* "Opcion no valida" *<<* *endl*;

            break;

        }

        opc = *menu*();

    }

}

4

*// ejemplo1.c: Muestra un archivo dos veces.*

#include <stdio.h>

int *main*()

{

*FILE* \*fichero;

   fichero = *fopen*("ejemplo1.c", "r");

   while (!*feof*(fichero))

*fputc*(*fgetc*(fichero), *stdout*);

*rewind*(fichero);

   while (!*feof*(fichero))

*fputc*(*fgetc*(fichero), *stdout*);

*fclose*(fichero);

*getchar*();

   return 0;

}

*// realizer la copia de un archivo origen en un archivo destino*

#include <stdio.h>

int *main*(int *argc*, char \*\**argv*)

{

*FILE* \*fe, \*fs;

   unsigned char buffer[2048]; *// Buffer de 2 Kbytes*

   int bytesLeidos;

   if (*argc* != 3)

   {

*printf*("Usar: copia <fichero\_origen> <fichero\_destino>\n");

      return 1;

   }

*// Abrir el fichero de entrada en lectura y binario*

   fe = *fopen*(*argv*[1], "rb");

   if (!fe)

   {

*printf*("El fichero %s no existe o no puede ser abierto.\n", *argv*[1]);

      return 1;

   }

*// Crear o sobreescribir el fichero de salida en binario*

   fs = *fopen*(*argv*[2], "wb");

   if (!fs)

   {

*printf*("El fichero %s no puede ser creado.\n", *argv*[2]);

*fclose*(fe);

      return 1;

   }

*// Bucle de copia:*

   while ((bytesLeidos = *fread*(buffer, 1, 2048, fe)))

*fwrite*(buffer, 1, bytesLeidos, fs);

*// Cerrar ficheros:*

*fclose*(fe);

*fclose*(fs);

   return 0;

}

5

*// mezcla.c : Ordenamiento de archivos secuenciales*

*// Ordena ficheros de texto por orden alfabético de líneas*

*// Usando el algoritmo de mezcla natural*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void *Mostrar*(*FILE* \**fich*);

void *Mezcla*(*FILE* \**fich*);

void *Separar*(*FILE* \**fich*, *FILE* \*\**aux*);

int *Mezclar*(*FILE* \**fich*, *FILE* \*\**aux*);

int *main*()

{

*FILE* \*fichero;

    fichero = *fopen*("mezcla.txt", "r+");

*puts*("Fichero desordenado\n");

*Mostrar*(fichero);

*puts*("Ordenando fichero\n");

*Mezcla*(fichero);

*puts*("Fichero ordenado\n");

*Mostrar*(fichero);

*fclose*(fichero);

*system*("PAUSE");

    return 0;

}

*// Muestra el contenido del fichero "fich"*

void *Mostrar*(*FILE* \**fich*)

{

    char linea[128];

*rewind*(*fich*);

*fgets*(linea, 128, *fich*);

    while (!*feof*(*fich*))

    {

*puts*(linea);

*fgets*(linea, 128, *fich*);

    }

}

*// Algoritmo de mezcla:*

void *Mezcla*(*FILE* \**fich*)

{

    int ordenado;

*FILE* \*aux[2];

*// Bucle que se repite hasta que el fichero esté ordenado:*

    do

    {

*// Crea los dos ficheros auxiliares para separar los tramos:*

        aux[0] = *fopen*("aux1.txt", "w+");

        aux[1] = *fopen*("aux2.txt", "w+");

*rewind*(*fich*);

*Separar*(*fich*, aux);

*rewind*(aux[0]);

*rewind*(aux[1]);

*rewind*(*fich*);

        ordenado = *Mezclar*(*fich*, aux);

*fclose*(aux[0]);

*fclose*(aux[1]);

    } while (!ordenado);

*// Elimina los archivos auxiliares:*

*remove*("aux1.txt");

*remove*("aux2.txt");

}

*// Separa los tramos ordenados alternando entre los archivos aux:*

void *Separar*(*FILE* \**fich*, *FILE* \*\**aux*)

{

    char linea[128], anterior[2][128];

    int salida = 0;

*// Volores iniciales para los últimos valores*

*// almacenados en los ficheros auxiliares*

*strcpy*(anterior[0], "");

*strcpy*(anterior[1], "");

*// Captura la primero línea:*

*fgets*(linea, 128, *fich*);

    while (!*feof*(*fich*))

    {

*// Decide a qué archivo de salida corresponde la línea leída:*

        if (salida == 0 && *strcmp*(linea, anterior[0]) < 0)

            salida = 1;

        else if (salida == 1 && *strcmp*(linea, anterior[1]) < 0)

            salida = 0;

*// Almacena la línea actual como la última añadida:*

*strcpy*(anterior[salida], linea);

*// Añade la línea al fichero auxiliar:*

*fputs*(linea, *aux*[salida]);

*// Lee la siguiente línea:*

*fgets*(linea, 128, *fich*);

    }

}

*// Mezcla los archivos auxiliares:*

int *Mezclar*(*FILE* \**fich*, *FILE* \*\**aux*)

{

    char ultima[128], linea[2][128], anterior[2][128];

    int entrada;

    int tramos = 0;

*// Lee la primera línea de cada fichero auxiliar:*

*fgets*(linea[0], 128, *aux*[0]);

*fgets*(linea[1], 128, *aux*[1]);

*// Valores iniciales;*

*strcpy*(ultima, "");

*strcpy*(anterior[0], "");

*strcpy*(anterior[1], "");

*// Bucle, mientras no se acabe ninguno de los archivos auxiliares(quedan tramos por mezclar) :*

    while (!*feof*(*aux*[0]) && !*feof*(*aux*[1]))

    {

*// Selecciona la línea que se añadirá:*

        if (*strcmp*(linea[0], linea[1]) <= 0)

            entrada = 0;

        else

            entrada = 1;

*// Almacena el valor como el último añadido:*

*strcpy*(anterior[entrada], linea[entrada]);

*// Añade la línea al fichero:*

*fputs*(linea[entrada], *fich*);

*// Lee la siguiente línea del fichero auxiliar:*

*fgets*(linea[entrada], 128, *aux*[entrada]);

*// Verifica fin de tramo, si termino copia el resto del tramo:*

        if (*strcmp*(anterior[entrada], linea[entrada]) > 0)

        {

            if (!entrada)

                entrada = 1;

            else

                entrada = 0;

            tramos++;

*// Copia lo que queda del tramo actual al archivo de Salida;*

            do

            {

*strcpy*(anterior[entrada], linea[entrada]);

*fputs*(linea[entrada], *fich*);

*fgets*(linea[entrada], 128, *aux*[entrada]);

            } while (!*feof*(*aux*[entrada]) && *strcmp*(anterior[entrada], linea[entrada]) <= 0);

        }

    }

*// Añadir tramos que queden sin mezclar:*

    if (!*feof*(*aux*[0]))

        tramos++;

    while (!*feof*(*aux*[0]))

    {

*fputs*(linea[0], *fich*);

*fgets*(linea[0], 128, *aux*[0]);

    }

    if (!*feof*(*aux*[1]))

        tramos++;

    while (!*feof*(*aux*[1]))

    {

*fputs*(linea[1], *fich*);

*fgets*(linea[1], 128, *aux*[1]);

    }

    return (tramos == 1);

}

INFORMAR SOBRE EL DESARROLLO DE LOS EJERCICIOS EN EL COMPUTADOR

|  |  |  |  |
| --- | --- | --- | --- |
| PROBLEMA | OBTUVO RESULTADOS | SOLO LO COMPILO | SOLO LO CODIFICO |
| 1 | El código tiene problemas de con las llaves y problema con la lógica al crear un archivo | No se pudo compilar a causa de los errores. | Lo codifiqué tal y como esta en el archivo Word y lo mejoré. |
| 2 | El código esta esta incorrecto ya que no detecta el producto con el código, el problema fue el método de comparación, ya que se debe usar strcmp | Se compilo de forma normal. | Lo codifiqué tal y como esta en el archivo Word y lo mejoré. |
| 3 | El código esta incorrecto porque algunas variables te tipo FILE no están inicializadas, además hay un retorno incorrecto de la función | No se pudo compilar a causa de los errores. | Lo codifiqué tal y como esta en el archivo Word y lo mejoré. |
| 4 | Los códigos funcionan, el primero muestra 2 veces el archivo y el segundo copia un archivo a otro. Incluso, en ambos casos, los muestra sin los símbolos de comentario de línea “//” | Se compilo de manera normal comentando los otros programas. | Lo codifiqué tal y como esta en el archivo Word. |
| 5 | El ultimo algoritmo de mezcla natural, funciona bien. Ordena los numero s del archivo de forma ascendente y crea 2 archivos auxiliares. | Se compilo de manera normal. | Lo codifiqué tal y como esta en el archivo Word. |