Fundació Jesuïtes Educació
UOC Universitat Oberta de Catalunya

Programa comandos personalizados para el sistema operativo

Código del proyecto: ICX0_P4

Producto 2 - Adquiriendo las destrezas básicas

Alumno: Jesús Real Tovar
Consultor/ra: Óscar Baltà Fabró
Fecha de entrega: 13 de mayo de 2019

1. Mostrar inicialmente un menú

Inicialmente mostramos el menú con las tres funciones implementadas en la función *show_menu*, esta función se encuentra en la aplicación de consola en el proyecto principal de la solución, hace uso de librerías estáticas para el uso de funciones de entrada/salida en consola o funciones de archivo.

```
/*
  Función para construir el menú de la aplicación de consola.
*/
void show_menu() {
  print_msg("
    ↪ ==========================================
    ↪ ");
  print_msg("Especifica que quieres hacer por su numero:");
  print_msg("1. Mostrar el contenido de un archivo.");
  print_msg("2. Guardar como un archivo.");
  print_msg("3. Chequear el contenido de un archivo.");
  // Preguntamos por la opción a ejecutar.
  int selected = ask_for_integer();
  print_msg_int("Has seleccionado: ", selected);
  switch (selected)
  {
    case 1:
      // Mostramos el contenido de un archivo
      show_content_of_file();
      break;
    case 2:
      // Guardamos como, un archivo
      save_file_as();
      break;
    case 3:
      // Chequeamos el contenido de un archivo
      check_content_of_file();
      break;
    default:
      print_msg("La opci\242n seleccionada no es correcta.");
      // Volvemos a mostrar el menú si la opción seleccionada no es correcta
      show_menu();
  }
}
```

2. Diseñar una función que muestre el contenido de un archivo

```
/*
  Función para mostrar el contenido de un archivo
*/
void show_content_of_file() {
  print_msg("
    ↪ ================================================
    ↪ ");
  print_msg("Introduce la ruta del archivo a leer:");
  const char * path[255];
  // Obetenemos mediante consola, la ruta del archivo
  read_string(path, 255);

  // Abrimos el archivo
  FILE * ptr_file = open_file(path);
  // Si el archivo no existe retornamos
  if (ptr_file == NULL)
  {
    print_msg("El archivo no existe.");
    return;
  }
  print_msg("========== Contenido del archivo: ==========");
  char c;
  c = fgetc(ptr_file);
  // Mientras que no lleguemos al final del archivo
  while (c != EOF)
  {
    // Escribimos en consola cada carácter del archivo
    print_char(c);
    c = fgetc(ptr_file);
  }
  print_line_break();
  // Cerramos el archivo
  fclose(ptr_file);
}
```

3. Diseñar una función que implemente el guardado como de un archivo

```
/*
  Función para guardar como, un archivo
*/
void save_file_as() {
  print_msg("================================================");
  print_msg("Introduce la ruta del archivo a copiar:");
  const char * path[255];
  // Obetenemos mediante consola, la ruta del archivo origen
  read_string(path, 255);

  // Abrimos el archivo
  FILE * ptr_file = open_file(path);
  // Si el archivo no existe retornamos
  if (ptr_file == NULL)
  {
    print_msg("El archivo no existe.");
    return;
  }

  print_msg("Introduce la ruta del archivo destino:");
  const char * path_cp[255];
  // Obetenemos mediante consola, la ruta del archivo destino
  read_string(path_cp, 255);

  // Creamos el archivo
  FILE * ptr_file_cp = create_file(path_cp);
  // Si ha habido error al crear el archivo retornamos
  if (ptr_file_cp == NULL)
  {
    print_msg("Error al crear el archivo.");
    return;
  }

  print_msg("Archivo destino creado.");

  char c;
  c = fgetc(ptr_file);
  // Mientras que no lleguemos al final del archivo
  while (c != EOF)
  {
    // Copiamos cada carácter del archivo origen, en el archivo destino
    fputc(c, ptr_file_cp);
    c = fgetc(ptr_file);
  }
  print_line_break();
  // Cerramos el archivo origen
  fclose(ptr_file);
  // Borramos el archivo origen
  if (remove(path) != 0)
    print_msg("No se pudo eliminar el archivo de origen.");
  else
    print_msg("Archivo origen eliminado.");
  // Cerramos el archivo destino
  fclose(ptr_file_cp);
}
```

4. Diseñar una función que chequee el contenido de un archivo

```
/*
Función para chequear el contenido de un archivo
*/
void check_content_of_file() {
  print_msg("================================================");
  print_msg("Introduce la ruta del archivo a buscar:");
  const char * path[255];
  // Obetenemos mediante consola, la ruta del archivo
  read_string(path, 255);

  // Abrimos el archivo
  FILE * ptr_file = open_file(path);
  // Si el archivo no existe retornamos
  if (ptr_file == NULL)
  {
    print_msg("El archivo no existe.");
    return;
  }

  print_msg("Introduce la palabra o frase a buscar:");
  const char * word[255];
  // Obetenemos mediante consola, la palabra o frase a buscar
  read_string(word, 255);

  char c;
  c = fgetc(ptr_file);
  bool found = FALSE;
  char line[1024];

  // Por cada linea del archivo
  while (fgets(line, 1024, ptr_file) != NULL) {
    // Comprobamos si contiene la palabra o frase a buscar
    if ((strstr(line, word)) != NULL) {
      // Si la contiene, seteamos como encontrada y rompemos el bucle
      found = TRUE;
      break;
    }
  }

  // Cerramos el archivo
  fclose(ptr_file);

  // Mostramos en consola el resultado
  if (found) {
    print_msg("Palabra encontrada!");
  }
  else {
    print_msg("Palabra no encontrada.");
  }
}
```

5. Realizar la aplicación modularizada

Para llevar a cabo la modularización de la aplicación, se ha distribuido el código en una aplicación de consola principal y dos librerías estáticas, una para las funciones de consola (librería *consolelib*), para el uso de funciones de entrada/salida en consola y otra para las funciones de archivos (librería *filelib*).

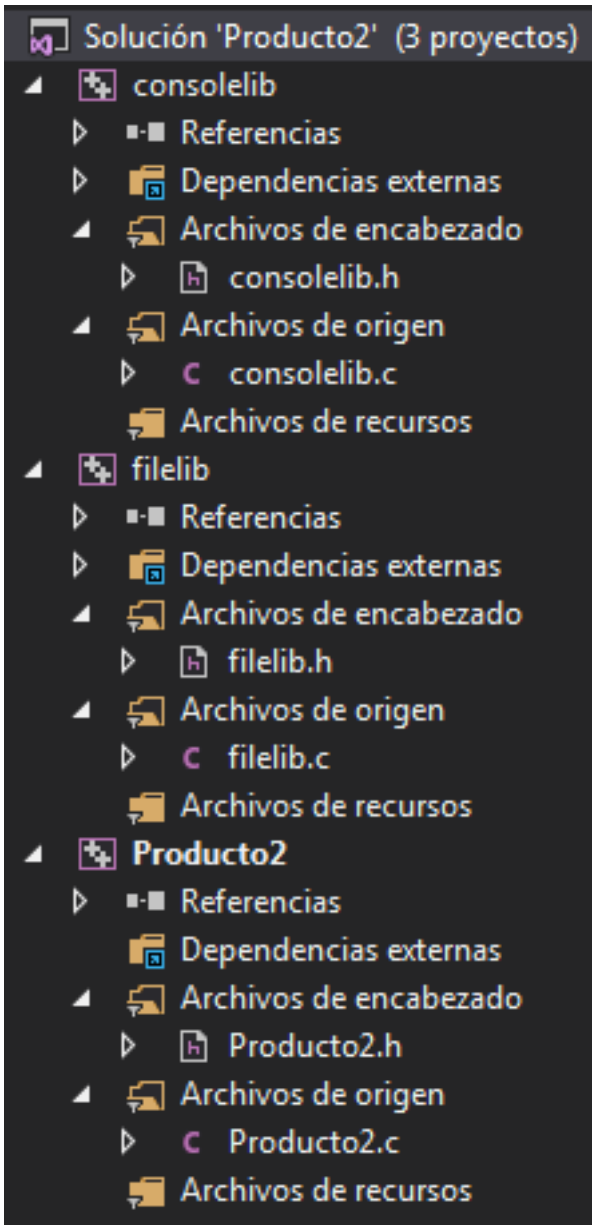

Figura 1: Solución en Visual Studio.