MEMORIA

PROYECTO CARTAS

Nombre: Aarón Riveiro Vilar
DNI: 39464391W

MÓDULO ENTRADA/SALIDA (inout.c y inout.h)

Este módulo contiene aquellas funciones que le pedirán al usuario que introduzca algún dato, las funciones que muestran por pantalla la información de abonados o mensajes, y las funciones que constituyen la carátula del programa.

Función plano:

- Descripción: función que imprime los bordes superior e inferior de la carátula.
- Prototipo: void plano(char, int);
- Parámetros formales: carácter que formará los bordes, y el número de veces que se repite ese carácter.
- Pseudocódigo:

```
-Inicio de la función
     -Hasta que se hayan impreso todos los caracteres:
          -Imprimir el carácter las veces que se
          indique
```

Función rotulo:

- Descripción: función que imprime la línea central de la carátula.
- Prototipo: void rotulo(char *, char, int);
- Parámetros formales: cadena con el nombre del programa, carácter que formará los bordes y el número de espacios que ocupará la carátula.
- Pseudocódigo:

```
-Inicio de la función:
   -Se imprime un carácter
   -Se imprimen la mitad de los espacios en blanco
   -Se imprime el nombre
   -Se imprimen la mitad de los espacios en blanco
   -Se imprime un carácter
```

Función confirmar:

- Descripción: función para seleccionar entre una afirmación o negación.
- Prototipo: int confirmar(char *);
- Parámetros formales: mensaje de invitación.
- Resultado: un entero que varía en función de la respuesta.
- Pseudocódigo:

```
-Inicio de la función:
  -Hasta que se introduzca una respuesta correcta:
    -Se pide una respuesta:
      -Si es afirmativa, el resultado es 1
      -Si es negativa, es 0
      -Si es incorrecta, se repite la pregunta
```

Funcion lee_cadena:

- Descripción: función que lee una cadena del teclado.
- Prototipo: void lee_cadena(char *, int, char *);
- Parámetros formales: dirección de la cadena, longitud y mensaje de invitación.
- Resultado: se guarda la cadena leída.
- Pseudocódigo:

```
-Inicio de la función:
  -Hasta que se lea una cadena correcta:
    -Se pide una cadena
      -Si es correcta, se guarda
      -Si es incorrecta, se repite la pregunta
```

Funcion lee_entero:

- Descripción: función que lee un entero del teclado.
- Prototipo: int lee_entero(int, char *);
- Parámetros formales: máximo valor del entero y mensaje de invitación.
- Resultado: el entero introducido.
- Pseudocódigo:

```
-Inicio de la función:
  -Hasta que se lea un entero correcto:
    -Se pide un entero
      -Si es correcto, se guarda como resultado
      -Si es incorrecto, se repite la pregunta
```

Función muestra_abonado:

- Descripción: función que muestra la información de un abonado en forma compacta (la identidad y el nombre).
- Prototipo: void muestra_abonado(struct unAbonado *);
- Parámetros formales: el registro con los datos necesarios.
- Pseudocódigo:
  - -Inicio de la función:
    - -Se muestran la identidad y el nombre del abonado

Función muestra_extensa:

- Descripción: función que muestra la información de un mensaje en forma extensa.
- Prototipo: void muestra_extensa(struct unMensaje *);
- Parámetros formales: el registro con los datos necesarios.
- Pseudocódigo:
  - -Inicio de la función:
    - -Se muestran la identidad del emisor y destinatario, y el texto del mensaje en forma extensa

Función muestra_corta:

- Descripción: función que muestra la información de un mensaje en forma corta.
- Prototipo: void muestra_corta(struct unMensaje *);
- Parámetros formales: el registro con los datos necesarios.
- Pseudocódigo:
  - -Inicio de la función:
    - -Se muestran la identidad del emisor y destinatario, y el texto del mensaje en forma corta

MÓDULO OPERACIONES (operation.c y operation.h)

Este módulo contiene aquellas funciones que realizan las operaciones que el usuario indique, tales como añadir o eliminar abonados o mensajes, pero también las funciones que leen los ficheros de texto.

Función fichero_Abonados:

- Descripción: función que lee la información del fichero "abonados.txt".
- Prototipo: void fichero_Abonados(struct unAbonado **);
- Parámetros formales: la lista donde se almacenará la información.
- Resultado: la lista con la información actualizada.
- Pseudocódigo:

```
-Inicio de la función:
  -Si existe el fichero "abonados.txt":
    -Hasta que se lean todas las líneas:
      -Se lee una línea
      -Se almacena su información en un nuevo nodo
```

Función fichero_Mensajes:

- Descripción: función que lee la información del fichero "mensajes.txt".
- Prototipo: void fichero_Mensajes(struct unAbonado **, struct unMensaje **);
- Parámetros formales: la lista de los abonados y la lista donde se almacenará la información.
- Resultado: la lista con la información actualizada.
- Pseudocódigo:

```
-Inicio de la función:
  -Si existe el fichero "mensajes.txt", y el fichero
  "abonados.txt" no está vacío:
    -Hasta que se lean todas las líneas:
      -Se lee una línea
      -Se almacena su información en un nuevo nodo
```

Función suscribir:

- Descripción: función que crea un nuevo abonado.
- Prototipo: void suscribir(struct unAbonado **);
- Parámetros formales: la lista de los abonados.
- Resultado: un nuevo abonado en la lista.
- Pseudocódigo:

```
-Inicio de la función:
   -Se crea un nodo nuevo
   -Se pide el nombre con lee_cadena()
   -Se busca el mayor ID ya existente
   -Se crea un nuevo nodo en la lista con ese nombre y con
   identidad superior en una unidad al mayor ID ya
   existente
   -Se muestra con muestra_abonado la información del
   abonado nuevo
```

Función escribir:

- Descripción: función que crea un nuevo mensaje.
- Prototipo: void escribir(struct unAbonado **, struct unMensaje **);
- Parámetros formales: la lista de los abonados y la de los mensajes.
- Resultado: un nuevo mensaje en la lista.
- Pseudocódigo:

```
-Inicio de la función:
   -Si hay abonados:
      -Se pide la identidad del emisor con lee_entero()
      -Se busca un abonado con esa identidad
         -Si existe:
            -Se pide la identidad del destinatario con
            lee_entero()
            -Se busca un abonado con esa identidad
               -Si existe:
                  -Se pide el mensajes con lee_cadena()
                  -Se crea un nodo nuevo en la lista de
                  mensajes con la información obtenida
                  -Se muestra con muestra_extensa la
                  información del mensaje nuevo
```

Función listar:

- Descripción: función que muestra los mensajes disponibles para un abonado.
- Prototipo: void listar(struct unAbonado **, struct unMensaje **);
- Parámetros formales: la lista de los abonados y la de los mensajes.
- Pseudocódigo:

```
-Inicio de la función:
  -Si hay abonados:
    -Se pide el nombre del abonado con lee_cadena()
    -Se busca un abonado con ese nombre
      -Si existe:
        -Se pide la identidad del destinatario con
        lee_entero()
        -Se busca un abonado con esa identidad
          -Si existe:
            -Se muestra con muestra_corta los mensajes
            para ese abonado
```

Función borrar:

- Descripción: función que elimina un mensaje.
- Prototipo: void borrar(struct unAbonado **, struct unMensaje **);
- Parámetros formales: la lista de los abonados y la de los mensajes.
- Resultado: se elimina un mensaje de la lista de mensajes.
- Pseudocódigo:

```
-Inicio de la función:
  -Si hay abonados:
    -Se pide la identidad del abonado con lee_entero()
    -Se busca un abonado con esa identidad
      -Si existe y tiene mensajes:
        -Se pide la posición del mensaje a eliminar con
        lee_entero
        -Se libera el nodo del mensaje indicado
```

Función retirar:

- Descripción: función que elimina un abonado y todos sus mensajes.
- Prototipo: void retirar(struct unAbonado **, struct unMensaje **);
- Parámetros formales: la lista de los abonados y la de los mensajes.
- Resultado: se elimina un mensaje de la lista de mensajes.
- Pseudocódigo:

```
-Inicio de la función:
  -Si hay abonados:
    -Se pide la identidad del abonado con lee_entero()
    -Se busca un abonado con esa identidad
      -Si existe:
        -Se liberan los nodos de todos los mensajes del
        abonado indicado
        -Se libera el nodo del abonado indicado
```

Función salir:

- Descripción: función que confirma si el usuario desea salir y guarda toda la información obtenida durante la ejecución del programa en caso de respuesta afirmativa.
- Prototipo: void salir(int *, struct unAbonado **, struct unMensaje **);
- Parámetros formales: el entero donde se guardará la respuesta, la lista de los abonados y la de los mensajes.
- Resultado: la respuesta del usuario y la información almacenada en los ficheros de texto.
- Pseudocódigo:

```
-Inicio de la función:
  -Se pide una confirmación con la función confirmar():
    -Si la respuesta es afirmativa:
      -Se lee línea por línea la información de la lista
      de abonados
      -Se almacena esa información en el fichero
      "abonados.txt"
      -Se lee línea por línea la información de la lista
      de mensajes
      -Se almacena esa información en el fichero
      "mensajes.txt"
```

MÓDULO PRINCIPAL (letters.c)

Este módulo contiene la función principal, main, que se encarga de realizar todas las llamadas a las demás funciones cuando sea necesario, además de mostrar el menú principal.

Función main:

- Descripción: función que muestra el menú principal y realiza todas las llamadas a las demás funciones del programa.
- Prototipo: int main();
- Resultado: cuando se termina esta función, se termina la ejecución del programa.
- Pseudocódigo:

```
-Inicio del programa:
   -Se crean las listas de abonados y mensajes, y se llama
   a las funciones fichero_Abonados() y fichero_Mensajes()
   -Se muestra la carátula mediante las funciones plano() y
   rotulo()
      -Se muestra el menú principal
         -Se pide al usuario que seleccione una operación
            -Si seleccionó 'A' o 'a', se llama a suscribir()
            -Si seleccionó 'E' o 'e', se llama a escribir()
            -Si seleccionó 'L' o 'l', se llama a listar()
            -Si seleccionó 'B' o 'b', se llama a borrar()
            -Si seleccionó 'J' o 'j', se llama a retirar()
            -Si seleccionó 'S' o 's', se llama a salir()
               -Si hubo respuesta afirmativa, se termina la
               ejecución del programa
               -Si hubo respuesta negativa, se vuelve a
               mostrar el menú
           -Si se seleccionó una opción incorrecta, se
           vuelve a mostrar el menú
```