

Práctica 4

Gestor de correo fdimail

Grado en Ingeniería Informática
Grado en Ingeniería del Software
Grado en Ingeniería de Computadores

Virginia Francisco Gilmartín
Facultad de Informática
Universidad Complutense



Descripción del gestor de correo (I)

Gestor de correo local en modo consola

Pantalla principal del sistema:

```
Elija la opcion deseada:  
1- Acceder a mi cuenta de correo  
2- Crear cuenta de correo  
  
0- Salir
```

1. Acceder a mi cuenta de correo:
 - a. Solicita usuario y contraseña
 - b. Comprueba si el usuario existe
 - c. Comprueba si la contraseña es correcta
 - d. Accede a al gestor de correo del usuario
2. Crear cuenta de correo
 - a. Solicita usuario y contraseña
 - b. Comprueba que el usuario no existe
 - c. Crea la cuenta
 - d. Accede al gestor de correo del usuario



Descripción del gestor de correo (II)

Correo de ginebra@fdimail.com

-----Bandeja de entrada-----

L N	EMISOR	ASUNTO	FECHA
* 1	- lancelot@fdimail.com	Besos y abrazos	2015/3/23
* 2	- arturo@fdimail.com	¡Mesa en oferta!	2015/3/23
3	- arturo@fdimail.com	Petición a Lancelot	2015/3/23

Elija una opción:

- 1- Leer correo
- 2- Enviar correo
- 3- Borrar correo
- 4- Ver bandeja de salida
- 5- Lectura rápida de correos sin leer
- 0- Cerrar sesión

Introduzca una opción:

De: arturo@fdimail.com 2015/3/23 <20:25:32>
Para: lancelot@fdimail.com
Asunto: Comprar mesa de reuniones

Hola,
he pensado que podíamos comprar una mesa redonda para las reuniones...
¿Cómo lo ves?

Un saludo,
Arturo

Elija una opción:

- 1- Contestar correo
- 0- Volver a la bandeja

Introduzca una opción:



Módulos

- ✓ Correo
- ✓ ListaCorreos
- ✓ ListaRegistros
- ✓ Usuario
- ✓ ListaUsuarios
- ✓ Gestor
- ✓ Programa principal



Versiones

Entrega: 17/05 (23:55)

Módulos Correo y ListaCorreos

➤ 17/04

Módulo ListaRegistros

➤ 24/04

Módulo Usuario y ListaUsuarios

➤ 1/05

Módulo Gestor y Principal

➤ 12/05



Fundamentos de Programación: Práctica 4

Pag. 4



Depuración

¡Prueba cada uno de los módulos por separado!

¡Prueba las demos!

¡Prueba tu programa con distintos casos de uso!



Fundamentos de Programación: Práctica 4

Pag. 5



Correo. Tipos de datos

✓ tCorreo:

- Emisor
- Destinatario
- Asunto
- Cuerpo
- Fecha → time_t (entero con el número de segundos transcurridos desde el 1 de Enero de 1970)
 typedef time_t tFecha;
- Identificador único → Emisor + fecha
 - Ejemplo: pepe@fdimail.com 143456443



Correo. Subprogramas (I)

✓ void correoNuevo(tCorreo &correo, string emisor)

- Establecemos emisor como el emisor del nuevo correo
- Solicitamos al usuario el destinatario, el asunto y el cuerpo del mensaje
- La fecha del nuevo correo se obtendrá del sistema:
 mensaje.fecha = time(0);
- Generamos el id único concatenando el emisor y la fecha:

```
stringstream id;  
id << correo.emisor << "_" << correo.fecha;  
correo.id = id.str();
```



Hay que incluir la librería ctime



Hay que incluir la librería sstream



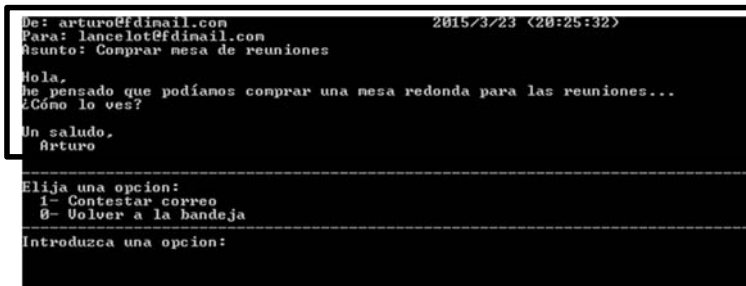
Correo. Subprogramas (II)

- ✓ void correoContestacion(const tCorreo &correoOriginal, tCorreo &correo, string emisor):
 - Establecemos emisor como el emisor del nuevo correo
 - Establecemos como destinatario del nuevo correo el emisor de correoOriginal
 - Establecemos como asunto del nuevo correo el del correoOriginal añadiendo "Re:" al principio
 - Solicitamos el cuerpo del mensaje al usuario y lo concatenamos al cuerpo de correoOriginal
 - La fecha del nuevo correo se obtendrá del sistema
 - Generamos el id único del nuevo correo concatenando el emisor y la fecha



Correo. Subprogramas (III)

- ✓ string aCadena(const tCorreo &correo)



- Para devolver un string con la información del correo concatenada usaremos el tipo *stringstream*

- Para mostrar la fecha:

```
string mostrarFecha(tFecha fecha){
    stringstream resultado;
    tm* ltm = localtime(&fecha);
    resultado<<1900 + ltm->tm_year<<"/"<<1 + ltm->tm_mon<<"/"<<ltm->tm_mday;
    resultado<<", "<<ltm->tm_hour<<":"<<ltm->tm_min<<":"<<ltm->tm_sec;
    return resultado.str();
}
```

- Para mostrar un correo por pantalla:

```
cout << aCadena(correo);
```



Correo. Subprogramas (III)

✓ `string obtenerCabecera(const tCorreo &correo)`

```
Correo de ginebra@fdimail.com
-----Bandeja de entrada-----
L N      EMISOR              ASUNTO                      FECHA
-----
* 1 -   Lancelot@fdimail.com  Besos y abrazos             2015/3/23
* 2 -   arturo@fdimail.com    Mensaje en oferta            2015/3/23
  3 -   arturo@fdimail.com    Petición a Lancelot          2015/3/23
-----
Elija una opcion:
1- Leer correo
2- Enviar correo
3- Borrar correo
4- Ver bandeja de salida
5- Lectura rapida de correos sin leer
0- Cerrar sesion
-----
Introduzca una opcion:
```

- Para devolver un string con la información de la cabecera concatenada usaremos el tipo *stringstream*

- Para mostrar la fecha sin hora:

```
string mostrarFechaSoloDia(tFecha fecha){
    stringstream resultado;
    tm* ltm = localtime(&fecha);
    resultado<<1900 + ltm->tm_year<<"/"<<1 + ltm->tm_mon<<"/"<<ltm->tm_mday;
    return resultado.str();
}
```

- Para mostrar la cabecera por pantalla:

```
cout << obtenerCabecera(correo);
```



Correo. Subprogramas (IV)

✓ `bool cargar(tCorreo &correo, ifstream &archivo):`

— Recibe el archivo de entrada y devuelve el correo leído

— Recibe el archivo ya abierto por parámetro para que cada módulo lea de archivo solo su información

```
bool cargar (tCorreo &correo, ifstream &archivo){
    bool ok;
    archivo >> correo.id;
    ...
    return ok;
}
```



Correo. Subprogramas (V)

```
arturo@fdimail.com_1426614381
1426614381
arturo@fdimail.com
ginebra@fdimail.com
Mensaje de prueba
Hola,
Parece que ya tenemos correo en Camelot!
Arturo
X
```

cargar (correo, archivo)

```
arturo@fdimail.com_1426614458
1426614458
arturo@fdimail.com
ginebra@fdimail.com;lancelot@fdimail.com
Comprar mesa de reuniones
Hola,
he pensado que podíamos comprar una mesa de forma redonda para las reuniones. ¿Os parece buena idea?
Arturo
X
```

cargar (correo, archivo)

```
lancelot@fdimail.com_1426613678
1426613678
lancelot@fdimail.com
ginebra@fdimail.com;arturo@fdimail.com
Cazamos mañana?
Hola a todos!

Os parece buena idea salir a cazar mañana.
Lancelot
X
```

cargar (correo, archivo)

XXX cargar (correo, archivo) → false



Correo. Subprogramas (VI)

- ✓ void guardar(const tCorreo &correo, ofstream& archivo):
 - Recibe el archivo de salida y el correo a guardar
 - Recibe el archivo ya abierto por parámetro para que cada módulo escriba en archivo solo su información

```
bool guardar(const tCorreo &correo, ofstream& archivo){
    archivo << correo.id << endl;
    archivo << correo.fecha << endl;
    ...
    archivo << CENTINELA_CUERPO << endl;
}
```



Correo. Subprogramas (VII)

```
arturo@fdimail.com_1426614381
1426614381
arturo@fdimail.com
ginebra@fdimail.com
Mensaje de prueba
Hola,
Parece que ya tenemos correo en Camelot!
Arturo
X
```

guardar(correo, archivo)

```
arturo@fdimail.com_1426614458
1426614458
arturo@fdimail.com
ginebra@fdimail.com;lancelot@fdimail.com
Comprar mesa de reuniones
Hola,
he pensado que podíamos comprar una mesa de forma redonda para las reuniones. ¿Os parece buena idea?
Arturo
X
```

guardar(correo, archivo)

```
lancelot@fdimail.com_1426613678
1426613678
lancelot@fdimail.com
ginebra@fdimail.com;arturo@fdimail.com
Cazamos mañana?
Hola a todos!

Os parece buena idea salir a cazar mañana.
Lancelot
X
XXX
```

guardar(correo, archivo)



Correo. Depuración (I)

- ✓ Creamos el módulo Principal con el main
- ✓ Incluimos el módulo Correo
- ✓ Probamos en el main cada uno de los subprogramas de Correo



Correo. Depuración (II)

```
#include <iostream>
#include <fstream>

#include "Correo.h"

using namespace std;

int main(){
    // Probamos la creación de un nuevo correo
    cout << "-----CREAMOS UN CORREO NUEVO-----" << endl;
    tCorreo correo;
    correoNuevo(correo, "virginia@fdi.ucm.es");
    // Probamos que el mostrar funciona
    cout << endl << "-----MOSTRAMOS EL CORREO CREADO-----" << endl;
    cout << aCadena(correo) << endl;
    // Probamos que se crean correctamente un correo de contestacion
    cout << endl << "-----CREAMOS UN CORREO DE CONTESTACION-----" << endl;
    tCorreo correoCon;
    correoContestacion(correo, correoCon, "raquelhb@fdi.ucm.es");
    cout << endl << "-----MOSTRAMOS EL CORREO DE CONTESTACION CREADO-----" << endl;
    cout << aCadena(correoCon) << endl;
    // Probamos que funciona la obtención de cabecera
    cout << endl << "-----MOSTRAMOS LA CABECERA DE LOS DOS CORREOS CREADOS-----" << endl;
    cout << obtenerCabecera(correo) << endl;
    cout << obtenerCabecera(correoCon) << endl;
    // Probamos el cargar
    cout << endl << "-----PROBAMOS EL CARGAR-----" << endl;
    ifstream archivoEntrada;
    archivoEntrada.open("pruebaCorreo.txt");
    if (archivoEntrada.is_open())
        cargar(correo, archivoEntrada);
    cout << endl << "-----MOSTRAMOS EL CORREO CARGADO DESDE FICHERO-----" << endl;
    cout << aCadena(correo) << endl;
    // Probamos el guardar
    cout << endl << "-----PROBAMOS EL GUARDAR-----" << endl;
    ofstream archivoSalida;
    archivoSalida.open("pruebaCorreoGuardar.txt");
    guardar(correo, archivoSalida);
    cout << "Se ha guardado el correo en el archivo" << endl;
```

arturo@fdimail.com_1426614381
1426614381
arturo@fdimail.com
ginebra@fdimail.com
Mensaje de prueba
Hola,
Parece que ya tenemos correo en Camelot!
Arturo
X



Correo. Depuración (III)

```
-----CREAMOS UN CORREO NUEVO-----
Destinatario: jarroyo@fdi.ucm.es
Introduce el asunto (una línea): Prueba de creación de correo
Escribe el cuerpo del correo <XXX para terminar>:
Esto es una prueba
del gestor de correo
XXX

-----MOSTRAMOS EL CORREO CREADO-----
De: virginia@fdi.ucm.es 2015/4/8 <20:17:38>
Para: jarroyo@fdi.ucm.es
Asunto: Prueba de creación de correo
Esto es una prueba
del gestor de correo

-----CREAMOS UN CORREO DE CONTESTACION-----
Escribe el cuerpo del correo <XXX para terminar>:
Te reenvio el correo de prueba
para ver si esto funciona
XXX

-----MOSTRAMOS EL CORREO DE CONTESTACION CREADO-----
De: raquelhb@fdi.ucm.es 2015/4/8 <20:18:0>
Para: virginia@fdi.ucm.es
Asunto: Re: Prueba de creación de correo
Te reenvio el correo de prueba
para ver si esto funciona

-----
De: virginia@fdi.ucm.es 2015/4/8 <20:17:38>
Para: jarroyo@fdi.ucm.es
Asunto: Prueba de creación de correo
Esto es una prueba
del gestor de correo


-----MOSTRAMOS LA CABECERA DE LOS DOS CORREOS CREADOS-----
virginia@fdi.ucm.es Prueba de creación de correo 2015/4/8
raquelhb@fdi.ucm.es Re: Prueba de creación de correo 2015/4/8

-----PROBAMOS EL CARGAR-----
-----MOSTRAMOS EL CORREO CARGADO DESDE FICHERO-----
De: arturo@fdimail.com 2015/3/17 <18:46:21>
Para: ginebra@fdimail.com
Asunto: Mensaje de prueba
Hola,
Parece que ya tenemos correo en Camelot!
Arturo

-----PROBAMOS EL GUARDAR-----
Se ha guardado el correo en el archivo
```



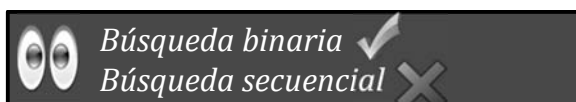
ListaCorreos (I)

- ✓ Tipos de datos:
 - tListaCorreos → Lista de tamaño variable de elementos de tipo tCorreo  Hay que incluir el módulo Correo
- ✓ Subprogramas:
 - void inicializar(tListaCorreos &correos)

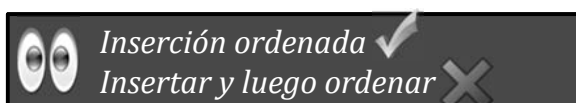


ListaCorreos (II)

- bool buscar(const tListaCorreos &correos, string id, int &pos):
 - ✓ Dado un identificador de correo y la lista, devuelve:
 - Si el identificador existe en la lista → true y su posición
 - Si el identificador no existe en la lista → false y la posición donde debería estar



- bool insertar(tListaCorreos &correos, const tCorreo &correo):
 - ✓ Inserta el correo en la lista correos de forma ordenada



ListaCorreos (III)

- `bool cargar(tListaCorreos &correos, string dominio):`
 - ✓ Abre el fichero `dominio_correos.txt`
 - ✓ Si se ha abierto correctamente:
 - ✓ Inicializa la lista de correos
 - ✓ Va leyendo uno a uno cada correo e insertándolo en la lista



Para cargar cada correo habrá que usar el subprograma `cargar` definido en el módulo `Correo`



Condiciones de parada:

- Centinela
- Superado el número máximo de elementos de la lista



ListaCorreos (IV)

- `void guardar(const tListaCorreos &correos, string dominio):`
 - ✓ Abre el fichero `dominio_correos.txt`
 - ✓ Se recorre la lista de correos y va guardando uno a uno cada correo



Para guardar cada correo habrá que usar el subprograma `guardar` definido en el módulo `Correo`

- `void ordenar_AF(tListaCorreos &correos):`
 - ✓ Ordena la lista correos por asunto y fecha



Como es una clave de ordenación doble, habrá que redefinir el operador de comparación en el módulo que corresponda

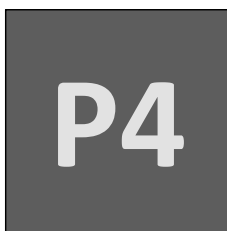


ListaCorreos. Depuración (I)

- ✓ Modificamos el módulo Principal para probar ahora el módulo ListaCorreos
- ✓ Incluimos el módulo ListaCorreos
- ✓ Probamos en el main cada uno de los subprogramas de ListaCorreos



Fundamentos de Programación



Práctica 4 Gestor de correo fdimail

Grado en Ingeniería Informática
Grado en Ingeniería del Software
Grado en Ingeniería de Computadores



Virginia Francisco Gilmartín
Facultad de Informática
Universidad Complutense

