8A Ejemplo de modularización

ANEXO

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

Luis Hernández Yáñez
Facultad de Informática
Universidad Complutense

Modularización de un programa

ventas.cpp

```
#include <iostream>
#include <string>
using namespace std;

const int NCLI = 100;
const int NPROD = 200;
const int NVENTAS = 3000;

typedef struct {
   int id_cli;
   string nif;
   string nombre;
   string telefono;
} tCliente;

typedef struct {
   tCliente clientes[NCLI];
   int cont;
} tListaClientes;

typedef struct {
   int id_prod;
   string codigo;
   string nombre;
   double precio;
   int unidades;
} tProducto;

typedef struct {
   tProducto productos[NPROD];
   int cont;
} tListaProductos;

typedef struct {
   int id;
   int id_prod;
   int id_cli;
   int unidades;
} tVenta;

typedef struct {
   tVenta ventas[NVENTAS];
   int cont;
} tListaVentas;

...
```


Modularización de un programa

```
tCliente nuevoCliente();
bool valida(tCliente cliente); // Función interna
bool operator<(tCliente opIzq, tCliente opDer); // Por NIF
void mostrar(tCliente cliente);
void inicializar(tListaClientes &lista);
void cargar(tListaClientes &lista);
void insertar(tListaClientes &lista, tCliente cliente, bool &ok);
void buscar(const tListaClientes &lista, string nif, tCliente &cliente, bool &ok);
void eliminar(tListaClientes &lista, string nif, bool &ok);
void mostrar(const tListaClientes &lista);
tProducto nuevoProducto();
bool valida(tProducto producto); // Función interna
bool operator<(tProducto opIzq, tProducto opDer); // Por código
void mostrar(tProducto producto);
void inicializar(tListaProductos &lista);
void cargar(tListaProductos &lista);
void insertar(tListaProductos &lista, tProducto producto, bool &ok);
void buscar(const tListaProductos &lista, string codigo, tProducto &producto,
            bool &ok);
void eliminar(tListaProductos &lista, string codigo, bool &ok);
...
```


Modularización de un programa

```
void mostrar(const tListaProductos &lista);
tVenta nuevaVenta(int id_prod, int id_cli, int unidades);
bool valida(tVenta venta); // Función interna
void mostrar(tVenta venta, const tListaClientes &clientes,
             const tListaProductos &productos);
void inicializar(tListaVentas &lista);
void cargar(tListaVentas &lista);
void insertar(tListaVentas &lista, tVenta venta, bool &ok);
void buscar(const tListaVentas &lista, int id, tVenta &venta, bool &ok);
void eliminar(tListaVentas &lista, int id, bool &ok);
void ventasPorClientes(const tListaVentas &lista);
void ventasPorProductos(const tListaVentas &lista);
double totalVentas(const tListaVentas &ventas, string nif,
                   const tListaClientes &clientes,
                   const tListaProductos &productos);
void stock(const tListaVentas &ventas, const tListaClientes &clientes,
           const tListaProductos &productos);
int menu();

int main() {
   ...
```


Estructuras de datos

```
#include <iostream>
#include <string>
using namespace std;

const int NCLI = 100;
const int NPROD = 200;
const int NVENTAS = 3000;

typedef struct {
   int id_cli;
   string nif;
   string nombre;
   string telefono;
} tCliente;

typedef struct {
   tCliente clientes[NCLI];
   int cont;
} tListaClientes;

typedef struct {
   int id_prod;
   string codigo;
   string nombre;
   double precio;
   int unidades;
} tProducto;

typedef struct {
   tProducto productos[NPROD];
   int cont;
} tListaProductos;

typedef struct {
   int id;
   int id_prod;
   int id_cli;
   int unidades;
} tVenta;

typedef struct {
   tVenta ventas[NVENTAS];
   int cont;
} tListaVentas;

...
```

Cliente

Lista de clientes

Producto

Lista de productos

Venta

Lista de ventas

Subprogramas de las estructuras de datos

```
tCliente nuevoCliente();
bool valida(tCliente cliente); // Función interna
bool operator<(tCliente opIzq, tCliente opDer); // Por NIF
void mostrar(tCliente cliente);

void inicializar(tListaClientes &lista);
void cargar(tListaClientes &lista);
void insertar(tListaClientes &lista, tCliente cliente, bool &ok);
void buscar(const tListaClientes &lista, string nif, tCliente &cliente,
            bool &ok);
void eliminar(tListaClientes &lista, string nif, bool &ok);
void mostrar(const tListaClientes &lista);

tProducto nuevoProducto();
bool valida(tProducto producto); // Función interna
bool operator<(tProducto opIzq, tProducto opDer); // Por código
void mostrar(tProducto producto);

...
```

Cliente

Lista de clientes

Producto

Subprogramas de las estructuras de datos

Lista de productos

```
void inicializar(tListaProductos &lista);
void cargar(tListaProductos &lista);
void insertar(tListaProductos &lista, tProducto producto, bool &ok);
void buscar(const tListaProductos &lista, string codigo, tProducto &producto,
            bool &ok);
void eliminar(tListaProductos &lista, string codigo, bool &ok);
void mostrar(const tListaProductos &lista);
```

Venta

```
tVenta nuevaVenta(int id_prod, int id_cli, int unidades);
bool valida(tVenta venta); // Función interna
void mostrar(tVenta venta, const tListaClientes &clientes,
             const tListaProductos &productos);
...
```


Subprogramas de las estructuras de datos

Lista de ventas

```
void inicializar(tListaVentas &lista);
void cargar(tListaVentas &lista);
void insertar(tListaVentas &lista, tVenta venta, bool &ok);
void buscar(const tListaVentas &lista, int id, tVenta &venta, bool &ok);
void eliminar(tListaVentas &lista, int id, bool &ok);
void ventasPorClientes(const tListaVentas &lista);
void ventasPorProductos(const tListaVentas &lista);
double totalVentas(const tListaVentas &ventas, string nif,
                   const tListaClientes &clientes,
                   const tListaProductos &productos);
void stock(const tListaVentas &ventas, const tListaClientes &clientes,
           const tListaProductos &productos);
int menu();

int main() {
   ...
```

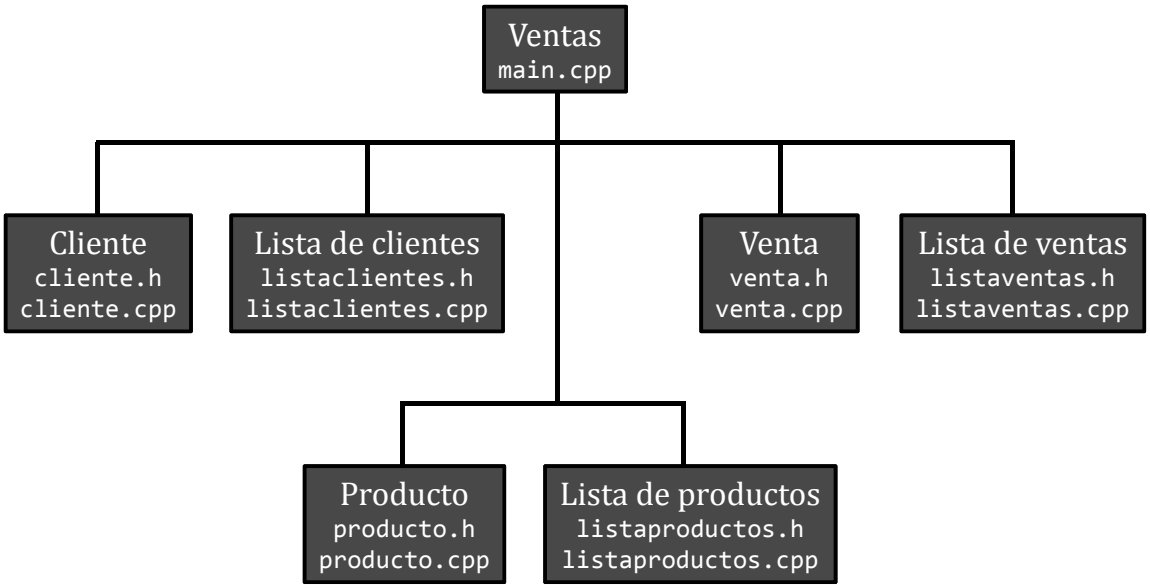

Módulos

- ✓ Cliente: `cliente.h` y `cliente.cpp`
- ✓ Lista de clientes: `listaclientes.h` y `listaclientes.cpp`
- ✓ Producto: `producto.h` y `producto.cpp`
- ✓ Lista de productos: `listaproductos.h` y `listaproductos.cpp`
- ✓ Venta: `venta.h` y `venta.cpp`
- ✓ Lista de ventas: `listaventas.h` y `listaventas.cpp`
- ✓ Programa principal: `main.cpp`

Distribución del código en los módulos:

- ✓ Declaraciones de tipos y datos en el archivo de cabecera (`.h`)
- ✓ Prototipos en el archivo de cabecera (`.h`) (excepto los de los subprogramas privados –internos–, que irán en el `.cpp`)
- ✓ Implementación de los subprogramas en el `.cpp`

Módulos

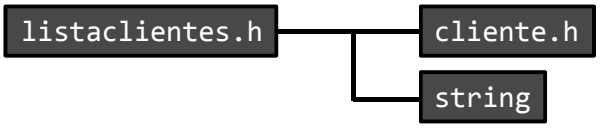


Dependencias entre módulos

Inclusiones (además de otras bibliotecas del sistema)

```
typedef struct {
   int id_cli;
   string nif;
   string nombre;
   string telefono;
} tCliente;
```

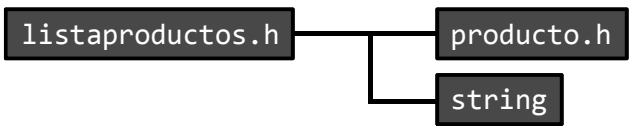

```
const int NCLI = 100;

typedef struct {
   tCliente clientes[NCLI];
   int cont;
} tListaClientes;

void buscar(const tListaClientes &lista, string nif, tCliente
            &cliente, bool &ok);
```



Dependencias entre módulos

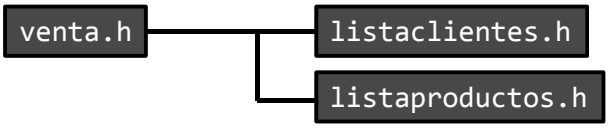
```
typedef struct {
   int id_prod;
   string codigo;
   string nombre;
   double precio;
   int unidades;
} tProducto;
```

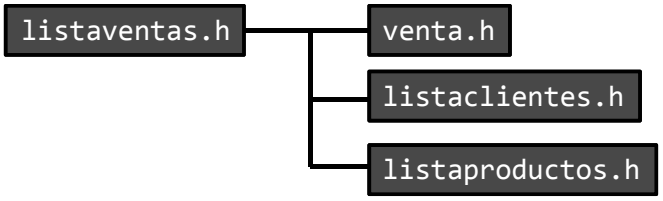

```
const int NPROD = 200;

typedef struct {
   tProducto productos[NPROD];
   int cont;
} tListaProductos;

void buscar(const tListaProductos &lista, string codigo, tProducto
            &producto, bool &ok);
```



Dependencias entre módulos

```
typedef struct {
   int id;
   int id_prod;
   int id_cli;
   int unidades;
} tVenta;

void mostrar(tVenta venta, const tListaClientes &clientes,
             const tListaProductos &productos);


const int NVENTAS = 3000;

typedef struct {
   tVenta ventas[NVENTAS];
   int cont;
} tListaVentas;

double totalVentas(const tListaVentas &ventas, string nif,
                   const tListaClientes &clientes,
                   const tListaProductos &productos);
```





Protección frente a inclusiones múltiples

```
#ifndef cliente_h
#define cliente_h

#include <string>
using namespace std;

typedef struct {
   int id_cli;
   string nif;
   string nombre;
   string telefono;
} tCliente;

tCliente nuevoCliente();
bool operator<(tCliente opIzq, tCliente opDer); // Por NIF
void mostrar(tCliente cliente);

#endif
```


Acerca de *Creative Commons*

Licencia CC (Creative Commons)

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:

- Reconocimiento (*Attribution*):
  En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
- No comercial (*Non commercial*):
  La explotación de la obra queda limitada a usos no comerciales.
- Compartir igual (*Share alike*):
  La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Pulsa en la imagen de arriba a la derecha para saber más.

Luis Hernández Yáñez