# Práctica 3:
# Ingredientes, Recetas y Dieta

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

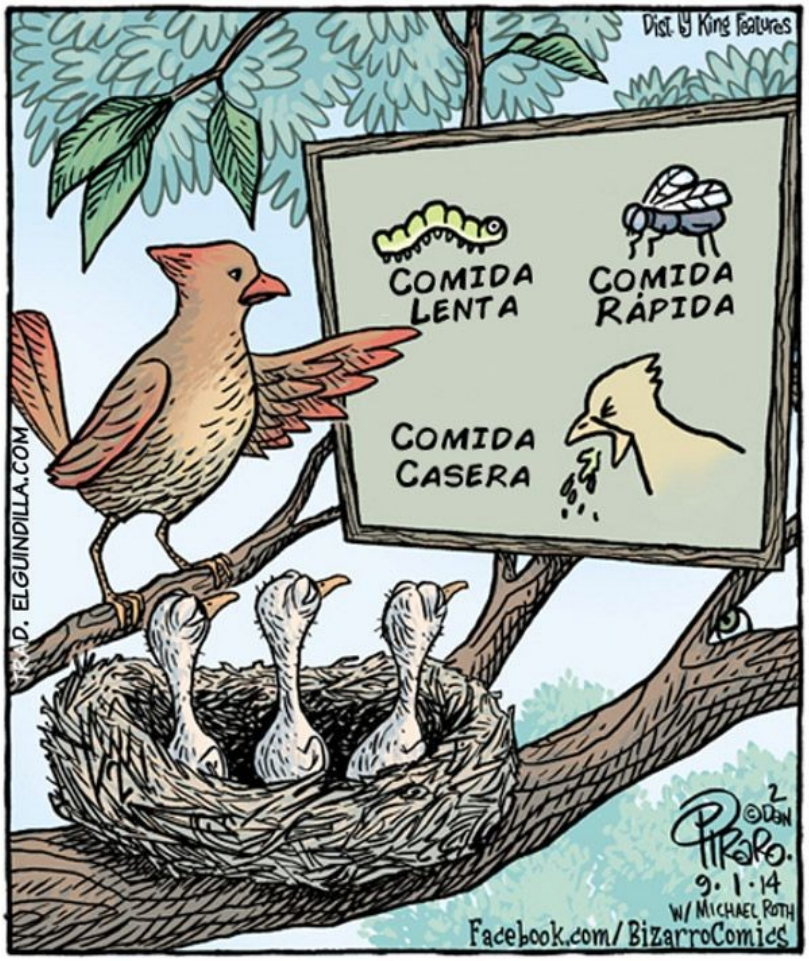
DECSAI

## Estructuras de Datos

Grado en Ingeniería Informática.
Doble Grado en Informática y ADE
Doble Grado en Informática y Matemáticas

# Índice de contenido





# 1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

1. Practicar con T.D.A implementados en la STL
2. Resolver un problema eligiendo las mejores estructuras de datos para las operaciones que se solicitan
3. Seguir asimilando los conceptos de documentación de un tipo de dato abstracto (T.D.A)

Los requisitos para poder realizar esta práctica son:

1. Haber estudiado el Tema 1: Introducción a la eficiencia de los algoritmos
2. Haber estudiado el Tema 2: Abstracción de datos. Templates.
3. Haber estudiado el Tema 3: T.D.A. Lineales.
4. Haber estudiado el Tema 4: STL e Iteradores.
5. Aunque no es un requisito indispensable, haber realizado la práctica 2 ayudará a entender mejor el problema.

# 2. STL

EL objetivo en esta práctica es hacer uso de la librería STL en C++ http://www.cplusplus.com/reference/stl/. Con tal objetivo vamos a centrarnos en resolver un problema con dicha librería. Previo al uso de esta librería, como en la anterior práctica se requiere que el alumno diseñe nuevos T.D.A eficientes para resolver el problema.

# 3. Ejercicio

Esta práctica tendrá un doble objetivo dado un conjunto de recetas de cocina. El primer objetivo será obtener la cantidad de nutrientes (calorías, hidratos de carbono, proteínas, grasas y fibra) que contiene una receta. Y en segundo lugar fijada una cantidad de calorías seleccionar las recetas que, sin sobrepasar esa cantidad de calorías, acumulan la mayor razón proteína hidrato de carbono.

Pero antes de abordar este problema vamos a retomar los TDA definidos en la práctica 2 y los vamos a redefinir. Así las tareas que debemos abordar son las siguientes:

1. Modificar el TDA Ingredientes
2. Probar que funciona con test_ingredientes (ver test_ingredientes.cpp situado en el directorio src del material). Este fichero no debe modificarse.
3. Crear el TDA Receta
4. Crear el TDA Recetas
5. Probar que funciona con test_recetas (ver test_recetas.cpp situado en el directorio src del material). Este fichero no debe modificarse.
6. Crear el programa ***nutricion_receta***

## 3.1. TDA Ingredientes

Un objeto del TDA Ingredientes contiene un conjunto de ingredientes. Como se vio en la práctica 2 cada ingrediente se describe por el nombre, calorías, hidratos de carbono,

proteínas, grasas, fibra y tipo de alimento.

En esta práctica vamos a modificar TDA Ingredientes para que la representación ahora use un vector de la STL. Modificar la implementación de acuerdo a esta nueva representación. Así la clase ingrediente tendrá como mínimo:

```
#include “ingrediente.h”
#include <vector>
using namespace std;
class ingredientes{
private:
  vector<ingrediente> datos;
  vector<int> indice_tipo;
  ...
```

Además vamos a añadir a la clase ingredientes dos iteradores uno no constante y otro constante que itere por los ingredientes (ordenadas por nombre).

De forma que nuestra clase será algo parecido a :

```
#include “ingrediente.h”
#include <vector>
using namespace std;
class ingredientes{
private:
  vector<ingrediente> datos;
  vector<int> indice_tipo;
  ...
public:
…

   class iterator{
       private:
          vector<ingrediente>::iterator it;
    …
    };
```

```
   class const_iterator{
       private:
        vector<ingrediente>::const_iterator it;
    …
    }:
    iterator begin();
    iterator end();
    const_iterator begin()const;
    const_iterator end()const;

};//end ingredientes
```

## 3.2. Programa test_ingredientes

El TDA Ingrediente e Ingrediente tiene que funcionar con el el fichero test_ingredientes.cpp. Para ello debe generar el programa test_ingredientes. Un ejemplo de llamada podría ser:

```
prompt>bin/test_ingredientes datos/ingredientes.txt
```

Este programa recibe un fichero con los ingredientes. Con estos datos testea los TDA Ingredientes e Ingrediente, además de los iteradores previamente comentados. Si visualizamos el fichero test_ingredientes.cpp el alumno/a verá seis partes diferenciadas:

- ***Sección 1***: En esta sección se comprueba la declaración, lectura, consulta y escritura del TDA ***ingrediente.*** El alumno debería también proporcionar operadores de modificación para cada uno de los atributos de un objeto ingrediente.
- ***Sección 2:*** En esta sección se comprueba la declaración, lectura y escritura del TDA ***ingredientes***. La lectura hará uso de los métodos de inserción. Hay que tener en cuenta que los ficheros de entrada no están ordenados. Y por lo tanto cada vez que se inserta un ingrediente se debe insertar de forma ordenada por nombre, y simultáneamente mantener la ordenación por tipo y nombre.
- ***Sección 3:*** Sobre el TDA ***ingredientes*** debemos mantener también la ordenación por tipo de alimento y nombre. En esta sección se comprueba que dicha ordenación se ha

realizado correctamente.

- ***Sección 4:*** En la cuarta sección se consulta la información de un ingrediente dando el nombre. Además se borra este alimento y se vuelve a listar los ingredientes. El alumno debe comprobar que tras el borrado la ordenación, tanto por nombre, como por tipo y nombre es correcta.
- ***Sección 5:*** En esta sección se obtiene los tipos diferentes de ingredientes que existen. Además se selecciona solamente los ingredientes de un tipo para a continuación mostrarlos.
- ***Sección 6***.- Se testea los iteradores. Con tal fin se imprime un objeto de tipo ingredientes de orden de menor a mayor nombre (alfabéticamente) y en el sentido contrario.

## 3.3. TDA Receta

El alumno/a creará el TDA Receta que representa a una receta de cocina. Los datos sobre una receta de cocina son:

- un código (único),
- un número entre 1-3 indicando si es un primer, segundo plato o postre.
- El nombre de la receta. Por ejemplo "***Salmorejo***"
- Una secuencia de pares de nombre de ingrediente y cantidad (en gramos).
- Además la receta mantiene el total de calorías, hidratos de carbono, grasas, proteínas y fibra que aporta

La representación que vamos a dar para el TDA **Receta** es:

```
#include <string>
#include  <list>
#include <iostream>
using namespace std;
class receta{
private:
  string code;
  unsigned int plato;
  string nombre;
  list<pair<string,unsigned int> > ings;
  float calorias,hc,grasas,proteinas,fibra;
  ...
public:
…

   class iterator{
 private:
 list<pair<string,unsigned int> >::iterator it;
    …
    };
   class const_iterator{
       private:
           list<pair<string,unsigned
int> ::const_iterator it;
    …
    }:
    iterator begin();
    iterator end();
    const_iterator begin()const;
    const_iterator end()const;

};//end receta
```

Como se puede observar hemos definido para receta dos iteradores uno no constante y otro constante, que itera por los pares nombre de ingrediente cantidad.

## 3.4. T.D.A Recetas

Un objeto del TDA Recetas representa un conjunto no repetido de objetos de tipo Receta. Vamos a elegir una representación que sea lo más eficiente en tiempo a la hora de hacer consultas de una receta concreta, insertar una nueva receta y borrar una receta. Con tal fin usaremos la siguiente representación:

```
#include “receta.h”
#include <string>
#include <map>
using namespace std;
class recetas{
private:
  map<string,receta> datos
  ...
public:
…

   class iterator{
       private:
          map<string,receta>::iterator it;
          …
          };
          class const_iterator{
              private:
                 map<string,receta>::const_iterator
it;
          …
          }:
          iterator begin();
          iterator end();
          const_iterator begin()const;
          const_iterator end()const;

};//end recetas
```

Se usará un mapa en el que el código de receta será la clave y la receta será la información asociada. Pensad en el conjunto de operaciones necesarias para poder interactuar con objetos de tipo recetas. Además implementaremos dos iteradores uno constante y otro no constante que permitirá iterar sobre los objetos de tipo receta almacenados en el objeto recetas. Las clases ***receta*** y ***recetas*** deberá funcionar con el programa test_receta.

## 3.5. Programa test_receta

Nuestros TDA receta y recetas debe funcionar con el código en test_receta.cpp. Para ello generaremos el programa test_recetas. Una posible ejecución desde la línea de ordenes podría ser la siguiente:

```
prompt>bin/test_receta datos/recetas.txt
```

Este programa recibe un fichero con un conjunto de recetas (ver sección 3.8).

Con estos datos testeamos los TDA receta y recetas, además de los iteradores previamente comentados. Si visualizamos el fichero test_recetas.cpp el alumno/a verá cinco partes diferenciadas:

- ***Sección 1***: Se comprueba la declaración, operadores de lectura y escritura de un objeto de tipo receta.
- ***Sección 2***: Se comprueba el buen funcionamiento de el iterador asociado a receta. Además comprobamos que el número de ingredientes de la receta son los que se han imprimido.
- ***Sección 3***: Se comprueba la declaración, lectura y escritura de un objeto de tipo recetas (*rall*). Además en el proceso de lectura se comprobará que la lectura de receta es correcta además del método de inserción de recetas
- ***Sección 4:*** Consulta de la información de una receta concreta. Y comprobación del operador de salida de receta.
- ***Sección 5:*** Borrado de una receta y comprobación de que el borrado es correcto. Para ello se imprime las recetas en orden alfabético (de forma creciente ) por código. Para la impresión usaremos el iterador definido en el TDA Recetas.

## 3.6. Programa nutricion_receta

El alumno/a deberá crear un programa con los siguiente objetivos:

1. Dado un conjunto de ingredientes y un conjunto de recetas obtener la cantidad de nutrientes esenciales de cada receta. Entendemos por nutrientes esenciales calorías, hidratos de carbono, proteínas, grasas, fibra.
2. A partir de ese conjunto de recetas obtener un subconjunto de recetas que maximizan

la acumulación de razón proteína hidratos de carbono sin superar un número de calorías.

Para ello creará el programa nutrición_receta. Una posible llamada desde la línea de órdenes es la siguiente:

```
prompt>bin/nutricion_receta datos/recetas.txt datos/ingredientes.txt 500
```

Los parámetros dados son:

1. El fichero con las recetas
2. El fichero con los ingredientes
3. Un número de calorías que no se debe superar al escoger las recetas que acumulan la mayor razón de proteína hidratos de carbonos

En primer lugar al ejecutar el programa deberá aparece el conjunto de recetas con sus valores nutricionales. Por ejemplo:

**Valores Nutricionales de la receta** ***R1;1;Ensalada Mixta;Lechuga 200;Tomate 50;Pepino 50;Cebolla 50;Aceite Oliva 5;Vinagre De Vino 10;Sal 1 son :***

**Calorias** 92.85 **Hidratos de Carb.** 7.1 **Proteinas** 3.5 **Grasas** 5 **Fibra** 5.5

**Valores Nutricionales de la receta** ***R10;1;Cocido;Garbanzos 200;Zanahoria 50;Judia Verde 50;Calabaza 100;Patata 100;Sal 1;Agua 2000 son :***

**Calorias** 939.5 **Hidratos de Carb.** 215.5 **Proteinas** 58.1 **Grasas** 28.65 **Fibra** 39

**Valores Nutricionales de la receta** ***R11;2;Merluza al Horno;Merluza 200;Ajo 25;Aceite Oliva 10;Limon 25;Perejil 10;Sal 1 son :***

**Calorias** 394 **Hidratos de Carb.** 9.75 **Proteinas** 138.25 **Grasas** 72 **Fibra** 2.15

**Valores Nutricionales de la receta** ***R12;2;Tofu a la Pimienta;Tofu 200;Nata Liquida para Cocinar 200;Pimienta 10;Aceite Oliva 15 son :***

**Calorias** 322.85 **Hidratos de Carb.** 42 **Proteinas** 84 **Grasas** 289 **Fibra** 0

...

A continuación el programa debe descubrir que subconjunto de recetas del conjunto total obtiene el mayor valor acumulado de la razón proteína hidrato de carbono sin superar el número de calorías dada por la línea de comandos. En el ejemplo el número de calorías máximo sería 500. Una posible solución es:

**Las recetas escogidas son:**

R1;1;Ensalada Mixta;Lechuga 200;Tomate 50;Pepino 50;Cebolla 50;Aceite Oliva 5;Vinagre De Vino 10;Sal 1

R11;2;Merluza al Horno;Merluza 200;Ajo 25;Aceite Oliva 10;Limon 25;Perejil 10;Sal 1

**Calorias Totales** 486.85 **Proteinas Totales** 141.75

## 3.7. Fichero con los ingredientes

El fichero con los ingredientes está compuesto de una serie de líneas. Cada línea se corresponde con la información de un ingrediente

```
Alimento (100 gramos);Calorias (Kcal.);Hidratos de Carb.;Proteinas;Grasas;Fibra;Tipo
Ketchup;98;24;2;0;0;Procesado
Salmon;182;0;18;12;0;Pescado
Cereales Cornflakes;368;85;9;2;11;Cereal
Gallo;81;0;17;1;0;Carne
Queso Emmental;377;0;29;29;0;Leche-Derivados
Cerezas;47;12;0;2;0;Fruta
Gatorade;39;10;0;0;0;Bebida
Salsa Ketchup;98;24;2;0;0;Procesado
Esparragos Enlatados;14;1;2;0;0;Verdura
Endibias;11;1;2;0;0;Verdura
Harina Trigo, Panificada;337;75;11;1;3;Cereal
Nuez Brasil;617;4;12;62;9;Frutos Secos
Gofio Millo;377;83;6;5;0;Cereal
Pavo;107;0;22;2;0;Carne
Mantequilla;740;0;0;82;0;Aceite
Yogur Liquido;78;11;3;2;0;Leche-Derivados
Cacao Polvo;357;11;20;24;38;Semillas
Arenque Ahumado;205;0;26;11;0;Pescado
….
```

El fichero contiene:

1. En la primera línea es un comentario. Indicando que atributos tendrá cada uno de los alimentos.
2. A continuación en cada línea viene la información de cada ingrediente. Cada atributo de un ingrediente se encuentra separada por “;”. Los atributos son
    - Nombre de ingrediente (puede tener más de una palabra)
    - Calorías por cada 100 gramos del ingrediente
    - Porcentaje de hidratos de carbono
    - Porcentaje de proteínas
    - Porcentaje de grasas
    - Porcentaje de Fibra
    - Tipo de ingrediente

## 3.8. Fichero con las Recetas

El fichero con las recetas contiene una línea por receta. Cada receta se describe por:

1. Un código de receta. A continuación un carácter ';'
2. Un número indicando si es un primer, segundo o tercer plato.
3. El nombre de la receta
4. Y un conjunto de pares separados ';'. Cada par se compone del nombre de un ingrediente y un número que indica la cantidad del ingrediente en gramos.

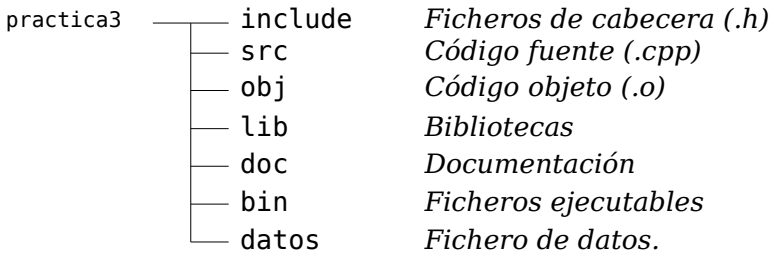
Un ejemplo de fichero de recetas es el siguiente:

```
R1;1;Ensalada Mixta;Lechuga 200;Tomate 50;Pepino 50;Cebolla 50;Aceite Oliva 5;Vinagre De Vino 10;Sal 1
R2;1;Ensaladilla Rusa;Patata cocida 100;Zanahoria 50;Judia Verde 50;Huevo Duro 100;Mayonesa 125;Sal 1
R3;1;Revuelto de Acelgas;Acelgas 250;Huevo 50;Ajo 4;Pimiento 15;Aceite Oliva 25;Sal 1
R4;1;Salteado de Brocoli con Jamon;Jamon Serrano 100;Brocoli 500;Ajo 20;Aceite Oliva 20;Sal 1
R5;1;Crema de Calabacin;Patata 250;Pimiento 100;Cebolla 200;Calabacin 250;Aceite Oliva 20;Queso Quark
50;Sal 1
R6;1;Salmorejo;Huevo Cocido 50;Tomate 125;Vinagre De Vino 5;Aceite Oliva 5;Ajo 5;Sal 1;Pan Blanco 75
R7;1;Ajo Blanco;Almendras 100;Ajo 10;Pan Blanco 50;Agua 1000;Aceite Oliva 15;Vinagre De Vino 20;S
```

## 4. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre "practica3.*tgz*" y entregarlo antes de la fecha que se publicará en la página web de la asignatura ( en PRADO). Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables, ni la carpeta datos. Es recomendable que haga una "limpieza" para eliminar los archivos temporales o que se puedan generar a partir de los fuentes.

El alumno debe incluir el archivo *Makefile* para realizar la compilación. Tenga en cuenta que los archivos deben estar distribuidos en directorios*:*




Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior (en el mismo nivel de la carpeta "*practica3*") para ejecutar:

```
prompt% tar zcv practica3.tgz practica3
```

*tras lo cual, dispondrá de un nuevo archivo practica3.tgz que contiene la carpeta practica3 así como todas las carpetas y archivos que cuelgan de ella.*