TIPOS INGREDIENTES

Generado por Doxygen 1.8.13

Índice general

Capítulo 1

Índice de clases

1.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 2

Documentación de las clases

2.1. Referencia de la Clase ingrediente

Métodos públicos

- ingrediente ()

  Constructor por defecto.
- ingrediente (string nombr, double cal, double hc, double prot, double gras, double fibr, string tip)

  Constructor con parámetros.
- string getNombre () const

  Devuelve el nombre del ingrediente que llama a la función.
- double getCalorias () const

  Devuelve las calorias por cada 100 gramos del ingrediente que llama a la función.
- double getHc () const

  Devuelve los carbohidratos por cada 100 gramos del ingrediente que llama a la función.
- double getProteinas () const

  Devuelve las proteinas por cada 100 gramos del ingrediente que llama a la función.
- double getGrasas () const

  Devuelve las grasas por cada 100 gramos del ingrediente que llama a la funcion.
- double getFibra () const

  Devuelve la fibras por cada 100 gramos del ingrediente que llama a la funcion.
- string getTipo () const

  Devuelve el tipo de alimento del ingrediente que llama a la función.
- void setNombre (string nombr)

  Establece el nombre del ingrediente que llama a la función, siendo el nuevo nombre nombr.
- void setCalorias (double cal)

  Establece las calorias por cada 100 gramos del ingrediente que llama a la funcion.
- void setHc (double hc)

  Establece los carbohidratos por cada 100 gramos del ingrediente que llama a la funcion.
- void setProteinas (double prot)

  Establece las proteinas por cada 100 gramos del ingrediente que llama a la funcion.
- void setGrasas (double gras)

  Establece las grasas por cada 100 gramos del ingrediente que llama a la funcion.
- void setFibra (double fibr)

  Establece la fibra por cada 100 gramos del ingrediente que llama a la funcion.
- void setTipo (string tip)

  Establece al grupo de alimentos que pertenecerá el alimento : tipo de alimento.
- ingrediente & operator= (const ingrediente &original)

  Sobrecargar el operador de asignacion para la clase.

Amigas

- std::ostream & operator<< (std::ostream &os, const ingrediente &i)

  Devuelve la información almacenada en un ingrediente i por medio del flujo de salida os.
- std::istream & operator>> (std::istream &is, ingrediente &i)

  Toma la información por medio del flujo de entrada is y se almacena en el ingrediente i.

2.1.1. Descripción detallada

Definición en la línea 9 del archivo ingrediente.h.

2.1.2. Documentación del constructor y destructor

2.1.2.1. ingrediente()

```
ingrediente::ingrediente (
             string nombr,
             double cal,
             double hc,
             double prot,
             double gras,
             double fibr,
             string tip )
```

Constructor con parámetros.

Parámetros

nombr	nombre que se le da al nuevo elemento de la clase ingrediente
cal	calorias por cada 100 gramos que se le dan al ingrediente
hc	carbohidratos por cada 100 gramos que se le dan al ingrediente
prot	proteinas por cada 100 gramos que se le dan al ingrediente
gras	grasas por cada 100 gramos que se le dan al ingrediente
fibr	fibra por cada 100 gramos que se le dan al ingrediente
tip	tipo de alimentos al que pertenecerá el ingrediente

Definición en la línea 20 del archivo ingrediente.cpp.

```
20
        {
21     nombre = nombr;
22     calorias = cal;
23     Hc = hc;
24     proteinas = prot;
25     grasas = gras;
26     fibra = fibr;
27     tipo = tip;
28 }
```

2.1.3. Documentación de las funciones miembro

2.1.3.1. operator=()

```
ingrediente & ingrediente::operator= (
            const ingrediente & original )
```

Sobrecargar el operador de asignacion para la clase.

Parámetros

original	el ingrediente que se quiere copiar

Definición en la línea 137 del archivo ingrediente.cpp.

```
137                                                         {
138     nombre=original.nombre;
139     calorias=original.calorias;
140     Hc=original.Hc;
141     proteinas=original.proteinas;
142     grasas=original.grasas;
143     fibra=original.fibra;
144     tipo=original.tipo;
145
146         return *this;
147 }
```

2.1.3.2. setCalorias()

```
void ingrediente::setCalorias (
            double cal )
```

Establece las calorias por cada 100 gramos del ingrediente que llama a la funcion.

Parámetros

cal	numero de calorias del ingrediente por cada 100 gramos

Definición en la línea 62 del archivo ingrediente.cpp.

```
62                                      {
63     calorias = cal;
64 }
```

2.1.3.3. setFibra()

```
void ingrediente::setFibra (
            double fibr )
```

Establece la fibra por cada 100 gramos del ingrediente que llama a la funcion.

Parámetros

fibr	numero de fibra del ingrediente por cada 100 gramos

Definición en la línea 78 del archivo ingrediente.cpp.

```
78                                      {
79     fibra = fibr;
80 }
```

2.1.3.4. setGrasas()

```
void ingrediente::setGrasas (
            double gras )
```

Establece las grasas por cada 100 gramos del ingrediente que llama a la funcion.

Parámetros

gras	numero de grasas del ingrediente por cada 100 gramos

Definición en la línea 74 del archivo ingrediente.cpp.

```
74                                      {
75     grasas = gras;
76 }
```

2.1.3.5. setHc()

```
void ingrediente::setHc (
            double hc )
```

Establece los carbohidratos por cada 100 gramos del ingrediente que llama a la funcion.

Parámetros

hc	numero de carbohidratos del ingrediente por cada 100 gramos

Definición en la línea 66 del archivo ingrediente.cpp.

```
66                                  {
67     Hc= hc;
68 }
```

2.1.3.6. setNombre()

```
void ingrediente::setNombre (
             string nombr )
```

Establece el nombre del ingrediente que llama a la función, siendo el nuevo nombre nombr.

Parámetros

nombr	nuevo nombre del ingrediente

Definición en la línea 58 del archivo ingrediente.cpp.

```
58                                      {
59     nombre = nombr;
60 }
```

2.1.3.7. setProteinas()

```
void ingrediente::setProteinas (
             double prot )
```

Establece las proteinas por cada 100 gramos del ingrediente que llama a la funcion.

Parámetros

prot	numero de proteinas del ingrediente por cada 100 gramos

Definición en la línea 70 del archivo ingrediente.cpp.

```
70                                       {
71     proteinas = prot;
72 }
```

2.1.4. Documentación de las funciones relacionadas y clases amigas

2.1.4.1. operator<<

```
std::ostream& operator<< (
             std::ostream & os,
             const ingrediente & i )  [friend]
```

Devuelve la información almacenada en un ingrediente i por medio del flujo de salida os.

Parámetros

os:flujo	de salida
i:ingrediente	el cuya información se quiere guardar en archivo o mostrar por pantalla

Definición en la línea 87 del archivo ingrediente.cpp.

```
87                                                                  {
88     os « i.getNombre() « ";" « i.getCalorias() « ";" « i.
      getHc() « ";" « i.getProteinas() « ";" « i.getGrasas() « ";" «  i.
      getFibra() « ";" « i.getTipo() « endl;
89
90     return os;
91 }
```

2.1.4.2. operator>>

```
std::istream& operator>> (
             std::istream & is,
             ingrediente & i )  [friend]
```

Toma la información por medio del flujo de entrada is y se almacena en el ingrediente i.

Parámetros

is	flujo de entrada
i	ingrediente donde se almacenará la información

Definición en la línea 93 del archivo ingrediente.cpp.

```
93                                                                  {
94     string cadena;
95     string delim=";";
96     int punt_com[6];
97     int pos_ini=0;
98     int contador=0;
99
100     getline(is,cadena);
101     if(is){
102
103         for(int i=0; i < 6; i++){
104             punt_com[i]=0;
105         }
106
107
108
109         while(cadena.find(delim,pos_ini)!=std::string::npos){
110             pos_ini=cadena.find(delim,pos_ini);
```

```
111             punt_com[contador]=pos_ini;
112             contador++;
113             pos_ini++;
114         }
115
116         for(int i=0; i <= 6; i++){
117             if(i==0)
118                 entrada.nombre=cadena.substr(0,punt_com[i]);
119             if(i==1)
120                 entrada.calorias=atof(cadena.substr(punt_com[i-1]+1,punt_com[i]).c_str());
121             if(i==2)
122                 entrada.Hc=atof(cadena.substr(punt_com[i-1]+1,punt_com[i]).c_str());
123             if(i==3)
124                 entrada.proteinas=atof(cadena.substr(punt_com[i-1]+1,punt_com[i]).c_str());
125             if(i==4)
126                 entrada.grasas=atof(cadena.substr(punt_com[i-1]+1,punt_com[i]).c_str());
127             if(i==5)
128                 entrada.fibra=atof(cadena.substr(punt_com[i-1]+1,punt_com[i]).c_str());
129             if(i==6)
130                 entrada.tipo=(cadena.substr(punt_com[i-1]+1,cadena.size()));
131         }
132     }
133
134     return is;
135 }
```

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- include/ingrediente.h
- src/ingrediente.cpp

2.2. Referencia de la Clase ingredientes

Métodos públicos

- ingredientes ()

  Constructor sin parametros.
- void Insertar (const ingrediente &insertado)

  Inserta un ingrediente en la posicion pos.
- int size () const

  Devuelve el numero de datos contenidos en el vector dinamico ordenado por nombre.
- void borrar (const string &nombre)

  Borra el ingrediente con el nombre indicado.
- ingrediente Obtener_ingrediente (int pos)

  Devuelve el ingrediente que se encuentra en la posicion pos del vector dinámico.
- void setIngrediente (int pos, const ingrediente &nuevo)

  Modifica el contenido de la posicion pos del vector de ingredientes por el ingrediente nuevo.
- ingrediente get (const string &n)

  Devuelve la información de un ingrediente dado su nombre.
- ingredientes getIngredienteTipo (const string &tipo)

  Devuelve todos los ingredientes de un tipo tipo.
- pair< bool, int > Buscar_nombre (const ingrediente &insertar)

  Busca si el ingrediente a insertar pertenece o no al vector dinamico, y en que posicion se debería insertar.
- pair< bool, int > Buscar_tipo (const ingrediente &insertar)

  Consulta y/o modifica la posicion i del vector dinamico de ingredientes.
- ingrediente & operator[ ] (int i)

  Para acceder a la posición i-ésima del dato miembro datos.
- ingrediente & operator[ ] (int i) const

  Se trata del operador [] constante que sirve para obtener la posición i-esima de datos.

- ingredientes operator= (const ingredientes &original)

  Sobrecarga Operador de asignación. Asigna a una variable ingrediente un dato de tipo ingrediente.
- void ImprimirPorTipo (std::ostream &os)

  Se imprimen los ingredientes del vector dinamico por tipo.
- VD< string > getTipos ()

  Se devuelve un vector dinamico de string con todos los tipos almacenados.

Amigas

- std::ostream & operator<< (std::ostream &os, const ingredientes &i)

  Se almacenan los datos de i en el fichero al que apunte el flujo de salida os.
- std::istream & operator>> (std::istream &is, ingredientes &i)

  Se almacenan los datos contenidos en el fichero al que apunta el flujo de entrada is en la variable i.

2.2.1. Descripción detallada

Definición en la línea 12 del archivo ingredientes.h.

2.2.2. Documentación de las funciones miembro

2.2.2.1. borrar()

```
void ingredientes::borrar (
            const string & nombre )
```

Borra el ingrediente con el nombre indicado.

Parámetros

pos:posicion	cuyo ingrediente se va a eliminar

Definición en la línea 103 del archivo ingredientes.cpp.

```
103                                             {
104     bool encontrado=false;
105
106     for(int i=0; i < datos.size() && !encontrado; i++){
107         if(datos[i].getNombre()==nombre){
108             datos.Borrar(i);
109             for (int j=0; j < indice.size();j++){
110                 if(indice[j]>i)
111                     indice[j]--;
112                 else if(indice[j]==i)
113                     indice.Borrar(j);
114             }
115             encontrado=true;
116         }
117     }
118
119     if(!encontrado)
120         cerr « "ERROR: el nombre introducido no es encuentra en datos." « endl;
121 }
```

2.2.2.2. Buscar_nombre()

```
pair< bool, int > ingredientes::Buscar_nombre (
            const ingrediente & insertar )
```

Busca si el ingrediente a insertar pertenece o no al vector dinamico, y en que posicion se debería insertar.

Parámetros

insertar	ingrediente que se comprueba si pertenece y en que posicion se va a insertar

Devuelve

pair<boo.,int> donde bool indica si esta o no en el vector dinámico y int en que posición se insertaría

Definición en la línea 10 del archivo ingredientes.cpp.

```
10                                                                  {
11     int inf=0;
12     int sup=datos.size()-1;
13     int mitad=0;
14     pair<bool,int> salida(false,0);
15
16     while(inf <= sup && !salida.first){
17         mitad=(inf+sup)/2;
18
19         if(datos[mitad].getNombre()==insertar.getNombre()&&datos[mitad].getTipo()==insertar.
       getTipo()){
20             salida.first=true;
21             salida.second=mitad;
22
23         }
24
25         else{
26             if(datos[mitad].getNombre()<insertar.getNombre())
27                 inf=mitad+1;
28             else
29                 sup=mitad-1;
30
31             salida.second=inf;
32         }
33
34
35     }
36
37     return salida;
38 }
```

Gráfico de llamadas para esta función:

2.3. Referencia de la plantilla de la Clase VD< T >

Métodos públicos

- VD (int tam=10)

  Constructor por defecto y con parametro.
- VD (const VD< T > &original)

Constructor de copia.

- ∼VD ()

  Destructor. Elimina la memoria asociada al vector dinámico.

- VD< T > & operator= (const VD< T > &v)

  Operador de asignacion.

- int size () const

  Devuelve el numero de datos almacenados en el vector dinamico.

- T & operator[ ] (int i)

  Consulta y modifica el elemento i-esimo.

- T & **operator[ ]** (int i) const
- void Insertar (const T dato_insertar, const int indice_insertar)

  Inserta un objeto en la posicion pos del vector dinámico.

- void Borrar (const int indice_borrar)

  Elimimina el elemento en la posición pos del vector dinamico.

2.3.1. Descripción detallada

template<class T>
class VD< T >

Definición en la línea 5 del archivo VD.h.

2.3.2. Documentación del constructor y destructor

2.3.2.1. VD() [1/2]

```
template<class T >
VD< T >::VD (
            int tam = 10 )
```

Constructor por defecto y con parametro.

Parámetros

tam:elementos	a reservar para el vector dinámico

Nota

si no se proporciona un valor para tam se tomara como 10

Definición en la línea 50 del archivo VD.cpp.

```
50                {
51
52     reservados = tam;
53     datos = new T[reservados];
54     n = 0;
55
56 }
```

2.3.2.2. VD() [2/2]

```
template<class T>
VD< T >::VD (
            const VD< T > & original )
```

Constructor de copia.

Parámetros

original	vector dinamico original que se va a copiar

Definición en la línea 59 del archivo VD.cpp.

```
59                              {
60
61     Copiar(original);
62 }
```

2.3.3. Documentación de las funciones miembro

2.3.3.1. Borrar()

```
template<class T >
void VD< T >::Borrar (
            const int indice_borrar )
```

Elimimina el elemento en la posición pos del vector dinamico.

Parámetros

pos	posicon del elemento a borrar

Definición en la línea 98 del archivo VD.cpp.

```
98                                          {
99
100     for(int i= indice_borrar; i < n-1; i++)
101         datos[i] = datos[i+1];
102
103     n--;
104
105     if(n < (reservados/4))
106         resize(reservados/2);
107 }
```

2.3.3.2. Insertar()

```
template<class T>
void VD< T >::Insertar (
             const T dato_insertar,
             const int indice_insertar )
```

Inserta un objeto en la posicion pos del vector dinámico.

Parámetros

d	objeto a insertar
pos	posicion donde insertar.

Precondición

pos debe estar comprendido entre 0 y size()

Postcondición

aumenta en uno el vector dinamico

Definición en la línea 84 del archivo VD.cpp.

```
84                                                              {
85
86     if (n >= (reservados/2))
87         resize(2*reservados);
88
89     for(int i=n; i > indice_insertar; i--)
90         datos[i] = datos[i-1];
91
92     datos[indice_insertar] = dato_insertar;
93     n++;
94
95 }
```

2.3.3.3. operator=()

```
template<class T>
VD< T > & VD< T >::operator= (
             const VD< T > & v )
```

Operador de asignacion.

Parámetros

v	vector dinámico fuente

Devuelve

una referencia al objeto al que apunta this

Definición en la línea 72 del archivo VD.cpp.

```
72                                      {
73
74      if(this != &v){
75
76          Liberar();
77          Copiar(v);
78      }
79
80      return *this;
81  }
```

2.3.3.4. operator[]()

```
template<class T>
T& VD< T >::operator[] (
             int i )  [inline]
```

Consulta y modifica el elemento i-esimo.

Parámetros

i	posicioin del vector a modifica y/o consultar

Devuelve

una referencia al i-esimo elemento del vector dinámico

Definición en la línea 58 del archivo VD.h.

```
58 {return datos[i];}  //version no constante
```

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- include/VD.h
- include/VD.cpp

Índice alfabético