

Proyecto Final – Entrega Final

Estructura de Datos

IL354 - Sección D01

Alumna: Cervantes Araujo Maria Dolores

Código: 217782452

Fecha de Elaboración: 07 mayo de 2023



Autoevaluación			
Concepto	Si	No	Acumulación
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	-100 pts	0 pts	0
Incluí el código fuente en formato de texto (sólo si funciona cumpliendo todos los requerimientos)	+25pts	0 pts	25
Incluí las impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)	+25pts	0 pts	25
Incluí una portada que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25
Incluí una descripción y conclusiones de mi trabajo	+25 pts	0 pts	25
Suma:			100

Introducción:

En la entrega del proyecto final solo fue necesario hacer cambios en la lista tanto de recetas como de ingredientes, utilizando la lista simplemente ligada para ingredientes y en el caso de recetas fue utilizada la lista doblemente enlazada con encabezado Dummy, a su vez se utilizó en cada lista el concepto de clases anidadas, implementado la clase Nodo, para cambiar el método de apuntador de la entrega preliminar por nodos en esta versión final.

Por último, se realizaron algunos cambios en la manera de hacer referencia entre otros elementos entre ambas listas mediante el menú y definimos varios métodos para que pudieran retornar por referencia, esto nos permitió recuperar los elementos de las listas, movernos por la lista de ingredientes desde la lista de recetas con la posición y realizar algunas manipulaciones de manera más directa; presentando los resultados a continuación.



Código Fuente:

main.cpp

```
#include "menu.hpp"
int main() {
    Menu start;
}
```

menu.hpp

```
#ifndef MENU_HPP_INCLUDED
#define MENU_HPP_INCLUDED
#include <string>
#include <iostream>
#include <cstdlib>
#include <windows.h>
#include "recipes.hpp"
#include "ingredientes.hpp"
#include "recipesList.hpp"
#include "ingredientList.hpp"

class Menu {
private:
    void marco(int, int, int, int);
    void gotoxy(int, int);
    void coordinates();

    void view(ListRecipes<Recetas> &);
    void recipeMenu(ListRecipes<Recetas> &);
    ListRecipes<Recetas> addRecipe();
    ListIngredient<Ingredients> addIngredients(ListIngredient<Ingredients>);
    void searchRecipe(ListRecipes<Recetas> &);
    void modifyRecipe(ListRecipes<Recetas> &);
    void deleteeR(ListRecipes<Recetas> &);
    void sortRecipe(ListRecipes<Recetas> &);

    void ingrediMenu(ListRecipes<Recetas> &);
    ListRecipes<Recetas> ingredientIntoRecipe(ListRecipes<Recetas>&);
    void modifyIngred(ListRecipes<Recetas> &);
    void deleteeI(ListRecipes<Recetas>&);
    void deleteAllIngred(ListRecipes<Recetas>&);

    void systemPause();
    void writeToDisk(ListRecipes<Recetas> &);
    void readFromDisk(ListRecipes<Recetas> &);

public:
    Menu();
};

#endif // MENU_HPP_INCLUDED
```



menu.cpp

```
#include "menu.hpp"

using namespace std;

Menu::Menu() {
    HANDLE wHnd;
    system("Color F5");
    marco(1,0,64,37);
    wHnd = GetStdHandle(STD_OUTPUT_HANDLE);
    SMALL_RECT windowSize = {20, 3, 85, 40};
    SetConsoleWindowInfo(wHnd, 3, &windowSize);

    ListRecipes<Recetas> listRecipe;
    view(listRecipe);
}

///      MENU PRINCIPAL

void Menu::view(ListRecipes<Recetas> &listRecipe) {
    int opc, e(1);
    Recetas recipe;

    while(e == 1) {
        system("cls");
        marco(1,0,64,37);
        gotoxy(7, 5);
        cout<< "\t\t\t\t\t RECETARIO DIGITAL" <<endl<<endl;
        cout<< "\t\t\t\t\t [1] Menu de Recetas" <<endl;
        cout<< "\t\t\t\t\t [2] Menu de Ingredientes " <<endl<<endl;

        cout << "\t\t\t\t\t MOSTAR RECETAS POR:"<<endl;
        cout<< "\t\t\t\t\t [3] Categoria Desayuno"<<endl;
        cout<< "\t\t\t\t\t [4] Categoria Comida "<<endl;
        cout<< "\t\t\t\t\t [5] Categoria Cena "<<endl;
        cout<< "\t\t\t\t\t [6] Categoria Navidenia"<<endl;
        cout<< "\t\t\t\t\t [7] Todas las Recetas "<<endl<<endl;
        cout<< "\t\t\t\t\t [8] Guardar a Disco"<<endl;
        cout<< "\t\t\t\t\t [9] Leer a Disco"<<endl;
        cout<< "\t\t\t\t\t [10] Salir "<<endl<<endl;
        cout<< "\t\t\t\t\t OPCION: ";
        cin>>opc;
        cin.ignore();

        switch(opc) {
            case 1:
                recipeMenu(listRecipe);
                break;

            case 2:
                ingrediMenu(listRecipe);
                break;
        }
    }
}
```



```
case 3:
    system("cls");
    gotoxy(8,1);
    cout<<" RECETAS DE LA CATEGORIA DESAYUNO"<<endl<<endl;
    gotoxy(2,2);
    cout <<"-----"
---"<<endl<<endl;
    if(listRecipe.isEmpty()) {
        cout<<"\tEl recetario se encuentra vacio"<<endl<<endl;
        systemPause();
    }
    else {
        recipe.setCategory("Desayuno");
        gotoxy(8,4);
        cout << listRecipe.category(recipe,
listRecipe.compareByCategory) << endl <<endl;
        systemPause();
    }
    break;

case 4:
    system("cls");
    gotoxy(8,1);
    cout<<" RECETAS DE LA CATEGORIA COMIDA"<<endl<<endl;
    gotoxy(2,2);
    cout <<"-----"
---"<<endl<<endl;
    if(listRecipe.isEmpty()) {
        cout<<"\tEl recetario se encuentra vacio"<<endl<<endl;
        systemPause();
    }
    else {
        recipe.setCategory("Comida");
        gotoxy(8,4);
        cout << listRecipe.category(recipe,
listRecipe.compareByCategory) << endl <<endl;
        systemPause();
    }
    break;

case 5:
    system("cls");
    gotoxy(8,1);
    cout<<" RECETAS DE LA CATEGORIA COMIDA"<<endl<<endl;
    gotoxy(2,2);
    cout <<"-----"
---"<<endl<<endl;
    if(listRecipe.isEmpty()) {
        cout<<"\tEl recetario se encuentra vacio"<<endl<<endl;
        systemPause();
    }
    else {
```



```
        recipe.setCategory("Cena");
        gotoxy(8,4);
        cout << listRecipe.category(recipe,
listRecipe.compareByCategory) << endl <<endl;
        systemPause();
    }
    break;

    case 6:
        system("cls");
        gotoxy(8,1);
        cout<<" RECETAS DE LA CATEGORIA NAVIDENIA"<<endl<<endl;
        gotoxy(2,2);
        cout <<"-----"
---"<<endl<<endl;
        if(listRecipe.isEmpty()) {
            cout<< "\tEl recetario se encuentra vacio"<<endl<<endl;
            systemPause();
        }
        else {
            recipe.setCategory("Navidenio");
            gotoxy(8,4);
            cout << listRecipe.category(recipe,
listRecipe.compareByCategory) << endl <<endl;
            systemPause();
        }
        break;

    case 7:
        system("cls");
        gotoxy(8,1);
        cout<<"\t RECETARIO COMPLETO"<<endl<<endl;
        gotoxy(2,2);
        cout <<"-----"
---"<<endl<<endl;
        if(listRecipe.isEmpty()) {
            cout<< "\tEl recetario se encuentra vacio"<<endl<<endl;
            systemPause();
        }
        else {
            gotoxy(8,4);
            cout<<listRecipe.toString();
            systemPause();
        }
        break;

    case 8:
        writeToDisk(listRecipe);
        break;

    case 9:
        readFromDisk(listRecipe);
        break;
```



```
case 10:
    cout<<"\n\tHASTA LA PROXIMA!! :D"<<endl;
    e+=1;
    exit(EXIT_PROCESS_DEBUG_EVENT);
    break;

default:
    std::cout<<"Ingresa solo valores que se te solicitan"<<std::endl;
    break;
}
}
}

///          MENU PARA RECETAS

void Menu::recipeMenu(ListRecipes<Recetas> &listRecipe) {
    int opc, j = 0;
    while(j==0) {
        coordinates();
        gotoxy(7, 5);
        cout << "\t MENU PARA RECETAS" <<endl<<endl;
        cout << "\t [1] Ingresar Receta " <<endl;
        cout << "\t [2] Buscar Receta"<<endl;
        cout << "\t [3] Modificar Procedimiento de una Receta"<<endl;
        cout << "\t [4] Ordenar las Recetas " <<endl;
        cout << "\t [5] Eliminar una Receta " <<endl;
        cout << "\t [6] Eliminar Todas las Recetas " <<endl;
        cout << "\t [7] Salir"<<endl<<endl;
        cout << "\t OPCION: ";
        cin>>opc;
        cin.ignore();

        switch(opc) {
            case 1:
                addRecipe();
                break;

            case 2:
                searchRecipe(listRecipe);
                break;

            case 3:
                modifyRecipe(listRecipe);
                break;

            case 4:
                sortRecipe(listRecipe);
                break;

            case 5:
                deleteer(listRecipe);
                break;
        }
    }
}
```




```
case 6:
    cout << "\n\t Eliminando..."<<endl;
    listRecipe.deleteAll();
    cout << "\n\t----- SE ELIMINARON TODAS LAS RECETAS -----
" << endl << endl;
    cin.ignore();
    break;

case 7:
    view(listRecipe);
    j+=2;
    break;
default:
    std::cout << "\t Ingresa solo valores que se te
solicitan" << std::endl;
    cin.get();
    break;
}
}
}

///      AÑADIR RECETA

ListRecipes<Recetas> Menu::addRecipe() {
    char x;
    int selec;
    Recetas recipe;
    NameA author;
    ListRecipes<Recetas> listRecipe;
    string myStr;

    do {
        coordinates();
        gotoxy(8,1);
        cout << "\t\tINGRESAR RECETA"<<endl;
        gotoxy(2,2);
        cout << "-----
" << endl << endl;
        gotoxy(8,4);
        cout << "CATEGORIAS: " << endl;
        cout << "\t [1] Desayuno" << endl;
        cout << "\t [2] Comida" << endl;
        cout << "\t [3] Cena" << endl;
        cout << "\t [4] Navidenio" << endl;
        cout << "\t Seleccion: ";
        cin >> selec;
        cin.ignore();

        switch(selec) {
            case 1:
                myStr = "Desayuno";
                break;

```




```
        case 2:
            myStr = "Comida";
            break;
        case 3:
            myStr = "Cena";
            break;
        case 4:
            myStr = "Navidenio";
            break;
        default:
            break;
    }
    recipe.setCategory(myStr);

    gotoxy(8,10);
    cout<<"Receta: ";
    getline(cin, myStr);
    recipe.setTitle(myStr);

    gotoxy(8,11);
    cout<<"Nombre Autor: ";
    getline(cin, myStr);
    author.setName(myStr);
    recipe.setNom(author);

    gotoxy(8,12);
    cout<<"Apellido Autor: ";
    getline(cin, myStr);
    author.setLastName(myStr);
    recipe.setNom(author);

    recipe.setIngredient(addIngredients(recipe.getIngredient()));

    cout<<"\n\tTiempo de Preparacion en min: ";
    getline(cin, myStr);
    recipe.setTime(atof(myStr.c_str()));

    cout<<"\tProceso: ";
    getline(cin, myStr);
    recipe.setProcess(myStr);

    listRecipe.insertData(listRecipe.getLastPosition(), recipe);
    cout<<endl<<endl<<"\n\t----- RECETARIO ACTUALIZADO -----"
"<<endl<<endl;
    cout << "\tDesea insertar otra receta? [y/n]: ";
    cin >> x;
    cin.ignore();
}
while(x=='y');
recipeMenu(listRecipe);
}

///      COMPLEMENTO DE INGREDIENTES A RECETA
```



```
ListIngredient<Ingredients> Menu::addIngredients(ListIngredient<Ingredients>
listIngrede) {
    char m;
    Ingredients food;
    string myStr;

    listIngrede.deleteAll();
    do {
        cout<<"\tIngrediente: ";
        getline(cin, myStr);
        food.setNameIngr(myStr);

        cout<<"\tCantidad: ";
        getline(cin, myStr);
        food.setQuantity(myStr);

        listIngrede.insertSortData(food);
        cout << "\tDesea agregar otro ingrediente? [y/n]: ";
        cin >> m;
        cin.ignore();
    }
    while(m=='y');
    return listIngrede;
}

///          BUSCAR RECETA

void Menu::searchRecipe(ListRecipes<Recetas> &listRecipe) {
    Recetas recipe;
    ListRecipes<Recetas>::Position pos;
    string myStr;

    system("cls");
    gotoxy(8,1);
    cout<<"\t\tBUSQUEDA DE RECETA"<<endl;
    gotoxy(2,2);
    cout <<"-----"
" <<endl<<endl;
    gotoxy(4,4);
    cout<<"Receta:"<<endl;
    cout<<"\t - ";
    getline(cin, myStr);
    recipe.setCategory(myStr);
    recipe.setTitle(myStr);

    pos = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);

    if(pos == nullptr) {
        gotoxy(4,6);
        cout<<"No pudimos encontrar la receta en la lista :("<<endl;
        Sleep(2000);
    }
}
```



```
else {
    Sleep(2000);
    gotoxy(4, 7);
    cout<<"\n\t----- RECETA ENCONTRADA -----"<<endl<<endl;
    cout<< listRecipe.retrieve(pos).toString();
    cout<<endl;
    systemPause();
}
recipeMenu(listRecipe);
}

///      MODIFICAR RECETA

void Menu::modifyRecipe(ListRecipes<Recetas> &listRecipe) {
    Recetas recipe, aux;
    ListRecipes<Recetas>::Position pos;
    string myStr;

    coordinates();
    gotoxy(9,1);
    cout<<"MODIFICAR PROCEDIMIENTO DE LA RECETA"<<endl;
    gotoxy(2,2);
    cout <<"-----"
" <<endl<<endl;
    gotoxy(4,4);
    cout<<"Receta: "<<endl;
    cout <<"\t -";
    getline(cin, myStr);
    recipe.setTitle(myStr);
    pos = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);

    if(pos == nullptr) {
        cout<<"\t NO EXISTE LA RECETA"<<endl;
        cin.get();
    }
    else {
        string process;
        aux = Recetas(listRecipe.retrieve(pos));
        cout<<"\n\t Receta:"<<listRecipe.retrieve(pos).getTitle();
        cout<<"\n\t Proceso Nuevo: "<<endl;
        cout<<"\t ";
        getline(cin, process);

        aux.setProcess(process);
        listRecipe.insertData(pos, aux);
        listRecipe.deleteData(pos);
        cout<<endl<<endl<<endl<<"\n\t----- MODIFICADO CON EXITO -----"
" <<endl<<endl;
        systemPause();
    }
}

///      ELIMINAR RECETA
```



```
void Menu::deleteeR(ListRecipes<Recetas> &listRecipe) {
    ListRecipes<Recetas>::Position pos;
    Recetas recipe;
    string myStr;

    coordinates();
    gotoxy(10,1);
    cout<<"\t ELIMINAR UNA RECETA"<<endl;
    gotoxy(2,2);
    cout <<"-----"
" <<endl<<endl;
    gotoxy(4,4);
    cout<<"\t Receta a eliminar: "<<endl;
    cout << "\t - ";
    getline(cin, myStr);

    recipe.setTitle(myStr);
    pos = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);
    cout<<endl<<"\t Eliminando..."<<endl<<endl;
    Sleep(2000);

    if(pos == nullptr) {
        cout<<"\t No existe la receta"<<endl;
    }
    else {
        listRecipe.deleteData(pos);
        cout<<endl<<endl<<endl<<endl<<"\t----- RECETA ELIMINADA -----"
- "<<endl<<endl;
    }
    systemPause();
}

/// ORDENAR RECETA

void Menu::sortRecipe(ListRecipes<Recetas> &listRecipe) {
    int opc;
    do {
        coordinates();
        gotoxy(8,1);
        cout <<"\t\t ORDENAR RECETAS"<<endl;
        gotoxy(2,2);
        cout <<"-----"
" <<endl<<endl;
        gotoxy(4,4);
        cout << "[1] Ordenar por nombre"<<endl;
        gotoxy(4,5);
        cout << "[2] Ordenar por tiempo de preparacion" << endl;
        gotoxy(4,6);
        cout << "[3] Salir"<<endl<<endl;
        gotoxy(4,7);
        cout << "OPCION: ";
        cin >> opc;
    }
```



```
system("cls");
if(opc == 1) {
    listRecipe.quickSort(listRecipe.compareByTitle);
    gotoxy(9,1);
    cout << " ORDENADO POR NOMBRE" << endl << endl;
    gotoxy(2,2);
    cout << "-----" << endl << endl;
    gotoxy(4,4);
    cout << listRecipe.toString();
    cin.get();
}
else if(opc == 2) {
    listRecipe.quickSort(listRecipe.compareByTime);
    gotoxy(7,1);
    cout << "ORDENADO POR TIEMPO DE PREPARACION" << endl << endl;
    gotoxy(2,2);
    cout << "-----" << endl << endl;
    gotoxy(4,4);
    cout << listRecipe.toString();
    cin.get();
}
else {
    cout << "\t Ingresa solo lo que se te pide" << endl << endl;
}
systemPause();
}
while(opc >= 3 || opc <= 0 && opc == 3);
recipeMenu(listRecipe);
}

///          MENU PARA INGREDIENTES

void Menu::ingrediMenu(ListRecipes<Recetas> &listRecipe) {
    int opc, y=0;
    string myStr;
    Ingredients food;
    Recetas recipe, aux;
    ListIngredient<Ingredients> lisIngr;
    ListIngredient<Ingredients>::Position posI;
    ListRecipes<Recetas>::Position posR;

    while(y==0) {
        coordinates();
        gotoxy(7, 5);
        cout << "\t MENU PARA INGREDIENTES" << endl << endl;
        cout << "\t [1] Ingresar ingredientes a una receta" << endl;
        cout << "\t [2] Modificar la cantidad de un ingrediente" << endl;
        cout << "\t [3] Eliminar un ingrediente " << endl;
        cout << "\t [4] Eliminar todos los ingredientes " << endl;
```



```
cout<< "\t [5] Salir del Menu"<<endl<<endl;
cout<< "\t OPCION: ";
cin>>opc;
cin.ignore();
switch(opc) {
    case 1:
        ingredientIntoRecipe(listRecipe);
        break;

    case 2:
        modifyIngred(listRecipe);
        break;

    case 3:
        deleteeI(listRecipe);
        break;

    case 4:
        deleteAllIngred(listRecipe);
        break;

    case 5:
        view(listRecipe);
        y+=1;
        break;

    default:
        std::cout<<"\t Ingresa solo valores que se te
solicitan"<<std::endl;
        break;
}
}

///      AÑADIR MAS INGREDIENTES A RECETAS

ListRecipes<Recetas> Menu::ingredientIntoRecipe(ListRecipes<Recetas> & listRecipe)
{
    Ingredients food;
    Recetas recipe, aux;
    ListIngredient<Ingredients> listIngred;
    ListRecipes<Recetas>::Position posR;
    string myStr;

    coordinates();
    gotoxy(8,1);
    cout<<"\t  INGRESAR OTRO INGREDIENTE"<<endl;
    gotoxy(2,2);
    cout <<"-----"
"<<endl<<endl;
    gotoxy(4,4);
    cout<<"\tReceta: ";
    getline(cin, myStr);
```



```
recipe.setTitle(myStr);
posR = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);

if(posR == nullptr) {
    cout<<"\t NO EXISTE LA RECETA"<<endl;
    Sleep(1000);
    ingrediMenu(listRecipe);
}

aux = Recetas(listRecipe.retrieve(posR));
listIngred = aux.getIngredient();
gotoxy(2,6);
cout <<"-----"
"<<endl<<endl;
cout<<"\t\t"<<aux.getTitle()<<endl<<endl;
cout<<"\tIngredientes Actuales:"<<endl<<endl;
cout<<aux.getIngredient().toString();
cout<<endl<<endl<<"\tNuevo Ingrediente: ";
getline(cin, myStr);
food.setNameIngr(myStr);

cout<<"\tCantidad: ";
getline(cin, myStr);
food.setQuantity(myStr);

listIngred.insertSortData(food);
aux.setIngredient(listIngred);
listRecipe.insertData(posR, aux);
listRecipe.deleteData(posR);

cout<<endl<<endl<<endl<<"\t----- INGREDIENTES ACTUALIZADOS -----"
"<<endl<<endl;
systemPause();
return listRecipe;
}

///          MODIFICAR INGREDIENTES

void Menu::modifyIngréd(ListRecipes<Recetas> &listRecipe) {
    Recetas recipe, aux;
    Ingredients food;
    ListIngredient<Ingredients>::Position posI;
    ListRecipes<Recetas>::Position posR;
    ListIngredient<Ingredients> listIngred;
    string myStr;

    coordinates();
    gotoxy(7,1);
    cout<<"\t\tMODIFICAR CANTIDAD DE UN INGREDIENTE"<<endl;
    gotoxy(2,2);
    cout <<"-----"
"<<endl<<endl;
    gotoxy(4,4);
```




```
cout<<"Receta: ";
getline(cin, myStr);
recipe.setTitle(myStr);
posR = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);

if(posR==nullptr) {
    cout<<"\t NO EXISTE LA RECETA"<<endl;
}
else {
    aux = Recetas(listRecipe.retrieve(posR));
    listIngred = aux.getIngredient();
    gotoxy(2,6);
    cout<<"-----"
"<<endl<<endl;
    cout<<"\t\t"<<aux.getTitle()<<endl;
    cout<<aux.getIngredient().toString()<<endl;
    cout<<endl<<"\n\tIngrediente: ";
    getline(cin, myStr);
    food.setNameIngr(myStr);
    posI = listIngred.findDatLin(food, listIngred.compareByNameIngr);

    if(posI == nullptr) {
        cout<<"\t NO EXISTE EL INGREDIENTE"<<endl;
    }

    else {

        string quantity;
        cout<<"\tCambio de Cantidad: ";
        getline(cin, quantity);

        listIngred.retrieve(posI).setQuantity(quantity);
        aux.setIngredient(listIngred);
        listRecipe.insertData(posR, aux);
        listRecipe.deleteData(posR);

        cout<<endl<<endl<<endl<<"\t----- MODIFICADO CON EXITO -----"
--"<<endl<<endl;
    }
}
systemPause();
}

///      ELIMINAR INGREDIENTE

void Menu::deleteeI(ListRecipes<Recetas> &listRecipe) {
    Recetas recipe, aux;
    Ingredients food;
    ListIngredient<Ingredients>::Position posI;
    ListRecipes<Recetas>::Position posR;
    ListIngredient<Ingredients> listIngred;
    string myStr;
```



```
coordinates();
gotoxy(7,1);
cout <<"\t ELIMINAR UN INGREDIENTE"<<endl;
gotoxy(2,2);
cout <<"-----"
"<<endl<<endl;
gotoxy(4,4);
cout<<"\tReceta: ";
getline(cin, myStr);
recipe.setTitle(myStr);
posR = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);

if(posR == nullptr) {
    cout<<"\t NO EXISTE LA RECETA"<<endl;
}
else {
    aux = Recetas(listRecipe.retrieve(posR));
    listIngred = aux.getInredient();

    gotoxy(2,6);
    cout <<"-----"
"<<endl<<endl;
    cout<<"\t\t"<<aux.getTitle()<<endl;
    cout<<aux.getInredient().toString()<<endl;
    cout<<endl<<"\t Ingrediente: ";
    getline(cin, myStr);
    food.setNameIngr(myStr);

    posI = listIngred.findDatLin(food, listIngred.compareByNameIngr);

    if(posI == nullptr) {
        cout<<endl<<"\t NO EXISTE EL INGREDIENTE"<<endl;
    }

    else {
        cout<<endl<<endl<<"\t Eliminando..."<< endl;
        listIngred.deleteData(posI);
        aux.setIngredient(listIngred);

        listRecipe.insertData(posR, aux);
        listRecipe.deleteData(posR);
        cout<<endl<<endl<<endl<<"\t----- ELIMINADO CON EXITO -----"
<<endl<<endl;
    }
}
systemPause();
}

/// ELIMINAR TODOS LOS INGREDIENTES

void Menu::deleteAllIngred(ListRecipes<Recetas> &listRecipe) {
    Recetas recipe, aux;
    Ingredients food;
```



```
ListIngredient<Ingredients>::Position posI;
ListRecipes<Recetas>::Position posR;
ListIngredient<Ingredients> listIngred;
string myStr;

coordinates();
gotoxy(7,1);
cout <<"\t ELIMINAR LISTA DE INGREDIENTES"<<endl;
gotoxy(2,2);
cout <<"-----"
"<<endl<<endl;
gotoxy(4,5);
cout<<"\t Receta: ";
getline(cin, myStr);
recipe.setTitle(myStr);
posR = listRecipe.findDatLin(recipe, listRecipe.compareByTitle);

if(posR==nullptr) {
    cout<<endl<<"\t NO EXISTE LA RECETA"<<endl;
}
else {
    aux = Recetas(listRecipe.retrieve(posR));
    listIngred = aux.getIngredient();

    cout<<endl<<endl<<"\t Eliminando..."<<endl;
    listIngred.deleteAll();
    aux.setIngredient(listIngred);
    listRecipe.insertData(posR, aux);
    listRecipe.deleteData(posR);
    cout<<endl<<endl<<endl<<endl<<endl<<"\t----- SE ELIMINARON TODOS LOS
INGREDIENTES -----"<<endl<<endl;
}
systemPause();
}

///LECTURA Y ESCRITURA DE DISCO
void Menu::writeToDisk(ListRecipes<Recetas>& listRecipe) {
    Recetas recipe;
    ListIngredient<Ingredients> lisIngr;
    string title;
    ListRecipes<Recetas>::Position posR, aux;

    cout<<endl<<endl<<endl<<"\t Escribiendo al disco..."<<endl;
    posR = listRecipe.getFirstPosition();
    aux = listRecipe.getLastPosition();

    while(aux != posR) {
        recipe = listRecipe.retrieve(posR);
        title = recipe.getTitle();

        lisIngr = listRecipe.retrieve(posR).getIngredient();
        lisIngr.writeToDisk(title+"_ingredients.txt");
        lisIngr.deleteAll();
    }
}
```



```
listRecipe.retrieve(posR).setIngredient(lisIngr);

posR = posR->getNext();

if(aux == posR) {
    recipe = listRecipe.retrieve(posR);
    title = recipe.getTitle();

    lisIngr = listRecipe.retrieve(posR).getIngredient();
    lisIngr.writeToDisk(title+"_ingredients.txt");
    lisIngr.deleteAll();
    listRecipe.retrieve(posR).setIngredient(lisIngr);
}

listRecipe.writeToDisk("RecipeBook.txt");
cout<<"\n\t----- Accion Completada -----"<<endl<<endl;
systemPause();
}

void Menu::readFromDisk(ListRecipes<Recetas>& listRecipe) {
    Recetas recipe;
    ListIngredient<Ingredients> lisIngr;
    ListRecipes<Recetas>::Position posR, aux;
    string title;

    cout<<endl<<endl<<endl<<"\t Leyendo del disco..."<<endl;
    listRecipe.readFromDisk("RecipeBook.txt");

    posR = listRecipe.getFirstPosition();
    aux = listRecipe.getLastPosition();

    while(aux != posR) {
        recipe = listRecipe.retrieve(posR);
        title = recipe.getTitle();
        lisIngr.readFromDisk(title+"_ingredients.txt");
        listRecipe.retrieve(posR).setIngredient(lisIngr);
        posR = posR->getNext();

        if(aux == posR) {
            recipe = listRecipe.retrieve(posR);
            title = recipe.getTitle();

            lisIngr.deleteAll();
            lisIngr.readFromDisk(title+"_ingredients.txt");
            listRecipe.retrieve(posR).setIngredient(lisIngr);
        }
    }

    cout<<"\n\t----- Accion Completada -----"<<endl<<endl;
    systemPause();
}
```



```
void Menu::systemPause() {
    cout << "\t Presiona enter para continuar...";
    cin.ignore();
    system("cls");
}

void Menu::gotoxy(int x, int y) {
    HANDLE hCon;
    hCon=GetStdHandle(STD_OUTPUT_HANDLE);
    COORD dwPos;
    dwPos.X = x;
    dwPos.Y = y;
    SetConsoleCursorPosition(hCon, dwPos);
}

void Menu::marco(int xs, int ys, int xi, int yi) {
    int i;

    i=xs;
    while(i<=xi) {
        gotoxy(i,ys);
        cout<<"=";
        gotoxy(i,yi);
        cout<<"=";
        i++;
    }

    i=ys;
    while(i<=yi) {
        gotoxy(xs,i);
        cout<<"|";
        gotoxy(xi,i);
        cout<<"|";
        i++;
    }

    gotoxy(xs, ys);
    cout<<"=";
    gotoxy(xi, yi);
    cout<<"=";
    gotoxy(xi, ys);
    cout<<"=";
    gotoxy(xs, yi);
    cout<<"=";
}

void Menu::coordinates() {
    system("cls");
    marco(1,0,64,37);
}
```



ingredientList.hpp

```
#ifndef INGREDIENTLIST_HPP_INCLUDED
#define INGREDIENTLIST_HPP_INCLUDED
#include <string>
#include <iostream>
#include <fstream>

template <class T>
class ListIngredient {
private:
    ///Clase anidada de NODO
    class Node {
private:
        T data;
        Node* next;
public:
        Node();
        Node(const T&);

        T& getData();
        Node* getNext() const;

        void setData(const T&);
        void setNext(Node*);
    };

    Node *anchor;
    bool validPosition(Node*) const;
    void copyAll(const ListIngredient<T>&);

public:
    typedef Node* Position;

    ///CLASE DE EXCEPTION
    class Exception: public std::exception {
private:
        std::string msg;
public:
        explicit Exception(const char* message): msg(message) {}
        explicit Exception(const std::string& message): msg(message) {}
        virtual ~Exception() throw() {}
        virtual const char* what() const throw() {
            return msg.c_str();
        }
    };

    ListIngredient();
    ListIngredient(const ListIngredient<T>&);
    ~ListIngredient();

    bool isEmpty();
    bool isSorted() const;
```



```
void insertData (Node*, const T&);
void insertSortData (T&);
void deleteData (Node*);
std::string category (const T&, int (const T&, const T&));
T& retrieve (Node*&);

Node* getFirstPosition();
Node* getLastPosition();
Node* getBeforePosition (Node*) const;
Node* getNextPosition (Node*) const;

static int compareByNameIngr (const T&, const T&);

Node* findDatLin (const T&, int (const T&, const T&)) const;

std::string toString() const;
void deleteAll();
void writeToDisk (const std::string&);
void readFromDisk (const std::string&);

ListIngredient<T>& operator = (const ListIngredient<T>&);
};

///+++++

using namespace std;
///Implementaci3n NODO

template <class T>
ListIngredient<T>::Node::Node() : next(nullptr) { }

template <class T>
ListIngredient<T>::Node::Node(const T& m) : data(m), next(nullptr) { }

template <class T>
T& ListIngredient<T>::Node::getData() {
    return data;
}

template <class T>
typename ListIngredient<T>::Node* ListIngredient<T>::Node::getNext() const {
    return next;
}

template <class T>
void ListIngredient<T>::Node::setData(const T& m) {
    data = m;
}

template <class T>
void ListIngredient<T>::Node::setNext(Node* pos) {
    next = pos;
}
```




```
}
```

///IMPLEMENTACIÓN LISTA

```
template <class T>
```

```
ListIngredient<T>::ListIngredient() : anchor(nullptr) {}
```

```
template<class T>
```

```
ListIngredient<T>::ListIngredient( const ListIngredient& l) : anchor(nullptr) {  
    copyAll(l);  
}
```

```
template<class T>
```

```
void ListIngredient<T>::copyAll(const ListIngredient<T>& obj) {  
    Node* aux(obj.anchor);  
    Node* last(nullptr);  
    Node* newNode;
```

```
    while(aux != nullptr) {  
        newNode = new Node(aux->getData());  
  
        if(last == nullptr) {  
            anchor = newNode;  
        }  
        else {  
            last->setNext(newNode);  
        }  
        last = newNode;  
        aux = aux->getNext();  
    }  
}
```

```
template <class T>
```

```
ListIngredient<T>::~ListIngredient() {  
    deleteAll();  
}
```

```
template<class T>
```

```
bool ListIngredient<T>::isEmpty() {  
    return anchor == nullptr;  
}
```

```
template<class T>
```

```
bool ListIngredient<T>::validPosition(Node* pos) const {  
    Node* aux(anchor);  
  
    while(aux != nullptr) {  
        if(aux == pos) {  
            return true;  
        }  
        aux = aux->getNext();  
    }  
    return false;  
}
```



```
    }

    ///METODO INSERTAR
    template<class T>
    void ListIngredient<T>::insertData(Node* pos, const T& obj) {
        if(pos != nullptr && !validPosition(pos)) {
            throw Exception("Posicion invalida, insertData");
        }

        Node* aux(new Node(obj));

        if(aux == nullptr) {
            throw Exception("Memoria no disponible, insertData");
        }

        ///Insertar al principio
        if(pos == nullptr) {
            aux->setNext(anchor);
            anchor = aux;
        }

        ///Insertar en cualquier otra posicion
        else {
            aux->setNext(pos->getNext());
            pos->setNext(aux);
        }
    }

    template <typename T>
    void ListIngredient<T>::insertSortData(T& obj) {
        Node* aux(anchor);
        bool flag = false;

        while(aux != nullptr && flag != true) {
            if(obj < aux->getData()) {
                insertData(getBeforePosition(aux), obj);
                flag = true;
            }
            aux = getNextPosition(aux);
        }

        if(flag == false) {
            insertData(getLastPosition(), obj);
        }
    }

    ///METODO ELIMINAR
    template<class T>
    void ListIngredient<T>::deleteData(Node* pos) {
        if(!validPosition(pos)) {
            throw Exception("Posicion invalida, deleteData");
        }

        ///Eliminar el primero
        if(pos == anchor) {
```



```
        anchor = anchor->getNext();
    }
    else {
        getBeforePosition(pos)->setNext(pos->getNext());
    }
    delete pos;
}

template <class T>
string ListIngredient<T>::category(const T& obj, int cmp(const T&, const T&)) {
    std::string resCateg;
    Node* aux;
    while(aux != nullptr) {
        if(cmp(obj, aux->getData()) == 0) {
            resCateg += aux->toString();
            resCateg += '\n';
        }
        aux->getNext();
    }
    return resCateg;
}

///RECUPERAR
template<class T>
T& ListIngredient<T>::retrieve(Node*& pos) {
    if(!validPosition(pos)) {
        throw Exception("Receta invalida, retrieve");
    }
    return pos->getData();
}

///PRIMERA POSICIÓN
template <class T>
typename ListIngredient<T>::Node* ListIngredient<T>::getFirstPosition() {
    return anchor;
}

///ÚLTIMA POSICIÓN
template <class T>
typename ListIngredient<T>::Node* ListIngredient<T>::getLastPosition() {
    if(isEmpty()) {
        return nullptr;
    }

    Node* aux(anchor);

    while(aux->getNext() != nullptr) {
        aux = aux->getNext();
    }
    return aux;
}

///ANTES DE CIERTA POSICIÓN
```



```
template <class T>
typename ListIngredient<T>::Node* ListIngredient<T>::getBeforePosition(Node* pos)
const {
    if(pos == anchor) {
        return nullptr;
    }

    Node* aux(anchor);

    while(aux != nullptr && aux->getNext() != pos) {
        aux = aux->getNext();
    }
    return aux;
}

//DESPUÉS DE CIERTA POSICIÓN
template <class T>
typename ListIngredient<T>::Node* ListIngredient<T>::getNextPosition(Node* pos)
const {
    if(!validPosition(pos)) {
        return nullptr;
    }
    return pos->getNext();
}

template <class T>
int ListIngredient<T>::compareByNameIngr(const T& orig, const T& copyc) {
    return orig.getNameIngr().compare(copyc.getNameIngr());
}

template <class T>
typename ListIngredient<T>::Node* ListIngredient<T>::findDatLin(const T& obj, int
com(const T&, const T&)) const {
    Node* aux(anchor);
    while(aux != nullptr && com(aux->getData(), obj)!=0) {
        aux = aux->getNext();
    }
    return aux;
}

template <class T>
void ListIngredient<T>::writeToDisk(const string& fileName) {
    ofstream myFile;
    myFile.open(fileName, ios_base::trunc);

    if(!myFile.is_open()) {
        throw Exception("No se puede abrir el archivo para escritura,
writeToDisk");
    }

    Node* aux = anchor;
    while(aux != nullptr) {
        myFile << aux->getData();
    }
}
```



```
        aux = aux->getNext();
    }
    myFile.close();
}

///Lectura a disco
template <class T>
void ListIngredient<T>::readFromDisk(const string& fileName) {
    ifstream myFile;

    myFile.open(fileName);
    if(!myFile.is_open()) {
        throw Exception("No se pudo abrir el archivo para lectura, readToDisk");
    }

    deleteAll();
    T myData;
    ListIngredient myListIng;

    try {
        while(myFile >> myData) {
            insertData(getLastPosition(), myData);
            //myListIng.insertSortData(myData);
        }
    }
    catch(Exception ex) {
        myFile.close();
        throw Exception(ex.what());
    }

    myFile.close();
}

template<class T>
string ListIngredient<T>::toString() const {
    string listComplete;
    Node* aux(anchor);
    while(aux!=nullptr) {
        listComplete += aux->getData().toString() + "\n";
        aux = aux->getNext();
    }
    return listComplete;
}

template <class T>
void ListIngredient<T>::deleteAll() {
    Node* aux;

    while(anchor != nullptr) {
        aux = anchor;
        anchor = anchor->getNext();
        delete aux;
    }
}
```



```
}
```

```
template<class T>
ListIngredient<T>& ListIngredient<T>::operator = (const ListIngredient<T>& obj) {
    deleteAll();
    copyAll(obj);
    return *this;
}
```

```
#endif // INGREDIENTLIST_HPP_INCLUDED
```

recipeList.hpp

```
#ifndef RECIPESLIST_HPP_INCLUDED
#define RECIPESLIST_HPP_INCLUDED
#include <string>
#include <iostream>
#include <exception>
#include <fstream>
```

```
template <class T>
class ListRecipes {
    private:
        ///Clase anidada de NODO
        class Node {
            private:
                T* dataPtr;
                Node* prev;
                Node* next;

            public:
                class Exception: public std::exception {
                    private:
                        std::string msg;
                    public:
                        explicit Exception(const char* message): msg(message) {}
                        explicit Exception(const std::string& message):
msg(message) {}

                        virtual ~Exception() throw() {}
                        virtual const char* what() const throw() {
                            return msg.c_str();
                        }
                };

                Node();
                Node(const T&);
                ~Node();

                T* getDataPtr() const;
                T& getData();
                Node* getPrev() const;
                Node* getNext() const;
```



```
        void setDataPtr(const T*);  
        void setData(const T&);  
        void setPrev(Node*);  
        void setNext(Node*);  
};  
  
Node* header;  
bool validPosition(Node*) const;  
void copyAll(const ListRecipes<T>&);  
void swapData(T&, T&);  
Node* sortDataQuick(Node*, Node*, int cmp(const Recetas&, const Recetas&));  
  
public:  
    typedef Node* Position;  
  
    ///CLASE DE EXCEPTION  
    class Exception: public std::exception {  
    private:  
        std::string msg;  
    public:  
        explicit Exception(const char* message): msg(message) {}  
        explicit Exception(const std::string& message): msg(message) {}  
        virtual ~Exception() throw() {}  
        virtual const char* what() const throw() {  
            return msg.c_str();  
        }  
    };  
  
ListRecipes();  
ListRecipes(const ListRecipes<T>&);  
~ListRecipes();  
  
bool isEmpty();  
  
void insertData(Node*, const T&);  
void insertSortData(T&);  
void deleteData(Node*);  
std::string category(const T&, int (const T&, const T&));  
T& retrieve(Node*&);  
  
Node* getFirstPosition() ;  
Node* getLastPosition() ;  
Node* getBeforePosition(Node*) const;  
Node* getNextPosition(Node*) const;  
Node* localiza(const T&) const;  
  
static int compareByNameIngr(const T&, const T&);  
static int compareByCategory(const T&, const T&);  
static int compareByTitle(const T&, const T&);  
static int compareByTime(const T&, const T&);  
static int compareByName(const T&, const T&);
```




```
Node* findDatLin(const T&,int (const T&,const T&)) const;
void quickSort(int cmp(const Recetas&, const Recetas&));

std::string toString() const;
void deleteAll();
void writeToDisk(const std::string&);
void readFromDisk(const std::string&);

ListRecipes<T>& operator = (const ListRecipes<T>&);
};

#endif // RECIPESLIST_HPP_INCLUDED

///+++++

using namespace std;
///-----Implementación NODO

template <class T>
ListRecipes<T>::Node::Node() : dataPtr(nullptr), prev(nullptr), next(nullptr) { }

template <class T>
ListRecipes<T>::Node::Node(const T& m) : dataPtr(new T(m)), prev(nullptr),
next(nullptr) {
    if(dataPtr == nullptr) {
        throw Exception("Memoria insuficiente, se creara un nodo");
    }
}

template <class T>
ListRecipes<T>::Node::~Node() {
    delete dataPtr;
}

template <class T>
T* ListRecipes<T>::Node::getDataPtr() const {
    return dataPtr;
}

template <class T>
T& ListRecipes<T>::Node::getData() {
    if(dataPtr == nullptr) {
        throw Exception("Dato inexistente, getData");
    }
    return *dataPtr;
}

template <class T>
typename ListRecipes<T>::Node* ListRecipes<T>::Node::getPrev() const {
    return prev;
}

template <class T>
```



```
typename ListRecipes<T>::Node* ListRecipes<T>::Node::getNext() const {
    return next;
}

template <class T>
void ListRecipes<T>::Node::setDataPtr(const T* pos) {
    dataPtr = pos;
}

template <class T>
void ListRecipes<T>::Node::setData(const T& e) {
    if(dataPtr == nullptr) {
        if((dataPtr = new T(e)) == nullptr) {
            throw Exception("Memoria no disponible, setData");
        }
    }
    else {
        *dataPtr = e;
    }
}

template <class T>
void ListRecipes<T>::Node::setPrev(Node* pos) {
    prev = pos;
}

template <class T>
void ListRecipes<T>::Node::setNext(Node* pos) {
    next = pos;
}

///IMPLEMENTACION LISTA
template <class T>
ListRecipes<T>::ListRecipes() : header(new Node) {
    if(header == nullptr) {
        throw Exception("Memoria no disponible, inicializando lista...");
    }
    header->setPrev(header);
    header->setNext(header);
}

template <class T>
ListRecipes<T>::ListRecipes(const ListRecipes& obj) : ListRecipes() {
    copyAll(obj);
}

template <class T>
ListRecipes<T>::~ListRecipes() {
    deleteAll();
    delete header;
}
```



```
template <class T>
void ListRecipes<T>::copyAll(const ListRecipes& obj) {
    Node* aux(obj.header->getNext());
    Node* newNode;

    while(aux != obj.header) {
        try {
            if((newNode = new Node(aux->getData())) == nullptr) {
                throw Exception("Memoria no disponible, copyAll");
            }
        }
        catch(typename Node::Exception ex) {
            throw Exception(ex.what());
        }

        newNode->setPrev(header->getPrev());
        newNode->setNext(header);

        header->getPrev()->setNext(newNode);
        header->setPrev(newNode);

        aux = aux->getNext();
    }
}

template <class T>
bool ListRecipes<T>::isEmpty() {
    return header->getNext() == header;
}

template <class T>
bool ListRecipes<T>::validPosition(Node* pos) const {
    Node* aux(header->getNext());

    while(aux != header) {
        if(aux == pos) {
            return true;
        }
        aux = aux->getNext();
    }
    return false;
}

///METODO INSERTAR
template<class T>
void ListRecipes<T>::insertData(Node* pos, const T& obj) {
    if(pos != nullptr && !validPosition(pos)) {
        throw Exception("Posicion invalida, insertData");
    }

    Node* aux;
    try {
        aux = new Node(obj);
    }
```



```
    }  
    catch (typename Node::Exception ex) {  
        throw Exception(ex.what());  
    }  
  
    if(aux == nullptr) {  
        throw Exception("Memoria no disponible, insertData");  
    }  
    if(pos == nullptr) {  
        pos = header;  
    }  
  
    aux->setPrev(pos);  
    aux->setNext(pos->getNext());  
  
    pos->getNext()->setPrev(aux);  
    pos->setNext(aux);  
}  
  
template <class T>  
void ListRecipes<T>::insertSortData(T& obj) {  
    Node* aux(header->getNext());  
    if(isEmpty()) {  
        insertData(obj, getLastPosition());  
    }  
  
    while(aux != header) {  
        if(obj < aux->getData()) {  
            insertData(obj, getBeforePosition(aux));  
        }  
        aux = getNextPosition(aux);  
    }  
}  
  
///METODO ELIMINAR  
template<class T>  
void ListRecipes<T>::deleteData(Node* pos) {  
    if(!validPosition(pos)) {  
        throw Exception("Posicion invalida, deleteData");  
    }  
    pos->getPrev()->setNext(pos->getNext());  
    pos->getNext()->setPrev(pos->getPrev());  
    delete pos;  
}  
  
template <class T>  
string ListRecipes<T>::category(const T& obj, int cmp(const T&, const T&)) {  
    std::string resCateg;  
    Node* aux(header->getNext());  
    while(aux != header) {  
        if(cmp(obj, aux->getData()) == 0) {  
            resCateg += aux->getData().toString();  
            resCateg += '\n';  
        }  
        aux = getNextPosition(aux);  
    }  
}
```



```
    }  
    aux = aux->getNext();  
    }  
    return resCateg;  
    }  
  
///RECUPERAR  
template<class T>  
T& ListRecipes<T>::retrieve(Node*& pos) {  
    if(!validPosition(pos)) {  
        throw Exception("Dato invalida, retrieve");  
    }  
    return pos->getData();  
}  
  
template <class T>  
typename ListRecipes<T>::Node* ListRecipes<T>::localiza(const T& obj) const {  
    Node* aux(header->getNext());  
  
    while(aux != header) {  
        if(aux->getData() == obj) {  
            return aux;  
        }  
        aux = aux->getNext();  
    }  
    return nullptr;  
}  
  
///PRIMERA POSICIÓN  
template<class T>  
typename ListRecipes<T>::Node* ListRecipes<T>::getFirstPosition() {  
    if(isEmpty()) {  
        return nullptr;  
    }  
    return header->getNext();  
}  
  
///ÚLTIMA POSICIÓN  
template <class T>  
typename ListRecipes<T>::Node* ListRecipes<T>::getLastPosition() {  
    if(isEmpty()) {  
        return nullptr;  
    }  
    return header->getPrev();  
}  
  
///ANTES DE CIERTA POSICIÓN  
template<class T>  
typename ListRecipes<T>::Node* ListRecipes<T>::getBeforePosition(Node* pos) const {  
    if(!validPosition(pos) || pos == header->getNext()) {  
        return nullptr;  
    }  
    if(pos->getPrev() == header) {
```



```
        return header->getPrev();  
    }  
    return pos->getPrev();  
}
```

///DESPUÉS DE CIERTA POSICIÓN

```
template<class T>  
typename ListRecipes<T>::Node* ListRecipes<T>::getNextPosition(Node* pos) const {  
    if(!validPosition(pos) || pos == header->getPrev()) {  
        return nullptr;  
    }  
    if(pos->getNext() == header) {  
        return header->getNext();  
    }  
    return pos->getNext();  
}
```

```
template <class T>  
int ListRecipes<T>::compareByNameIngr(const T& orig, const T& copyc) {  
    return orig.getNameIngr().compare(copyc.getNameIngr());  
}
```

```
template <class T>  
int ListRecipes<T>::compareByTime(const T& orig, const T& copyc) {  
    return orig.getTime() - copyc.getTime();  
}
```

```
template <class T>  
int ListRecipes<T>::compareByTitle(const T& orig, const T& copyc) {  
    return orig.getTitle().compare(copyc.getTitle());  
}
```

```
template <class T>  
int ListRecipes<T>::compareByName(const T& orig, const T& copyc) {  
    return orig.getName().compare(copyc.getName());  
}
```

```
template <class T>  
int ListRecipes<T>::compareByCategory(const T& orig, const T& copyc) {  
    return orig.getCategory().compare(copyc.getCategory());  
}
```

```
template <class T>  
typename ListRecipes<T>::Node* ListRecipes<T>::findDatLin(const T& obj, int  
com(const T&, const T&)) const {  
    Node* aux(header->getNext());  
  
    while(aux != header) {  
        if(com(aux->getData(), obj) == 0) {  
            return aux;  
        }  
        aux = aux->getNext();  
    }  
}
```



```
    return nullptr;
}

template<class T>
void ListRecipes<T>::swapData(T& a, T& b) {
    T aux(a);
    a = b;
    b = aux;
}

template<class T>
void ListRecipes<T>::quickSort(int cmp(const Recetas&, const Recetas&)) {
    Node* aux(header->getNext());
    sortDataQuick(aux, getLastPosition(), cmp);
}

template<class T>
typename ListRecipes<T>::Node* ListRecipes<T>::sortDataQuick(Node* leftExt, Node*
rightExt, int cmp(const Recetas&, const Recetas&)) {
    Node* extrem(rightExt);
    Node* i(leftExt->getPrev());
    Node* j(leftExt);

    if(rightExt != nullptr && leftExt != rightExt && leftExt != rightExt-
>getNext()) {
        while(j != rightExt) {
            if(cmp(j->getData(), extrem->getData()) <= 0) {
                i = (i==nullptr) ? leftExt : i->getNext();
                swapData(i->getData(), j->getData());
            }
            j = j->getNext();
        }

        i = (i == nullptr) ? leftExt : i->getNext();
        swapData(i->getData(), extrem->getData());
        Node* pivot = i;
        sortDataQuick(leftExt, pivot->getPrev(), cmp);
        sortDataQuick(pivot->getNext(), rightExt, cmp);
    }
}

template <class T>
void ListRecipes<T>::writeToDisk(const string& fileName) {
    ofstream myFile;
    myFile.open(fileName, ios_base::trunc);

    if(!myFile.is_open()) {
        throw Exception("No se puede abrir el archivo para escritura,
writeFromDisk");
    }

    Position aux;
    aux = header->getNext();
}
```




```
while(aux != header) {
    myFile << aux->getData();
    aux = aux->getNext();
}
myFile.close();
}
```

```
///Lectura a disco
template <class T>
void ListRecipes<T>::readFromDisk(const string& fileName) {
    ifstream myFile;

    myFile.open(fileName);
    if(!myFile.is_open()) {
        throw Exception("No se pudo abrir el archivo para lectura, readFromDisk");
    }

    deleteAll();
    T myData;
    try {
        while(myFile >> myData) {
            insertData(getLastPosition(), myData);
        }
    }
    catch(Exception ex) {
        myFile.close();
        throw Exception(ex.what());
    }

    myFile.close();
}

template<class T>
string ListRecipes<T>::toString() const {
    string listComplete;
    Node* aux(header->getNext());
    while(aux != header) {
        listComplete += aux->getData().toString() + "\n";
        aux = aux->getNext();
    }
    return listComplete;
}

template <class T>
void ListRecipes<T>::deleteAll() {
    Node* aux;

    while(header->getNext() != header) {
        aux = header->getNext();
        header->setNext(aux->getNext());
        delete aux;
    }
    header->setPrev(header);
}
```



```
}
```

```
template<class T>
ListRecipes<T>& ListRecipes<T>::operator = (const ListRecipes<T>& obj) {
    deleteAll();
    copyAll(obj);
    return *this;
}
```

recipes.hpp

```
#ifndef RECIPES_HPP_INCLUDED
#define RECIPES_HPP_INCLUDED
#include <iostream>
#include <string>
#include <iomanip>
#include "author_name.hpp"
#include "ingredientes.hpp"
#include "ingredientList.hpp"

class Recetas {
private:
    NameA nom;
    ListIngredient<Ingredients> ingredient;
    std::string category;
    std::string title;
    int time;
    std::string process;
    void copyAll(const Recetas&);

public:
    Recetas();
    Recetas(const Recetas&);

    bool operator == (const Recetas& const);
    bool operator != (const Recetas& const);
    bool operator >= (const Recetas& const);
    bool operator <= (const Recetas& const);
    bool operator > (const Recetas& const);
    bool operator < (const Recetas& const);

    static int compareByCategory(const Recetas&, const Recetas&);
    static int compareByTitle(const Recetas&, const Recetas&);
    static int compareByTime(const Recetas&, const Recetas&);

    NameA getNom() const;
    ListIngredient<Ingredients> getIngredient() const;
    std::string getCategory() const;
    std::string getTitle() const;
    int getTime() const;
    std::string getProcess() const;

    void setNom(const NameA&);
}
```



```
void setIngredient(const ListIngredient<Ingredients>&);  
void setCategory(const std::string&);  
void setTitle(const std::string&);  
void setTime(const int&);  
void setProcess(const std::string&);  
  
std::string toString() const;  
  
friend std::istream& operator >> (std::istream&, Recetas&);  
friend std::ostream& operator << (std::ostream&, Recetas&);  
Recetas& operator = (const Recetas&);  
  
};
```

```
#endif // RECIPES_HPP_INCLUDED
```

recipes.cpp

```
#include "recipes.hpp"
```

```
using namespace std;
```

```
Recetas::Recetas() {}
```

```
Recetas::Recetas(const Recetas& r) {  
    copyAll(r);  
}
```

```
void Recetas::copyAll(const Recetas& obj) {  
    category = obj.category;  
    title = obj.title;  
    time = obj.time;  
    nom = obj.nom;  
    ingredient = obj.ingredient;  
    process = obj.process;  
}
```

```
Recetas& Recetas::operator=(const Recetas& obj) {  
    copyAll(obj);  
    return *this;  
}
```

```
bool Recetas::operator==(const Recetas& obj) const {  
    return title == obj.title;  
}
```

```
bool Recetas::operator!=(const Recetas& obj) const {  
    return title != obj.title;  
}
```

```
bool Recetas::operator>=(const Recetas& obj) const {  
    return title >= obj.title;  
}
```



```
    }

    bool Recetas::operator<=(const Recetas& obj) const {
        return title <= obj.title;
    }

    bool Recetas::operator>(const Recetas& obj) const {
        return title > obj.title;
    }

    bool Recetas::operator<(const Recetas& obj) const {
        return title < obj.title;
    }

    int Recetas::compareToTitle(const Recetas& a, const Recetas& b) {
        return a.title.compareTo(b.title);
    }

    int Recetas::compareToCategory(const Recetas& a, const Recetas& b) {
        return a.category.compareTo(b.category);
    }

    int Recetas::compareToTime(const Recetas& a, const Recetas& b) {
        return a.time - b.time;
    }

    NameA Recetas::getNom() const {
        return nom;
    }

    ListIngredient<Ingredients> Recetas::getIngredient() const {
        return ingredient;
    }

    string Recetas::getCategory() const {
        return category;
    }

    string Recetas::getTitle() const {
        return title;
    }

    int Recetas::getTime() const {
        return time;
    }

    string Recetas::getProcess() const {
        return process;
    }

    void Recetas::setNom(const NameA& _nom) {
        nom = _nom;
    }
}
```



```
void Recetas::setIngredient(const ListIngredient<Ingredients>& _ingredient) {
    ingredient = _ingredient;
}

void Recetas::setCategory(const string& _category) {
    category = _category;
}

void Recetas::setTitle(const string& _title) {
    title = _title;
}

void Recetas::setTime(const int& _time) {
    time = _time;
}

void Recetas::setProcess(const string& _process) {
    process = _process;
}

std::ostream& operator << (std::ostream& os, Recetas& obj) {
    ListIngredient<Ingredients>::Position aux;
    aux = obj.ingredient.getFirstPosition();

    os << obj.category << endl;
    os << obj.title << endl;
    os << obj.nom << endl;
    os << obj.time << endl;
    os << obj.process << endl;
    return os;
}

std::istream& operator >> (istream& is, Recetas& obj) {
    string myStr;
    getline(is, obj.category);
    getline(is, obj.title);
    is >> obj.nom;
    getline(is, myStr);
    obj.time = atoi(myStr.c_str());
    getline(is, obj.process);
    return is;
}

string Recetas::toString() const {
    string myStr;

    myStr += "\tCategoria: ";
    myStr += category;
    myStr += "\n";

    myStr += "\tReceta: ";
    myStr += title;
}
```



```
myStr += "\n";

myStr += "\tTiempo Estimado: ";
myStr += to_string(time);
myStr += " min.\n";

myStr += "\tAuthor: ";
myStr += nom.toString();
myStr += "\n";

myStr += "\n\tIngredientes: \n";
myStr += getIngredient().toString();
myStr += "\n";

myStr += "\tProcedimiento: \n";
myStr += process;
myStr += "\n";
return myStr;
}
```

ingredients.hpp

```
#ifndef INGREDIENTES_HPP_INCLUDED
#define INGREDIENTES_HPP_INCLUDED
#include <string>
#include <iostream>

class Ingredients {
private:
    std::string nameIngr;
    std::string quantity;
    void copyAll(const Ingredients&);

public:
    Ingredients();
    Ingredients(const Ingredients&);

    bool operator == (const Ingredients&) const;
    bool operator != (const Ingredients&) const;
    bool operator >= (const Ingredients&) const;
    bool operator <= (const Ingredients&) const;
    bool operator > (const Ingredients&) const;
    bool operator < (const Ingredients&) const;

    static int compareByNameIngr(const Ingredients&, const Ingredients&);

    std::string getNameIngr() const;
    std::string getQuantity() const;

    void setNameIngr(const std::string&);
    void setQuantity(const std::string&);

    std::string toString() const;
}
```



```
friend std::istream& operator >> (std::istream&, Ingredients&);  
friend std::ostream& operator << (std::ostream&, Ingredients&);  
Ingredients& operator = (const Ingredients&);  
  
};
```

```
#endif // INGREDIENTES_HPP_INCLUDED
```

ingredients.cpp

```
#include "ingredientes.hpp"
```

```
using namespace std;
```

```
Ingredients::Ingredients() { }
```

```
Ingredients::Ingredients(const Ingredients& i) {  
    copyAll(i);  
}
```

```
void Ingredients::copyAll(const Ingredients& obj) {  
    nameIngr = obj.nameIngr;  
    quantity = obj.quantity;  
}
```

```
Ingredients& Ingredients::operator=(const Ingredients& obj) {  
    copyAll(obj);  
    return *this;  
}
```

```
bool Ingredients::operator==(const Ingredients& obj) const {  
    return nameIngr == obj.nameIngr;  
}
```

```
bool Ingredients::operator!=(const Ingredients& obj) const {  
    return nameIngr != obj.nameIngr; //!(*this == c);  
}
```

```
bool Ingredients::operator>=(const Ingredients& obj) const {  
    return nameIngr >= obj.nameIngr; //!(*this < c);  
}
```

```
bool Ingredients::operator<=(const Ingredients& obj) const {  
    return nameIngr <= obj.nameIngr; /*this < c || *this == c;  
}
```

```
bool Ingredients::operator>(const Ingredients& obj) const {  
    return nameIngr > obj.nameIngr; //!(*this <= c);  
}
```

```
bool Ingredients::operator<(const Ingredients& obj) const {
```



```
    return nameIngr < obj.nameIngr;
}

int Ingredients::compareByNameIngr(const Ingredients& a, const Ingredients& b) {
    return a.nameIngr.compare(b.nameIngr);
}

string Ingredients::getNameIngr() const {
    return nameIngr;
}

string Ingredients::getQuantity() const {
    return quantity;
}

void Ingredients::setNameIngr(const string& _nameIngr) {
    nameIngr = _nameIngr;
}

void Ingredients::setQuantity(const string& _quantity) {
    quantity = _quantity;
}

std::istream& operator >> (istream& is, Ingredients& obj) {
    getline(is, obj.nameIngr);
    getline(is, obj.quantity);
    return is;
}

std::ostream& operator << (std::ostream& os, Ingredients& obj) {
    os << obj.nameIngr << "\t" << obj.quantity << std::endl << endl;
    return os;
}

string Ingredients::toString() const {
    string myStr;
    myStr += "\t--> ";
    myStr += nameIngr;
    myStr.resize(25, ' ');
    myStr += quantity;
    return myStr;
}
```




author name.hpp

```
#ifndef AUTHOR_NAME_HPP_INCLUDED
#define AUTHOR_NAME_HPP_INCLUDED
#include <iostream>
#include <string>
#include <iomanip>

class NameA {
private:
    std::string name;
    std::string lastname;
    void copyAll (const NameA&);

public:
    NameA ();
    NameA (const NameA&);

    bool operator == (const NameA& const;
    bool operator != (const NameA& const;
    bool operator >= (const NameA& const;
    bool operator <= (const NameA& const;
    bool operator > (const NameA& const;
    bool operator < (const NameA& const;

    static int compareByName (const NameA&, const NameA&);

    std::string getName () const;
    std::string getLastName () const;

    void setName (const std::string&);
    void setLastName (const std::string&);

    std::string toString () const;

    friend std::istream& operator >> (std::istream&, NameA&);
    friend std::ostream& operator << (std::ostream&, NameA&);
    NameA& operator = (const NameA&);
};
#endif // AUTHOR_NAME_HPP_INCLUDED
```



author_name.cpp

```
#include "author_name.hpp"
using namespace std;

NameA::NameA() { }

NameA::NameA(const NameA& n) {
    copyAll(n);
}

void NameA::copyAll(const NameA& obj) {
    name = obj.name;
    lastname = obj.lastname;
}

NameA& NameA::operator=(const NameA& obj) {
    copyAll(obj);
    return *this;
}

bool NameA::operator==(const NameA& obj) const {
    return name == obj.name && lastname == obj.lastname;
}

bool NameA::operator!=(const NameA& obj) const {
    return name != obj.name && lastname != obj.lastname;
}

bool NameA::operator>=(const NameA& obj) const {
    return name >= obj.name && lastname >= obj.lastname;
}

bool NameA::operator<=(const NameA& obj) const {
    return name <= obj.name && lastname <= obj.lastname;
}

bool NameA::operator>(const NameA& obj) const {
    return name > obj.name && lastname > obj.lastname;
}

bool NameA::operator<(const NameA& obj) const {
    return name < obj.name && lastname < obj.lastname;
}

string NameA::getName() const {
    return name;
}

string NameA::getLastName() const {
    return lastname;
}
```



```
void NameA::setName(const string& _name) {
    name = _name;
}

void NameA::setLastName(const string& _lastname) {
    lastname = _lastname;
}

istream& operator >> (istream& is, NameA& obj) {
    getline(is, obj.name);
    getline(is, obj.lastname);
    return is;
}

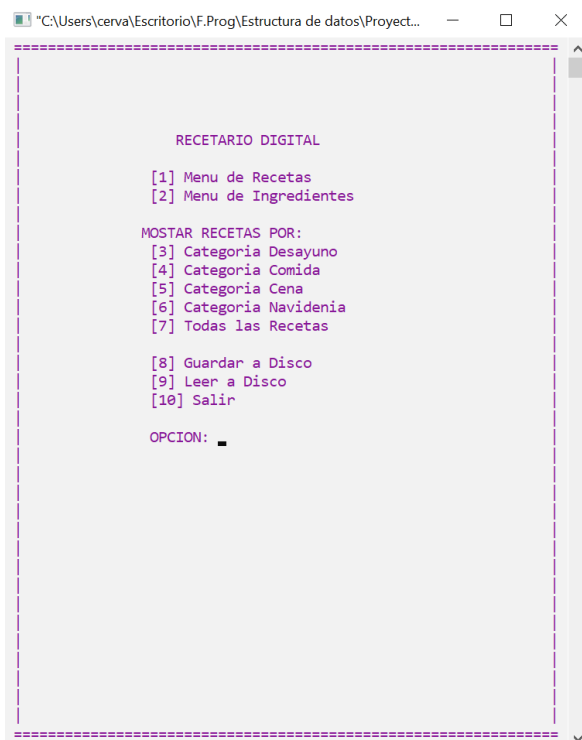
ostream& operator << (std::ostream& os, NameA& obj) {
    os << obj.name << " " << obj.lastname << std::endl;
    return os;
}

string NameA::toString() const {
    string myStr;
    myStr += name;
    myStr += " ";
    myStr += lastname;
    return myStr;
}
```

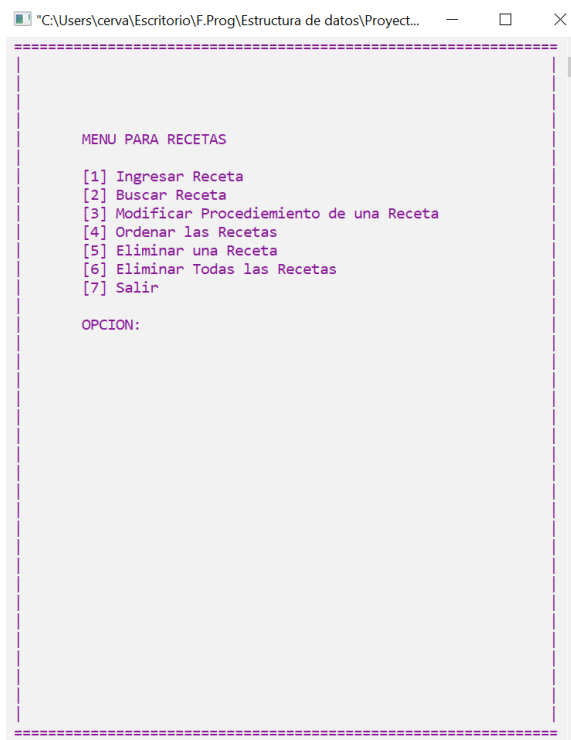


Impresiones de Pantalla:

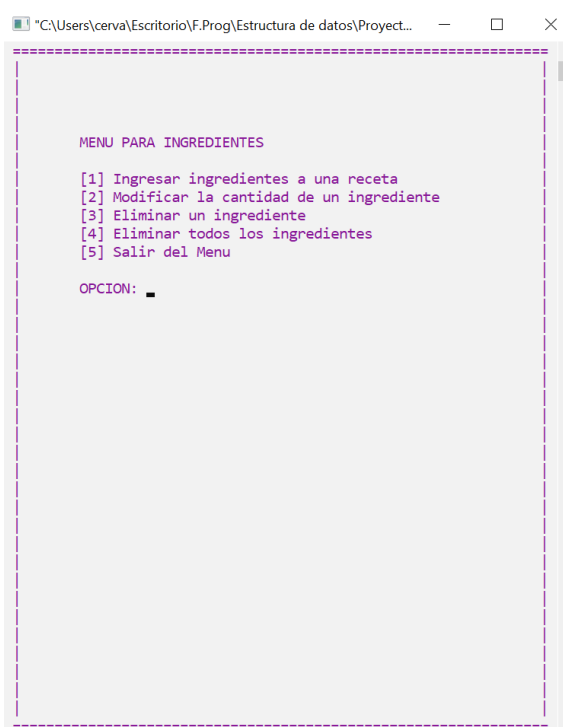
Menú Principal:



Menú Secundario de Recetas:

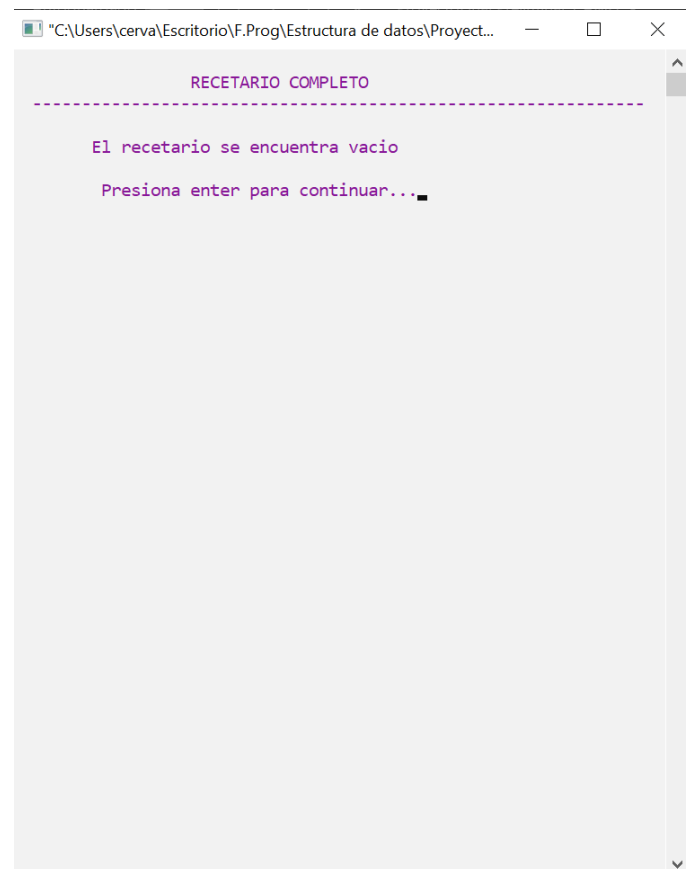
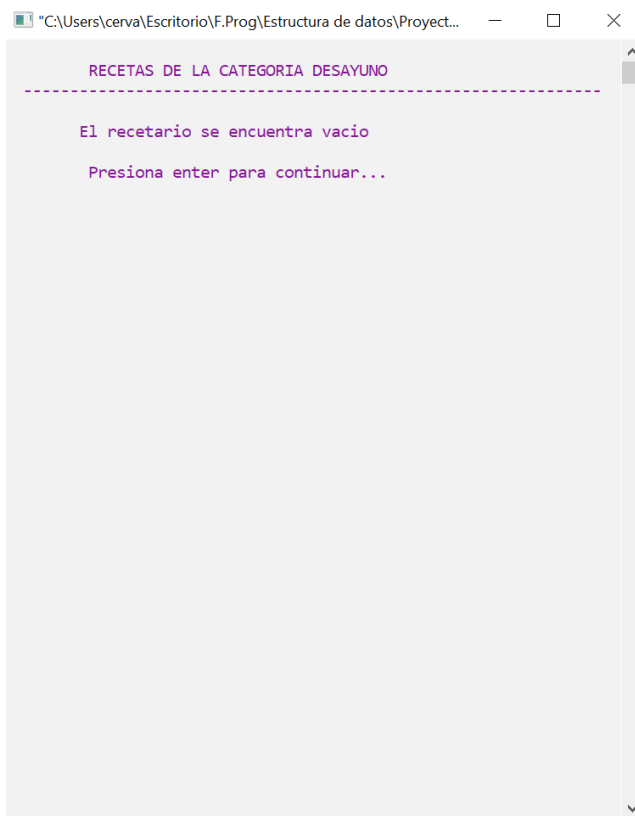


Menú Secundario de Ingredientes:





NOTA: Si el usuario entra a cualquier categoría de recetas o mostrar el recetario completo cuando está vacío, le aparecerá el mismo mensaje en todos los casos.





Ingresar Receta:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
=====
INGRESAR RECETA
=====
CATEGORIAS:
[1] Desayuno
[2] Comida
[3] Cena
[4] Navidenio
Seleccion: 1
Receta: Cereal
Nombre Autor: Pepe
Apellido Autor: Ortega
Ingrediente: Cereal
Cantidad: 200 gr
Desea agregar otro ingrediente? [y/n]: y
Ingrediente: Leche
Cantidad: 150 ml
Desea agregar otro ingrediente? [y/n]: y
Ingrediente: Platano
Cantidad: 2 pzas
Desea agregar otro ingrediente? [y/n]: n

Tiempo de Preparacion en min: 20
Proceso: Se vierte el cereal, la leche y los platanos.
```

->Se realiza una inyección de 12 recetas para realizar todas las pruebas del programa...

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
=====
INGRESAR RECETA
=====
CATEGORIAS:
[1] Desayuno
[2] Comida
[3] Cena
[4] Navidenio
Seleccion: 1
Receta: Cereal
Nombre Autor: Pepe
Apellido Autor: Ortega
Ingrediente: Cereal
Cantidad: 200 gr
Desea agregar otro ingrediente? [y/n]: y
Ingrediente: Leche
Cantidad: 150 ml
Desea agregar otro ingrediente? [y/n]: y
Ingrediente: Platano
Cantidad: 2 pzas
Desea agregar otro ingrediente? [y/n]: n

Tiempo de Preparacion en min: 20
Proceso: Se vierte el cereal, la leche y los platanos

----- RECETARIO ACTUALIZADO -----

Desea insertar otra receta? [y/n]:
```



Buscar Receta:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."
----- BUSQUEDA DE RECETA -----

Receta:
- Barbacoa

----- RECETA ENCONTRADA -----

Categoria: Comida
Receta: Barbacoa
Tiempo Estimado: 90 min.
Author: Caritina Huitrado

Ingredientes:
--> Agua          1 tza
--> Ajo           4 pzas
--> Ajonjolí      1 cdta
--> Canela        1 pza
--> Carne para barbacoa 2 kg
--> Chile mirasol  1 pza
--> Chiles anchos  2 pzas
--> Clavos de olor 2 pzas
--> Comino        1 cdita
--> Limones       10 pzas

Procedimiento:
Cortamos la carne en trocitos y la colocamos en la olla, le agrega
de canela, 4 hojas de laurel, clavo pimienta, 2 dientes de ajo, 1
de agua y agregamos a la carne, con agua medida; se deja cocer co
a carne y se da vuelta, chicoteamos los chiles con un tenedor y lo
bollas (reservamos). Licuamos los chiles de árbol, 1 diente de ajo
1 tomate; aparte doramos los tacos con grasa de la barbacoa y dora

Presiona enter para continuar...
```

Modificar Receta:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."
===== MODIFICAR PROCEDIMIENTO DE LA RECETA =====

Receta:
-Barbacoa

Receta:Barbacoa
Proceso Nuevo:
Freir todo, cuidarlo que no se queme y degustar.

----- MODIFICADO CON EXITO -----

Presiona enter para continuar...
```

Volvemos a buscar la receta para comprobar el cambio:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."
----- BUSQUEDA DE RECETA -----

Receta:
- Barbacoa

----- RECETA ENCONTRADA -----

Categoria: Comida
Receta: Barbacoa
Tiempo Estimado: 90 min.
Author: Caritina Huitrado

Ingredientes:
--> Agua          1 tza
--> Ajo           4 pzas
--> Ajonjolí      1 cdta
--> Canela        1 pza
--> Carne para barbacoa 2 kg
--> Chile mirasol  1 pza
--> Chiles anchos  2 pzas
--> Clavos de olor 2 pzas
--> Comino        1 cdita
--> Limones       10 pzas

Procedimiento:
Freir todo, cuidarlo que no se queme y degustar.

Presiona enter para continuar...
```



Ordenar Recetas Por Nombre:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."
=====
ORDENAR RECETAS
=====
[1] Ordenar por nombre
[2] Ordenar por tiempo de preparacion
[3] Salir
OPCION: 1_

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."
ORDENADO POR NOMBRE
-----
Categoria: Comida
Receta: Barbacoa
Tiempo Estimado: 90 min.
Author: Caritina Huitrado

Ingredientes:
--> Agua 1 tza
--> Ajo 4 pzas
--> Ajonjolí 1 cdta
--> Canela 1 pza
--> Carne para barbacoa 2 kg
--> Chile mirasol 1 pza
--> Chiles anchos 2 pzas
--> Clavos de olor 2 pzas
--> Comino 1 cdita
--> Limones 10 pzas

Procedimiento:
Freir todo, cuidarlo que no se queme y degustar.

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:
--> Avena 150 gr
--> Azucar 2 cda
--> Base 30 1 pza
--> Ciruela 250 gr
--> Crema para batir 500 ml
--> Grenetina 8 cda
--> Hojas de menta 4 pzas
--> Leche 1 lt
--> Nestle 1 pza
```




Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect... — □ ×

Picar todo, echarle miel y meterlo al refri.

Categoría: Comida
Receta: Camarones a la diablo
Tiempo Estimado: 140 min.
Author: Norberto Cervantes Gomez

Ingredientes:

--> Ajo	1 pza
--> Camaron	1 kg
--> Catsup	500 ml
--> Chipotle	1 pza
--> Galletas o tostadas	50 pzas
--> Jugo de limon	4 pzas
--> Knorr de camaron	1 pza
--> Mantequilla	1 kg
--> Pimienta	1 cedita
--> Refractario	1 pza
--> Salsa huichola	50 ml

Procedimiento:

Licuamos el knorr caldo de camaron, el chipotle, un trozo de cebolla, un diente de ajo y 250 ml de agua, aparte en un sartén con una barra de mantequilla y unas gotitas de aceite freímos un trozo de cebolla y un diente de ajo y freímos el camaron ya en color naranja, con pimienta integramos el adobo ya licuado, sazonomos de sal dejamos que de un hervor y servimos con arroz blanco y verduras al vapor.

Categoría: Navidenio
Receta: Ensalada navidenia
Tiempo Estimado: 40 min.
Author: Blas Heriberto Briseño

Ingredientes:

--> Crema	600 gr
--> Duraznos	1 lata
--> Lechera	1 pza
--> Malvaviscos	150 gr

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect... — □ ×

con la manzana el kiwi.

Categoría: Cena
Receta: Ensalada verde
Tiempo Estimado: 25 min.
Author: Lola Cervantes

Ingredientes:

--> Aceite de oliva	130 ml
--> Apio	1 pzas
--> Chayote	1 pza
--> Cherri jitomate	1 pza
--> Chicaro	250 gr
--> Chile morron	2 pzas
--> Chiles serranos	3 pzas
--> Lechuga italiana	1 pza
--> Manojito de perejil	1 pza
--> Manojito espinaca	125 gr
--> Nuez	100 gr
--> Pepino cambrian	4 pzas
--> Queso mozzarella	250 gr

Procedimiento:

Picamos todo en cubos y vaciamos a un tazón, ahí mismo el apio, nuez, perejil y lechuga finamente picado; al terminar integramos con el aceite de oliva y espolvoreamos con el queso mozzarella o manchego y servimos con una guarnición.

Categoría: Comida
Receta: Filete miñón
Tiempo Estimado: 90 min.
Author: Aldo Carriola

Ingredientes:

--> Arroz	250 gr
--> Crema bechamel	500 ml
--> Engrasador	10 ml
--> Filete en caña	1 kg
--> Knorr suiza	1 pza



Ordenar Por Tiempo de Preparación:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

=====
ORDENAR RECETAS
=====

[1] Ordenar por nombre
[2] Ordenar por tiempo de preparacion
[3] Salir
OPCION: 2_

=====

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

ORDENADO POR TIEMPO DE PREPARACION
=====

Categoria: Cena
Receta: Ensalada verde
Tiempo Estimado: 25 min.
Author: Lola Cervantes

Ingredientes:
--> Aceite de oliva      130 ml
--> Apio                 1 pzas
--> Chayote              1 pza
--> Cherri jitomate      1 pza
--> Chicaro              250 gr
--> Chile morron         2 pzas
--> Chiles serranos      3 pzas
--> Lechuga italiana     1 pza
--> Manojito de perejil  1 pza
--> Manojito espinaca    125 gr
--> Nuez                 100 gr
--> Pepino cambrain      4 pzas
--> Queso mozzarella    250 gr

Procedimiento:
Picamos todo en cubos y vaciamos a un tazón, ahí mismo el apio, nu
ez, perejil y lechuga finamente picado; al terminar integramos con
el aceite de oliva y espolvoreamos con el queso mozzarella o manche
go y servimos con una guarnición.

Categoria: Desayuno
Receta: Huevos con jamon
Tiempo Estimado: 25 min.
Author: Martha Sandoval Herrera

Ingredientes:
--> Aceite                1 chta
--> Cebolla              1 pza
--> Consome de Tomate     1 pza
```



```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
Categoria: Cena
Receta: Platanos al horno
Tiempo Estimado: 25 min.
Author: Diana Krystal Arrellano

Ingredientes:
--> Barra canela      1 pza
--> Botella agua      270 ml
--> Brandi o Ron       250 ml
--> Endulzante         150 gr
--> Maicena            150 gr
--> Mantequilla        500 gr
--> Molde              1 pza
--> Nestle             1 pza
--> Nieve              250 gr
--> Platano macho      6 pzas
--> Ralladura de naranja 1 cda
--> Salero             1 cdita
--> Vainilla           2 cda

Procedimiento:
En un tazón agregamos el azúcar, la vainilla, el agua, la ralladura de naranja, la canela y la maicena mezclamos e integramos el brandi. En un molde acomodamos rebanadas de platano y encima una salsa, después otra capa de platano, hasta terminar, horneamos a 200 grados por 20 min. e ir checando hasta que el platano este cocido y la salsa baje un poco, servimos con una bola de nieve.

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:
--> Avena              150 gr
--> Azucar             2 cda
--> Base 30            1 pza
--> Ciruela            250 gr
--> Crema para batir   500 ml
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
Nestle, crema, integramos todo vaciado al refractario y decoramos con la manzana el kiwi.

Categoria: Navidenio
Receta: Rollitos navidenios
Tiempo Estimado: 40 min.
Author: Edgar Ortega

Ingredientes:
--> Carnation          1 pza
--> Charola            1 pza
--> Fruta deshidratada  380 gr
--> Leche              100 ml
--> Naranja            1 pza
--> Pasta Hojaldre      500 gr
--> Polvo de canela     1 cdita
--> Saborizante        250 ml
--> Spray Aceite       250 ml
--> Vainilla           1 cda

Procedimiento:
En un tazón colocamos las frutas finamente picados, 1 cda. De vainilla, la Nestle, y llevamos al fuego hasta que se forme una pasta (palanqueta) retiramos del fuego y dejamos enfriar. Aparte extendemos en harina espolvoreada en tiras de pasta de 10 o 15 cm. ancho quedando delgada la pasta, cortamos rectangulos acomodando una porcion de fruta, doblando los dos extremos y barnizamos con agua, al terminar freímos con aceite caliente, ya dorados espolvoreamos con azúcar y canela.

Categoria: Navidenio
Receta: Lomo navidenio
Tiempo Estimado: 50 min.
Author: Penny Coral Reyes

Ingredientes:
--> Almendra           60 gr
--> Arandano deshidratado 70 gr
--> Chipotle           2 pzas
```

Eliminar una Receta:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
=====
ELIMINAR UNA RECETA
=====

Receta a eliminar:
- Rollitos Navidenios

Eliminando...

No existe la receta
Presiona enter para continuar...
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
=====
BUSQUEDA DE RECETA
=====

Receta:
- Rollitos navidenio
No pudimos encontrar la receta :(
```



Agregar Ingrediente:

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."

INGRESAR OTRO INGREDIENTE

Receta: Barras de cereal

Barras de cereal

Ingredientes Actuales:

--> Avena	150 gr
--> Azucar	2 cda
--> Base 30	1 pza
--> Ciruela	250 gr
--> Crema para batir	500 ml
--> Grenetina	8 cda
--> Hojas de menta	4 pzas
--> Leche	1 lt
--> Nestle	1 pza
--> Yogurt de Ciruela	1 lt

Nuevo Ingrediente: Danonino
Cantidad: 8 pzas

----- INGREDIENTES ACTUALIZADOS -----

Presiona enter para continuar...

Seleccionar "C:\Users\cerva\Escritorio\F.Prog\Estructura de da..."

BUSQUEDA DE RECETA

Receta:
- Barras de cereal

----- RECETA ENCONTRADA -----

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:

--> Avena	150 gr
--> Azucar	2 cda
--> Base 30	1 pza
--> Ciruela	250 gr
--> Crema para batir	500 ml
--> Danonino	8 pzas
--> Grenetina	8 cda
--> Hojas de menta	4 pzas
--> Leche	1 lt
--> Nestle	1 pza
--> Yogurt de Ciruela	1 lt

Procedimiento:
Picar todo, echarle miel y meterlo al refri.

Presiona enter para continuar...



Modificar Cantidad de Ingrediente:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
=====
MODIFICAR CANTIDAD DE UN INGREDIENTE
=====

Receta: Barras de cereal

=====

Barras de cereal
--> Avena          150 gr
--> Azucar         2 cda
--> Base 30        1 pza
--> Ciruela        250 gr
--> Crema para batir 500 ml
--> Danonino       8 pzas
--> Grenetina      8 cda
--> Hojas de menta  4 pzas
--> Leche           1 lt
--> Nestle         1 pza
--> Yogurt de Ciruela 1 lt

Ingrediente: Grenetina
Cambio de Cantidad: 400 gr

----- MODIFICADO CON EXITO -----

Presiona enter para continuar..._
```

```
Seleccionar "C:\Users\cerva\Escritorio\F.Prog\Estructura de da...
=====
BUSQUEDA DE RECETA
=====

Receta:
- Barras de cereal

----- RECETA ENCONTRADA -----

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:
--> Avena          150 gr
--> Azucar         2 cda
--> Base 30        1 pza
--> Ciruela        250 gr
--> Crema para batir 500 ml
--> Danonino       8 pzas
--> Grenetina      400 gr
--> Hojas de menta  4 pzas
--> Leche           1 lt
--> Nestle         1 pza
--> Yogurt de Ciruela 1 lt

Procedimiento:
Picar todo, echarle miel y meterlo al refri.

Presiona enter para continuar..._
```



Eliminar un Ingrediente:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  — □ ×
=====
ELIMINAR UN INGREDIENTE
=====

Receta: Barras de cereal

-----
Barras de cereal
--> Avena          150 gr
--> Azucar         2 cda
--> Base 30        1 pza
--> Ciruela        250 gr
--> Crema para batir 500 ml
--> Danonino       8 pzas
--> Grenetina      400 gr
--> Hojas de menta  4 pzas
--> Leche          1 lt
--> Nestle         1 pza
--> Yogurt de Ciruela 1 lt

Ingrediente: Base 30

Eliminando...

----- ELIMINADO CON EXITO -----

Presiona enter para continuar...

=====

"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  — □ ×
=====
BUSQUEDA DE RECETA
=====

Receta:
- Barras de cereal

----- RECETA ENCONTRADA -----

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:
--> Avena          150 gr
--> Azucar         2 cda
--> Ciruela        250 gr
--> Crema para batir 500 ml
--> Danonino       8 pzas
--> Grenetina      400 gr
--> Hojas de menta  4 pzas
--> Leche          1 lt
--> Nestle         1 pza
--> Yogurt de Ciruela 1 lt

Procedimiento:
Picar todo, echarle miel y meterlo al refri.

Presiona enter para continuar...
```



Eliminar Todos los Ingredientes de una Receta:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  X

ELIMINAR LISTA DE INGREDIENTES

Receta: Huevos con jamon

Eliminando...

----- SE ELIMINARON TODOS LOS INGREDIENTES -----

Presiona enter para continuar...
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  X

BUSQUEDA DE RECETA

-----

Receta:
- Huevos con jamon

----- RECETA ENCONTRADA -----

Categoria: Desayuno
Receta: Huevos con jamon
Tiempo Estimado: 25 min.
Author: Martha Sandoval Herrera

Ingredientes:

Procedimiento:
Pones a precalentar un sartén a llama baja, pasados 5 min. colocamos el aceite. Aparte cortamos el jamon, jitomate, cebolla y ajo en trocitos y reservamos en un plato, una vez calentado el aceite vamos a cocción del jamon y dejamos dorar un poco, posteriormente le echamos los huevos procurando no dejar caer cascara al sartén y por ultimo le vertimos la sal y el oregano; revolvemos por aprox. 10 min. vertimos el consome de tomate y apagamos la estufa, retiramos del fuego y montamos en el plato de nuestra preferencia.

Presiona enter para continuar...
```



Categoría Desayuno:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

RECETAS DE LA CATEGORIA DESAYUNO
-----

                Categoria: Desayuno
Receta: Frijoles puercos
Tiempo Estimado: 130 min.
Author: Juan Gutierrez

Ingredientes:
--> Caldo                1 cdita
--> Cebolla              1 pza
--> Chile ancho          1 pza
--> Chiles de arbol      10 pza
--> Chorizo              30 gr
--> Frijol               500 gr
--> Jitomates            4 pzas
--> Manita de cerdo      2 pzas
--> Pierna de cerdo      500 gr
--> Tocino               150 gr

Procedimiento:
Cortamos la salchicha en media lunas, el jamon en cubitos y el to
cino en tiras por separado.Desvenamos y lavamos los chiles ancho, m
irasol, cascabel y árbol al gusto, los doramos en manteca caliente
por separado y dejamos remojar en caldo de frijoles caliente sepa
rando el chile cascabel. Freímos el tocino en donde se frió los ch
iles (reservamos). Licuamos un diente de ajo, los chiles, clavos d
e olor y pimientos, colamos y freímos en el mismo sartén freímos e
l adobo y agregamos los frijoles, dejamos hervir y machacamos, agr
egamos las carnes frías, granos de elote, se dejan por 15 min. Y d
ecoramos con chiles cascabel, totopos y tiras de queso.

                Categoria: Desayuno
Receta: Huevos con jamon
Tiempo Estimado: 25 min.
Author: Martha Sandoval Herrera

Ingredientes:
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

egamos las carnes frías, granos de elote, se dejan por 15 min. Y d
ecoramos con chiles cascabel, totopos y tiras de queso.

                Categoria: Desayuno
Receta: Huevos con jamon
Tiempo Estimado: 25 min.
Author: Martha Sandoval Herrera

Ingredientes:

Procedimiento:
Pones a precalentar un sartén a llama baja, pasados 5 min. colocam
os el aceite. Aparte cortamos el jamon, jitomate, cebolla y ajo en
trozos y reservamos en un plato, una vez calentado el aceite va
ciamos el jamon y dejamos dorar un poco, posteriormente le echamos
los huevos procurando no dejar caer cascarras al sartén y por últi
mo le vertimos la sal y el oregano; revolvemos por aprox. 10 min.
vertimos el consome de tomate y apagamos la estufa, retiramos del
fuego y montamos en el plato de nuestra preferencia.

                Presiona enter para continuar...
```




Categoría Comida:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."

RECETAS DE LA CATEGORIA COMIDA

-----

          Categoria: Comida
Receta: Barbacoa
Tiempo Estimado: 90 min.
Author: Caritina Huitrado

Ingredientes:
--> Agua                1 tza
--> Ajo                 4 pzas
--> Ajonjolí            1 cdta
--> Canela              1 pza
--> Carne para barbacoa 2 kg
--> Chile mirasol       1 pza
--> Chiles anchos       2 pzas
--> Clavos de olor      2 pzas
--> Comino              1 cdita
--> Limones             10 pzas

Procedimiento:
Freir todo, cuidarlo que no se queme y degustar.

          Categoria: Comida
Receta: Camarones a la diablo
Tiempo Estimado: 140 min.
Author: Norberto Cervantes Gomez

Ingredientes:
--> Ajo                 1 pza
--> Camaron             1 kg
--> Catsup              500 ml
--> Chipotle            1 pza
--> Galletas o tostadas 50 pzas
--> Jugo de limon        4 pzas
--> Knorr de camaron    1 pza
--> Mantequilla         1 kg
--> Pimienta            1 cdita
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect..."

Procedimiento:
Licuamos el knorr caldo de camaron, el chipotle, un trozo de cebolla, un diente de ajo y 250 ml de agua, aparte en un sartén con una barra de mantequilla y unas gotitas de aceite freímos un trozo de cebolla y un diente de ajo y freímos el camaron ya en color naranja, con pimienta integramos el adobo ya licuado, sazonamos de sal dejamos que de un hervor y servimos con arroz blanco y verduras al vapor.

          Categoria: Comida
Receta: Filete miñon
Tiempo Estimado: 90 min.
Author: Aldo Carriola

Ingredientes:
--> Arroz                250 gr
--> Crema bechamel       500 ml
--> Engrasador           10 ml
--> Filete en caña       1 kg
--> Knorr suiza          1 pza
--> Pimienta             1 pza
--> Rebanadas tocino     300 gr
--> Refractario         1 pza
--> Salsa inglesa        500 ml
--> Sazonador            1 cdita

Procedimiento:
Al filete se le retira toda grasa, enseguida los medallones de 1 cm. En un tazón integramos salsa inglesa, pimienta, un ajo picado, media cucharada de knorr suiza o sal. Impregnamos los medallones y se deja reposar por 30 min, formamos el filetito y envolvemos toda la orilla con una tira de tocino y presionamos con palillos, la terminamos freímos en aceite caliente, y dejamos reposar en servitolas. Retiramos del fuego ya con la crema integrada y servimos el filete con el espejo de verduras, arroz y verduras.

Presiona enter para continuar...
```



Categoría Cena:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

RECETAS DE LA CATEGORIA COMIDA
-----

          Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:
--> Avena          150 gr
--> Azucar         2 cda
--> Ciruela        250 gr
--> Crema para batir 500 ml
--> Danonino       8 pzas
--> Grenetina      400 gr
--> Hojas de menta  4 pzas
--> Leche          1 lt
--> Nestle         1 pza
--> Yogurt de Ciruela 1 lt

Procedimiento:
Picar todo, echarle miel y meterlo al refri.

          Categoria: Cena
Receta: Ensalada verde
Tiempo Estimado: 25 min.
Author: Lola Cervantes

Ingredientes:
--> Aceite de oliva 130 ml
--> Apio            1 pzas
--> Chayote         1 pza
--> Cherri jitomate 1 pza
--> Chicaro         250 gr
--> Chile morron    2 pzas
--> Chiles serranos 3 pzas
--> Lechuga italiana 1 pza
--> Manojito de perejil 1 pza
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

--> Queso mozzarella 250 gr

Procedimiento:
Picamos todo en cubos y vaciamos a un tazón, ahí mismo el apio, nuez, perejil y lechuga finamente picado; al terminar integramos con el aceite de oliva y espolvoreamos con el queso mozzarella o manchego y servimos con una guarnición.

          Categoria: Cena
Receta: Plátanos al horno
Tiempo Estimado: 25 min.
Author: Diana Krystal Arrellano

Ingredientes:
--> Barra canela      1 pza
--> Botella agua     270 ml
--> Brandi o Ron      250 ml
--> Endulzante        150 gr
--> Maicena           150 gr
--> Mantequilla       500 gr
--> Molde             1 pza
--> Nestle            1 pza
--> Nieve             250 gr
--> Plátano macho     6 pzas
--> Ralladura de naranja 1 cda
--> Salero            1 cda
--> Vainilla          2 cda

Procedimiento:
En un tazón agregamos el azúcar, la vainilla, el agua, la ralladura de naranja, la canela y la maicena mezclamos e integramos el brandi. En un molde acomodamos rebanadas de plátano y encima una salsa, después otra capa de plátano, hasta terminar, hornear a 200 g por 20 min. e ir chequeando hasta que el plátano esté cocido y la salsa baje un poco, servimos con una bola de nieve.

Presiona enter para continuar...
```



Categoría Navideña:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...

RECETAS DE LA CATEGORIA NAVIDENIA

-----

Categoria: Navidenio
Receta: Ensalada navidenia
Tiempo Estimado: 40 min.
Author: Blas Heriberto Briseño

Ingredientes:
--> Crema          600 gr
--> Duraznos       1 lata
--> Lechera        1 pza
--> Malvaviscos    150 gr
--> Manzanas rojas  6 pzas
--> Moscada Nuez   1 cda
--> Nuez en mitades 150 gr
--> Pasas o arandanos 150 gr
--> Pechuga de pollo 500 gr
--> Piña en almibar 1 lata

Procedimiento:
En un tazón colocamos el durazno en cubos, pinia, manzana con piel
ebrada,500 gr de cda. De sal,250 gr de cda. Polvo de nuez moscada
refractorio y decoramos con la manzana el kiwi.

Categoria: Navidenio
Receta: Lomo navidenio
Tiempo Estimado: 50 min.
Author: Penny Coral Reyes

Ingredientes:
--> Almendra       60 gr
--> Arandano deshidratad70 gr
--> Chipotle       2 pzas
--> Dientes de ajo  2 pzas
--> Jamon de pierna 180 gr
--> Lomo           1 kg
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
refractorio y decoramos con la manzana el kiwi.

Categoria: Navidenio
Receta: Lomo navidenio
Tiempo Estimado: 50 min.
Author: Penny Coral Reyes

Ingredientes:
--> Almendra       60 gr
--> Arandano deshidratad70 gr
--> Chipotle       2 pzas
--> Dientes de ajo  2 pzas
--> Jamon de pierna 180 gr
--> Lomo           1 kg
--> Mangos        2 pzas
--> Manzana        2 pzas
--> Nuez           60 gr
--> Pimienta       1 cdita
--> Queso Crema    500 gr
--> Suiza knorr    1 cda

Procedimiento:
Limpiamos la carne, ya limpia, le hacemos orificios por todos lado
. Introducimos salchicha en rodajas, tocino picado y asado, aránd
ellamos en poco grasa. Colocamos en una charola, cubrimos de alumi
para la crema freímos la cebolla rebanada con 500 gr barra de man
con la maicena, el clavel y los champiñones. Dejamos hervir sazón
rebanado.

Presiona enter para continuar...
```



Mostrar Todas Las Recetas:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  — □ ×

RECETARIO COMPLETO

-----

Categoria: Comida
Receta: Barbacoa
Tiempo Estimado: 90 min.
Author: Caritina Huitrado

Ingredientes:
--> Agua                1 tza
--> Ajo                 4 pzas
--> Ajonjolí            1 cdta
--> Canela              1 pza
--> Carne para barbacoa 2 kg
--> Chile mirasol       1 pza
--> Chiles anchos       2 pzas
--> Clavos de olor      2 pzas
--> Comino              1 cdita
--> Limones             10 pzas

Procedimiento:
Cortamos la carne en trocitos y la colocamos en la olla, le agrega
mos el aceite, 500 gr de cda. De sal. Licuamos un trozo de canela,
4 hojas de laurel, clavo pimienta, 2 dientes de ajo, los chiles, ½
cdta. Mejorana, 2 pizcas de comino, ½ lt. de agua y agregamos a l
a carne, con agua medida; se deja cocer con el vinagre y el aceite
por 50 min. Se deja enfriar la carne y se da vuelta, chicoteamos
los chiles con un tenedor y los freímos con un poco de grasa de la
barbacoa y las cebollas (reservamos). Licuamos los chiles de árbo
l, 1 diente de ajo, 100 gr de cdta de sal, un poco de agua. Y vaci
amos el tomate; aparte doramos los tacos con grasa de la barbacoa
y dorados se sirven con la guarnición.

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  — □ ×

Categoria: Cena
Receta: Barras de cereal
Tiempo Estimado: 30 min.
Author: Ma. Ramirez

Ingredientes:
--> Avena                150 gr
--> Azucar               2 cda
--> Base 30              1 pza
--> Ciruela              250 gr
--> Crema para batir     500 ml
--> Grenetina            8 cda
--> Hojas de menta       4 pzas
--> Leche                1 lt
--> Nestle               1 pza
--> Yogurt de Ciruela    1 lt

Procedimiento:
Doramos la avena en seco a fuego lento, cuando ya esta dejamos enf
riar. En un tazón colocamos los cereales, la nuez picada, crema de
cacahuete y miel de maple 2 cda. sooperas, formando una plancha, v
aciando en una charola, congelamos de 20 a 25 min. y cortamos en b
arritas, se pueden acompañar vaseando un vaso de leche.

Categoria: Comida
Receta: Camarones a la diablo
Tiempo Estimado: 140 min.
Author: Norberto Cervantes Gomez

Ingredientes:
--> Ajo                 1 pza
--> Camaron             1 kg
--> Catsup              500 ml
--> Chipotle            1 pza
--> Galletas o tostadas  50 pzas
--> Jugo de limon        4 pzas
--> Knorr de camaron     1 pza
--> Mantequilla          1 kg
```



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
ICOM – Ingeniería en Computación
Módulo Estructura de Datos



```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
Procedimiento:
Picamos todo en cubos y vaciamos a un tazón, ahí mismo el apio, nu
ez, perejil y lechuga finamente picado; al terminar integramos con
el aceite de oliva y espolvoreamos con el queso mozzarella o manche
go y servimos con una guarnición.

Categoria: Comida
Receta: Filete miñón
Tiempo Estimado: 90 min.
Author: Aldo Carriola

Ingredientes:
--> Arroz                250 gr
--> Crema bechamel       500 ml
--> Engrasador           10 ml
--> Filete en caña       1 kg
--> Knorr suiza          1 pza
--> Pimienta             1 pza
--> Rebanadas tocino     300 gr
--> Refractario          1 pza
--> Salsa inglesa        500 ml
--> Sazonador            1 cdita

Procedimiento:
Al filete se le retira toda grasa, enseguida los medallones de 1 c
m. En un tazón integramos salsa inglesa, pimienta, un ajo picado, y
media cucharada de knorr suiza o sal. Impregnamos los medallones y
se deja reposar por 30 min, formamos el filetito y envolvemos tod
a la orilla con una tira de tocino y presionamos con palillos, la
terminar freímos en aceite caliente, y dejamos reposar en servitoa
llas. Retiramos del fuego ya con la crema integrada y servimos el
filete con el espejo de verduras, arroz y verduras.

Categoria: Desayuno
Receta: Frijoles puercos
Tiempo Estimado: 130 min.
Author: Juan Gutierrez

Ingredientes:
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...
os por 45 min. A 1 hora aparte para la crema freímos la cebolla re
banada con 500 gr barra de mantequilla y un trozo de ajo, agregamo
s la leche diluida con la maicena, el clavel y los champiñones. De
jamos hervir sazonando de sal ya guisado retiramos del fuego y ser
vimos rebanado.

Categoria: Cena
Receta: Platanos al horno
Tiempo Estimado: 25 min.
Author: Diana Krystal Arrellano

Ingredientes:
--> Barra canela         1 pza
--> Botella agua         270 ml
--> Brandi o Ron         250 ml
--> Endulzante           150 gr
--> Maicena              150 gr
--> Mantequilla          500 gr
--> Molde                 1 pza
--> Nestle               1 pza
--> Nieve                250 gr
--> Platano macho        6 pzas
--> Ralladura de naranja 1 cda
--> Salero               1 cdita
--> Vainilla             2 cda

Procedimiento:
En un tazón agregamos el azúcar, la vainilla, el agua, la ralladur
a de naranja, la canela y la maicena mezclamos e integramos el bra
ndi. En un molde acomodamos rebanadas de platano y encima una sals
a, después otra capa de platano, hasta terminar, hornearmos a 200 g
rados por 20 min. e ir chequeando hasta que el platano este cocido y
la salsa baje un poco, servimos con una bola de nieve.

Presiona enter para continuar...
```



Guardar a Disco:

Ingresamos más recetas y guardamos.

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

=====
RECETARIO DIGITAL

[1] Menu de Recetas
[2] Menu de Ingredientes

MOSTAR RECETAS POR:
[3] Categoria Desayuno
[4] Categoria Comida
[5] Categoria Cena
[6] Categoria Navidenia
[7] Todas las Recetas

[8] Guardar a Disco
[9] Leer a Disco
[10] Salir

OPCION: 8

Escribiendo al disco...

----- Accion Completada -----

Presiona enter para continuar..._
=====
```



Eliminar Todas Las Recetas:

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  
-----  
MENU PARA RECETAS  
  
[1] Ingresar Receta  
[2] Buscar Receta  
[3] Modificar Procedimiento de una Receta  
[4] Ordenar las Recetas  
[5] Eliminar una Receta  
[6] Eliminar Todas las Recetas  
[7] Salir  
  
OPCION: 6  
  
Eliminando...  
  
----- SE ELIMINARON TODAS LAS RECETAS -----
```

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  
RECETARIO COMPLETO  
-----  
El recetario se encuentra vacio  
Presiona enter para continuar...  
-----
```



Leer de Disco:

Borramos el recetario y leemos de disco para comprobar que funcione adecuadamente.

```
"C:\Users\cerva\Escritorio\F.Prog\Estructura de datos\Proyect...  -  □  ×

=====
RECETARIO DIGITAL

[1] Menu de Recetas
[2] Menu de Ingredientes

MOSTAR RECETAS POR:
[3] Categoria Desayuno
[4] Categoria Comida
[5] Categoria Cena
[6] Categoria Navidenia
[7] Todas las Recetas

[8] Guardar a Disco
[9] Leer a Disco
[10] Salir

OPCION: 9

Leyendo del disco...

----- Accion Completada -----

Presiona enter para continuar...

=====

"RECETARIO COMPLETO

Categoria: Desayuno
Receta: Huevos con jamon
Tiempo Estimado: 25 min.
Author: Martha Sandoval Herrera

Ingredientes:
--> Aceite      1 cdta
--> Cebolla     1 pza
--> Consome de Tomate  1
--> Diente de Ajo    1 pza
--> Huevos       3 pzas
--> Jamon        300 gr
--> Jitomate     2 pzas
--> Oregano      1 cdita
--> Sal 1 cdita
--> Tortillas    4 pzas

Procedimiento:
Pones a precalentar un sartén a llama baja, pasados 5 min. colocam
lla y ajo en trocitos y reservamos en un plato, una vez calentado
osteriormente le echamos los huevos procurando no dejar caer casca
gano; revolvemos por aprox. 10 min. vertimos el consome de tomate
n el plato de nuestra preferencia.

Categoria: Comida
Receta: Barbacoa
Tiempo Estimado: 90 min.
Author: Caritina Huitrado

Ingredientes:
--> Agua        1 tza
--> Ajo 4 pzas
--> Ajonjolí    1 cdta
--> Canela      1 pza

"

=====
RECETARIO DIGITAL

[1] Menu de Recetas
[2] Menu de Ingredientes

MOSTAR RECETAS POR:
[3] Categoria Desayuno
[4] Categoria Comida
[5] Categoria Cena
[6] Categoria Navidenia
[7] Todas las Recetas

[8] Guardar a Disco
[9] Leer a Disco
[10] Salir

OPCION: 10

HASTA LA PROXIMA!! :D

Process returned 5 (0x5)  execution time : 443.485 s
Press any key to continue.
```

Salir:



Conclusión

Al término del proyecto puedo concluir que el manejo de memoria dinámica es un tanto delicado si no sabes trabajar bien con nodos (posiciones en la memoria); se realizaron cambios en el código de la entrega preliminar, dado que para la entrega final se realizó con otros criterios, las modificaciones fueron relativamente sencillas, lo único que me dio error fue con el método de quickSort para la clase de recetas y al momento de implementar el guardado a disco, dado que trabajamos con lista doblemente enlazada con encabezado Dummy, este tipo de listas nunca tienen valor nulo, entonces no podía salir del ciclo a menos que diera posición inválida, pero me cerraba el programa. Otro método que me llevo a dar problemas fue al insertar las recetas porque en cada inserción me duplicaba los ingredientes y aunque le implementaba el “deleteAll()” me percate que lo estaba haciendo en el orden incorrecto, yo lo ponía después de terminar de guardar los datos y era antes para asegurar que siempre estuviera vacía la lista de ingredientes.

Una vez realizados los cambios y corregido los fallos con el manejo de parámetros en la lista y el menú, pudimos culminar el curso con el proyecto antes presentado, utilice algunos recursos extras para mejor apreciación, en lo personal quedo satisfecha con el producto final y con lo aprendido en el curso, aunque me falta practica más varios temas.