



Tecnológico de Monterrey

“Actividad Integral estructura de datos lineales”

Programación de estructuras de datos y algoritmos fundamentales

Profesora: María Valentina Narváez Terán

Bruno Fernando Zabala Peña - A00838627

16/10/2024

Problemática

En un mundo que está en constante movimiento, la industria del food delivery está tomando un impulso bastante importante, pues ofrece ventajas tanto para clientes y proveedores por igual, tales como:

- Abarcar un mercado mucho más amplio sin la necesidad de una expansión física de un negocio.
- Tener acceso a alimentos sin la necesidad de transportarse.
- Ahorro de tiempo al no tener la necesidad de cocinar.
- Mejorar la imagen de un negocio y hacerlo mucho más competitivo.
- Aumentar las ventas.
- Etc.

Sin embargo, y aunque existen herramientas y plataformas que están haciendo cada vez más sencillo adentrarse en este modelo de ventas (Uber Eats, Rappi, etc.), los negocios aún deben contar con una infraestructura o sistemas que le permitan llevar un monitoreo de lo que implica este servicio, como el registro de ventas o clientes.

En esta primera etapa de la solución, se creó una aplicación capaz de manejar un conjunto de órdenes de food delivery, ordenarlas de manera cronológica y permitir al usuario la búsqueda de datos específicos.

Código

El cambio más significativo respecto a la primera etapa, fue la implementación del algoritmo de ordenamiento quicksort de forma iterativa y haciendo uso de una pila con una lista doblemente enlazada. Esto representa un aspecto positivo respecto, pues facilitó el proceso de creación de las funciones para ordenar los datos según su fecha y restaurante. Una desventaja que surgió a partir de este cambio fue el tener que tomar en cuenta el tipo de dato con el que se trabajaba en los nodos en todo momento, pues estos eran inicializados con un tipo genérico, sin embargo, esto no derivó en problemas de funcionamiento.

La función para asignar un valor numérico según la fecha (get_value) no sufrió alteraciones. Esta recibe un string como parámetro y devuelve un valor entero. Primero, se busca en el string recibido la fecha encontrando el índice en el que aparece el primer espacio y este fragmento de texto es almacenado en una variable. Este proceso se repite hasta que se tienen también el día y la hora. Una vez que se tienen estos datos, se inicializa una variable y se le asigna un valor inicial según el mes encontrado (este se busca en un arreglo que contiene las iniciales de los meses y el índice en el que se encontró es multiplicado por 100000000), a esa luego se le suma el valor del día multiplicado por 100000000, las horas multiplicadas por 10000, los minutos multiplicado por 100 y los segundos.

```
// Funcion para asignar un valor a cada orden de acuerdo a su fecha y hora
// Complejidad: O(1)
int get_value(string order){
    string months[12] = {"ene", "Feb", "Mar", "Abr", "May", "Jun", "Jul", "Ago", "Sep", "Oct", "Nov", "Dic"};
    string aux;

    int month_end_index = order.find(" ");
    string month = order.substr(0, month_end_index);

    aux = order.substr(month_end_index + 1);

    int day_end_index = aux.find(" ");
    string day = aux.substr(0, day_end_index);

    aux = aux.substr(day_end_index + 1);

    int time_end_index = aux.find(" ");
    string time = aux.substr(0, time_end_index);

    int value = 0;

    // Pasa por el arreglo de meses
    for(int i = 0; i < 12; i++){
        if(month == months[i]){
            // Asigna un valor segun el mes
            value += (i + 1) * 100000000;
            break;
        }
    }

    value += stoi(day) * 1000000;

    int hour_end_index = aux.find(":");
    string hour = time.substr(0, hour_end_index);

    aux = time.substr(hour_end_index + 1);

    int minute_end_index = aux.find(":");
    string minute = aux.substr(0, minute_end_index);

    aux = aux.substr(minute_end_index + 1);

    int second_end_index = aux.find(" ");
    string second = aux.substr(0, second_end_index);

    value += stoi(hour) * 10000;
    value += stoi(minute) * 100;
    value += stoi(second);

    return value;
}
```

De forma similar, se implementó una nueva función para obtener el restaurante de una orden (get_restaurant). Esta recibe un string como entrada y devuelve un dato del mismo tipo. Primero, se busca el índice en el que inicia el nombre del restaurante (después del indicador R:) y en el que termina (antes del indicador O:) y se extrae el string a partir de estos.

```
// Funcion para obtener el nombre del restaurante de una orden
// Complejidad: O(1)
string get_restaurant(string order){
    string restaurant;

    // Encuentra los índices en los que inicia y termina el nombre del restaurante
    int restaurant_start_index = order.find("R:");
    int restaurant_end_index = order.find(" O:");

    // Extrae el nombre del restaurante
    restaurant = order.substr(restaurant_start_index + 2, restaurant_end_index - restaurant_start_index - 2);

    return restaurant;
}
```

Como se mencionó al inicio, se cambió el algoritmo de ordenamiento a quickSort, para los requerimientos de la etapa, se crearon dos funciones individuales, una para ordenar por fechas y otra para hacerlo según el restaurante (alfabéticamente).

```
// Funcion para particionar la lista de ordenes (ordenar por restaurantes)
template<class T>
Node<T>* partitionR(List<T> datos, Node<T> *L, Node<T> *R){
    Node<T> *j = L;
    Node<T> *i = NULL;
    Node<T> *pivot = R;

    T aux = "";

    // Compara con pivot e intercambios
    while (j != R){
        // cout << "\tComparo " << j->value << " y " << pivot->value << endl;

        if (get_restaurant(j->value) < get_restaurant(pivot->value))
        {
            if (i == NULL)
            {
                i = L;
            }
            else
            {
                i = i->next;
            }

            aux = j->value;
            j->value = i->value;
            i->value = aux;
        }

        j = j->next;
    }

    // Coloca el valor pivote en su sitio
    if (i == NULL)
    {
        i = L;
    }
    else
    {
        i = i->next;
    }

    aux = i->value;
    i->value = pivot->value;
    pivot->value = aux;

    return i; // Devuelve la direccion de donde quedo el valor pivote
}
```

```

// quicksort
// Ordena por restaurantes
// Complejidad: O(nlog(n))
template<class T>
void quickSortR(List<T> datos){
    Stack<Node<string>*> stack;

    Node<T> *L = datos.getFirst();
    Node<T> *R = datos.getLast();
    Node<T> *pivot = NULL;

    // El primer par de L y R entran en la stack
    // Esta particion inicial son el inicio y fin de la lista de datos
    stack.push(L);
    stack.push(R);

    // En cada iteracion un par L y R salen de la stack
    // La particion de L a R es reordenada
    // Y segun donde quede el pivote, agrega nuevos pares de L y R a la stack
    while (stack.isEmpty() == false){
        R = stack.pop();
        L = stack.pop();

        pivot = partitionR(datos, L, R);

        if (pivot != L && pivot->next != L){
            stack.push(L);
            stack.push(pivot->prev);
        }

        if (pivot != R && pivot->prev != R){
            stack.push( pivot->next );
            stack.push( R );
        }

        // datos.showList(); // <---- Descomenta para ver el paso a paso
    }
}

```

```

// Funcion para particionar la lista de ordenes (ordenar por fechas)
template<class T>
Node<T>* partition( List<T> datos, Node<T> *L, Node<T> *R ){
    Node<T> *j = L;
    Node<T> *i = NULL;
    Node<T> *pivot = R;

    T aux = "";

    // Compara con pivot e intercambios
    while (j != R){
        // cout << "\tComparo " << j->value << " y " << pivot->value << endl;

        if (get_value(j->value) < get_value(pivot->value))
        {
            if (i == NULL)
            {
                i = L;
            }
            else
            {
                i = i->next;
            }

            aux = j->value;
            j->value = i->value;
            i->value = aux;
        }

        j = j->next;
    }

    // Coloca el valor pivote en su sitio
    if (i == NULL)
    {
        i = L;
    }
    else
    {
        i = i->next;
    }

    aux = i->value;
    i->value = pivot->value;
    pivot->value = aux;

    return i; // Devuelve la direccion de donde quedo el valor pivote
}

```

```

// quicksort
// Ordena por fechas
// Complejidad:  $O(n\log(n))$ 
template<class T>
void quickSort(List<T> datos){
    Stack<Node<string>*> stack;

    Node<T> *L = datos.getFirst();
    Node<T> *R = datos.getLast();
    Node<T> *pivot = NULL;

    // El primer par de L y R entran en la stack
    // Esta particion inicial son el inicio y fin de la lista de datos
    stack.push(L);
    stack.push(R);

    // En cada iteracion un par L y R salen de la stack
    // La particion de L a R es reordenada
    // Y segun donde quede el pivote, agrega nuevos pares de L y R a la stack
    while (stack.isEmpty() == false){
        R = stack.pop();
        L = stack.pop();

        pivot = partition(datos, L, R);

        if (pivot != L && pivot->next != L){
            stack.push(L);
            stack.push(pivot->prev);
        }

        if (pivot != R && pivot->prev != R){
            stack.push( pivot->next );
            stack.push( R );
        }

        // datos.showList(); // <---- Descomenta para ver el paso a paso
    }
}

```

Aunque se cambio de método, este tiene la misma complejidad temporal promedio que mergeSort, por lo que no hubo un cambio significativo en el rendimiento del programa.

Para la búsqueda de órdenes, se agregó la opción de hacer búsquedas por restaurantes:

```

// Funcion para buscar ordenes por restaurante
// Complejidad: O(n)
void search_restaurant(string restaurant, List<string> orders, int n){
    // Archivo en el que se almacenan los resultados de busqueda
    ofstream search_results("outputs/search_results.txt");

    List<string> results;
    int i, j;

    // Pasa por todos los elementos de la lista de ordenes
    for(i = 0; i < n; i++){
        // Busca coincidencias exactas con el nombre del restaurante
        if(get_restaurant(orders.getIndex(i)->value) == restaurant){
            // Guarda las coincidencias en la lista de resultados
            results.insertLast(orders.getIndex(i)->value);
        }
    }

    // Imprime los resultados de la busqueda y los escribe en el archivo de salida
    cout << "Resultados de la busqueda (" << results.getSize() << " coincidencias):" << endl;
    search_results << "Resultados de la busqueda \"" << restaurant << "\" (" << results.getSize() << " coincidencias):" << endl;

    if(results.getSize() == 0){
        cout << "No hay resultados para este restaurante";
    }

    else{
        quickSort(results);

        for(j = 0; j < results.getSize(); j++){
            cout << j + 1 << "- " << results.getIndex(j)->value << endl;
            search_results << j + 1 << "- " << results.getIndex(j)->value << endl;
        }
    }

    search_results.close();
}

```

Y se mantuvieron las funciones de búsqueda por fecha:

```

// Funcion para buscar ordenes por fecha y hora
// Complejidad: O(n)
void single_search(string date, int date_value, List<string> orders, int n){
    // Archivo en el que se almacenan los resultados de busqueda
    ofstream search_results("outputs/search_results.txt");

    List<string> results;
    int i, j;

    // Pasa por todos los elementos de la lista de ordenes
    for(i = 0; i < n; i++){
        // Busca coincidencias exactas con la fecha dada
        if(get_value(orders.getIndex(i)->value) == date_value){
            // Guarda las coincidencias en la lista de resultados
            results.insertLast(orders.getIndex(i)->value);
        }
    }

    // Imprime los resultados de la busqueda y los escribe en el archivo de salida
    cout << "Resultados de la busqueda (" << results.getSize() << " coincidencias):" << endl;
    search_results << "Resultados de la busqueda \"" << date << "\" (" << results.getSize() << " coincidencias):" << endl;

    if(results.getSize() == 0){
        cout << "No hay resultados para esta fecha";
    }

    else{
        for(j = 0; j < results.getSize(); j++){
            cout << j + 1 << "- " << results.getIndex(j)->value << endl;
            search_results << j + 1 << "- " << results.getIndex(j)->value << endl;
        }
    }

    search_results.close();
}

```

```

// Funcion para buscar ordenes por rango de fechas y horas
// Complejidad: O(n)
void range_search(string lower_date, string upper_date, int lower_date_value, int upper_date_value, List<string> orders, int n){
    // Archivo en el que se almacenan los resultados de busqueda
    ofstream search_results("outputs/search_results.txt");

    List<string> results;
    int i, j;

    // Pasa por todos los elementos de la lista de ordenes
    for(i = 0; i < n; i++){
        // Busca coincidencias exactas con las fechas dadas
        if(get_value(orders.getIndex(i)->value) >= lower_date_value && get_value(orders.getIndex(i)->value) <= upper_date_value){
            // Guarda las coincidencias en la lista de resultados
            results.insertLast(orders.getIndex(i)->value);
        }
    }

    // Imprime los resultados de la busqueda y los escribe en el archivo de salida
    cout << "Resultados de la busqueda (" << results.getSize() << " coincidencias):" << endl;
    search_results << "Resultados de la busqueda entre \"" << lower_date << "\" y \"" << upper_date << "\" (" << results.getSize() << " coincidencias):" << endl;

    if(results.getSize() == 0){
        cout << "No hay resultados para este rango de fechas";
    }

    else{
        for(j = 0; j < results.getSize(); j++){
            cout << j + 1 << "- " << results.getIndex(j)->value << endl;
            search_results << j + 1 << "- " << results.getIndex(j)->value << endl;
        }
    }

    search_results.close();
}

```

Se mantiene el funcionamiento propuesto en la primera etapa, incluso para la búsqueda por restaurantes. Se itera por la lista de órdenes y se almacenan las coincidencias (encontradas con `get_restaurant` y `get_value` respectivamente) en una lista que luego es mostrada al usuario, junto con el número de resultados encontrados. La salida mostrada en la consola es además almacenada en un archivo de texto al que el usuario puede acceder.

Debido a los requerimientos establecidos para esta etapa, se dejó la organización del código por módulos (.h) y ahora todas las funciones están contenidas dentro del mismo archivo de código.

Pruebas

Ordenamiento por restaurante


```

PS C:\Users\bruno\OneDrive\Documentos\situacion problema\Etapa 2> ./main.exe orders.tx
Primeras 10 ordenes
Feb 20 17:25:26 R:Ana O:Tarta de Manzana(149)
Ago 18 14:13:49 R:Ana O:Pescado empapelado(305)
Sep 25 19:52:23 R:Ana O:Tacos de Pescado(298)
Jun 17 13:45:4 R:Ana O:Pollo a la Brasa(307)
Dic 27 20:8:2 R:Ana O:Spaghetti Carbonara(301)
Dic 25 20:15:35 R:Ana O:Pollo Kiev(444)
Mar 13 13:19:54 R:Ana O:Filete Mignon(462)
Dic 23 16:34:32 R:Ana O:Croquetas de Jamon(152)
May 22 18:10:14 R:Ana O:Sopa de Tomate(150)
Jul 24 19:43:31 R:Ana O:Pollo al Curry(303)

```

```

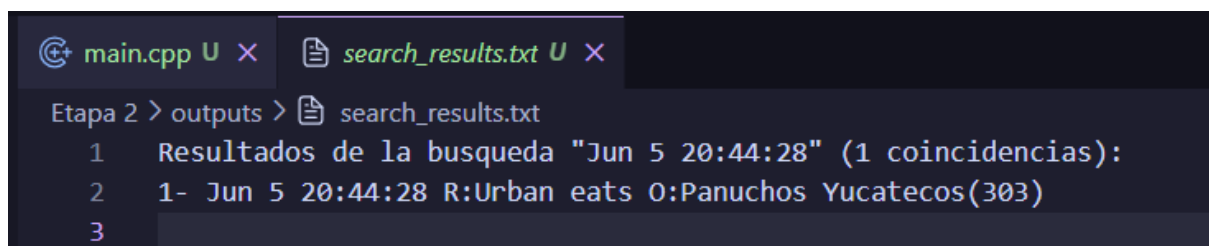
main.cpp U x sorted_orders.txt U x
Etapa 2 > outputs > sorted_orders.txt
1 Feb 20 17:25:26 R:Ana O:Tarta de Manzana(149)
2 Ago 18 14:13:49 R:Ana O:Pescado empapelado(305)
3 Sep 25 19:52:23 R:Ana O:Tacos de Pescado(298)
4 Jun 17 13:45:4 R:Ana O:Pollo a la Brasa(307)
5 Dic 27 20:8:2 R:Ana O:Spaghetti Carbonara(301)
6 Dic 25 20:15:35 R:Ana O:Pollo Kiev(444)
7 Mar 13 13:19:54 R:Ana O:Filete Mignon(462)
8 Dic 23 16:34:32 R:Ana O:Croquetas de Jamon(152)
9 May 22 18:10:14 R:Ana O:Sopa de Tomate(150)
10 Jul 24 19:43:31 R:Ana O:Pollo al Curry(303)
11 May 23 14:15:9 R:Ana O:Osso Buco(455)
12 Abr 23 14:32:46 R:Ana O:Pasta al Pesto(301)
13 Mar 6 19:10:13 R:Ana O:Dim Sum(301)
14 Jul 27 18:31:21 R:Ana O:Bacalao a la Vizcaina(446)
15 Sep 13 22:21:25 R:Ana O:Pechuga de Pollo Rellena(303)
16 Dic 15 13:27:40 R:Ana O:Fideua(300)
17 Sep 14 17:0:55 R:Ana O:escargots a la Bourguignonne(313)
18 Dic 1 16:7:40 R:Ana O:Sopa de Tomate(150)
19 Dic 18 15:6:9 R:Ana O:Hummus con Pita(154)
20 Sep 24 21:13:33 R:Ana O:Enchiladas Suizas(296)
21 Dic 3 12:13:37 R:Ana O:ensalada Waldorf(155)
22 Dic 8 15:28:4 R:Ana O:Bacalao a la Vizcaina(446)
23 Nov 5 17:25:18 R:Ana O:Moussaka(303)
24 Feb 15 18:55:57 R:Ana O:Salmon a la Parrilla(307)
25 Feb 1 15:45:49 R:Ana O:Cordero Asado(454)
26 Dic 7 19:7:39 R:Ana O:Pad Thai(300)
27 Mar 21 13:0:28 R:Ana O:Pasta con Mariscos(449)
28 Dic 2 18:35:53 R:Ana O:Lomo Saltado(303)
29 Mar 19 14:17:57 R:Ana O:Fideua(300)
30 Sep 8 16:47:46 R:Ana O:Cordero Asado(454)
31 Abr 2 12:50:45 R:Ana O:Arroz Frito(295)
32 Feb 9 14:24:7 R:Ana O:Beef Stroganoff(449)
33 Mar 27 15:50:26 R:Ana O:Gnocchi al Pesto(301)
34 Jun 11 18:53:45 R:Ana O:Croquetas de Jamon(152)
35 Nov 5 21:51:14 R:Ana O:Yakitori de Pollo(296)
36 Feb 23 15:57:17 R:Ana O:Sopa de Tortilla(151)
37 Nov 23 21:47:7 R:Ana O:Tacos de Barbacoa(298)

```

Búsqueda por fecha y hora

Fecha y hora específica

```
-----  
Busqueda de ordenes  
-----  
Seleccione una opcion:  
1- Busqueda por fecha  
2- Busqueda por restaurante  
> 1  
  
-----  
Busqueda de ordenes por fecha y hora  
-----  
Seleccione una opcion:  
1- Busqueda individual  
2- Busqueda por rango  
> 1  
  
Ingrese la fecha y hora a buscar (ej. "Jun 12 06:51:16"):  
> Jun 5 20:44:28  
  
Resultados de la busqueda (1 coincidencias):  
1- Jun 5 20:44:28 R:Urban eats O:Panuchos Yucatecos(303)
```



```
main.cpp U X search_results.txt U X  
Etapa 2 > outputs > search_results.txt  
1 Resultados de la busqueda "Jun 5 20:44:28" (1 coincidencias):  
2 1- Jun 5 20:44:28 R:Urban eats O:Panuchos Yucatecos(303)  
3
```

Rango de fechas

Busqueda de ordenes

Seleccione una opcion:

- 1- Busqueda por fecha
 - 2- Busqueda por restaurante
- > 1

Busqueda de ordenes por fecha y hora

Seleccione una opcion:

- 1- Busqueda individual
 - 2- Busqueda por rango
- > 2

Ingrese las fechas y horas (ej. "Jun 12 06:51:16"):

Limite inferior: Jun 12 00:00:00

Limite superior: Jun 12 23:59:59

Resultados de la busqueda (38 coincidencias):

- 1- Jun 12 17:43:46 R:Bistro Boulevard O:Pollo al Limon(294)
- 2- Jun 12 19:52:8 R:City Bistro O:Sushi Roll(301)
- 3- Jun 12 18:2:18 R:Classic Cafe O:Arroz Frito(291)
- 4- Jun 12 15:29:40 R:Dos hermanos O:ensalada Cesar(298)
- 5- Jun 12 15:9:19 R:El Cafe de la Abuela O:Crepas con Nutella(131)
- 6- Jun 12 14:59:43 R:El Cafe de la Abuela O:Tacos al Pastor(132)
- 7- Jun 12 15:27:11 R:El Cafe de la Plaza O:Kebab de Cordero(273)
- 8- Jun 12 19:32:7 R:El Grande O:Fajitas de Res(296)
- 9- Jun 12 20:36:38 R:El Mar O:Mole Poblano(261)
- 10- Jun 12 18:28:55 R:El Otro Rincon O:Chimichurri Steak(410)
- 11- Jun 12 18:46:3 R:El Restaurante de Ana O:Arroz con Leche(137)
- 12- Jun 12 21:33:10 R:El Rincon del Chef O:Pollo Tikka Masala(296)
- 13- Jun 12 18:38:38 R:El Uro O:Raviolis de espinaca(280)
- 14- Jun 12 19:46:12 R:El dragon O:Pozole Blanco(136)
- 15- Jun 12 13:10:5 R:La Bella Notte O:Coq au Vin(421)
- 16- Jun 12 13:52:24 R:La Cocina de Pablo O:Tacos de Birria(273)
- 17- Jun 12 19:0:4 R:La Cocina de la Abuela O:Burrata con Tomate(272)
- 18- Jun 12 15:34:51 R:La Cocina del Chef O:Panqueques con Miel(130)

```
main.cpp U x search_results.txt U x
Etapa 2 > outputs > search_results.txt
1 Resultados de la búsqueda entre "Jun 12 00:00:00" y "Jun 12 23:59:59" (38 coincidencias):
2 1- Jun 12 17:43:46 R:Bistro Boulevard O:Pollo al Limon(294)
3 2- Jun 12 19:52:8 R:City Bistro O:Sushi Roll(301)
4 3- Jun 12 18:2:18 R:Classic Cafe O:Arroz Frito(291)
5 4- Jun 12 15:29:40 R:Dos hermanos O:ensalada Cesar(298)
6 5- Jun 12 15:9:19 R:El Cafe de la Abuela O:Crepas con Nutella(131)
7 6- Jun 12 14:59:43 R:El Cafe de la Abuela O:Tacos al Pastor(132)
8 7- Jun 12 15:27:11 R:El Cafe de la Plaza O:Kebab de Cordero(273)
9 8- Jun 12 19:32:7 R:El Grande O:Fajitas de Res(296)
10 9- Jun 12 20:36:38 R:El Mar O:Mole Poblano(261)
11 10- Jun 12 18:28:55 R:El Otro Rincon O:Chimichurri Steak(410)
12 11- Jun 12 18:46:3 R:El Restaurante de Ana O:Arroz con Leche(137)
13 12- Jun 12 21:33:10 R:El Rincon del Chef O:Pollo Tikka Masala(296)
14 13- Jun 12 18:38:38 R:El Uro O:Raviolis de espinaca(280)
15 14- Jun 12 19:46:12 R:El dragon O:Pozole Blanco(136)
16 15- Jun 12 13:10:5 R:La Bella Notte O:Coq au Vin(421)
17 16- Jun 12 13:52:24 R:La Cocina de Pablo O:Tacos de Birria(273)
18 17- Jun 12 19:0:4 R:La Cocina de la Abuela O:Burrata con Tomate(272)
19 18- Jun 12 15:34:51 R:La Cocina del Chef O:Panqueques con Miel(130)
20 19- Jun 12 12:52:19 R:La Francesa O:Fajitas de Res(264)
21 20- Jun 12 20:31:24 R:La Pasticceria O:Curry de Garbanzos(308)
22 21- Jun 12 18:48:57 R:La esquina del Taco O:ensalada Griega(129)
23 22- Jun 12 16:21:10 R:Las salsas O:Sopa de Cebolla(151)
24 23- Jun 12 16:41:20 R:Las salsas O:Costillas BBQ(445)
25 24- Jun 12 16:26:51 R:Rios O:empanadas de Carne(151)
26 25- Jun 12 20:5:26 R:The Bee O:ensalada Caprese(131)
27 26- Jun 12 14:57:53 R:The Farmhouse Cafe O:Arroz Frito(287)
28 27- Jun 12 18:14:33 R:The Green Room O:Arrachera Asada(316)
29 28- Jun 12 13:9:48 R:The Local Bistro O:Pollo Tikka Masala(281)
30 29- Jun 12 18:42:58 R:The Lounge O:Ratatouille(283)
31 30- Jun 12 14:49:21 R:The Modern Palate O:Ceviche Mixto(279)
32 31- Jun 12 12:38:3 R:The Rustic Spoon O:Tacos de Lengua(279)
33 32- Jun 12 20:3:43 R:The Savory O:Cochinita Pibil(276)
34 33- Jun 12 16:4:8 R:The Seafood Shack O:Moussaka(277)
35 34- Jun 12 15:12:59 R:The Stylish Diner O:Ceviche Mixto(292)
36 35- Jun 12 19:3:51 R:The Stylish Diner O:Filete Mignon(439)
37 36- Jun 12 17:28:16 R:The elegant Kitchen O:Ribeye Steak(413)
```

Búsqueda por restaurante

Busqueda de ordenes

Seleccione una opcion:

- 1- Busqueda por fecha
- 2- Busqueda por restaurante
- > 2

Busqueda de ordenes por restaurante

Ingrese el nombre del restaurante tomando en cuenta mayusculas (ej. "The Bee"):
> The Bee

Resultados de la busqueda (80 coincidencias):

- 1- ene 1 18:15:14 R:The Bee O:Chop Suey(257)
- 2- ene 14 19:33:34 R:The Bee O:Lasana Vegetariana(259)
- 3- Feb 1 12:54:42 R:The Bee O:Tacos de Barbacoa(256)
- 4- Feb 3 13:52:14 R:The Bee O:Gnocchi al Pesto(258)
- 5- Feb 8 17:6:17 R:The Bee O:Lasana Vegetariana(259)
- 6- Feb 9 18:29:13 R:The Bee O:Panuchos Yucatecos(252)
- 7- Feb 11 21:44:59 R:The Bee O:Shawarma de Pollo(254)
- 8- Feb 17 19:16:2 R:The Bee O:Salmon Gravlax(258)
- 9- Feb 21 13:32:57 R:The Bee O:Filete de Res(393)
- 10- Feb 22 14:53:26 R:The Bee O:Ceviche de Pulpo(263)
- 11- Feb 26 12:32:40 R:The Bee O:Osso Buco(390)
- 12- Mar 2 19:42:47 R:The Bee O:Tacos de Lengua(256)
- 13- Mar 3 19:39:39 R:The Bee O:Sopa de Pescado(382)
- 14- Mar 4 14:50:46 R:The Bee O:Beef Stroganoff(385)
- 15- Mar 9 12:25:22 R:The Bee O:Curry de Garbanzos(255)
- 16- Mar 9 17:1:35 R:The Bee O:Burrata con Tomate(255)
- 17- Mar 13 16:17:19 R:The Bee O:Pollo al Curry(260)
- 18- Mar 15 19:22:31 R:The Bee O:Pollo Alfredo(259)
- 19- Mar 16 14:47:58 R:The Bee O:Pollo a la Brasa(263)
- 20- Mar 18 22:19:35 R:The Bee O:Tacos al Pastor(131)
- 21- Mar 21 19:30:46 R:The Bee O:Sashimi de Atun(259)
- 22- Mar 24 20:49:58 R:The Bee O:Sopa de Pescado(382)
- 23- Mar 26 16:54:29 R:The Bee O:Bruschetta(128)
- 24- Mar 27 18:29:35 R:The Bee O:Cochinillo Asado(395)

```

main.cpp U x search_results.txt U x
Etapa 2 > outputs > search_results.txt
1 Resultados de la búsqueda "The Bee" (80 coincidencias):
2 1- ene 1 18:15:14 R:The Bee O:Chop Suey(257)
3 2- ene 14 19:33:34 R:The Bee O:Lasana Vegetariana(259)
4 3- Feb 1 12:54:42 R:The Bee O:Tacos de Barbacoa(256)
5 4- Feb 3 13:52:14 R:The Bee O:Gnocchi al Pesto(258)
6 5- Feb 8 17:6:17 R:The Bee O:Lasana Vegetariana(259)
7 6- Feb 9 18:29:13 R:The Bee O:Panuchos Yucatecos(252)
8 7- Feb 11 21:44:59 R:The Bee O:Shawarma de Pollo(254)
9 8- Feb 17 19:16:2 R:The Bee O:Salmon Gravlax(258)
10 9- Feb 21 13:32:57 R:The Bee O:Filete de Res(393)
11 10- Feb 22 14:53:26 R:The Bee O:Ceviche de Pulpo(263)
12 11- Feb 26 12:32:40 R:The Bee O:Osso Buco(390)
13 12- Mar 2 19:42:47 R:The Bee O:Tacos de Lengua(256)
14 13- Mar 3 19:39:39 R:The Bee O:Sopa de Pescado(382)
15 14- Mar 4 14:50:46 R:The Bee O:Beef Stroganoff(385)
16 15- Mar 9 12:25:22 R:The Bee O:Curry de Garbanzos(255)
17 16- Mar 9 17:1:35 R:The Bee O:Burrata con Tomate(255)
18 17- Mar 13 16:17:19 R:The Bee O:Pollo al Curry(260)
19 18- Mar 15 19:22:31 R:The Bee O:Pollo Alfredo(259)
20 19- Mar 16 14:47:58 R:The Bee O:Pollo a la Brasa(263)
21 20- Mar 18 22:19:35 R:The Bee O:Tacos al Pastor(131)
22 21- Mar 21 19:30:46 R:The Bee O:Sashimi de Atun(259)
23 22- Mar 24 20:49:58 R:The Bee O:Sopa de Pescado(382)
24 23- Mar 26 16:54:29 R:The Bee O:Bruschetta(128)
25 24- Mar 27 18:29:35 R:The Bee O:Cochinillo Asado(395)
26 25- Abr 3 12:57:57 R:The Bee O:Sopa de Tortilla(130)
27 26- Abr 13 14:1:59 R:The Bee O:Tiradito de Pescado(256)
28 27- Abr 15 13:23:55 R:The Bee O:Tiramisu(128)
29 28- Abr 16 20:15:55 R:The Bee O:Fish and Chips(257)
30 29- Abr 17 18:23:42 R:The Bee O:Costillas de Cordero(393)
31 30- Abr 18 14:44:47 R:The Bee O:Pasta al Pesto(258)
32 31- Abr 26 20:19:57 R:The Bee O:Sopa de Tomate(129)
33 32- May 2 19:36:40 R:The Bee O:Tempura de Vegetales(254)
34 33- May 3 13:44:1 R:The Bee O:Falafel con Hummus(254)
35 34- May 12 14:25:50 R:The Bee O:Cazuela de Mariscos(385)
36 35- May 13 14:43:9 R:The Bee O:Sopa de Tomate(129)
37 36- May 14 12:52:50 R:The Bee O:Crepas de Champinones(253)

```

Referencias

Findus Foodservice & <https://www.findusfoodservices.es/humans.txt>. (s. f.). *¿Qué es el*

Delivery Food? Findus Foodservices.

<https://www.findusfoodservices.es/novedades/que-es-el-delivery-food/>

Anexos

Repositorio (carpeta “Etapa 2”): <https://github.com/burnoz/situacion-problema>

Nota: se toma la ruta del archivo orders.txt como argumento desde la consola