

Week4 - Lab Practice : Queue

Group 6 :

- Chea Kuyhong (IDTB110172)
- Chhuong Sophakvatey (IDTB110187)
- Ean Mana (IDTB110367)
- Ly Sokunnita (IDTB110369)
- Nget Sokunkanha (IDTB110054)

- Overview :

This project implements a cafeteria ordering queue using a Linked List with **Node** objects. New orders are added at the rear (tail) and served from the front (head). This enforces FIFO (first-in, first-out) so the first order added is the first served.

- Add orders
- Serve the next order
- Display all waiting orders
- Exit.

We chose to use Linked List (head/tail) for this project because it fits a real cafeteria queue because customers continuously join and leave. The number of orders is unpredictable, so a dynamic structure is needed. Linked lists allow fast insertions at the end (enqueue) and quick removals from the front (dequeue) without shifting elements or defining a fixed size. Each operation is O(1). This makes it more efficient than an array-based queue, which may require compaction or resizing.

Diagram :

Rear



Front -> [Order 1] -> [Order 2] -> [Order 3] -> null

Front points to the oldest order (to be served next). Rear points to the newest order (just added).

- Methods :

Class : Node

- `Node(int id, string name, double price, int amount, int d, int m, int y)` – Creates a new order node with item details, date, and total price.
- `void display_Date()` – Prints the order date in DD/MM/YY format.

Class : Queue

- `void Enqueue(...)` : Add a new order at the rear of the queue. ($O(1)$)
- `void Dequeue()` : Removes the order at the front; prints a message if empty. ($O(1)$)
- `void display_Queue()` : Displays all orders in FIFO order. ($O(n)$)

Data Members :

- `Node* Front` – Points to the first order (head).
- `Node* Rear` – Points to the last order (tail).
- `int length` – Tracks number of active orders.

Behavior:

- On empty: `Dequeue()` and `display_Queue()` print “The queue is empty!”.
- On full: Not applicable (linked list expands dynamically).

Invariants:

`Front == nullptr <=> Queue empty;` `Rear->next == nullptr;` `length equals total nodes.`

- Edge Cases Handled

Case	Description	Location
Empty Dequeue	Checks if <code>Front == nullptr</code> before deleting	<code>Queue::Dequeue()</code>
Empty Display	Checks if <code>Front == nullptr</code> and prints message	<code>Queue::display_Queue()</code>
Removing Last Node	If <code>Front == Rear</code> , both set to <code>nullptr</code>	<code>Queue::Dequeue()</code>
Memory Safety	Nodes are new on enqueue and delete on dequeue	<code>queue.hpp</code>
User Input	Uses <code>cin</code> and <code>getline</code> , minimal validation	<code>main.cpp</code>

- Complexity and Space Analysis :
 - Enqueue : $O(1)$ — adds one node at the end.
 - Dequeue : $O(1)$ — removes one node at front.
 - Display Queue : $O(n)$ — traverses all nodes.
 - Space Cost : $O(n)$ — one node per active order.
 - Compaction : Not required (linked list variant grows and shrinks dynamically).

This efficiency makes linked list queues ideal for continuously changing data sizes, like customer lines.

- Output :

Menu :

```
--- Cafeteria Queue Menu ---
1. Add Order
2. Serve Order
3. Display Queue
4. Exit
Enter your choice: █
```

Choice 1 :

```
Enter your choice: 1
Enter item name: Fried rice
Enter item price ($): 3
Enter item amount: 1
Order added to queue.
```

Choice 2 :

```
Enter your choice: 2
Serving Order No.1
Item: Fried rice
Amount: 1
Total Price: 3
Order Date: DD/MM/YY : 12 / 11 / 2025
```

Choice 3 :

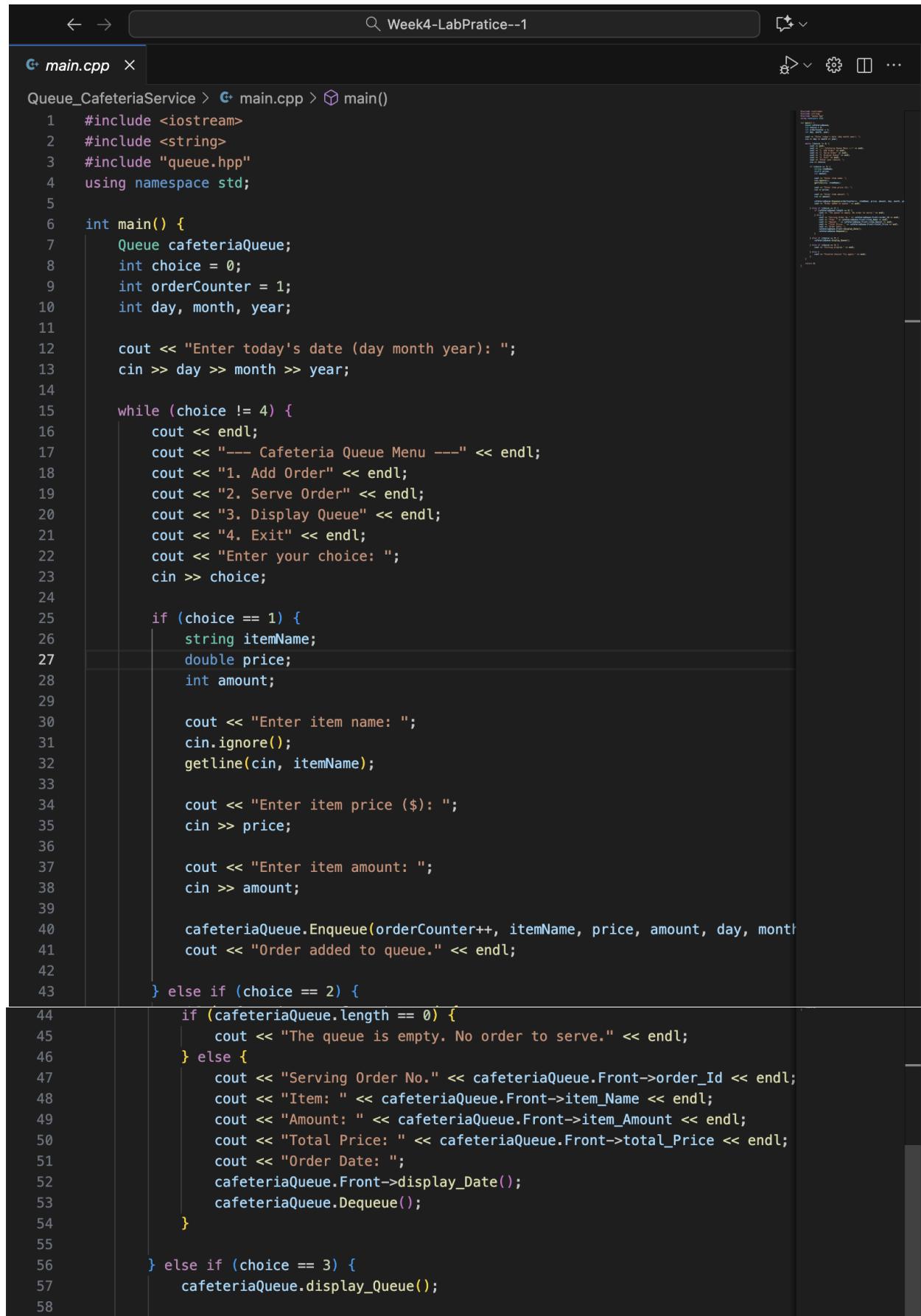
```
Enter your choice: 3
--- Queue ---
Queue length: 2

- Order No.2 -
Item: Steamed buns
Amount: 5
Price per unit: 1$
Total price: 5$
DD/MM/YY : 12 / 11 / 2025
```

```
- Order No.3 -
Item: Dimsum
Amount: 3
Price per unit: 2$
Total price: 6$
DD/MM/YY : 12 / 11 / 2025
```

- Code

main.cpp



The screenshot shows a code editor window with the following details:

- Title Bar:** Week4-LabPratice--1
- File Tab:** C++ main.cpp X
- Code Area:** The main() function of a C++ program. The code uses #include directives for iostream, string, and queue.hpp, and the standard namespace. It initializes a Queue named cafeteriaQueue, sets choice to 0, orderCounter to 1, and day, month, year to 0. It prompts the user to enter today's date. A while loop continues until choice is 4. Inside the loop, it prints a menu with four options: Add Order, Serve Order, Display Queue, and Exit. It then prompts for the user's choice. If choice is 1, it asks for item name, price, and amount, then adds the item to the queue with orderCounter++. If choice is 2, it checks if the queue is empty. If not, it serves the first order by displaying its details (order ID, item name, amount, total price, and date) and then dequeuing it. If choice is 3, it displays the current queue.
- Right Panel:** Shows a sidebar with various icons and a preview area showing the output of the program.

```
Queue_CafeteriaService > C++ main.cpp > main()
```

```
1 #include <iostream>
2 #include <string>
3 #include "queue.hpp"
4 using namespace std;
5
6 int main() {
7     Queue cafeteriaQueue;
8     int choice = 0;
9     int orderCounter = 1;
10    int day, month, year;
11
12    cout << "Enter today's date (day month year): ";
13    cin >> day >> month >> year;
14
15    while (choice != 4) {
16        cout << endl;
17        cout << "---- Cafeteria Queue Menu ----" << endl;
18        cout << "1. Add Order" << endl;
19        cout << "2. Serve Order" << endl;
20        cout << "3. Display Queue" << endl;
21        cout << "4. Exit" << endl;
22        cout << "Enter your choice: ";
23        cin >> choice;
24
25        if (choice == 1) {
26            string itemName;
27            double price;
28            int amount;
29
30            cout << "Enter item name: ";
31            cin.ignore();
32            getline(cin, itemName);
33
34            cout << "Enter item price ($): ";
35            cin >> price;
36
37            cout << "Enter item amount: ";
38            cin >> amount;
39
40            cafeteriaQueue.Enqueue(orderCounter++, itemName, price, amount, day, month);
41            cout << "Order added to queue." << endl;
42
43        } else if (choice == 2) {
44            if (cafeteriaQueue.length == 0) {
45                cout << "The queue is empty. No order to serve." << endl;
46            } else {
47                cout << "Serving Order No." << cafeteriaQueue.Front->order_Id << endl;
48                cout << "Item: " << cafeteriaQueue.Front->item_Name << endl;
49                cout << "Amount: " << cafeteriaQueue.Front->item_Amount << endl;
50                cout << "Total Price: " << cafeteriaQueue.Front->total_Price << endl;
51                cout << "Order Date: ";
52                cafeteriaQueue.Front->display_Date();
53                cafeteriaQueue.Dequeue();
54            }
55
56        } else if (choice == 3) {
57            cafeteriaQueue.display_Queue();
58        }
    }
```

```
59         } else if (choice == 4) {
60             cout << "Exiting program." << endl;
61
62         } else {
63             cout << "Invalid choice! Try again." << endl;
64         }
65     }
66
67     return 0;
68 }
69
```

queue.hpp

Week4-LabPratice--1

queue.hpp

Queue_CafeteriaService > queue.hpp > Node > Node(int, string, double, int, int, int)

```
1 #include <iostream>
2 using namespace std;
3
4 class Node {
5     public:
6         int order_Id;
7         int day, month, year;
8         string item_Name;
9         double item_Price;
10        int item_Amount;
11        int total_Price;
12        Node* next = nullptr;
13
14    public:
15        Node(int id, string name, double price, int amount, int d, int m, int y){
16            order_Id = id;
17            day = d;
18            month = m;
19            year = y;
20            item_Name = name;
21            item_Price = price;
22            item_Amount = amount;
23            total_Price = item_Price * item_Amount;
24        }
25
26    void display_Date(){
27        cout << "DD/MM/YY : " << day << " / " << month << " / " << year << endl;
28    }
29
30};
31
32 class Queue {
33     public:
34         Node* Front = nullptr;
35         Node* Rear = nullptr;
36         int length = 0;
37
38     public:
39        void Enqueue(int id, string name, double price, int amount, int d, int m, int y){
40            Node* nNode = new Node(id, name, price, amount, d, m, y);
41            if (Front == nullptr){
42                Front = nNode;
43                Rear = nNode;
```

```
44     } else {
45         Rear->next = nNode;
46         Rear = nNode;
47     }
48     length++;
49 }
50
51 void Dequeue(){
52     if (Front == nullptr){
53         cout << "The queue is empty!" << endl;
54         return;
55     }
56     Node* Temp = Front;
57     if (Front == Rear){
58         Front = nullptr;
59         Rear = nullptr;
60         delete Temp;
61     } else {
62         Front = Front->next;
63         delete Temp;
64     }
65     length--;
66 }
67
68 void display_Queue(){
69     Node* cur = Front;
70     if (Front == nullptr){
71         cout << "The queue is empty!" << endl;
72         return;
73     }
74     cout << "---- Queue ----" << endl;
75     cout << "Queue length: " << length << endl;
76     while (cur != nullptr){
77         cout << endl << "- Order No." << cur->order_Id << " -" << endl;
78         cout << "Item: " << cur->item_Name << endl;
79         cout << "Amount: " << cur->item_Amount << endl;
80         cout << "Price per unit: " << cur->item_Price << "$" << endl;
81         cout << "Total price: " << cur->total_Price << "$" << endl;
82         cur->display_Date();
83         cur = cur->next;
84     }
85 }
86 };
```