Assignment No. 6: Queue

PROBLEM STATEMENT:
Coffee Shop Line (Simple Queue):
Arrival: Customers arrive at the coffee shop and stand in line. Order Processing: The first customer in line gets their order taken, and the barista starts making the coffee.Serving: Once the first customer is served, they leave the queue, and the next customer in line moves forward to be served. Write a program to implement a simple queue

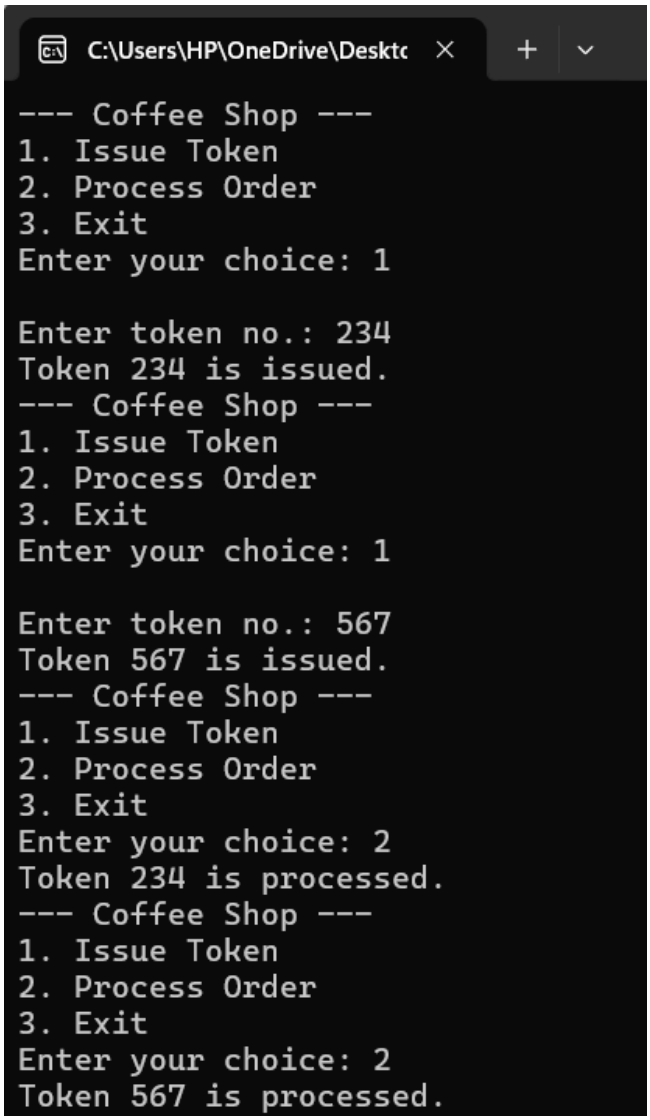
PROGRAM:

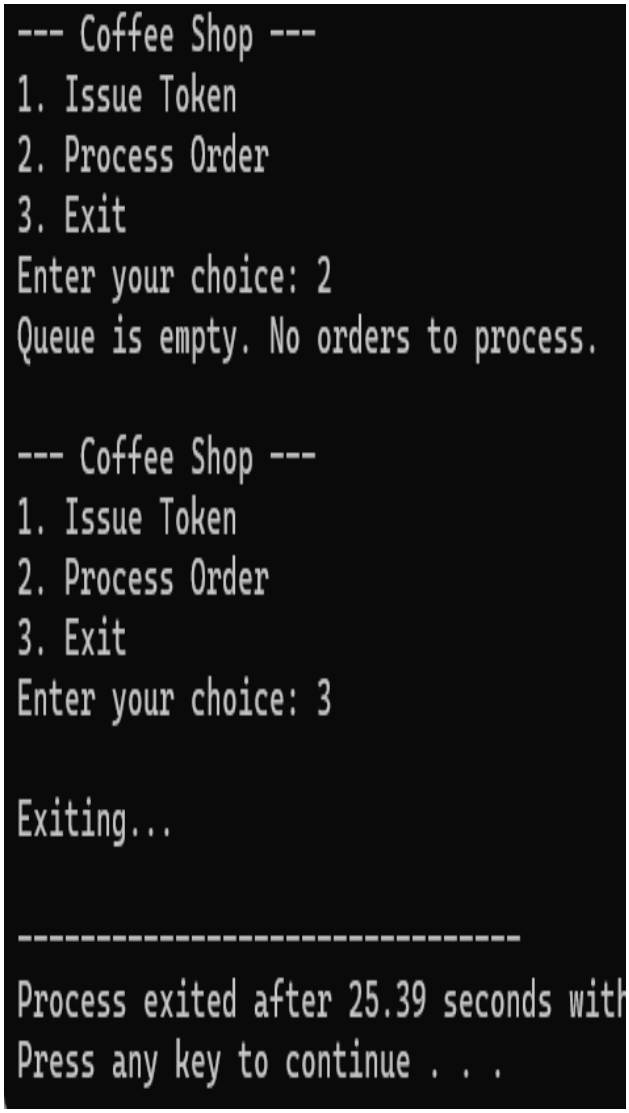
```
#include <iostream>
using namespace std;
class shop {
    int size = 5;
    int token_shop[5];
    int r = -1; // rear
    int f = 0;  // front
public:
    void Enqueue(int t);
    int Dequeue();
    int isEmpty();
    int isFull();
};

int shop::isEmpty() {
    return (f > r);
}
int shop::isFull() {
    return (r == size - 1);
}
void shop::Enqueue(int t) {
    if (isFull()) {
        cout << "Queue is full. Cannot issue token."<<endl;
    } else {
        r++;
        token_shop[r] = t;
        cout << "Token " << t << " is issued.";
    }
}
```

```
int shop::Dequeue() {
    if (isEmpty()) {
        cout << "Queue is empty. No orders to process."<<endl;
        return -1;
    } else {
        int x = token_shop[f];
        f++;
        cout << "Token " << x << " is processed.";
        return x;
    }
}

int main() {
    shop s;
    int t; // token no.
    int choice;

    do {
        cout << "--- Coffee Shop ---";
        cout << "\n1. Issue Token";
        cout << "\n2. Process Order";
        cout << "\n3. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "\nEnter token no.: ";
            cin >> t;
            s.Enqueue(t);
            break;

        case 2:
            s.Dequeue();
            break;

        case 3:
            cout << "\nExiting...";
            break;
```

```
        default:
            cout << "\nInvalid choice. Please try again.";
        }
        cout<<"\n";
    } while (choice != 3);
    return 0;
}
```

OUTPUT:





Printer Spooler (Circular Queue):
In a multi-user environment, printers often use a circular queue to manage print jobs. Each print job is added to the queue, and the printer processes them in the order they arrive. Once a print job is completed, it moves out of the queue, and the next job is processed, efficiently managing the flow of print tasks. Implement the Printer Spooler system using a circular queue without using built-in queues.

PROGRAM:

```
#include <iostream>
#include <string>
using namespace std;

class PrinterSpooler {
   string queue[100];
   int front, rear;
   int size;

public:
   PrinterSpooler(int s);          // constructor
   int isFull();                   // check if queue is full
   int isEmpty();                  // check if queue is empty
   void addJob(string jobName);    // add job to queue
   void processJob();              // process job
   void displayQueue();            // show all jobs
};

// Constructor
PrinterSpooler::PrinterSpooler(int s) {
   size = s;
   front = -1;
   rear = -1;
}

// Check if queue is full
int PrinterSpooler::isFull() {
   if (front == (rear + 1) % size)
      return 1;
   else
      return 0;
}
```

```
// Check if queue is empty
int PrinterSpooler::isEmpty() {
    if (front == -1)
        return 1;
    else
        return 0;
}

// Add print job
void PrinterSpooler::addJob(string jobName) {
    if (isFull()) {
        cout << "Printer Queue is Full! Cannot add new job.\n";
        return;
    }

    if (isEmpty())
        front = 0;

    rear = (rear + 1) % size;
    queue[rear] = jobName;
    cout << "Job \"" << jobName << "\" added to the printer queue.\n";
}

// Process print job
void PrinterSpooler::processJob() {
    if (isEmpty()) {
        cout << "No jobs to process. Queue is empty!\n";
        return;
    }

    cout << "Processing Job: " << queue[front] << endl;

    if (front == rear) {
        front = -1;
        rear = -1;
    } else {
        front = (front + 1) % size;
    }
}
```

```
// Display queue
void PrinterSpooler::displayQueue() {
    if (isEmpty()) {
        cout << "Printer queue is empty.\n";
        return;
    }

    cout << "\nCurrent Print Queue:\n";
    int i = front;
    while (1) {
        cout << "   - " << queue[i] << endl;
        if (i == rear)
            break;
        i = (i + 1) % size;
    }
}

// Main Function
int main() {
    int s, choice;
    string jobName;

    cout << "===== PRINTER SPOOLER SYSTEM (Circular Queue) =====\n";
    cout << "Enter the size of the printer queue (max 100): ";
    cin >> s;

    if (s > 100 || s <= 0) {
        cout << " Invalid size! Setting default size = 5.\n";
        s = 5;
    }

    PrinterSpooler spooler(s);

    do {
        cout << "\nMenu:\n";
        cout << "1. Add Print Job\n";
        cout << "2. Process Print Job\n";
        cout << "3. Show Print Queue\n";
        cout << "4. Exit\n";
```

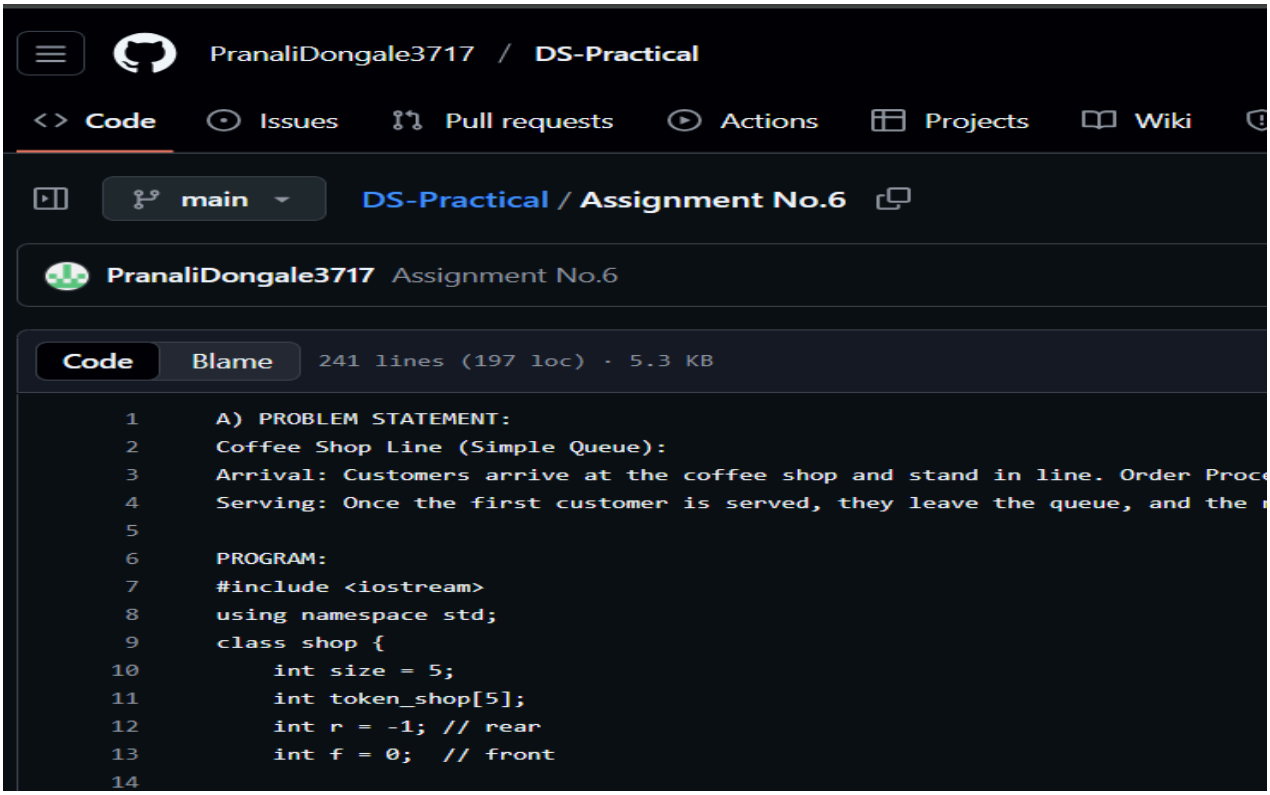
```
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter job name: ";
            cin >> jobName;
            spooler.addJob(jobName);
            break;
        case 2:
            spooler.processJob();
            break;
        case 3:
            spooler.displayQueue();
            break;
        case 4:
            cout << "Exiting Printer Spooler System...\n";
            break;
        default:
            cout << "Invalid choice. Try again.\n";
        }
    } while (choice != 4);
    return 0;
}
```

OUTPUT:

```
===== PRINTER SPOOLER SYSTEM (Circular Queue) =====
Enter the size of the printer queue : 2

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 1
Enter job name: Doc1
Job "Doc1" added to the printer queue.

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 1
Enter job name: Report
Job "Report" added to the printer queue.

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 1
Enter job name: Image
Printer Queue is Full! Cannot add new job.

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 2
Processing Job: Doc1
```

```
Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 1
Enter job name: Doc2
Job "Doc2" added to the printer queue.

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 2
Processing Job: Report

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 3

Current Print Queue:
   - Doc2

Menu:
1. Add Print Job
2. Process Print Job
3. Show Print Queue
4. Exit
Enter your choice: 4
Exiting Printer Spooler System...

--------------------------------
Process exited after 92.25 seconds with re
Press any key to continue . . .
```

GitHub:


PranaliDongale3717 / DS-Practical
Code
Issues
Pull requests
Actions
Projects
Wiki
main
DS-Practical / Assignment No.6
PranaliDongale3717 Assignment No.6
Code
Blame
241 lines (197 loc) · 5.3 KB
A) PROBLEM STATEMENT:
Coffee Shop Line (Simple Queue):
Arrival: Customers arrive at the coffee shop and stand in line. Order Proce
Serving: Once the first customer is served, they leave the queue, and the
PROGRAM:
#include <iostream>
using namespace std;
class shop {
int size = 5;
int token_shop[5];
int r = -1; // rear
int f = 0; // front