

Activity No. <n>	
<Hands-on Activity 5.1 Queues>	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 09/11/25
Section: CPE 010-CPE21S4	Date Submitted: 09/11/25
Name(s): KERWIN JAN B. CATUNGAL	Instructor: Jimlord Quejado

6. Output

5.1 Queues using C++ STL

```
1 #include <iostream>
2 #include <queue>
3 #include <string>
4
5 int main() {
6     std::queue<std::string> q;
7     std::string characters[] = {
8         "Kim Dokja",
9         "Yoo Joonghyuk",
10        "Uriel",
11        "Sun Wukong",
12        "Persephone",
13        "Hades"
14    };
15
16    for (auto& name : characters) {
17        std::cout << "Enqueue: " << name << std::endl;
18        q.push(name);
19    }
20
21    std::cout << "Queue after enqueues: ";
22    std::queue<std::string> temp = q;
23    while (!temp.empty()) {
24        std::cout << temp.front() << " ";
25        temp.pop();
26    }
27    std::cout << "\n";
28
29    while (!q.empty()) {
30        std::cout << "Dequeue: " << q.front() << std::endl;
31        q.pop();
32    }
33 }
```

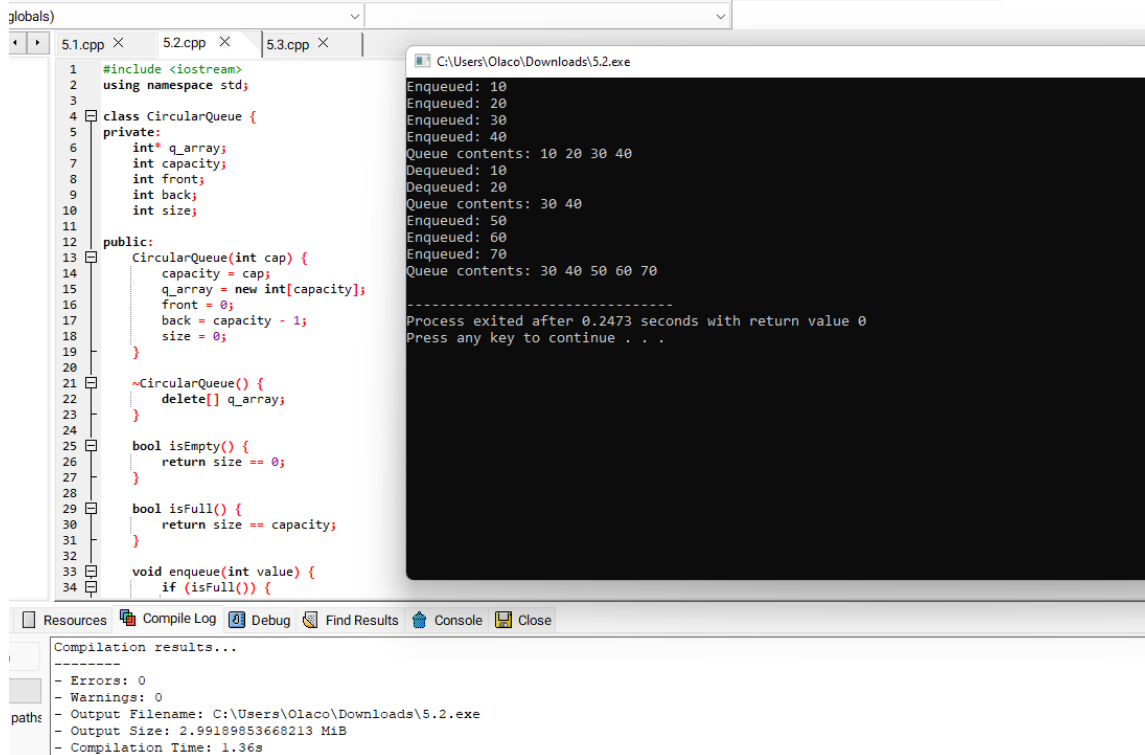
```
Enqueue: Kim Dokja
Enqueue: Yoo Joonghyuk
Enqueue: Uriel
Enqueue: Sun Wukong
Enqueue: Persephone
Enqueue: Hades
Queue after enqueues: Kim Dokja Yoo Joonghyuk Uriel Sun Wukong Persephone Hades
Dequeue: Kim Dokja
Dequeue: Yoo Joonghyuk
Dequeue: Uriel
Dequeue: Sun Wukong
Dequeue: Persephone
Dequeue: Hades
-----
Process exited after 0.242 seconds with return value 0
Press any key to continue . . .
```

Compilation results...

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Olaco\Downloads\5.1.exe
- Output Size: 3.06792736053467 MiB
- Compilation Time: 1.80s
```

Here it shows how a queue works using some character names. First each name is added to the queue and enqueue. Then it prints out the current queue by copying it into a temporary one.

5-2. Queues using Linked List Implementation



```
1 #include <iostream>
2 using namespace std;
3
4 class CircularQueue {
5 private:
6     int* q_array;
7     int capacity;
8     int front;
9     int back;
10    int size;
11
12 public:
13     CircularQueue(int cap) {
14         capacity = cap;
15         q_array = new int[capacity];
16         front = 0;
17         back = capacity - 1;
18         size = 0;
19     }
20
21     ~CircularQueue() {
22         delete[] q_array;
23     }
24
25     bool isEmpty() {
26         return size == 0;
27     }
28
29     bool isFull() {
30         return size == capacity;
31     }
32
33     void enqueue(int value) {
34         if (isFull()) {
35             // ... (code for handling full queue) ...
36         }
37     }
38
39     // ... (code for dequeue) ...
40
41     // ... (code for display) ...
42 }
43
```

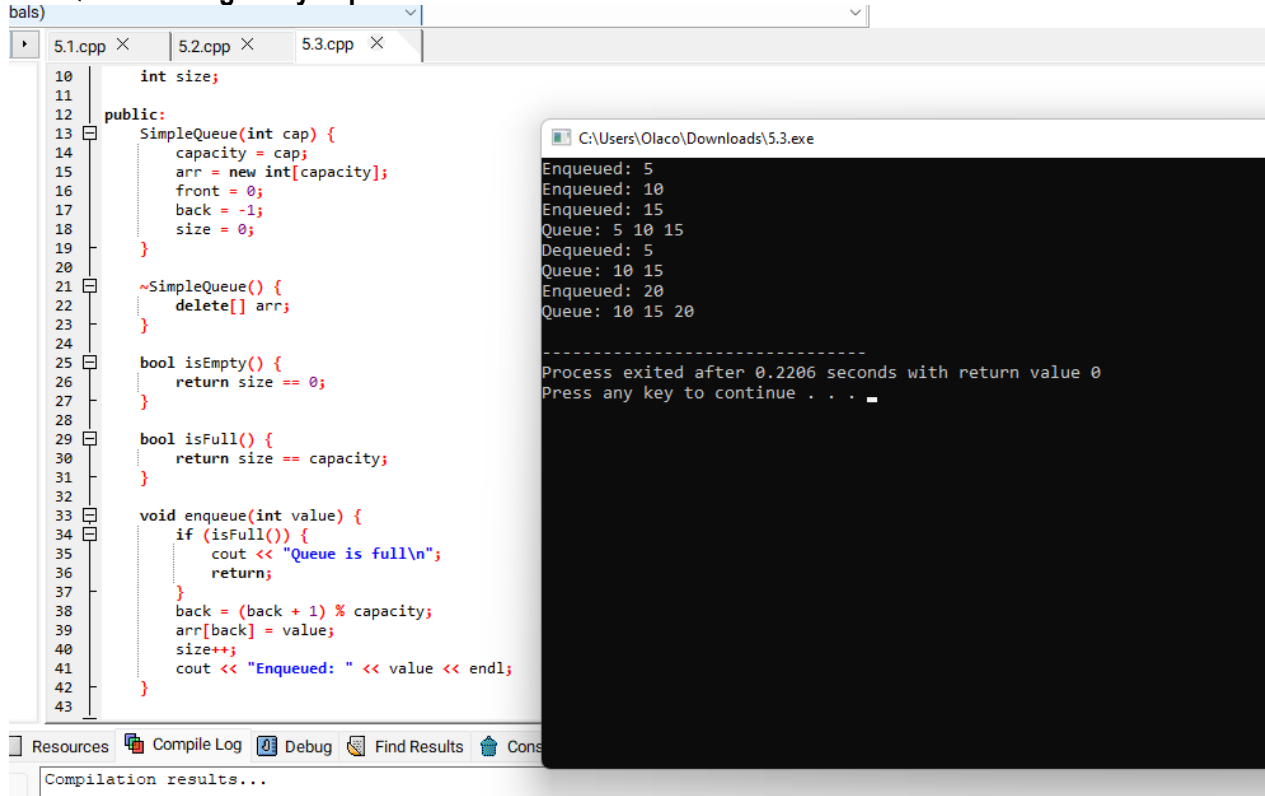
```
Enqueued: 10
Enqueued: 20
Enqueued: 30
Enqueued: 40
Queue contents: 10 20 30 40
Dequeued: 10
Dequeued: 20
Queue contents: 30 40
Enqueued: 50
Enqueued: 60
Enqueued: 70
Queue contents: 30 40 50 60 70
-----
Process exited after 0.2473 seconds with return value 0
Press any key to continue . . .
```

Compilation results...

Errors: 0
Warnings: 0
Output Filename: C:\Users\Olaco\Downloads\5.2.exe
Output Size: 2.99189853668213 MiB
Compilation Time: 1.36s

Here It can also store numbers and follows the first-in, first-out order and when you add data it goes to the back enqueue and when you remove it comes from the front which is the dequeue.

5-3. Queues using Array Implementation



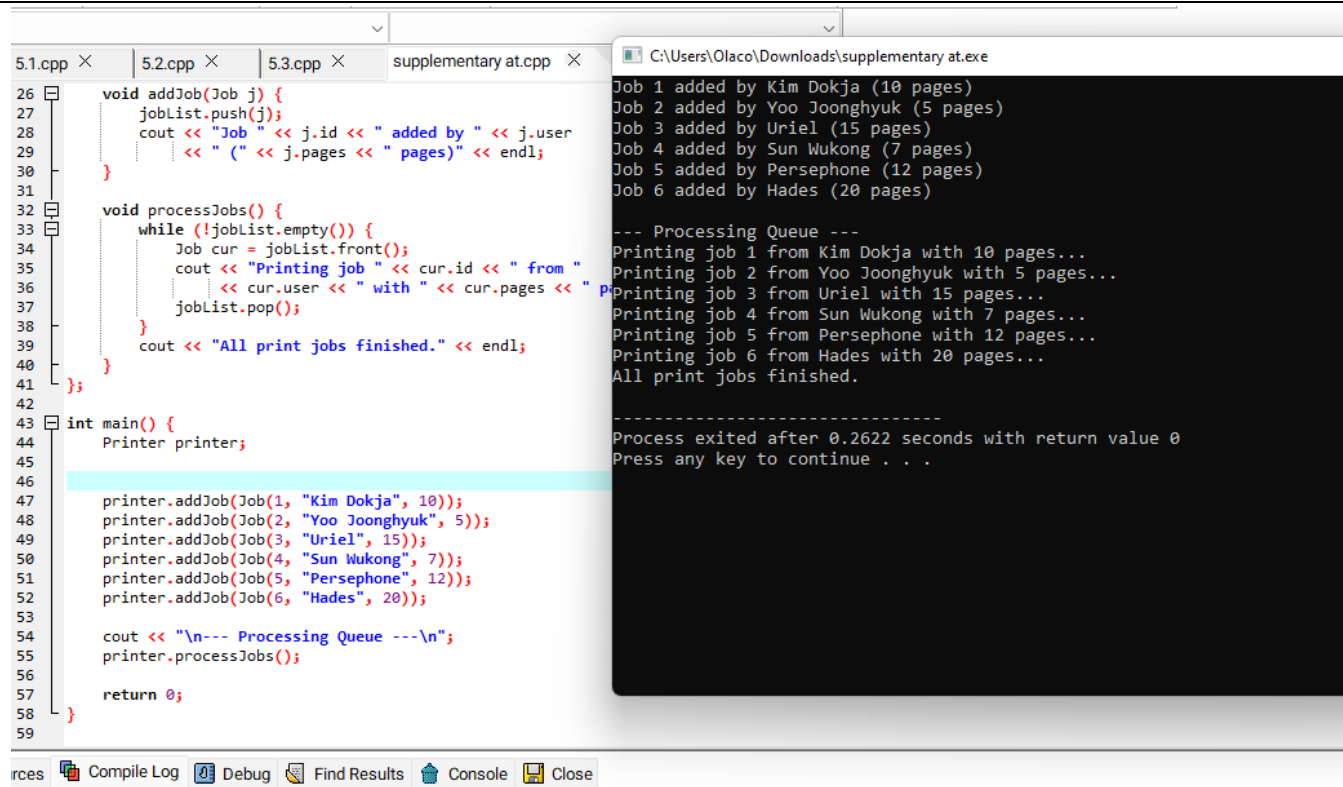
```
10 int size;
11
12 public:
13     SimpleQueue(int cap) {
14         capacity = cap;
15         arr = new int[capacity];
16         front = 0;
17         back = -1;
18         size = 0;
19     }
20
21     ~SimpleQueue() {
22         delete[] arr;
23     }
24
25     bool isEmpty() {
26         return size == 0;
27     }
28
29     bool isFull() {
30         return size == capacity;
31     }
32
33     void enqueue(int value) {
34         if (isFull()) {
35             cout << "Queue is full\n";
36             return;
37         }
38         back = (back + 1) % capacity;
39         arr[back] = value;
40         size++;
41         cout << "Enqueued: " << value << endl;
42     }
43
44     // ... (code for dequeue) ...
45
46     // ... (code for display) ...
47 }
48
```

```
Enqueued: 5
Enqueued: 10
Enqueued: 15
Queue: 5 10 15
Dequeued: 5
Queue: 10 15
Enqueued: 20
Queue: 10 15 20
-----
Process exited after 0.2206 seconds with return value 0
Press any key to continue . . .
```

Compilation results...

Here you can add numbers with enqueue and remove them with dequeue like always following first-in, first-out order It also checks if the queue is empty or full before doing anything and it has also the display function that shows the current contents

7. Supplementary Activity



The screenshot shows a C++ IDE with a file named `supplementary at.cpp` open. The code implements a queue using a `jobList` and a `printer` class. The `addJob` function adds jobs to the queue, and the `processJobs` function processes them in order. The `main` function creates a `printer` object, adds six jobs, and then calls `processJobs`. The output window shows the execution results, including the jobs added and the order in which they are processed.

```
26 void addJob(Job j) {
27     jobList.push(j);
28     cout << "Job " << j.id << " added by " << j.user
29         << " (" << j.pages << " pages)" << endl;
30 }
31
32 void processJobs() {
33     while (!jobList.empty()) {
34         Job cur = jobList.front();
35         cout << "Printing job " << cur.id << " from "
36             << cur.user << " with " << cur.pages << " pages" << endl;
37         jobList.pop();
38     }
39     cout << "All print jobs finished." << endl;
40 }
41 };
42
43 int main() {
44     Printer printer;
45
46     printer.addJob(Job(1, "Kim Dokja", 10));
47     printer.addJob(Job(2, "Yoo Joonghyuk", 5));
48     printer.addJob(Job(3, "Uriel", 15));
49     printer.addJob(Job(4, "Sun Wukong", 7));
50     printer.addJob(Job(5, "Persephone", 12));
51     printer.addJob(Job(6, "Hades", 20));
52
53     cout << "\n--- Processing Queue ---\n";
54     printer.processJobs();
55
56     return 0;
57 }
58
59
```

Output:

```
Job 1 added by Kim Dokja (10 pages)
Job 2 added by Yoo Joonghyuk (5 pages)
Job 3 added by Uriel (15 pages)
Job 4 added by Sun Wukong (7 pages)
Job 5 added by Persephone (12 pages)
Job 6 added by Hades (20 pages)

--- Processing Queue ---
Printing job 1 from Kim Dokja with 10 pages...
Printing job 2 from Yoo Joonghyuk with 5 pages...
Printing job 3 from Uriel with 15 pages...
Printing job 4 from Sun Wukong with 7 pages...
Printing job 5 from Persephone with 12 pages...
Printing job 6 from Hades with 20 pages...
All print jobs finished.

Process exited after 0.2622 seconds with return value 0
Press any key to continue . . .
```

So here it works like a printer line where the first job that enters is also the first one to come out. I just used manhwa character names as examples for the print jobs so it looks more fun. The program keeps going until everything in the queue is printed.

8. Conclusion

I came to understand the queue concept in arrays through practicing enqueue and dequeue using a circular layout. This helped me to better understand how to perform simple tasks of adding, removing, or displaying values. The main learning point. I performed an activity and as a result it became clearer for me, actually not yet but overall I think I did well. However, there is still room for improvement as far as explanation of my code and making it clear are concerned.

9. Assessment Rubric