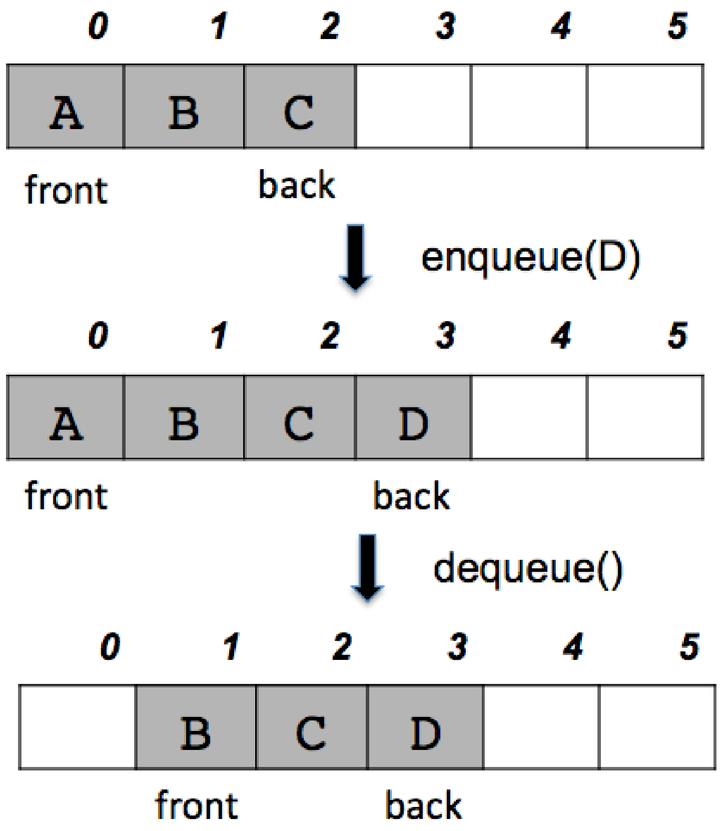
ARRAY-BASED QUEUE

Lab 3

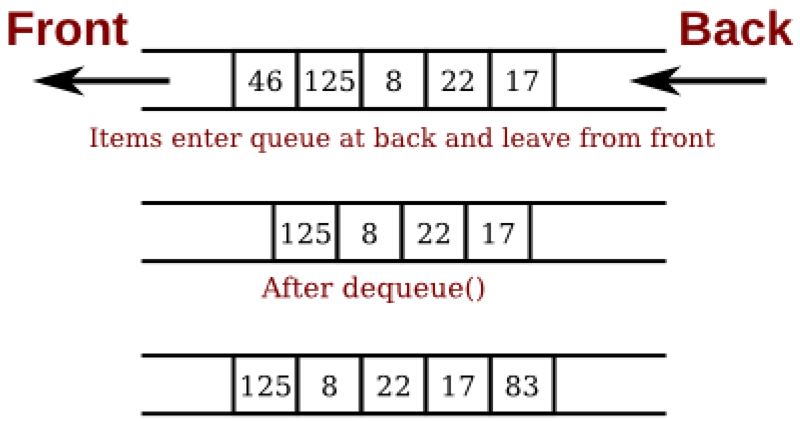
AGENDA

1. **General idea of templates & queue.**
2. **Array-based queue class, Constructors and Destructor using template.**
3. **Enqueue, Dequeue and Front functions using template .**
4. **TASK 1: Queue Application.**

GENERAL QUEUE CONCEPT



In a queue, all operations take place at one end of the queue oe the other. The "enqueue" operation adds an item to the "back" of the queue. The "dequeue" operation removes the item at the "front" and returns it.



Items enter queue at back and leave from front

After dequeue()

After enqueue(83)

ARRAY-BASED QUEUE, CLASS “HEADER FILE CONSTRUCTOR & DESTRUCTOR ”

<u>Data Members:</u>

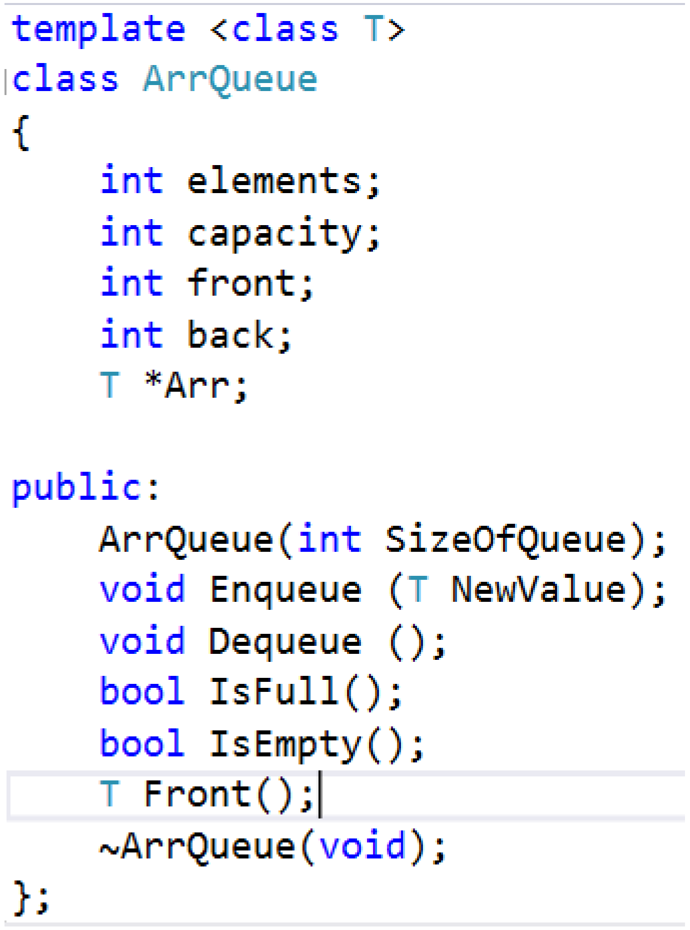
1. Pointer to queue array.
2. Number of elements in queue.
3. Total size of queue array.
4. Front index.
5. Back index

<u>Methods:</u>

1. Constructor & Destructor.
2. Enqueue.
3. Dequeue.
4. Front.
5. Full.
6. Empty.

ARRAY-BASED QUEUE CLASS "HEADER FILE, CONSTRUCTOR & DESTRUCTOR"

1. ArrayQueue class (Header File)



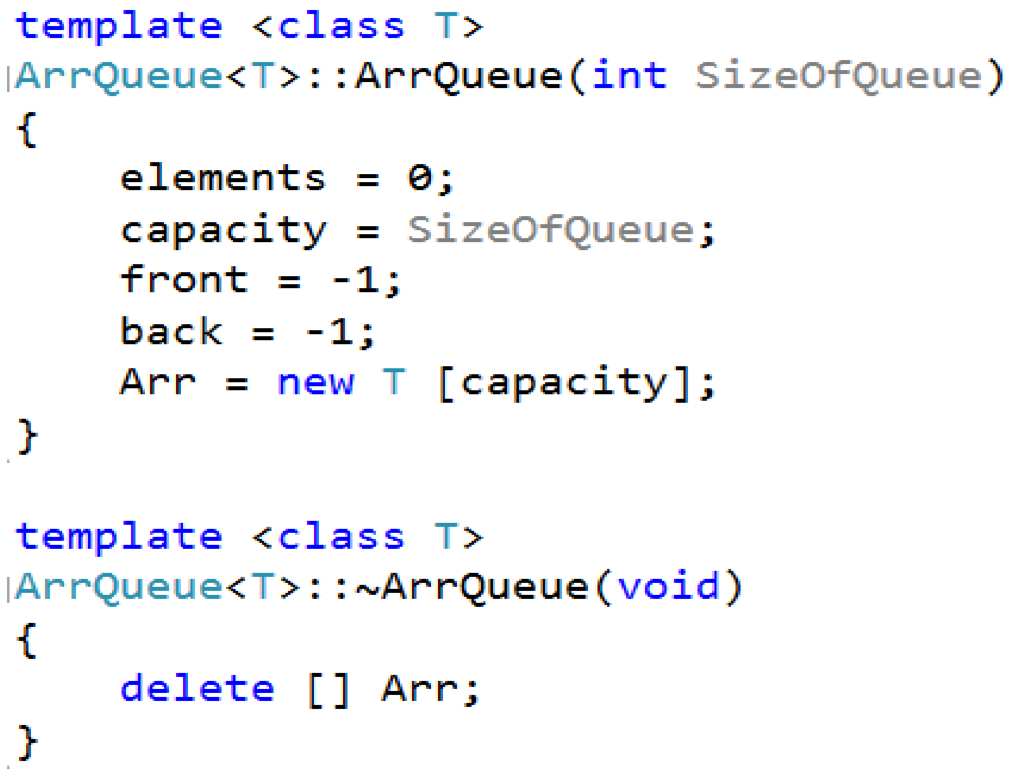
```cpp
template <class T>
class ArrQueue
{
	int elements;
	int capacity;
	int front;
	int back;
	T *Arr;

public:
	ArrQueue(int SizeOfQueue);
	void Enqueue (T NewValue);
	void Dequeue ();
	bool IsFull();
	bool IsEmpty();
	T Front();
	~ArrQueue(void);
};
```

ARRAY-BASED QUEUE CLASS “HEADER FILE, CONSTRUCTOR & DESTRUCTOR”

2. ArrayQueue class (.CppFile)

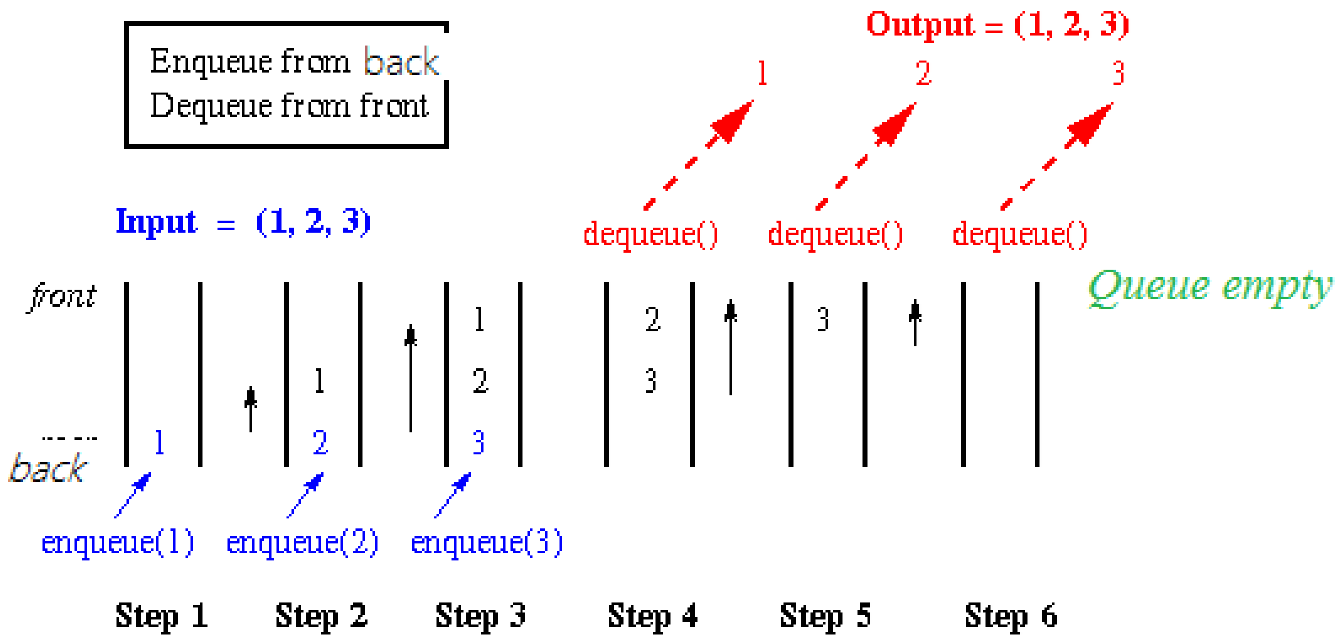
Constructor & Destructor



```
template <class T>
ArrQueue<T>::ArrQueue(int SizeOfQueue)
{
    elements = 0;
    capacity = SizeOfQueue;
    front = -1;
    back = -1;
    Arr = new T [capacity];
}

template <class T>
ArrQueue<T>::~ArrQueue(void)
{
    delete [] Arr;
}
```

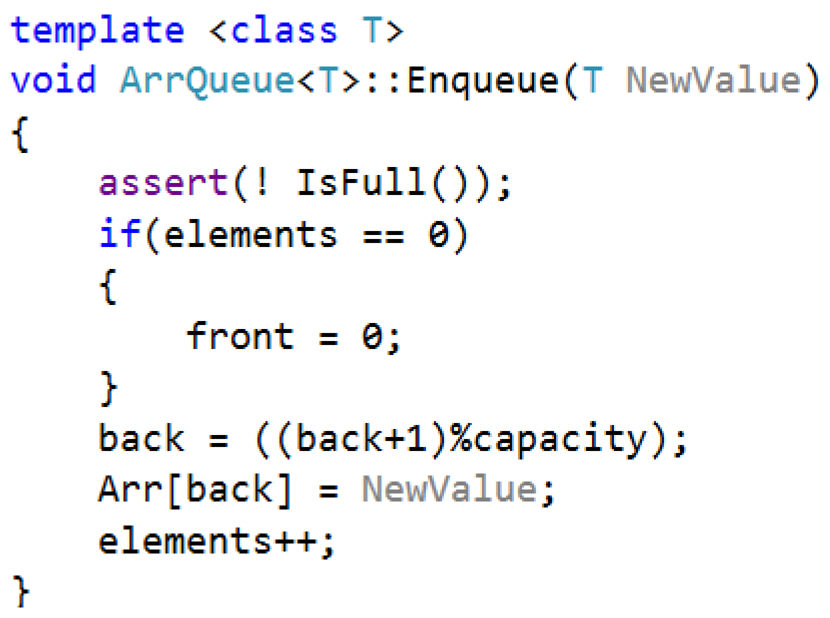
IMPLEMENT ENQUEUE, DEQUEUE & FRONT



IMPLEMENT ENQUEUE, DEQUEUE & FRONT

ArrayQueue class (.CppFile)

Enqueue

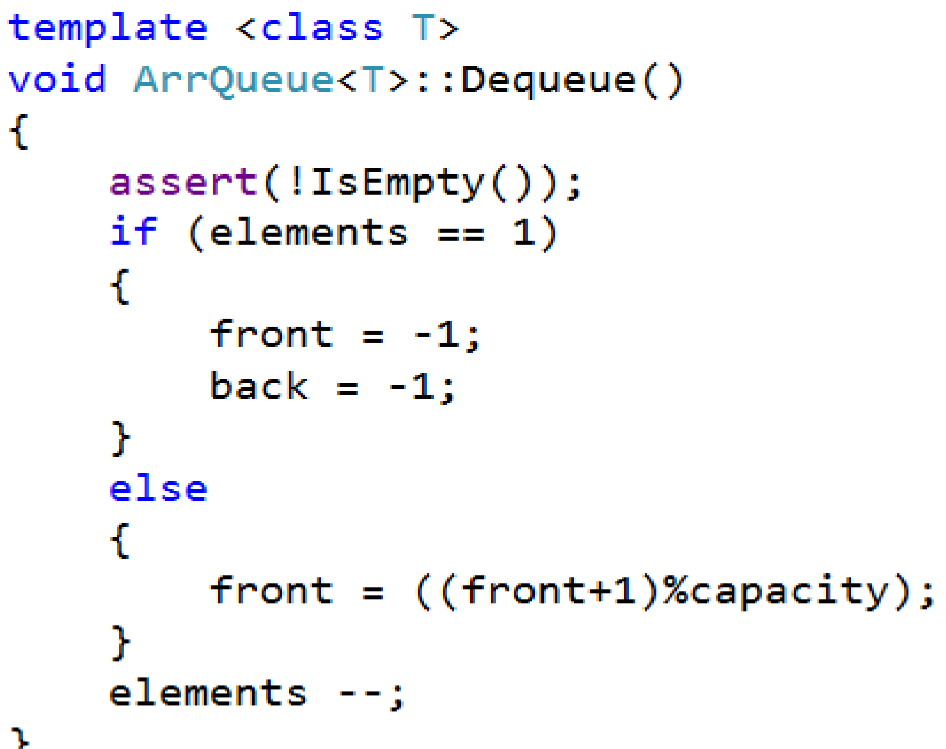


```
template <class T>
void ArrQueue<T>::Enqueue(T NewValue)
{
    assert(! IsFull());
    if(elements == 0)
    {
        front = 0;
    }
    back = ((back+1)%capacity);
    Arr[back] = NewValue;
    elements++;
}
```

IMPLEMENT ENQUEUE, DEQUEUE & FRONT

ArrayQueue class (.CppFile)

Dequeue

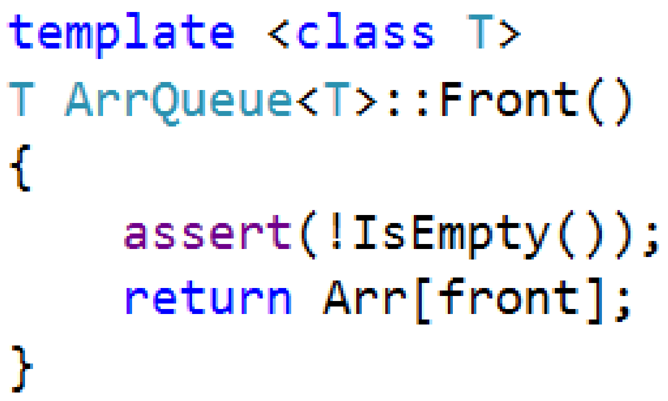


```
template <class T>
void ArrQueue<T>::Dequeue()
{
    assert(!IsEmpty());
    if (elements == 1)
    {
        front = -1;
        back = -1;
    }
    else
    {
        front = ((front+1)%capacity);
    }
    elements --;
}
```

IMPLEMENT ENQUEUE, DEQUEUE & FRONT

ArrayQueue class (.CppFile)

Front

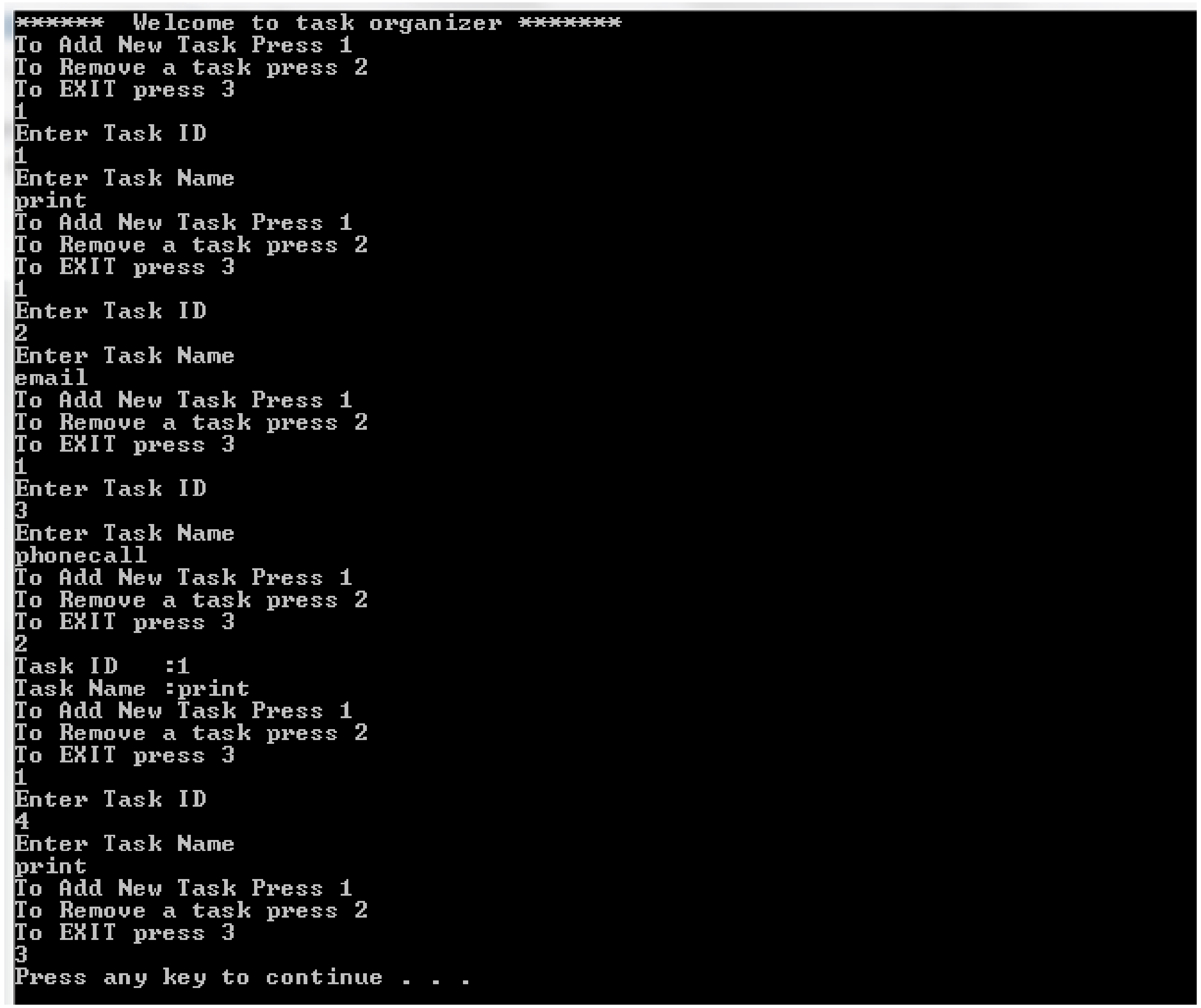


```
template <class T>
T ArrQueue<T>::Front()
{
    assert(!IsEmpty());
    return Arr[front];
}
```

TASK 1: TASK ORGANIZER APPLICATION

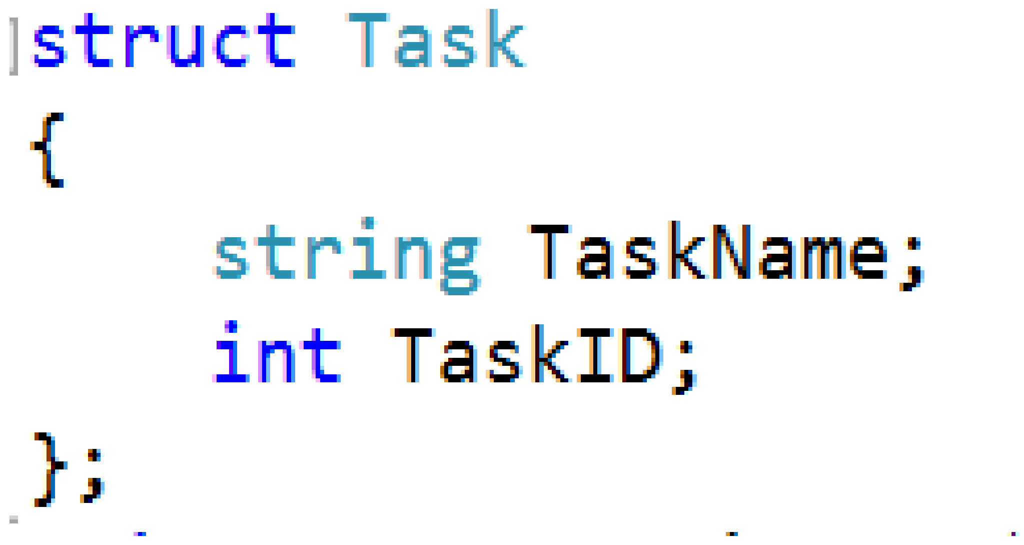
Implement a task organizer application such that each task has a name and ID using first in first out property of the queue.

SAMPLE RUN:



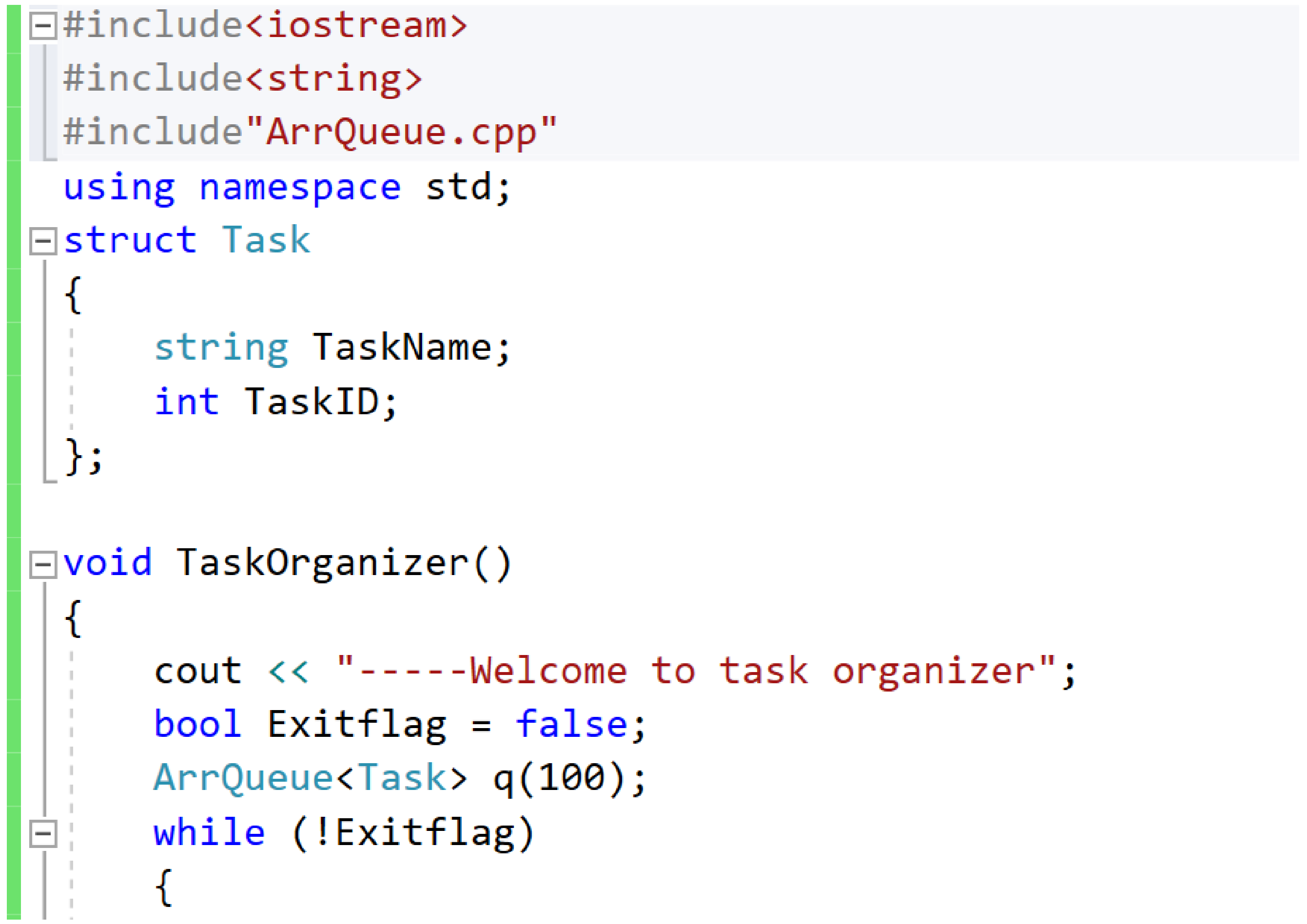
```
*******  Welcome to task organizer ********
To Add New Task Press 1
To Remove a task press 2
To EXIT press 3
1
Enter Task ID
1
Enter Task Name
print
To Add New Task Press 1
To Remove a task press 2
To EXIT press 3
1
Enter Task ID
2
Enter Task Name
email
To Add New Task Press 1
To Remove a task press 2
To EXIT press 3
1
Enter Task ID
3
Enter Task Name
phonecall
To Add New Task Press 1
To Remove a task press 2
To EXIT press 3
2
Task ID   :1
Task Name :print
To Add New Task Press 1
To Remove a task press 2
To EXIT press 3
1
Enter Task ID
4
Enter Task Name
print
To Add New Task Press 1
To Remove a task press 2
To EXIT press 3
3
Press any key to continue . . .
```

TASK 1: TASK ORGANIZER APPLICATION *"SOLUTION"*



```
struct Task
{
      string TaskName;
      int TaskID;
};
```

TASK 1: TASK ORGANIZER APPLICATION *"SOLUTION"*

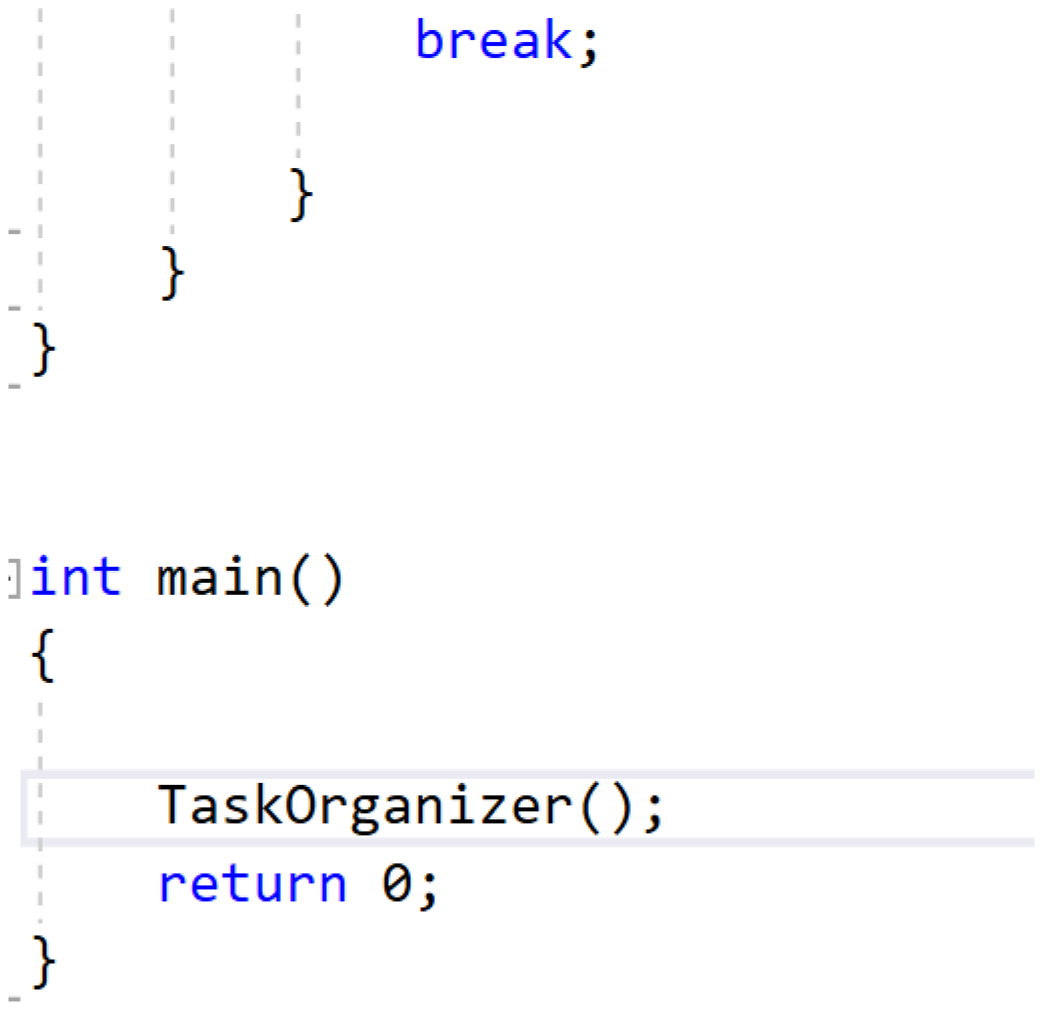


```cpp
#include<iostream>
#include<string>
#include"ArrQueue.cpp"
using namespace std;
struct Task
{
    string TaskName;
    int TaskID;
};

void TaskOrganizer()
{
    cout << "-----Welcome to task organizer";
    bool Exitflag = false;
    ArrQueue<Task> q(100);
    while (!Exitflag)
    {
```

```
while (!Exitflag)
{
    int choice;
    cout << "To Add New Task Press 1";
    cout << "To Remove Task Press 2";
    cout << "To Exit Press 3";
    cin >> choice;
    switch (choice)
    {
    case 1:
    {
        Task t1;
        cout << "Enter Task ID:";
        cin >> t1.TaskID;
        cout << "Enter Task Name:";
        cin >> t1.TaskName;
        q.Enqueue(t1);
        break;
```

```cpp
    break;
}
case 2:
{
    Task t2 = q.Front();
    cout << "Task ID : " << t2.TaskID << endl;
    cout << "Task Name : " << t2.TaskName << endl;
    q.Dequeue();
    break;
}
case 3:
{
    Exitflag = true;
    break;
}
default:
    break;
```



```
            break;

        }
    }
}


int main()
{

    TaskOrganizer();
    return 0;
}
```

thank
you