

Queue Data Structure

Queue is a data structure that maintain "First In First Out" (FIFO) order. And can be viewed as people queueing up to buy a ticket. In programming, queue is usually used as a data structure for BFS (Breadth First Search).

Queue operations

Operations on queue Q are :

1. enqueue - insert item at the back of queue Q
2. dequeue - return (and virtually remove) the front item from queue Q
3. init - initialize queue Q, reset all variables.

There are other operations such as full and empty, but usually we don't use this when doing [Valladolid Online Judge](#) or Programming Contest Problems, since we know the largest size of queue. You can implement those operations by yourself :) Fairly easy!

Implementation in C

```
/* Problem : Queue - Data Structure
 * Author  : Stephanus
 * Lang.   : C
 * Date    : 20 January 2004
 */

#include <stdio.h >

#define QUEUE_SIZE 100

typedef struct {
    int q[QUEUE_SIZE];
    int first,last;
    int count;
} queue;

void init_queue(queue *q)
{
    q->first = 0;
    q->last  = QUEUE_SIZE - 1;
    q->count = 0;
}

void enqueue(queue *q,int x)
{
    q->last = (q->last + 1) % QUEUE_SIZE;
    q->q[ q->last ] = x;
    q->count = q->count + 1;
}

int dequeue(queue *q)
{
    int x = q->q[ q->first ];
    q->first = (q->first + 1) % QUEUE_SIZE;
    q->count = q->count - 1;
}
```

```

        return x;
    }

int main()
{
    queue q;

    init_queue(&q);

    enqueue(&q,1);
    enqueue(&q,2);
    enqueue(&q,3);

    while (q.count)
        printf("%d\n",dequeue(&q));

    return 0;
}

```

Implementation in C++

```

/* Problem : Queue - Data Structure
 * Author   : Stephanus
 * Lang.    : C++
 * Date     : 20 January 2004
 */

#include < iostream >

#define QUEUE_SIZE 100

using namespace std;

class Queue {
    int q[QUEUE_SIZE];
    int first,last;
    int count;

public:
    Queue();
    void enqueue(int x);
    int  dequeue();
    int  getSize();
};

Queue::Queue()
{
    first = 0;
    last  = QUEUE_SIZE - 1;
    count = 0;
}

void Queue::enqueue(int x)
{
    last = (last + 1) % QUEUE_SIZE;

```

```

        q[last] = x;
        count++;
    }

int Queue::dequeue()
{
    int x = q[first];
    first = (first + 1) % QUEUE_SIZE;
    count--;
    return x;
}

int Queue::getSize()
{
    return count;
}

int main()
{
    Queue q;

    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);

    while (q.getSize())
        cout << q.dequeue() << endl;

    return 0;
}

```

Implementation in JAVA

```

/* Problem : Queue - Data Structure
 * Author  : Stephanus
 * Lang.   : JAVA
 * Date    : 20 January 2004
 */

class Queue
{
    final int QUEUE_SIZE = 100;

    private int[] q = new int[QUEUE_SIZE];
    private int first,last;
    private int count;

    Queue()
    {
        first = 0;
        last  = QUEUE_SIZE - 1;
        count = 0;
    }

    public void enqueue(int x)

```

```

    {
        last = (last + 1) % QUEUE_SIZE;
        q[last] = x;
        count++;
    }

    public int dequeue()
    {
        int x = q[first];
        first = (first + 1) % QUEUE_SIZE;
        count--;
        return x;
    }

    public int getSize()
    {
        return count;
    }
}

public class queue
{
    public static void main(String[] args)
    {
        Queue q = new Queue();

        q.enqueue(1);
        q.enqueue(2);
        q.enqueue(3);

        while (q.getSize() != 0)
            System.out.println(q.dequeue());
    }
}

```