



รายงาน

เรื่อง โปรแกรม INSERT/DELETE function of Circular Queue

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายงาน

เรื่อง โปรแกรม INSERT/DELETE function of Circular Queue

จัดทำโดย

นายพงษ์พันธุ์ เลาวพงศ์

รหัสนักศึกษา 66543206019-2

เสนอ

อาจารย์ปิยพล ยืนยงสถาวร

รายงานนี้เป็นส่วนหนึ่งของวิชา

ENGCE124

โครงสร้างข้อมูลและขั้นตอนวิธี

(Data Structures and Algorithms)

หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาคพายัพ เชียงใหม่

ภาคเรียนที่ 1 ปีการศึกษา 2567

## คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ENGCE124 โครงสร้างข้อมูลและขั้นตอนวิธี (Data Structures and Algorithms) หลักสูตร วศ.บ.วิศวกรรมคอมพิวเตอร์ สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา ภาควิชาฯ เชียงใหม่ ในระดับปริญญาตรีปีที่ 2 โดยมีจุดประสงค์ในการอธิบายโค้ดของโปรแกรม INSERT/DELETE function of Circular Queue รวมถึงอธิบายหลักการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue และอธิบายผลลัพธ์การใช้งานโปรแกรม INSERT/DELETE function of Circular Queue

ผู้จัดทำรายงานหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้สนใจ หรือนักศึกษาทุกท่านที่กำลังหาศึกษาในหัวข้อของโปรแกรม INSERT/DELETE function of Circular Queue หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้ และขออภัยมา ณ ที่นี้

ผู้จัดทำ

นายพงษ์พันธุ์ เลาวพงศ์

วันที่ 29/07/2567

## สารบัญ

	หน้า
คำนำ	ก
สารบัญ	๗
โค้ดของโปรแกรม INSERT/DELETE function of Circular Queue พร้อมคำอธิบาย	1
หลักการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue	5
ผลลัพธ์การใช้งานโปรแกรม INSERT/DELETE function of Circular Queue	14
บรรณานุกรม	16

## โค้ดของโปรแกรม INSERT/DELETE function of Circular Queue พร้อมคำอธิบาย

```
#include <stdio.h> // ใช้สำหรับ printf()

#include <conio.h> // ใช้สำหรับ getch()

#define N 5 // กำหนดขนาดสูงสุดของคิว

int Q[N]; // เตรียมคิว ขนาด 0 ถึง N-1

int x, Qnumber = 0, F = 0, R = 0; // ประกาศตัวแปร x และกำหนดค่าเริ่มต้นให้กับ Qnumber, Front (F), Rear (R)

char status = 'N'; // กำหนดสถานะเริ่มต้นเป็น 'N' (NORMAL)

char ch; // ตัวแปรสำหรับอ่านค่าจากคีย์บอร์ด

void insertCQ(int y) // ฟังก์ชันแทรกข้อมูลเข้าไปในคิว
{
    if((R == F - 1) || (R == N - 1 && F == 1)) // ตรวจสอบว่าคิวเต็มหรือไม่
    {
        printf("!!!OVER FLOW!!!...\n");
        status = 'O'; // กำหนดสถานะเป็น 'O' (OVER FLOW)
    }
    else
    {
        if(R == N - 1) // ถ้า R ถึงขีดจำกัดสูงสุด ให้กลับไปอยู่ที่ 1
        {
            R = 1;
        }
        else
        {
            R++; // เพิ่มค่า R ถ้าไม่ถึงขีดจำกัดสูงสุด

            if(F == 0) // ถ้า F เป็น 0 ให้เปลี่ยนเป็น 1
            {
                F = 1;
            }
        }

        Qnumber++; // เพิ่มจำนวนคิว

        printf("You are queue number: %d\n", Qnumber); // แสดงหมายเลขคิว

        Q[R] = y; // ใส่ข้อมูลลงในคิว

        status = 'N'; // กำหนดสถานะเป็น 'N' (NORMAL)
    }
}
```

## โค้ดของโปรแกรม INSERT/DELETE function of Circular Queue พร้อมคำอธิบาย (ต่อ)

```

int deleteCQ() // ฟังก์ชันลบข้อมูลจากคิว
{
    int y;
    if (F == 0) // ตรวจสอบว่าคิวว่างหรือไม่
    {
        printf("\n!!!UNDER FLOW!!!...\n");
        status = 'U'; // กำหนดสถานะเป็น 'U' (UNDER FLOW)
    }
    else
    {
        y = Q[F]; // ดึงข้อมูลจากคิว

        if (F == R) // ถ้า F และ R มีค่าเท่ากัน ให้เปลี่ยนเป็น 0
        {
            F = 0; R = 0;
        }
        else
        {
            if (F == N - 1) // ถ้า F ถึงขีดจำกัดสูงสุด ให้กลับไป 1
            {
                F = 1;
            }
            else
            {
                F++; // เพิ่มค่า F ถ้าไม่ถึงขีดจำกัดสูงสุด
            }
        }

        status = 'N'; // กำหนดสถานะเป็น 'N' (NORMAL)

        return(y); // ส่งค่าข้อมูลที่ลบออกไป
    }
}

```

## โค้ดของโปรแกรม INSERT/DELETE function of Circular Queue พร้อมคำอธิบาย (ต่อ)

```

int DataInQueue() // คำนวณจำนวนข้อมูลที่รออยู่ในคิว
{
    int y = 0;
    if (F != 0 && R != 0) // ถ้า F และ R ไม่เป็น 0 จึงจะคำนวณได้
    {
        if (F <= R)
            y = R - F + 1; // กรณี F และ R ปกติ
        else
            y = (N - 1) - F + 1 + R; // กรณี R วนกลับ
    }
    return(y);
}

void ShowAllQueue() // ฟังก์ชันแสดงข้อมูลทั้งหมดในคิว
{
    int i; // ตัวแปรนับ
    printf("N : %d\n", N - 1);
    printf("Status = %c\n", status); // แสดงสถานะ
    printf("Data waiting in queue = %d\n", DataInQueue()); // แสดง
    จำนวนข้อมูลที่รออยู่ในคิว
    printf("F = %d / R = %d\n", F, R); // แสดงค่า F และ R
    for (i = 1; i < N; i++)
    {
        printf("%d:%d / ", i, Q[i]); // แสดงข้อมูลทั้งหมดในคิว
    }
    printf("\n-----\n");
}

```

## โค้ดของโปรแกรม INSERT/DELETE function of Circular Queue พร้อมคำอธิบาย (ต่อ)

```
int main()
{
    printf("CIRCULAR QUEUE PROGRAM...\n");
    printf("=====\n");
    ch = ' ';
    while (ch != 'E' && ch != 'e')
    {
        printf("\n[1=INSERT : 2=DELETE E:Exit] : "); // แสดงเมนู

        ch = getch(); // รอและอ่านค่าจากคีย์บอร์ดโดยไม่ต้องกด Enter

        switch(ch) // ตรวจสอบค่าที่อ่านได้จากคีย์บอร์ด
        {
            case '1' :
                printf("\nInsert Number : ");

                scanf("%d", &x); // อ่านข้อมูลจากคีย์บอร์ด

                insertCQ(x); // เรียกใช้ฟังก์ชันแทรกข้อมูลเข้าไปในคิว

                ShowAllQueue(); // แสดงข้อมูลทั้งหมดในคิว

                break;
            case '2' :
                x = deleteCQ(); // ลบข้อมูลจากคิว

                printf("\nData from Queue = %d\n", x); // แสดงข้อมูลที่ถูกลบ

                ShowAllQueue(); // แสดงข้อมูลทั้งหมดในคิว

                break;
        } // จบ switch case

    } // จบ while loop

    printf("\n"); // เพิ่มบรรทัดว่าง

    return(0); // ส่งค่ากลับจาก main function
} // จบ main function
```



## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue

### 1. การประกาศและกำหนดค่าเริ่มต้น

```
#include <stdio.h> // ใช้สำหรับ printf()

#include <conio.h> // ใช้สำหรับ getch()

#define N 5 // กำหนดขนาดสูงสุดของคิว

int Q[N]; // เตรียมคิว ขนาด 0 ถึง N-1

int x, Qnumber = 0, F = 0, R = 0; // ประกาศตัวแปร x และกำหนดค่าเริ่มต้น
// ให้กับ Qnumber, Front (F), Rear (R)

char status = 'N'; // กำหนดสถานะเริ่มต้นเป็น 'N' (NORMAL)

char ch; // ตัวแปรสำหรับอ่านค่าจากคีย์บอร์ด
```

ส่วนนี้เป็นการรวม header files และการประกาศตัวแปรที่จำเป็นต้องใช้ในโปรแกรม:

- #define N 5: กำหนดขนาดสูงสุดของคิวเป็น 5
- int Q[N];: ประกาศ array Q ขนาด 5 เพื่อใช้เก็บข้อมูลในคิว
- int x;: ตัวแปรชั่วคราวสำหรับเก็บข้อมูลที่จะใช้ในการ insert หรือ delete
- int Qnumber = 0, F = 0, R = 0;: ตัวแปรสำหรับเก็บจำนวนคิว, ตัวชี้ตำแหน่งหน้า (Front), และตัวชี้ตำแหน่งหลัง (Rear) ที่ตำแหน่งเริ่มต้น
- char status = 'N';: ตัวแปรสถานะของคิว ('N': NORMAL, 'O': OVER FLOW, 'U': UNDER FLOW)
- char ch;: ตัวแปรสำหรับเก็บค่าที่อ่านจากคีย์บอร์ด

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 2. ฟังก์ชันแทรกข้อมูลเข้าไปในคิว

```
void insertCQ(int y) // ฟังก์ชันแทรกข้อมูลเข้าไปในคิว
{
    if((R == F - 1) || (R == N - 1 && F == 1)) // ตรวจสอบว่าคิวเต็มหรือไม่
    {
        printf("!!!OVER FLOW!!!...\n");
        status = 'O'; // กำหนดสถานะเป็น 'O' (OVER FLOW)
    }
    else
    {
        if(R == N - 1) // ถ้า R ถึงขีดจำกัดสูงสุด ให้กลับไป 1
        {
            R = 1;
        }
        else
        {
            R++; // เพิ่มค่า R ถ้าไม่ถึงขีดจำกัดสูงสุด

            if(F == 0) // ถ้า F เป็น 0 ให้เปลี่ยนเป็น 1
            {
                F = 1;
            }
        }
        Qnumber++; // เพิ่มจำนวนคิว

        printf("You are queue number: %d\n", Qnumber); // แสดงหมายเลขคิว

        Q[R] = y; // ใส่ข้อมูลลงในคิว

        status = 'N'; // กำหนดสถานะเป็น 'N' (NORMAL)
    }
}
```

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 2. ฟังก์ชันแทรกข้อมูลเข้าไปในคิว

ฟังก์ชันนี้ใช้ในการเพิ่มข้อมูลเข้าในคิว:

- เช็คว่าคิวเต็มหรือไม่
- ถ้าคิวเต็ม จะแสดงข้อความ "!!!OVER FLOW!!!" และเปลี่ยนสถานะเป็น 'O'
- ถ้าคิวไม่เต็ม จะเพิ่มค่า R ถ้า R ถึงขีดจำกัดสูงสุดจะวนกลับไป 1
- ถ้า F เป็น 0 จะเปลี่ยนเป็น 1
- เพิ่มจำนวนคิว (Qnumber)
- แสดงหมายเลขคิว
- ใส่ข้อมูลลงในคิวที่ตำแหน่ง R
- กำหนดสถานะเป็น 'N' (NORMAL)

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 3. ฟังก์ชันลบข้อมูลจากคิว

```
int deleteCQ() // ฟังก์ชันลบข้อมูลจากคิว
{
    int y;
    if (F == 0) // ตรวจสอบว่าคิวว่างหรือไม่
    {
        printf("\n!!!UNDER FLOW!!!...\n");
        status = 'U'; // กำหนดสถานะเป็น 'U' (UNDER FLOW)
    }
    else
    {
        y = Q[F]; // ดึงข้อมูลจากคิว

        if (F == R) // ถ้า F และ R มีค่าเท่ากัน ให้เปลี่ยนเป็น 0
        {
            F = 0; R = 0;
        }
        else
        {
            if (F == N - 1) // ถ้า F ถึงขีดจำกัดสูงสุด ให้กลับไป 1
            {
                F = 1;
            }
            else
            {
                F++; // เพิ่มค่า F ถ้าไม่ถึงขีดจำกัดสูงสุด
            }
        }

        status = 'N'; // กำหนดสถานะเป็น 'N' (NORMAL)

        return(y); // ส่งค่าข้อมูลที่ลบออกไป
    }
}
```

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 3. ฟังก์ชันลบข้อมูลจากคิว

ฟังก์ชันนี้ใช้ในการดึงข้อมูลออกจากคิว:

- เช็คว่าคิวว่างหรือไม่
- ถ้าคิวว่าง จะแสดงข้อความ "!!!UNDER FLOW!!!" และเปลี่ยนสถานะเป็น 'U'
- ถ้าคิวไม่ว่าง จะดึงข้อมูลจากคิวที่ตำแหน่ง F
- ถ้า F และ R มีค่าเท่ากัน จะเปลี่ยนเป็น 0
- ถ้า F ถึงขีดจำกัดสูงสุดจะวนกลับไป 1
- เพิ่มค่า F ถ้าไม่ถึงขีดจำกัดสูงสุด
- กำหนดสถานะเป็น 'N' (NORMAL)
- ส่งคืนค่าข้อมูลที่ลบออกไป

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 4. ฟังก์ชันคำนวณจำนวนข้อมูลที่รอกอยู่ในคิว

```
int DataInQueue() // คำนวณจำนวนข้อมูลที่รอกอยู่ในคิว
{
    int y = 0;
    if (F != 0 && R != 0) // ถ้า F และ R ไม่เป็น 0 จึงจะคำนวณได้
    {
        if (F <= R)
            y = R - F + 1; // กรณี F และ R ปกติ
        else
            y = (N - 1) - F + 1 + R; // กรณี R วนกลับ
    }
    return(y);
}
```

ฟังก์ชันนี้ใช้ในการคำนวณจำนวนข้อมูลที่รอกอยู่ในคิว:

- ถ้า F และ R ไม่เป็น 0 จะทำการคำนวณ
- ถ้า F น้อยกว่าหรือเท่ากับ R จะคำนวณแบบปกติ
- ถ้า F มากกว่า R แสดงว่า R วนกลับ จะคำนวณแบบวงกลม

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 5. ฟังก์ชันแสดงข้อมูลทั้งหมดในคิว

```
void ShowAllQueue() // ฟังก์ชันแสดงข้อมูลทั้งหมดในคิว
{
    int i; // ตัวแปรนับ
    printf("N : %d\n", N - 1);
    printf("Status = %c\n", status); // แสดงสถานะ
    printf("Data waiting in queue = %d\n", DataInQueue()); // แสดง
    จำนวนข้อมูลที่รออยู่ในคิว
    printf("F = %d / R = %d\n", F, R); // แสดงค่า F และ R
    for (i = 1; i < N; i++)
    {
        printf("%d:%d / ", i, Q[i]); // แสดงข้อมูลทั้งหมดในคิว
    }
    printf("\n-----\n");
}
```

ฟังก์ชันนี้ใช้ในการแสดงข้อมูลทั้งหมดในคิว:

- แสดงขนาดสูงสุดของคิว
- แสดงสถานะปัจจุบันของคิว
- แสดงจำนวนข้อมูลที่รออยู่ในคิว
- แสดงค่า F และ R
- แสดงข้อมูลทั้งหมดในคิว

## หลักการการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 6. ฟังก์ชัน main

```
int main()
{
    printf("CIRCULAR QUEUE PROGRAM...\n");
    printf("=====\n");
    ch = ' ';
    while (ch != 'E' && ch != 'e')
    {
        printf("\n[1=INSERT : 2=DELETE E:Exit] : "); // แสดงเมนู

        ch = getch(); // รอและอ่านค่าจากคีย์บอร์ดโดยไม่ต้องกด Enter

        switch(ch) // ตรวจสอบค่าที่อ่านได้จากคีย์บอร์ด
        {
            case '1' :
                printf("\nInsert Number : ");
                scanf("%d", &x); // อ่านข้อมูลจากคีย์บอร์ด

                insertCQ(x); // เรียกใช้ฟังก์ชันแทรกข้อมูลเข้าไปในคิว

                ShowAllQueue(); // แสดงข้อมูลทั้งหมดในคิว

                break;
            case '2' :
                x = deleteCQ(); // ลบข้อมูลจากคิว

                printf("\nData from Queue = %d\n", x); // แสดงข้อมูลที่ถูกลบ

                ShowAllQueue(); // แสดงข้อมูลทั้งหมดในคิว

                break;
        } // จบ switch case
    } // จบ while loop

    printf("\n"); // เพิ่มบรรทัดว่าง

    return(0); // ส่งค่ากลับจาก main function
} // จบ main function
```



## หลักการทำงานของโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 6. ฟังก์ชัน main

ฟังก์ชันนี้เป็นฟังก์ชันหลักของโปรแกรม:

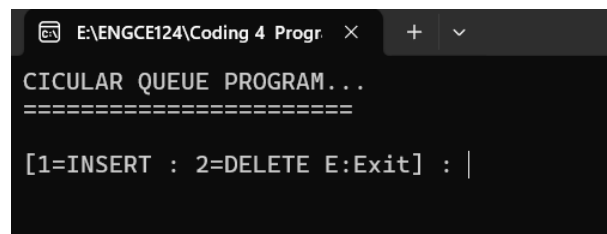
- แสดงข้อความต้อนรับและเมนูการใช้งาน (INSERT, DELETE, EXIT)
- รอและอ่านค่าจากคีย์บอร์ดโดยไม่ต้องกด Enter
- ใช้คำสั่ง switch-case เพื่อตรวจสอบค่าที่อ่านมา:
  - ถ้ากด '1' ให้กรอกข้อมูลและเรียกฟังก์ชัน insertCQ จากนั้นแสดงข้อมูลทั้งหมดในคิว
  - ถ้ากด '2' ให้เรียกฟังก์ชัน deleteCQ และแสดงข้อมูลที่ดึงออกมา จากนั้นแสดงข้อมูลทั้งหมดในคิว
- วนลูปไปเรื่อย ๆ จนกว่าจะกด 'E' หรือ 'e' เพื่อออกจากโปรแกรม

การทำงานโดยรวมของโปรแกรมนี้อคือการจัดการข้อมูลในคิวด้วยการแทรกและลบข้อมูล พร้อมทั้งแสดงข้อมูลในคิวทุกครั้งที่มีการเปลี่ยนแปลง

## ผลลัพธ์การใช้งานโปรแกรม INSERT/DELETE function of Circular Queue

### 1. เมื่อเริ่มโปรแกรม:

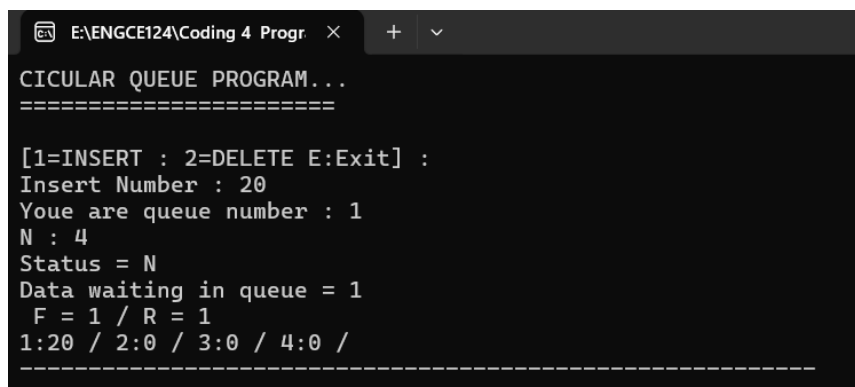
- โปรแกรมจะแสดงข้อความต้อนรับและเมนูการใช้งาน
- ผู้ใช้สามารถกด '1' เพื่อแทรกข้อมูล, '2' เพื่อลบข้อมูล, หรือ 'E' เพื่อออกจากโปรแกรม



```
E:\ENGCE124\Coding 4 Progr. X + v
CICULAR QUEUE PROGRAM...
=====
[1=INSERT : 2=DELETE E:Exit] : |
```

### 2. เมื่อแทรกข้อมูล:

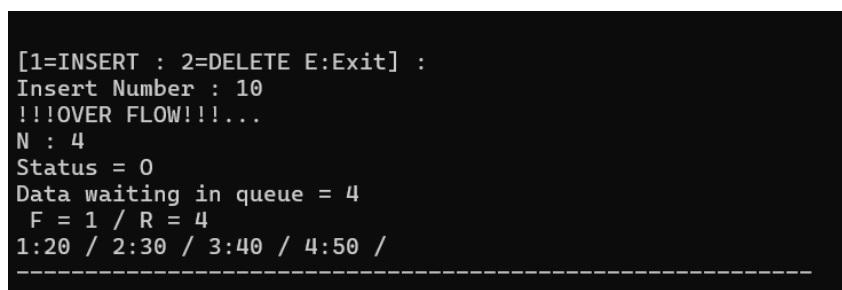
- ถ้าผู้ใช้กด '1' โปรแกรมจะขอให้ผู้ใช้กรอกข้อมูล
- เมื่อผู้ใช้กรอกข้อมูลและกด Enter โปรแกรมจะเพิ่มข้อมูลลงในคิวและแสดงผลลัพธ์



```
E:\ENGCE124\Coding 4 Progr. X + v
CICULAR QUEUE PROGRAM...
=====
[1=INSERT : 2=DELETE E:Exit] :
Insert Number : 20
Youe are queue number : 1
N : 4
Status = N
Data waiting in queue = 1
F = 1 / R = 1
1:20 / 2:0 / 3:0 / 4:0 /
=====
```

### 3. เมื่อคิวเต็ม:

- ถ้าผู้ใช้พยายามแทรกข้อมูลในขณะที่คิวเต็ม โปรแกรมจะแสดงข้อความ



```
[1=INSERT : 2=DELETE E:Exit] :
Insert Number : 10
!!!OVER FLOW!!!...
N : 4
Status = 0
Data waiting in queue = 4
F = 1 / R = 4
1:20 / 2:30 / 3:40 / 4:50 /
=====
```

## ผลลัพธ์การใช้งานโปรแกรม INSERT/DELETE function of Circular Queue (ต่อ)

### 4. เมื่อลบข้อมูล:

- ถ้าผู้ใช้กด '2' โปรแกรมจะลบข้อมูลจากคิวและแสดงผลลัพธ์

```
[1=INSERT : 2=DELETE E:Exit] :
Data from Queue = 10
N : 4
Status = N
Data waiting in queue = 3
F = 2 / R = 4
1:10 / 2:20 / 3:30 / 4:40 /
-----
[1=INSERT : 2=DELETE E:Exit] : |
```

### 5. เมื่อคิวว่าง:

- ถ้าผู้ใช้พยายามลบข้อมูลในขณะที่คิวว่าง โปรแกรมจะแสดงข้อความ

```
[1=INSERT : 2=DELETE E:Exit] :
!!!UNDER FLOW!!!...

Data from Queue = 0
N : 4
Status = U
Data waiting in queue = 0
F = 0 / R = 0
1:10 / 2:20 / 3:30 / 4:40 /
-----
```

### 6. เมื่อออกจากโปรแกรม:

- ถ้าผู้ใช้กด 'E' หรือ 'e' โปรแกรมจะออกจากลูปและแสดงบรรทัดว่าง จากนั้นจะสิ้นสุดการทำงาน

```
-----
[1=INSERT : 2=DELETE E:Exit] :
-----
Process exited after 907.3 seconds with return value 0
Press any key to continue . . . |
```

โปรแกรมคิววงกลม ถูกออกแบบมาเพื่อจัดการข้อมูลในลักษณะของคิววงกลม ซึ่งมีคุณสมบัติในการจัดเก็บข้อมูลที่เข้ามาและออกไปตามลำดับที่เข้ามาในรูปแบบของวงกลม การทำงานของโปรแกรมเริ่มต้นด้วยการตั้งค่าคิวให้มีขนาดสูงสุด 5 ช่อง โดยการใช้การประกาศและกำหนดค่าเริ่มต้นของตัวแปรต่างๆ เช่น F (Front), R (Rear), และ Qnumber เพื่อใช้ในการติดตามสถานะและการจัดการคิว

### บรรณานุกรม

ChatGPT. ( - ). Implementing and Managing a Circular Queue in C. สืบค้น 29 กรกฎาคม 2567,  
จาก <https://chatgpt.com/>