

Ring buffer

```
#include <stdio.h>
#include <windows.h>
#include <stdarg.h>
#include <conio.h>
#include <time.h>

typedef unsigned int Cmd_t;
#define BUFSIZE      8
#define EMPTY -1
static int nRdCnts    = 0;    // make sure the EnQ don't write to it
static int nWrCnts    = 0;    // make sure the DeQ don't write to it
static int nRear      = 0;
static int nFront     = 0;
static Cmd_t cElement = 0;
static Cmd_t RingBuf[BUFSIZE];

int iEnQ(Cmd_t nCommand);
int iDeQ(Cmd_t* nCommand);

int main(void)
{
    int i = 0;
    Cmd_t n = 0;
    Cmd_t poke = 0;

    printf(" ----- Simple Circular buffer practice ----- \n\n");

    n = 600;
    printf(" Add %4d ->", n); iEnQ(n); // to location #0
    n = 34;
    printf(" Add %4d ->", n); iEnQ(n); // to location #1
    n = 5678;
    printf(" Add %4d ->", n); iEnQ(n); // to location #2
    n = 0;
    printf(" Add %4d ->", n); iEnQ(n); // to location #3
    printf("\n");

    n = 891;
    printf(" Add %4d ->", n); iEnQ(n); // to location #4
    n = 7;
    printf(" Add %4d ->", n); iEnQ(n); // to location #5
    n = 12;
    printf(" Add %4d ->", n); iEnQ(n); // to location #6
    n = 446;
    printf(" Add %4d ->", n); iEnQ(n); // to location #7
    printf("\n");

    n = 77;
    printf(" Add %4d ->", n); iEnQ(n); // to location #0
    printf("\n");
    n = 321;
    printf(" Add %4d ->", n); iEnQ(n); // to location #1
    printf("\n");
    iDeQ(&poke);
    printf(" Get %4d, ", (int)poke);           // fr location #2
    printf("\n");
    n = 4;
```

```

printf(" Add %4d ->", n); iEnQ(n); // to location #2
printf("\n");
do {
    if (iDeQ(&poke) == EMPTY)
        break;
    printf(" Get %4d, ", (int)poke);
} while(i++ <= BUFSIZE);
printf("\n");

printf("\n\nThis program %s \n\t is compiled (%s)\n\t using C compiler version %lu\n\n",
        __FILE__, __TIMESTAMP__, __MSC_FULL_VER);
printf("press a key to end this program...");
getch();
return 0;
}

```

```

int iEnQ(Cmd_t nCommand)
{
    // make sure don't write to nRdCnts
    // the unsigned of diff reveals the Q's % fullness [0...100]
    int diff = nWrCnts - nRdCnts;
    if (diff < 0)
        diff = !diff;

    if ((nFront == nRear) && (7 <= diff))
    {
        // true when Q is full
        nRear = nFront;
        nFront++; // make the next one the oldest
    }
    printf("[%d] ", nRear);
    RingBuf[nRear] = nCommand;
    nWrCnts++;
    nRear++;
    if (nRear >= 8)
        nRear = 0;
    if (nWrCnts > BUFSIZE) {
        nWrCnts = 8;
    }
    return 1;
}

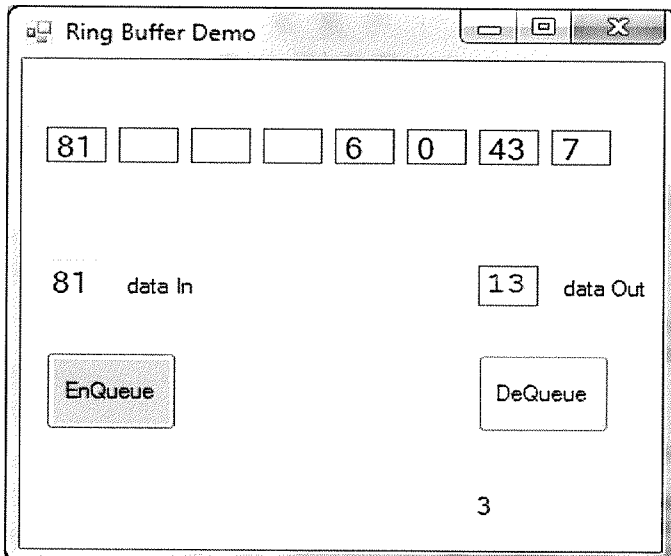
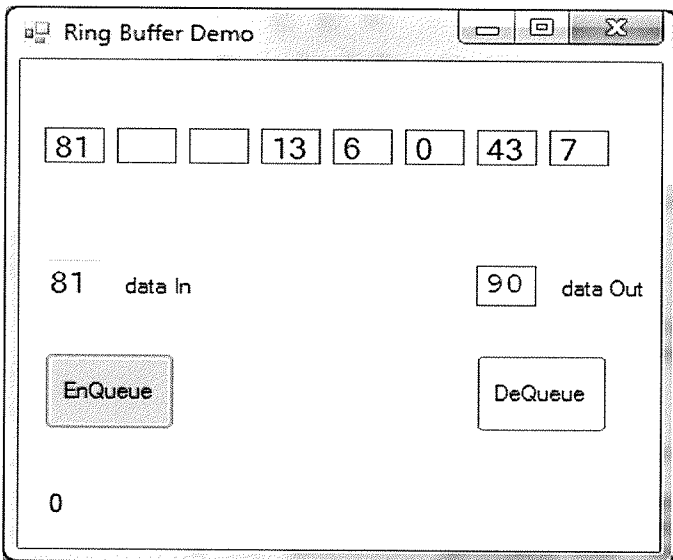
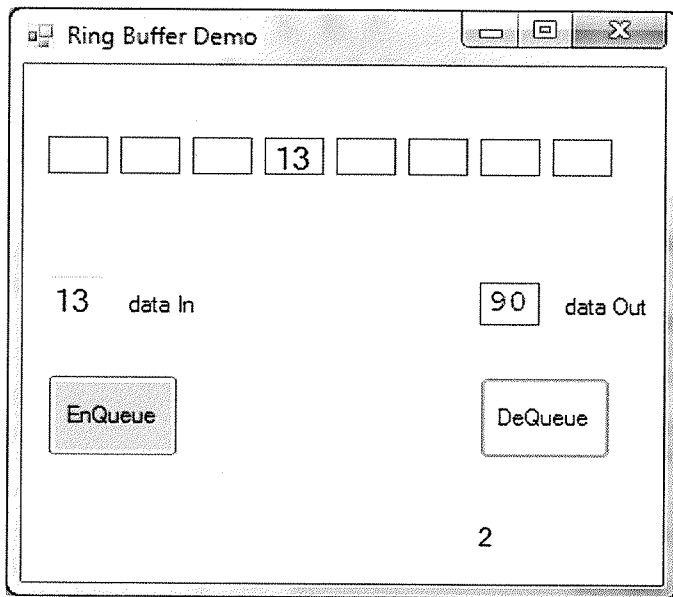
```

```

int iDeQ(Cmd_t* nCommand)
{
    // make sure don't write to nWrCnts
    if ((nRear == nFront) && (nWrCnts == nRdCnts)) // it's empty
        return EMPTY;
    printf("[%d] ", nFront);
    *nCommand = RingBuf[nFront];
    nFront++;
    nRdCnts++;
    if (nRdCnts > BUFSIZE)
        nRdCnts = 8;
    if (nFront >= BUFSIZE)
        nFront = 0; // wrap around to the first

    return 1;
}

```




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace RingbufDemo
{
    public partial class RingBufDemoForm : Form
    {
        private int nWrCnts = 0;
        private int nRdCnts = 0;
        private int nRear = 0;
        private int nFront = 0;
        private int Currentslot = 0;
        private int BUFSIZE = 8;
        private int nItems = 0;
        private int EMPTY = -1;

        private string[] RingBuf = new string[8];

        public RingBufDemoForm()
        {
            InitializeComponent();
        }

        private void btn_EnQ_Click(object sender, EventArgs e)
        {
            switch (iEnQ(txt_data.Text))
            {
                case 0:
                    label1.Text = txt_data.Text;
                    break;
                case 1:
                    label2.Text = txt_data.Text;
                    break;
                case 2:
                    label3.Text = txt_data.Text;
                    break;
                case 3:
                    label4.Text = txt_data.Text;
                    break;
                case 4:
                    label5.Text = txt_data.Text;
                    break;
                case 5:
                    label6.Text = txt_data.Text;
                    break;
                case 6:
                    label7.Text = txt_data.Text;
                    break;
                case 7:
                    label8.Text = txt_data.Text;
                    break;
                default:
                    break;
            }
        }

        private void btn_DeQ_Click(object sender, EventArgs e)
        {

```

```
int slot = 0;
string command = "{}";
slot = iDeQ(ref command);
lbl_DataOut.Text = command;
switch (slot)
{
    case 0:
        label1.Text = "";
        break;
    case 1:
        label2.Text = "";
        break;
    case 2:
        label3.Text = "";
        break;
    case 3:
        label4.Text = "";
        break;
    case 4:
        label5.Text = "";
        break;
    case 5:
        label6.Text = "";
        break;
    case 6:
        label7.Text = "";
        break;
    case 7:
        label8.Text = "";
        break;
    default:
        break;
}
}

private int iEnQ(string command)
{
    RingBuf[nRear] = command;
    nItems += 1;

    Currentslot = nRear;
    nWrCnts += 1;
    nRear += 1;

    if (nWrCnts >= BUFSIZE)
        nWrCnts = 8;
    if (nRear >= 8)
        nRear = 0;
    if (nItems > 8)
    {
        nItems = 8;
        nFront += 1;    //# make the previous one the oldest
    }
    if (nFront >= 8)
        nFront = 0;
    if ((nFront == nRear))
        lbl_eslot.Text = " Last item to fill" + Currentslot.ToString();

    lbl_eslot.Text = Currentslot.ToString();
    if (nItems == 8)
        lbl_eslot.Text = Currentslot.ToString() + "{full}";
    lbl_dslot.Text = "";
    return Currentslot;
}

private int iDeQ(ref string command)
```

```
{
    if ((nRear == nFront) && (nWrCnts == nRdCnts))
    {
        nRear = 0;
        nFront = 0;
        nWrCnts = 0;
        nRdCnts = 0;
        lbl_dslot.Text = "(Empty)";
        lbl_eslot.Text = "";
        return EMPTY;
    }
    command = RingBuf[nFront];
    nItems -= 1;
    Currentslot = nFront;
    nFront += 1;
    nRdCnts += 1;
    if (nRdCnts > BUFSIZE)
        nRdCnts = 8;
    if (nFront >= BUFSIZE)
        nFront = 0; // # wrap around to the first

    lbl_dslot.Text = Currentslot.ToString();
    lbl_eslot.Text = "";
    return Currentslot;
}
}
```