

# 자료구조

## 과제2

컴퓨터공학과

2019305059

이현수

[소스코드]

```
#include<iostream>
using namespace std;

typedef struct nodeRecord{
    int Data;           //노드내부의데이터
    nodeRecord* Next;   //노드타입을 가리킴
}node;
typedef node* Nptr;

class listClass
{
public:
    listClass();
    ~listClass();
    void Insert(int Position, int Item);
    void Delete(int Position);
    int Retrieve(int Position);
    bool IsEmpty();
    int Lenght();
private:
    int Count; //리스트안에있는 의미있는 값의 개수
    Nptr Head; //헤드포인터
};

listClass::listClass() {
    Count = 0;    //개수 0으로 초기화
    Head = NULL;  //헤드가 NULL을 가리키게 함
}

listClass::~~listClass() { }

void listClass::Insert(int Position, int Item) {
    if ((Position > (Count + 1)) || (Position < 1))
    {    //Position이 이격발생 혹은 음수,0일경우
        cout<<"Position out of Range"<<endl;
    }
    else {
        Nptr p = new node; //삽입할 노드 동적할당
        p->Data = Item; //데이터 삽입
        p->Next = NULL; //삽입노드의next가 NULL가리킴
        if (Position == 1) { //첫위치에 삽입할 경우
            p->Next = Head; //삽입node가 현재 첫node 가리킴
            Head = p; //헤드가 삽입노드를 가리킴
        }
        else { //첫위치가 아닌경우. 중간혹은 마지막
            Nptr Temp = Head; //Temp가 첫노드 가리킴
            for (int i = 1; i < (Position - 1); i++) {
                Temp = Temp->Next; //Temp가 삽입직전 노드를 가리키게
            }
            p->Next = Temp->Next; //삽입node의next와 삽입 후 뒤에있을 노드와연결
            Temp->Next = p;      //삽입앞노드next를 삽입노드에 연결
        }
        Count += 1; //개수 1증가
    }
}

void listClass::Delete(int Position) {
    if (IsEmpty()) { //리스트가 비어있을 경우
        cout<<"list empty"<<endl;
    }
    else if ((Position > (Count)) || (Position < 1))
    {    //삭제할위치가 비어있는 공간 혹은 음수,0일경우
        cout<<"Position out of Range"<<endl;
    }
    else {
        if (Position == 1) { //첫번째 노드 삭제할경우
```

```

        Nptr p = Head; //p가 첫노드가리킴
        Head = Head->Next; //헤드가 삭제할노드 다음노드가리킴
        delete p; //공간해체
    }
    else { //그 외
        Nptr Temp = Head; //Temp가 첫노드 가리킴
        for (int i = 1; i < (Position - 1); i++) {
            Temp = Temp->Next; //Temp가 삭제할노드 직전노드가리킴
        }
        Nptr p = Temp->Next; //p가 삭제할 노드 가리킴
        Temp->Next = p->Next; //삭제앞노드가 삭제뒤노드와 연결
        delete p; //공간해체
    }
    Count -= 1; //개수 1감소
}
}

```

```

int listClass::Retrieve(int Position) {
    Nptr Temp;
    Temp = Head; //노드포인터 Temp가 첫노드가리킴
    for (int i = 0; i < Position - 1; i++)
    { //Temp가 Position번째(반환할노드) 노드가리킴
        Temp = Temp->Next;
    }
    return Temp->Data; //Temp가 가리키는 노드 Data 반환
}

```

```

bool listClass::IsEmpty() {
    if (Count == 0) return true; //개수가 0이면 1반환
    else return false; //아니면 0반환
}

```

```

int listClass::Lenght() {
    return Count; //개수 반환
}

```

```

class queueClass
{
public:
    queueClass();
    ~queueClass();
    void Add(int Item);
    void Remove();
    int getFront();
    bool IsEmpty();
    bool IsFull();
private:
    listClass L;
};

```

```

queueClass::queueClass() { }

```

```

queueClass::~~queueClass() {
    while (!IsEmpty()) { //큐가 빌 때까지
        Remove(); //데이터 삭제
    }
}

```

```

void queueClass::Add(int Item) {
    if (IsFull()) //큐가 꽉차 있을 경우
    {
        cout << "queue is full" << endl;
    }
    else {
        L.Insert(L.Lenght() + 1, Item); //Item을 리스트 마지막에 삽입
    }
}

```

```

void queueClass::Remove() {
    if (IsEmpty()) //큐가 비어있다면
    {
        cout << "list empty" << endl;
    }
    else{
        L.Delete(1); //맨 첫번째 데이터 삭제
    }
}

int queueClass::getFront() {
    if (IsEmpty()) //큐가 비어있다면
    {
        cout << "list empty" << endl;
    }
    else {
        return L.Retrieve(1); //맨 첫번째 데이터 반환
    }
}

bool queueClass::IsEmpty() {
    if (L.IsEmpty())return true;//리스트가 비어있다면 true반환
    else return false;        //아니면 false 반환
}

int total; //IsFull()함수를 위해 선언

bool queueClass::IsFull() {
    if (L.Lenght() == total) return true;//데이터수가 사용자입력total개라면 true반환
    else return false;        //아니면 false반환
}

int main(void) {
    int interval; //데이터개수, 간격변수
    cout << "데이터 개수? 간격?";
    cin >> total >> interval;

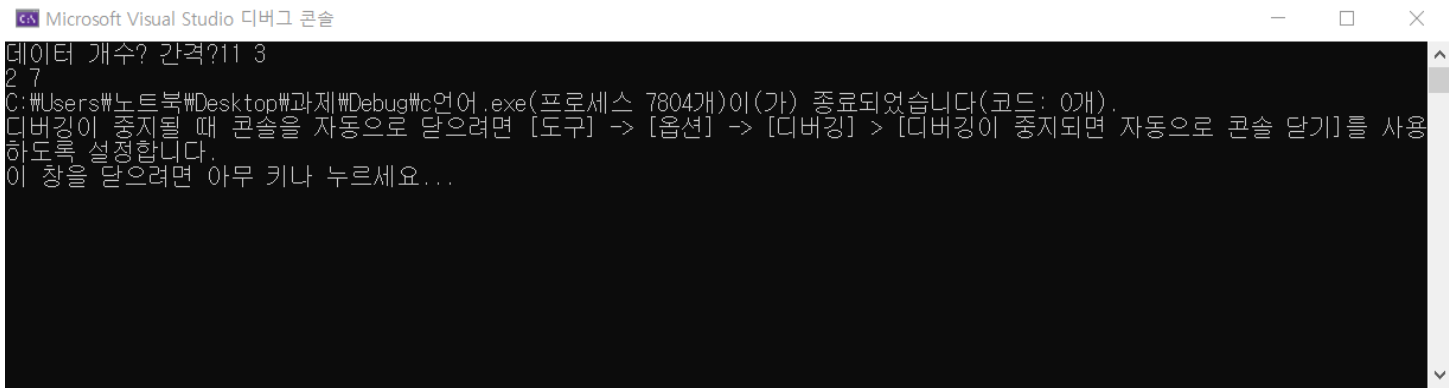
    queueClass Q; //큐 Q선언
    for (int i = 1; i <= total; i++)
    { //1번째부터 total번째까지 1~total 값 삽입
        Q.Add(i);
    }
    int n = 0; //삭제할데이터 판별할때 사용
    int survivor = total; //생존자변수 total로 초기화

    //생존자 수가 간격수-1보다 크면 계속 반복
    while (survivor > interval -1) {
        n++;
        if (n% interval ==0) //n이 interval 배수일 때
        {
            Q.Remove(); //첫번째 데이터 삭제
            survivor--; //생존자수 1개 줄임
        }
        else //그 외
        { //temp변수에 담지 않고 데이터삽입 후 삭제 시 큐가 꽉차있어서 안됨.
            int temp = Q.getFront();//temp변수에 첫번째 데이터 저장
            Q.Remove(); //첫번째 데이터 삭제
            Q.Add(temp);//첫번째 데이터 맨 뒤로 삽입
        }
    }
    while (!Q.IsEmpty()){//큐가 빌때까지
        cout << Q.getFront() << ' ';//첫번째 데이터 출력
        Q.Remove(); //첫번째 데이터 삭제
    }
}

```

## [실행]

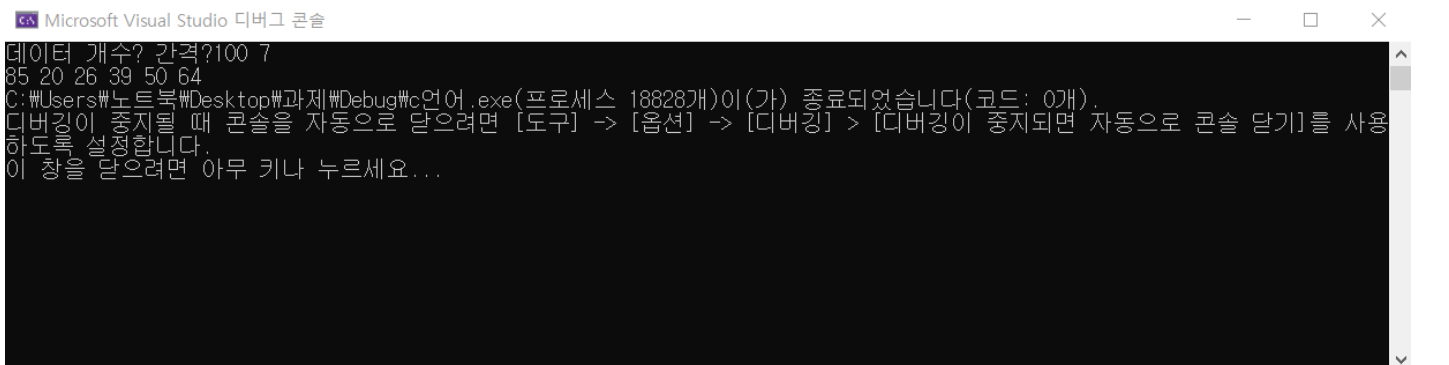
1) 데이터 개수 : 11 / 제거하는 간격 수 : 3



```
Microsoft Visual Studio 디버그 콘솔
데이터 개수? 간격?11 3
2 7
C:\Users\#노트북\Desktop\#과제\#Debug\#c언어.exe(프로세스 7804개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

최종 남은 데이터는 2, 7번째.

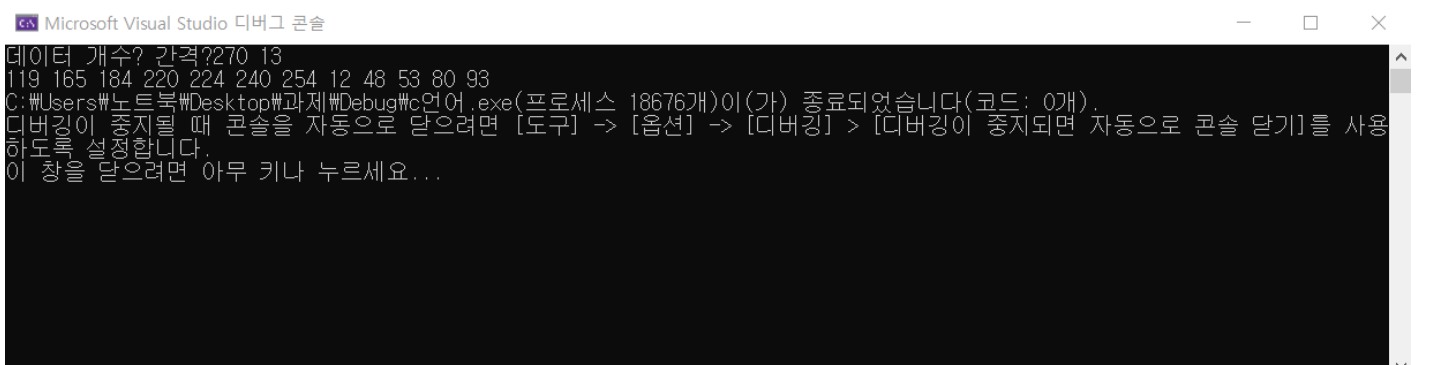
2) 데이터 개수 : 100 / 제거하는 간격 수 : 7



```
Microsoft Visual Studio 디버그 콘솔
데이터 개수? 간격?100 7
85 20 26 39 50 64
C:\Users\#노트북\Desktop\#과제\#Debug\#c언어.exe(프로세스 18828개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

최종 남은 데이터는 20, 26, 39, 50, 64, 85번째.

3) 데이터 개수 : 270 / 제거하는 간격 수 : 13



```
Microsoft Visual Studio 디버그 콘솔
데이터 개수? 간격?270 13
119 165 184 220 224 240 254 12 48 53 80 93
C:\Users\#노트북\Desktop\#과제\#Debug\#c언어.exe(프로세스 18676개)이(가) 종료되었습니다(코드: 0개).
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 중지되면 자동으로 콘솔 닫기]를 사용
하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요...
```

최종 남은 데이터는 12, 48, 53, 80, 93, 119, 165, 184, 220, 224, 240, 254번째.