

LAB #10 (11/29 목)

1. 아래의 코드를 기반으로 두 점의 x, y값을 입력 받고 두 점 사이의 거리를 구하는 프로그램을 작성하라. 이 때 Point class의 x, y 값의 타입은 int, double, float이 될 수 있다. (*.h와 *.cpp를 사용하여 class를 선언 및 정의)

```
int main()
{
    // type에는 int, double, float 중 어느 것이든 들어갈 수 있다
    Point<type> p;

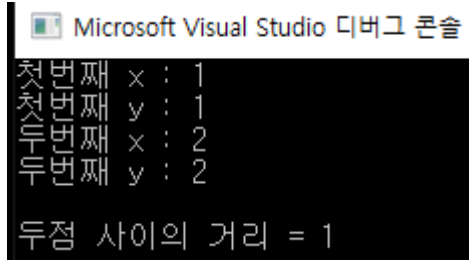
    // 두 점의 position을 입력 받는 함수
    p.setPointFromKeyboard();

    // 두 점 사이의 거리를 출력하는 함수
    p.print();

    return 0;
}
```

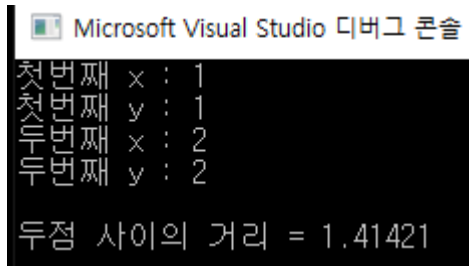
1 - 출력화면 :

<type이 int인 경우>



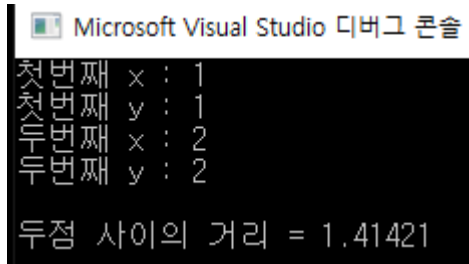
```
Microsoft Visual Studio 디버그 콘솔
첫번째 x : 1
첫번째 y : 1
두번째 x : 2
두번째 y : 2
두점 사이의 거리 = 1
```

<type이 double인 경우>



```
Microsoft Visual Studio 디버그 콘솔
첫번째 x : 1
첫번째 y : 1
두번째 x : 2
두번째 y : 2
두점 사이의 거리 = 1.41421
```

<type이 float인 경우>



```
Microsoft Visual Studio 디버그 콘솔
첫번째 x : 1
첫번째 y : 1
두번째 x : 2
두번째 y : 2
두점 사이의 거리 = 1.41421
```

2. 아래의 조건을 만족하는 프로그램을 작성하라.

1. 크기가 10인 vector1과 vector2를 만든다.
2. vector1의 범위는 0~10이고 vector2의 범위는 0~20이며 난수로 채워진다.
3. vector1에 있는 어떠한 수와 vector2의 있는 어떠한 수를 곱 했을 때 가장 큰 경우, 즉 곱의 최대값, 최소값을 찾는다.
4. 이 때 vector의 데이터에 접근하기 위해서 iterator만을 사용한다.

2 - 출력화면 :

```
<vector 1>
8 9 7 5 1 7 5 6 3 9
<vector 2>
8 19 16 0 20 5 6 8 14 18

최대값 = 180
최소값 = 0
```

3. 아래 코드를 기반으로 다양한 type을 사용하여 추가, 삭제, 출력 기능을 하는 List class를 구현하고 이를 사용하는 프로그램을 작성하라. (*.h와 *.cpp를 사용하여 class를 선언 및 정의)

```
int command()
{
    int num;

    cout << "WnWt---- menu ----" << endl;
    cout << "Wt1. 리스트 추가" << endl;
    cout << "Wt2. 리스트 삭제" << endl;
    cout << "Wt3. 리스트 출력" << endl;
    cout << "Wt4. 프로그램 종료" << endl;
    cout << "WnWt입력 --> ";
    cin >> num;

    return num;
}

int main()
{
    CList<type> list; // type형으로 list 선언
    type input; // list에 입력 할 데이터
    int com; // 선택한 기능

    while (1)
    {
        com = command(); // 기능을 선택

        switch (com)
        {
            case 1: // 추가
                cout << "Wn추가할 데이터 : ";
```

```

        cin >> input;
        list.Add(input);
        break;
    case 2: // 삭제
        cout << "Wn삭제할 데이터 : ";
        cin >> input;
        list.Delete(input);
        break;
    case 3: // 출력
        list.Print();
        break;
    case 4: // 프로그램 종료
        cout << "WnWt프로그램을 종료합니다Wn";
        return 0;
        break;
    default:
        break;
    }
}
return 0;
}

```

[참조 1]

```

template <typename T>
class CList
{
public:
    CList();
    ~CList();

    bool IsEmpty(); // list가 비어 있으면 1, 아니면 0
    bool IsFull();  // list가 꽉 차 있으면 1, 아니면 0

    void Add(T data);           // list에 데이터 추가
    void Delete(T data);        // list에 데이터 삭제
    void Print();               // list에 데이터 출력

private:
    T m_Array[5];              // 데이터를 저장할 공간
    int m_Length;              // list에 있는 데이터 수
};

```

3 – 출력화면 :

<기본 화면>

```

----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 -->

```

<추가>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 1

추가할 데이터 : 1
```

<추가 할 때 list가 차 있을 때>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 1

추가할 데이터 : 4

List is full.
```

<삭제>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 2

삭제할 데이터 : 5
```

<삭제 할 때 list가 비어 있을 때>

```
----- menu -----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 2

삭제할 데이터 : 1

List is empty.
```

<출력>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 3

※ Current List
1 3 2 6 5

```

<종료>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 4

프로그램을 종료합니다

```

4. 3번 문제를 기반으로 list를 오름차순으로 정렬하여라.

조건 1. 오름차순으로 정렬을 하기 위해서는 중복된 데이터가 list 안에 있으면 안된다.
 조건 2. list에 데이터를 추가하는 순간 정렬이 되어 있어야한다.

Example)

입력 순서 → 3 - 4 - 5 - 1 - 2

list 안 데이터 순서 → 1 - 2 - 3 - 4 - 5

4 - 출력화면 :

<중복 된 데이터가 있을 시>

```

---- menu ----
1. 리스트 추가
2. 리스트 삭제
3. 리스트 출력
4. 프로그램 종료

입력 --> 1

추가할 데이터 : 1

중복된 데이터가 존재

```

5. [사진관리 프로그램 작성을 위한 단계 5]

lab#9에서 작성한 프로그램을 보다 신속하게 event에 속한 사진들을 검색할 수 있도록 확장한다. 확장된 프로그램은 photoList가 아닌 사진들을 event별로 분류해 놓은 eventList를 이용하여 event에 속한 사진들을 검색한다.

1) eventType class를 다음과 같이 정의한다.

■ 멤버 변수:

- ◆ eventName: 이벤트 이름으로 고유한 값을 갖는다.
- ◆ memPhotoList: 해당 이벤트에 촬영한 사진들의 파일명(14자리 primary key)를 포함하며 vector를 이용하여 구현한다.
- ◆ numOfphoto: memPhotoList에 저장된 사진파일 수를 저장한다.

■ 멤버 함수:

- ◆ GetEventName(): 현 이벤트명을 리턴
- ◆ GetFileNameList(임시 리스트): memPhotoList에 있는 파일명(primary key)들을 임시 리스트에 리턴한다.
- ◆ AddFileName(string photoName): memPhotoList에 새 사진파일명을 추가하고 numOfphoto를 update 한다.
- ◆ 그 밖의 비교 연산자 overloading 함수들과 변수 초기화 함수들

2) Application class을 다음과 같이 확장한다.

■ 멤버 변수

- ◆ eventList: 이벤트별로 소속된 사진을 관리하는 리스트

■ 멤버 함수:

- ◆ AddEventToList(): eventList에 새 이벤트 추가
 - 이벤트가 리스트에 존재하면 해당 리스트에 파일을 추가하고, 존재하지 않으면 새로운 이벤트 노드를 리스트에 추가
- ◆ SearchEventList(): eventList에서 이벤트를 검색하여 index 리턴
- ◆ DisplayEventList(): [그림1]과 같이 Event list를 화면에 출력
- ◆ RetrieveFromEventList(): 이벤트명을 입력받아서 해당 이벤트에 소속된 모든 사진파일의 자세한 정보를 화면에 출력한다. SearchEventList()와 BinarySearchByPrimarykey()을 이용하여 구현
- ◆ DisplayEventPhotos(): [그림 2]와 같이 이벤트 사진 리스트에 있는 primary key를 이용하여 photoList를 검색하여 자세한 사진정보를 화면에 출력한다.

5 - 출력화면 :

<그림 1>

```

Choose a Command--> 4

***** [ Display on screen ] *****
Record 0 : 20180930160637 , 2016생일 , 23 , 이성규 김산 경북궁
Record 1 : 20180930160715 , 2016생일 , 32 , 김산 조영규 경북궁
Record 2 : 20180930160946 , 2018미국여행 , 23 , 이성규 정석 요세미티
Record 3 : aaa , 2017졸업식 , 33 , 이성규 김산 경희대국제캠퍼스
Record 4 : ddd , 2017졸업식 , 23 , 김강식 이성규 장일성 경희대국제캠퍼스
Record 5 : ffff , 2018미국여행 , 33 , 이성규 김산 금문교

--- ID ----- Command -----
| 1. Add a new record to list |
| 2. Read all records from disk |
| 3. Write all records to disk |
| 4. Display all record on screen |
| 5. Retrieve by photo name |
| 6. Delete by photo name |
| 7. Retrieve by event |
| 8. Retrieve by contents |
| 9. Display Event List |
| 10. Retrieve from event list |
| 0. Quit |
-----

Choose a Command--> 9

***** [ Display Event List ] *****
Event 0 2016생일
Members: 20180930160637 20180930160715

Event 1 2018미국여행
Members: 20180930160946 ffff

Event 2 2017졸업식
Members: aaa ddd

```

<그림 2>

```

| 9. Display Event List |
| 10. Retrieve from event list |
| 0. Quit |
-----

Choose a Command--> 10

***** [ Retrieve From Event List ] *****
Enter Event Name -->2016생일
Record : : 20180930160637 , 2016생일 , 23 , 이성규 김산 경북궁
Record : : 20180930160715 , 2016생일 , 32 , 김산 조영규 경북궁

```