

## LAB #10

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.

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

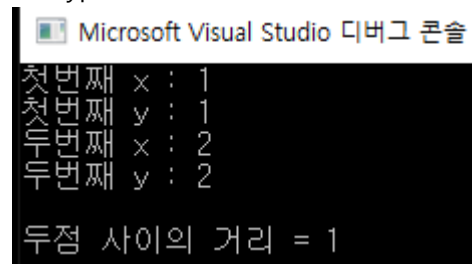
    // 두 점의 position을 입력 받는 함수
    p.setPointFromKeybord();

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

    return 0;
}
```

1 - 출력화면 :

<type이 int인 경우>

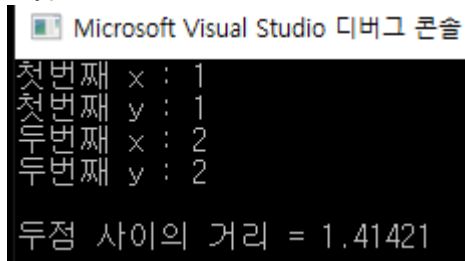


Microsoft Visual Studio 디버그 콘솔

```
첫번째 x : 1
첫번째 y : 1
두번째 x : 2
두번째 y : 2

두점 사이의 거리 = 1
```

<type이 double인 경우>

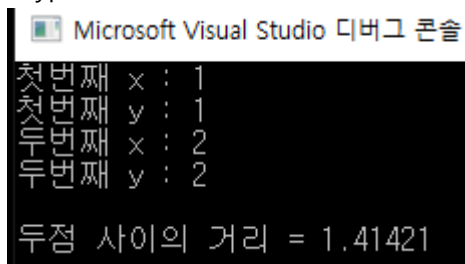


Microsoft Visual Studio 디버그 콘솔

```
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(곱의 최댓값)과 최솟값을 찾는다.
4. 이 때 vector의 데이터에 접근하기 위해서 iterator만을 사용한다.

2 – 출력화면 :

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

최댓값 = 180
최솟값 = 0
```

3. 아래 코드를 기반으로 다양한 type(int, double, float)을 사용하여 추가, 삭제, 출력 기능을 하는 List class를 구현하고 이를 사용하는 프로그램을 작성하라.

```
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 << "ㄴ삭제할 데이터 : ";
        cin >> input;
        list.Delete(input);
        break;
    case 3: // 출력
        list.Print();
        break;
    case 4: // 프로그램 종료
        cout << "ㄴㄴ프로그램을 종료합니다ㄴㄴ";
        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

```