과제 1번

1. 두 지점간의 거리를 **double** 변수에 저장하는 **distance**라는 기본 클래스를 생성하는 프로그램을 작성하여라. 그리고 그 거리를 지나가는 데 걸리는 시간을 출력하는 **trav_time( )**이라는 가상함수를 **distance**에 생성하여라. 여기서 거리는 마일(mile) 단위이고 속도는 시속 60마일이라고 가정한다. **metric**이라는 파생 클래스에서는 거리를 킬로미터 단위로, 속도를 시속 100킬로미터라고 가정했을 때의 걸리는 시간을 출력하도록 **trav_time( )**을 오버라이드하여라. 두 클래스의 객체를 생성한 후에 시간을 출력하는 main 함수를 작성하라.

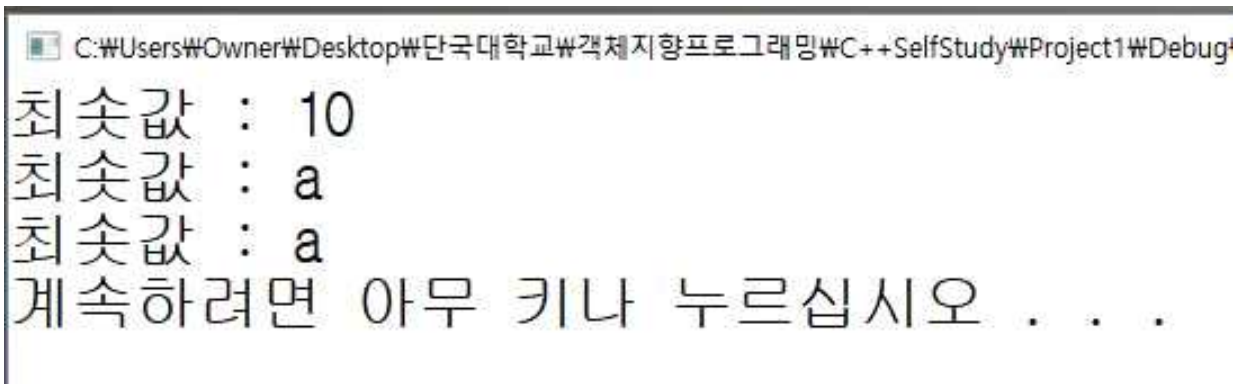
```
C:₩Users₩Owner₩Desktop₩단국대학교₩객체지향프로그래밍₩C++SelfStudy₩Project1₩Debug₩Project1.exe
두 지점의 좌표 입력 : 200 100
mile단위 & 시속 60mile일때 걸리는 시간 : 1.66667시간
kilometer단위 & 시속 100km일때 걸리는 시간 : 1시간
계속하려면 아무 키나 누르십시오 . . .
```

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

class ddistance {
public:
    double x, y; // 두 지점의 좌표
    double speed_m = 60; // 단위가 mile일 때의 속력

    double ddistance_m(double x, double y) {
        double dis; // 두 지점의 거리 변수

        if (x > y) { // x값과 y값중 어떤 값이 클지 모르기 때문에
                     // 두 값을 비교하여 큰값 - 작은값
            dis = x - y;
        }
        else {
            dis = y - x;
        }

        return dis; // 두 지점의 거리 반환
    }
    virtual double trav_time(double dis) {
        double time;
        time = dis / speed_m; // 시간을 구하기 위해 거리를 속력으로 나눠준다

        return time; // mile단위에서 걸리는 시간 반환
    }

};

class metric:public ddistance {
public:
    double speed_k = 100; // 단위가 km일 때의 속력

    double trav_time(double dis) {
        double time;
        time = dis / speed_k; // 시간을 구하기 위해 거리를 속력으로 나눠준다

        return time; // km단위에서 걸리는 시간 반환
    }
};

int main() {

    ddistance *p;
    ddistance ddis;
    metric me;

    double i, j;
    cout << "두 지점의 좌표 입력 : ";
    cin >> i; cin >> j;

    p = &ddis;
    cout << "mile단위 & 시속 60mile일때 걸리는 시간 : ";
    cout << p->trav_time(p->ddistance_m(i, j)) << "시간" << endl;

    p = &me;
    cout << "kilometer단위 & 시속 100km일때 걸리는 시간 : ";
    cout << p->trav_time(p->ddistance_m(i, j)) << "시간" << endl;

    return 0;
}
```

과제 2번

2. 두 인수 중 작은 값을 반환하는 min( )이라는 템플리트 함수를 작성하여라. 예를 들면, min(3, 4)는 3을 반환하고 min('c', 'a')는 a를 반환한다. 이 함수를 사용하는 main 함수를 작성하라.

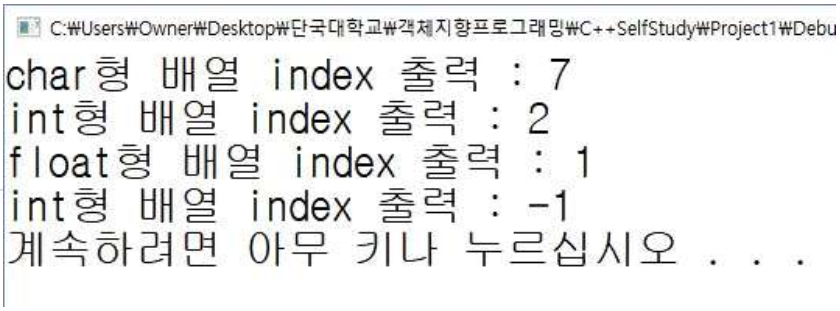
```cpp
#include <iostream>
using namespace std;

template<class type1, class type2>
void mmin(type1 a, type2 b) {
    // a와 b의 자료형이 다를 수 있기 때문에 변수를 2개 선언해준다
    if (a < b) {
        cout << "최솟값 : " << a << endl;
    }
    else {
        cout << "최솟값 : " << b << endl;
    }
}

int main() {
    mmin(10, 20); // int와 int의 비교
    mmin('a', 'b'); // char과 char의 비교
    mmin('a', 100); // char과 int의 비교

    return 0;
}
```



과제 3번

3. 템플리트 함수에 대한 예로 **find( )**라는 함수를 작성하여라. 이 함수는 값을 찾기 위해 배열을 탐색한다. 그 값이 있으면 그것의 첨자를 반환하고, 없다면 -1을 반환한다. 다음은 **find( )**의 int 형 버전을 위한 원형이다. **find( )**를 템플리트 함수로 작성하고 main 함수에서 int, float, char 형에 대해 실행해 보아라.(**size** 매개변수는 배열에 있는 원소의 수를 나타낸다)

```
int find (int object, int *list, int size)
{
  // ...
}
```




```cpp
#include <iostream>
#include <Windows.h>
using namespace std;

template<class type>
type find(type object, type *list, int size) {

    for (int i = 0; i < size; i++) {
        if (object == list[i]) {
            return i; // 값이 배열에 있을 경우 배열의 인덱스를 반환
        }
    }

    return -1; // 값이 배열에 없을 경우 -1 반환
}

void main() {
    int index; // 인덱스의 값을 저장하기 위한 변수
    int arr_i[10] = { 1,2,3,4,5,6,7,8,9,10 };
    float arr_f[5] = { 1.2, 3.5, 5.3, 8.2, 4.4 };
    char arr_c[] = "lllllllIll";
    int k = strlen(arr_c); // arr_c의 길이를 저장하기 위한 변수

    index = find('I', arr_c, k); // 크기가 k인 arr_c에서 'I'를 찾는다
    cout << "char형 배열 index 출력 : " << index << endl;

    index = find(3, arr_i, 10); // 크기가 10인 arr_i에서 3를 찾는다
    cout << "int형 배열 index 출력 : " << index << endl;

    index = find((float)3.5, arr_f, 5); // 크기가 5인 arr_f에서 3.5를 찾는다
    cout << "float형 배열 index 출력 : " << index << endl;

    index = find(11, arr_i, 10); // 크기가 10인 arr_i에서 11를 찾는다
    cout << "int형 배열 index 출력 : " << index << endl; // 값이 배열에 없는 경우

    system("pause");
}
```