#include "stdafx.h"

#include <iostream>

using namespace std;

/\* -------------------- 과제4-1 START --------------------

1. 두 지점간의 거리를 double 변수에 저장하는 myDistance라는

기본 클래스를 생성하는 프로그램을 작성하여라.

그리고 그 거리를 지나가는 데 걸리는 시간을 출력하는 trav\_time()이라는

가상함수를 myDistance에 생성하여라.

여기서 거리는 마일(mile) 단위이고 속도는 시속 60마일이라고 가정한다.

metric이라는 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.

\*/

//distance 라는 이름이 예약어임(사용 불가)

class myDistance {

protected :

double length;

public :

//생성자 : 두 지점 사이의 거리를 구한다.

myDistance(double a, double b) {

length = a - b;

(length < 0) && (length = -length);

}

//가상 함수로 지정 - 자식 클래스에서 재정의 가능

virtual void trav\_time() {

cout << "60mile/h로 " << length << "mile 을 가는데 걸리는 시간(Time) : " << (length / 60) << " hour " << endl;

}

};

//myDistance를 상속한 metric 클랙스 정의

class metric : public myDistance {

public:

//부모 생성자를 상속받음.

metric(double a, double b) : myDistance(a, b) {}

//부모의 객체의 trav\_time을 재정의

void trav\_time() {

cout << "100km/h로 " << length << "km 를 가는데 걸리는 시간(Time) : " << (length / 100) << " hour " << endl;

}

};

//main 함수에서 호출될 homework1 함수

void homework1() {

//두 지점을 입력받는다.

double a, b;

cout << "숫자 2개를 입력해 주세요 : ";

cin >> a; cin >> b;

//myDistance 객체와 metric 객체 생성

myDistance mdt(a, b);

metric mt(a, b);

//거리에 대한 시간 출력

mdt.trav\_time();

mt.trav\_time();

}

/\*-------------------- 과제4-1 END --------------------\*/

/\*-------------------- 과제4-2 START --------------------

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.

\*/

//int 형에 대한 min 함수

int min(int a, int b) {

if (a > b) return a;

else if (a < b) return b;

}

//char 형에 대한 min 함수

char min(char a, char b) {

if (a > b) return a;

else if (a < b) return b;

}

//main 함수에서 호출될 homework2 함수

void homework2() {

cout << " 3, 4 : " << min(3, 4) << endl;

cout << " a, b : " << min('a', 'b') << endl;

}

/\*-------------------- 과제 4-2 END --------------------\*/

/\*-------------------- 과제 4-3 START --------------------

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.

int find (int object, int \*list, int size);

find()를 템플리트 함수로 작성하고 main 함수에서 int, float, char 형에 대해 실행해 보아라.

(size 매개변수는 배열에 있는 원소의 수를 나타낸다)

\*/

//find 함수 정의

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

for (int i = 0; i < size; i++) {

//반복 중 일치하는 요소를 찾으면 바로 return. 반복문 봉료

if (list[i] == object) return i;

}

//반복 종료 후 일치하는 요소 없으면 -1 반환

return -1;

}

//main에서 호출될 homework3 정의

void homework3() {

int list\_int[] = { 1,2,3,4,5,6,7 };

char list\_char[] = "abcdefghijk";

float list\_float[] = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };

cout << "1234567에서 3의 위치 : " << find(3, list\_int, 7) << endl;

cout << "1234567에서 0의 위치 : " << find(0, list\_int, 7) << endl;

cout << "abcdefghijk에서 d의 위치 : " << find('d', list\_char, 11) << endl;

cout << "abcdefghijk에서 z의 위치 : " << find('z', list\_char, 11) << endl;

cout << "1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7에서 2.2의 위치 : " << find((float)2.2, list\_float, 7) << endl;

cout << "1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7에서 0.0의 위치 : " << find((float)0.0, list\_float, 7) << endl;

}

/\*-------------------- 과제4-3 END --------------------\*/

//번호를 입력받아 3개의 프로그램 중 번호에 해당하는 프로그램을 호출

int main(){

int type;

cout << "과제4 번호 : ";

cin >> type;

// 번호를 입력 받아 해당 숙제에 대한 프로그램을 호출

switch (type) {

case 1: homework1(); break;

case 2: homework2(); break;

case 3: homework3(); break;

default: return 0; break;

}

cout << endl;

main();

return 0;

}

