```cpp
// W8_1_0.cpp : Defines the entry point for the console application.
//              Przeciazenie operatorow == >  <  &&  ||

#include "stdafx.h"
#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;

class coord
{
    double x;
    double y;
public:
    coord(double xx, double yy) { x = xx; y = yy; }
    coord() { x = 0.0;  y = 0.0; }
    void set(double xx, double yy) { x = xx; y = yy; }
    coord get() { return *this; }
    int operator == (const coord &praw) const;
    int operator > (const coord &praw) const;
    int operator && (coord &praw);
    int operator || (coord &praw);
    void disp(char *tit) { cout << tit << ": " << x << "  " << y << endl; }
    void disp_len() {
        double len = sqrt(x*x+y*y);
        cout << "E2 = " << len << endl;
    }
};

int fun_comp_ascending(const void * ele1, const void * ele2)
/*=============================================================
funkcja porownawcza dla sortowania rosnacego
=============================================================*/
{
    const coord *e1 = (const coord *)ele1;
    const coord *e2 = (const coord *)ele2;

    if(*e1 > *e2)
        return 1;
    else if (*e1 == *e2)
        return 0;
    else
        return -1;
}

int fun_comp_descending(const void * ele1, const void * ele2)
/*=============================================================
funkcja porownawcza dla sortowania malujacego
=============================================================*/
{
    const coord *e1 = (const coord *)ele1;
    const coord *e2 = (const coord *)ele2;

    if(*e1 > *e2)
        return -1;
    else if (*e1 == *e2)
        return 0;
    else
        return 1;
}

int coord::operator == (const coord &praw) const
{
    return (x == praw.x && y == praw.y);
}

int coord::operator > (const coord &praw) const
{
    double ro_lew, ro_praw;
    ro_lew  = x*x+y*y;
    ro_praw = praw.x*praw.x+praw.y*praw.y;
    return (ro_lew > ro_praw);
}
```

```
int coord::operator && (coord &praw)
{
    return ((x && praw.x) && (y && praw.y));
}

int coord::operator || (coord &praw)
{
    return ((x || praw.x) || (y || praw.y));
}

int _tmain(int argc, _TCHAR* argv[])
{
    coord ob1(1, 2), ob2(3, 4), ob3(1, 2), ob4(1, 1);
    ob1.disp("ob1");
    ob2.disp("ob2");
    ob3.disp("ob3");
    ob4.disp("ob4");

    //test 1: operator ==
    cout << "ob1 == ob2    ob1 == ob3   ob1 == ob4" << endl;
    cout << (ob1 == ob2) << " " << (ob1 == ob3) << " " << (ob1 == ob4) << endl;

    //test 2: operator >
    cout << "ob1 > ob2    ob1 > ob3   ob1 > ob4" << endl;
    cout << (ob1 > ob2) << " " << (ob1 > ob3) << " " << (ob1 > ob4) << endl;

    //test 3: operator &&
    coord ob5, ob6, ob7(0, 1);
    ob5.disp("ob5");
    ob6.disp("ob6");
    ob7.disp("ob7");
    cout << "ob7 && ob5    ob6 && ob5   ob7 && ob4  ob7 && ob1  ob1 && ob2" << endl;
    cout << (ob7 && ob5) << " " << (ob6 && ob5) << " " << (ob7 && ob4);
    cout << " " << (ob7 && ob1) << " " << (ob1 && ob2) << endl;

    //test 4: operator ||
    cout << "ob7 || ob5    ob6 || ob5   ob7 || ob4  ob4 || ob5" << endl;
    cout << (ob7 || ob5) << " " << (ob6 || ob5) << " " << (ob7 || ob4);
    cout << " " << (ob4 || ob5) << endl;

    //sortowanie w kolejności rosnacej & malujacej
    coord ptab[] = {
        ob1,
        ob2,
        ob3,
        ob4,
        ob5,
        ob6,
        ob7
    };

    //tablica wskaznikow do funkcji porownawczych
    typedef int (*PTR_FUN_COMP)(const void *, const void *);
    PTR_FUN_COMP ptr_fun_comp_tab[] = {
        fun_comp_ascending,        //sortowanie rostnace
        fun_comp_descending        //sortowanie malujace
    };

    //ilosc elementow w tablice
    size_t num = sizeof(ptab)/sizeof(ptab[0]);

    //do sortowania
    cout << "do sortowania\n";
    for(size_t it=0; it<num; ++it)
        ptab[it].disp_len();

    //void qsort(
    //            void *base,   //wskaznik do elementu tablicy, poczynajac z ktorego      ↙
    sortujemy elementy
    //            size_t num,   //ilosc elementow do sortowania
    //            size_t width, //rozmiar w bajtach jednego elementu
    //            int (__cdecl *compare )(const void *, const void *) //funkcja          ↙
    porownawcza
    //);
```

```
    //sortowanie rosnace
    qsort((void *)ptab, num, sizeof(ptab[0]), ptr_fun_comp_tab[0]);

    //po sortowaniu
    cout << "po sortowaniu rosnancemu\n";
    for(size_t it=0; it<num; ++it)
        ptab[it].disp_len();

    //sortowanie malujace
    qsort((void *)ptab, num, sizeof(ptab[0]), ptr_fun_comp_tab[1]);

    //po sortowaniu
    cout << "po sortowaniu malujacemu\n";
    for(size_t it=0; it<num; ++it)
        ptab[it].disp_len();

    system("pause");
    return 0;
}
```