```
// Schildt_7_Usw_3.cpp : Defines the entry point for the console application.
//    Przeciazenie operatorow. Dziedziczenie operatorow, przeciazonych w klasie bazowej

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

#define WARIANT_1

class coord {
public:
    int x, y; // wartosci wspolrzednych
public:
    coord() { x = 0; y= 0; }
    coord(int i, int j) { x = i; y = j; }
    void get_xy(int &i, int &j) { i = x; j = y; }
    coord operator+(coord ob2);
      coord operator-(coord ob2);
    coord operator=(coord ob2);
};

//Przeciazenie operatora +
coord coord::operator+(coord ob2)
{
      coord temp;
      cout << "operator+()\n";
      temp.x = x+ob2.x;
      temp.y = y+ob2.y;
    return temp;
}

//Przeciazenie operatora -
coord coord::operator-(coord ob2)
{
      coord temp;
      cout << "operator-()\n";
      temp.x = x-ob2.x;
      temp.y = y-ob2.y;
    return temp;
}

//Przeciazenie operatora =
```

```
coord coord::operator= (coord ob2)
{
      cout << "class coord: operator=()\n";
      x = ob2.x;
      y = ob2.y;
    return *this;
}

class quad : public coord
{
      int quadrant;
public:
      quad() {x = 0; y = 0; quadrant = 0;}
      quad(int x, int y): coord(x, y)
      {
            if(x >= 0  && y >= 0) quadrant = 1;
            else if(x<0 && y>=0) quadrant = 2;
            else if(x<0 && y<0) quadrant = 3;
            else quadrant = 4;
      }

      void showq()
      {
            cout << "Punkt w kwadrancie: " << quadrant << "\n";
      }

#ifdef WARIANT_1
      quad operator=(coord ob2);
#else
      quad operator=(quad &ob2);
      quad operator+(const quad &ob);
      quad operator-(const quad &ob);
#endif
};

#ifdef WARIANT_1
quad quad::operator=(coord ob2)
{
      cout << "class quad: operator=(coord ob)\n";
      x = ob2.x;
      y = ob2.y;
      if(x >= 0  && y >= 0) quadrant = 1;
```

```
      else if(x<0 && y>=0) quadrant = 2;
      else if(x<0 && y<0) quadrant = 3;
      else quadrant = 4;

      return *this;
}
#else
quad quad::operator=(quad &ob2)
{
      cout << "class quad: operator=(quad ob)\n";
      x = ob2.x;
      y = ob2.y;
      quadrant = ob2.quadrant;
      return *this;
}

quad quad::operator+(const quad &ob)
{
      cout << "class quad: operator+(quad ob)\n";
      quad ret;
      ret.x = x+ob.x;
      ret.y = y+ob.y;
      if(ret.x >= 0  && ret.y >= 0) ret.quadrant = 1;
      else if(ret.x<0 && ret.y>=0) ret.quadrant = 2;
      else if(ret.x<0 && ret.y<0) ret.quadrant = 3;
      else ret.quadrant = 4;
      return ret;
}

quad quad::operator-(const quad &ob)
{
      cout << "class quad: operator+(quad ob)\n";
      quad ret;
      ret.x = x-ob.x;
      ret.y = y-ob.y;
      if(ret.x >= 0  && ret.y >= 0) ret.quadrant = 1;
      else if(ret.x<0 && ret.y>=0) ret.quadrant = 2;
      else if(ret.x<0 && ret.y<0) ret.quadrant = 3;
      else ret.quadrant = 4;
      return ret;
}
#endif
```

```
int main()
{
    quad o1(10, 10), o2(15, 3), o3;
      int x, y;

      //dodawanie dwoch obiektow -> operator +()
      //o1, o2 - ma typ quad, przeciez operator+() zwraca typ coord.
      //wiec (o1+o2) - typ coord, dla tego drugi operand operatora quad::operator=()
      //ma typ coord, a zwraca typ quad - to kompletnie odpowiada typu o3.
      //operator=() nie podlega dziedziczeniu, wiec trzeba przeciazyc w klasie quad operator=()
      o3 = o1+o2;

      o3.get_xy(x, y);
      o3.showq();

      cout << "(o1+o2) X: " << x << " Y: " << y << "\n";

      //odejmowanie dwoch obiektow -> operator -()
      o3 = o1-o2;

      o3.get_xy(x, y);
      o3.showq();

      cout << "(o1-o2) X: " << x << " Y: " << y << "\n";

      //przypisanie obiektow
      //Obiekty o1, o3 maja typ quad, przeciez quad::operator=() dla drugiego operandu ma typ coord,
      //wystepuje uciecie typu klasy pochodnej do typu klasy bazowej - coord::operator=()
      //skladowa quadrant kopiuje sie domyslnie!

      //Jesli odblokowac przeciazona funkcje-operator quad::operator=(),
      //bedzie wywolana wlasnie ta funkcje-operator, a nie funkcje-operator klasy bazowej
      o3 = o1;
      o3.get_xy(x, y);
      o3.showq();

      cout << "(o3=o1) X: " << x << " Y: " << y << "\n";

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