Output

Klasa Punkt

```
class Punkt
{
private:
     int x;
     int y;
public:
     Punkt();
     Punkt(int x, int y);
     int get_x() const;
     int get_y() const;
     string to_string();
};
```

```
Punkt::Punkt()
{
     x = 0;
     y = 0;
}

Punkt::Punkt(int x0, int y0)
{
     x = x0;
     y = y0;
}

int Punkt::get_x() const
{
     return x;
}

int Punkt::get_y() const
{
     return y;
}
```

Rozwiązanie w duchu C++

```
ostream& operator<<(ostream& os, const Punkt& punkt)
{
    os << "(" << punkt.get_x() << ',' << punkt.get_y() << ")";
    return os;
}

int main()
{
    Punkt p(3,4);
    cout << p << endl;
}
```

Rozwiązanie alternatywne – to_string()

```
string Punkt::to_string()
{
    return  "(" + ::to_string(x) + "," + ::to_string(y) + ")";
}

int main()
{
    Punkt p(3,4);
    cout << p.to_string() << endl;
}
```

Klasa Kolo

```
class Kolo
{
private:
    Punkt srodek;
    int promien;
public:
    Kolo();
    Kolo(const Punkt& srodek, int promien);
    Punkt get_srodek() const;
    int get_promien() const;
    string to_string();
};
```

```
Kolo::Kolo()
{
    srodek = Punkt(0,0);
    promien = 1;
}

Kolo::Kolo(const Punkt& srodek0,
int promien0)
{
    srodek = srodek0;
    promien = promien0;
}

Punkt Kolo::get_srodek() const
{
    return srodek;
}

int Kolo::get_promien() const
{
    return promien;
}
```

Rozwiązanie w duchu C++

```
ostream& operator<<(ostream& os, const Kolo& kolo)
{
    os << "Kolo(" << kolo.get_srodek() << ',' << kolo.get_promien() << ")";
    return os;
}

int main()
{
    Punkt p(3,4);
    Kolo k(p, 3);
    cout << k << endl;
}
```

Rozwiązanie alternatywne – to_string()

```
string Kolo::to_string()
{
    return "Kolo(" + srodek.to_string() + "," + ::to_string(promien) + ")";
}

int main()
{
    Punkt p(3,4);
    Kolo k(p, 3);
    cout << k.to_string() << endl;
}
```

Możemy mieć kilka wersji to_string()

- Może się przydać np. w zadaniu: Wyświetlanie dat

```
2018-12-25
25 grudnia 2018 r.
25/12/2018
```