Listy inicjalizacyjne konstruktorów

Konstruktory wykorzystuje się najczęściej do zainicjalizowania pól klasy. Przykładowo:

```
class Liczba
{
  private:
    int wartosc;

  public:
    Liczba(int w)
    {
      wartosc = w;
    }
};
```

Identyczny efekt można uzyskać przy użyciu tzw. **listy inicjalizacyjnej** (która z definicji służy do inicjalizacji składowych klasy). Zwróćcie uwagę na dwukropek po nagłówku konstruktora:

```
class Liczba
{
  private:
    int wartosc;

  public:
    Liczba(int w):
      wartosc(w)
    {
    }
};
```

Na liście inicjalizacyjnej można przypisać początkowe wartości dowolnej liczbie pól (inicjalizacje kolejnych pól oddziela się przecinkami):

```
class Punkt
{
  private:
    int x;
    int y;

  public:
    Punkt(int wx, wy):
      x(wx), y(wy)
    {
    }
};
```

Istnieje kilka przypadków, w których inicjalizacja musi zostać wykonana na liście inicjalizacyjnej (bo inaczej się nie da).

Inicjalizacja stałej (próba „zwykłego” przypisania wartości w konstruktorze nie skompiluje się):

```
class LiczbaStala
{
  private:
    const int wartosc;

  public:
    LiczbaStala(int w):
      wartosc(w)
    {
    }
};
```

Inicjalizacja obiektu – użycie konstruktora z argumentami:

```
class Kalkulator
{
  private:
    Liczba liczba;

  public:
    Kalkulator():
      liczba(123)
    {
    }
};
```

I najciekawszy przypadek – wywołanie konstruktora klasy bazowej. Czasami klasa bazowa nie ma konstruktora bezargumentowego, więc utworzenie obiektu wymaga użycia konstruktora z argumentami (i przekazania mu tych argumentów). Robi się to właśnie przy użyciu list inicjalizacyjnych:

```
class Bazowa
{
  public:
    Bazowa(int a, int b, int c)
    {
    }
};


class Pochodna : public Bazowa
{
  public:
    Pochodna():
      Bazowa(1, 2, 3)
    {
    }
};
```

Co ciekawe, listy inicjalizacyjne rozwiązują problem konfliktu między nazwami pól i argumentów konstruktura. Poniższy kod zadziała tak, jak należałoby tego oczekiwać – pole zostanie zainicjalizowane wartością argumentu:

```
class Liczba
{
  private:
    int wartosc;

  public:
    Liczba(int wartosc):
      wartosc(wartosc)
    {
    }
};
```