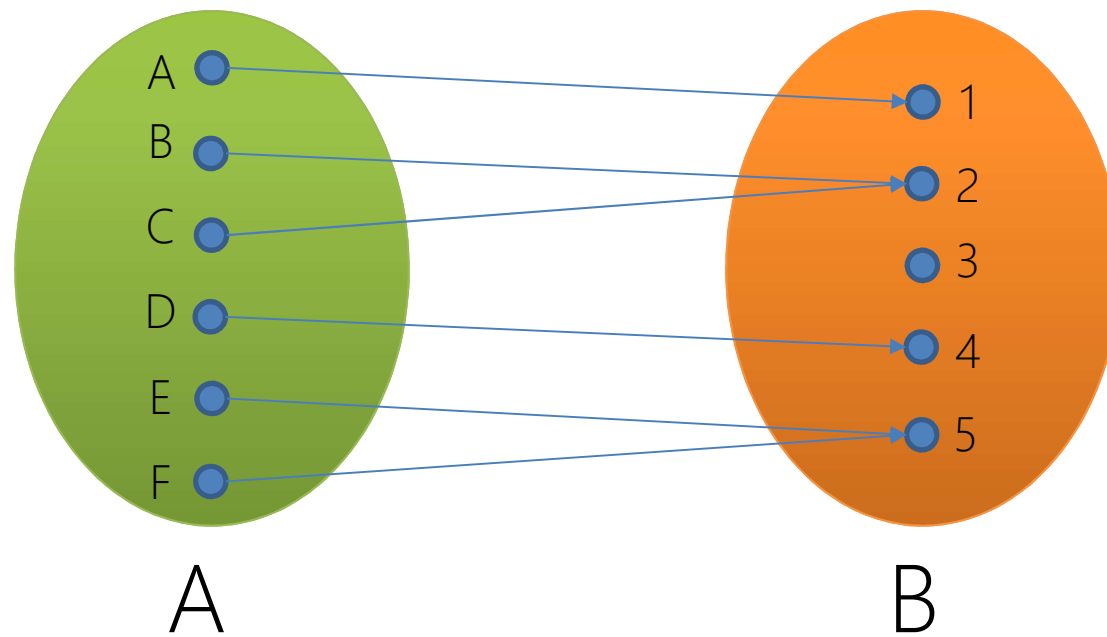


PROGRAMOWANIE FUNKCYJNE

Programowanie funkcyjne jest
to programowanie
z funkcjami matematycznymi.

Funkcje



$$x \rightarrow x^2 + 2x + 3$$

Funkcje

$$f: x \rightarrow x^2 + 2x + 3$$

$$f: A \rightarrow B$$

$$f: \mathbb{R} \rightarrow \mathbb{R}$$

Funkcje

Funkcje matematyczne są funkcjami czystymi

Wynik zależy tylko od wartości parametrów funkcji

$$f(3) \rightarrow 3^2 + 2 \cdot 3 + 3 \rightarrow 18$$

Funkcje

Dzięki czystości funkcje możemy zastąpić wartością,
a wartość aplikacją funkcji

Referencyjna transparentność

$$f(3) \rightarrow 3^2 + 2 \cdot 3 + 3 \rightarrow 18$$

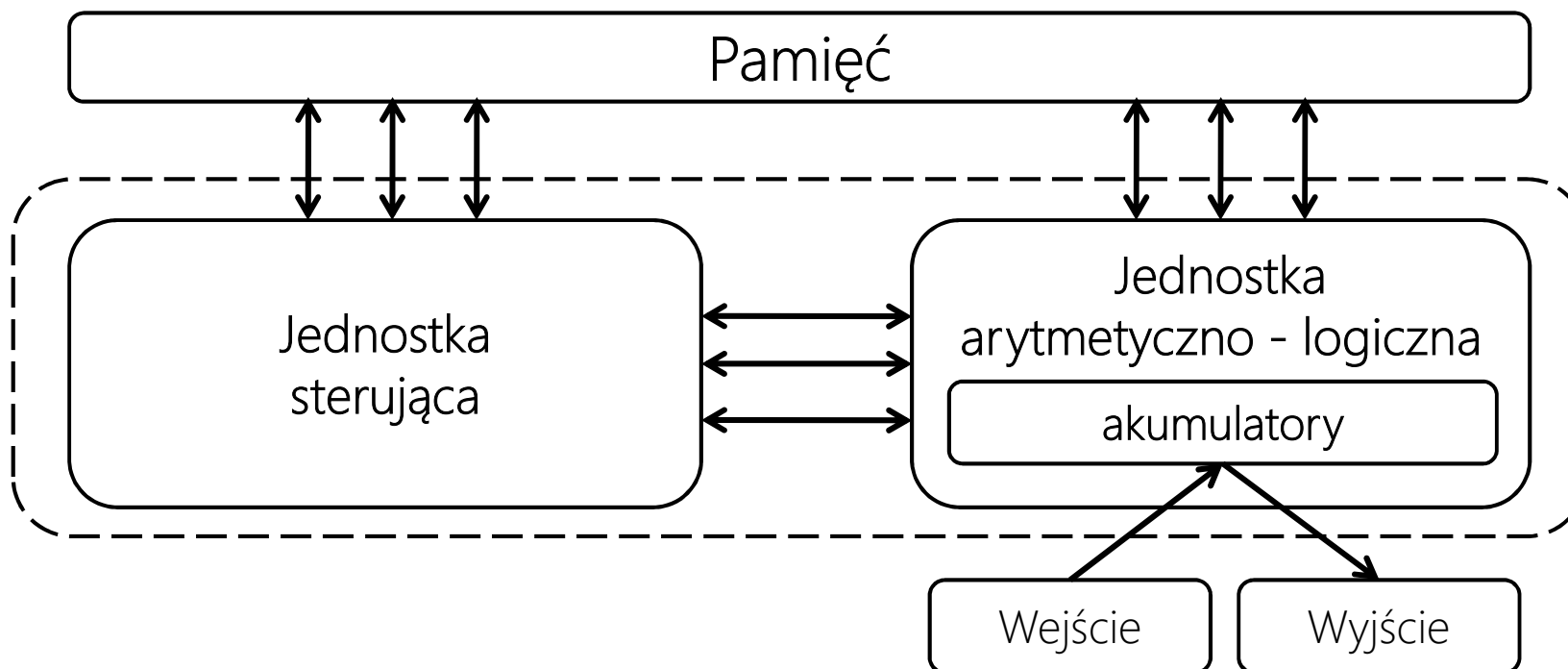
Funkcje

Funkcje mogą mieć więcej niż jeden parametr

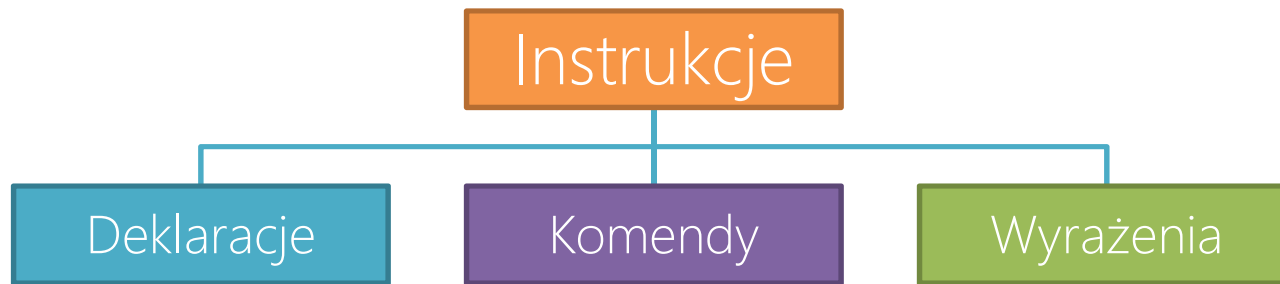
$$f: x, y \rightarrow x^2 + 2y + 3$$

$$f: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

Komputery mają pamięć



Rodzaje instrukcji



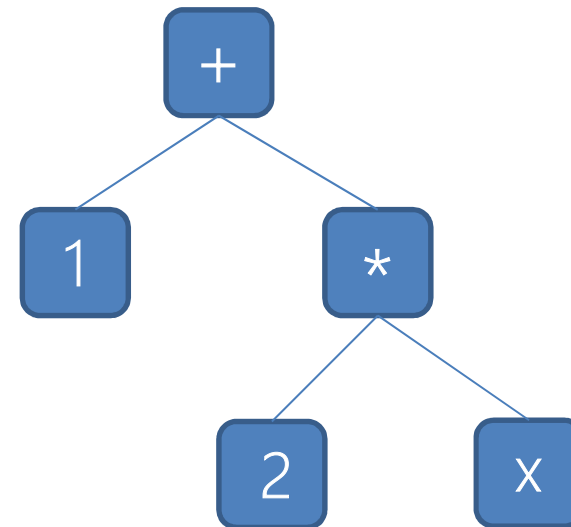
```
var x = 12;  
Console.WriteLine(x);
```

Nie wszystkie języki mają wszystkie rodzaje instrukcji

Instrukcje złożone



```
var x = 12;  
Console.WriteLine(1+2*x);  
Console.ReadKey();
```



Abstrakcje w programowaniu

Procedury

```
void Metoda(int x)
{
    if(x>0)
        Console.WriteLine("Dodatnia");
    else
        Console.WriteLine("Ujemna")
}
```

Funkcje

```
double Oblicz(double x)
{
    return 1+2*x;
}
```

Komendy i wyrażenia

Komendy

- Modyfikują stan programu
- Ich działanie można obserwować tylko poprzez analizę stanu

Wyrażenia

- Tworzą wartości
- Ich rezultat można obserwować poprzez analizę wartości zwracanej

Czyste funkcje

Są to funkcje, które nie mają efektów ubocznych, a ich wynik zależy tylko od wartości parametrów



```
double Oblicz(double x)
{
    return 1+2*x;
}
```

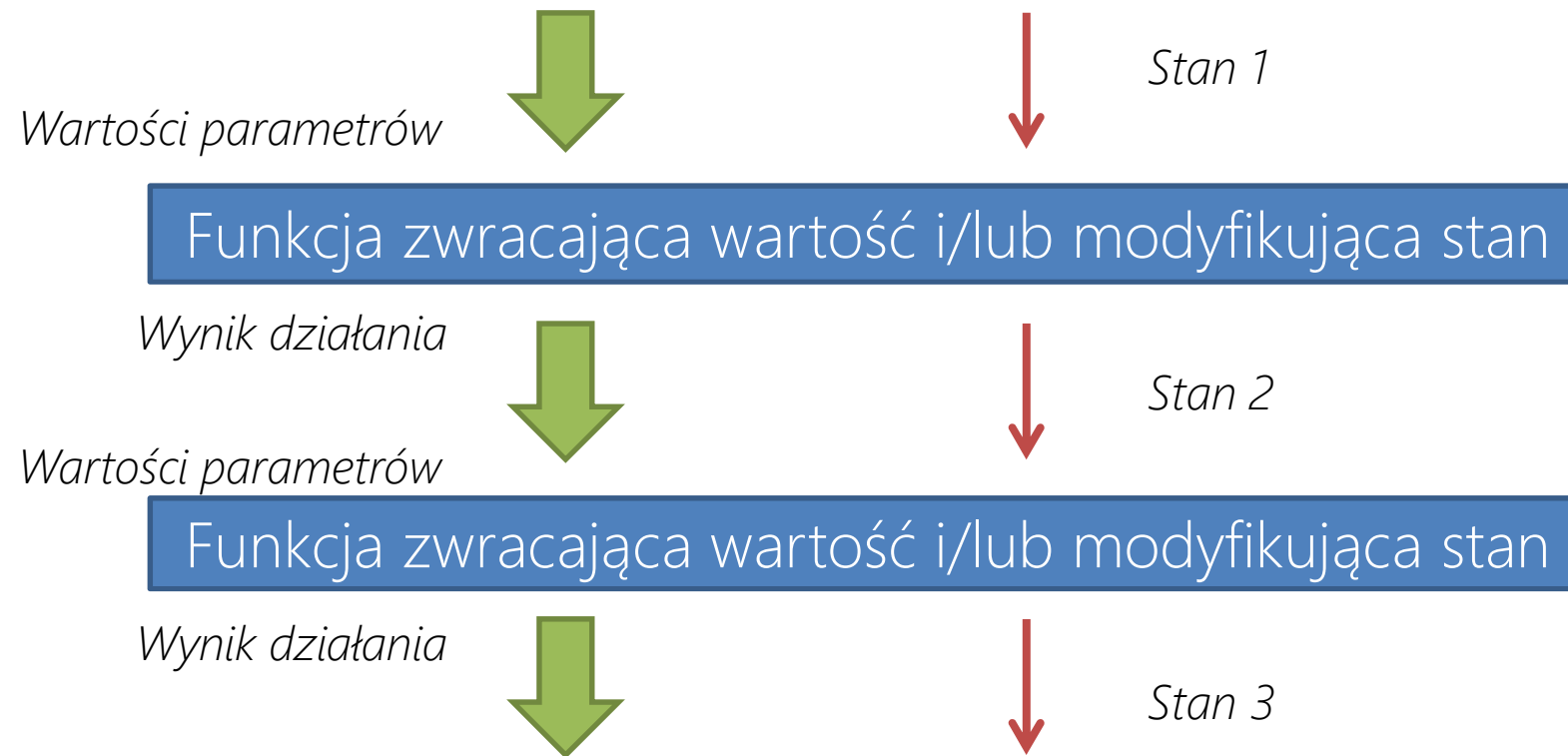
Efekty uboczne

Efekt uboczny – wszelkie działania, które dana instrukcja wykonuje, a które nie są wykonywane przez instrukcje z grupy do której ta instrukcja należy.

```
int x,y;  
x = y = 12;
```

```
class Przyklad {  
    private int a = 0;  
  
    public int f(int b, int c) {  
        int rezultat = a + b + c;  
        a += b + c;  
        return rezultat;  
    }  
}
```

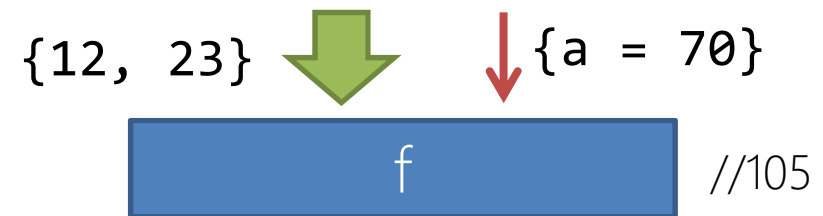
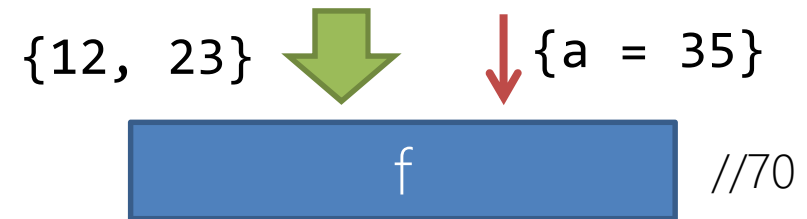
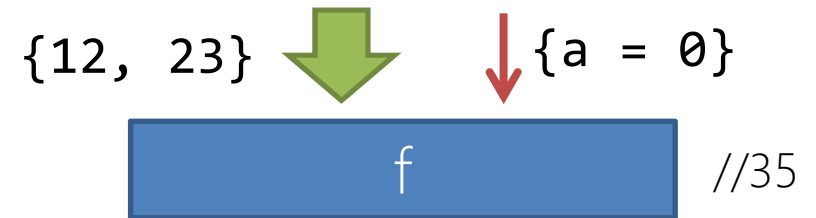
Efekty uboczne



Analiza programu

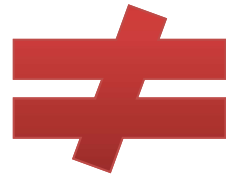
```
var przyklad = new Przyklad();  
Console.WriteLine(przyklad.f(12, 23));  
Console.WriteLine(przyklad.f(12, 23));  
Console.WriteLine(przyklad.f(12, 23));
```

```
class Przykład {  
    private int a = 0;  
  
    public int f(int b, int c) {  
        int rezultat = a + b + c;  
        a += b + c;  
        return rezultat;  
    }  
}
```



Analiza programu

$f(12, 23) + f(34, 45)$



$f(34, 45) + f(12, 23)$

Wartości niemodyfikowalne

Wartości niemodyfikowalne są to wartości, których składowe mogą być ustalone tylko w konstruktorze.

Wartości niemodyfikowalne



Zwiększają czytelność kodu



W pojedynczym punkcie należy sprawdzać warunki poprawności



Ułatwiają tworzenie aplikacji wielowątkowych



Wymagają większej ilości pamięci



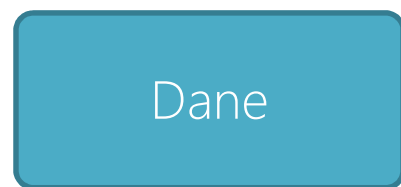
Zwiększają zużycie procesora

Funkcje wyższych rzędów

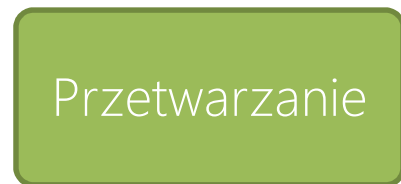
Funkcje wyższych rzędów, są to funkcje które przyjmują inne funkcje jako swoje parametry, lub tworzą nowe funkcje

$$\frac{df}{dx} = \frac{d(x^2 + 2x + 3)}{dx} = 2x + 2$$

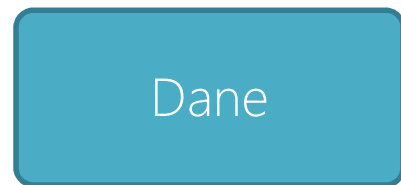
Tworzenie aplikacji w stylu funkcyjnym



Co mam dane?



W jaki sposób?



Co chcę uzyskać?

Operacje wejściowe
(ze skutkami ubocznymi)



Rdzeń aplikacji
(czysty)



Operacje wyjściowe
(ze skutkami ubocznymi)



Programowanie funkcyjne

