

Paradygmaty programowania

dr inż. Łukasz Bartczuk

Instytut Inteligentnych Systemów Informatycznych
p. 517

Lukasz.Bartczuk@iisi.pcz.pl

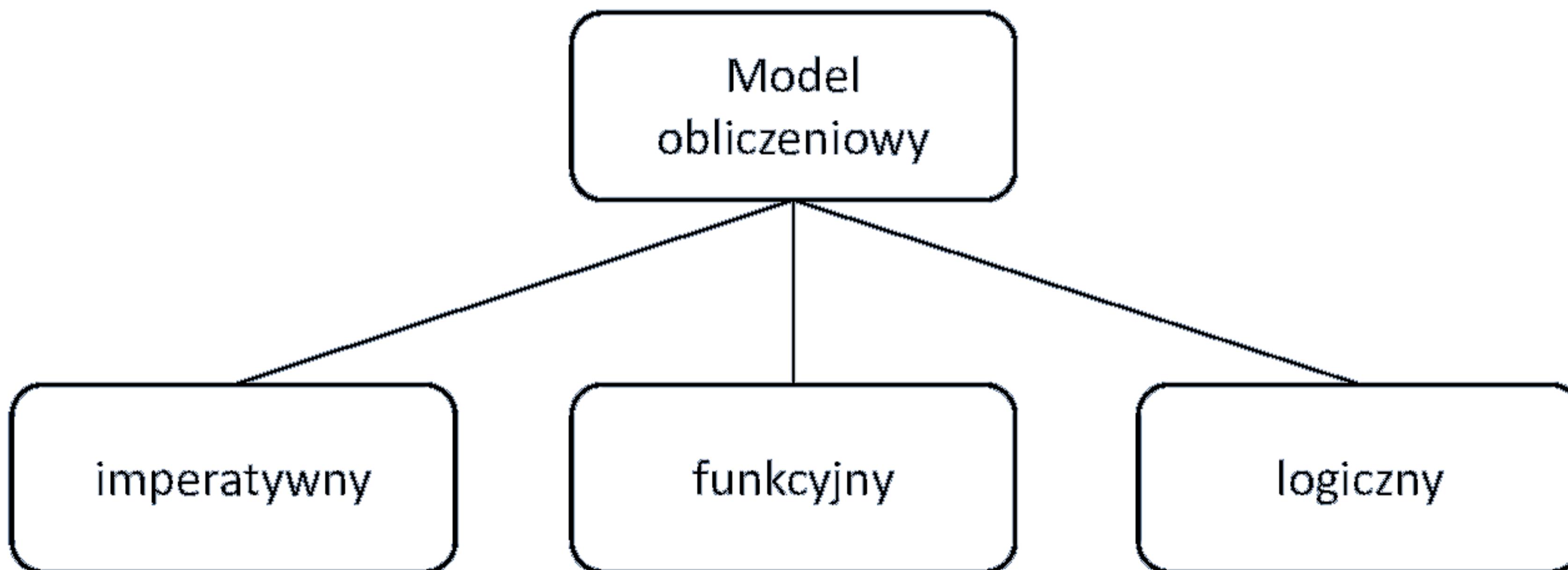
iisi.pcz.pl/lb/pp

Co to jest komputer?

Co to jest program
komputerowy?

Model obliczeniowy

Model obliczeniowy jest to zbiór wartości (prostych lub złożonych), skojarzonych z nimi operacji oraz operacji wykorzystywanych do zdefiniowania obliczeń.



Model funkcyjny

Model ten składa się z wartości i funkcji (przy czym funkcje też są wartością), a podstawową operacją do wykonywania obliczeń jest aplikacja funkcji do wartości.

$$f(x) = x^2 + x + 5$$

$$f(2) \rightarrow 2^2 + 2 + 5 \rightarrow 11$$

$$f(\sin(y)) \rightarrow \sin(y)^2 + \sin(y) + 5$$

$$\rightarrow g(y) = \sin(y)^2 + \sin(y) + 5$$

Model logiczny

Model ten składa się z faktów, relacji i zapytań, a metodą dokonywania obliczeń jest wnioskowanie logiczne.

1. $\text{student}(\text{Tomek})$.
2. $\text{LubiParadygmaty}(X)$ jeżeli $\text{student}(X)$.
3. $\neg \text{LubiParadygmaty}(Y)$.

4. $\neg \text{student}(Y)$.

5. $Y = \text{Tomek}$.

Model imperatywny

Model imperatywny składa się wartości oraz stanu i operacji przypisania.

$$\{x=2, y=3\}$$

$$y := x^2 + x + 5$$

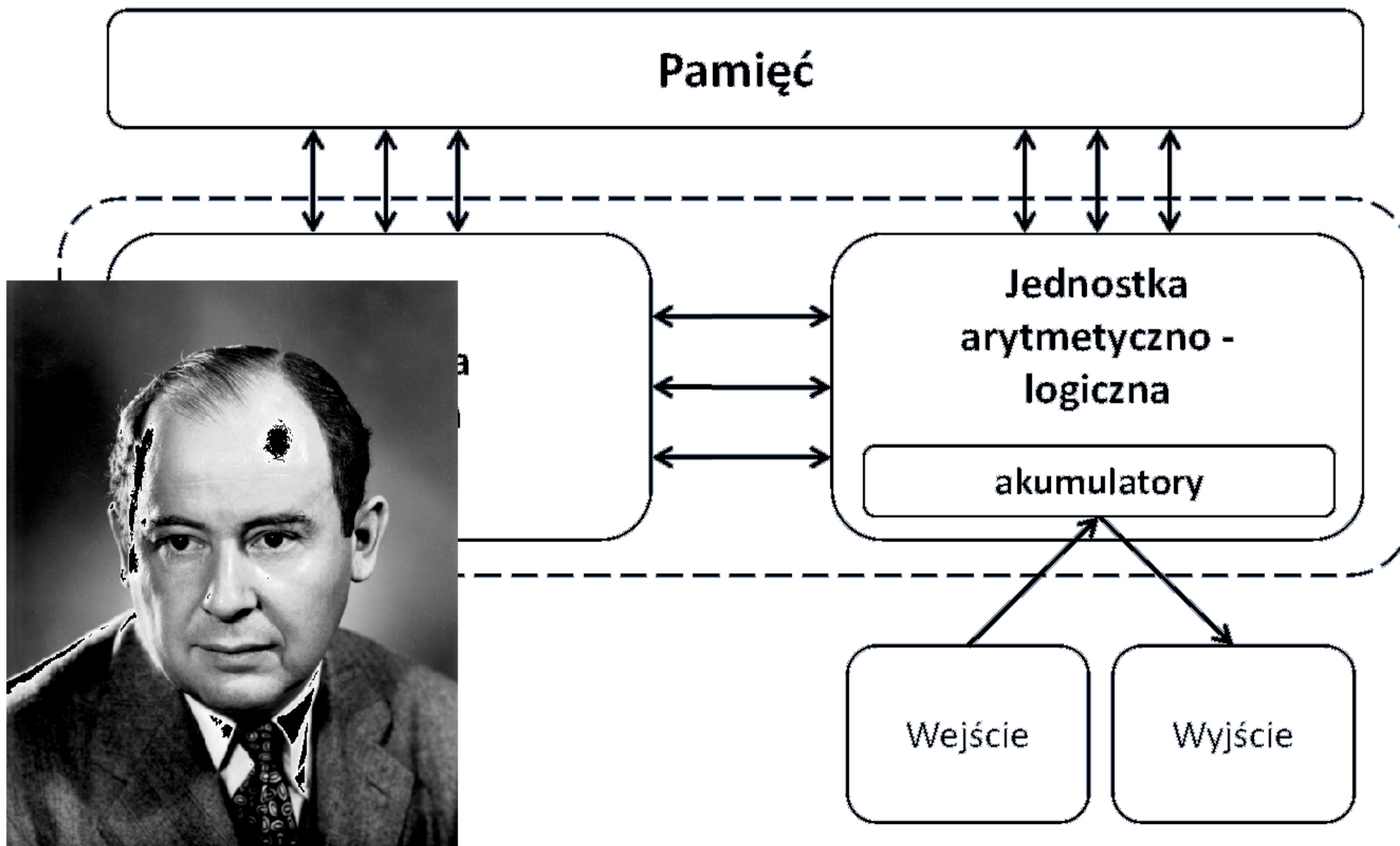
$$\{x=2, y=11\}$$

$$\{x=2, y=11\}$$

$$y := y + 5$$

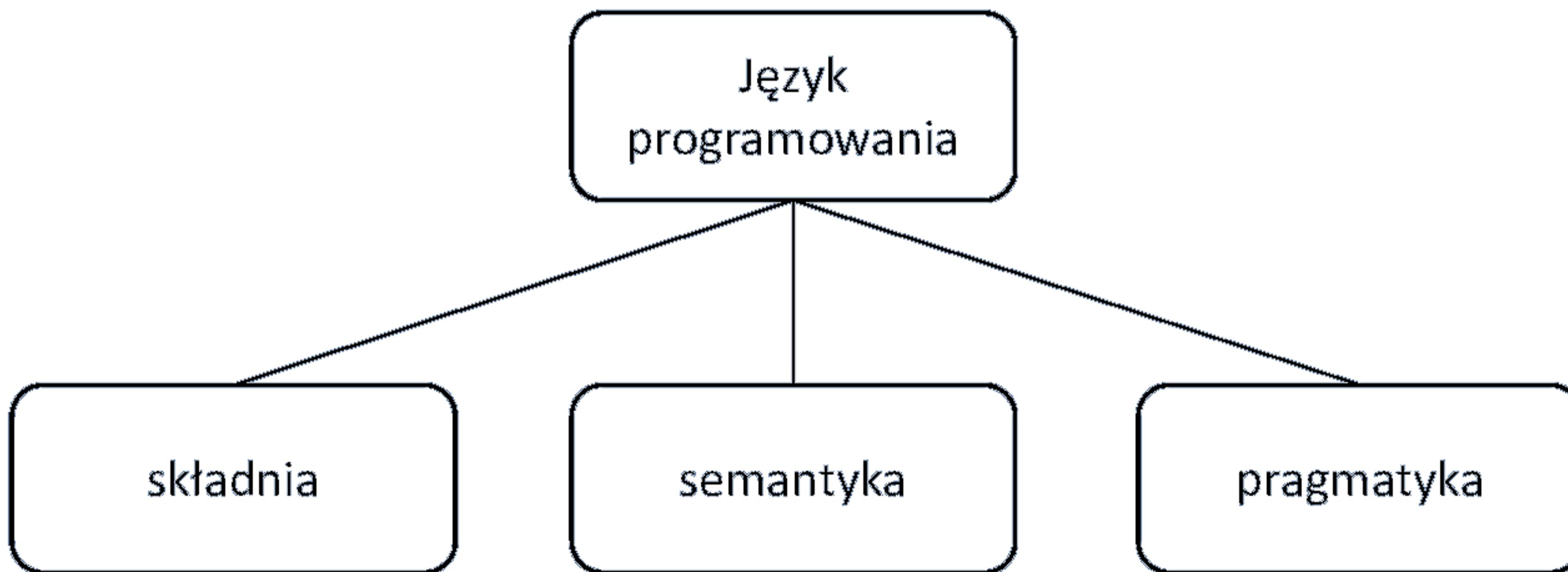
$$\{x=2, y=16\}$$

Maszyna John'a von Neumanna



1903-1957

Co to jest język programowania?



Składnia języków programowania

Zbiór zasad określających, który ciąg znaków jest poprawną instrukcją w danym języku programowania.

Do definiowania składni języka programowania najczęściej używa się notacji BNF (Backus-Naur Form)

```
<expression> ::= <term> |
                  <expression> '+' <term> |
                  <expression> '-' <term>
<term>      ::= <factor> |
                  <term> '*' <factor> |
                  <term> '/' <factor>
<factor> ::= <number> | '(' <expression> ')' 
```

Semantyka języków programowania

Określa relację pomiędzy elementami składni języka programowania, a modelem obliczeniowym.

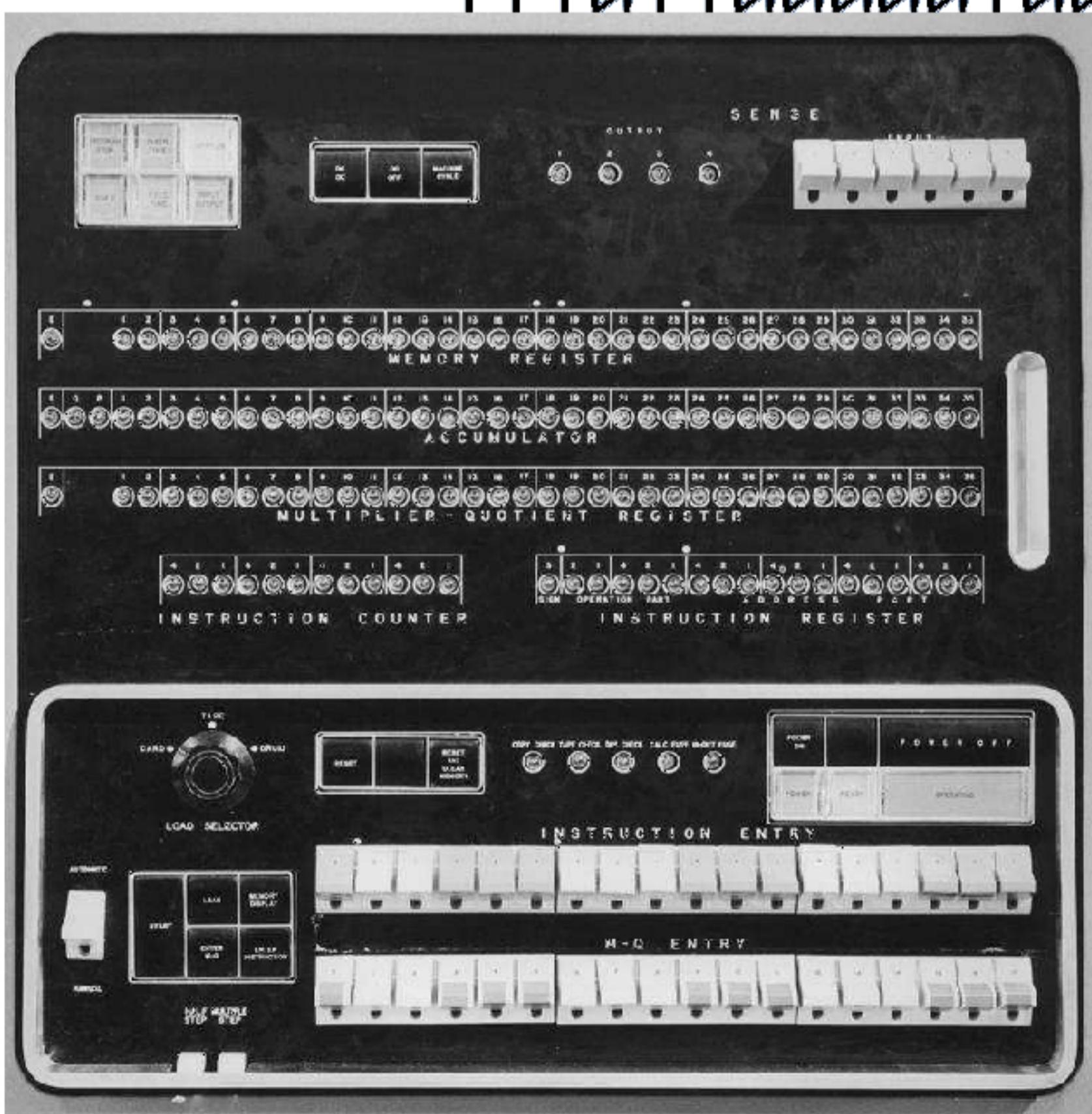
- Semantyka aksjomatyczna
- Semantyka denotacyjna
- Semantyka operacyjna

Pragmatyka języków programowania

Opisuje praktyczne aspekty implementacji języka i korzystania z niego. Określa m.in. jak konstrukcje i cechy języka mogą być wykorzystane do osiągnięcia określonego celu.

Ewolucja języków programowania

Początkowo programy wyglądały tak:



010101011000100:
1110110000001000

IDENTIFICATION	LOCATION	OPERATION	COMMENT
00000000	0000	00000000000000000000000000000000	
11111111	1111	11111111111111111111111111111111	
22222222	2222	22222222222222222222222222222222	
33333333	3333	33333333333333333333333333333333	
44444444	4444	44444444444444444444444444444444	
55555555	5555	55555555555555555555555555555555	
66666666	6666	66666666666666666666666666666666	
77777777	7777	77777777777777777777777777777777	
88888888	8888	88888888888888888888888888888888	
99999999	9999	99999999999999999999999999999999	
A	A	A A A A A A A A A A A A A A A A A A A A	
0	0	01111100010000110	
1	1	001000110100000100	
0	0	000100010111111100	
0	0	00000000000000000000	
0	0	011100001110010000	
0	0	001001000100100000	

01111100010000110
001000110100000100
000100010111111100
00000000000000000000
011100001110010000
001001000100100000

Ewolucja języków programowania

Funkcja w języku assembler (nowoczesnym)

```
push ebp  
mov ebp, esp  
sub esp, 0x10  
mov DWORD PTR [ebp-0xc], 0x5  
mov DWORD PTR [ebp-0x8], 0x6  
mov eax, DWORD PTR [ebp-0x8]  
mov edx, DWORD PTR [ebp-0xc]  
lea eax, [edx+eax*1]  
mov DWORD PTR [ebp-0x4], eax  
mov eax, 0x0  
leave  
ret
```

Ewolucja języków programowania

Za twórcę pierwszego assemblera uważany jest Nathaniel Rochester III



1919 - 2001

Ewolucja języków programowania

... i w C

```
int main()
{
    int a = 5;
    int b = 6;
    int c = a+b;
    return 0;
}
```

Ewolucja języków programowania

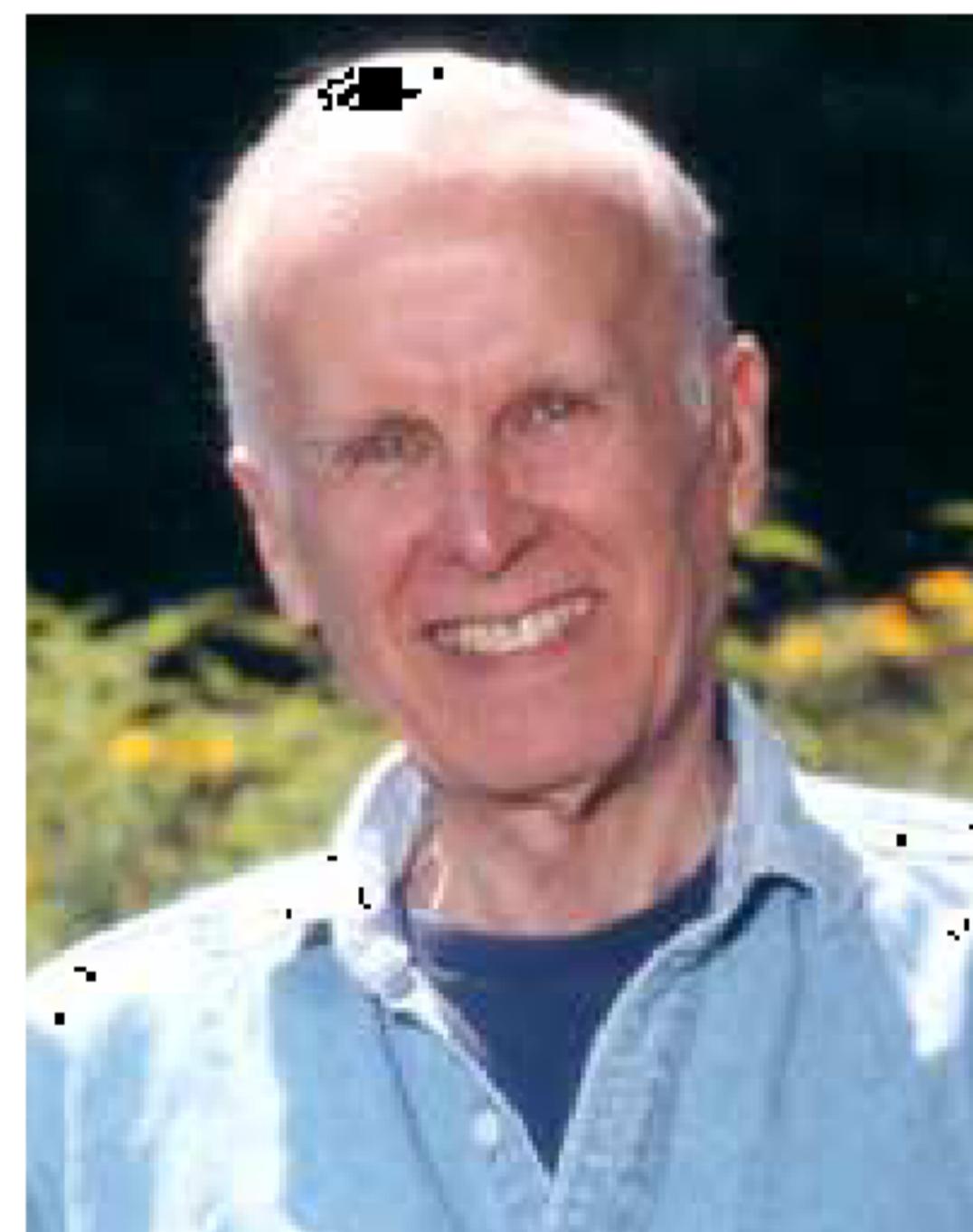
Twórca pierwszego
kompilatora



1906 - 1992

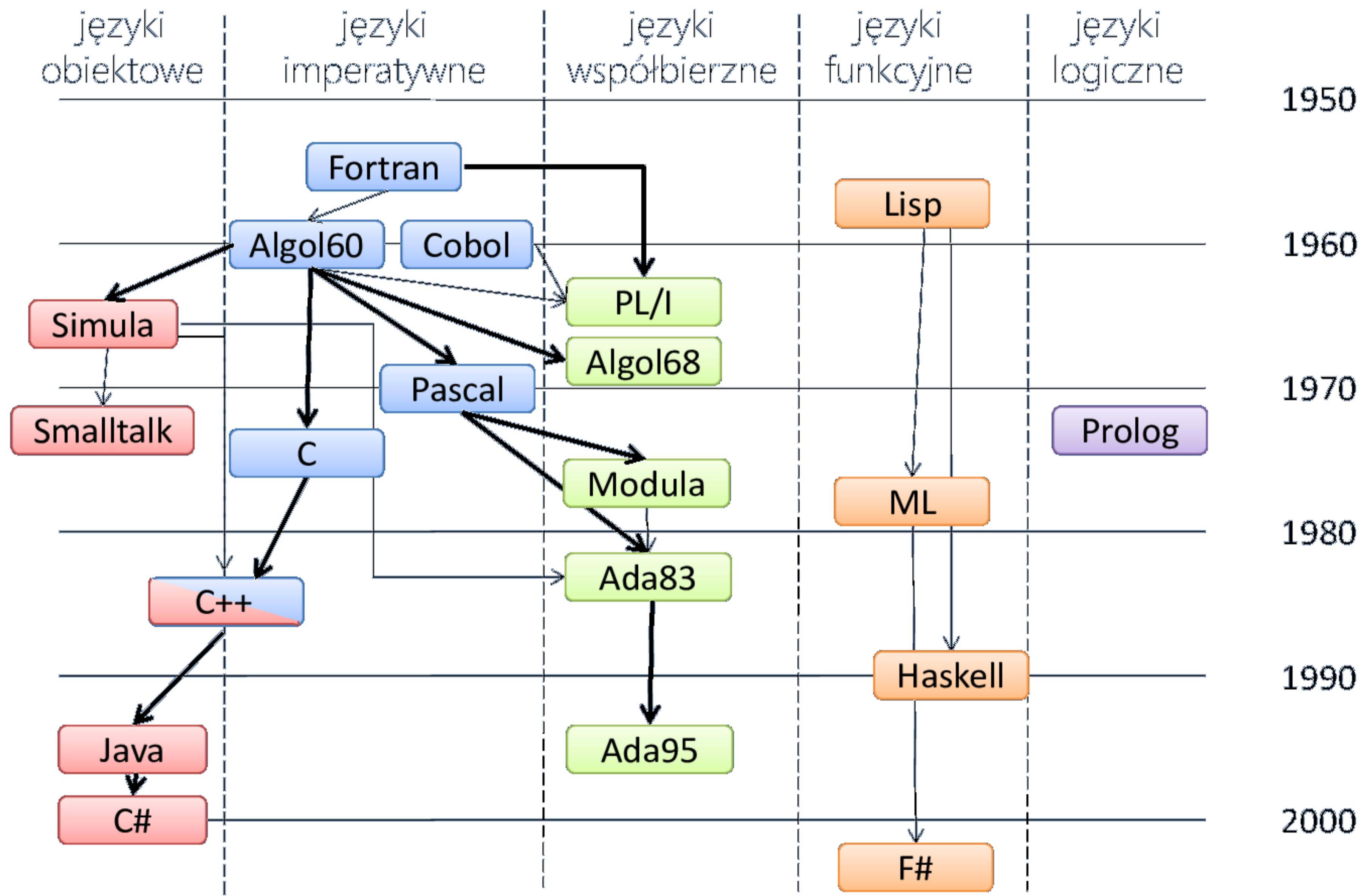
Twórca języka
Fortran

John Backus



1924 - 2007

Ewolucja języków programowania

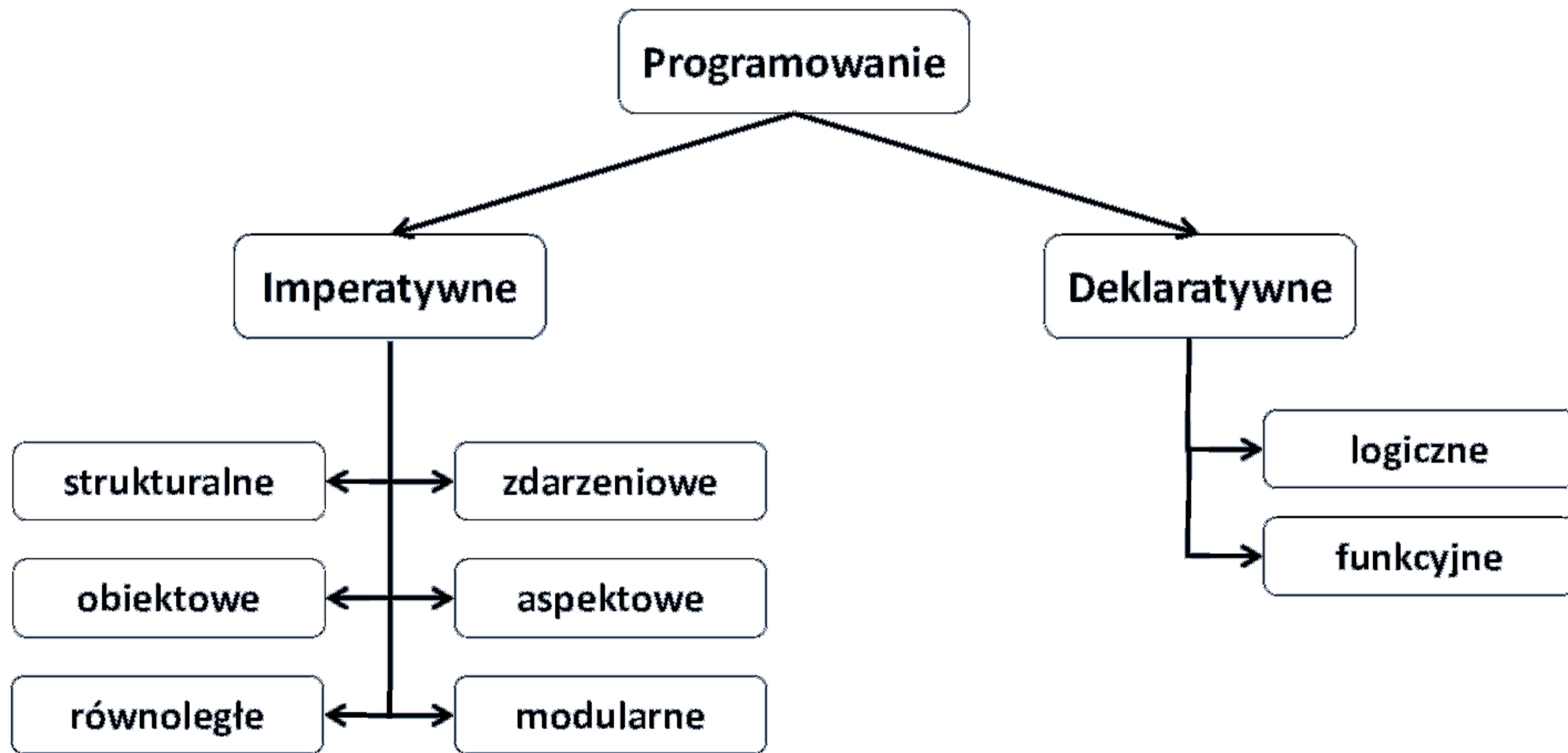


Paradygmat programowania

Paradygmat programowania jest wyróżniającym się stylem programowania. Każdy paradygmat jest scharakteryzowany przez dominację pewnych kluczowych koncepcji.

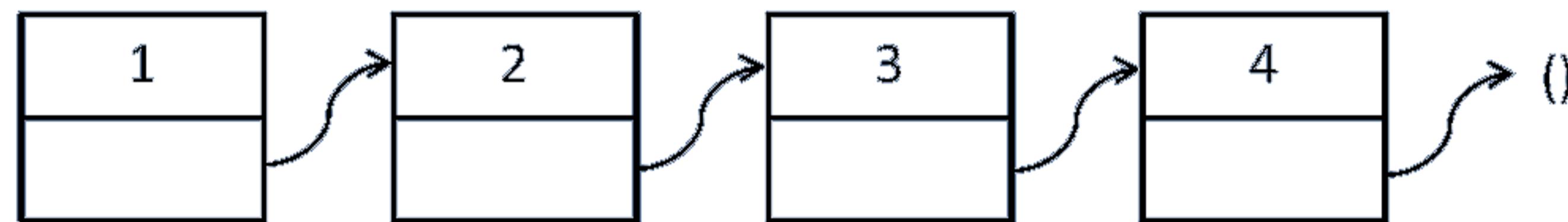
- zbiór koncepcji reprezentujących podejście do implementacji algorytmów
- zbiór mechanizmów używanych przez programistę do pisania programów i określających jak te programy będą następnie wykonywane przez komputer

Paradygmat programowania



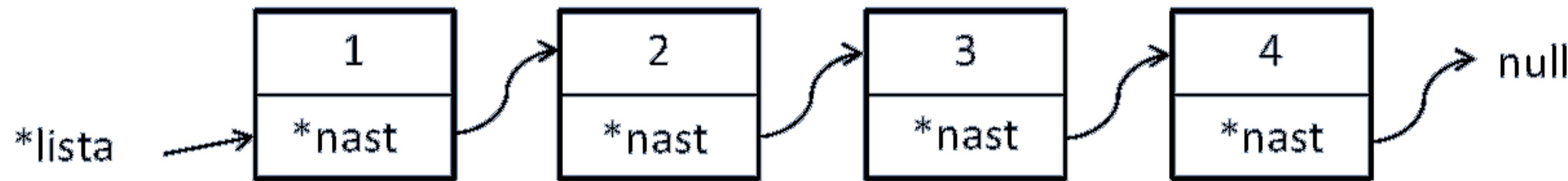
Problem do rozwiązania

Dana jest lista liczb całkowitych:



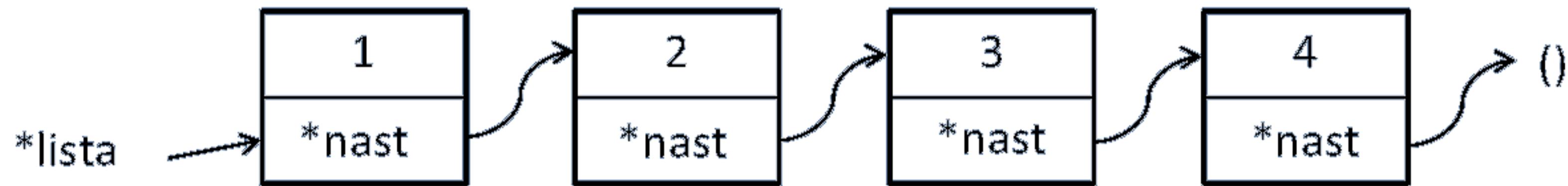
Napisać funkcję, która będzie obliczała sumę elementów na liście

Programowanie imperatywne



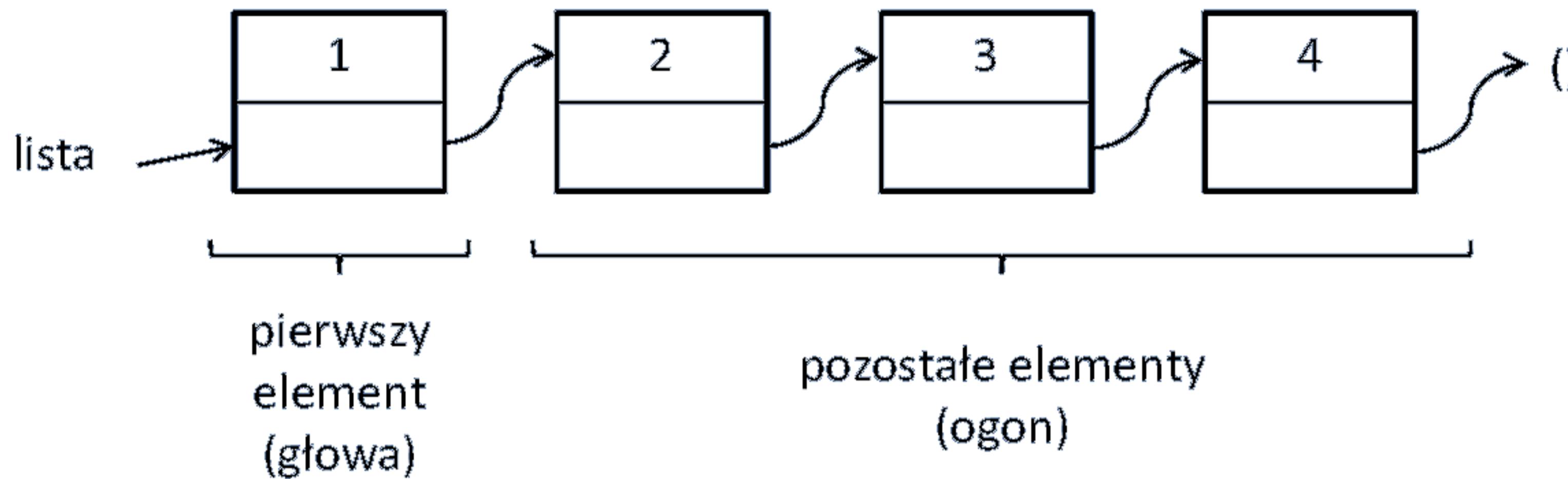
1. Stwórz zmienną **biezacy** określającą aktualny element listy i przypisz do niej adres pierwszego elementu na liście (parametr funkcji)
2. Stwórz zmienną **suma** przechowującą sumę wartości kolejnych elementów listy
3. W pętli (dopóki zmienna **biezacy** ma inną wartość niż **null**):
 1. Pobierz wartość składowej **wartosc** aktualnego elementu listy, pobierz wartość zmiennej **suma** zawierającą sumę wszystkich dotychczasowych elementów. Dodaj te dwie wartości do siebie. Wynik przypisz do zmiennej **suma**.
 2. Pobierz wartość wskaźnika **nast** aktualnego elementu na liście i przypisz go do zmiennej **biezacy**
4. Zwróć wartość zmiennej **suma** do programu wywołującego

Programowanie imperatywne



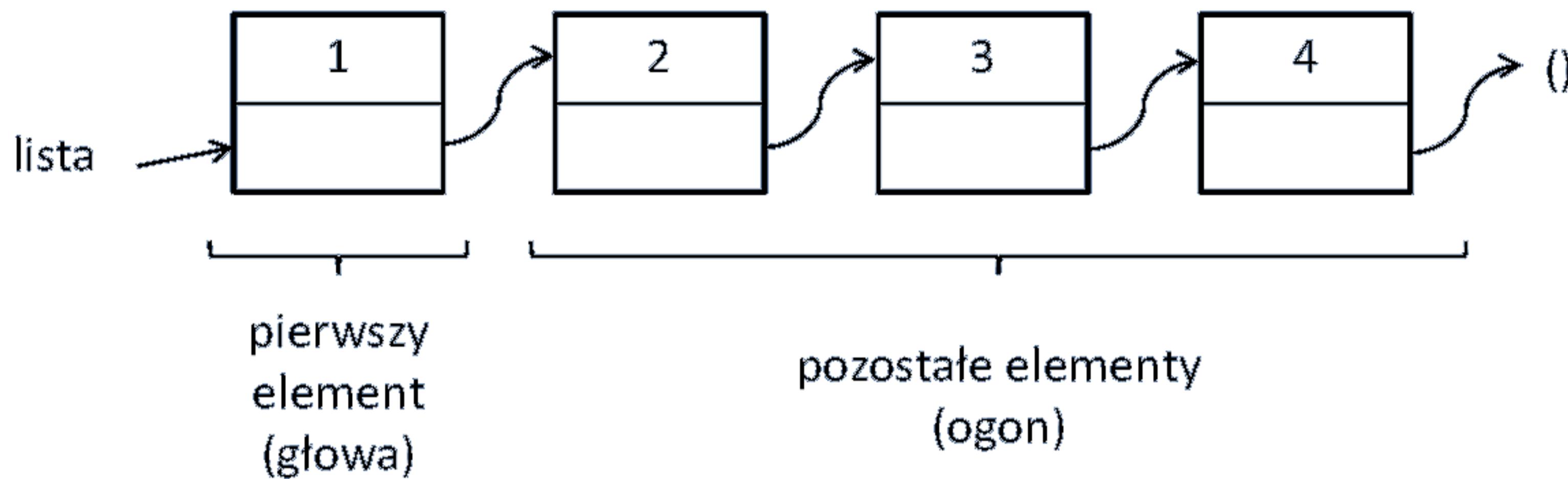
```
int Sumuj(Lista* lista)
{
    Lista* biezacy = lista;
    int suma = 0;
    while(biezacy != NULL)
    {
        suma = suma + biezacy->wartosc;
        biezacy = biezacy->nast;
    }
    return suma;
}
```

Programowanie funkcyjne



1. Jeżeli lista jest pusta to zwróć 0.
2. Jeżeli lista nie jest pusta to pobierz wartość pierwszego elementu na liście i dodaj go do sumy pozostałych elementów na liście

Programowanie funkcyjne



```
(define (Sumuj lista)
  (if (empty lista)
      0
      (+ (first lista) (Sumuj (rest lista)))))
```

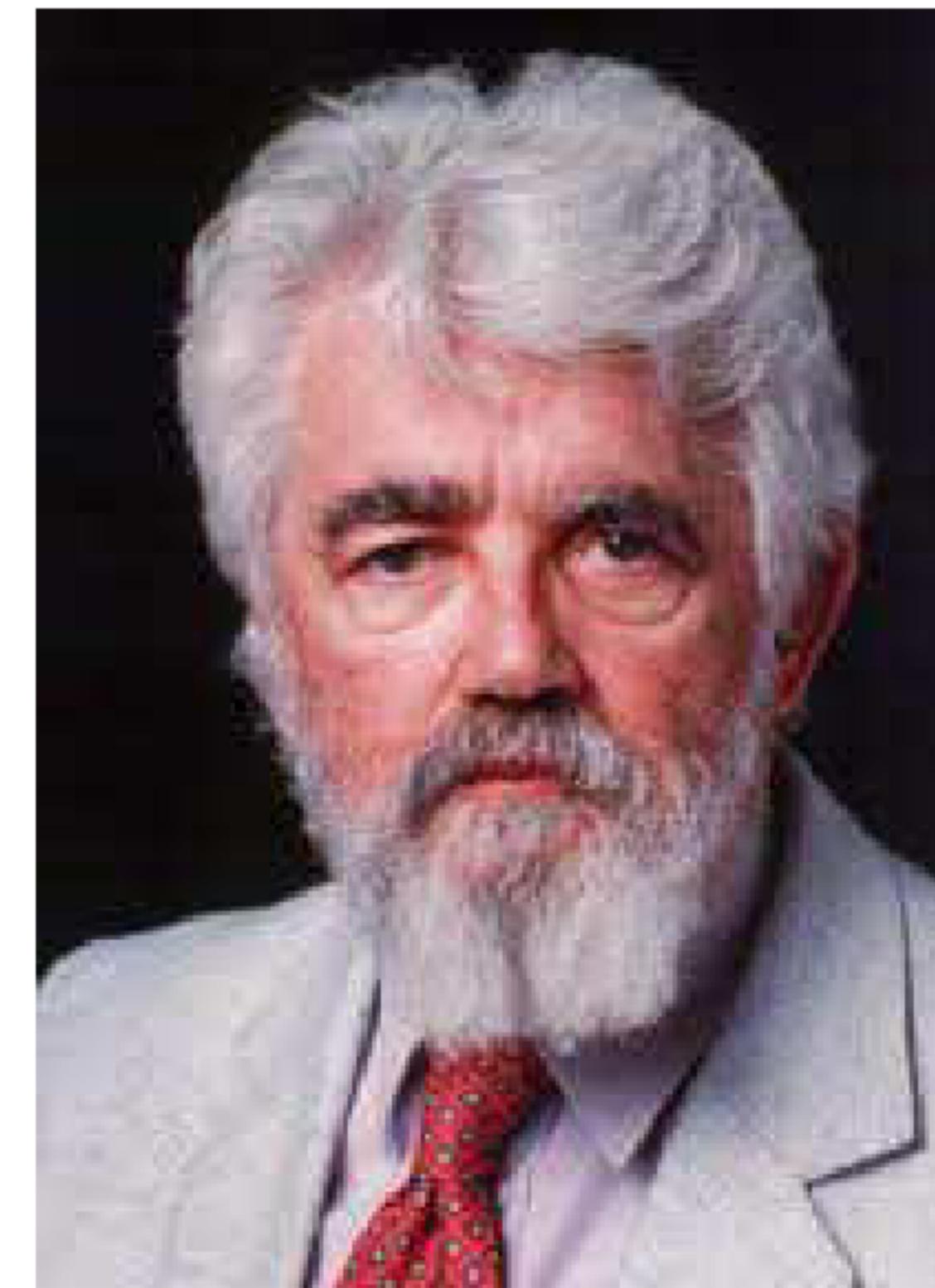
Programowanie funkcyjne

Alonzo Church
(twórca rachunku lambda)



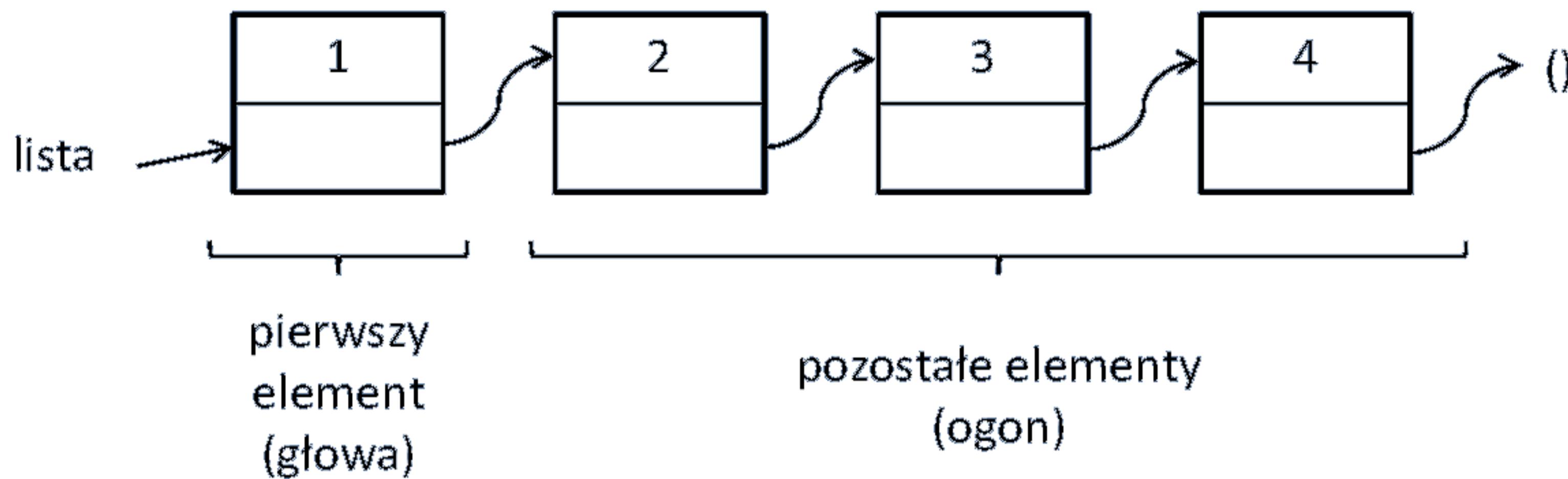
1903 - 1995

John McCarthy
(projektant języka Lisp)



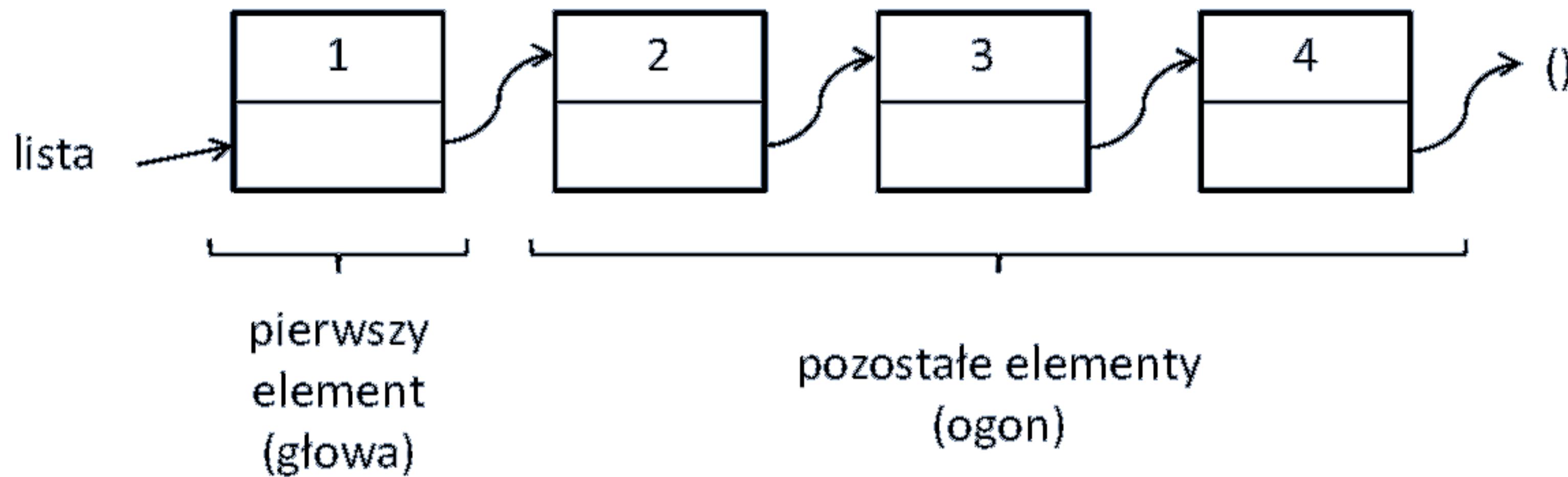
1927 - 2011

Programowanie w logice



1. Jeżeli lista jest listą pustą to prawdą jest, że suma jej elementów Y równa się 0.
2. Jeżeli lista ma głowę X i ogon Xs oraz suma elementów w ogonie to Suma, to prawdą jest, że odpowiedzią jest X plus Suma

Programowanie w logice



```
sumuj([], Y) :- Y is 0.
```

```
sumuj([X|Xs], Y) :- sumuj(Xs, Suma), Y is Suma+X.
```

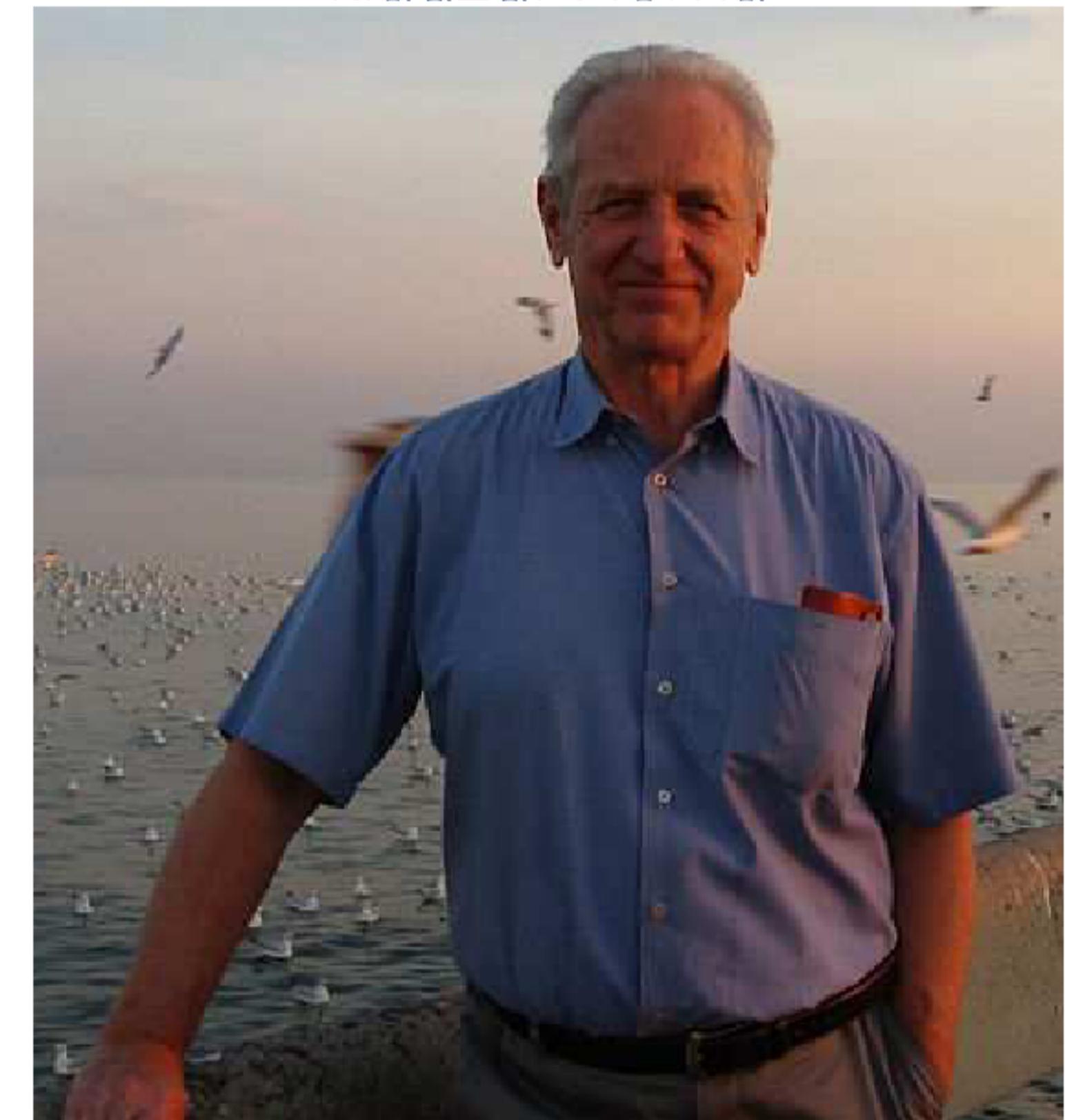
Programowanie w logice

Alain Colmerauer
Twórca języka Prolog



1941-

Robert "Bob" Anthony Kowalski
Twórca proceduralnej interpretacji
klauzul Horna



1941-

Agenda wykładu

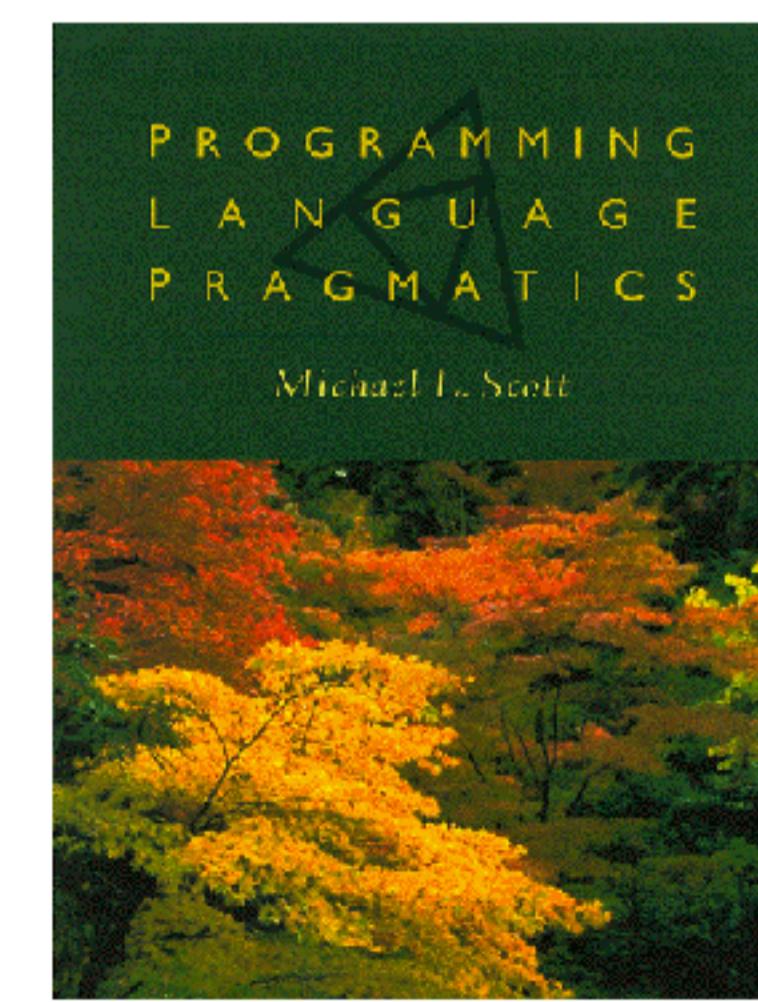
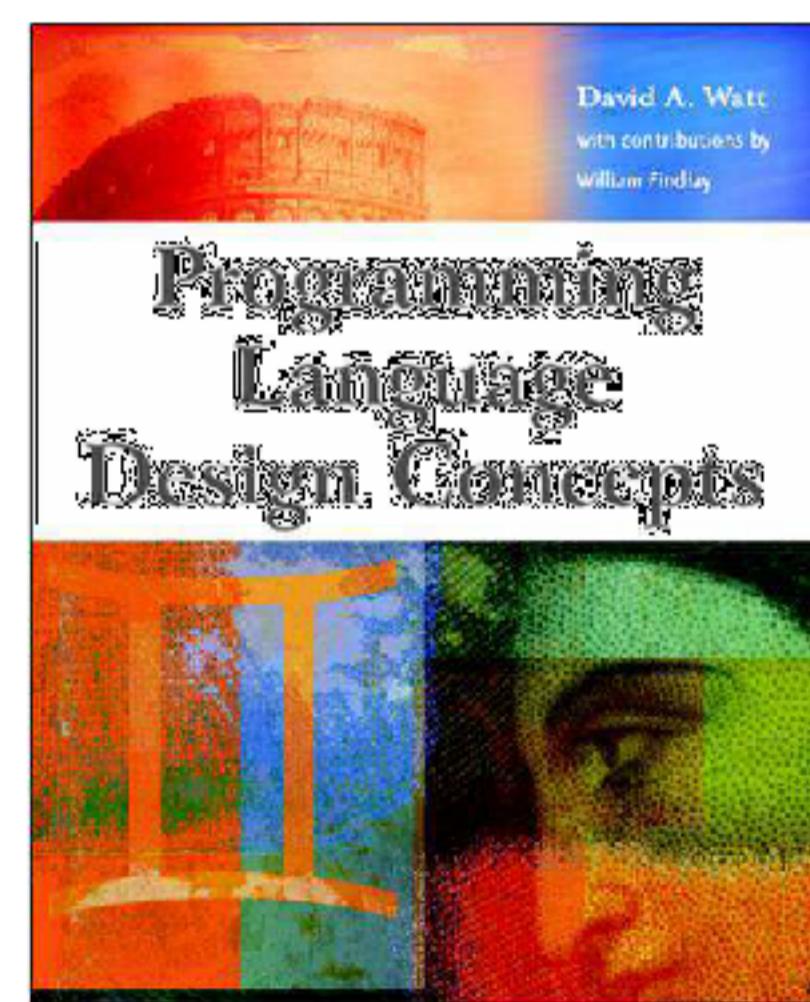
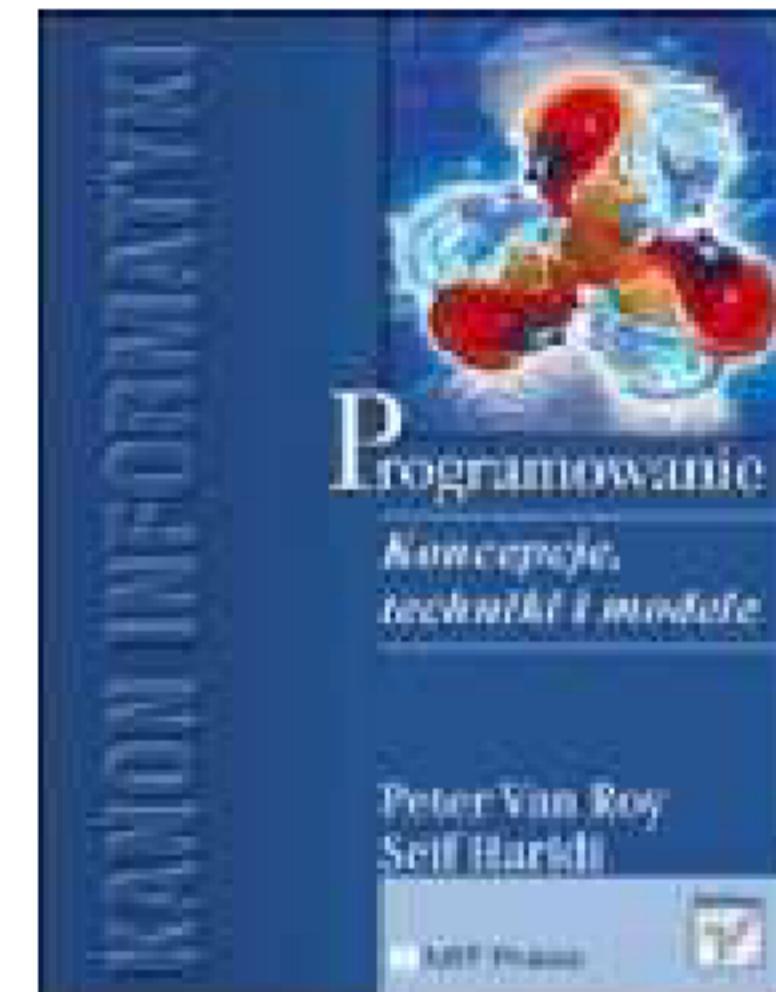
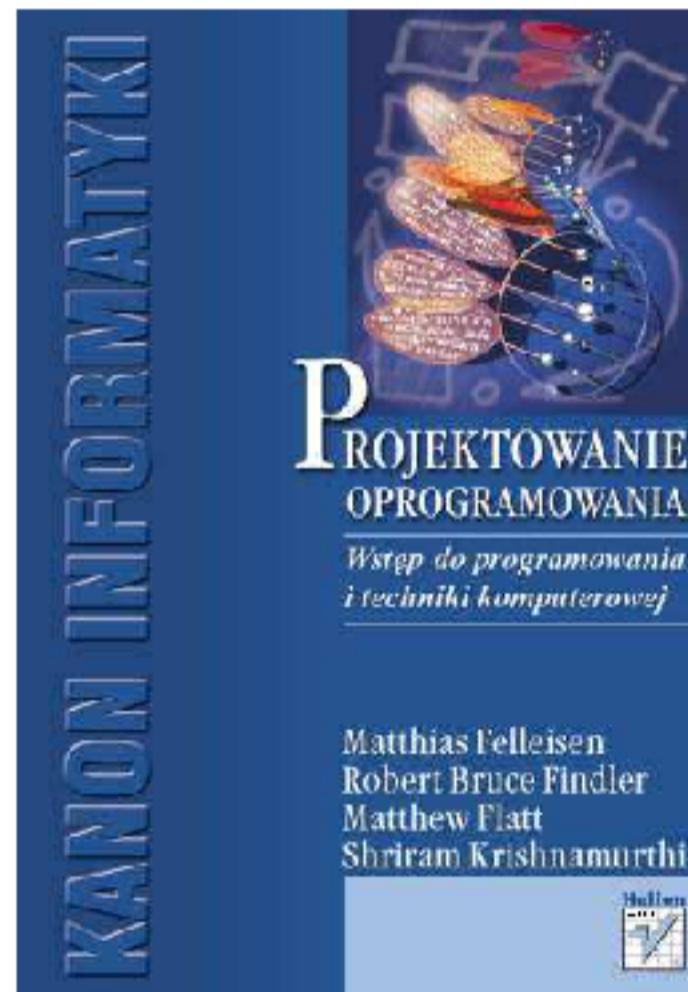
1. Wprowadzenie
 - I. Programowanie imperatywne i nie tylko
2. Typy danych
3. Zmienne
4. Nazwy i ich wiązanie
5. Abstrakcje programistyczne
 - II. Programowanie funkcyjne
6. Wstęp do programowania funkcyjnego
7. Język Scheme
8. Język F# i programowanie funkcyjne w języku C#
9. Rachunek Lambda
 - III. Programowanie w logice
10. Wstęp do programowania w logice
11. Język Prolog
 - IV. Inne paradigmaty programowania
12. Programowanie obiektowe inaczej
13. Domain Specific Languages (DSL)

Zaliczenie przedmiotu

Przedmiot kończy się egzaminem

1. Do egzaminu będą dopuszczone tylko osoby, które uzyskały zaliczenie laboratorium.
2. Nie będzie zwolnień z egzaminu.
3. Egzamin będzie przeprowadzony w formie testowej.

Literatura



Literatura

Dostępne z komputerów na uczelni

- **Programming Languages: Principles and Paradigms**
Maurizio Gabbrielli and Simone Martini, Springer 2010
<http://link.springer.com/book/10.1007/978-1-84882-914-5>
- **Principles of Programming Languages**
Gilles Dowek, Springer 2009
<http://link.springer.com/book/10.1007/978-1-84882-032-6>
- **Programming and Meta-Programming in Scheme**
Jon Pearce, Springer 1998
<http://link.springer.com/book/10.1007/978-1-4612-1682-7>
- **Foundations of F#**
Robert Pickering, Springer 2007
<http://link.springer.com/book/10.1007/978-1-4302-0358-2>
- **Beginning F#**
Robert Pickering, Springer 2010
<http://link.springer.com/book/10.1007/978-1-4302-2390-0>
- **Expert F# 3.0**
Don Syme, Adam Granicz, Antonio Cisternino, Springer 2012
<http://link.springer.com/book/10.1007/978-1-4302-4651-0>

Literatura

Ogólnodostępne

- ***Introduction to Programming Languages***
Anthony A. Aaby
[http://staffweb.worc.ac.uk/DrC/Courses%202006-7/Comp%204070/
Reading%20Materials/book\[1\].pdf](http://staffweb.worc.ac.uk/DrC/Courses%202006-7/Comp%204070/Reading%20Materials/book[1].pdf)
- **The Scheme – Programming Language Fourth Edition**
The MIT Press 2009
R. Kent Dybvig
<http://www.scheme.com/tspl4/>
- **Learn Prolog Now!**
College Publications 2006
Patrick Blackburn, Johan Bos, and Kristina Striegnitz
<http://www.learnprolognow.org/>