

Poetycki aspekt
programowania

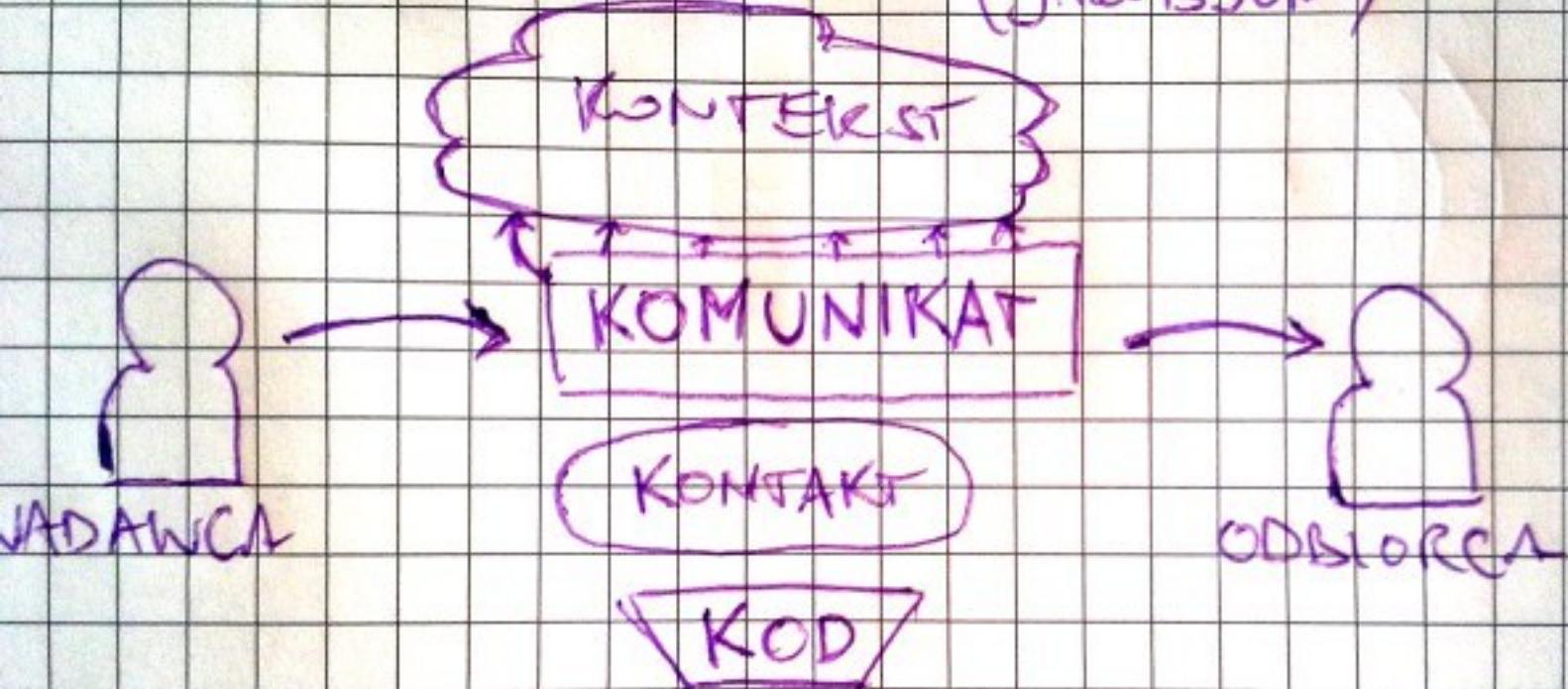
Poetycki aspekt programowania

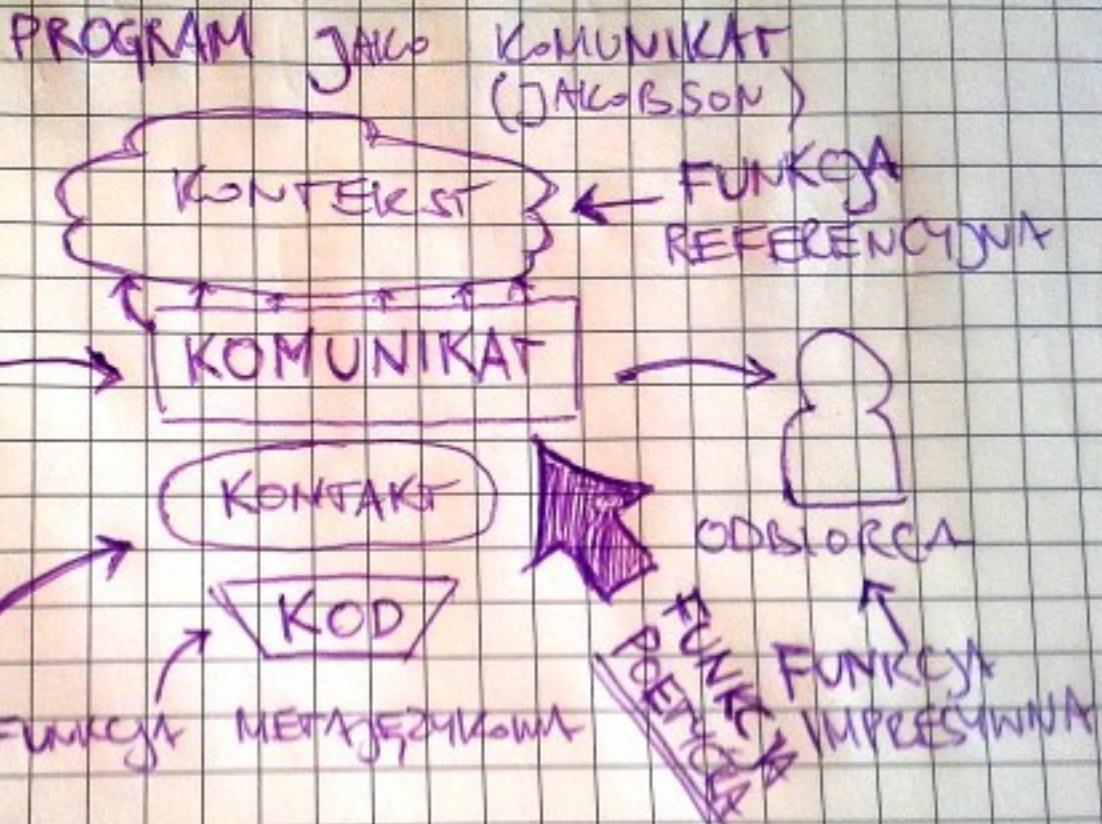
(w ujęciu strukturalistycznym)

PROGRAM JÄLG KOMUNIKAT
(JÄLG-BSON)

KOMUNIKAT

PROGRAM JAKO KOMUNIKAT (JACOBSSON)





JAKIW Guarana

#!/usr/bin/perl

APPEAL:

listen (please, please);

open yourself, wide;

join (you, me),

connect (us,together),

tell me.

do something if distressed;

@dawn, dance;

@evening, sing;

read (books,\$poems,stories) until peaceful;

study if able;

write me if-you-please;

sort your feelings, reset goals, seek (friends, family, anyone);

do-not-die (like this)

if sin abounds;

play (laugh), open (eyes, mind), tell secrets;

Rozgrzewka :

KRAZENIE RAMION !

NA ZACHĘTE - PROGRAMOWANIE A POZIOME

MY CHĘCIĘ, ŻE WIEŚĆ, GDY SIĘ NAUCZYŁ, MAM WIEKOMI PERNÓCIĘ, GDY ZDAŁ EGZAMIN, JEDNEGO WIEKOMI MAMY, GDY NAUCZYŁ INNYCH, LECH CZEŁKOWICZ DZIĘKI WTEJ, GDY POSTAWIŁ SIĘ TU ZAPROGRAMOWAĆ

- ALAN PEDUS
(lowest regardy funinge)

PROGRAM JAKO TEKST.

CZYM JEST TEKST?

PROGRAM JAKO TEKST.

CZYM JEST TEKST?

- ELEMENTARNE JEDNOSTKI (SLOWA)

PROGRAM JAK TEKST.

CZYM JEST TEKST?

- ELEMENTARNE JEDNOSTKI (SŁOWA),
z których w grąku o
- REGUŁY KOMPOZYCJI

PROGRAM JAK TEKST.

CZYM JEST TEKST?

- ELEMENTARNE JEDNOSTKI (SŁOWA),
z kreskami, w grupach o
- REGUŁY KOMPOZYCJI
BUDUJĄCICH
- ZEZWOLENE CATOSCI (ZDANIA)

STRUKTURA TEKSTU:

- SEKWENCJA SŁÓW
- PEŁNE ELEMENTY SYNAZJONISTYCZNE
GDZIE INNYCH (np. Antyku, dawno
zaczynają się od chłopów, mówią
pomiędzy plemionami)
- MÓWIĄCY DŁA KOGO TAKU ZBUDOWĄ MIAŁO
ROZWIÓRZ GRAMATYCZNEGO

PRZYKŁAD: BIEDEM MAMY SKŁADNIK WYZWOLONY Z WOLNEJ
DOSTĘPNOŚCI RÓWNIEŻ SIE NAPRAWD

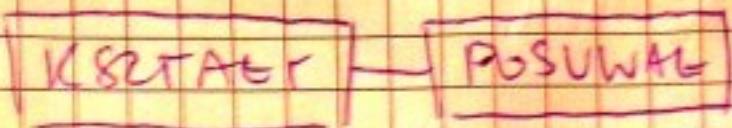
KTO?

CO ROBI?

PRZYKŁAD: BIEDNY MAMY SICUŁOMY KONTAKT Z WOLNA
I OSZTROŻNIE POSUWAĆ SIĘ WŁAŚCIWOD

KTO?

CO ROBIĘ?



PRZYKŁAD: BIEDEM MAMY SŁOWNY KONTRAKT Z WOCHE
I OSTRZENIE POSUWATE SIĘ NAPEZWÓD

KTO?

KONTAKT

CO ROBIĘ?

POSUWAM

DOKOŁ?

NAPEZWÓD

KOGO?

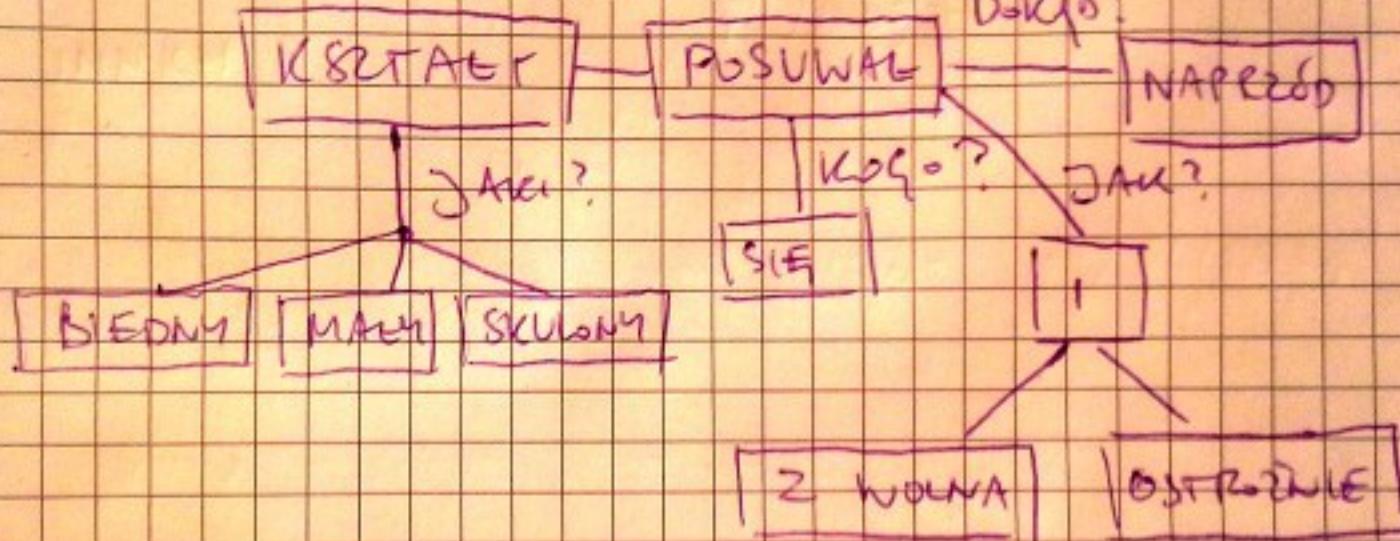
SIĘ

PRZYKŁAD: BIEDEM MAMY SKULONY KONTAKT Z WOLNA
I OSTRZENIE POSUWAJE SIĘ NAPEZWÓD

KTO?

CO ROBIĘ?

DOKOŁ?



STĘPSTWA REPREZENTACJA DRZEWIA

(POSUWATE #; KTO?

(KSIĘTAŁT #; JAKI? BIEDNE MATY SKŁADNY)

#; KOGO? SIE

#; DOKÓD NAPRZED

#; JAK? ((2-MOLNA OSTRZENIE))

ANALYSTENSTU - OWM JEST ZAKOŃCZENIE?

ANALIZA TEKSTU - COIM JEST ZNACZENIE?

HIPOTEZA: ZNACZENIEM WYRAIENIA JEST
INNE WYRAIENIE

ANALIZA TEKSTU - CHYM JEST ZNACZENIE?

HYPOTEZA: ZNACZENIEM WYRAŻENIA JEST
INNE WYRAŻENIE

KOMPOZYCJONALISMO: ZNACZENIE WYRAŻENIA
ZMIENI SIĘ OD:

- ZNACZENIA ELEMENTÓW SŁUŻĄcych
- SPOSOBU ICH POŁĄCZENIA

ANALIZA TEKSTU - Czym jest założenie?

HIPOTEZA: ZNACZENIEM WYRAŻENIA JEST
INNE WYRAŻENIE

KOMPOZYCJONALNOŚĆ: ZNACZENIE WYRAŻENIA
ZMIENIA OD:

- ZNACZENIA ELEMENTÓW SKŁADOWYCH
- = SPOSÓB ICH ROZDZIENIA
(i TYLKO OD TEGO!)

ANALIZA TEKSTU - CO JEST ZNACZENIE?

HIPOTEZA: ZNACZENIEM WYRAŻENIA JEST
INNE WYRAŻENIE

KOMPOZYCJONALNOŚĆ: ZNACZENIE WYRAŻENIA
ZMIENIĘ OD:

- ZNACZENIA ELEMENTÓW SKŁADOWYCH

= SPOSÓBU ICH ROTACJI

(I TYLKO OD TEGO!)

UWAGA! KOMPOZYCJONALNOŚĆ CZASEM NIE ZAKŁADA

KOMPOZYCJONALNOŚĆ: ZNACZENIE WYRAZEM
ZMĘCIĆ OBÓZ:

- ZNACZENIA ELEMENTÓW SŁUŻĄCYCH
- SPOSÓBU ICH PRZYGOTOWANIA
(! TYLKO OD TEGO!)

UWAGA! KOMPOZYCJONALNOŚĆ CRASH NIE ZAKOŃCZA

NP. DLA NAZW „SWINKA MORSKA”, „JEDNORĘKA GIGANT”

(I TYLKO OD TEGO!)

UWAGA! Kompozycjonalność czasu nie zachowana

np. Dla nazw „Swinka Morska”, „Jednorzędowy smutek”

Dla „Cząsteczek intensywności”, np.

„Louis Lane wie, że Superman umie latać”

vs. „Louis umie wie, że Clark Kent umie latać”

(Mimo że „Superman” = „Clark Kent”

„CZŁOWIĘK SW. INTENSYWNYM” , N.P.

„LOUIS LANE WIE, ŻE SUPERMAN UMIE LATAC”

„LOUIS UMIE WIE, ŻE KLAUS KENT UMIE LATAC”

UMIESCIĘ „SUPERMAN” = „KLAUS KENT”)

KOMPOZYCYJNOŚĆ = PRZEROCZYSTOŚĆ
ODNIESIENIOWA

ZNACZENIE PROGRAMU

ZNACZENIE PROGRAMU

- ZNACZENIE WYRAŻEŃ SKŁADOWYCH
- I

ZNACZENIE PROGRAMU

- ZNACZENIE WYRAŻEŃ SKŁADOWYCH
- MODYFIKACJA ŚRODOWISKA

ZNACZENIE PROGRAMU

- ZNACZENIE WYRAŻEŃ SKŁADOWYCH
- MODYFIKACJA ŚRODOWISKA

$$\langle e^*, c^* \rangle = m(e, c), \text{ gdzie}$$

e - kod programu

c - środowisko (kontekst)

e^* - wynik programu

c^* - wstęp do środowiska po wykonaniu programu

- ZNACZENIE WYRAŻEŃ SKŁADOWYCH

- MODYFIKACJA ŚRODOWISKA

$$\langle e^*, c^* \rangle = m(e, c), \text{ gdzie}$$

e - kod programu

c - środowisko (kontekst)

e^* - wynik programu

c^* - środowisko środowiska po wykonaniu programu

PROGRAM FUNKCYJNY:

$$\langle e^*, c \rangle = m(e, c)$$

PROGRAM IMPERATYWNY:

$$\langle \phi, c^* \rangle = m(e, c)$$

With referential transparency, no distinction is made or difference recognized between a reference to a thing and the corresponding thing itself. Without referential transparency, such difference can be easily made and utilized in programs.

Another example [edit]

As an example, let's use two functions, one which is referentially opaque, and the other which is referentially transparent

```
globalValue = 0;

integer function rq(integer x)
begin
    globalValue = globalValue + 1;
    return x + globalValue;
end

integer function rt(integer x)
begin
    return x + 1;
end
```

The function `rt` is referentially transparent, which means that $rt(x) = rt(y)$ if $x = y$. For instance, $rt(6) = 1 = 7$, $rt(4) = 4 + 1 = 5$, and so on. However, we can't say any such thing for `rq` because it uses a global variable that it modifies.

The referential opacity of `rq` makes reasoning about programs more difficult. For example, say we wish to reason about the following statement:

```
integer p = rq(x) + rq(y) * (rq(x) - rq(x));
```

One may be tempted to simplify this statement to:

Pytanie: Czy lista słów, które mogą
wystąpić w tekście, jest skończona?

PYTAŃE: Czy LISTA SŁÓW, KTÓRE MOGĄ
WYSTAĆ W TEKŚCIE, JEST SKONCZONA?

ODPOWIEDŹ: MOŻEMY DEFINOWAĆ
NOWE SŁOWA!

PYTANIE: CZY LISTA SŁÓW, KTÓRE MOGĄ
WYSTAĆ W TEKŚCIE, JEST SKONCZONA?

ODPOWIEDŹ: MOŻEMY DEFINOWAĆ
NOWE SŁOWA!

WNIOSEK: POTRZEBUJEMY TERMINÓW
PIERWOTNYCH, (SR-DKSW) EACENIA I (SR-DKSW)
ABSTRAKCJI

KOLEJNY WNIOSZEK : WYKONANIE PROGRAMU

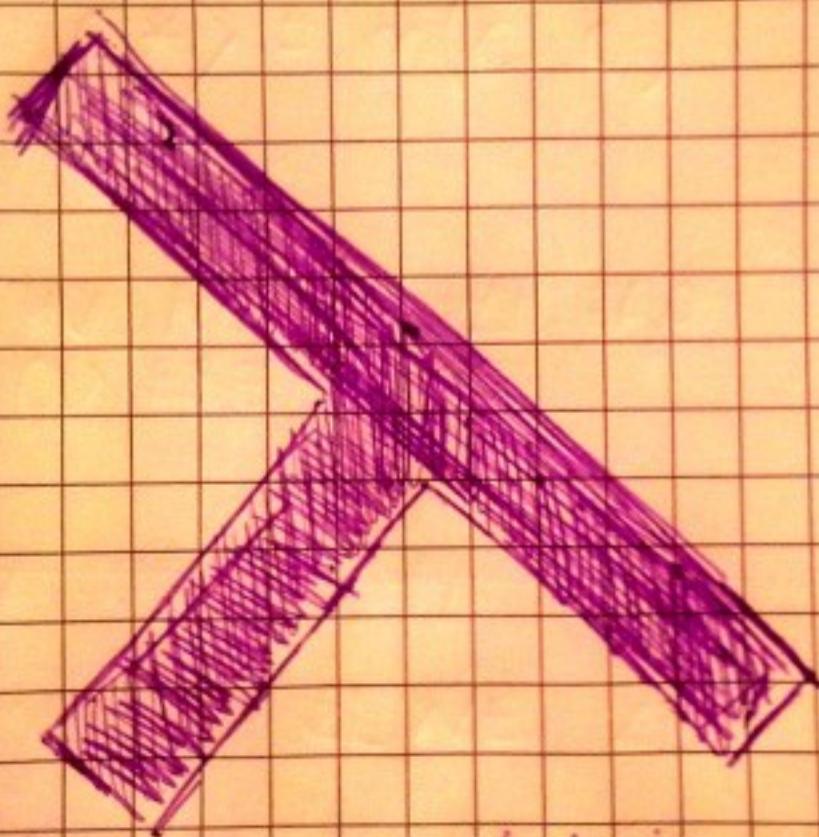
BĘDZIE POLEGAĆ NA SPRAWADZENIU POGÓRZŁOŻYCH

→ NAJPRZYSTĘPNIĘSZYM TERMINOW

KOLEJNY WNIOSZEK : WYKONANIE PROGRAMU
BEDZIE POLEGACO NA SPRAWADZENIU POCZECI ZLOZONYCH
DO NAIPIRZYSTSZYCH TERMINOW

PYTANIE : JAKI JEST NAJMNIĘJSZY ZBIOR
REGUL POWZWAŁAĆYCH NA BUDOWANIE ZLOZONYCH
WYKRADEŃ ? (MINIMUM FUNKCJONALNE PROGGRAMU)

ODPOWIEDZ:



z β - redukcja

PRZYKŁAD.

$$(\lambda(x.y) \times \text{me } y)$$

PRZYKŁAD.

$(\lambda \{x \ y\} \ x \ \text{ma} \ y)$

argumenty ciężo

PRZYKŁAD.

$$(\lambda(x\ y)\ x\ \text{ma}\ y)$$

argumenty ciężo

$$((\lambda(x\ y)\ x\ \text{ma}\ y) \text{ Dorota}\ \text{kota})$$

PRZYKŁAD.

$$(\lambda(x\ y)\ x\ \text{ma}\ y)$$

argumenty cięte

$$((\lambda(x\ y)\ x\ \text{ma}\ y) \text{ Dorota}\ \text{kota})$$

$$\rightsquigarrow \text{Dorota}\ \text{ma}\ \text{kota}$$

B

DEFINICJE POMOCNICZE

DEFINICJE POMOCNICZE

(define varie variabili)

DEFINICE POMOCNICE

(define *newname* *newbody*)

(if *newrule* *newbody* *alternativa*?)

PRZYKŁAD. Znajomy, że dane są funkcje

- , \times , $=$. Zdefiniujmy:

PRZYKŁAD, ZNOTUJMY, ZE DANE SĄ FUNKCJE PIERWOTNE

- * , = . ZDEFINIUJMY :

(define ! (lambda (n) (if (= n 0) 1 (* n (! (- n 1))))))

PRZYKŁAD. ZNOŚMY, ZIE DANE SĄ FUNKCJE PIERWOTNE

- * , = . ZDEFINIUJMY ;

(define ! (lambda (n) (if (= n 0) 1 (* n (! (- n 1))))))

JAKA JEST WARTOŚĆ WYKILĘTA

(! 5) ?

PRZYGOTOWANIE, ZE DANE SĄ FUNKCJE PIERWOTNE

- * , = . ZDEFINUJMY:

(define ! ($\lambda(n)(if (= n 0) 1 (* n (! (- n 1))))$))

DAJKA JEST WARTOŚĆ WYRZECZNA

(! 5) ?

(($\lambda(n)(if (= n 0) 1 (* n (! (- n 1))))$)) 5)

PRZYKŁAD: Znajdujmy, zię dana sę funkcje pierwotne

- $*$, $=$. Zdefiniujmy:

(define ! ($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1)))))))

JAKA JEST WARTOŚĆ WYRAŻENIA

(! 5) ?

(($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1))))) 5)
(if (= 5 0) 1 (* 5 (! (- 5 1)))))

PRZYKŁAD. ZADANIE, ZE DANE SĄ FUNKCJE PIERWOTNE

- \times , $=$. ZDEFINIUJMY:

(define ! ($\lambda(n)$ (if (= n 0) 1 ($\times n (! (- n 1))$))))

JAKA JEST WŁASNOŚĆ WYRZĘDNA

(! 5) ?

(($\lambda(n)$ (if (= n 0) 1 ($\times n (! (- n 1))$)))) 5)
(if (= 5 0) 1 ($\times \underline{5} (! (- 5 1)))$))

PRZYKŁAD, ZNOTUJY, ZE DANE SĄ FUNKCJE PIERWOTNE

- \times , $=$. ZDEFINIUJMY:

(define ! ($\lambda(n)$ (if (= n 0) 1 ($\times n (! (- n 1))$))))

JAKA JEST WARTOŚĆ WYKONIENIA

(! 5) ?

(($\lambda(n)$ (if (= n 0) 1 ($\times n (! (- n 1))$)))) 5)

(if (= 5 0) 1 ($\times 5 (! (- 5 1))$))

($\times 5 (! 4)$)

PRZYGŁĘD, ZAŁOŻYMY, ŻE DANE SĄ FUNKCJE PIERWOTNE

- \times , $=$. ZDEFINIUJMY:

$(\text{define } ! (\lambda(n) (\text{if } (= n 0) 1 (\times n (! (- n 1)))))))$

JAKA JEST WŁASNOŚĆ WYRAŻENIA

$(! 5)$?

$((\lambda(n) (\text{if } (= n 0) 1 (\times n (! (- n 1)))))) 5)$

$(\text{if } (= 5 0) 1 (\underbrace{\times 5 (! (- 5 1)))})$

$(\times 5 (! 4))$

$(\times 5 ((\lambda(n) (\text{if } (= n 0) 1 (\times n (! (- n 1)))))) 4))$

> REZULTAT, ZNOSIEMY, ZE SAME SI FUNKCJE PIERWOTNE

- * , = . ZDEFINIUJMY:

(define ! ($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1)))))

DANIA JEST WŁASNOŚĆ WYKONIENIA

(! 5) ?

(($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1))))) 5)

(if (= 5 0) 1 (* 5 (! (- 5 1))))

(* 5 (! 4))

(* 5 (($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1))))) 4))

(* 5 (if (= 4 0) 1 (* 4 (! (- 4 1)))))

PRZYKŁAD. ZADANIE, ZE DANE SĄ FUNKCJE PIERWOTNE

- $*$, $=$. ZDEFINIUJMY:

(define ! ($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1)))))

JAKA JEST WYKROBÓD WYKONIENIA

(! 5) ?

(($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1))))) 5)

(if (= 5 0) 1 (* 5 (! (- 5 1))))

(* 5 (! 4))

(* 5 (($\lambda(n)$ (if (= n 0) 1 (* n (! (- n 1))))))) 4))

(* 5 (if (= 4 0) 1 (* 4 (! (- 4 1)))))

PRZYKŁAD. ZROBMY, ZIE DANE SĄ FUNKCJE PIERWOTNE

- * , = . ZDEFINIUJMY:

(define ! ($\lambda(n)(\text{if } (= n 0) 1 (* n (! (- n 1))))))$)

JAKA JEST WARTOŚĆ WYKŁĘDZIA

(! 5) ?

(($\lambda(n)(\text{if } (= n 0) 1 (* n (! (- n 1)))))) 5$)

(if (= 5 0) 1 (* 5 (! (- 5 1)))))

(* 5 (! 4))

(* 5 (($\lambda(n)(\text{if } (= n 0) 1 (* n (! (- n 1)))))) 4$))

(* 5 (if (= 4 0) 1 (* 4 (! (- 4 1)))))

(* 5 (* 4 (! 3))))

PRZYKŁAD. ZADANIE, ZE DANE SĄ FUNKCJE PIERWOTNE

- \times , $=$. ZDEFINIOWANY:

(define ! ($\lambda(n)$ (if (= n 0) 1 ($\times n (! (- n 1)))$))))

JAKA JEST WYKRODZI WYKONIENIA

(! 5) ?

(($\lambda(n)$ (if (= n 0) 1 ($\times n (! (- n 1)))$))) 5)

(if (= 5 0) 1 ($\times 5 (! (- 5 1))$))

($\times 5 (! 4)$)

($\times 5 ((\lambda(n)(if (= n 0) 1 ($\times n (! (- n 1)))$))) 4))$

($\times 5 (if (= 4 0) 1 ($\times 4 (! (- 4 1))$))$

($\times 5 (\times 4 (! 3))$)

($\times 5 (\times 4 (\times 3 (! 2)))$)

(* 5 (! 4))

(* 5 ((λ(n)(if (= n 0) 1 (* n (! (- n 1)))))))

(* 5 (if (= 4 0) 1 (* 4 (! (- 4 1)))))

(* 5 (* 4 (! 3))))
.....

(* 5 (* 4 (* 3 (! 2)))))
.....

(* 5 (* 4 (* 3 (* 2 (! 1)))))

(if (= 5 0) 1 (* n (! (- 5 1))))

(* 5 (! 4))

(* 5 (((lambda(n)(if (= n 0) 1 (* n (! (- n 1))))))) 4)

(* 5 (if (= 4 0) 1 (* 4 (! (- 4 1)))))

(* 5 (* 4 (! 3)))

.....
(* 5 (* 4 (* 3 (! 2)))))

.....
(* 5 (* 4 (* 3 (* 2 (! 1))))))

.....
(* 5 (* 4 (* 3 (* 2 (* 1 (! 0)))))))

$((\lambda(n)(if (= n 0) 1 (* n (! (- n 1)))))) 5)$

$(if (= 5 0) 1 (* 5 (! (- 5 1))))$

$(* 5 (! 4))$

$(* 5 ((\lambda(n)(if (= n 0) 1 (* n (! (- n 1)))))) 4))$

$(* 5 (if (= 4 0) 1 (* 4 (! (- 4 1))))))$

$(* 5 (* 4 (! 3))))$

$(* 5 (* 4 (* 3 (! 2))))$

$(* 5 (* 4 (* 3 (* 2 (! 1))))))$

$(* 5 (* 4 (* 3 (* 2 (* 1 (! 0)))))))$

$(* 5 (* 4 (* 3 (* 2 (* 1 (if (= 0 0) n ...)))))))$

$((\lambda(n)(\text{if } (= n 0) 1 (* n (\cdot \text{. } (= n 1))))))$
(if (= 5 0) 1 $(* 5 (! (- 5 1)))$)
 $(* 5 (! 4))$
 $(* 5 ((\lambda(n)(\text{if } (= n 0) 1 (* n (! (- n 1))))))$
 $(* 5 (\text{if } (= 4 0) 1 (* 4 (! (- 4 1)))))$
 $(* 5 (* 4 (* 3 (! 2))))$
 $(* 5 (* 5 (* 3 (* 2 (! 1))))$
 $(* 5 (* 5 (* 3 (* 2 (* 1 (! 0))))$
 $(* 5 (* 4 (* 3 (* 2 (* 1 ((\text{if } (= 0 0)))))$
 $(* 5 (* 5 (* 3 (* 2 (* 1 1))))$

ABSTRACTA SYNTAKYCZNA: OPERATOR QUOTE["]
(CUDZYSTÓW)

ABSTRAKCE A SYNTAKTICKÝ CINT: OPERATOR "QUOTE"
("CUDZÝSLOV")

(quote ($\lambda(x)x$))
 $\rightsquigarrow (\lambda(x)x)$

ABSTRACT & SYNTACTIC: OPERATOR QUOTE
(CUDZYSTOW)

(quote (λ (x) x))

\rightsquigarrow ($x(x)$ x)

(quote x)

\rightsquigarrow x

ABSTRAKCYJNA SYNTAXYCZNA: OPERATOR QUOTE^a
("CUDZYSTÓW")

(quote ($\lambda (x) x$))

\rightsquigarrow ($\lambda (x) x$)

(quote x)

\rightsquigarrow x

OBSERWACJA: Z SYNTAXYCZNOU PUNKTU
WIDZIENIU, PROGRAMU TO TYLKO ~~SYMBOLI~~
POZAKONIECZNE LISTY SYMBOLI

CZY PROGRAMOWANIE JEST
LITERATURA?

CZY PROGRAMOWANIE JEST LITERATURĄ?

— DUŻO DEFINICJI,
MAŁO STWIERDZEŃ

CZY PROGRAMOWANIE JEST LITERATURA?

- DUŻO DEFINICJI,
MAŁO STWIERDZEŃ
- BARDZO DUŻE ZNACZENIE
SZCZEGÓLNU (WIĘKSZE
NIŻ W POWIEŚCIACH
KRYMINALNYCH!)

JAK SPRAWIĆ, BY PROGRAMOWANIE
STAŁO SIE LITERATURĄ?

JAK SPRAWIĆ, BY PROGRAMOWANIE
STAŁO SIĘ LITERATURĄ?

- MÓWIĆ TO SAMO NA WIELE
SPOSOBÓW

;; Powiemy, że s jest podtablicą r wtedy, gdy istnieją takie i oraz j,
;; że dla każdego $n \in [0, (\text{height } r))$ oraz dla każdego $m \in [0, (\text{width } r))$,
;; $r[i+n][j+m] = s[n][m]$. Wówczas parę (i, j) nazwiemy współrzędnymi
;; podtablicy s w tablicy r.

```
(define (subrect-indices #:of subrect #:in rect)
  (let ((w (rect-width subrect))
        (h (rect-height subrect)))
    (filter (lambda ((left top))
              (rect-match? (take-subrect rect left top w h) subrect))
            (cart (iota (- (rect-width rect) w -1))
                  (iota (- (rect-height rect) h -1))))))

(define (displacement #:of figure #:from source #:to dest)
  (assert (and (rect? source)
                (rect? dest)))
  (let ((dest-position (subrect-indices #:of `((,figure)) #:in dest))
        (source-position (subrect-indices #:of `((,figure)) #:in source)))
    (if (or (null? source-position) (null? dest-position))
        #f
        (map - (match dest-position ((x) x))
              (match source-position ((x) x))))))

(e.g. (displacement #:of '▲ #:from '(_ ?)
                      (? ?)
                      (? ▲)) #:to '((▲ ?)
                                     (? ?)
                                     (? _))) ==> (-1 -2))
```

JAK SPRAWIĆ, BY PROGRAMOWANIE STAŁO SIĘ LITERATURĄ?

- MÓWIĆ TO SAMO NA WIELE SPOSOBÓW
- REDUNDANCJA: ASERCJE I TESTY JEDNOSTKOWE

JAK SPRAWIĆ, BY PROGRAMOWANIE STAŁO SIĘ LITERATURĄ?

- MÓWIĆ TO SAMO NA WIELE SPOSOBÓW
- REDUNDANCJA: ASERCJE I TESTY JEDNOSTKOWE
- KOMENTARZE I DOCSTRNGI

JAK SPRAWIĆ BY PROGRAMOWANIE
STAŁO SIĘ LITERATURĄ?

- MÓWIĆ TO SAMO NA WIELE SPOSOBÓW
- REDUNDANCJA: ASERCJE I TESTY JEDNOSTKOWE
- ~~KOMENTARZE I DOCSSTRINGI~~

3 Recursively in Inform 7

(It reminds you of COBOL because it's for writing text adventures; proportional font is deliberate):

To decide what number is the factorial of (n - a number):

If n is zero, decide on one;

otherwise decide on the factorial of (n minus one) times n.

If you want to actually call this function ("phrase") from a game you need to define an action and grammar rule:

"The factorial game" [this must be the first line of the source]

There is a room. [there has to be at least one!]

Factorialing is an action applying to a number.

Understand "factorial [a number]" as factorialing.

Carry out factorialing:

Let n be the factorial of the number understood;

Say "It's [n]".

share

edited Sep 18 '08 at 9:12

community wiki

2 revs

Hugh Allen

show 2 more comments

JAK SPRAWIĆ BY PROGRAMOWANIE STAŁO SIĘ LITERATURĄ?

- NÓWIĆ TO SAMO NA WIELE SPOSOBÓW
- REDUNDANCJA: ASERCJE I TESTY JEDNOSTKOWE
- ~~KOMENTARZE → DOCSSTRINGS!~~
- PROGRAMOWAĆ LITERATURĘ!

JĘZYK SCHEMĘ
- RACHUNEK X Z (ONAWIASOWANA)
NOTACJI POLSKIEJ

PYTANIA?

godek.maciek@gmail.com

@PawelGodek