Stefan Sokołowski

JĘZYKI FORMALNE I METODY KOMPILACJI

wykład 10

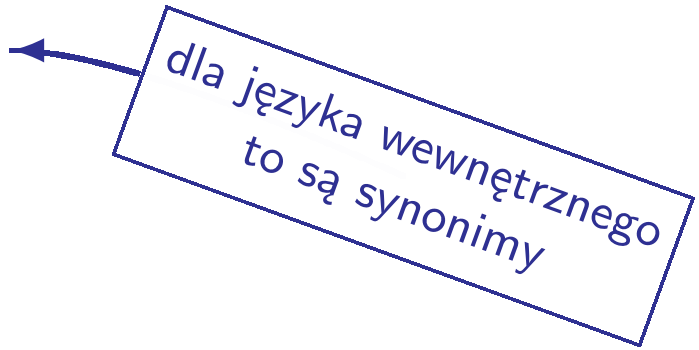
Inst. Informatyki Stosowanej, ANS
Elbląg, 2024/2025

Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

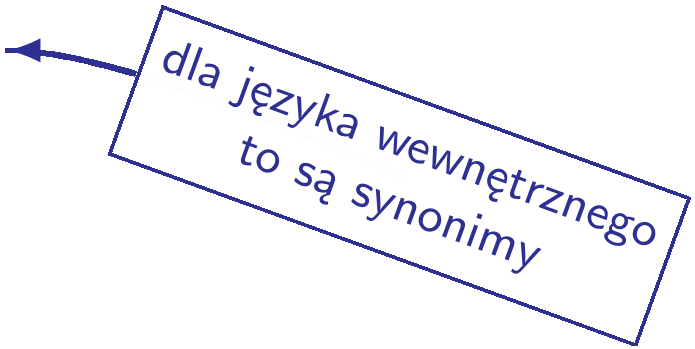
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...



Duży program tworzony po kawałku

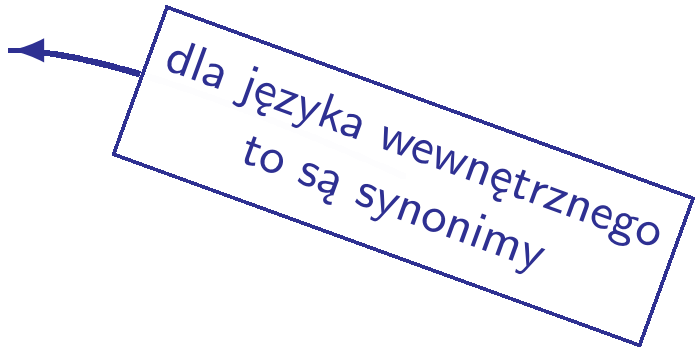
Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...



Jak program współpracuje z funkcją?

Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...



Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami

Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty

Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi

Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.

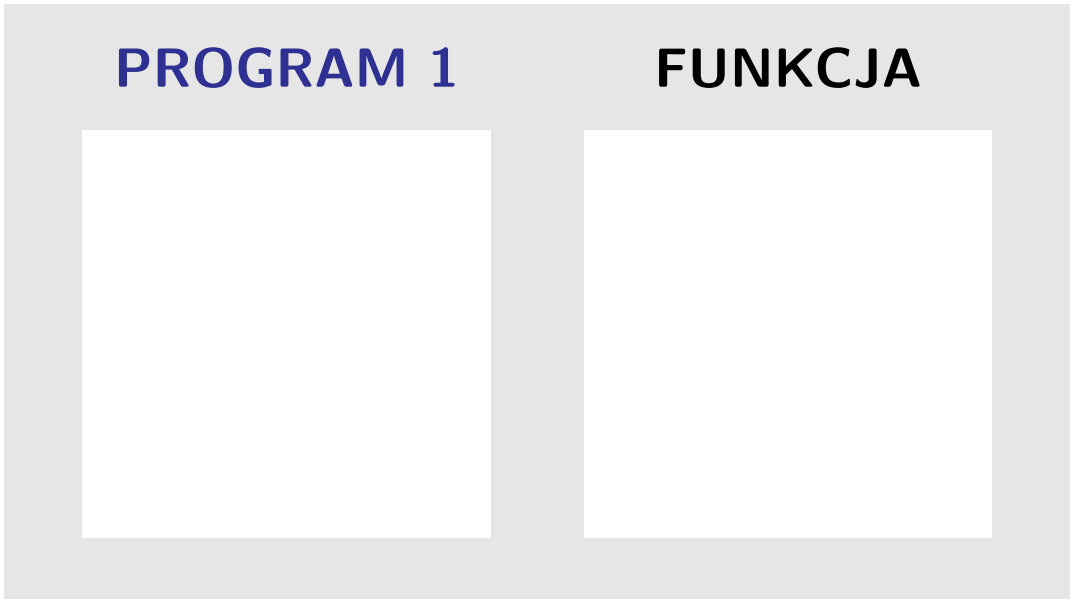
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



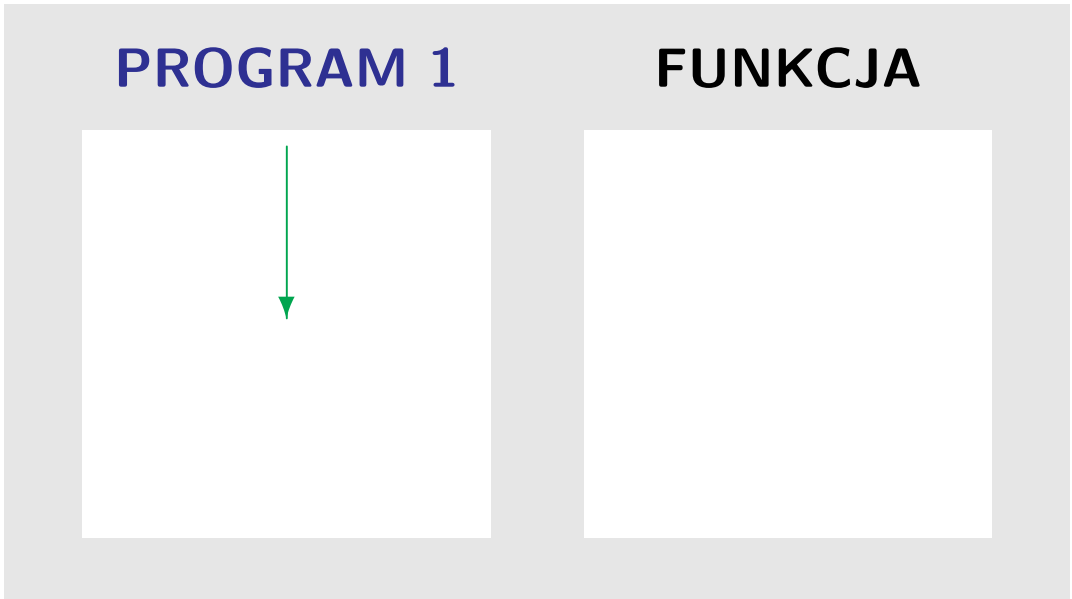
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



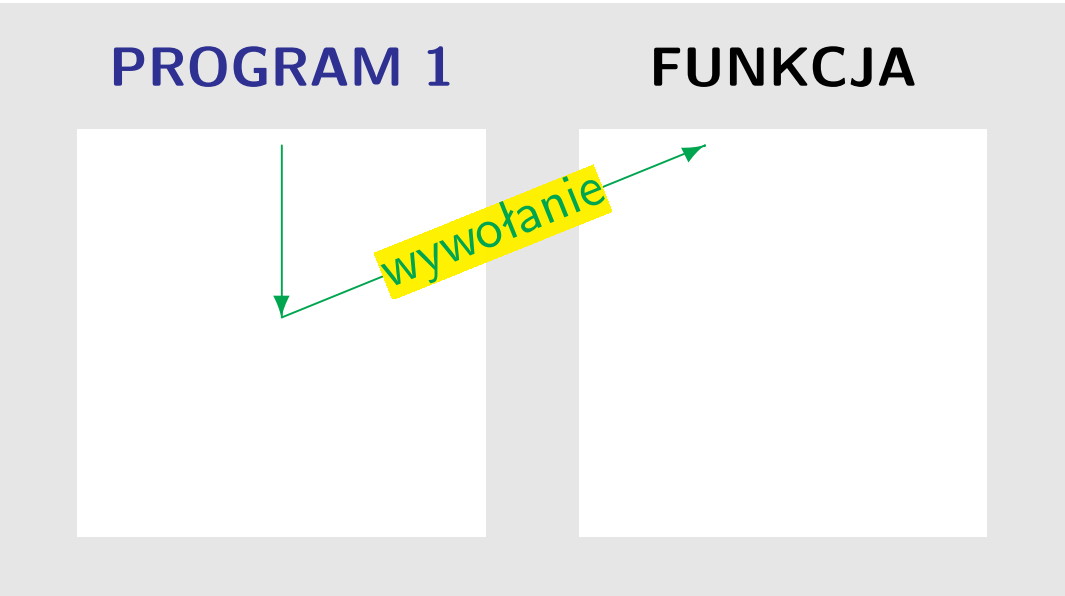
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



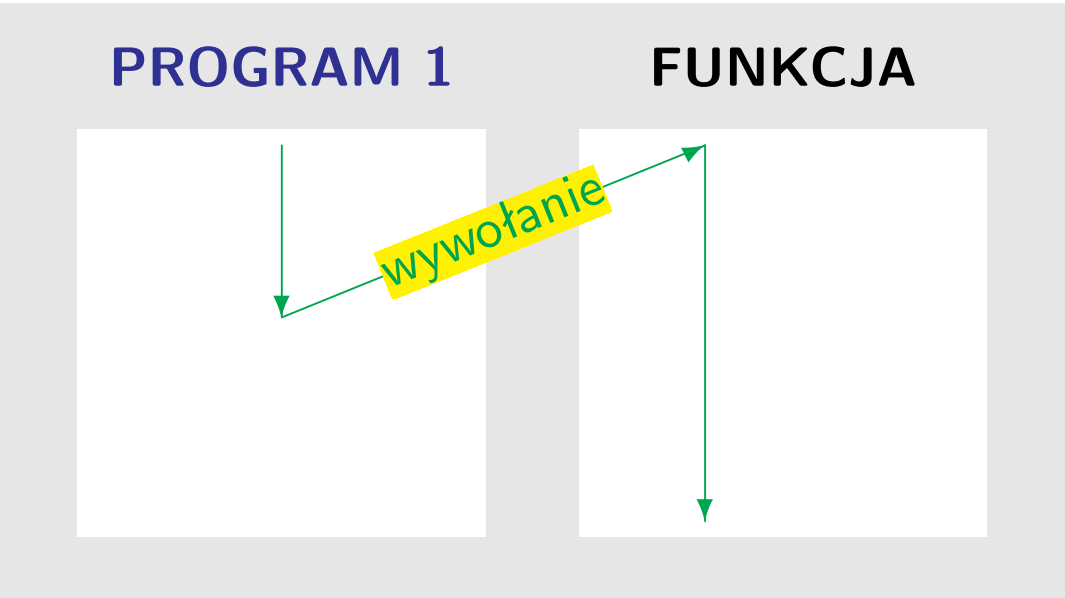
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



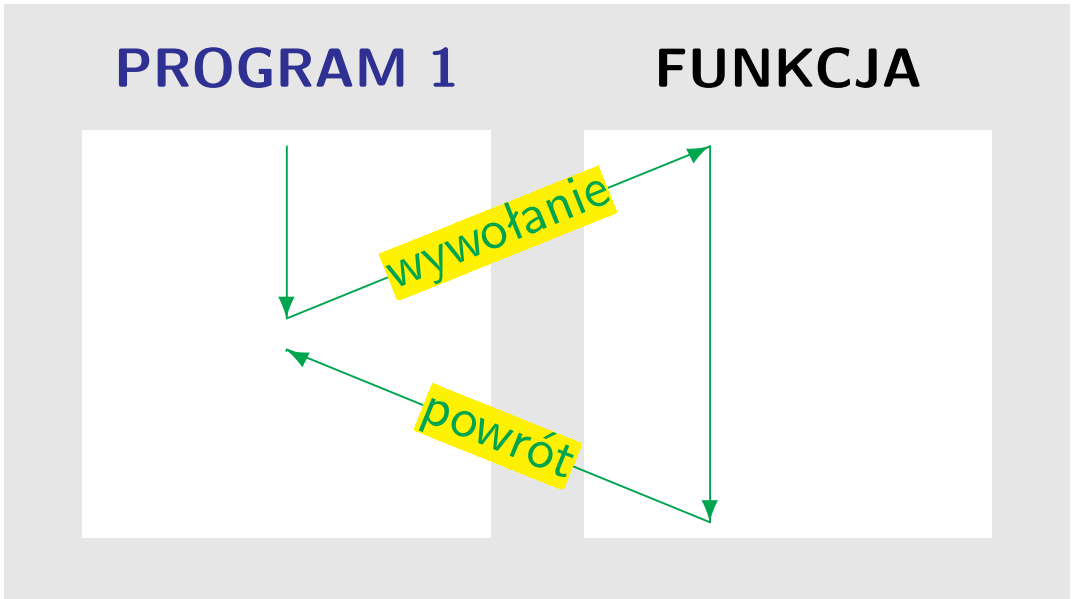
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



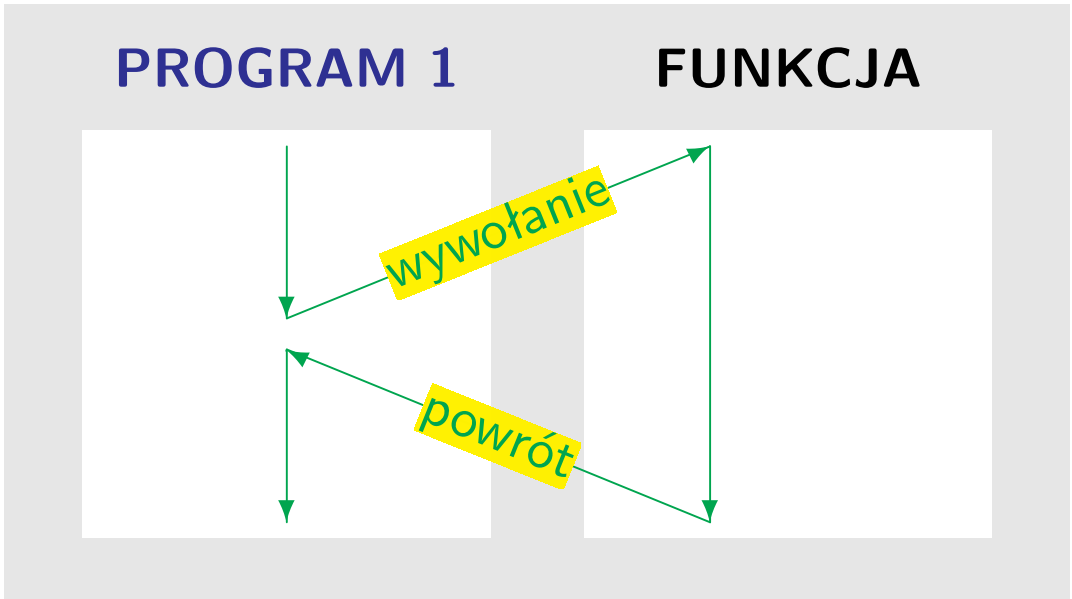
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



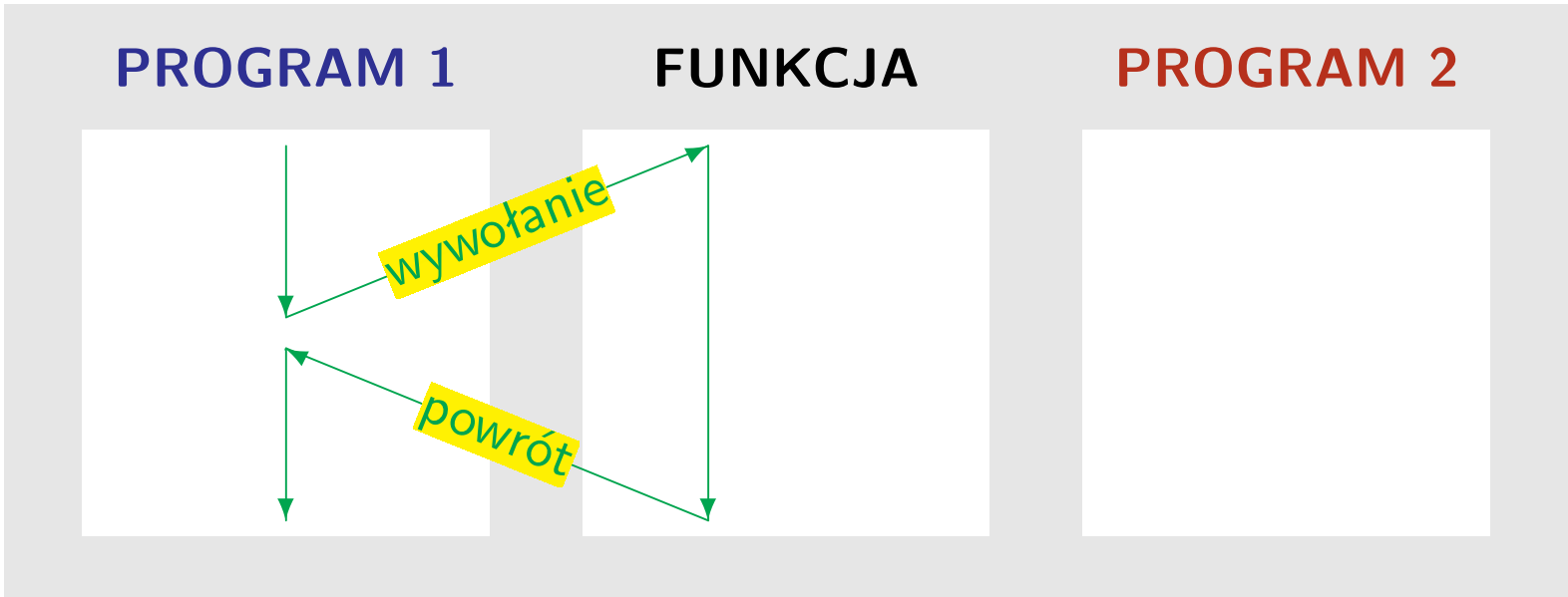
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



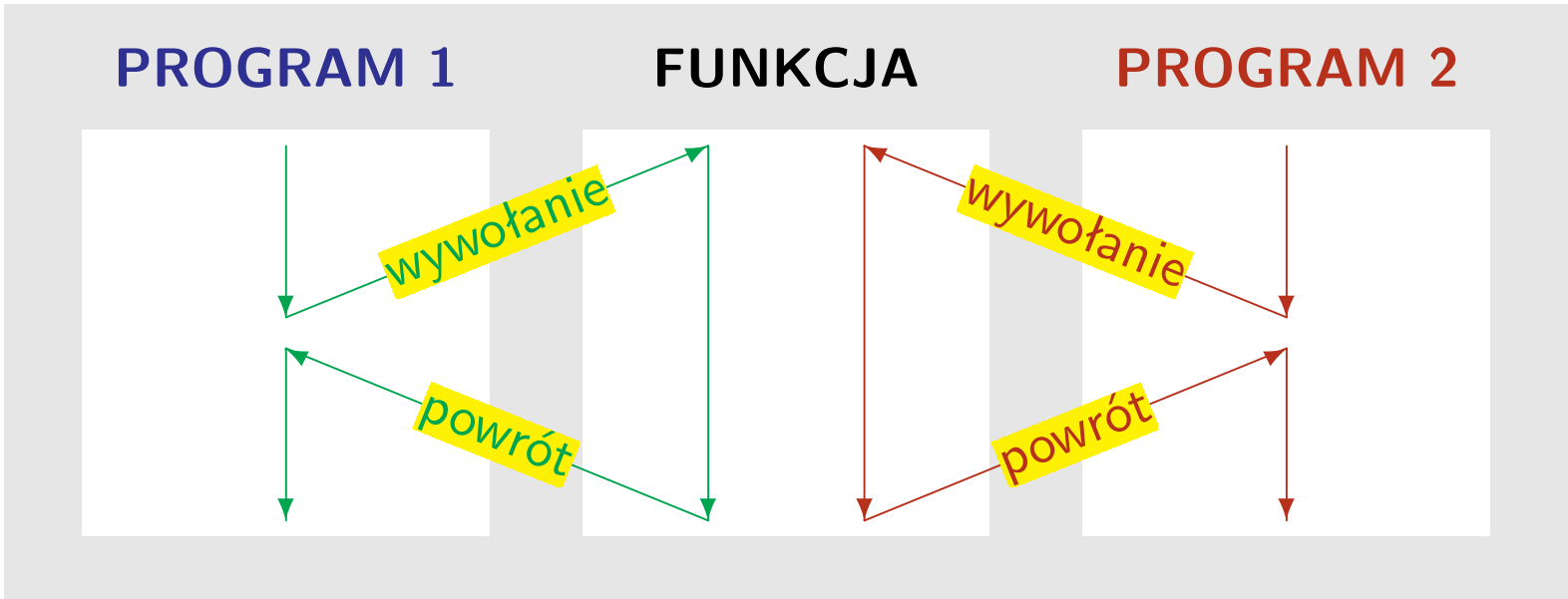
Duży program tworzony po kawałku

Duże programy **muszą** składać się ze współpracujących kawałków/podprogramów/procedur/funkcji/modułów/...

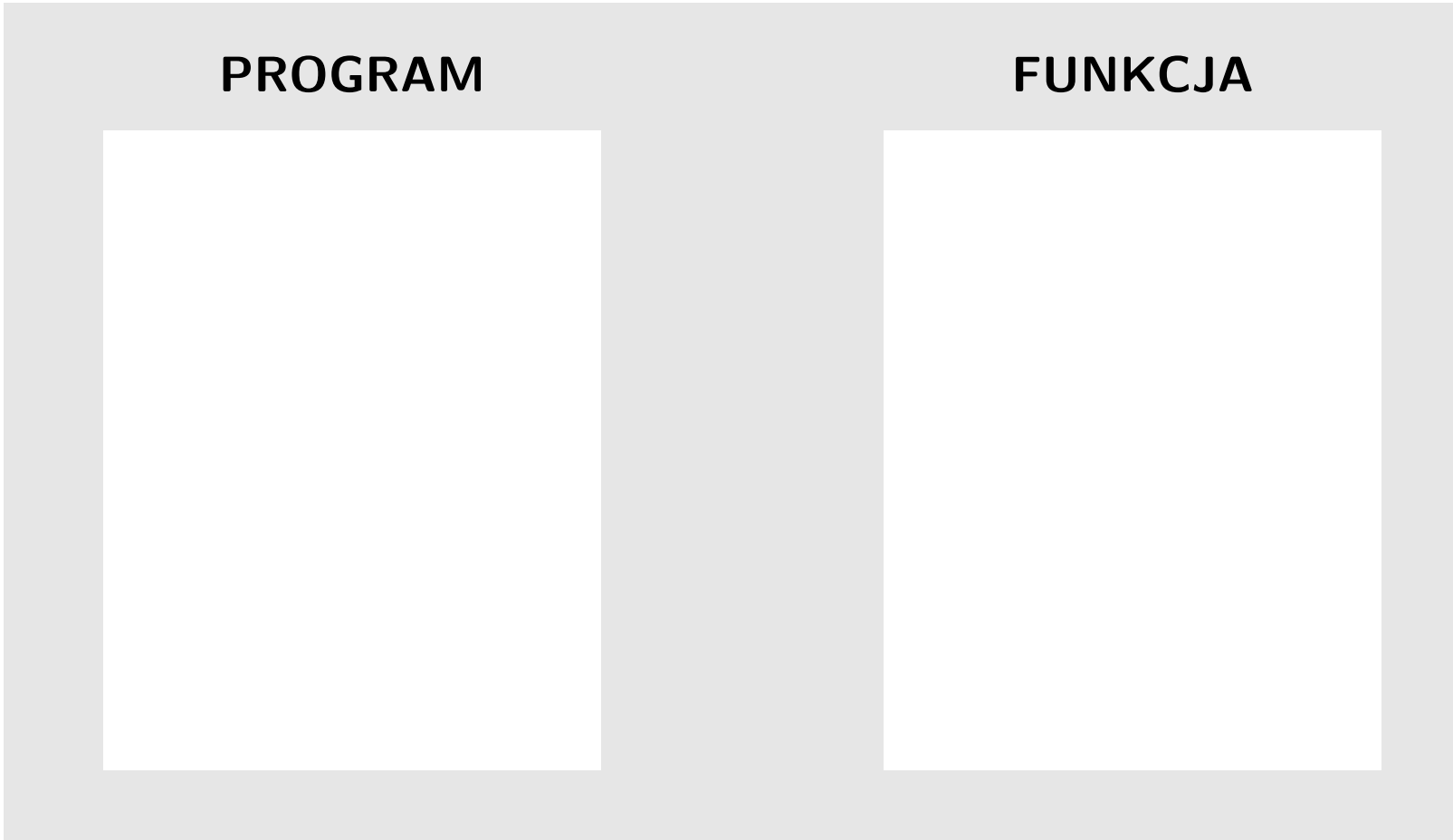
dla języka wewnętrznego to są synonimy

Jak program współpracuje z funkcją?

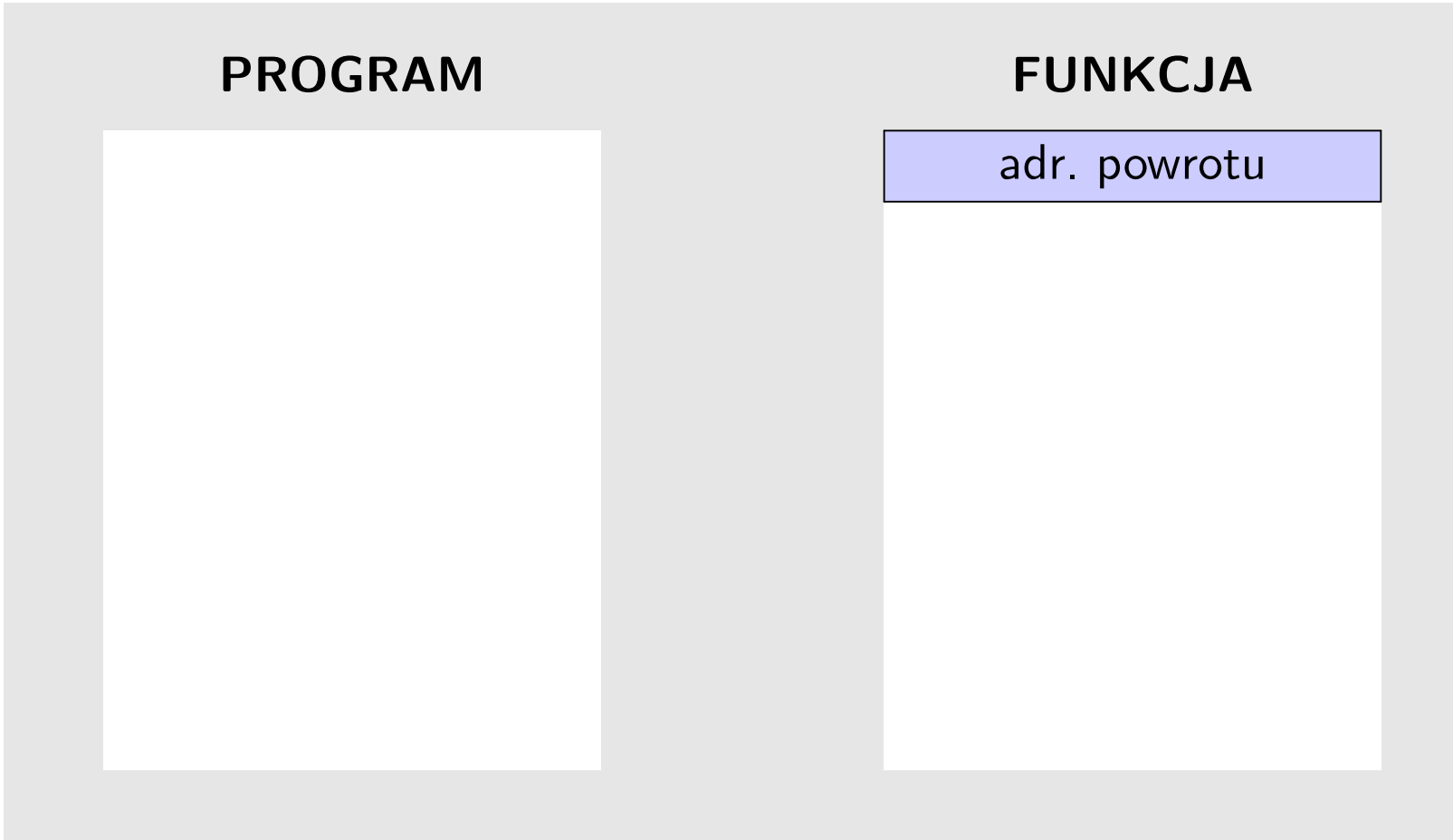
- funkcja współdziała z różnymi programami,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi,
- po zakończeniu działania funkcji, sterowanie wraca tam, skąd przyszło, czyli dla każdego wywołującego programu gdzie indziej.



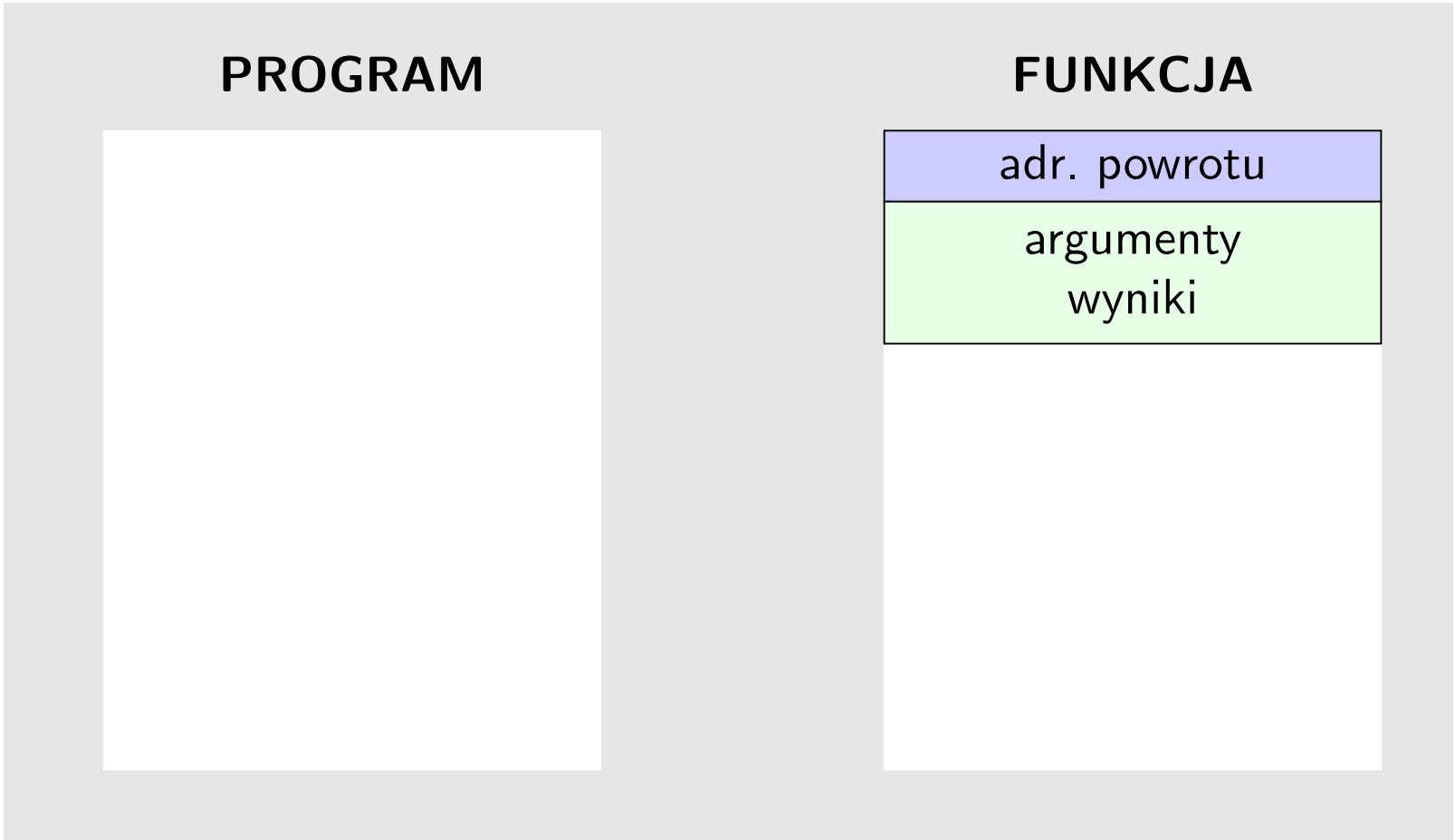
Duży program tworzony po kawałku



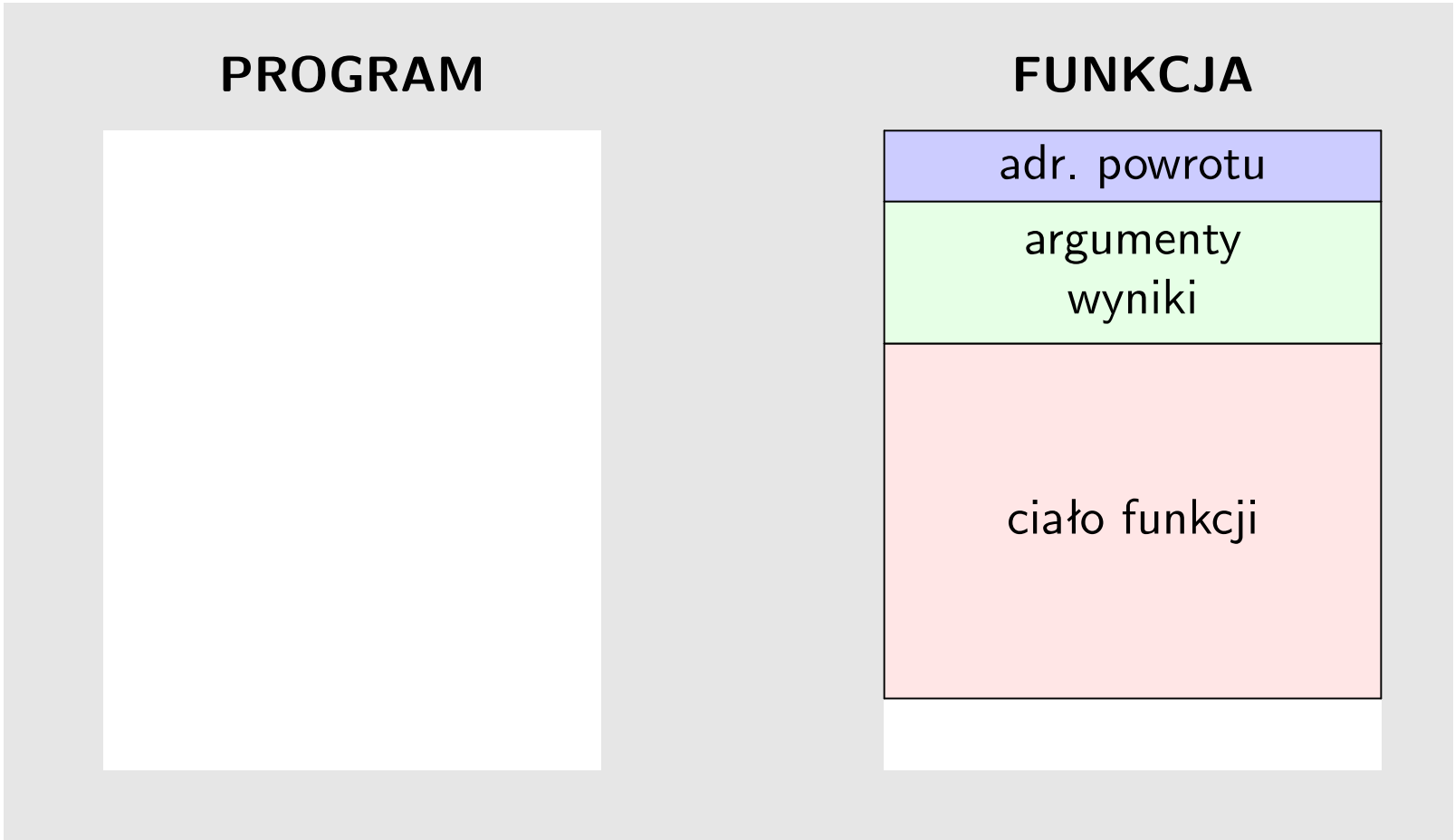
Duży program tworzony po kawałku



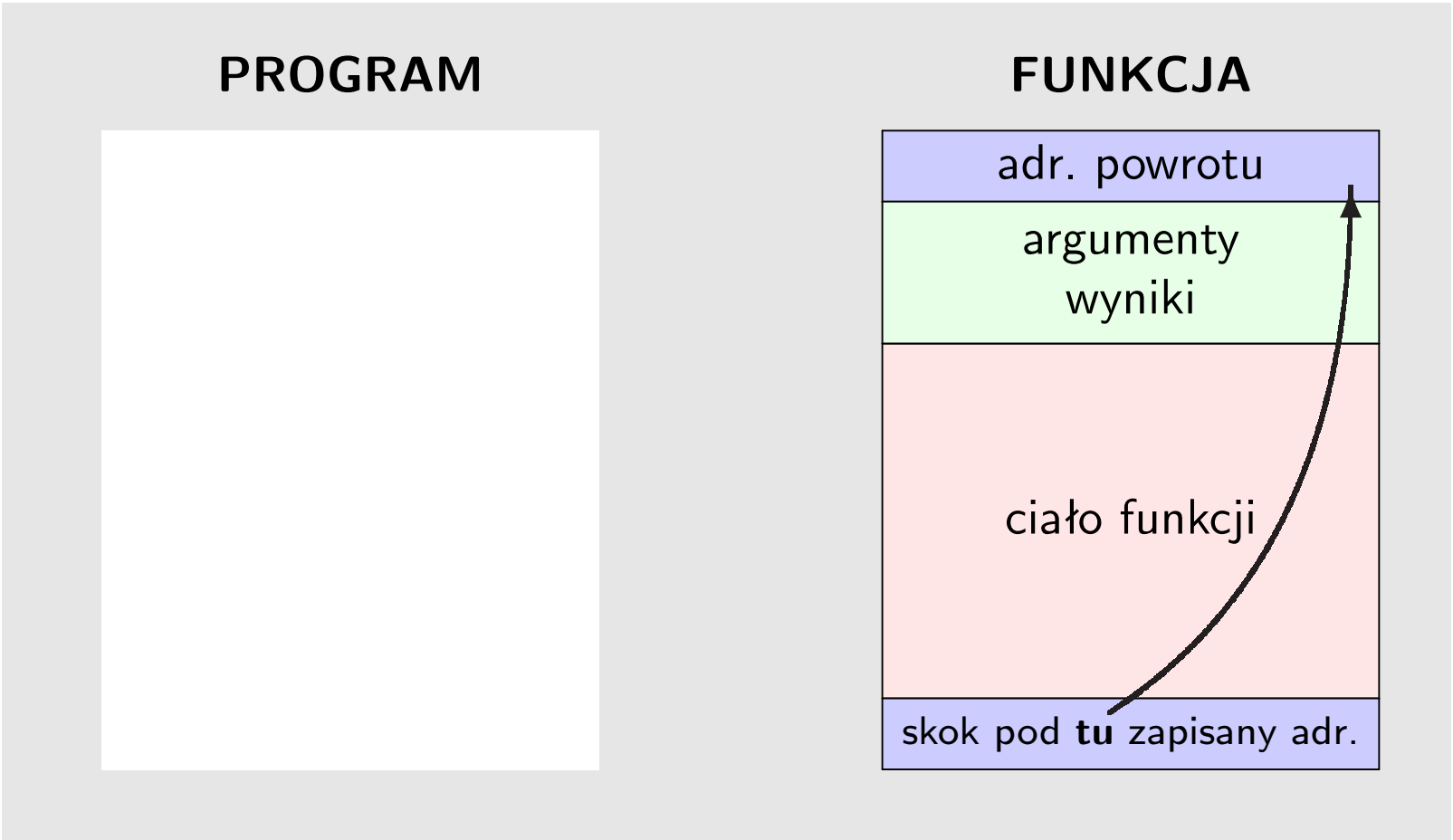
Duży program tworzony po kawałku



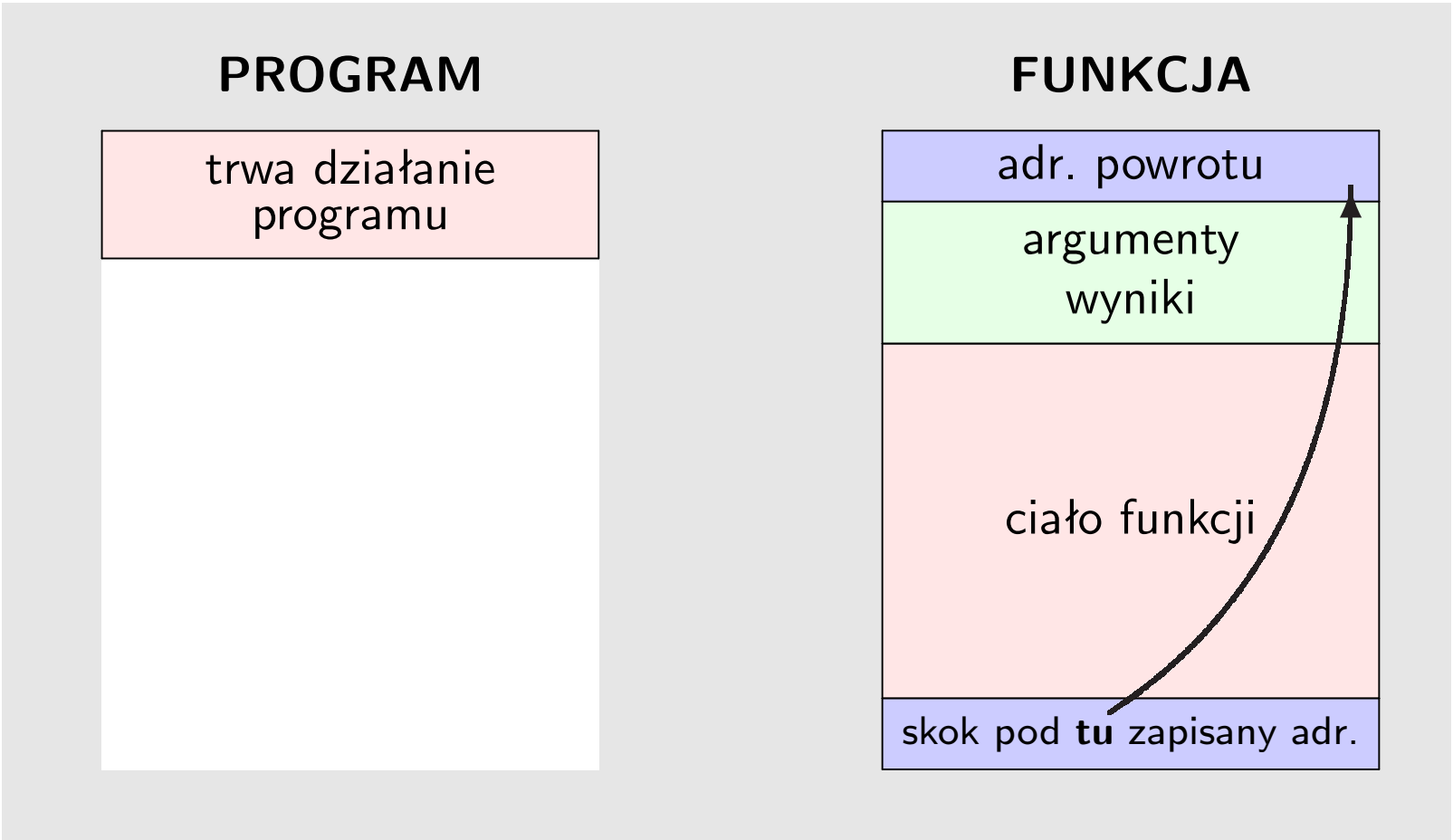
Duży program tworzony po kawałku



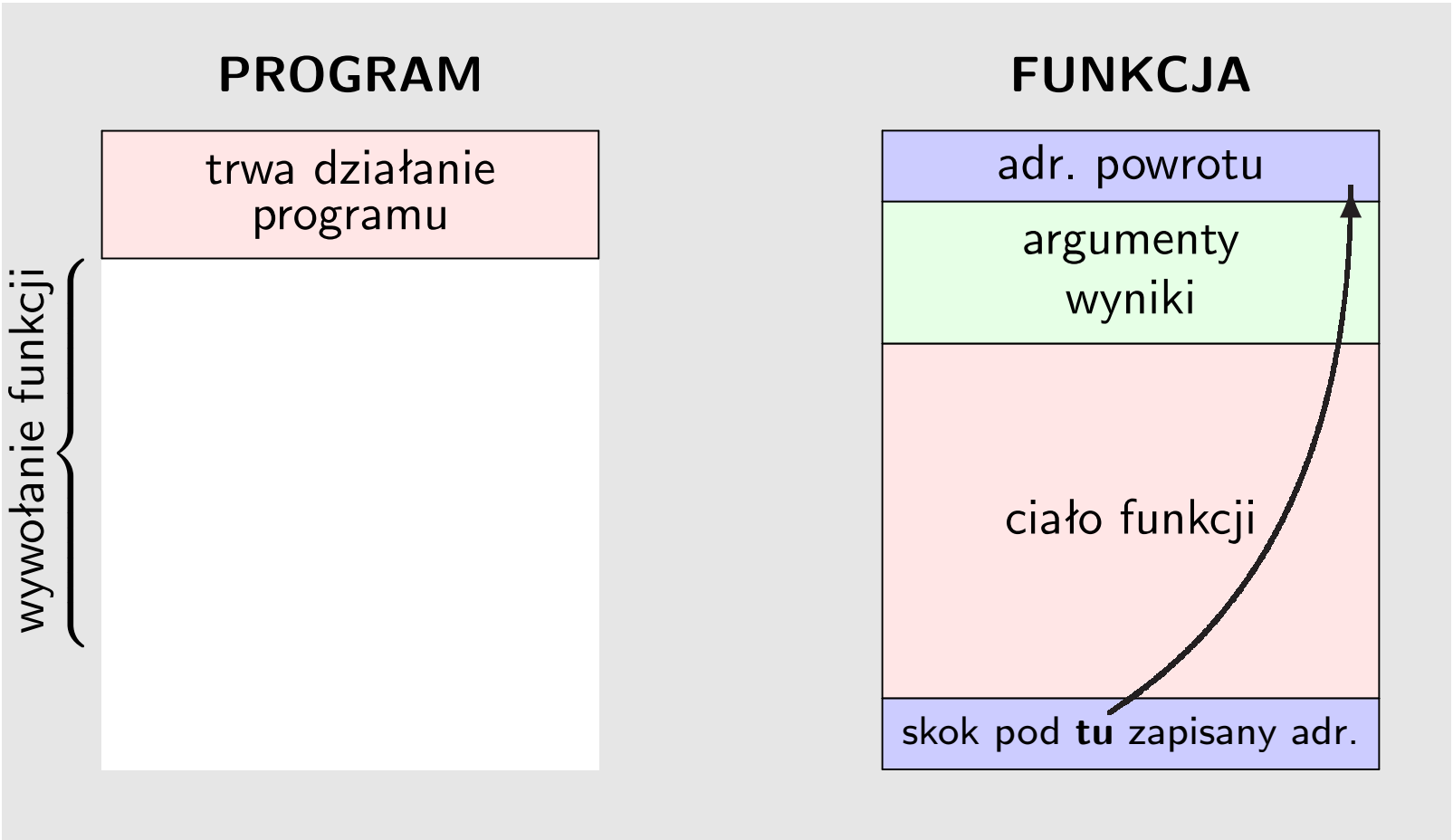
Duży program tworzony po kawałku



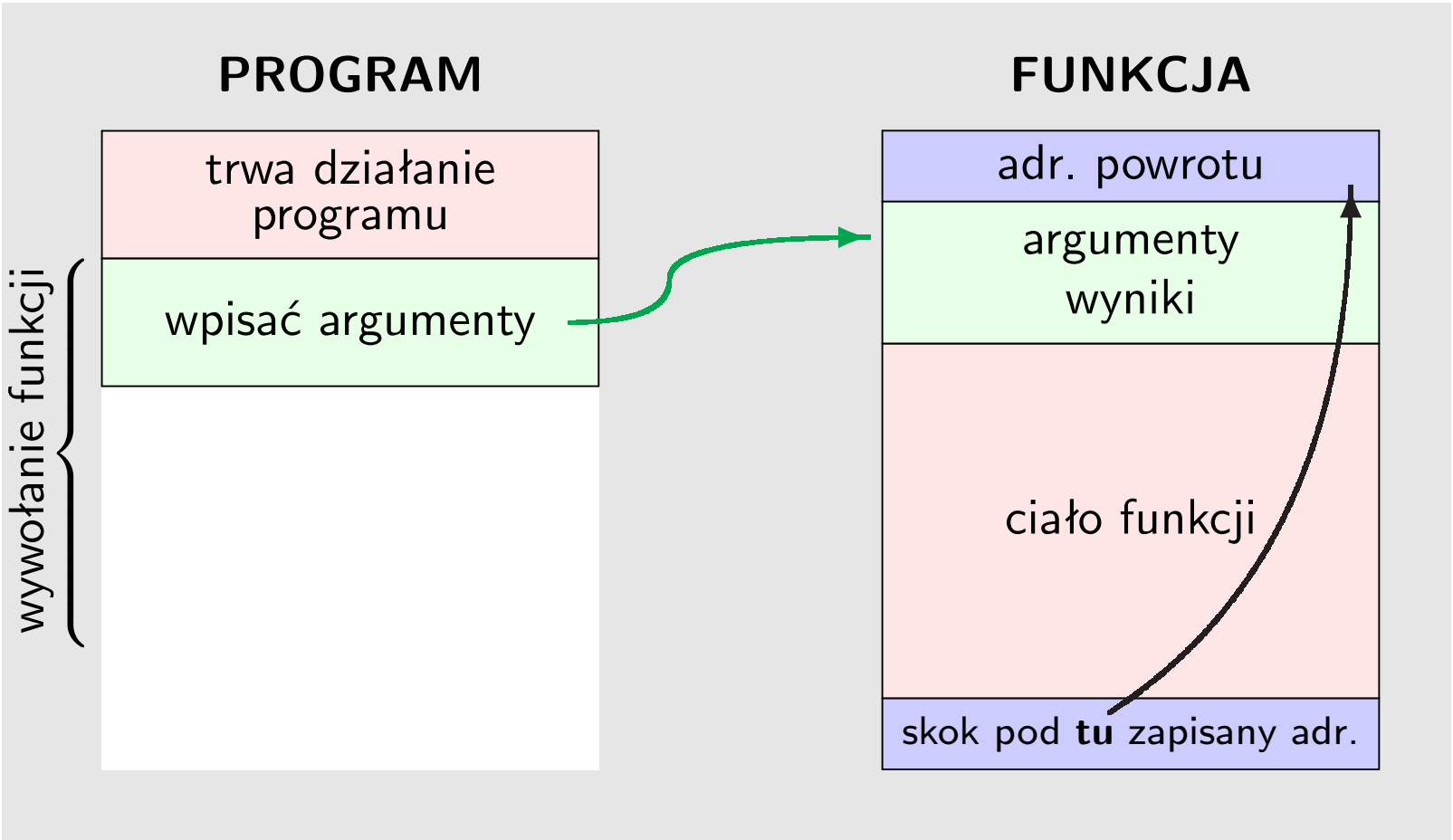
Duży program tworzony po kawałku



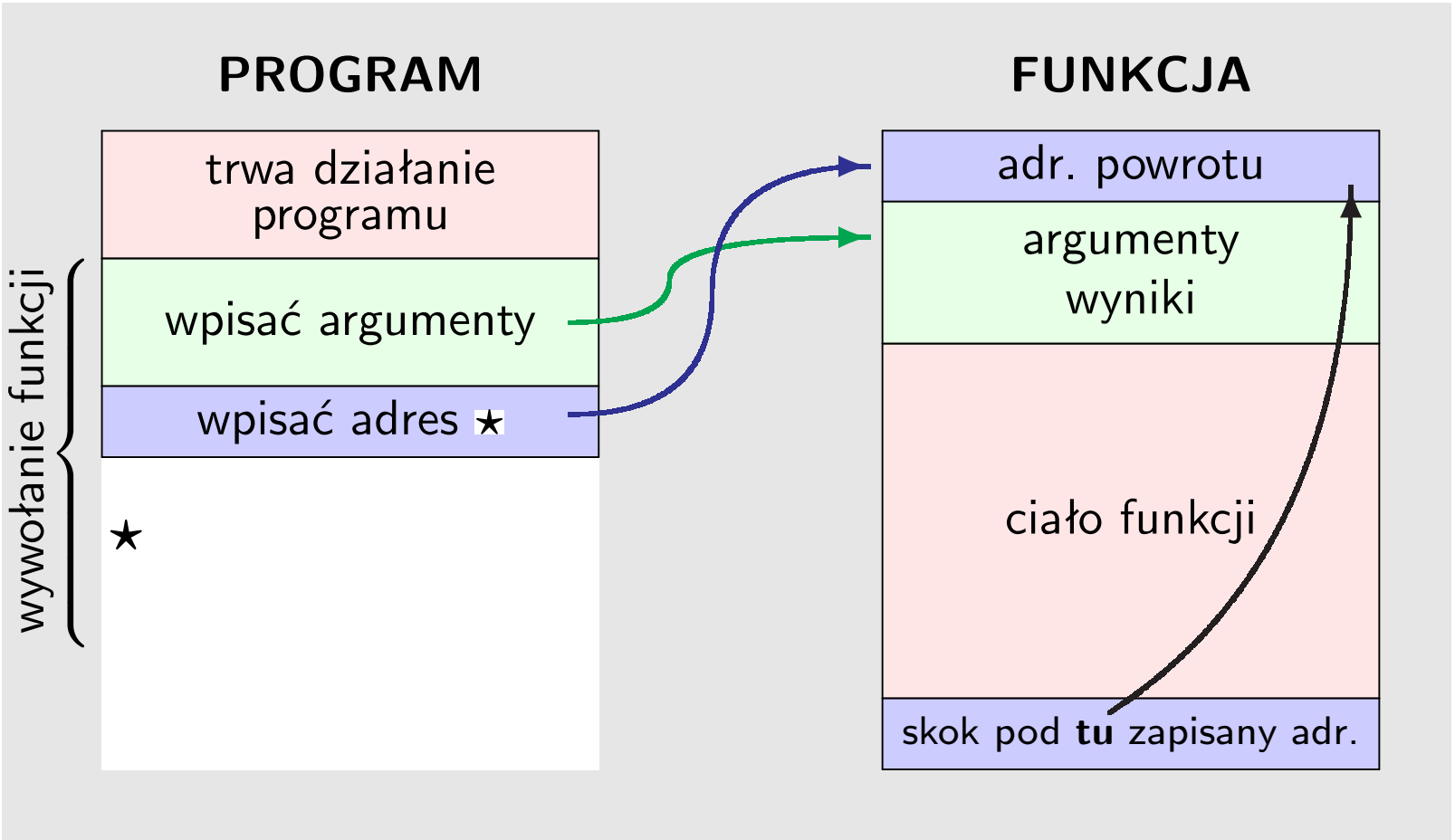
Duży program tworzony po kawałku



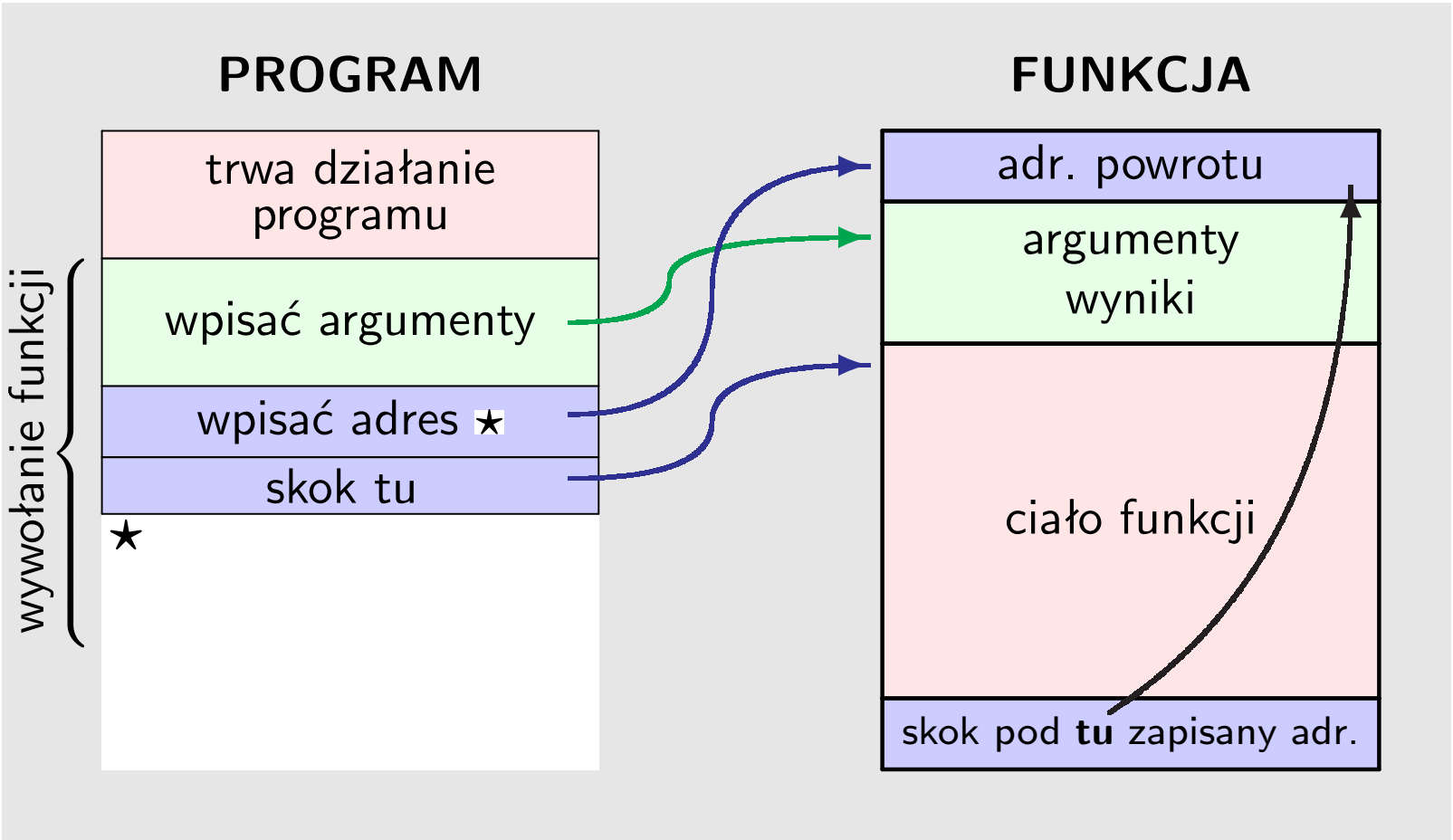
Duży program tworzony po kawałku



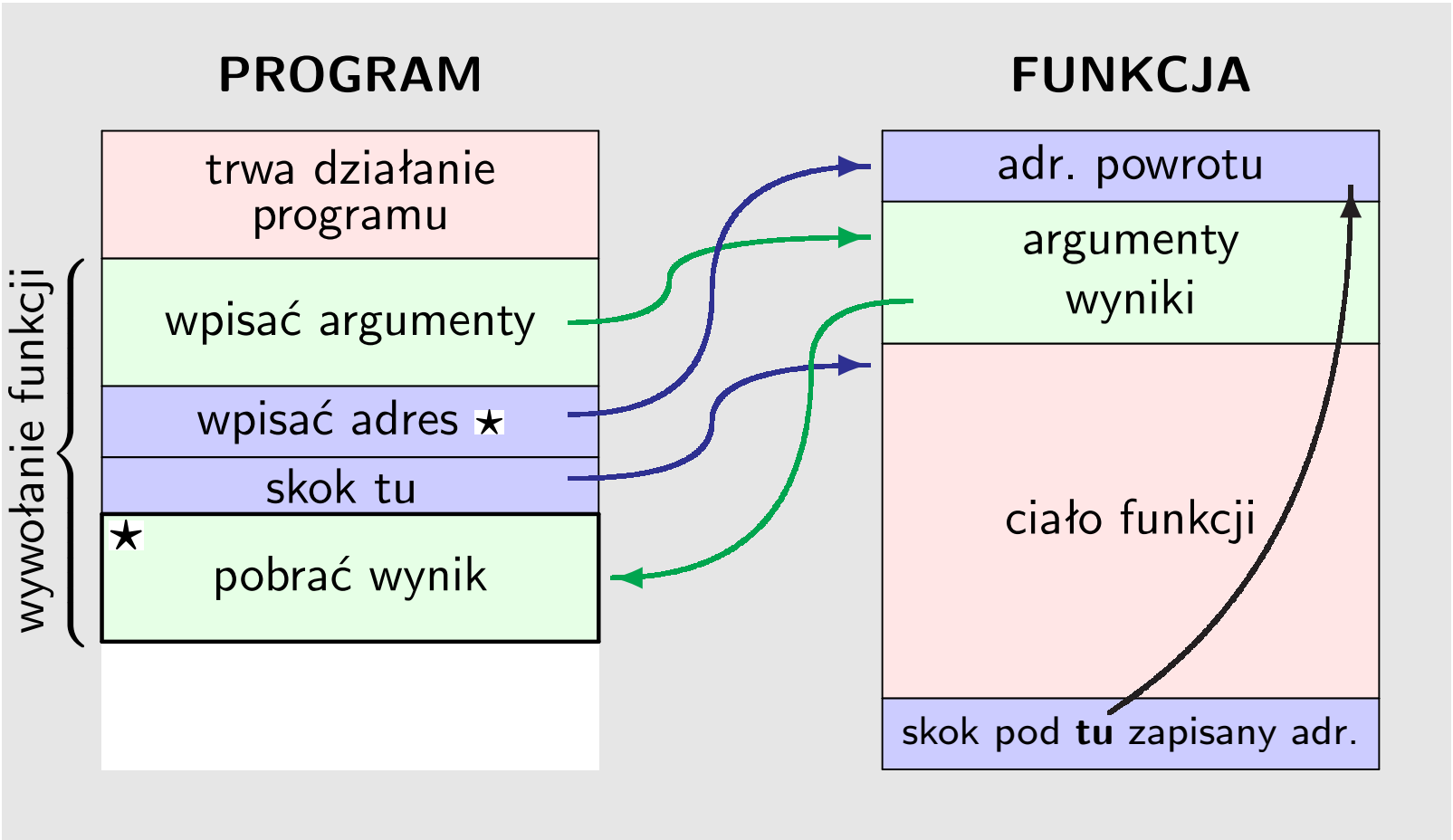
Duży program tworzony po kawałku



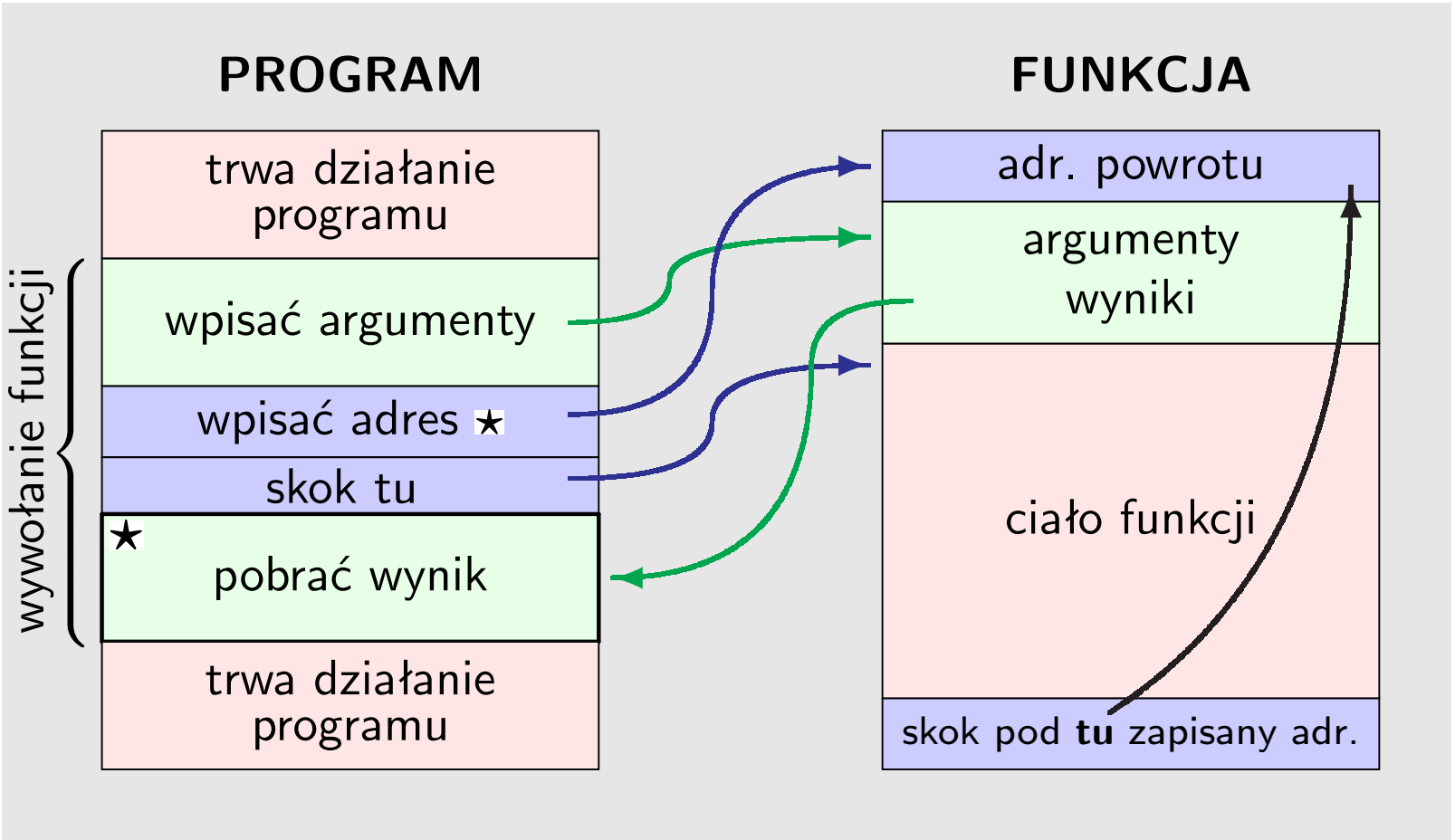
Duży program tworzony po kawałku



Duży program tworzony po kawałku



Duży program tworzony po kawałku



Duży program tworzony po kawałku

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

$$n = k \cdot q + r \quad \wedge \quad 0 \leqslant r < k$$

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

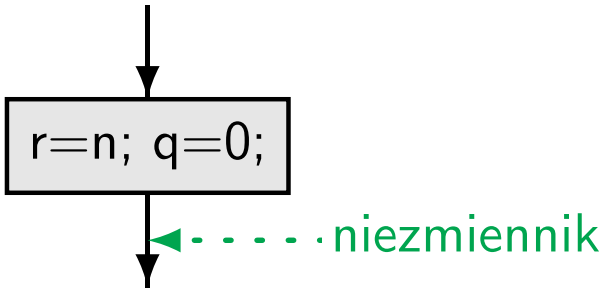
$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

zgodne z C dla $n, k > 0$
poza tym — hmmm...

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.



Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

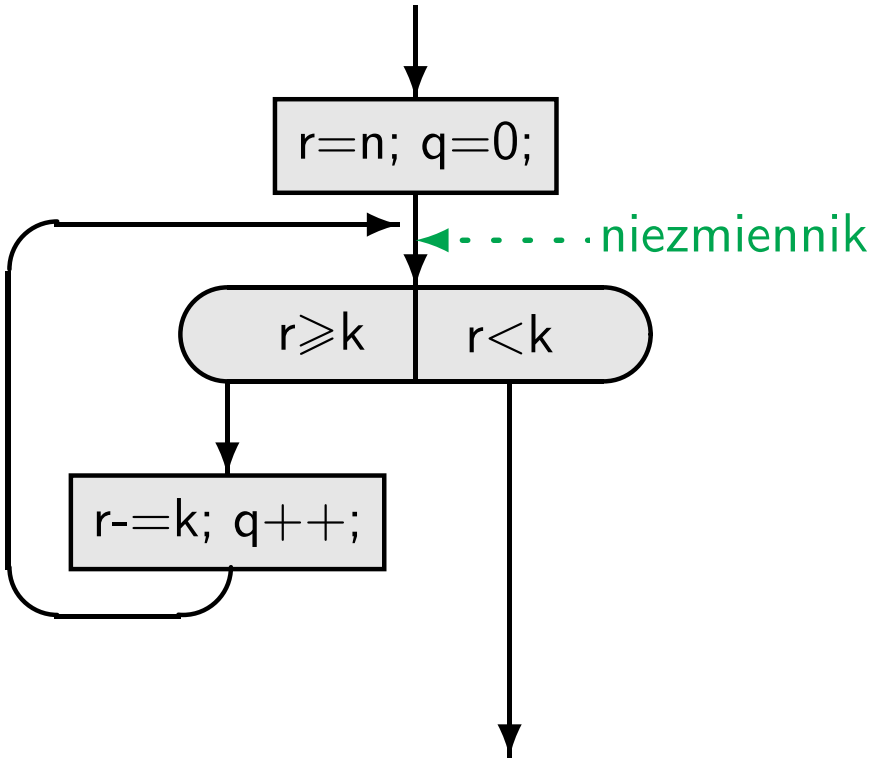
$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

zgodne z C dla $n, k > 0$
poza tym — hmmm...

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.



Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

zgodne z C dla $n, k > 0$
poza tym — hmmm...

Duży program tworzony po kawałku

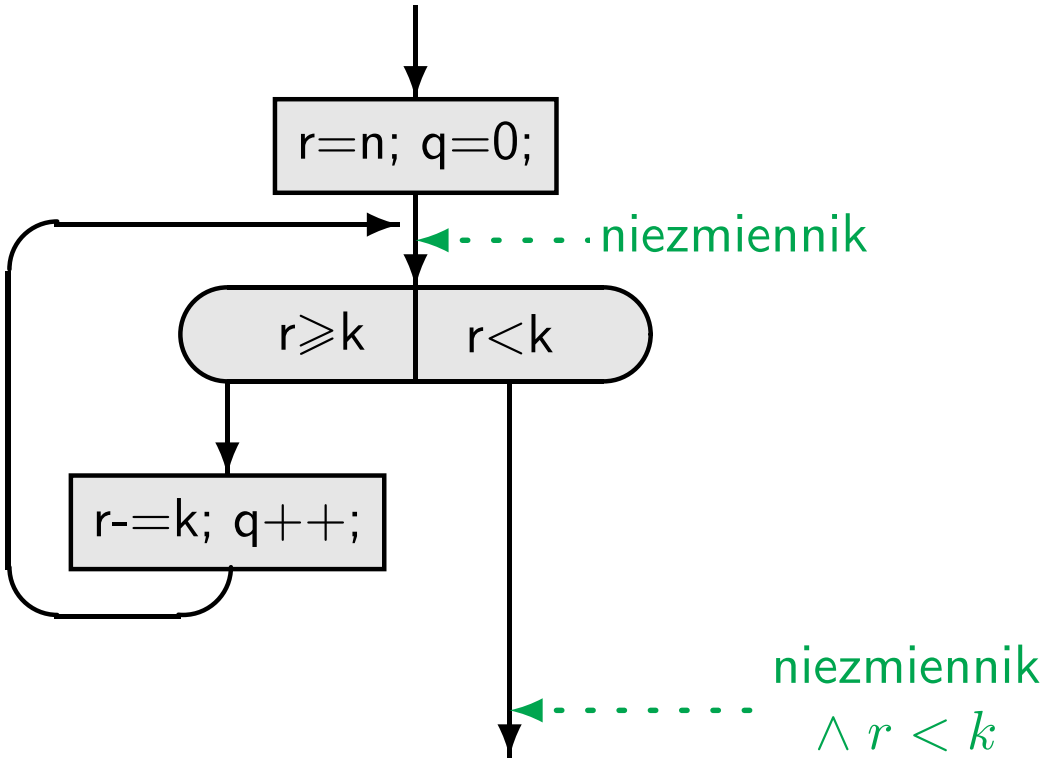
Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

zgodne z C dla $n, k > 0$
poza tym — hmmm...



Duży program tworzony po kawałku

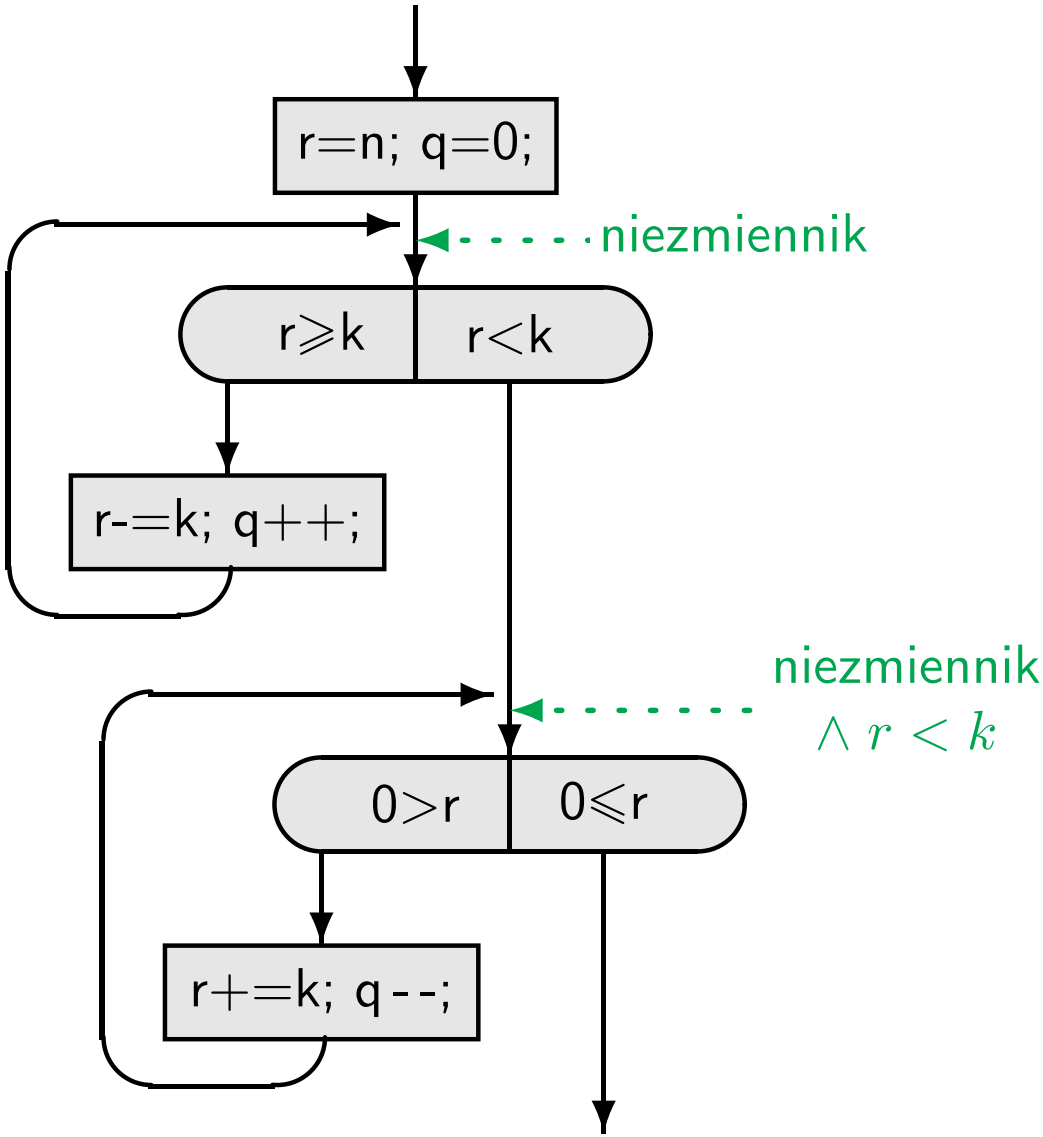
Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

zgodne z C dla $n, k > 0$
poza tym — hmmm...



Duży program tworzony po kawałku

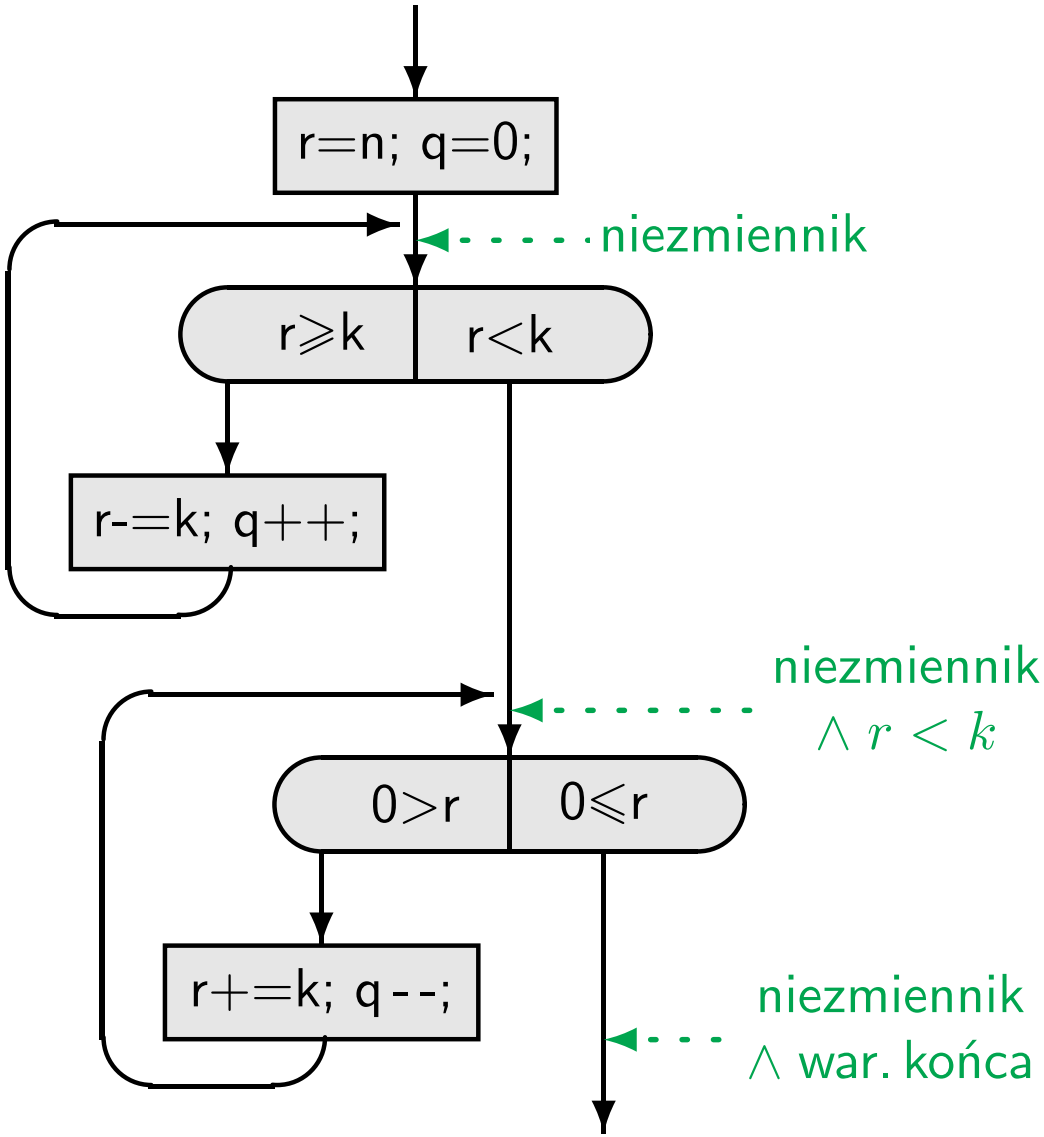
Kawałek 1: (dzielenie z resztą)

Wśród komend języka wewnętrznego **nie ma** dzielenia. Trzeba je zrealizować przez wielokrotne odejmowanie/dodawanie.

Dzielimy n przez k
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$
iloraz całkowity: q reszta: r
więc chcemy mieć:

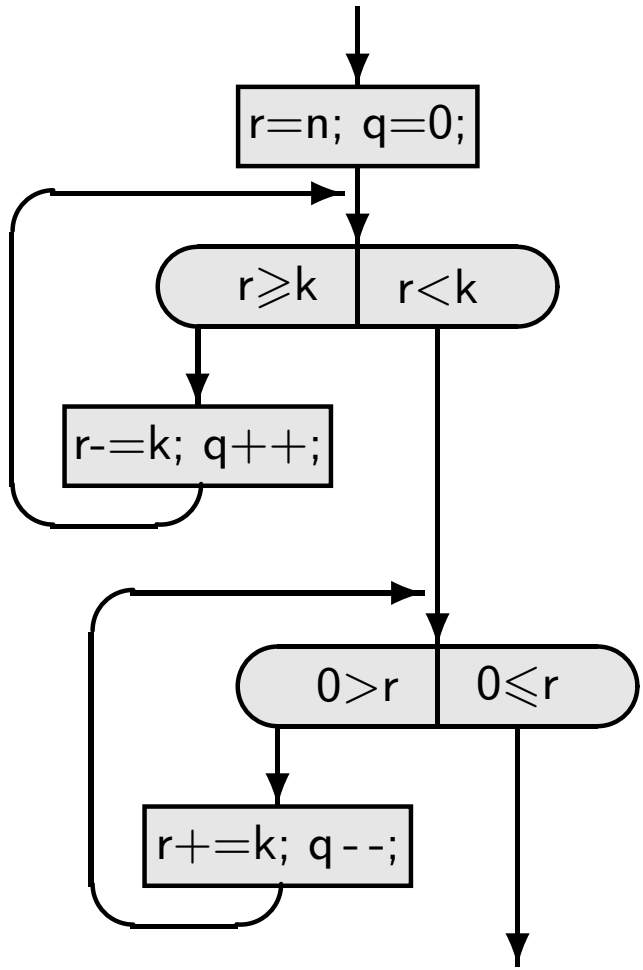
$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leqslant r < k}_{\text{war. końca}}$$

zgodne z C dla $n, k > 0$
poza tym — hmmm...



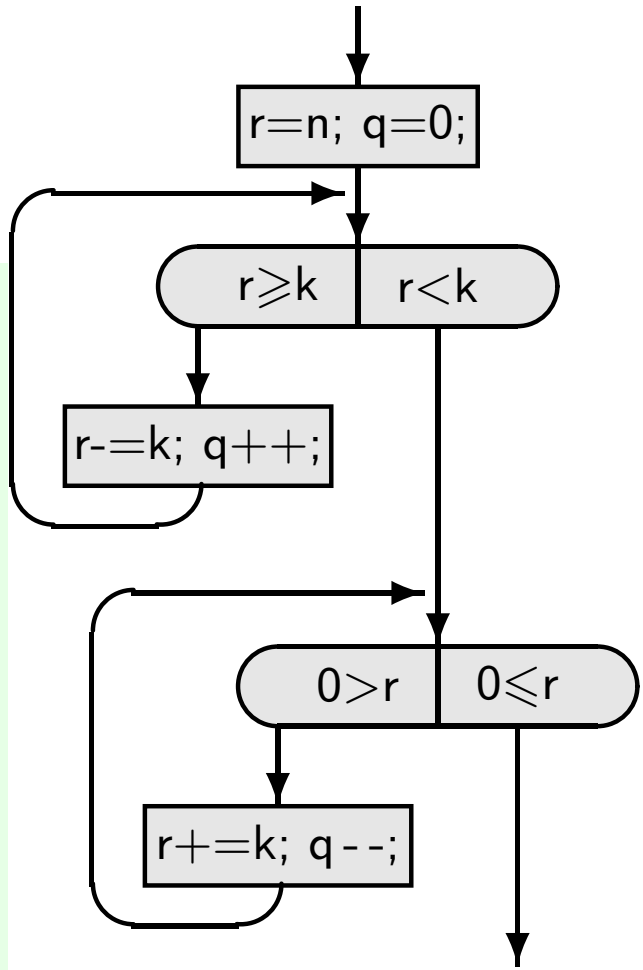
Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)



Duży program tworzony po kawałku

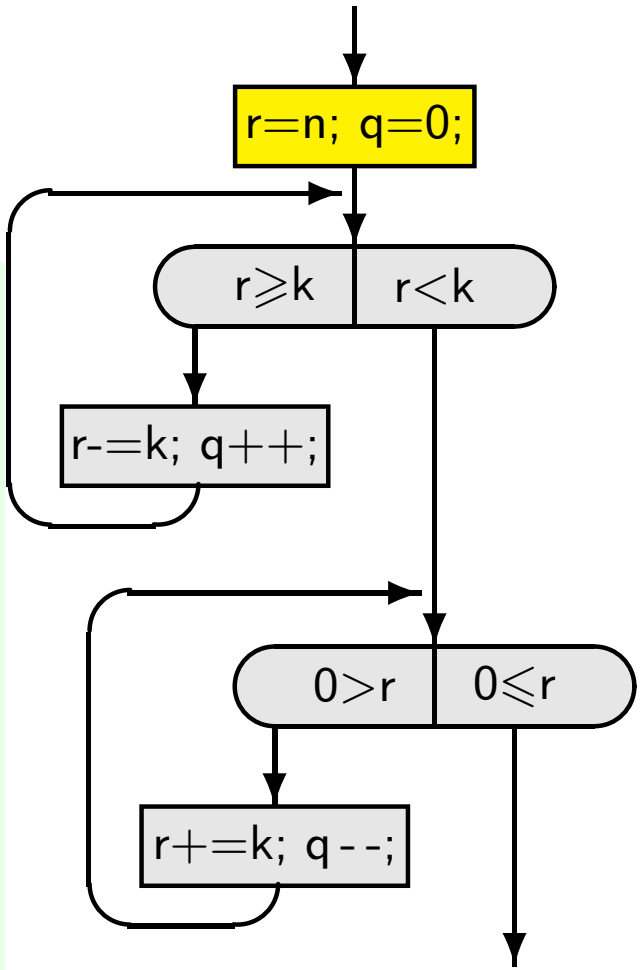
Kawałek 1: (dzielenie z resztą)



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

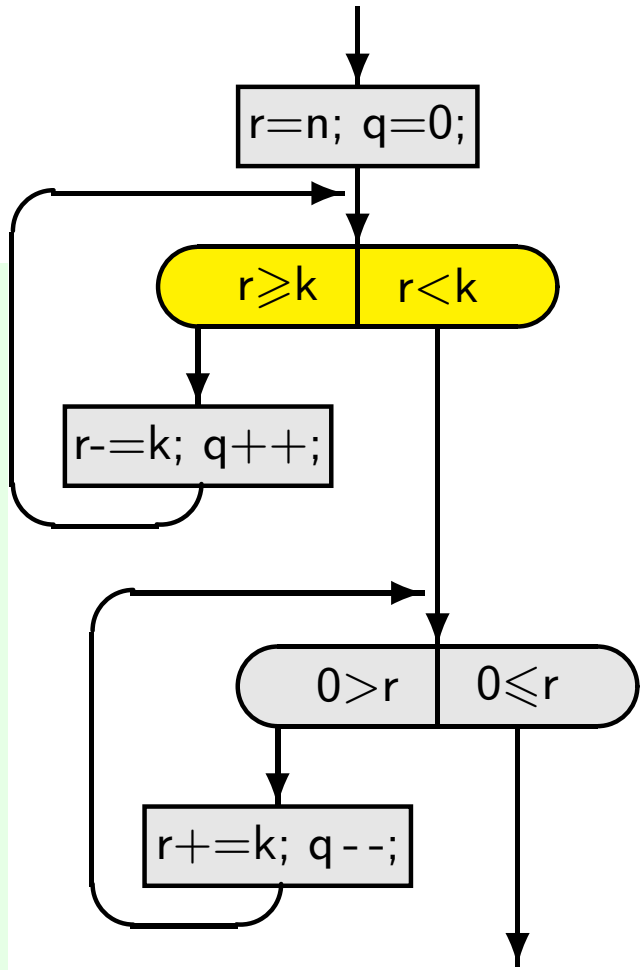
```
loada   n
store   r
loadn   0
store   q
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

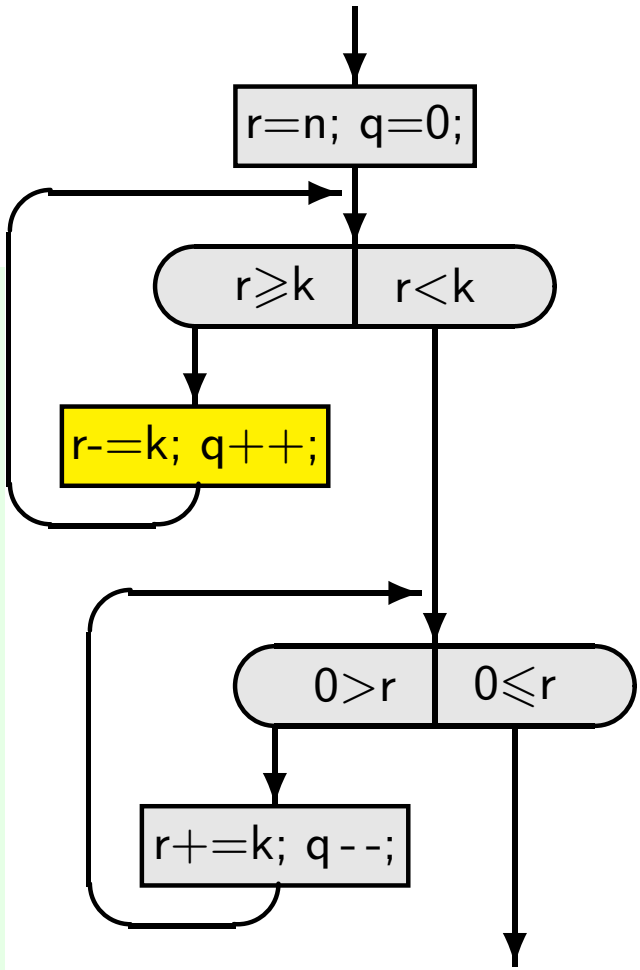
```
loada   n
store   r
loadn   0
store   q
loada   r # petla1
sub     k
mingoto petla2
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

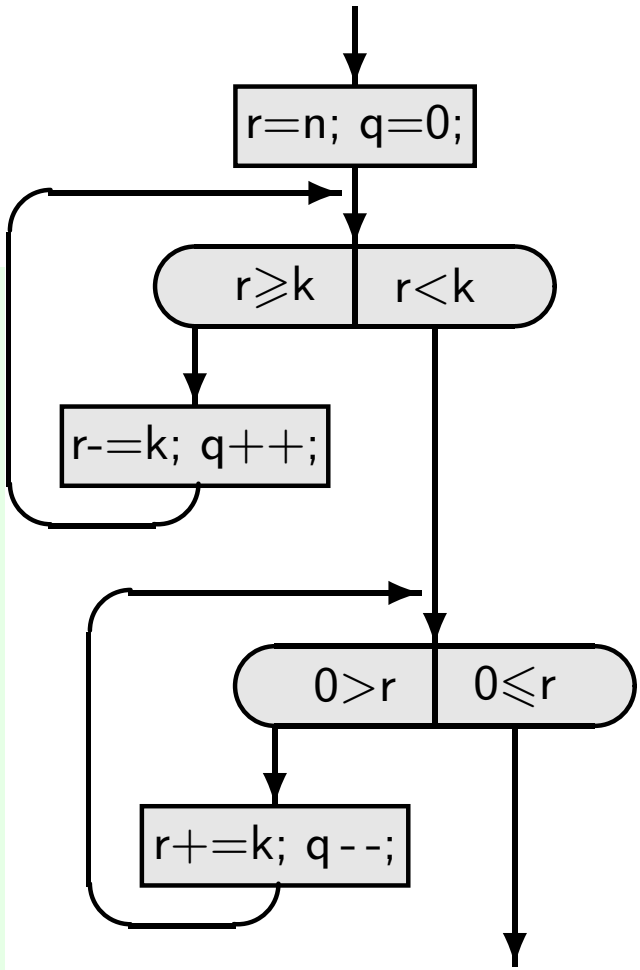
```
loada   n
store   r
loadn   0
store   q
loada   r # petla1
sub     k
mingoto petla2
loada   r
sub     k
store   r
incr    q
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

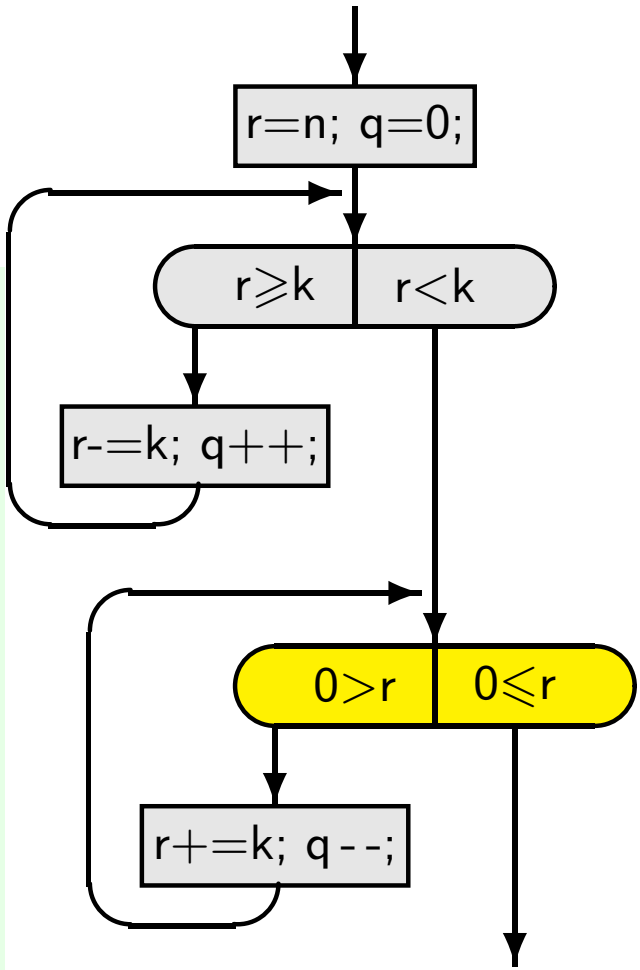
```
loada   n
store   r
loadn   0
store   q
loada   r # petla1
sub     k
mingotopetla2
loada   r
sub     k
store   r
incr    q
goto    petla1
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

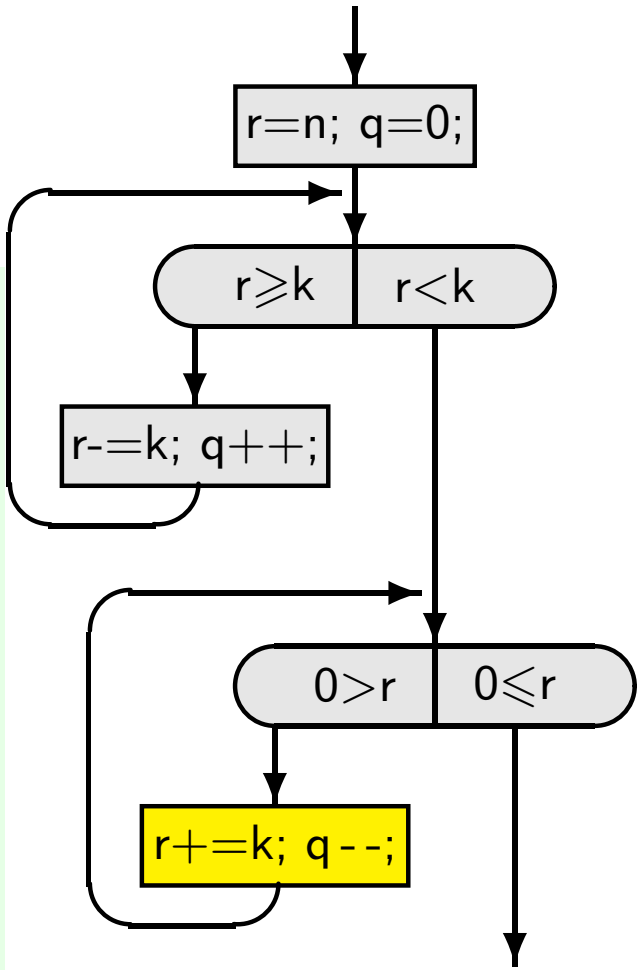
```
loada   n
store   r
loadn   0
store   q
loada   r # petla1
sub     k
mingoto petla2
loada   r
sub     k
store   r
incr    q
goto    petla1
loada   r # petla2
plsgoto KONIEC
zergoto KONIEC
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

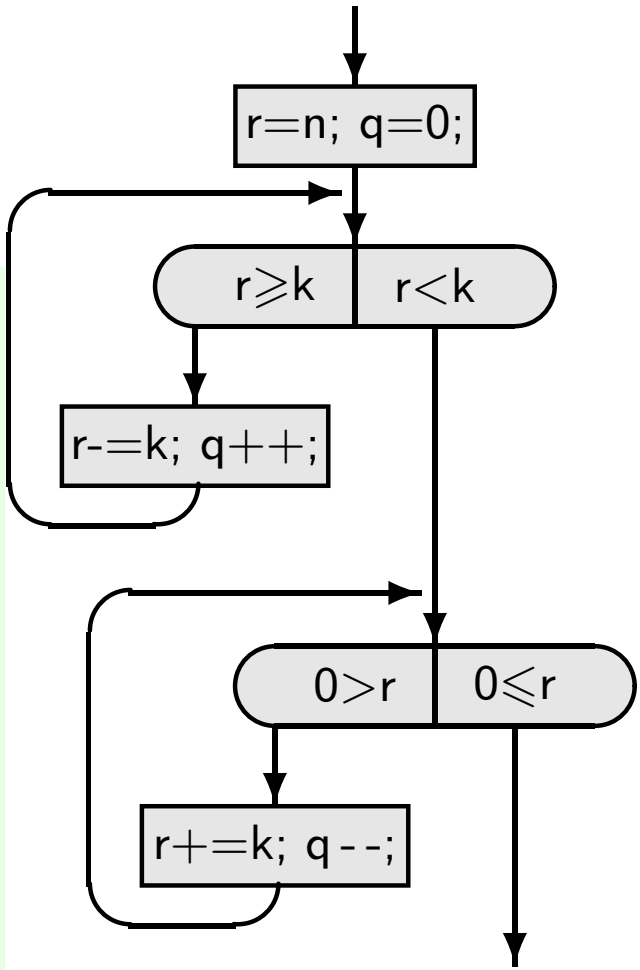
```
loada   n
store   r
loadn   0
store   q
loada   r # petla1
sub     k
mingoto petla2
loada   r
sub     k
store   r
incr    q
goto    petla1
loada   r # petla2
plsgoto KONIEC
zergoto KONIEC
loada   r
add     k
store   r
decr    q
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

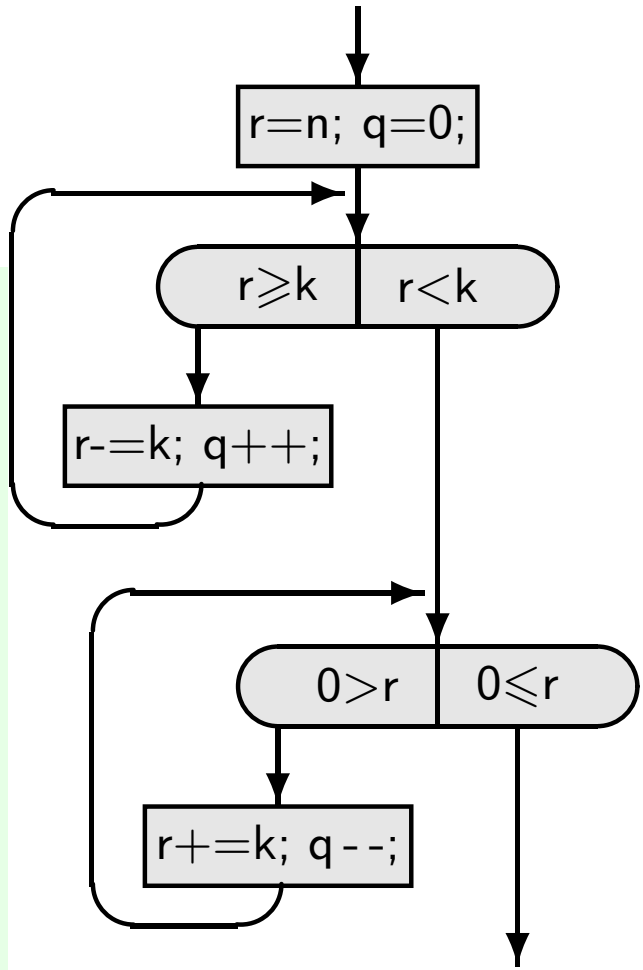
```
loada   n
store   r
loadn   0
store   q
loada   r # petla1
sub     k
mingotopetla2
loada   r
sub     k
store   r
incr    q
goto    petla1
loada   r # petla2
plsgotoKONIEC
zergotoKONIEC
loada   r
add     k
store   r
decr    q
goto    petla2
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

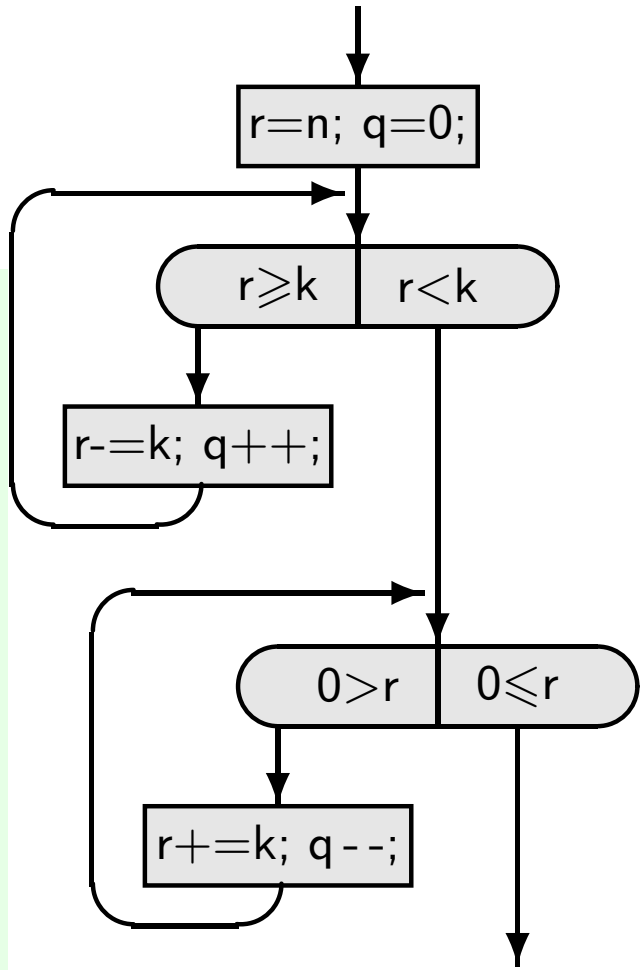
```
                    sub     k
                    store   r
                    incr    q
                    goto    petla1
                    loada   r # petla2
                    plsgoto KONIEC
                    zergoto KONIEC
loada   n           loada   r
store   r           add     k
loadn   0           store   r
store   q           decr    q
loada   r # petla1  goto    petla2
sub     k           stop      # KONIEC
mingoto petla2
loada   r
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

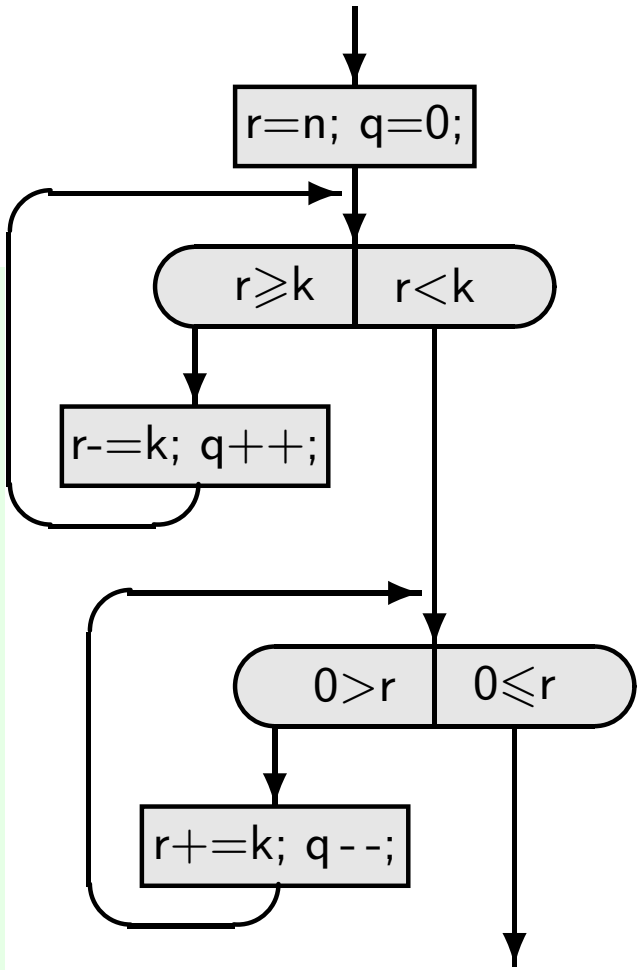
```
                          sub     k
1: 0        # n (dana)    store   r
2: 0        # k (dana)    incr    q
3: 0        # q (wynik)   goto    petla1
4: 0        # r (wynik)   loada   r # petla2
   loada    n           plsgoto KONIEC
   store    r           zergoto KONIEC
   loadn    0           loada   r
   store    q           add     k
   loada    r # petla1  store   r
   sub      k           decr    q
   mingoto  petla2      goto    petla2
   loada    r           stop      # KONIEC
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

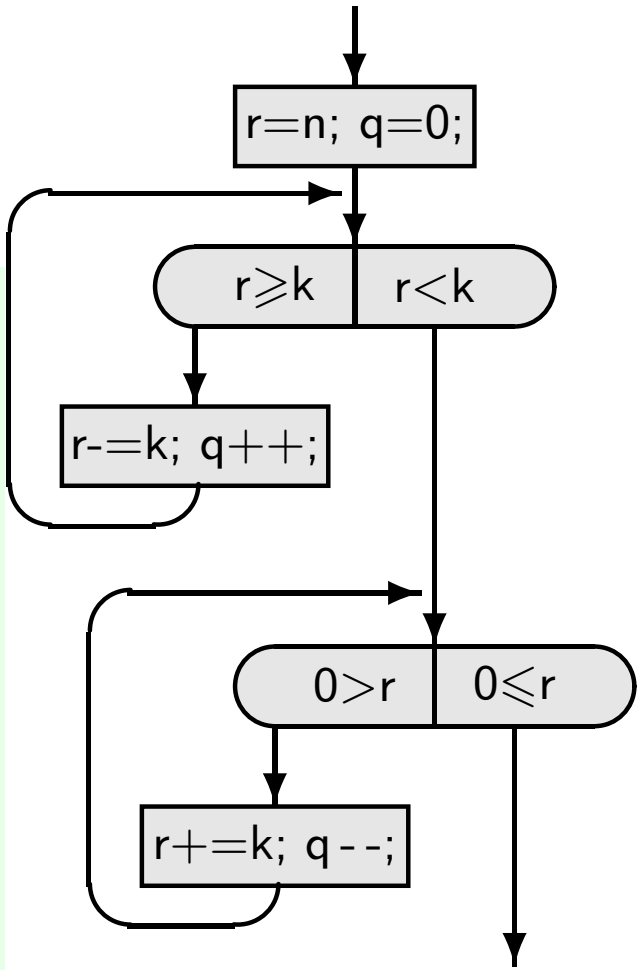
```
 1: 0       # n (dana)
 2: 0       # k (dana)
 3: 0       # q (wynik)
 4: 0       # r (wynik)
 5: loada   n
 6: store   r
 7: loadn   0
 8: store   q
 9: loada   r # petla1
10: sub     k
11: mingoto petla2
12: loada   r
13: sub     k
14: store   r
15: incr    q
16: goto    petla1
17: loada   r # petla2
18: plsgoto KONIEC
19: zergoto KONIEC
20: loada   r
21: add     k
22: store   r
23: decr    q
24: goto    petla2
25: stop      # KONIEC
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

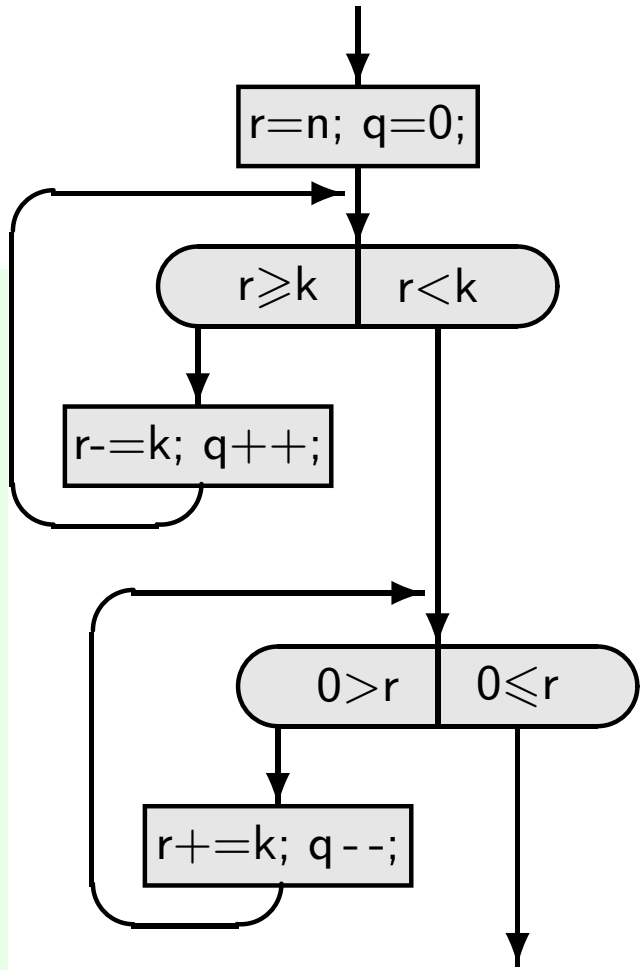
```
 1: 0       # n (dana)
 2: 0       # k (dana)
 3: 0       # q (wynik)
 4: 0       # r (wynik)
 5: loada   n
 6: store   4
 7: loadn   0
 8: store   q
 9: loada   4 # petla1
10: sub     k
11: mingoto petla2
12: loada   4
13: sub     k
14: store   4
15: incr    q
16: goto    petla1
17: loada   4 # petla2
18: plsgoto KONIEC
19: zergoto KONIEC
20: loada   4
21: add     k
22: store   4
23: decr    q
24: goto    petla2
25: stop      # KONIEC
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

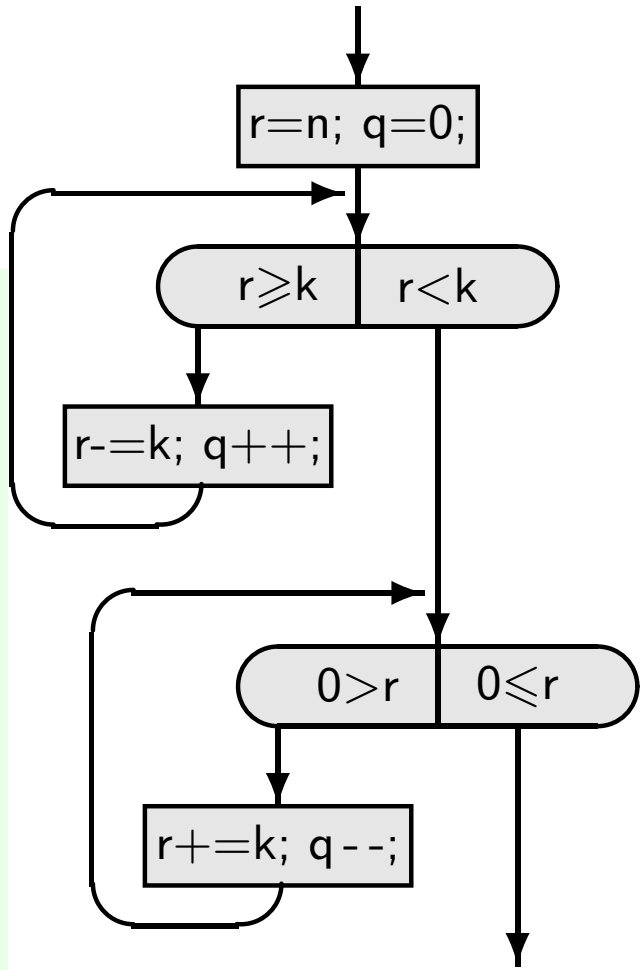
```
 1: 0       # n (dana)
 2: 0       # k (dana)
 3: 0       # q (wynik)
 4: 0       # r (wynik)
 5: loada   1 # POCZATEK
 6: store   4
 7: loadn   0
 8: store   3
 9: loada   4 # petla1
10: sub     2
11: mingoto 17
12: loada   4
13: sub     2
14: store   4
15: incr    3
16: goto    9
17: loada   4 # petla2
18: plsgoto 25
19: zergoto 25
20: loada   4
21: add     2
22: store   4
23: decr    3
24: goto    17
25: stop      # KONIEC
```



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

```
 1: 0       # n (dana)
 2: 0       # k (dana)
 3: 0       # q (wynik)
 4: 0       # r (wynik)
 5: loada   1 # POCZATEK
 6: store   4
 7: loadn   0
 8: store   3
 9: loada   4 # petla1
10: sub     2
11: mingoto 17
12: loada   4
13: sub     2
14: store   4
15: incr    3
16: goto    9
17: loada   4 # petla2
18: plsgoto 25
19: zergoto 25
20: loada   4
21: add     2
22: store   4
23: decr    3
24: goto    17
25: stop      # KONIEC
```



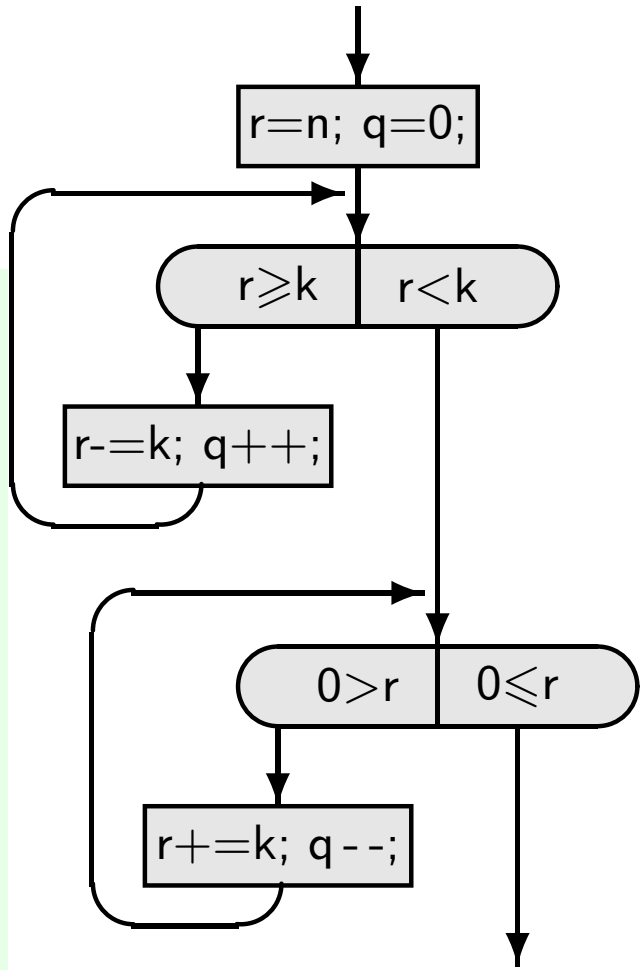
Żeby w innym programie móc użyć tego dzielenia — trzeba zrobić z niego **podprogram**.

Program nadrzędny będzie do komórek 1 i 2 wpisywał swoje dzielną i dzielnik, do komórki 0 — adres, do którego należy skoczyć po zakończeniu, i będzie sam skakał do 5.

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

```
 0: 0        # ADR. POWROTU
 1: 0        # dzielna (dana)
 2: 0        # dzielnik (dana)
 3: 0        # iloraz (wynik)
 4: 0        # reszta (wynik)
 5: loada    1 # POCZATEK
 6: store    4
 7: loadn    0
 8: store    3
 9: loada    4 # petla1
10: sub      2
11: mingoto  17
12: loada    4
13: sub      2
14: store    4
15: incr     3
16: goto     9
17: loada    4 # petla2
18: plsgoto  25
19: zergoto  25
20: loada    4
21: add      2
22: store    4
23: decr     3
24: goto     17
25: goto 0+[0]    # POWROT
```



Żeby to dzielenie móc użyć w innym programie — zrobić z niego **podprogram**.

- do komórek 1 i 2 dzielną i dzielnik;
- do komórki 0 adres, do którego skoczyć po zakończeniu;
- skoczyć do komórki 5.

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

$$\underbrace{\mathit{wynik} \cdot x^n = \mathit{podst}^{\mathit{wykl}}}_{\text{niezmiennik}}$$

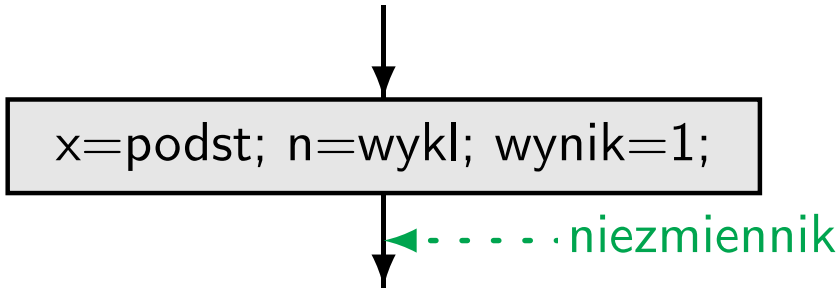
Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!



Chcemy mieć:

$$\underbrace{wynik \cdot x^n = podst^{wykl}}_{\text{niezmiennik}}$$

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

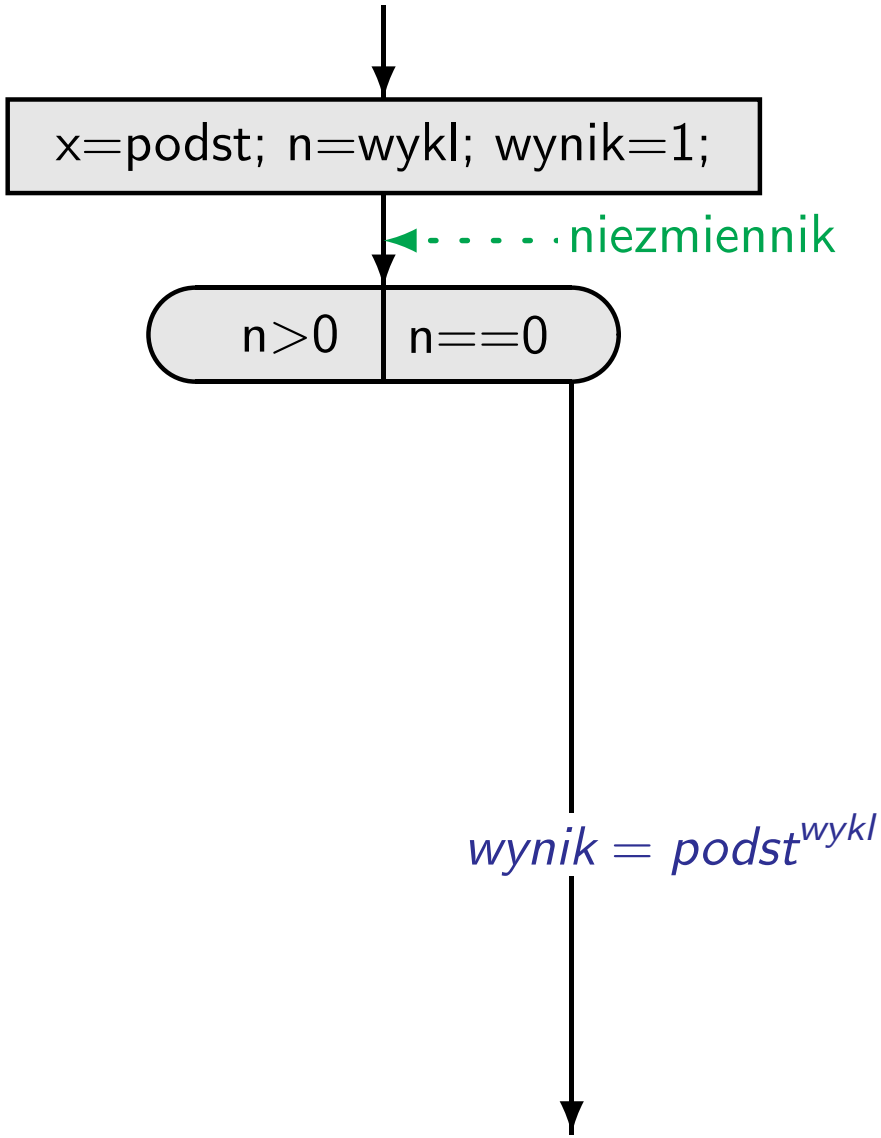
Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

$$\underbrace{wynik \cdot x^n = podst^{wykl}}_{\text{niezmiennik}}$$



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

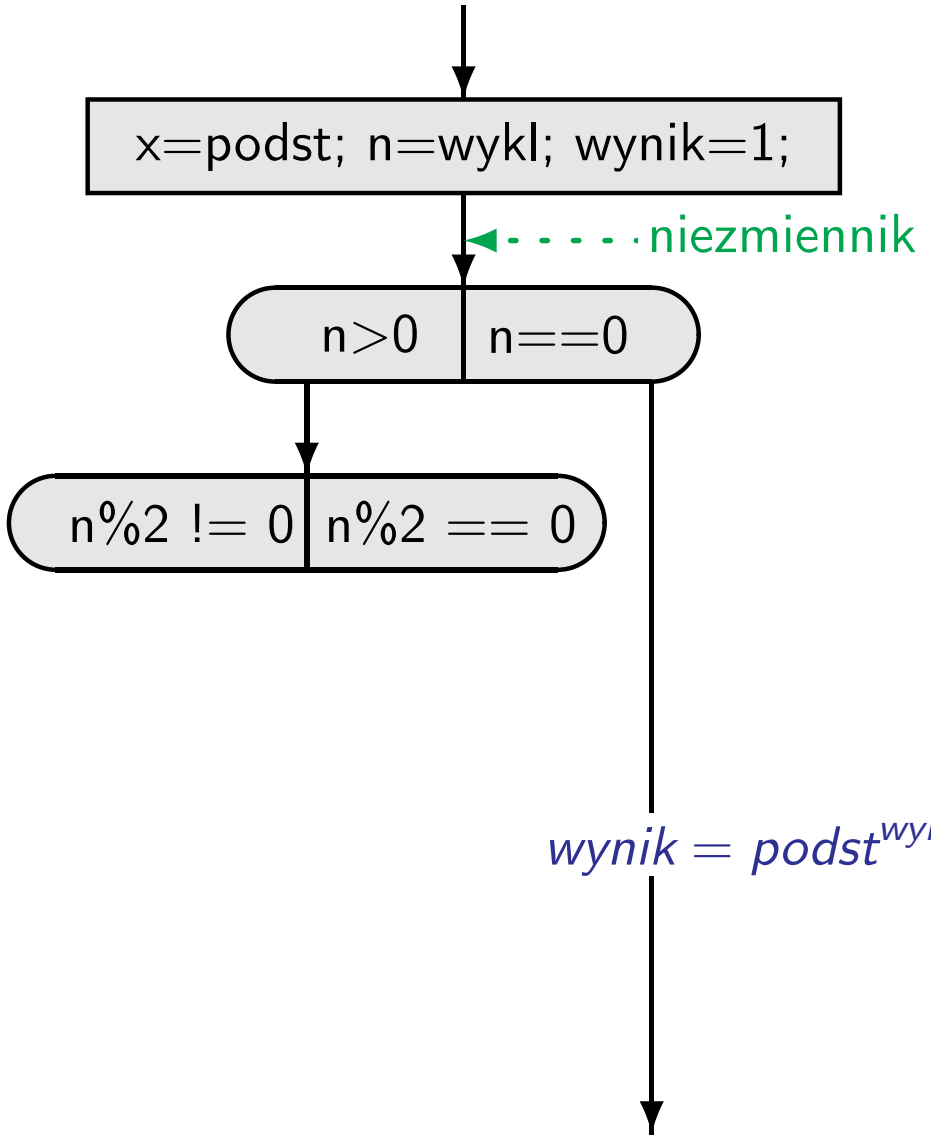
Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

$$\underbrace{wynik \cdot x^n = podst^{wykl}}_{\text{niezmiennik}}$$



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

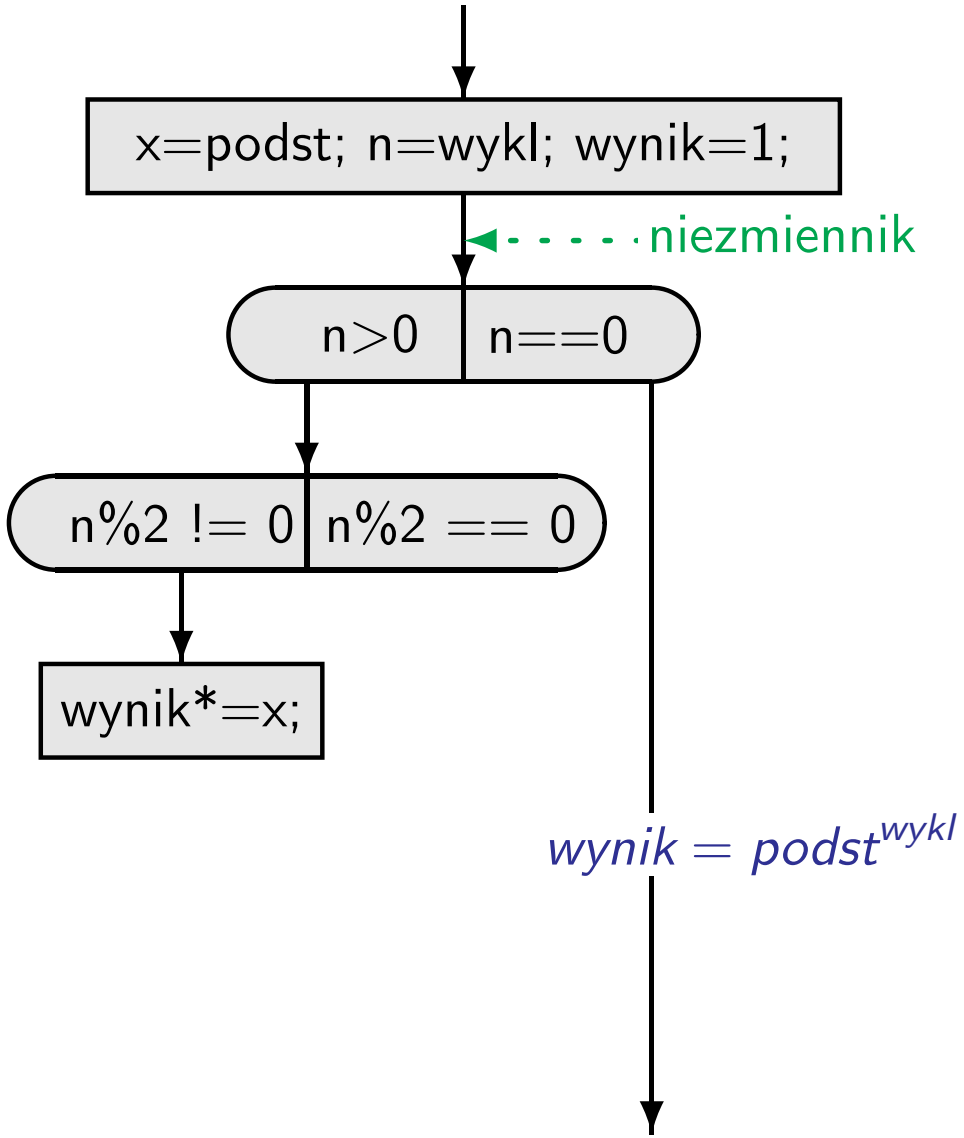
Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

$$\underbrace{wynik \cdot x^n = podst^{wykl}}_{\text{niezmiennik}}$$



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

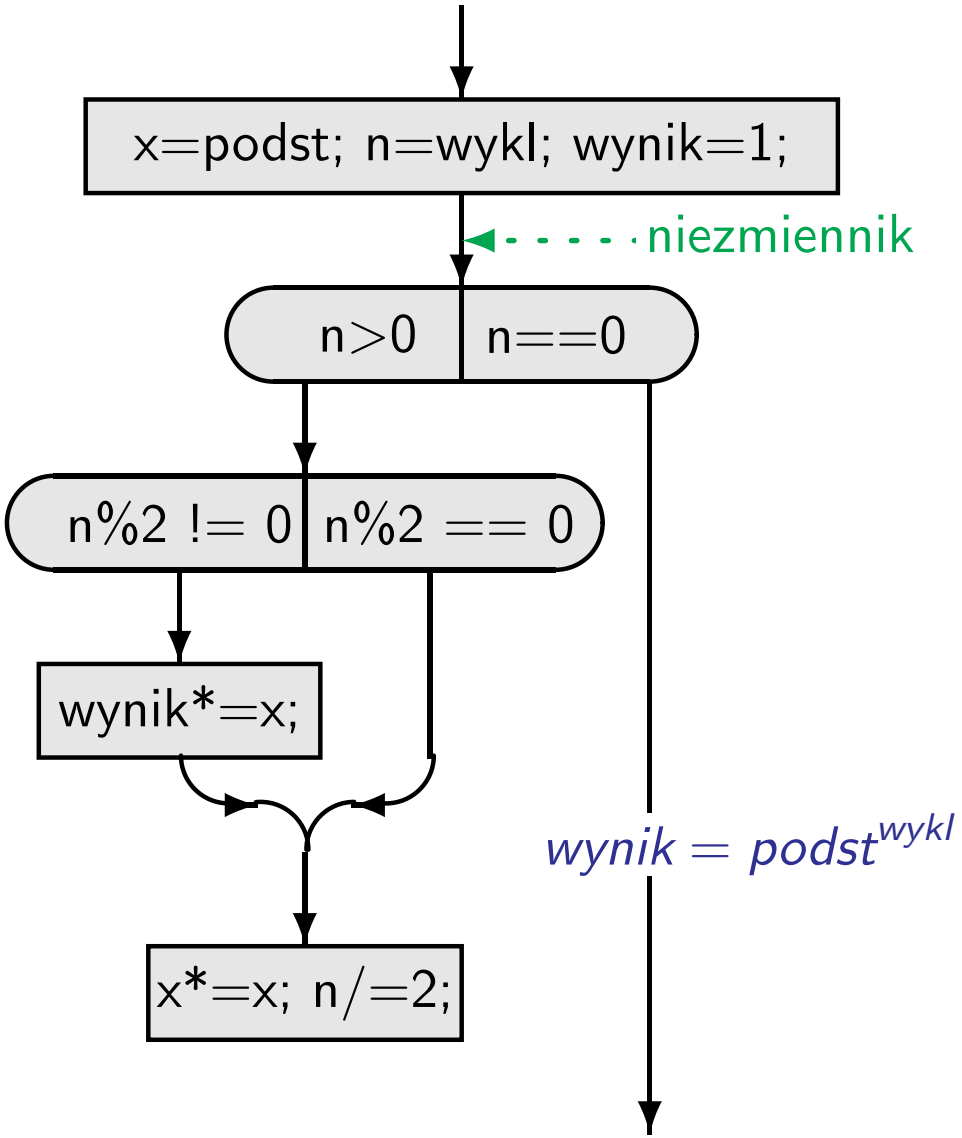
Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

$$\underbrace{wynik \cdot x^n = podst^{wykl}}_{\text{niezmiennik}}$$



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

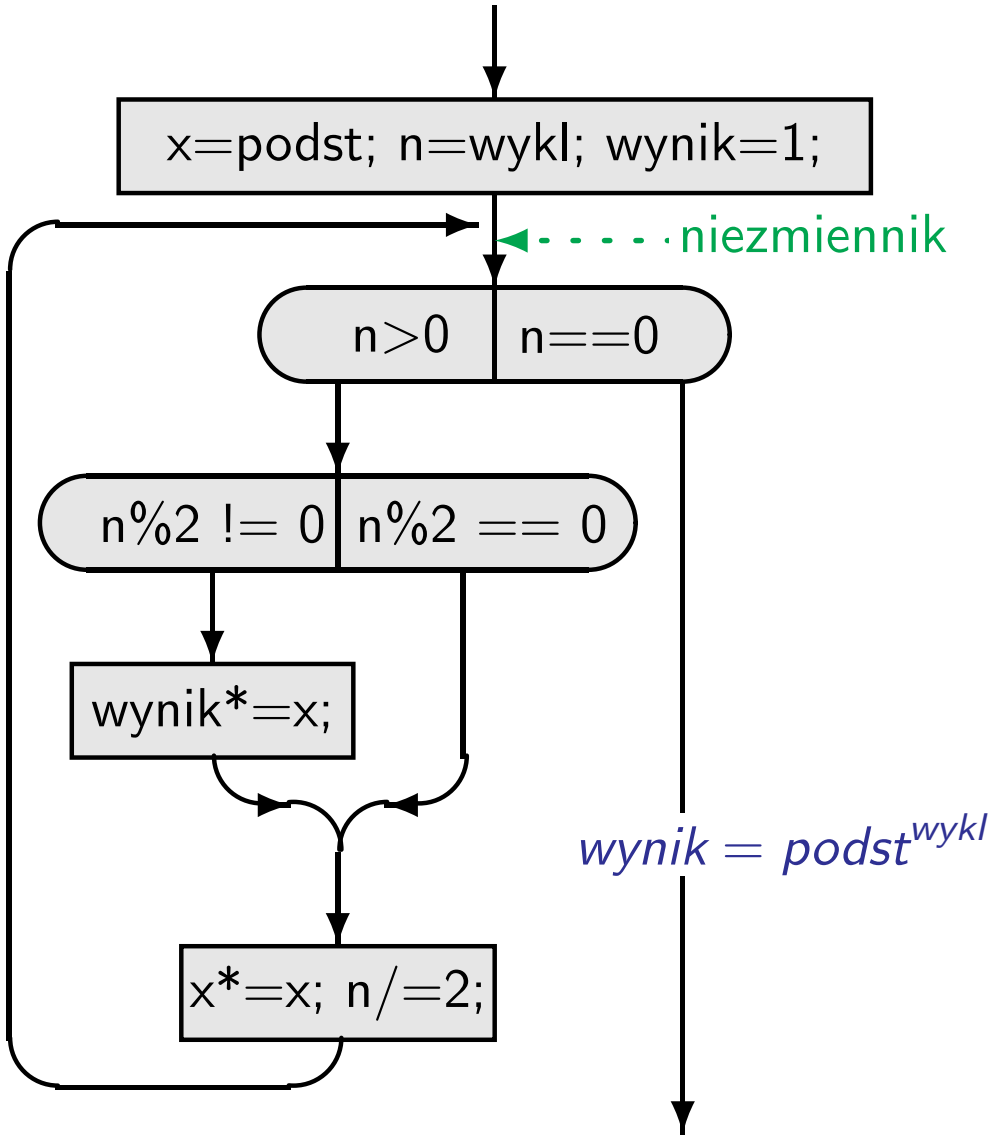
Za każdym obrotem pętli

- podnieść podstawę do kwadratu,
- podzielić wykładnik przez 2,

ale *uważać* na nieparzyste wykładniki!

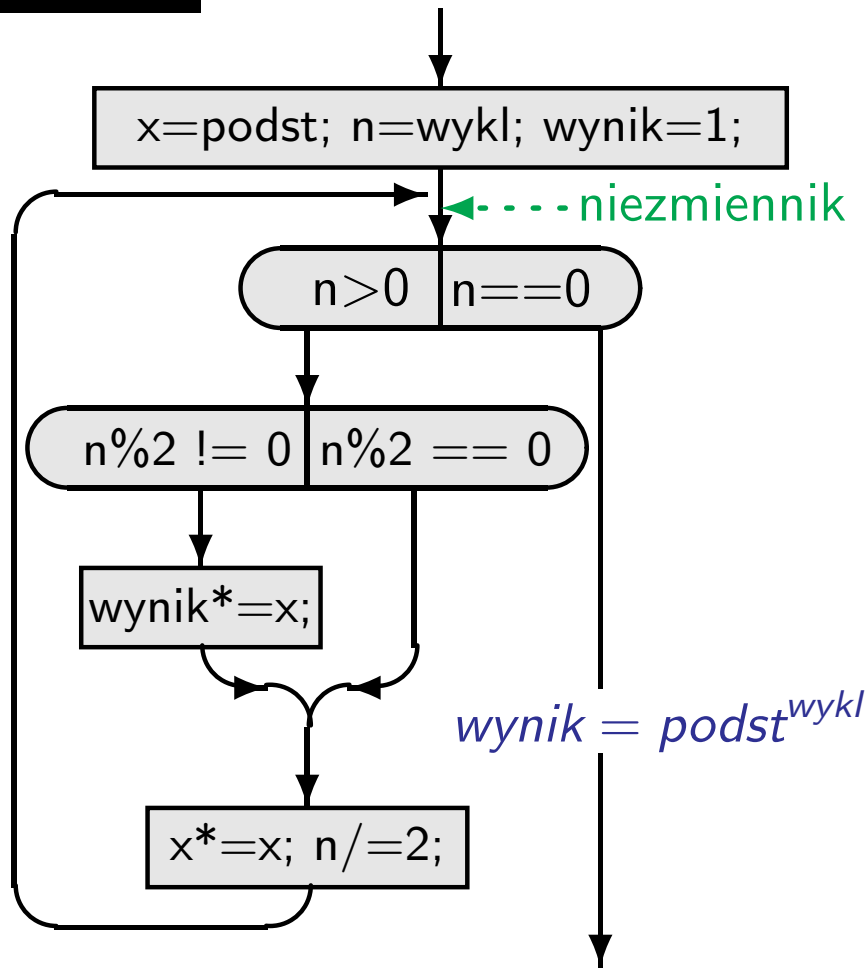
Chcemy mieć:

$$\underbrace{wynik \cdot x^n = podst^{wykl}}_{\text{niezmiennik}}$$



Duży program tworzony po kawałku

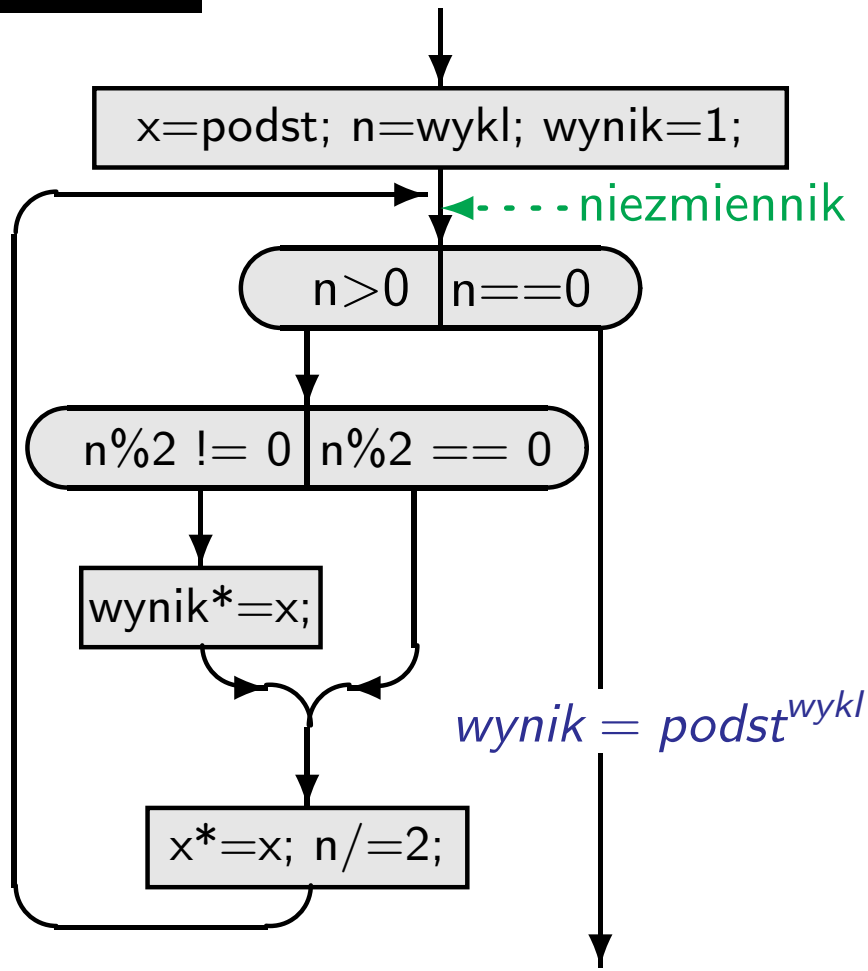
Kawałek 2: (potęgowanie binarne)



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

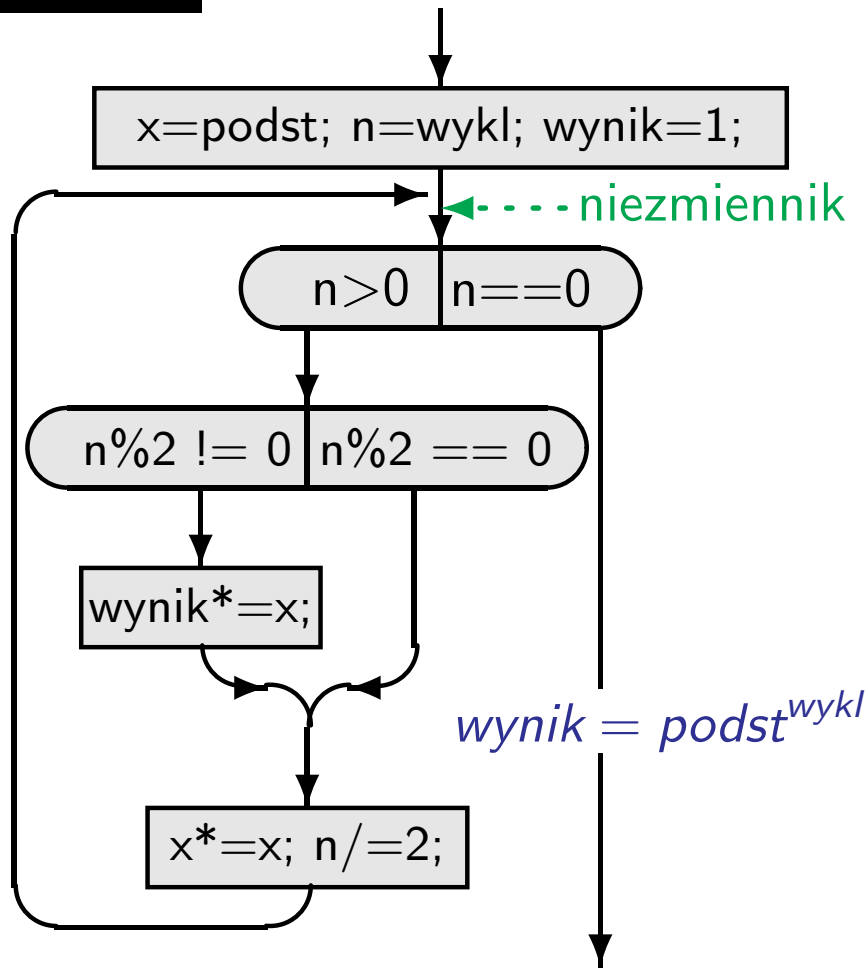
```
200: loada   podst
201: store   x
202: loada   wykl
203: store   n
204: loadn   1
205: store   wynik
206: loada   n # while
207: zergoto koniec
```



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

```
200: loada  podst
201: store  x
202: loada  wykl
203: store  n
204: loadn  1
205: store  wynik
206: loada  n # while
207: zergoto koniec
208: store  1 # n jest dzielna
209: loadn  2
210: store  2 # 2 jest dzielnikiem
211: loadn  214
212: store  0 # adresem powrotu jest 214
213: goto   5 # wywolanie podprogramu
                . . . . . .
```

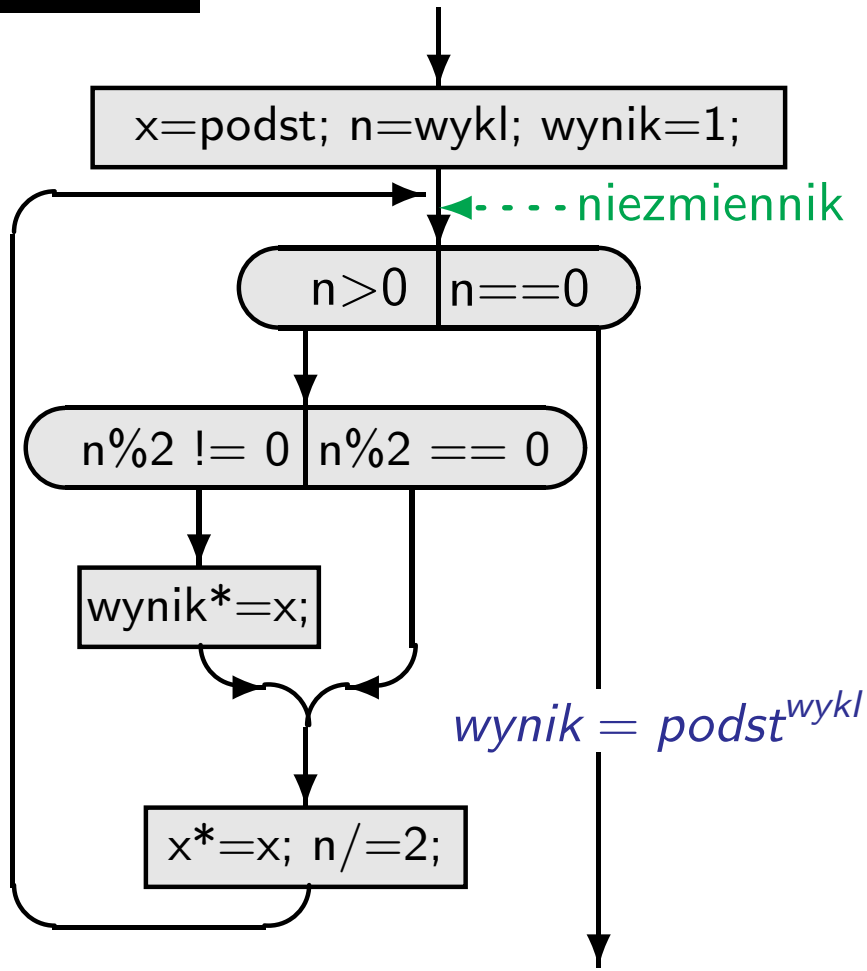


na żółto:
parametry i wywołanie podprogramu

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

```
                . . . . . .
213: goto    5 # wywolanie podprogramu
214: loada   4 # pobrac reszte
215: zergoto 219 # przeskoczyc mnozenie
216: loada   wynik
217: mul     x
218: store   wynik
219: loada   x
220: mul     x
221: store   x
222: loada   3 # pobrac iloraz
223: store   n
224: goto    while
225: stop    # koniec
226: 0       # podst
227: 0       # wykl
228: 0       # wynik
229: 0       # x
230: 0       # n
END
```



na żółto:
parametry i wywołanie podprogramu

Duży program tworzony po kawałku

Programy z tego wykładu:

iis.ans-elblag.pl/∼stefan/Dydaktyka/
JezForm/Wyklady/10-Wewn/