

Stefan Sokołowski

JĘZYKI FORMALNE I METODY KOMPILACJI

wykład 10

**Inst. Informatyki Stosowanej, ANS
Elbląg, 2025/2026**

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/...

dla języka wewnętrznego
to są synonimy

Duży program tworzony po kawałku

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

Jak program współpracuje z funkcją?

dla języka wewnętrznego
to są synonimy

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

dla języka wewnętrznego
to są synonimy

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,
- program podaje funkcji argumenty

dla języka wewnętrznego
to są synonimy

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,
- program podaje funkcji argumenty,
- funkcja oblicza wartość wyniku i oddaje programowi

dla języka wewnętrznego
to są synonimy

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,
- 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.

dla języka wewnętrznego
to są synonimy

Duży program tworzony po kawałku

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

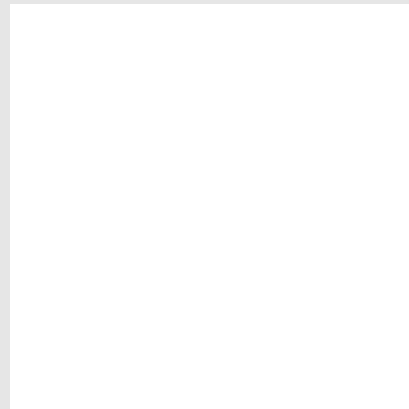
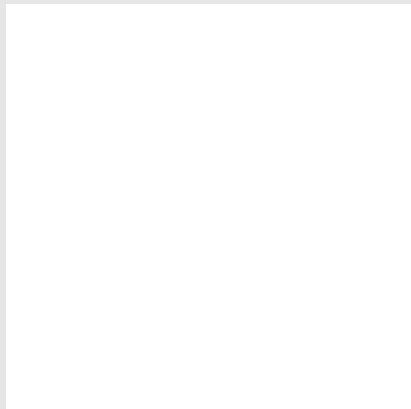
dla języka wewnętrznego
to są synonimy

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.

PROGRAM 1

FUNKCJA



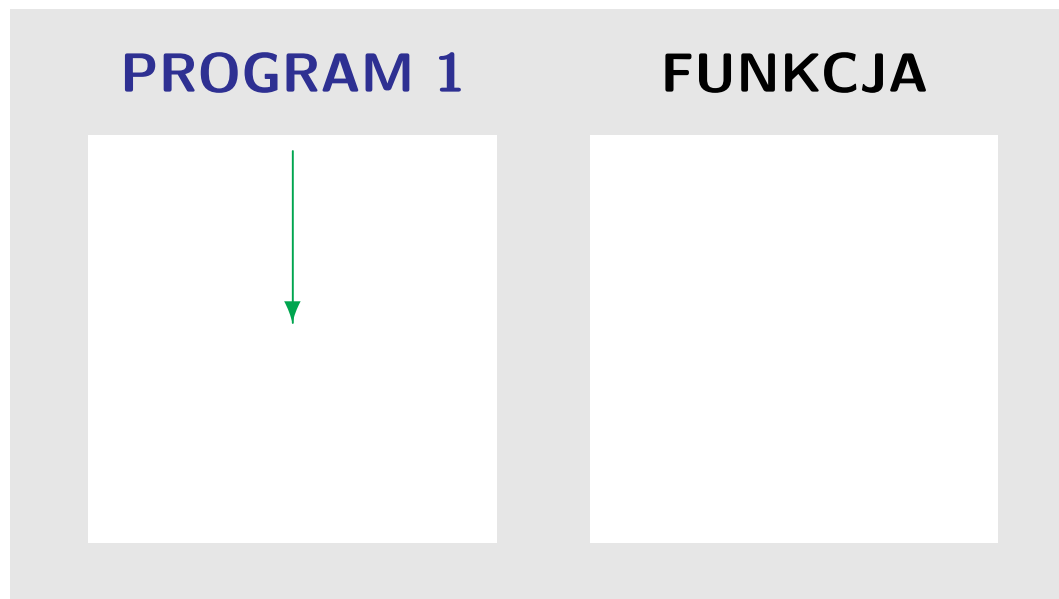
Duży program tworzony po kawałku

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.



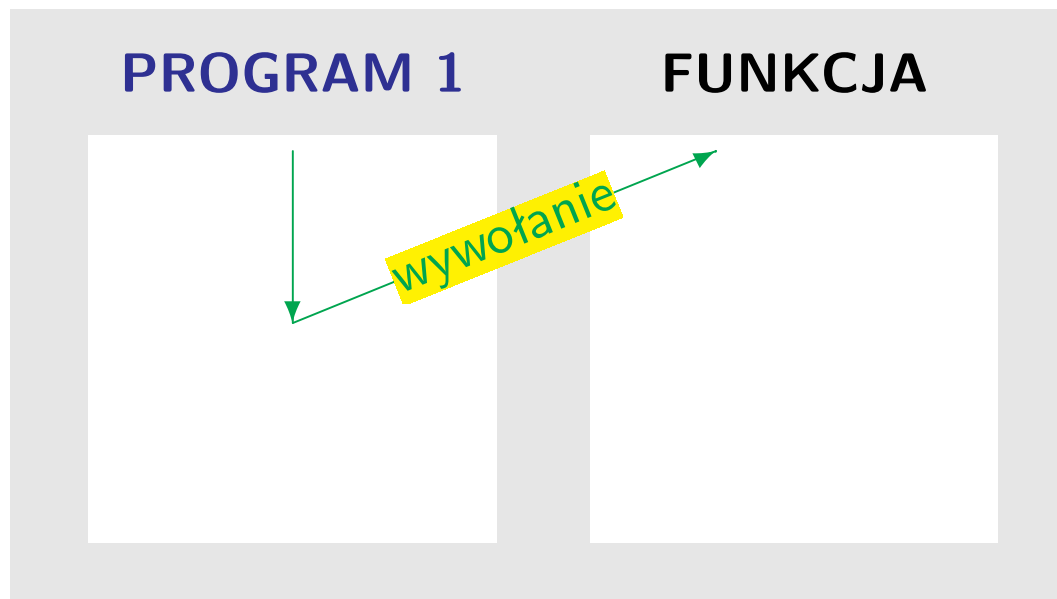
Duży program tworzony po kawałku

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.



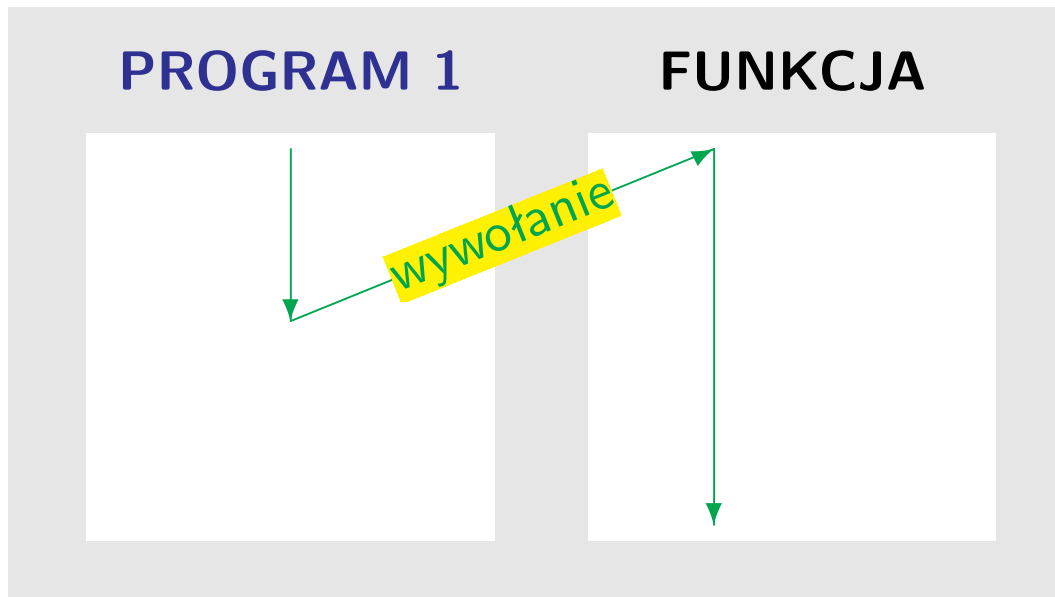
Duży program tworzony po kawałku

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.



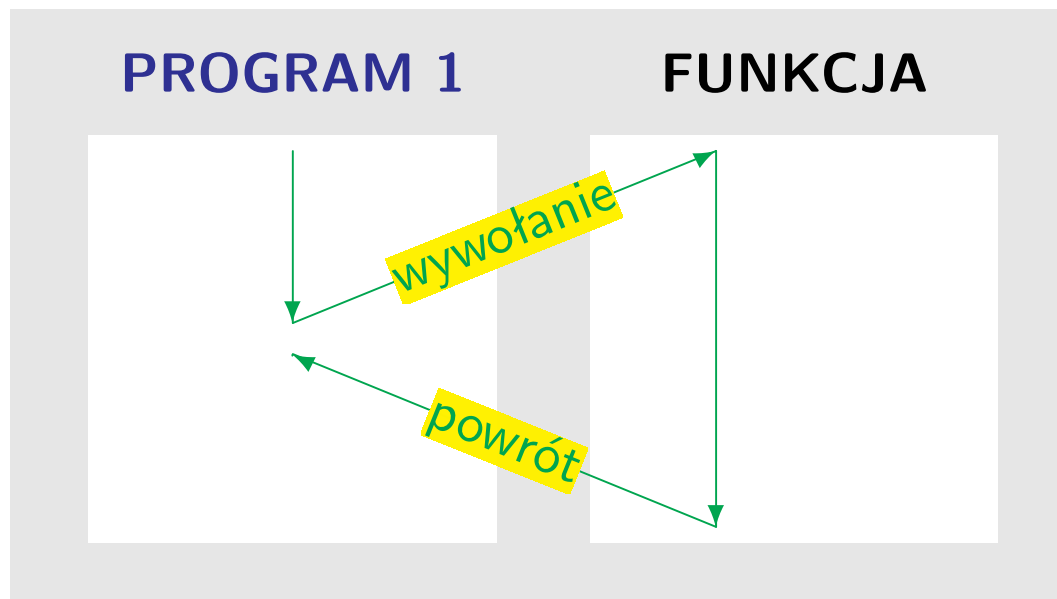
Duży program tworzony po kawałku

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.



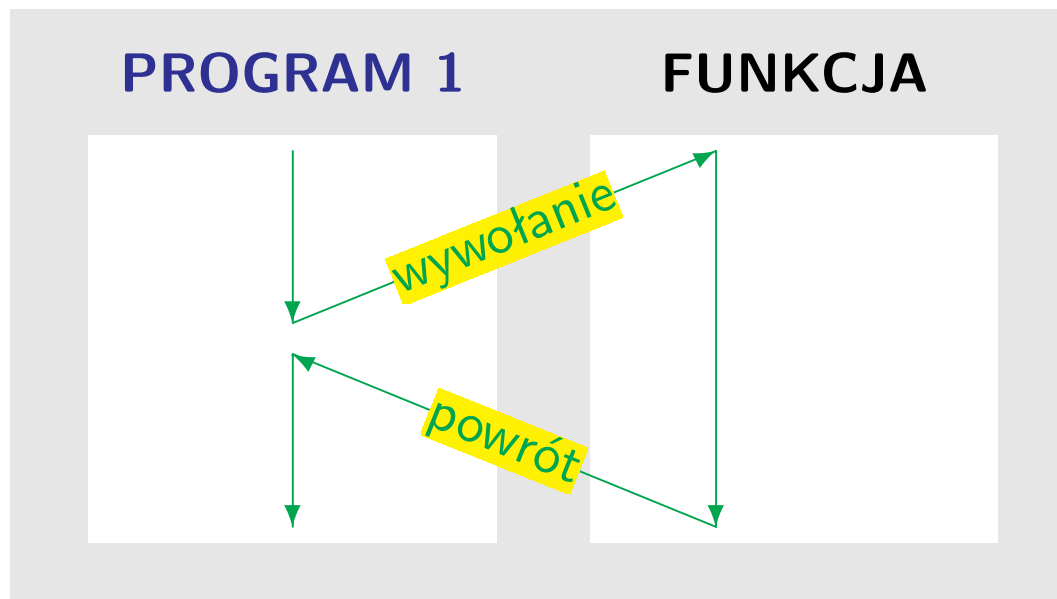
Duży program tworzony po kawałku

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.



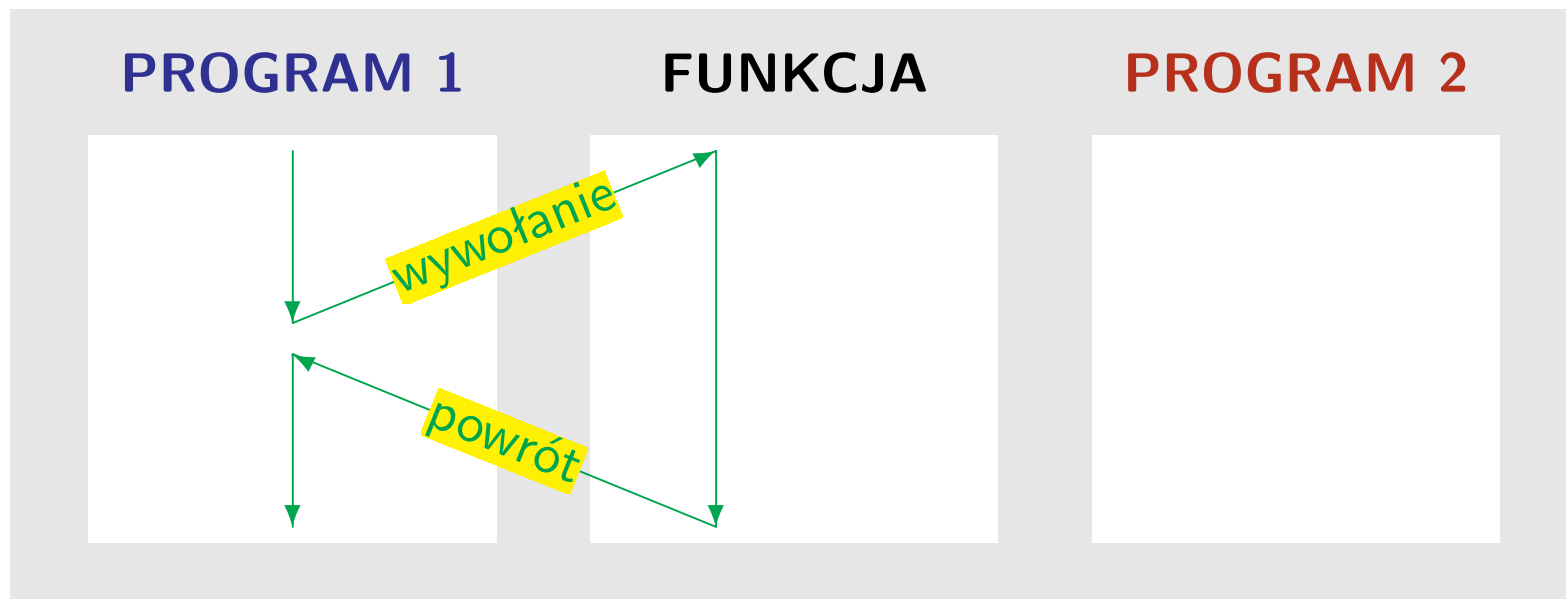
Duży program tworzony po kawałku

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.



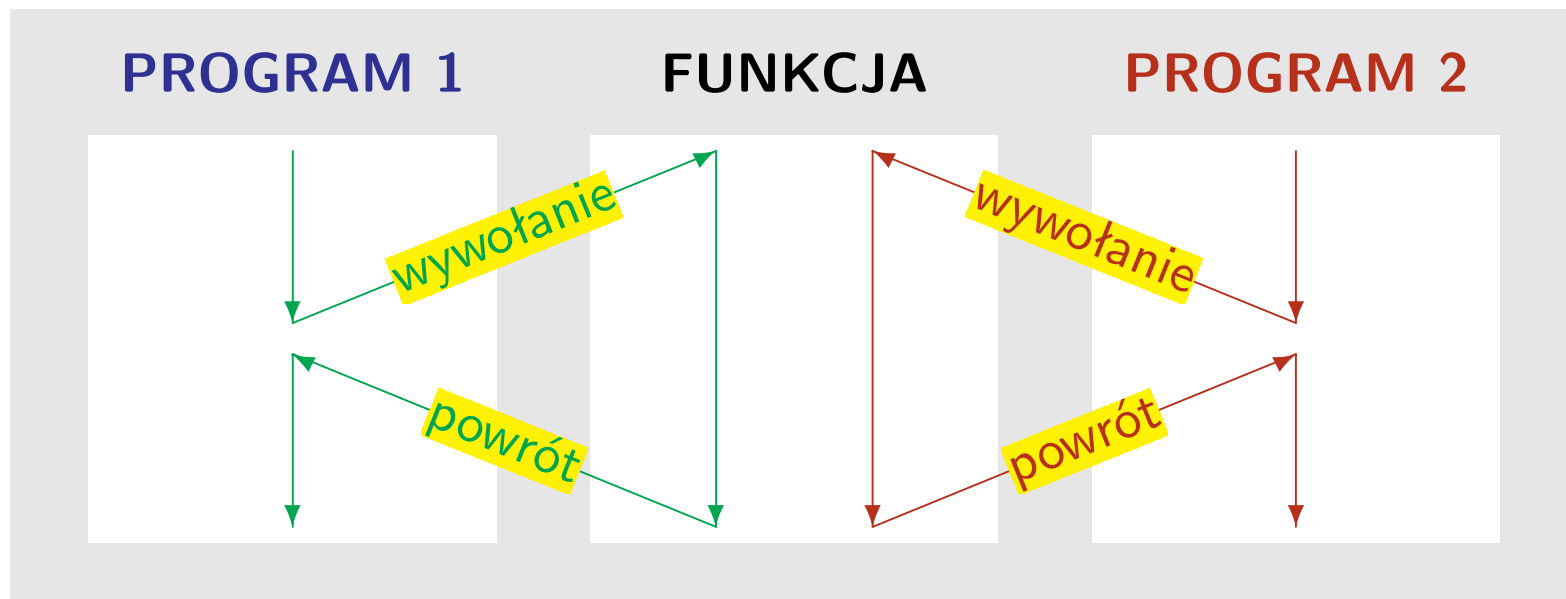
Duży program tworzony po kawałku

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.



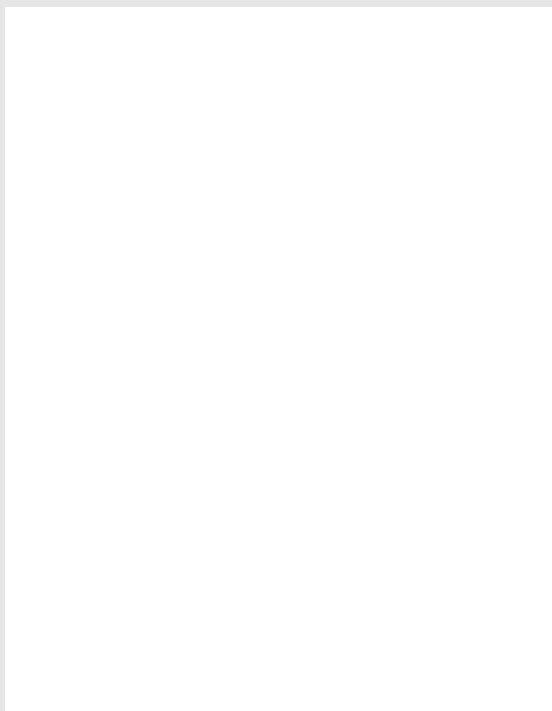
Duży program tworzony po kawałku

PROGRAM

FUNKCJA

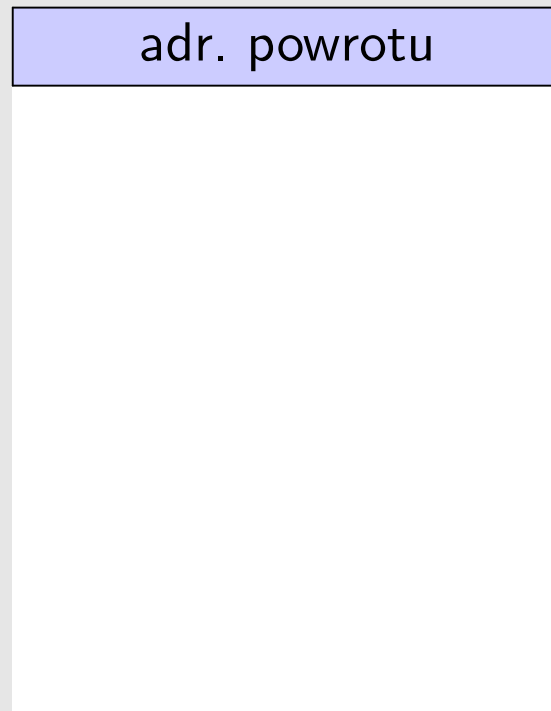
Duży program tworzony po kawałku

PROGRAM



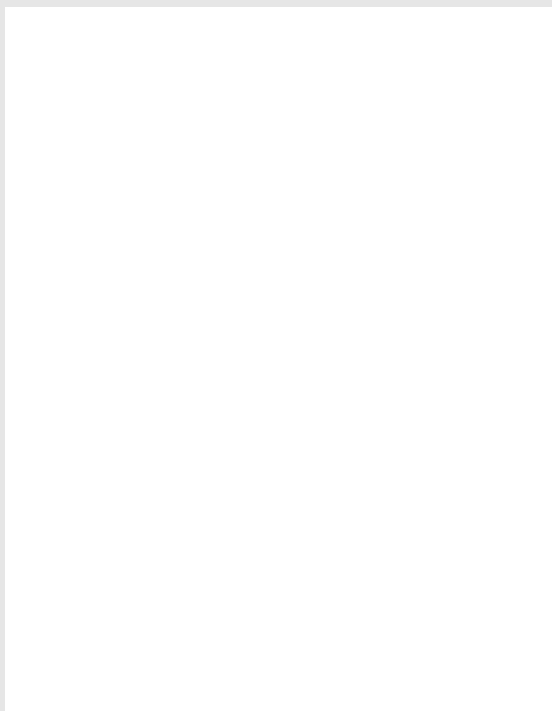
FUNKCJA

adr. powrotu



Duży program tworzony po kawałku

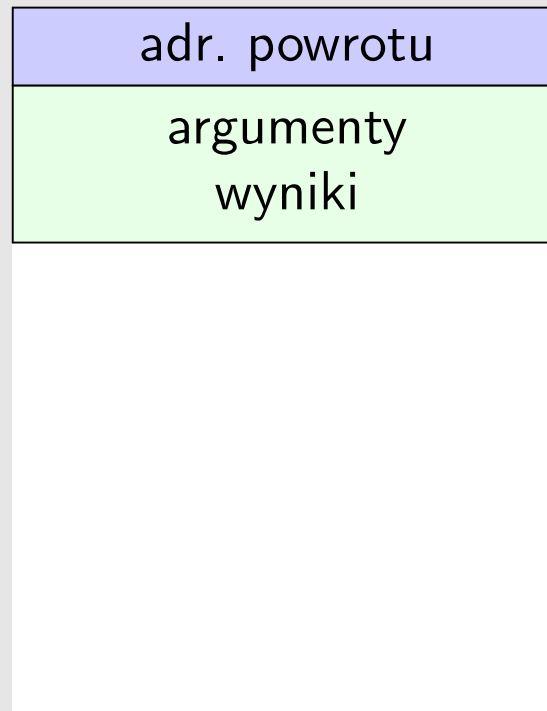
PROGRAM



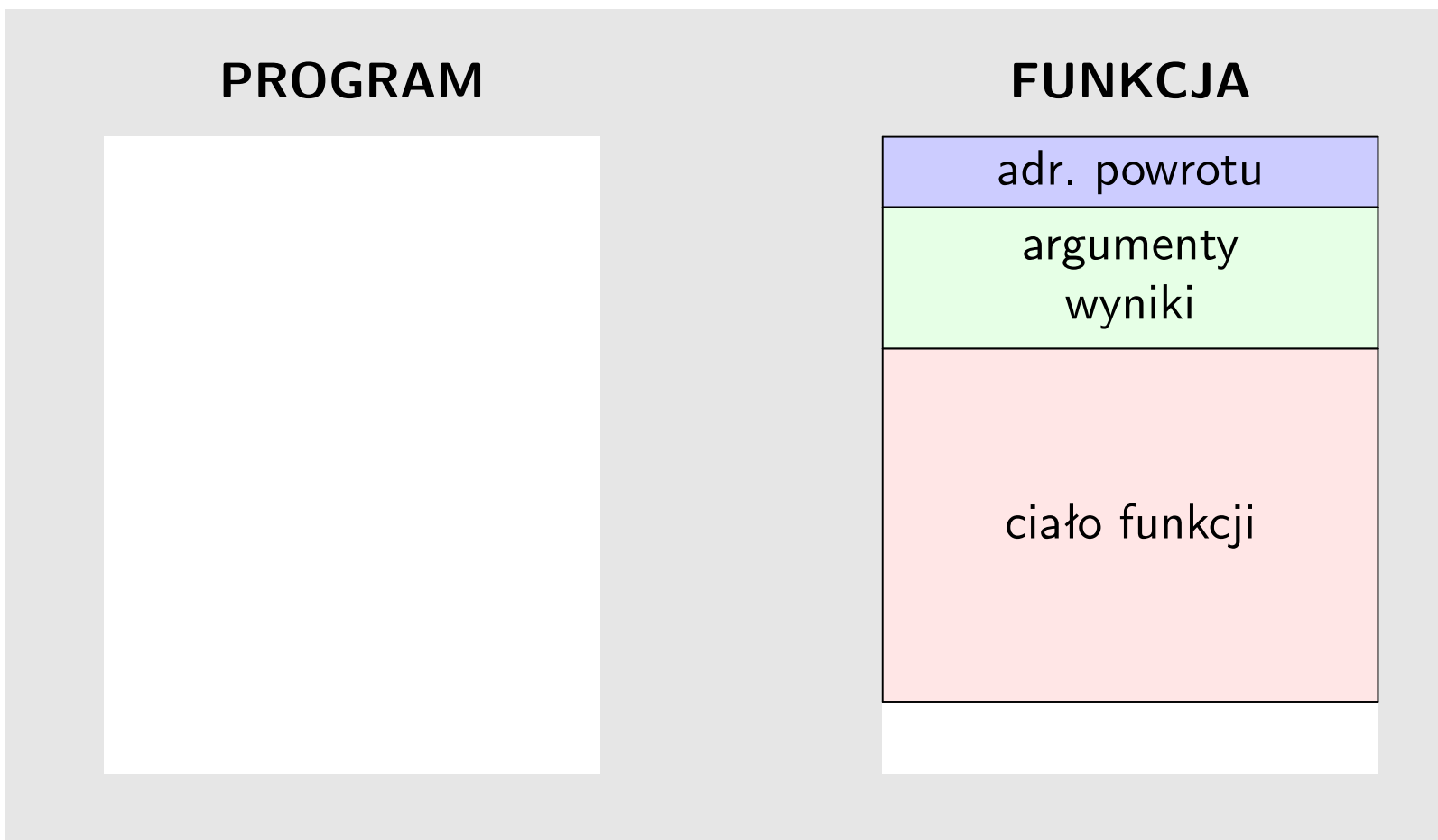
FUNKCJA

adr. powrotu

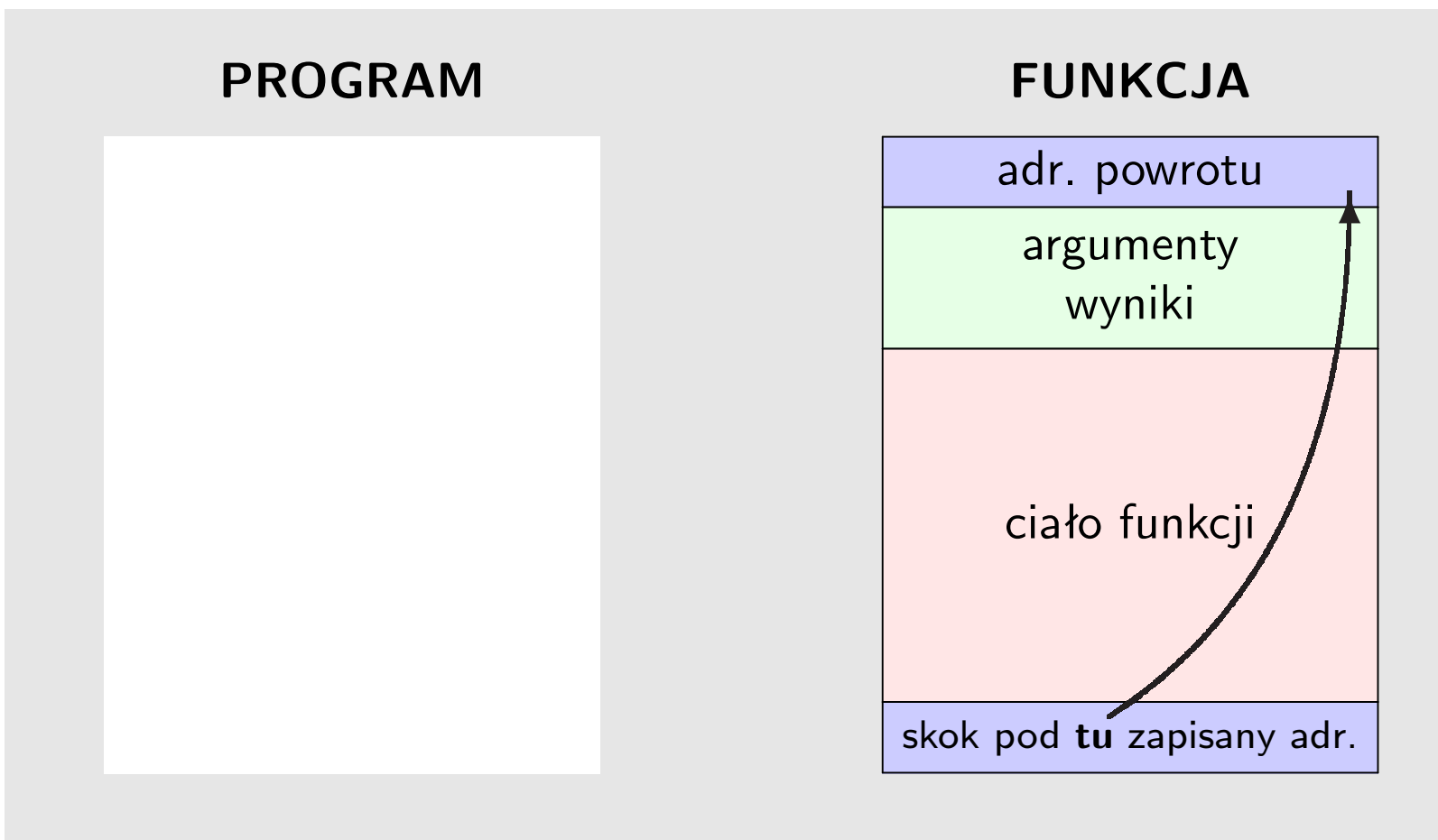
argumenty
wyniki



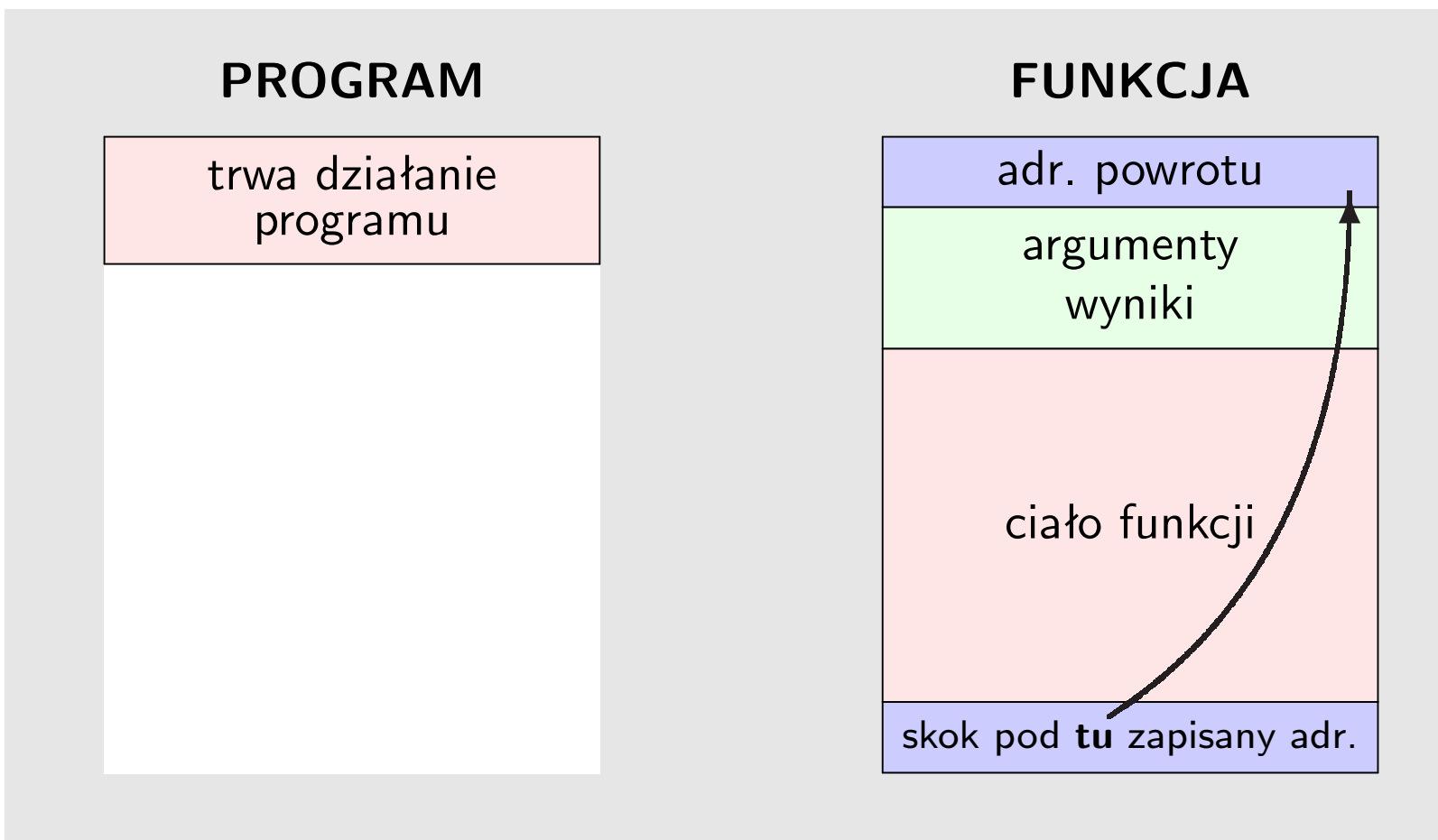
Duży program tworzony po kawałku



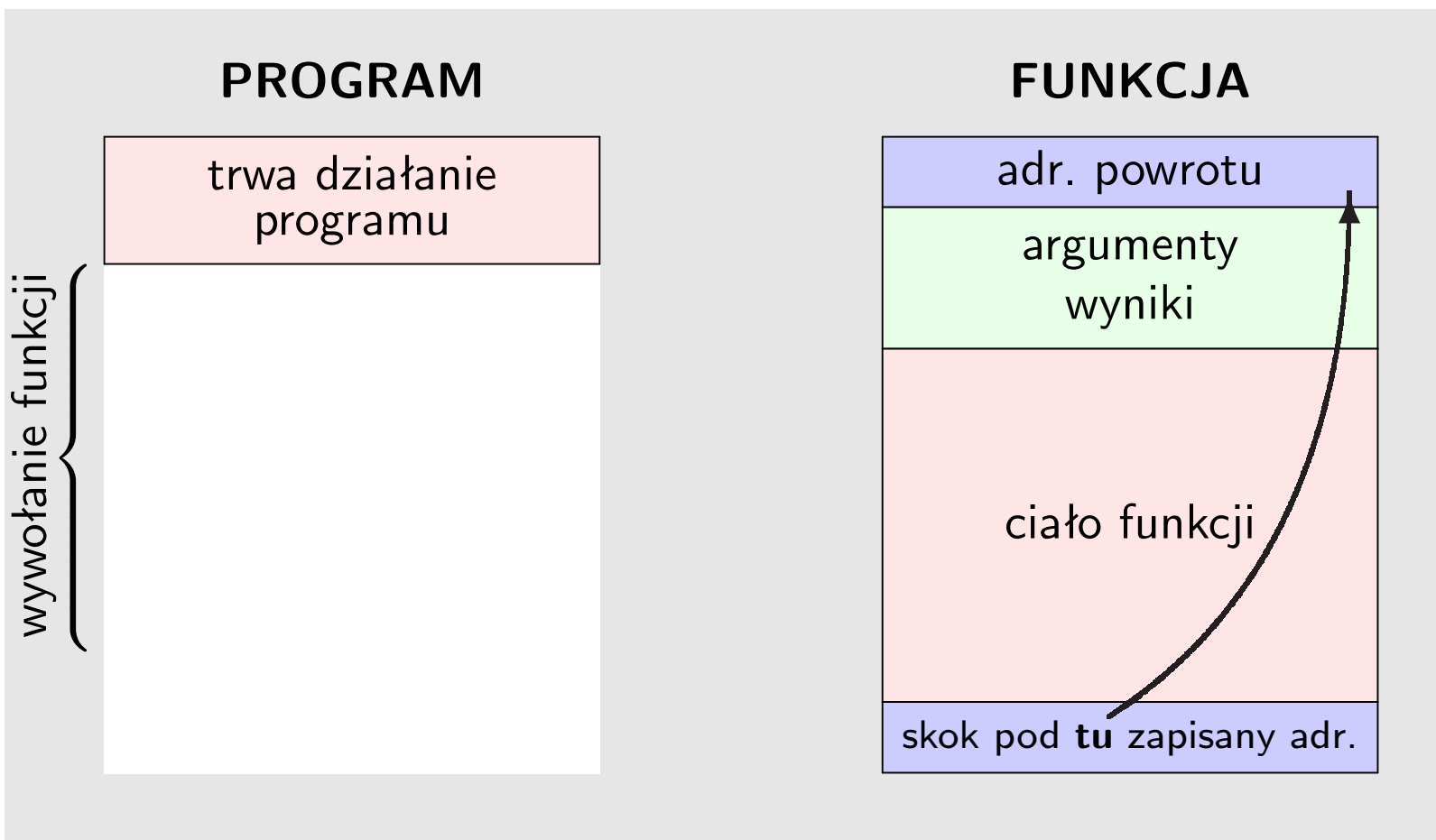
Duży program tworzony po kawałku



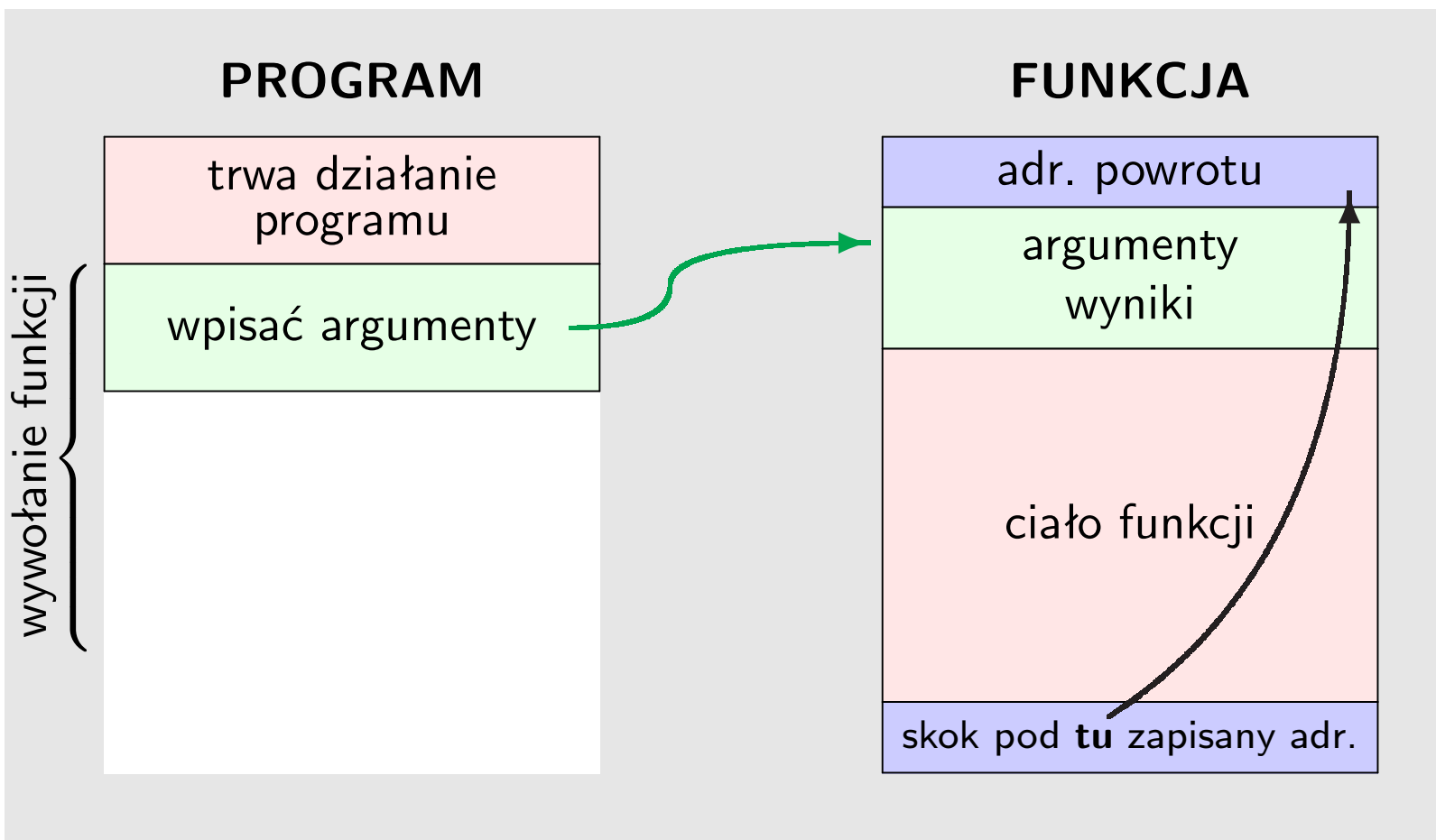
Duży program tworzony po kawałku



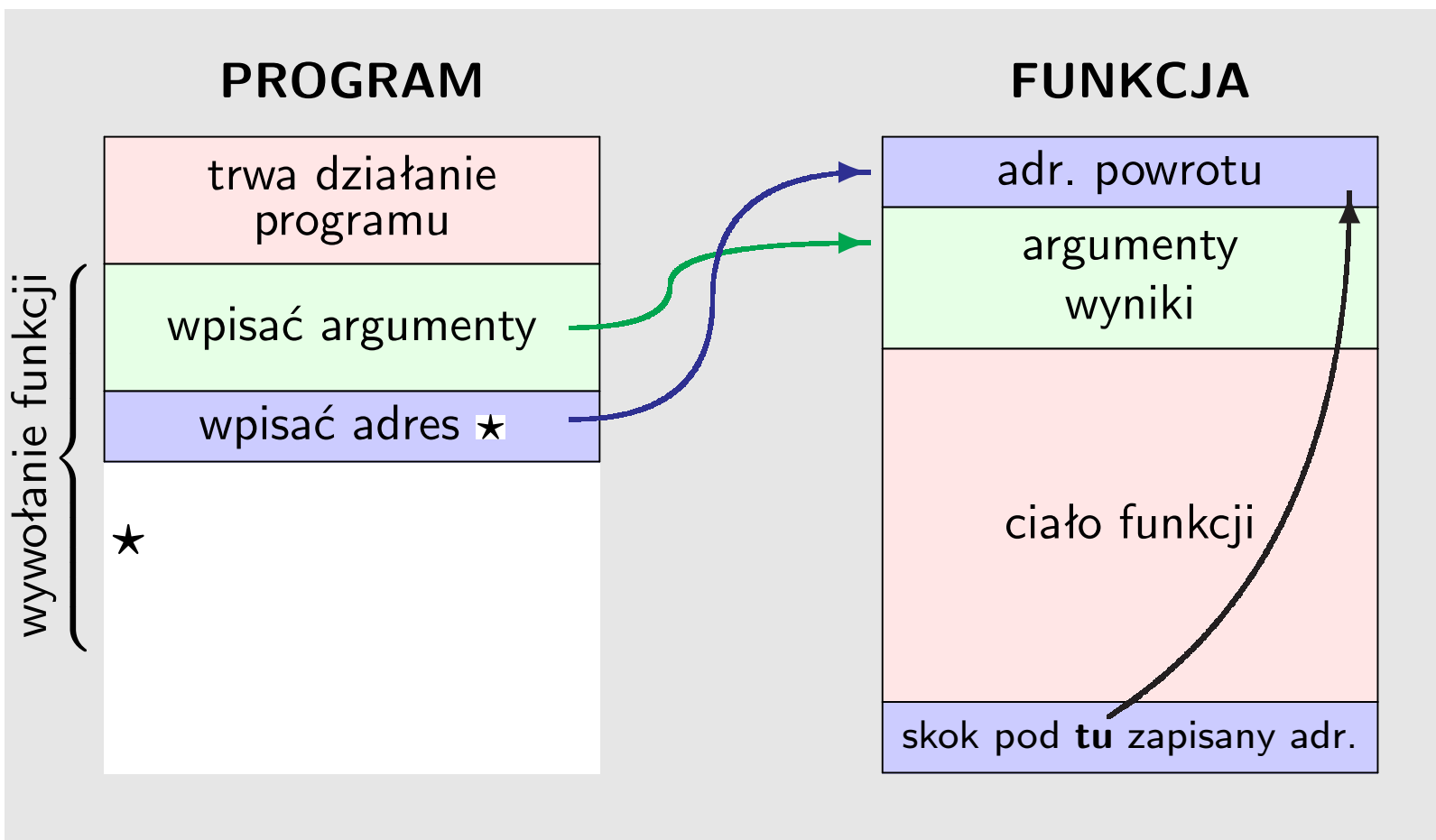
Duży program tworzony po kawałku



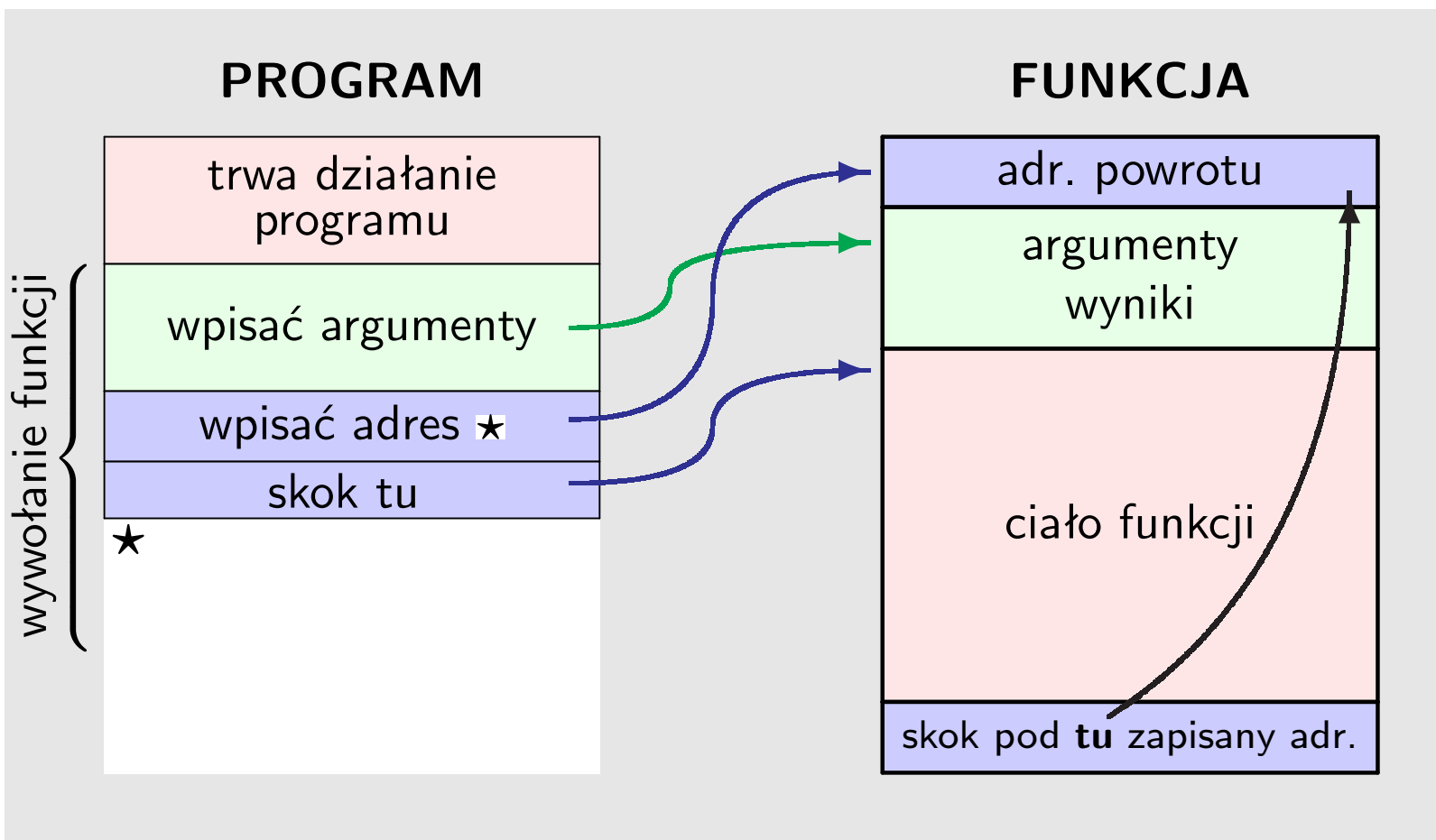
Duży program tworzony po kawałku



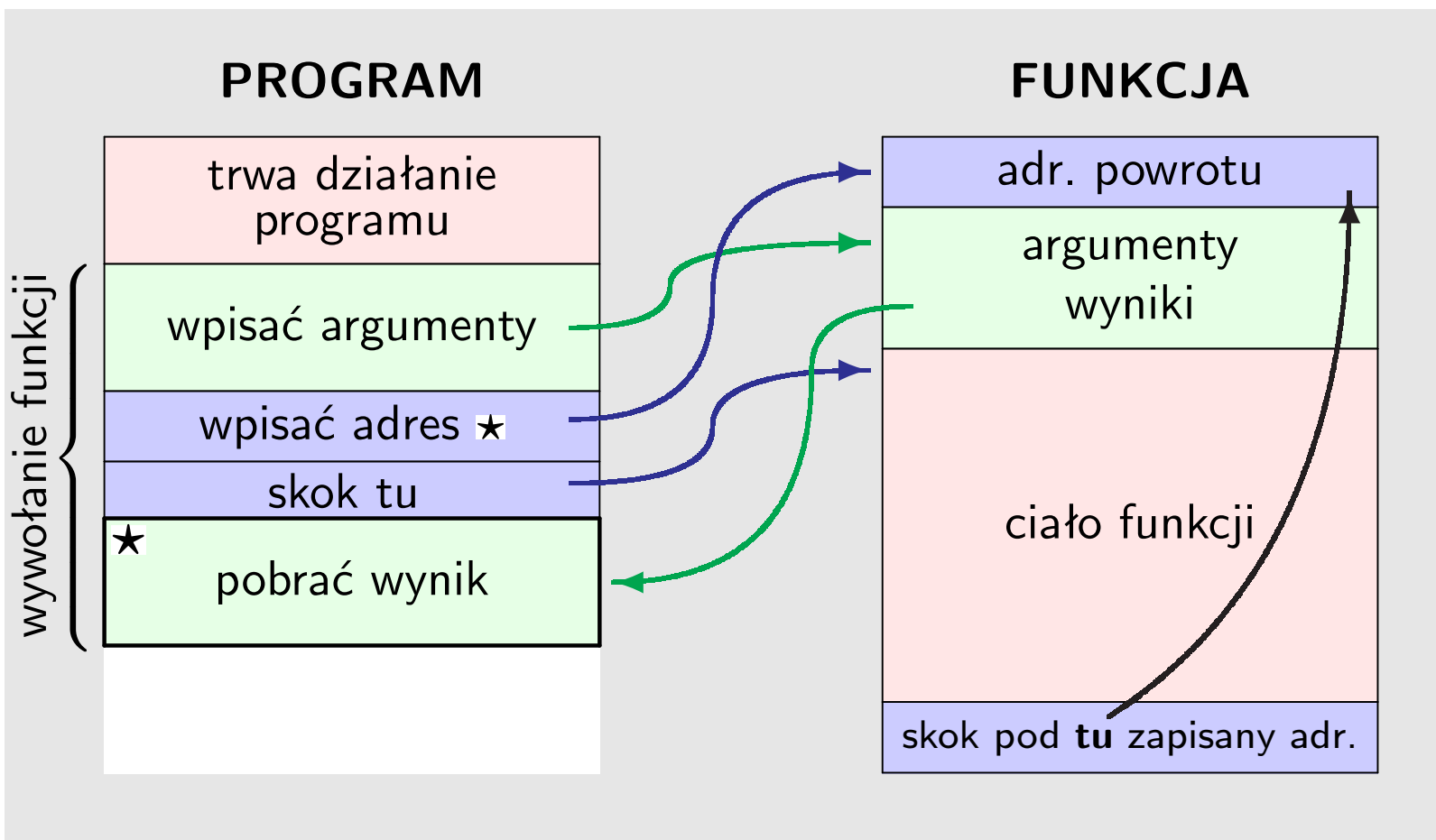
Duży program tworzony po kawałku



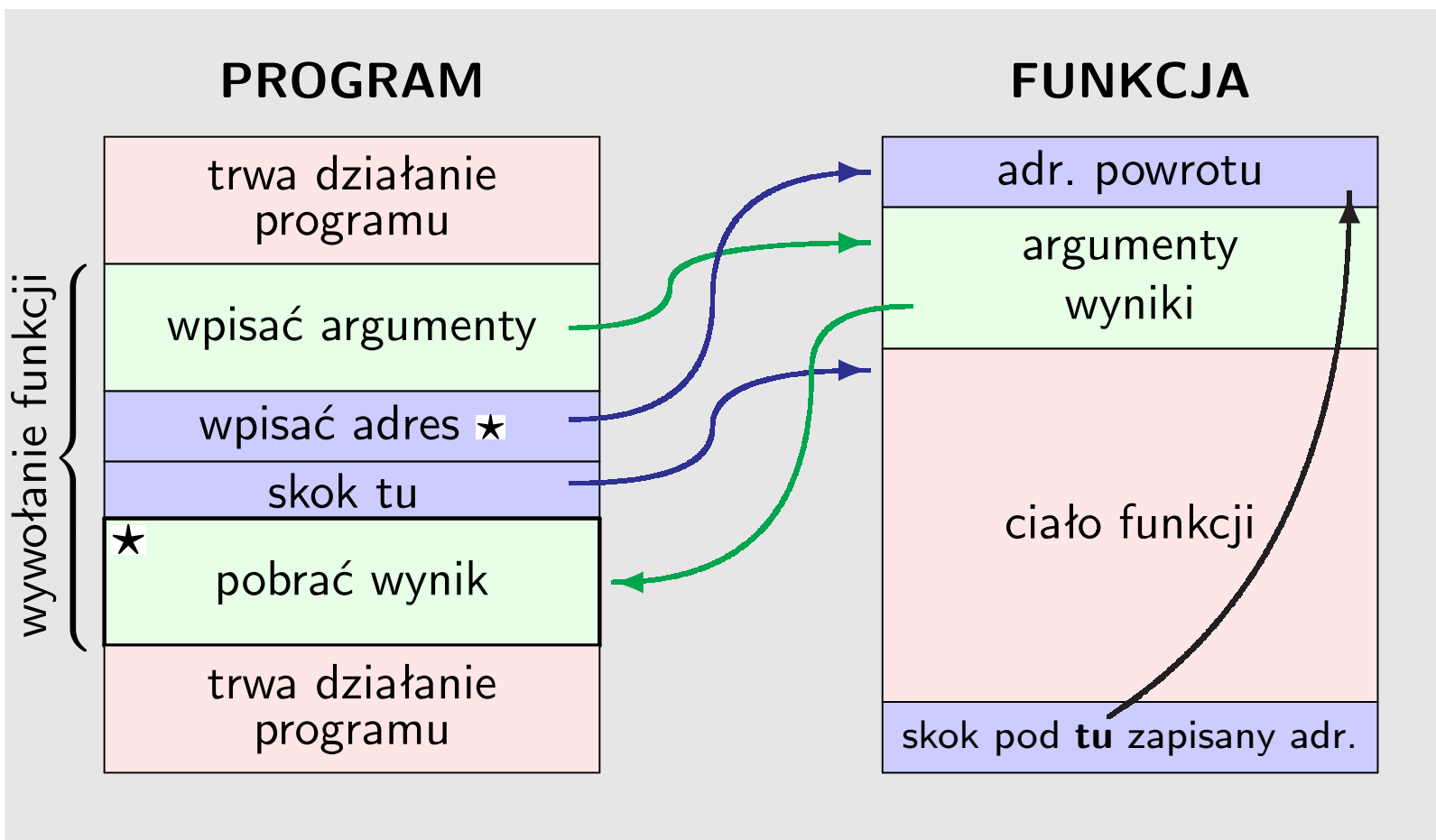
Duży program tworzony po kawałku



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:

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

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:

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

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:

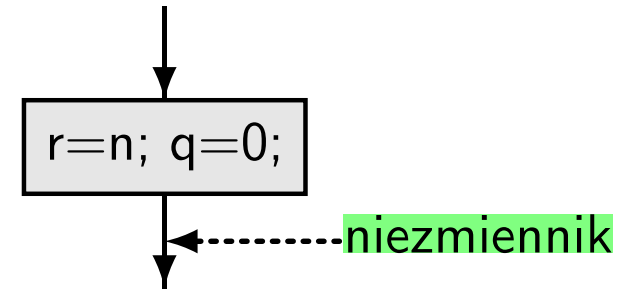
$$\underbrace{n = k \cdot q + r}_{\text{niezmiennik}} \quad \wedge \quad \underbrace{0 \leq r < k}_{\text{war. końca}}$$

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:

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

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.

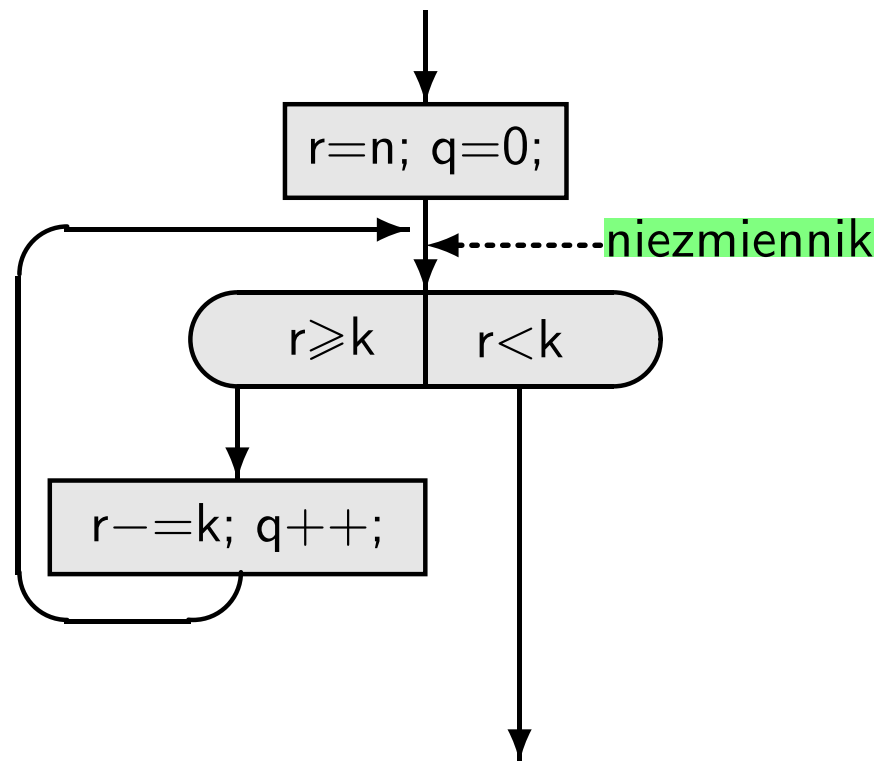
Dzielimy n przez k

zakładamy: $n, k \in \mathbb{Z}$, $k > 0$

iloraz całkowity: q reszta: r

więc chcemy mieć:

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



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.

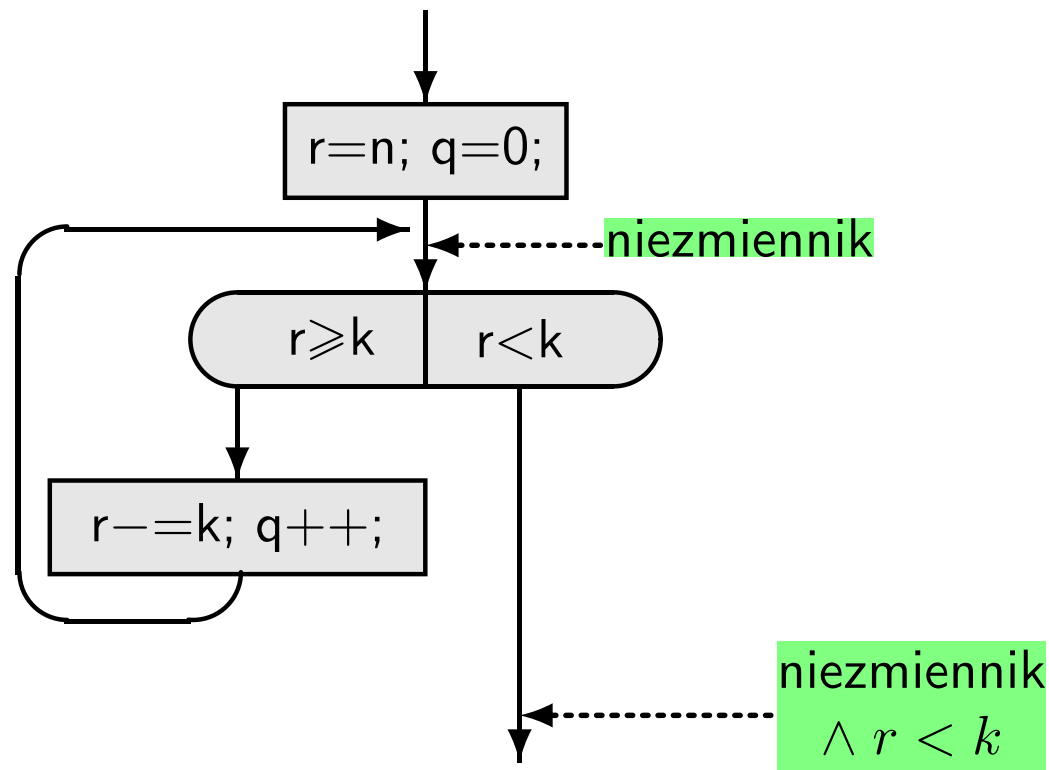
Dzielimy n przez k

zakładamy: $n, k \in \mathbb{Z}$, $k > 0$

iloraz całkowity: q reszta: r

więc chcemy mieć:

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



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

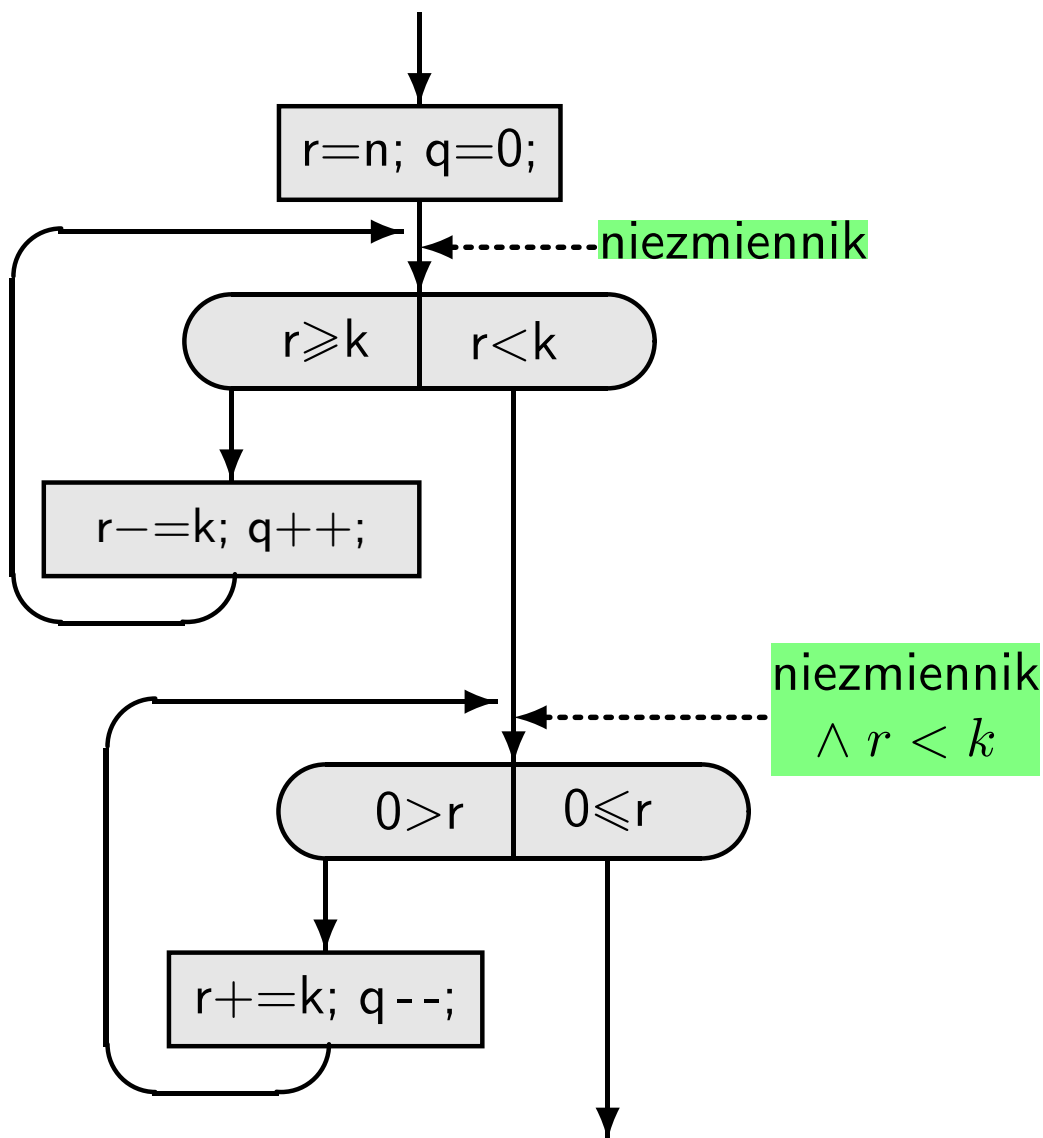
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$

iloraz całkowity: q reszta: r

więc chcemy mieć:

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

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

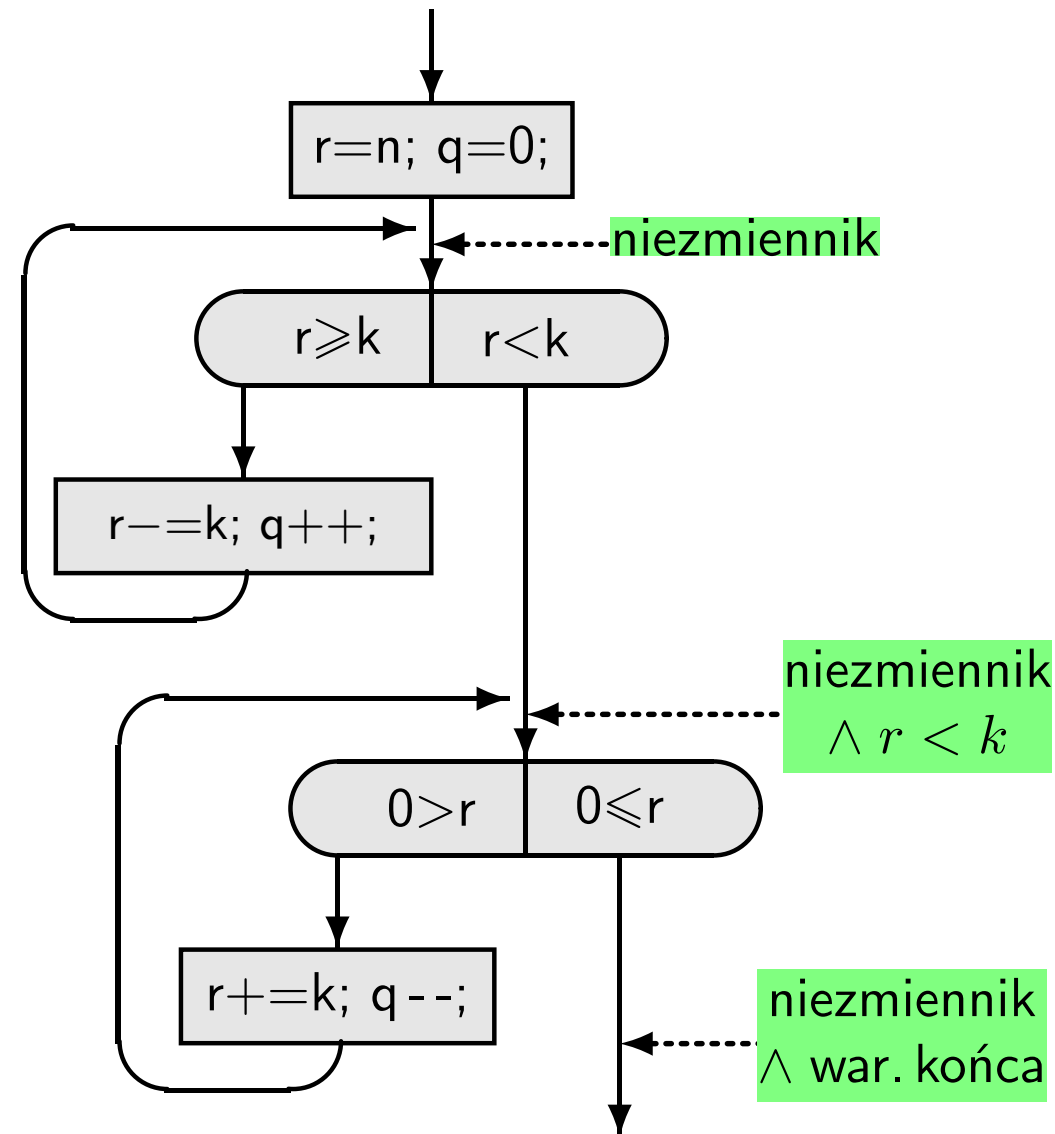
zakładamy: $n, k \in \mathbb{Z}$, $k > 0$

iloraz całkowity: q reszta: r

więc chcemy mieć:

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

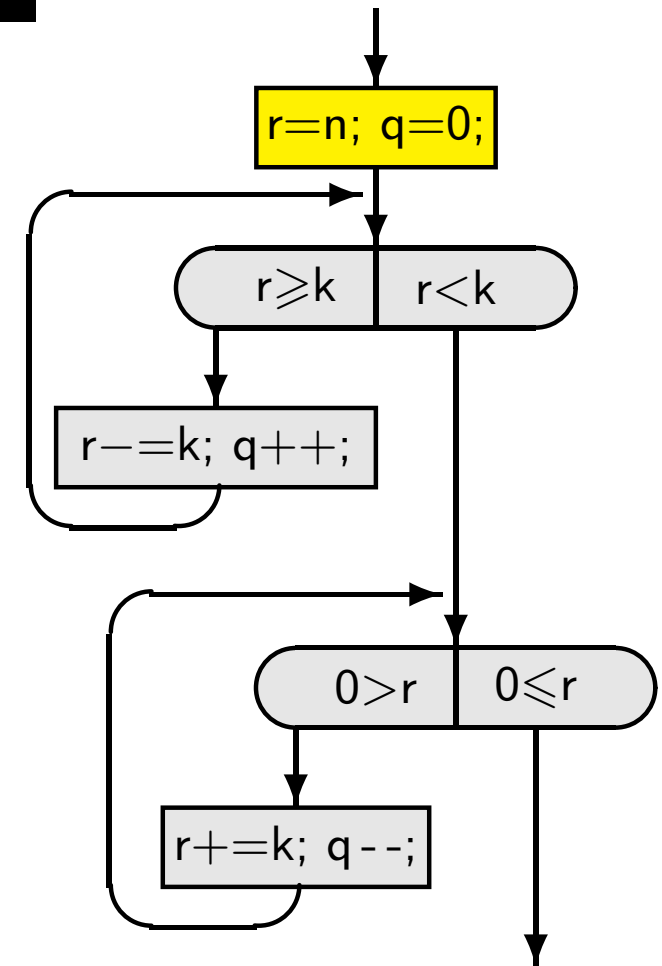
zgodne z C dla $n, k > 0$
poza tym — hmmm...



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

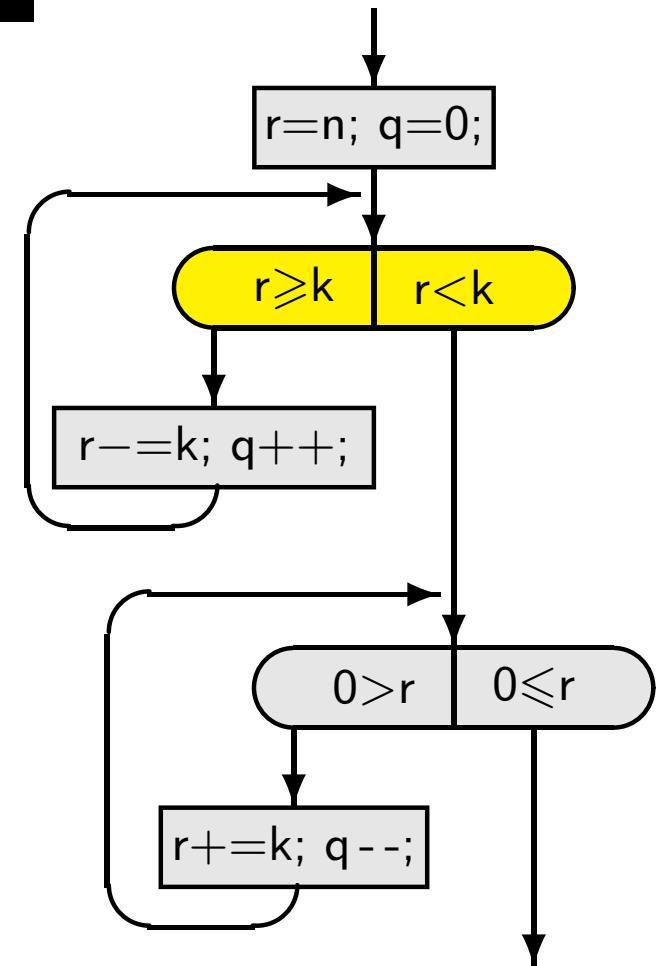
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

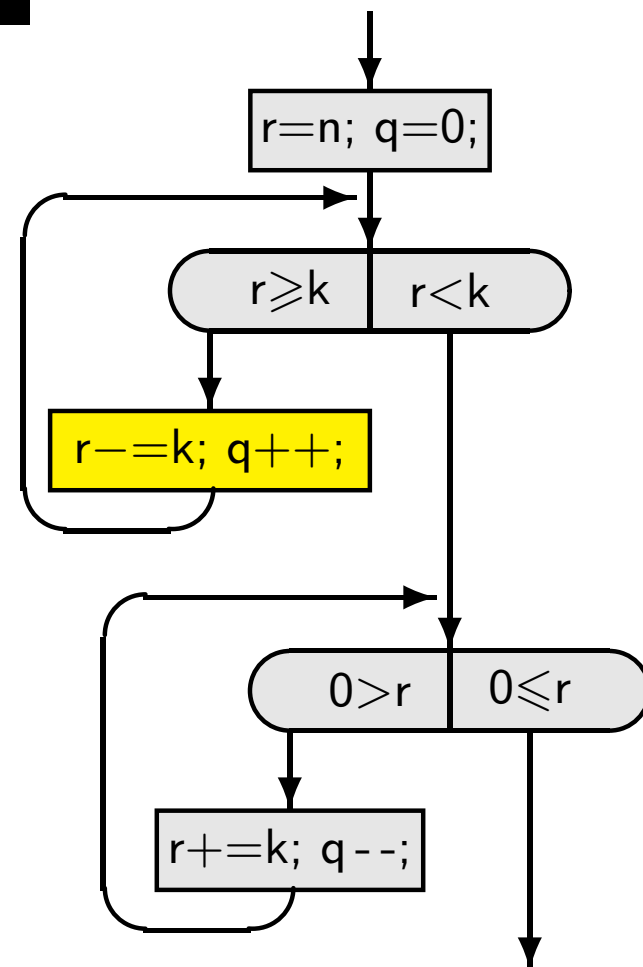
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

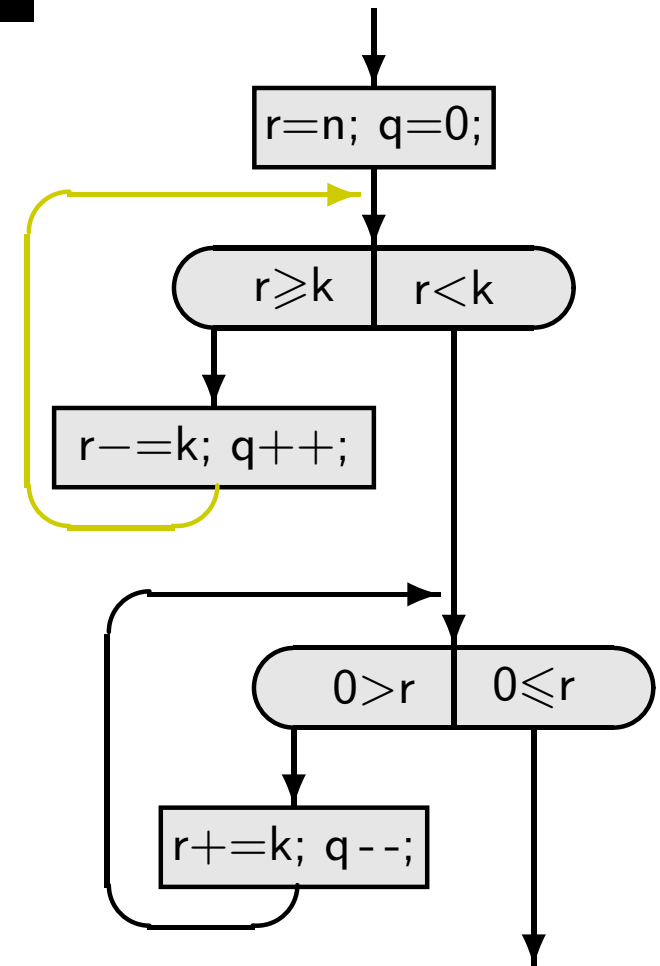
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

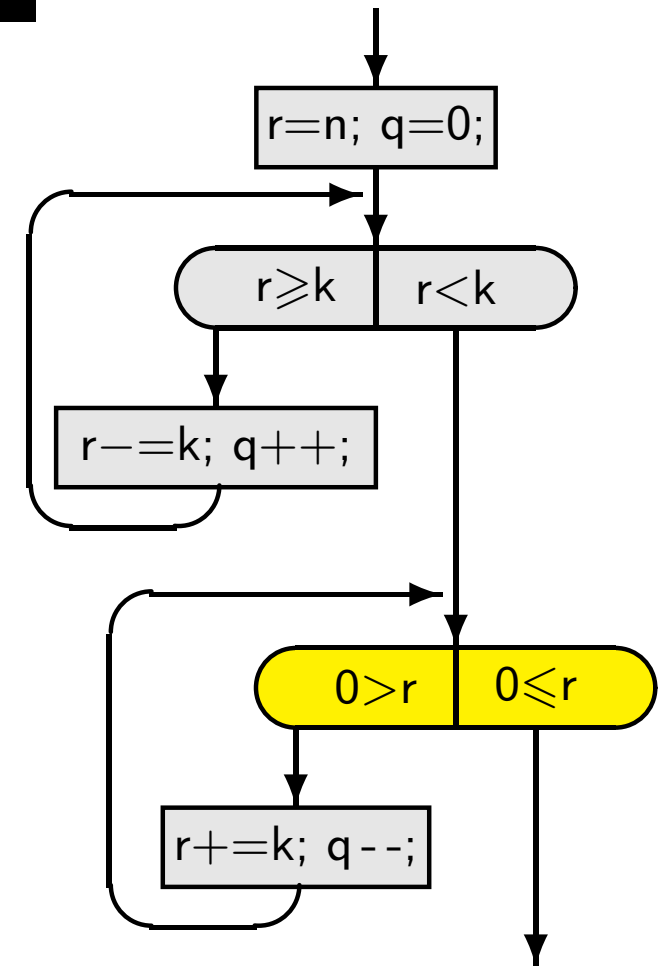
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

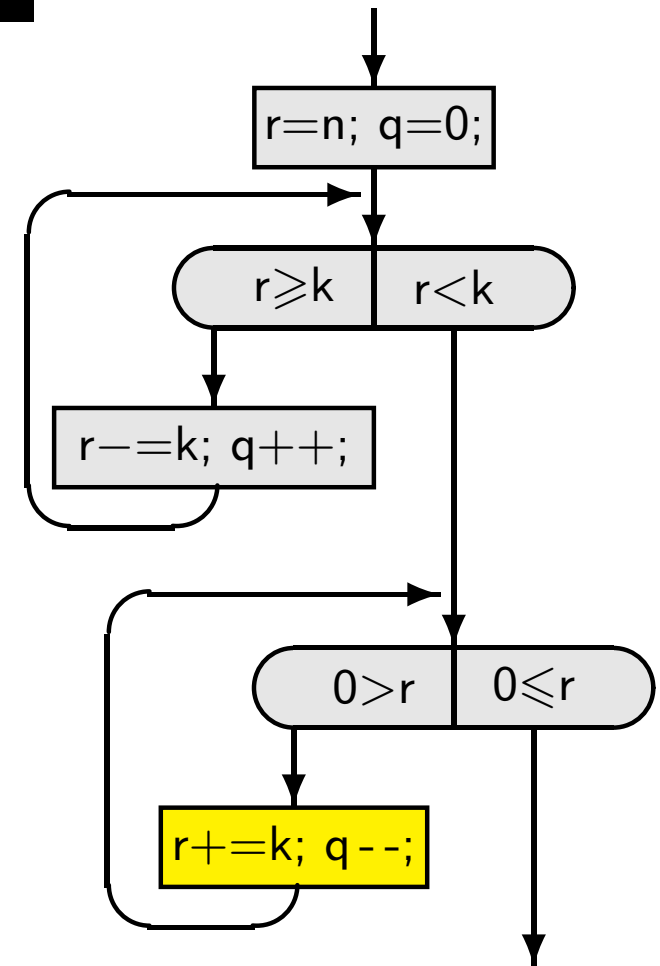
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

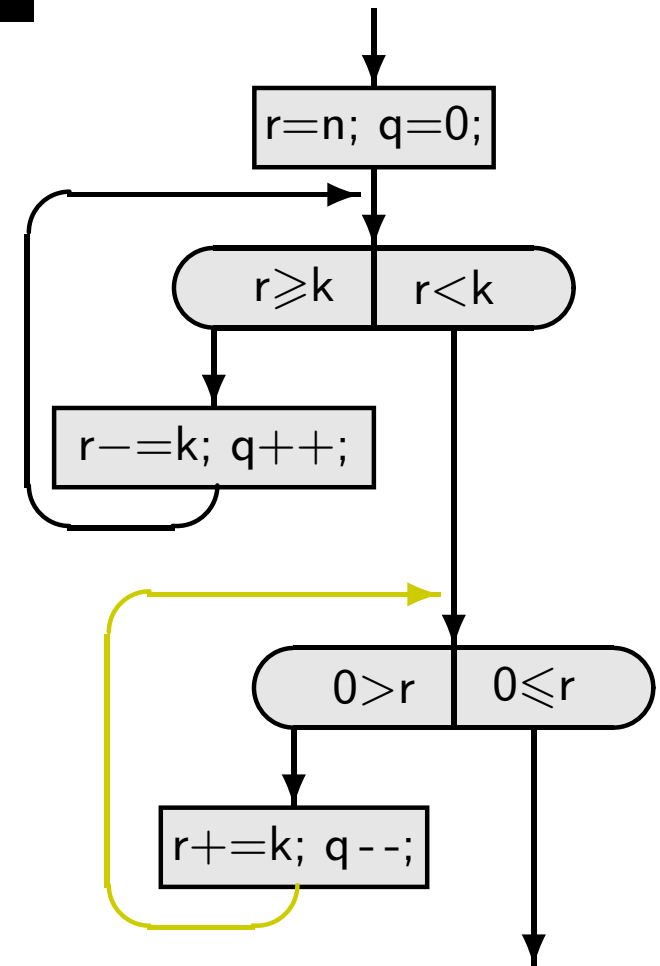
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

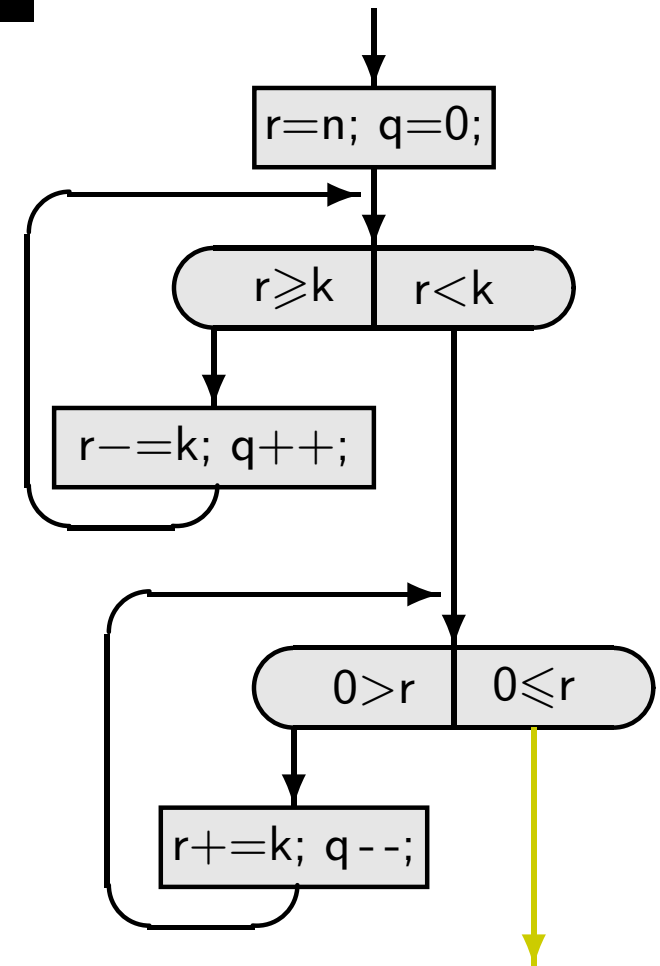
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

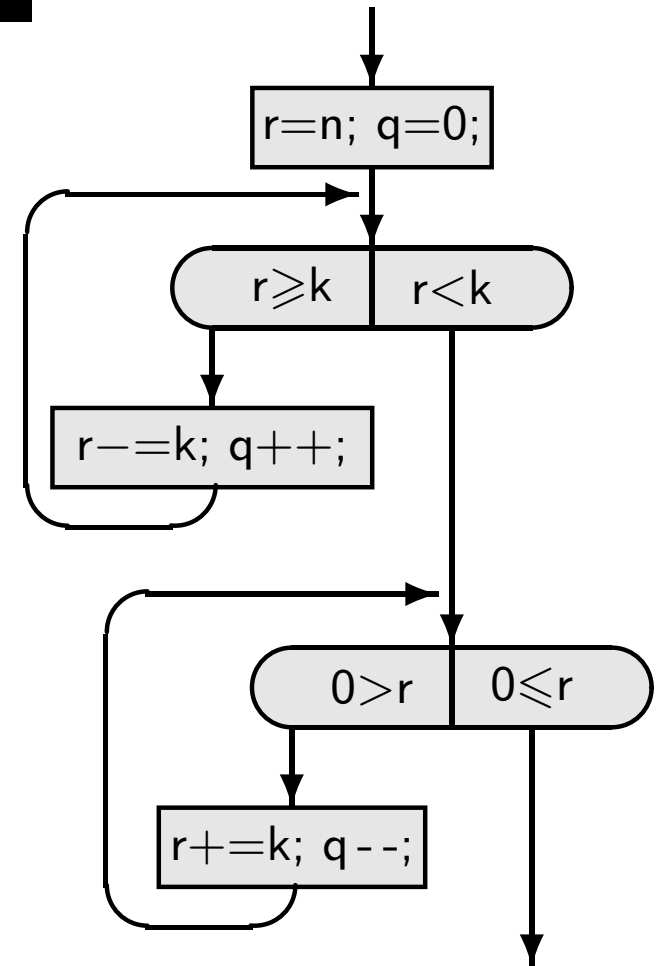
1: 0	# n (dana)	13: sub	k
2: 0	# k (dana)	14: store	r
3: 0	# q (wynik)	15: incr	q
4: 0	# r (wynik)	16: goto	petla1
5: loada	n # POCZATEK	17: loada	r # petla2
6: store	r	18: plsgoto	KONIEC
7: loadn	0	19: zergoto	KONIEC
8: store	q	20: loada	r
9: loada	r # petla1	21: add	k
10: sub	k	22: store	r
11: mingoto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



Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

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

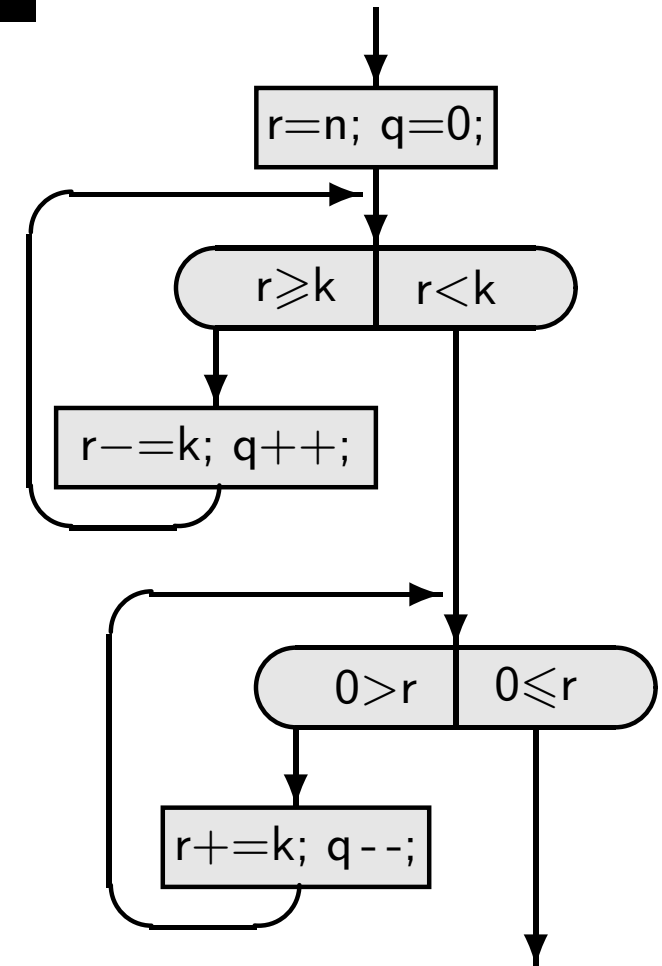


Żeby w innym programie móc użyć tego dzielenia — trzeba zrobić z niego **podprogram**.

Duży program tworzony po kawałku

Kawałek 1: (dzielenie z resztą)

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



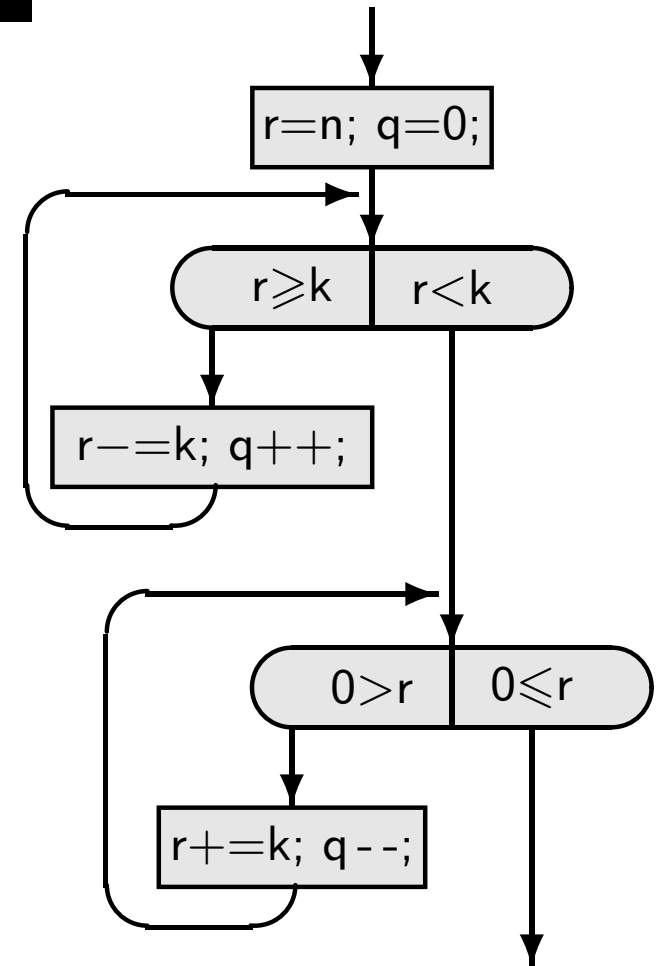
Żeby w innym programie móc użyć tego dzielenia — trzeba zrobić z niego **podprogram**.

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



To już jest **podprogram**. Wywołanie:

- do komórek 1 i 2 wpisać dzielną i dzielnik;
- do komórki 0 wpisać adres powrotu;
- skoczyć do komórki 5;
- po powrocie wyniki będą w komórkach 3 i 4.

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:

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

Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

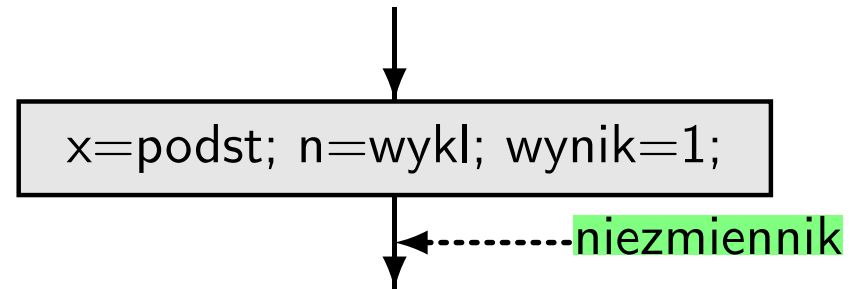
Za każdym obrotem pętli

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

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

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



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

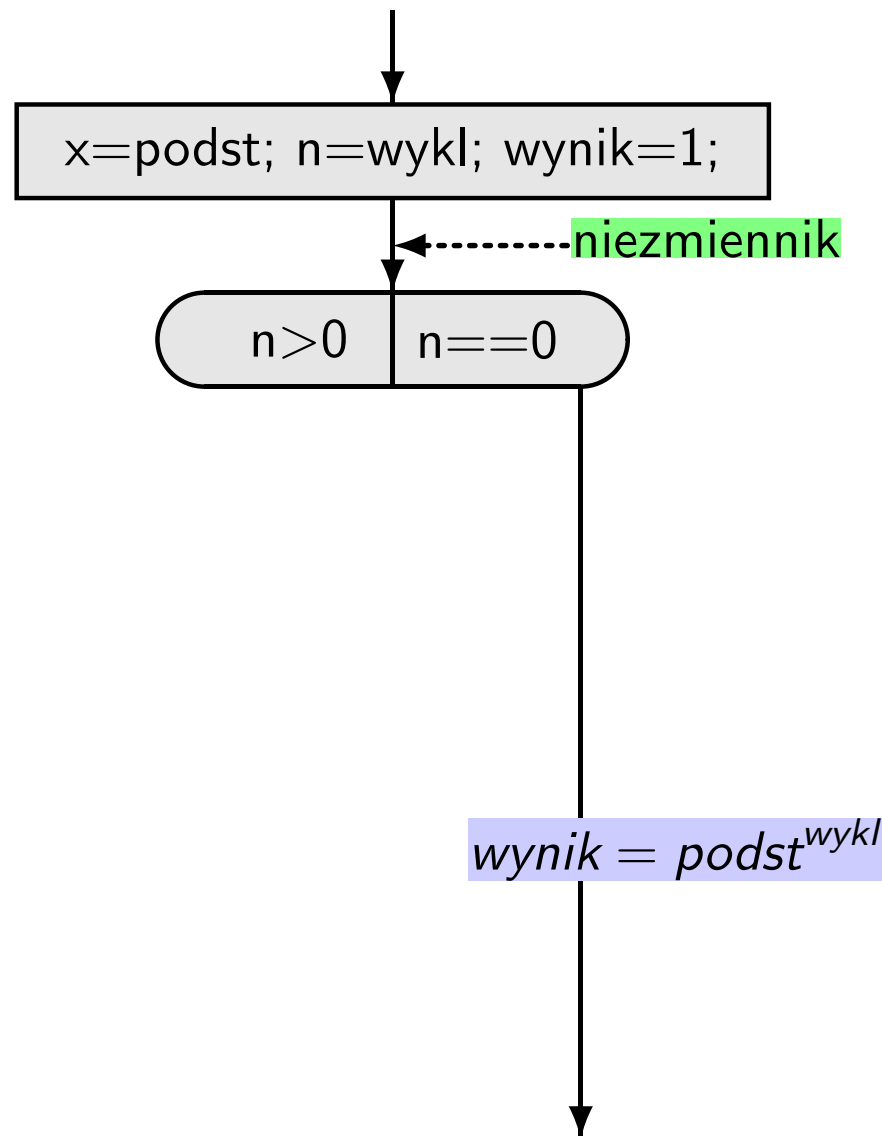
Za każdym obrotem pętli

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

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

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



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

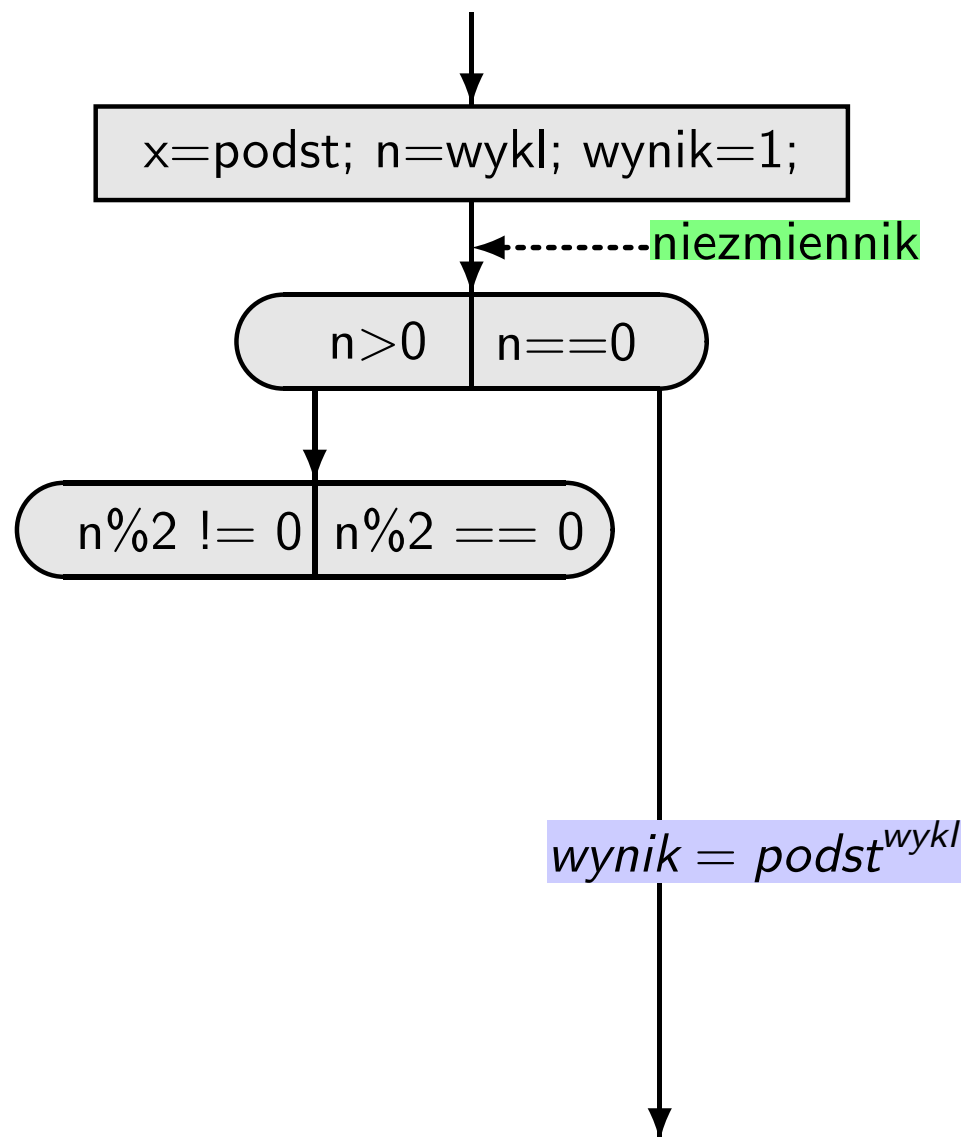
Za każdym obrotem pętli

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

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

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



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

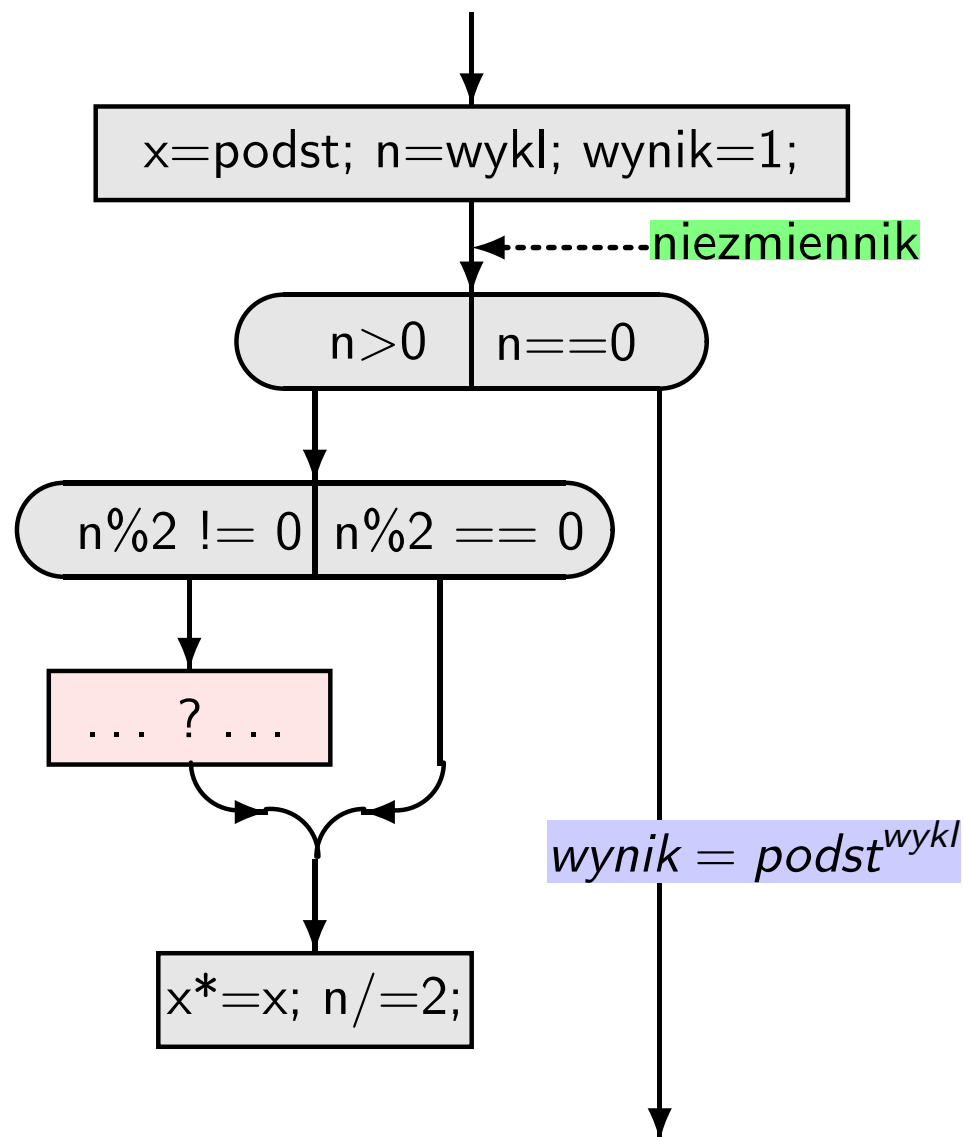
Za każdym obrotem pętli

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

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

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



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

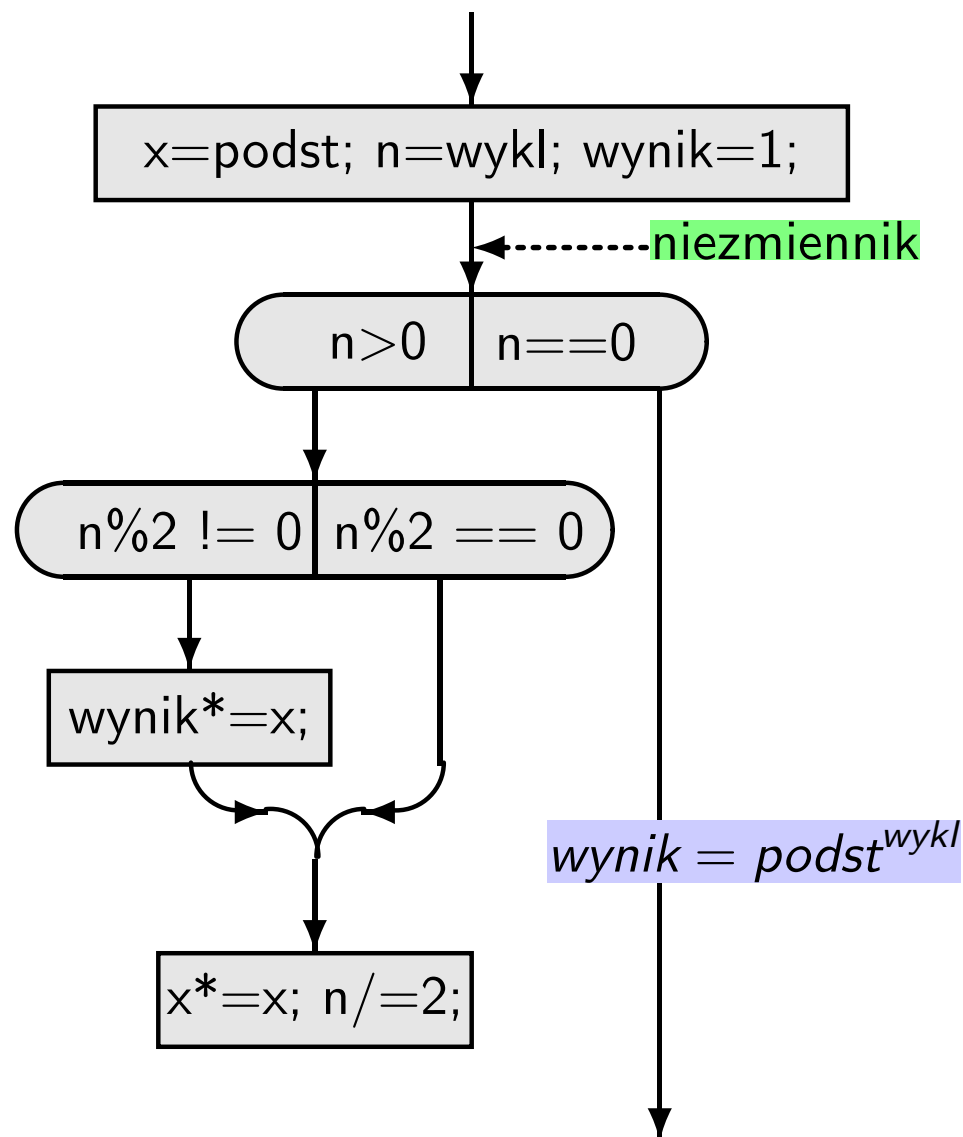
Za każdym obrotem pętli

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

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

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



Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

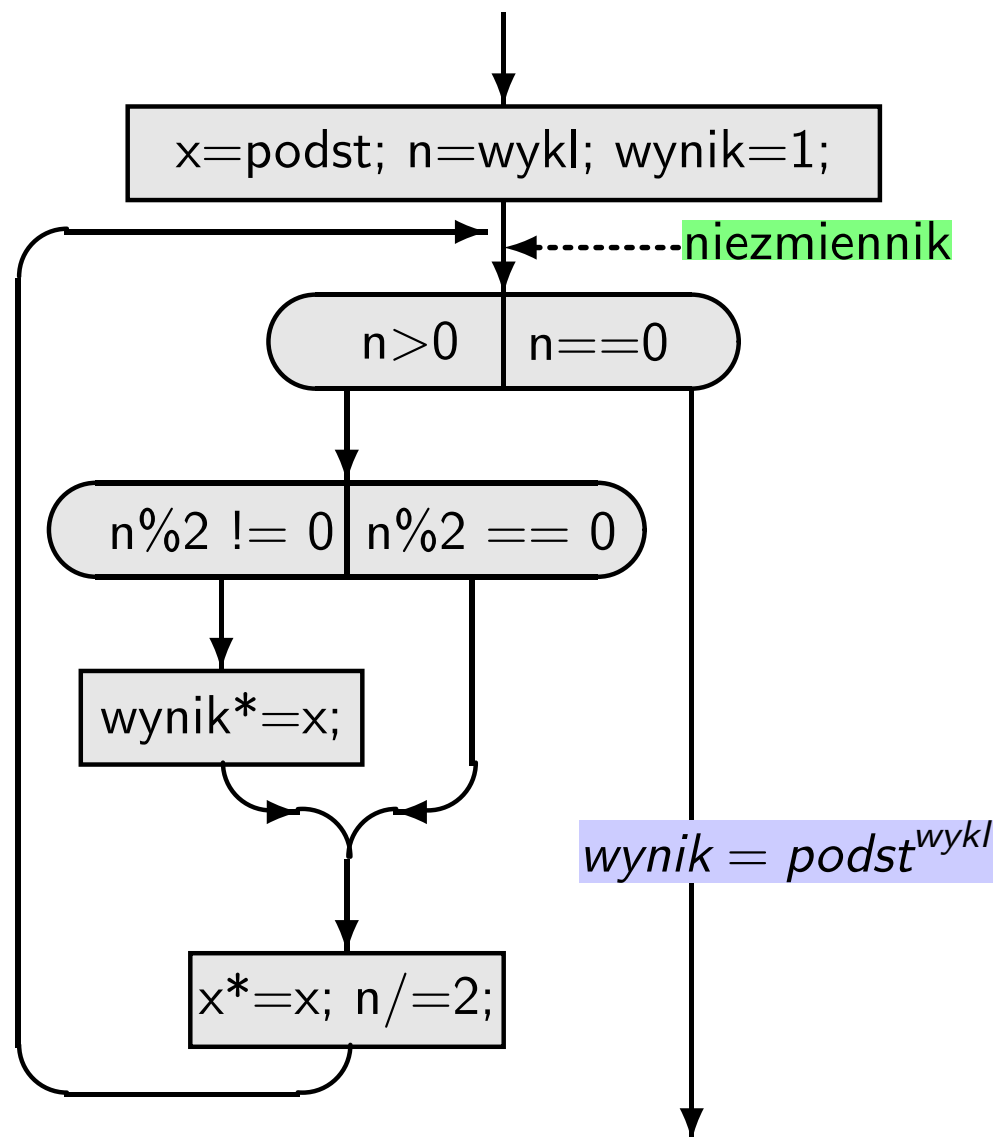
Za każdym obrotem pętli

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

ale *uważać* na nieparzyste wykładniki!

Chcemy mieć:

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



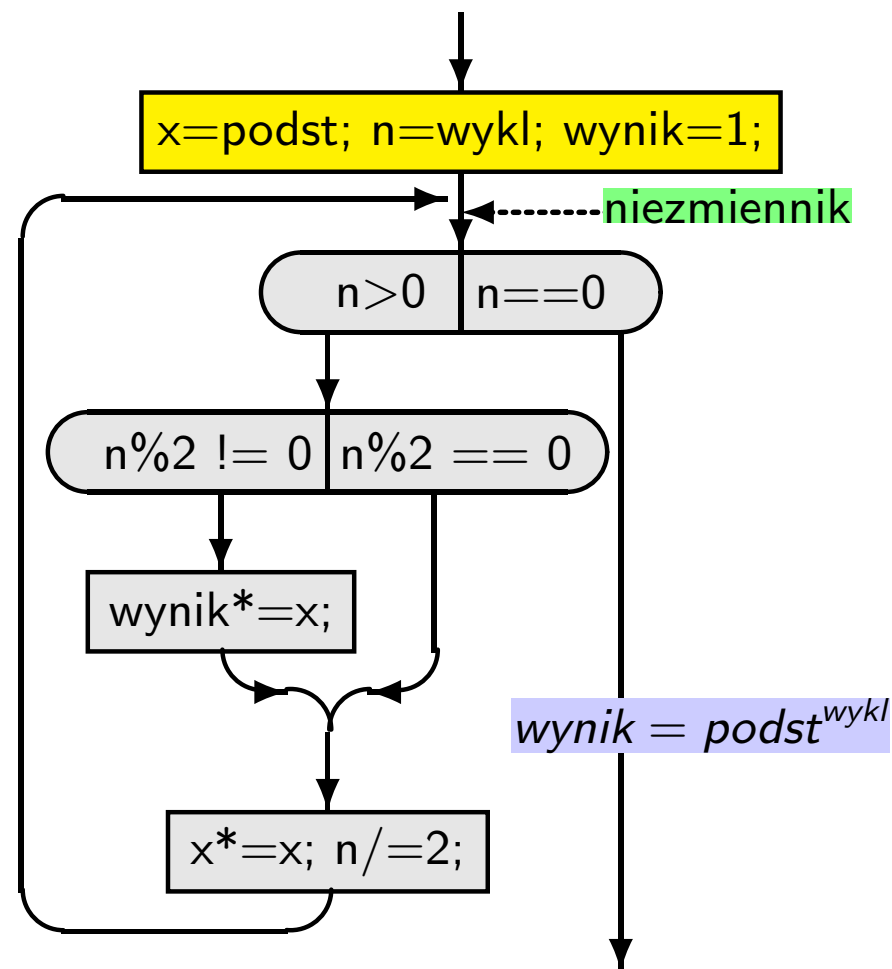
Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

```

200: loada  podst
201: store  x
202: loada  cykl
203: store  n
204: loadn  1
205: store  wynik
206: loada  n # while
207: zergoto koniec
208: store  1 # dzielna n
209: loadn  2
210: store  2 # dzielnik 2
211: loadn  214
212: store  0 # adr.powr.: 214
213: goto   5 # wywol. podprog.
      . . . . .

```



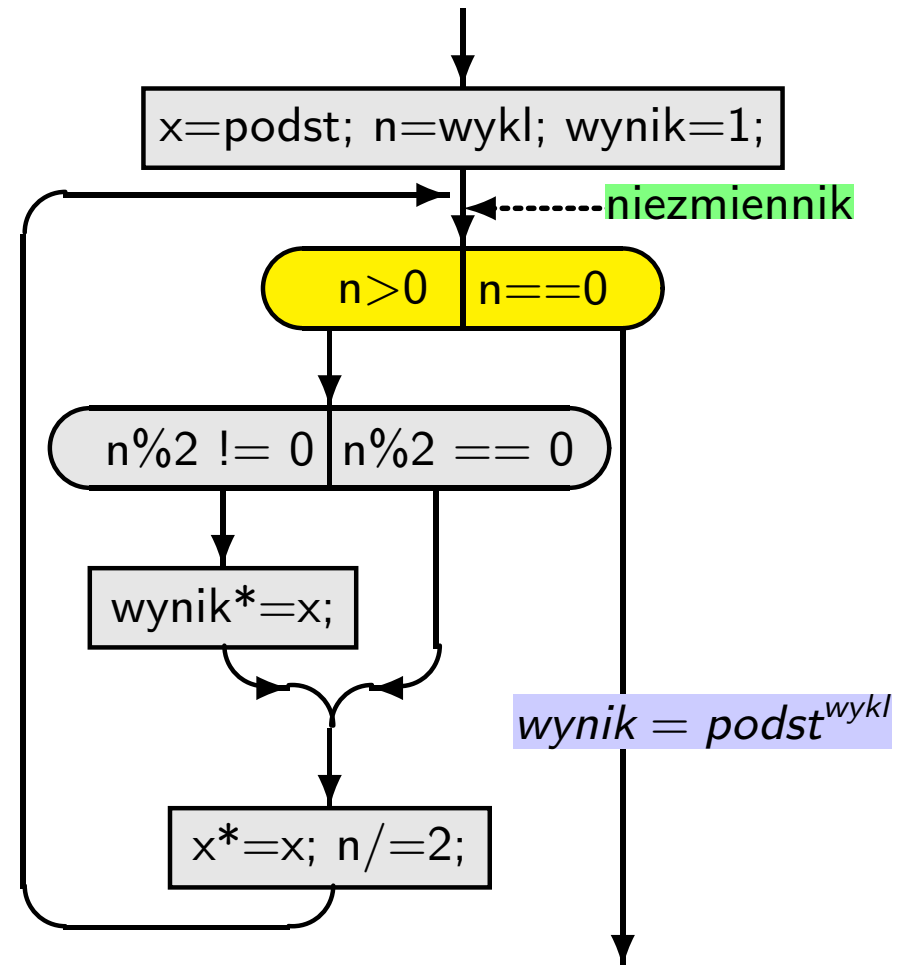
Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

```

200: loada  podst
201: store   x
202: loada  cykl
203: store   n
204: loadn  1
205: store   wynik
206: loada  n # while
207: zergoto koniec
208: store   1 # dzielna n
209: loadn   2
210: store   2 # dzielnik 2
211: loadn   214
212: store   0 # adr.powr.: 214
213: goto    5 # wywol. podprog.
. . . . .

```



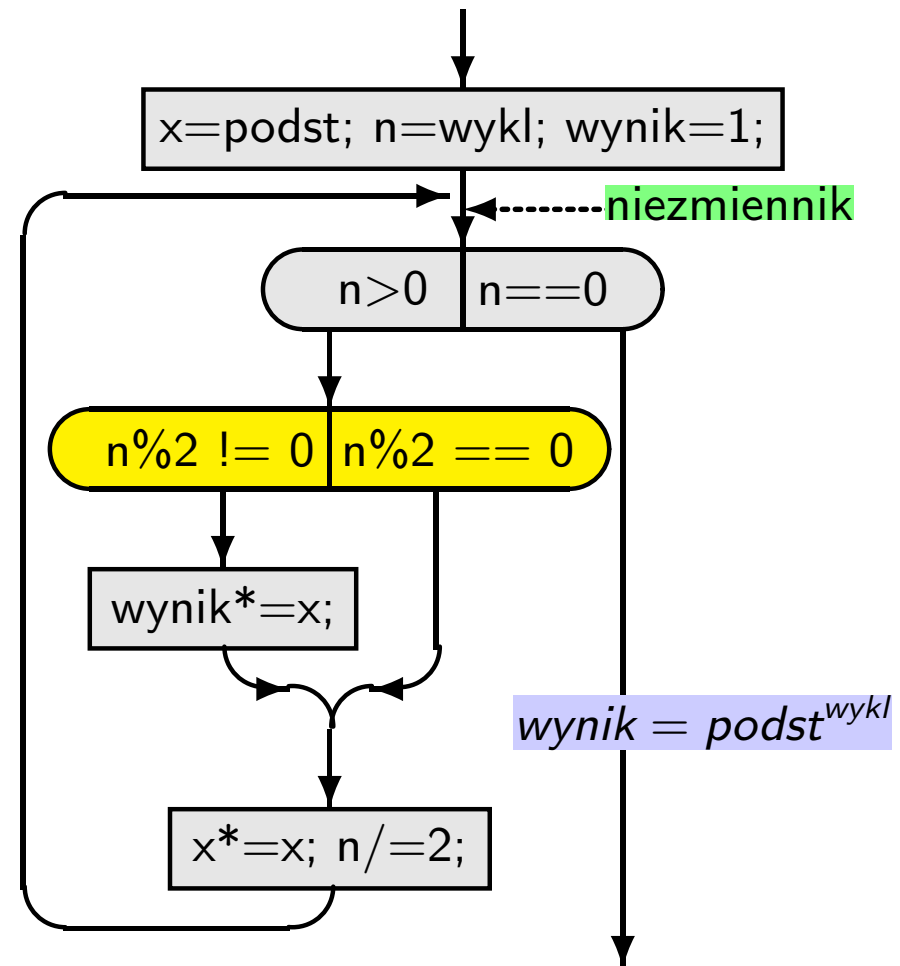
Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

```

200: loada  podst
201: store   x
202: loada  cykl
203: store   n
204: loadn  1
205: store   wynik
206: loada  n # while
207: zergoto koniec
208: store   1 # dzielna n
209: loadn   2
210: store   2 # dzielnik 2
211: loadn   214
212: store   0 # adr.powr.: 214
213: goto    5 # wywol. podprog.
      . . . . .

```



wpisanie parametrów

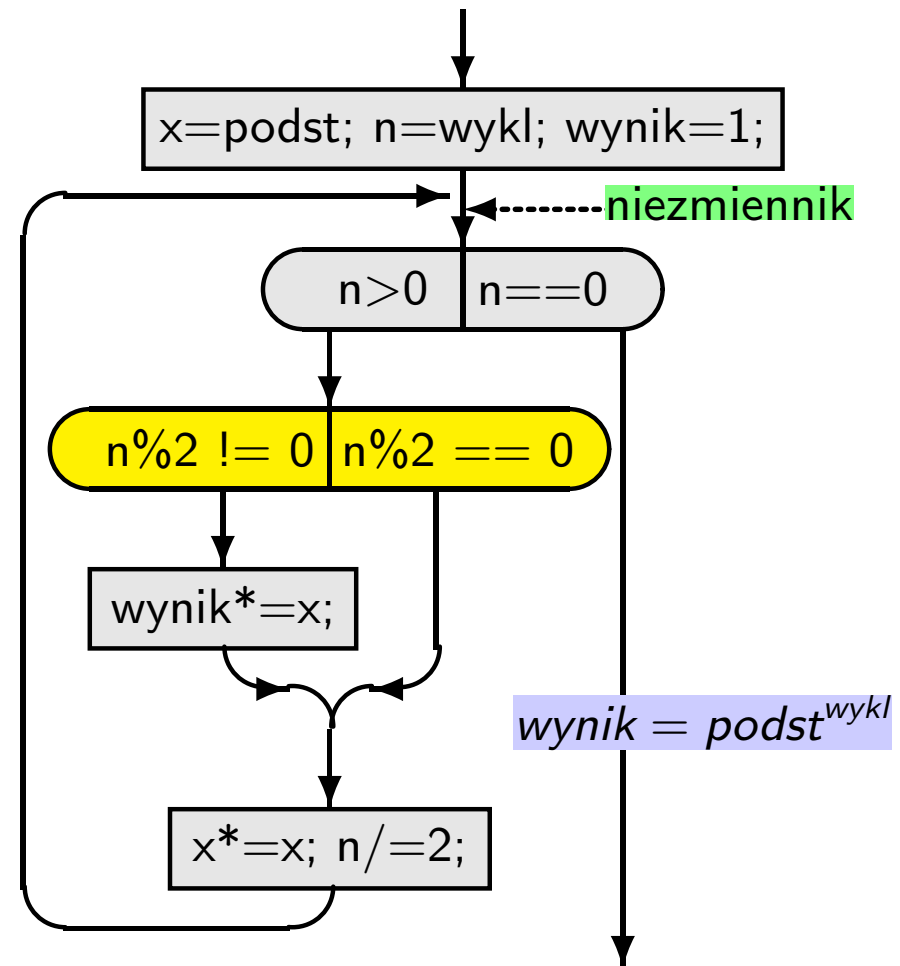
Duży program tworzony po kawałku

Kawałek 2: (potęgowanie binarne)

```

200: loada  podst
201: store   x
202: loada  wyk1
203: store   n
204: loadn  1
205: store   wynik
206: loada  n # while
207: zergoto koniec
208: store  1 # dzielna n
209: loadn  2
210: store  2 # dzielnik 2
211: loadn  214
212: store  0 # adr.powr.: 214
213: goto   5 # wywol. podprog.
      . . . . .

```



wpisanie parametrów i
wywołanie podprogramu

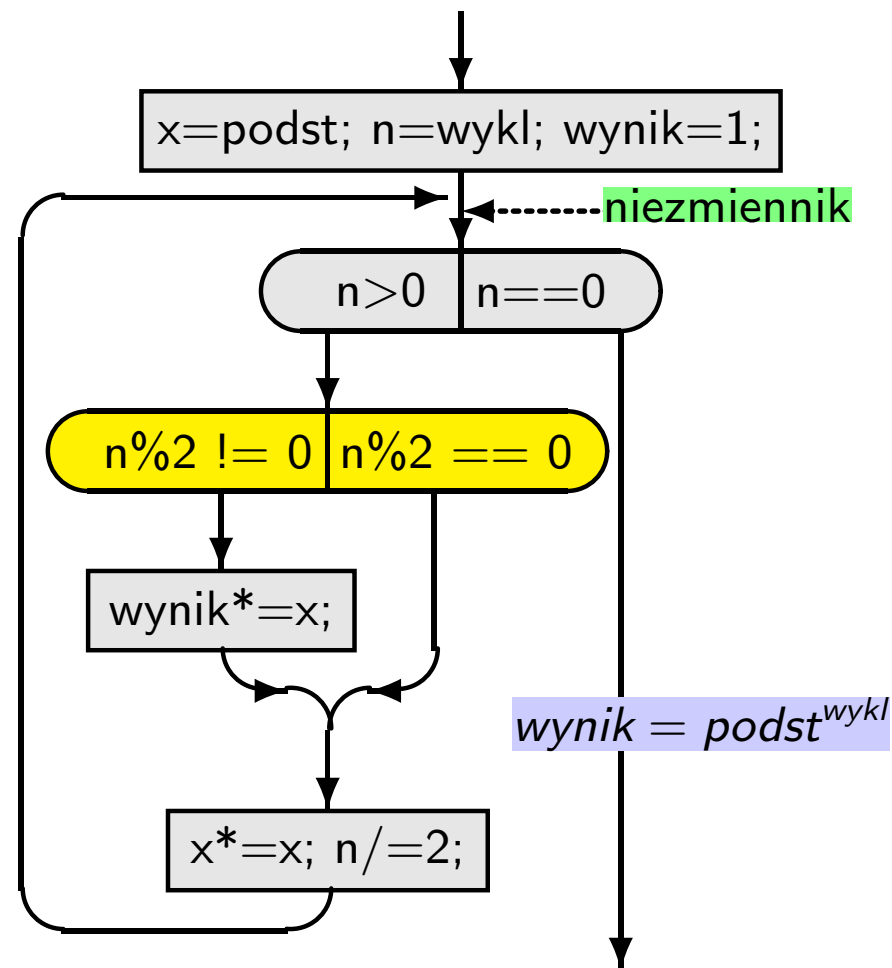
Duży program tworzony po kawałku

Kawałek 2 — c.d.: (potęgowanie binarne)

```

213: goto    5 # wywol. podprog.
214: loada   4 # pobrac reszte
215: zergoto 219 # nie mnozyc
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: ...      # podst } tu
227: ...      # wykł  } wpisać
228: 0         # wynik } dane
229: 0         # x
230: 0         # n
END

```



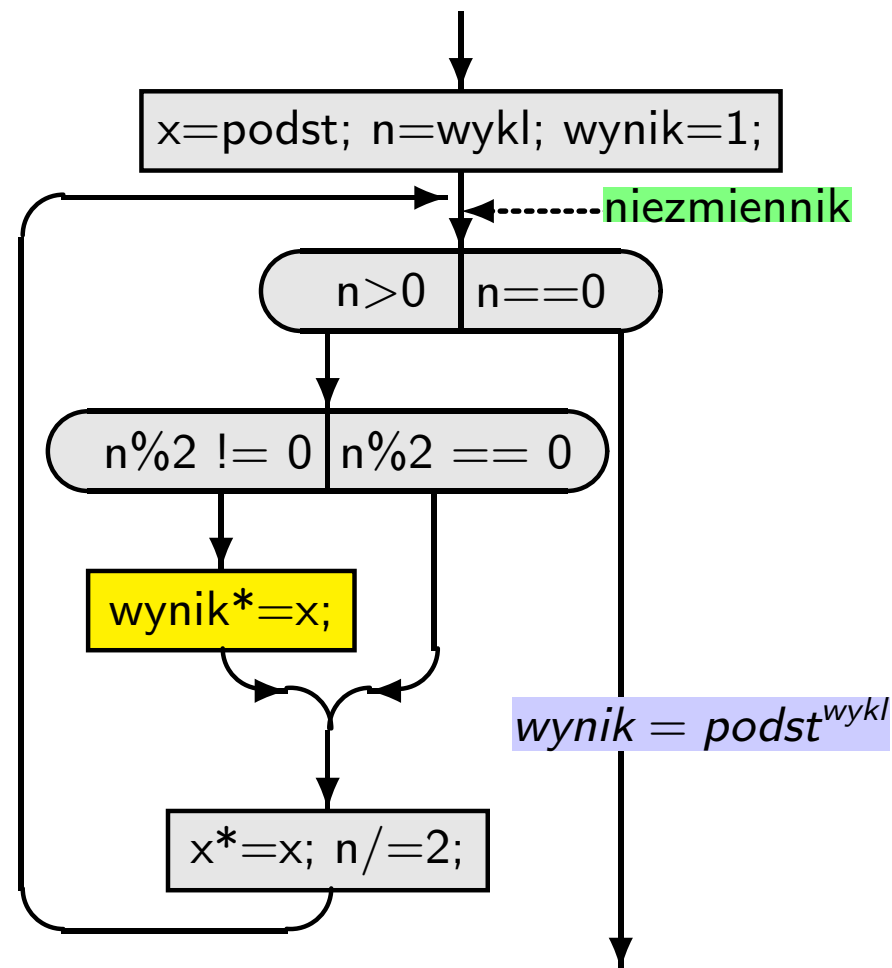
Duży program tworzony po kawałku

Kawałek 2 — c.d.: (potęgowanie binarne)

```

213: goto    5 # wywol. podprog.
214: loada   4 # pobrac reszte
215: zergoto 219 # nie mnozyc
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: ...     # podst } tu
227: ...     # wykł  } wpisać
228: 0        # wynik } dane
229: 0        # x
230: 0        # n
END

```



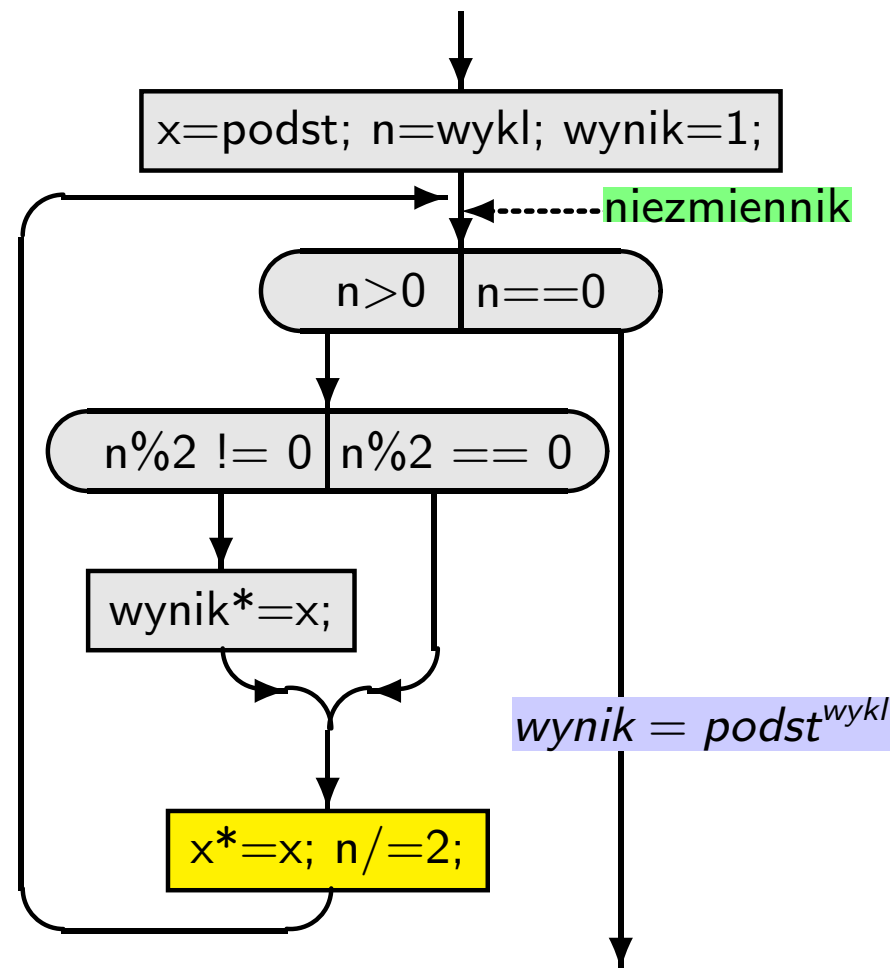
Duży program tworzony po kawałku

Kawałek 2 — c.d.: (potęgowanie binarne)

```

213: goto    5 # wywol. podprog.
214: loada   4 # pobrac reszte
215: zergoto 219 # nie mnozyc
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: ...     # podst } tu
227: ...     # wykł  } wpisać
228: 0        # wynik } dane
229: 0        # x
230: 0        # n
END

```



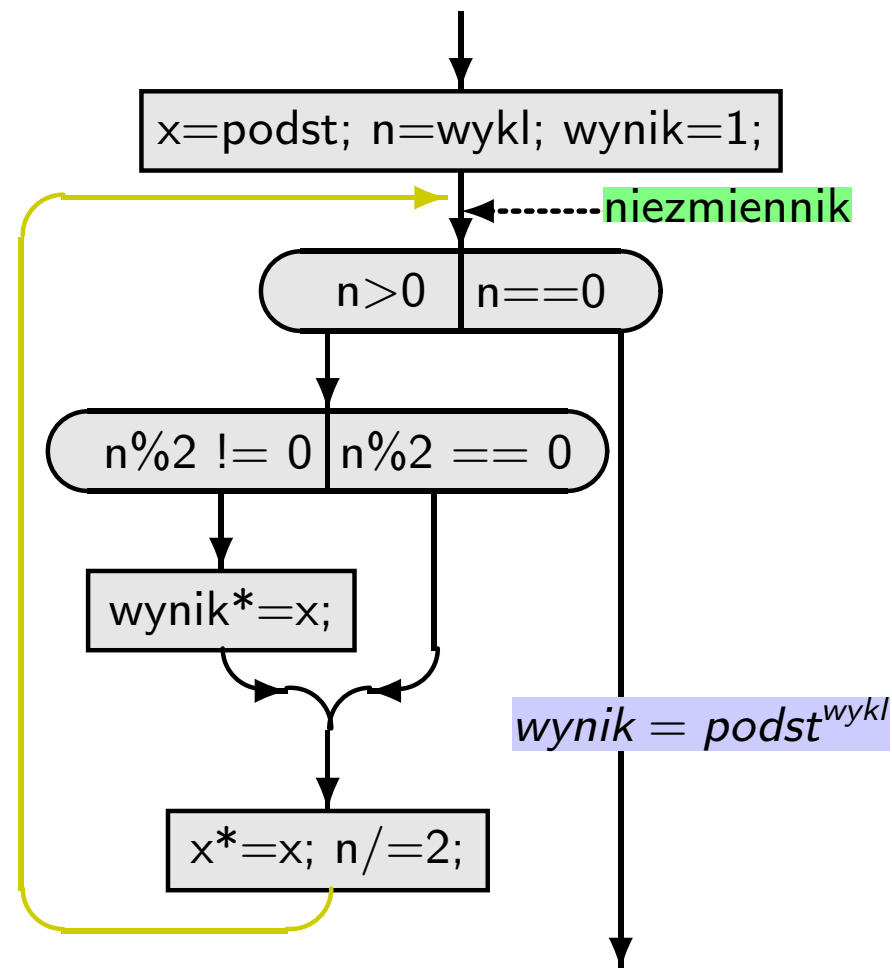
Duży program tworzony po kawałku

Kawałek 2 — c.d.: (potęgowanie binarne)

```

213: goto    5 # wywol. podprog.
214: loada   4 # pobrac reszte
215: zergoto 219 # nie mnozyc
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: ...      # podst } tu
227: ...      # wykl  } wpisać
228: 0         # wynik } dane
229: 0         # x
230: 0         # n
END

```



Duży program tworzony po kawałku

Kawałek 2 — c.d.: (potęgowanie binarne)

```

213: goto    5 # wywol. podprog.
214: loada   4 # pobrac reszte
215: zergoto 219 # nie mnozyc
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: ...     # podst } tu
227: ...     # wyk1  } wpisać
228: 0       # wynik } dane
229: 0       # x
230: 0       # n
END

```

