

Podstawowe elementy programu

Zestaw komend stojący do dyspozycji programisty

Podstawowe elementy programu

Zestaw komend stojący do dyspozycji programisty

- zależy od języka programowania

Podstawowe elementy programu

Zestaw komend stojący do dyspozycji programisty

- zależy od języka programowania;
- jest ograniczony

Podstawowe elementy programu

Zestaw komend stojący do dyspozycji programisty

- zależy od języka programowania;
- jest ograniczony;
- jest na tyle bogaty, że daje się z nich złożyć (jak z klocków) sensowne programy.

Podstawowe elementy programu

Zestaw komend stojący do dyspozycji programisty

- zależy od języka programowania;
- jest ograniczony;
- jest na tyle bogaty, że daje się z nich złożyć (jak z klocków) sensowne programy.

Umiejętność programowania jest sztuką odpowiedniego układania tych klocków.

Podstawowe elementy programu

Zestaw komend stojący do dyspozycji programisty

- zależy od języka programowania;
- jest ograniczony;
- jest na tyle bogaty, że daje się z nich złożyć (jak z klocków) sensowne programy.

Umiejętność programowania jest sztuką odpowiedniego układania tych klocków.

Do typowych komend należą:

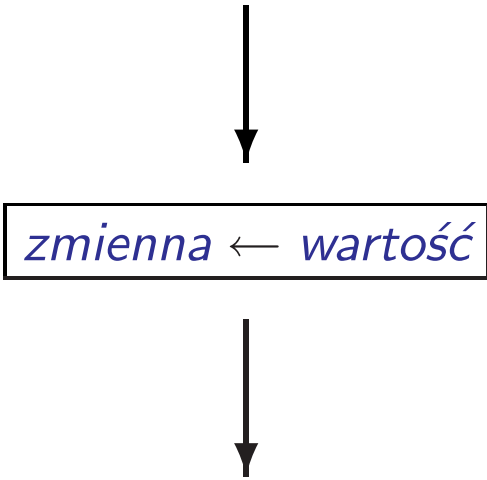


Podstawowe elementy programu

Komendy proste

Podstawowe elementy programu

Komendy proste:

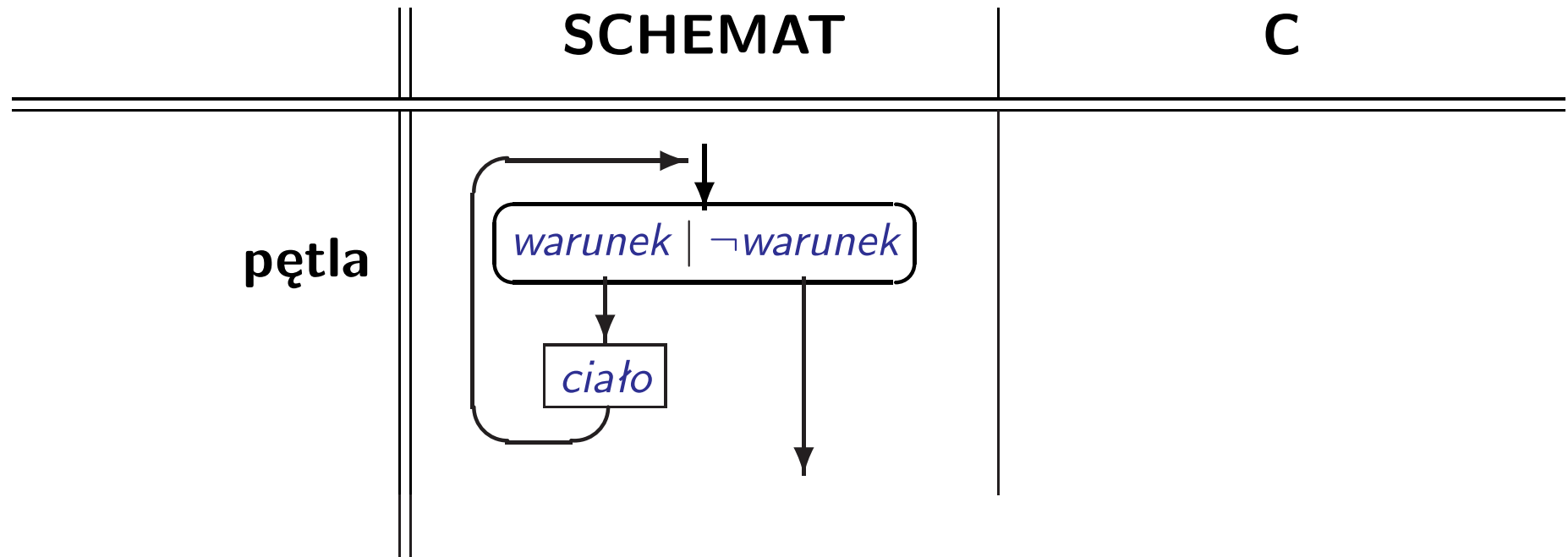
	SCHEMAT	C
przypisanie wartości zmiennej	 <p>The diagram shows a vertical arrow pointing down to a box containing the text <i>zmienna</i> ← <i>wartość</i>. Below the box is another vertical arrow pointing down.</p>	<i>zmienna = wartość ;</i>

Podstawowe elementy programu

Komendy złożone (strukturalne)

Podstawowe elementy programu

Komendy złożone (strukturalne):



Podstawowe elementy programu

Komendy złożone (strukturalne):

	SCHEMAT	C
pętla	<pre> graph TD Entry(()) --> Decision{warunek ¬warunek} Decision --> Body[ciało] Body --> Decision Decision --> Exit(()) </pre>	<pre>while (<i>warunek</i>) { <i>ciało</i> }</pre>

Podstawowe elementy programu

Komendy złożone (strukturalne):

	SCHEMAT	C
pętla	<pre> graph TD Entry(()) --> Decision{warunek ¬warunek} Decision --> true Entry Decision --> false Body[ciało] Body --> Entry Decision --> Exit(()) </pre>	<pre>while (<i>warunek</i>) { <i>ciało</i> }</pre>
rozgałęzienie	<pre> graph TD Entry(()) --> Decision{warunek ¬warunek} Decision --> true I1[instrukcja1] Decision --> false I2[instrukcja2] I1 --> Exit(()) I2 --> Exit </pre>	

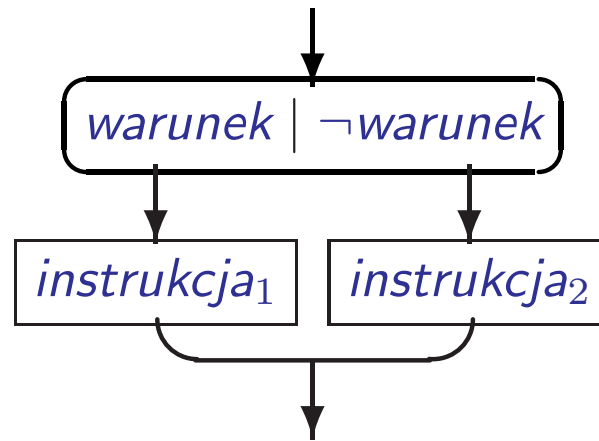
Podstawowe elementy programu

Komendy złożone (strukturalne):

	SCHEMAT	C
pętla	<pre> graph TD Entry(()) --> Decision{warunek ¬warunek} Decision -- warunek --> Entry Decision -- ¬warunek --> Body[ciało] Body --> Entry Decision -- ¬warunek --> Exit(()) </pre>	<pre>while (<i>warunek</i>) { <i>ciało</i> }</pre>
rozgałęzienie	<pre> graph TD Entry(()) --> Decision{warunek ¬warunek} Decision -- warunek --> Instr1[instrukcja1] Decision -- ¬warunek --> Instr2[instrukcja2] Instr1 --> Exit(()) Instr2 --> Exit </pre>	<pre>if (<i>warunek</i>) { <i>instrukcja</i>₁ } else { <i>instrukcja</i>₂ }</pre>

Rozgałęzienie warunkowe

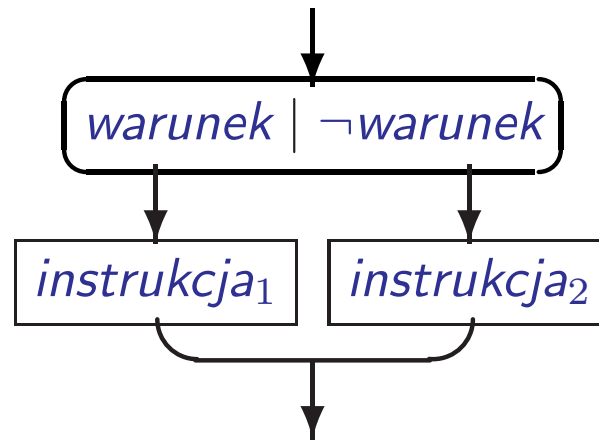
pełne



```
if ( warunek ) {  
    instrukcja1  
} else {  
    instrukcja2  
}
```

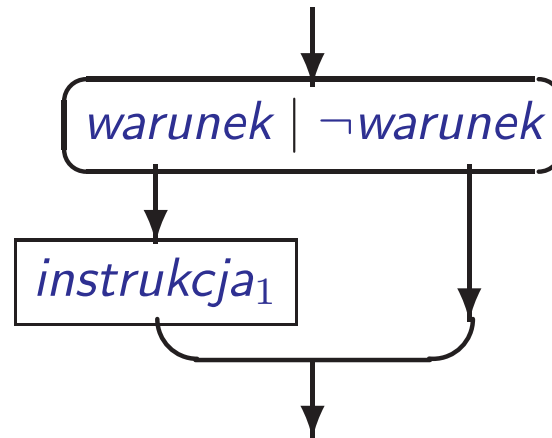
Rozgałęzienie warunkowe

pełne



```
if ( warunek ) {  
    instrukcja1  
} else {  
    instrukcja2  
}
```

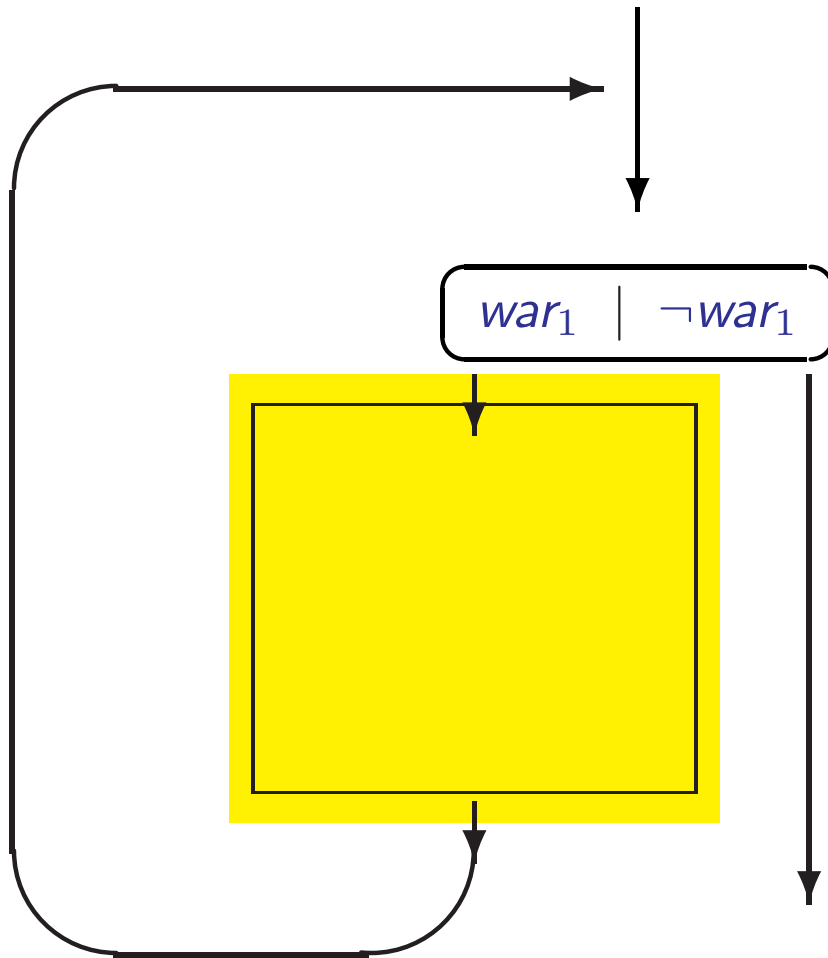
uproszczone



```
if ( warunek ) {  
    instrukcja1  
}
```

Budowa prostych programów

Zagnieżdżanie instrukcji:



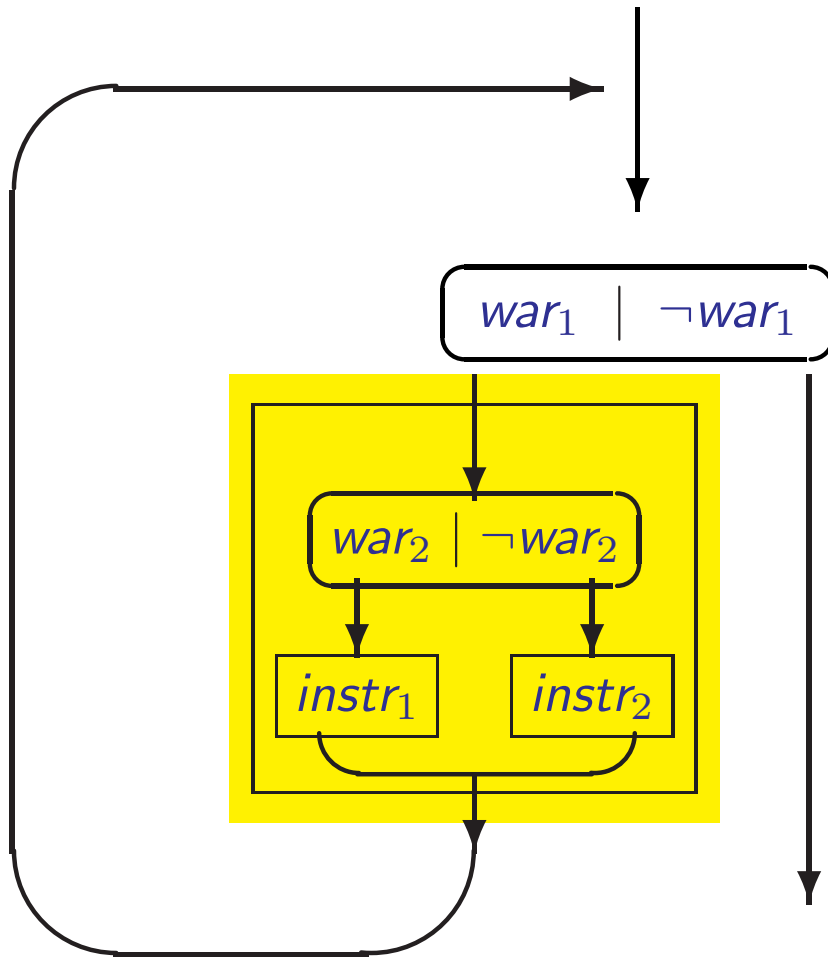
schemat

```
while (war1) {
```

C

Budowa prostych programów

Zagnieżdżanie instrukcji:



schemat

```
while ( $war_1$ ) {
  if ( $war_2$ ) {
     $instr_1$ 
  }
  else {
     $instr_2$ 
  }
}
```

C

Przykład programu: największy wspólny dzielnik

```
a=A; b=B;
```

```
while (a != b) {
```

```
}
```

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```
a=A; b=B;  
while (a != b) {  
    if (a > b)  a = a-b;  
    else      b = b-a;  
}
```

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```
a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}
```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48
180	48	84	48

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48
180	48	84	48
180	48	36	48

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48
180	48	84	48
180	48	36	48
180	48	36	12

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48
180	48	84	48
180	48	36	48
180	48	36	12
180	48	24	12

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48
180	48	84	48
180	48	36	48
180	48	36	12
180	48	24	12
180	48	12	12

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: największy wspólny dzielnik

```

a=A; b=B;
while (a != b) {
    if (a > b)  a = a-b;
    else      b = b-a;
}

```

<i>A</i>	<i>B</i>	<i>a</i>	<i>b</i>
180	48		
180	48	180	48
180	48	132	48
180	48	84	48
180	48	36	48
180	48	36	12
180	48	24	12
180	48	12	12

!= — oznaczenie relacji „nierówne” w języku C

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}\left(\text{nwd}\left(\text{nwd}(a_1, a_2), \dots\right), a_n\right)$$

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}\left(\text{nwd}\left(\text{nwd}(a_1, a_2), \dots\right), a_n\right)$$

```
wczytaj a; ile = 1;
while (ile < n){
    wczytaj b;
    ile = ile+1;
    while (a != b) {
        if (a > b) a = a-b;
        else b = b-a;
    }
}
```

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}\left(\text{nwd}\left(\text{nwd}(a_1, a_2), \dots\right), a_n\right)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
----------	----------	----------	------------	-------------

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}\left(\text{nwd}\left(\text{nwd}(a_1, a_2), \dots\right), a_n\right)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168
3	90	108	2	168

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168
3	90	108	2	168
3	18	18	2	168

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168
3	90	108	2	168
3	18	18	2	168
3	18	168	2	

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168
3	90	108	2	168
3	18	18	2	168
3	18	168	2	
3	18	168	3	

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168
3	90	108	2	168
3	18	18	2	168
3	18	168	2	
3	18	168	3	
3	6	6	3	

Przykład programu: NWD wielu liczb

$$\text{nwd}(a_1, a_2, a_3) = \text{nwd}(\text{nwd}(a_1, a_2), a_3)$$

$$\text{nwd}(a_1, a_2, \dots, a_n) = \text{nwd}(\text{nwd}(\text{nwd}(a_1, a_2), \dots), a_n)$$

```
wczytaj a; ile = 1;
```

```
while (ile < n){
```

```
    wczytaj b;
```

```
    ile = ile+1;
```

```
    while (a != b) {
```

```
        if (a > b) a = a-b;
```

```
        else b = b-a;
```

```
    }
```

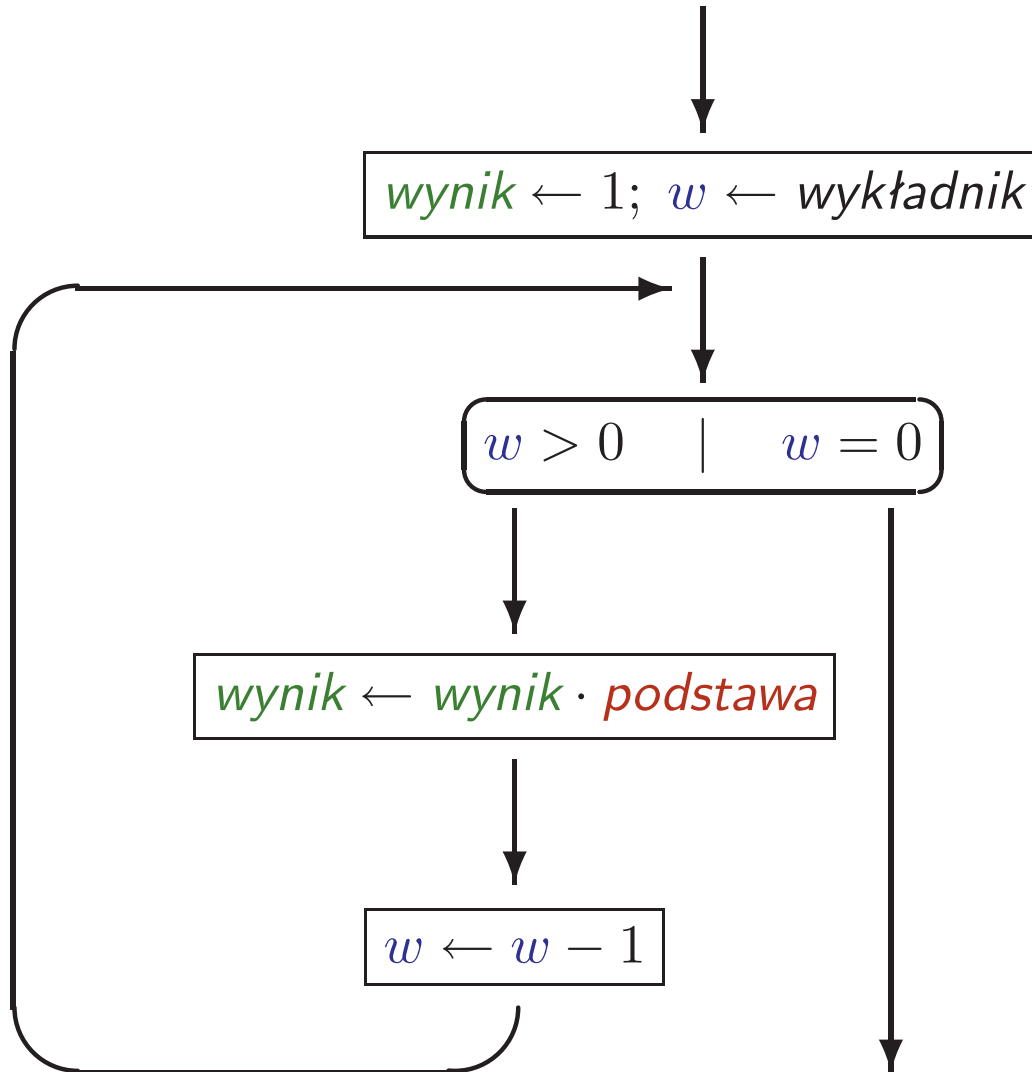
```
}
```

<i>n</i>	<i>a</i>	<i>b</i>	<i>ile</i>	niewczytane
3				90 108 168
3	90			108 168
3	90		1	108 168
3	90	108	1	168
3	90	108	2	168
3	18	18	2	168
3	18	168	2	
3	18	168	3	
3	6	6	3	

Poprawność algorytmów

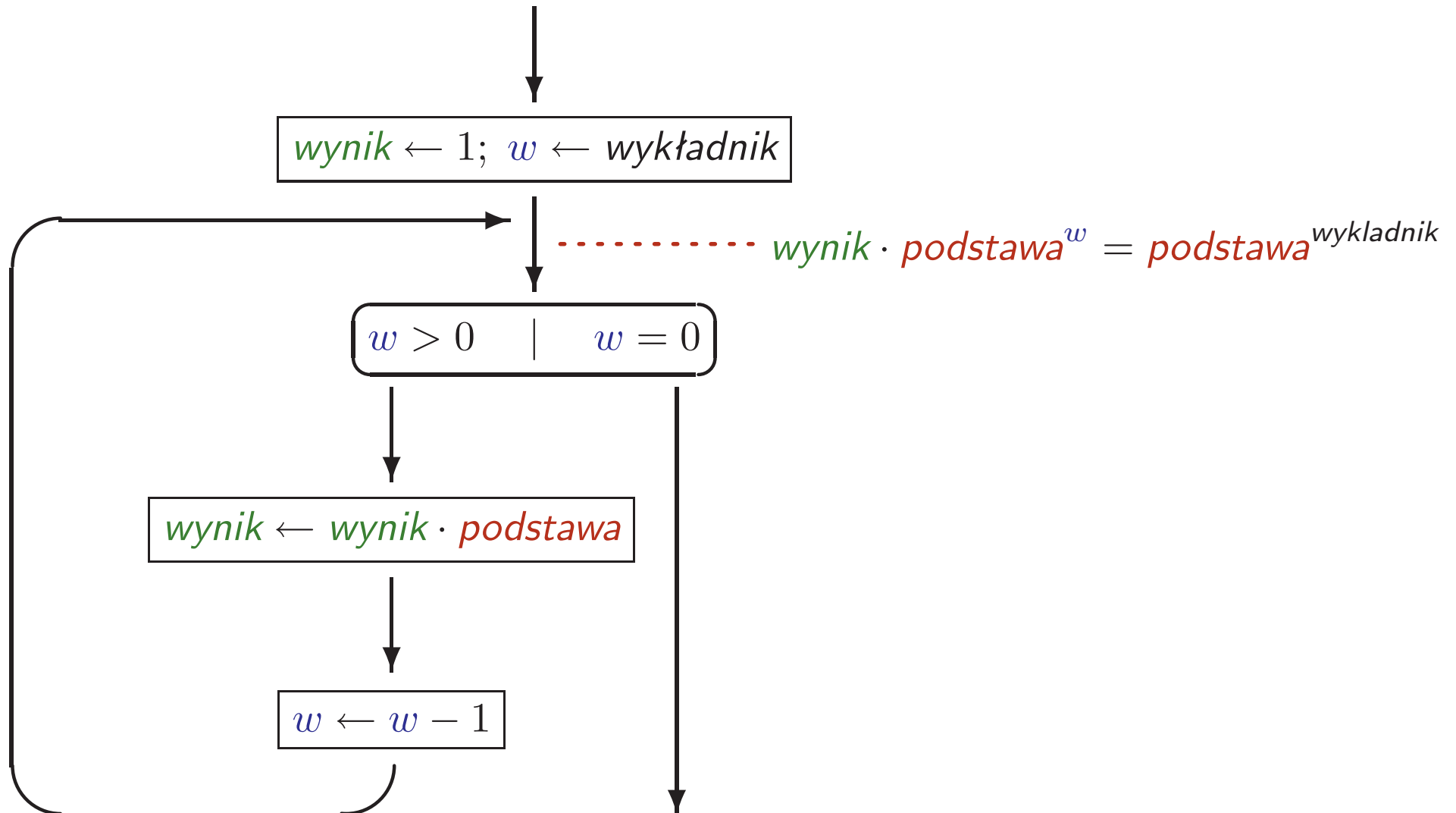
Poprawność algorytmów

Przykład: (potęgowanie przez wielokrotne mnożenie)



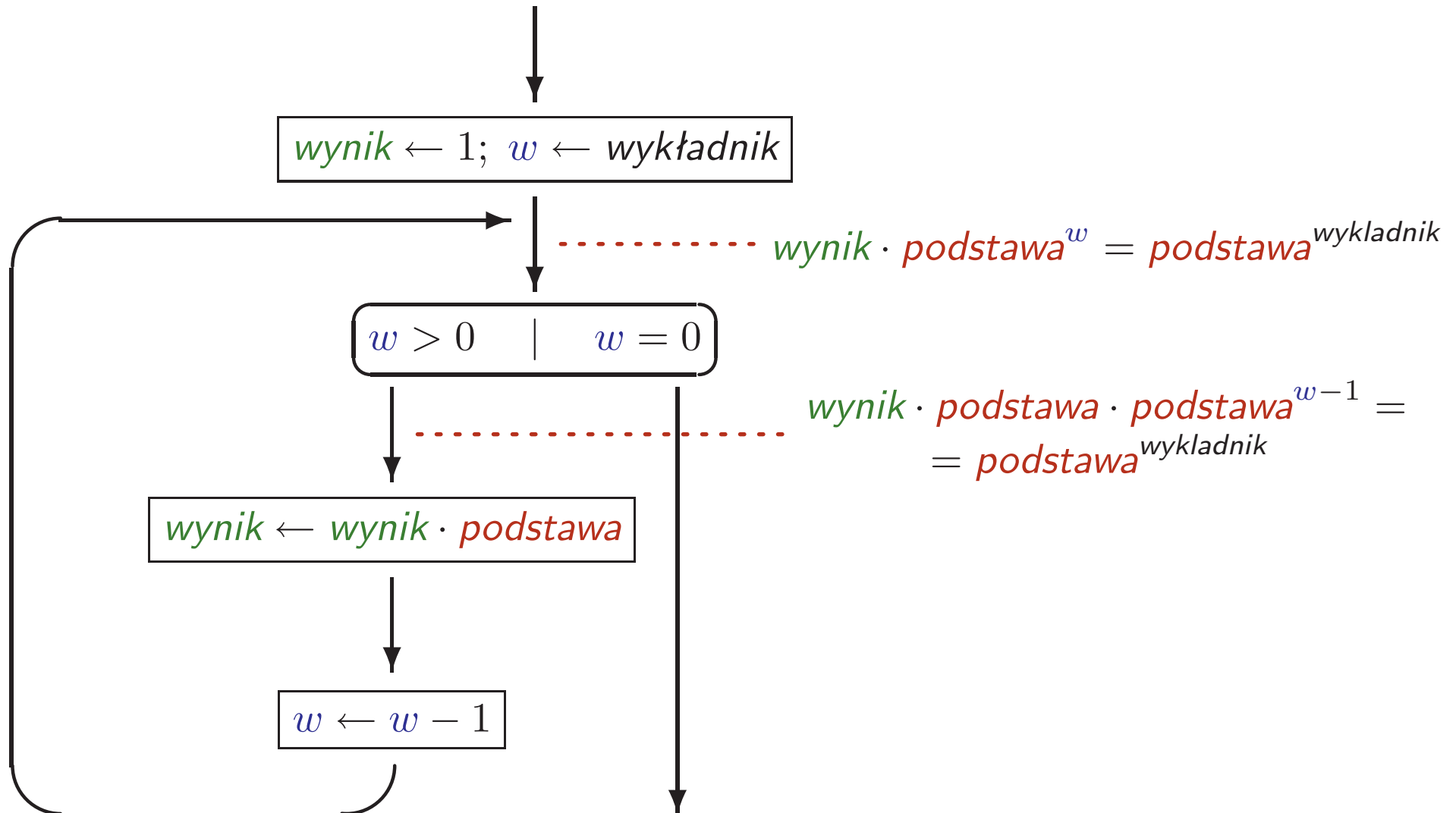
Poprawność algorytmów

Przykład: (potęgowanie przez wielokrotne mnożenie)



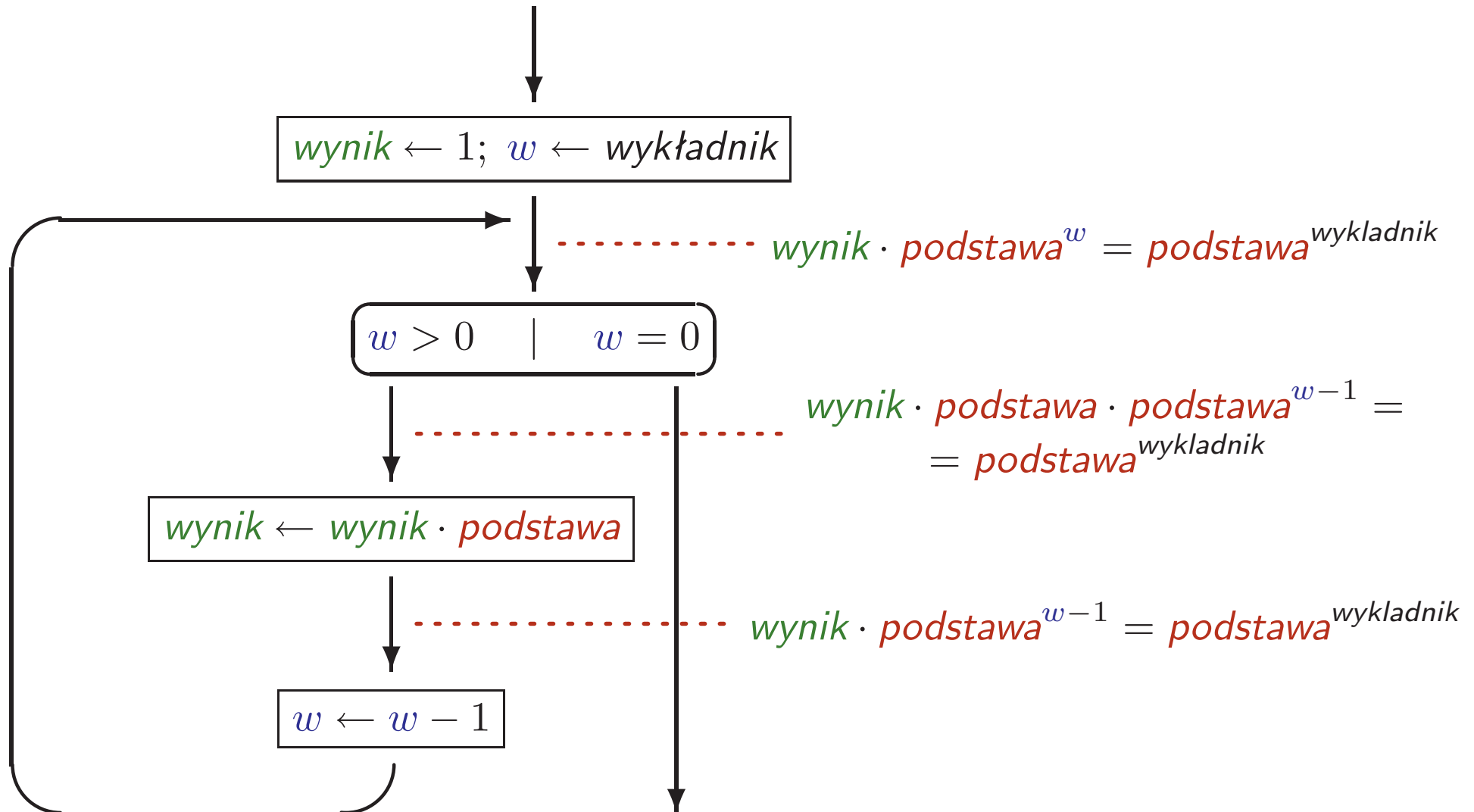
Poprawność algorytmów

Przykład: (potęgowanie przez wielokrotne mnożenie)



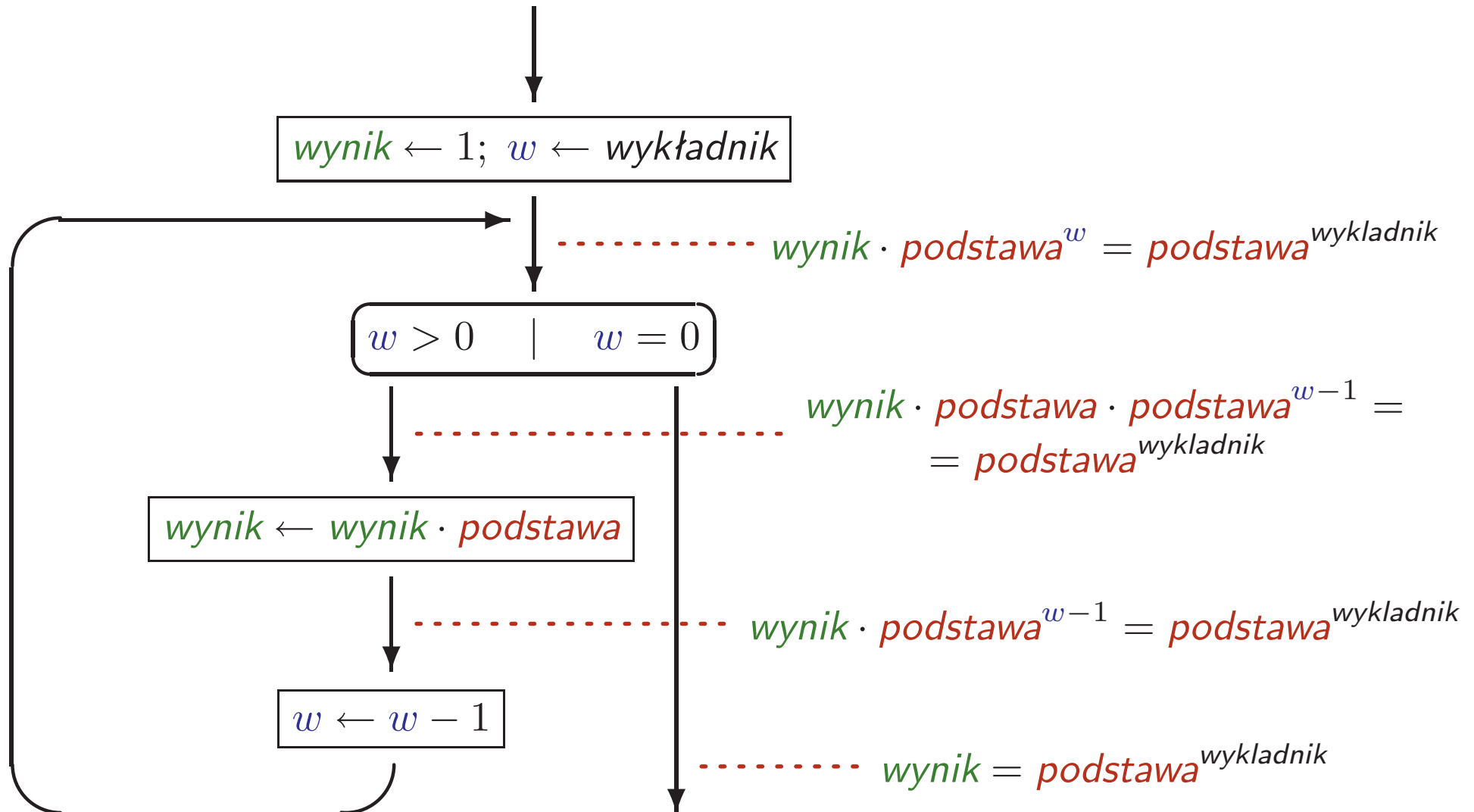
Poprawność algorytmów

Przykład: (potęgowanie przez wielokrotne mnożenie)



Poprawność algorytmów

Przykład: (potęgowanie przez wielokrotne mnożenie)



Metoda niezmienników

DEFINICJA:

Niezmiennik pętli `while (war) instr` — dowolna własność P zmiennych programu taka, że

Metoda niezmienników

DEFINICJA:

Niezmiennik pętli `while (war) instr` — dowolna własność P zmiennych programu taka, że

jeśli przed wykonaniem `instr` spełnione jest $P \ \& \ \text{war}$,
to po wykonaniu `instr` spełnione jest P

Metoda niezmienników

DEFINICJA:

Niezmiennik pętli `while (war) instr` — dowolna własność P zmiennych programu taka, że

jeśli przed wykonaniem `instr` spełnione jest $P \ \& \ \text{war}$,
to po wykonaniu `instr` spełnione jest P

TWIERDZENIE:

Jeśli P jest niezmiennikiem pętli `while (war) instr` i po inicjalizacji pętli ten niezmiennik jest spełniony, to na wyjściu z pętli jest spełnione $P \ \& \ \neg \text{war}$.

Niezmienniki spoza informatyki



Niezmienniki spoza informatyki



(wzięte z [http://pl.wikipedia.org/wiki/Meander_\(geografia\)](http://pl.wikipedia.org/wiki/Meander_(geografia))
oraz z http://pl.wikipedia.org/wiki/Kanał_wodny)

Niezmienniki spoza informatyki



(wzięte z [http://pl.wikipedia.org/wiki/Meander_\(geografia\)](http://pl.wikipedia.org/wiki/Meander_(geografia))
oraz z http://pl.wikipedia.org/wiki/Kanał_wodny)

Jaka będzie prędkość przepływu wody
po zamknięciu rzeki w regularnym korycie?

Niezmienniki spoza informatyki



Niezmienniki spoza informatyki



PRZEKRÓJ
KORYTA

przekrój koryta: 6 m^2

Niezmienniki spoza informatyki



PRZEKRÓJ
KORYTA

przekrój koryta: 6 m^2
średnia prędkość: 0.75 m/s

Niezmienniki spoza informatyki



PRZEKRÓJ
KORYTA

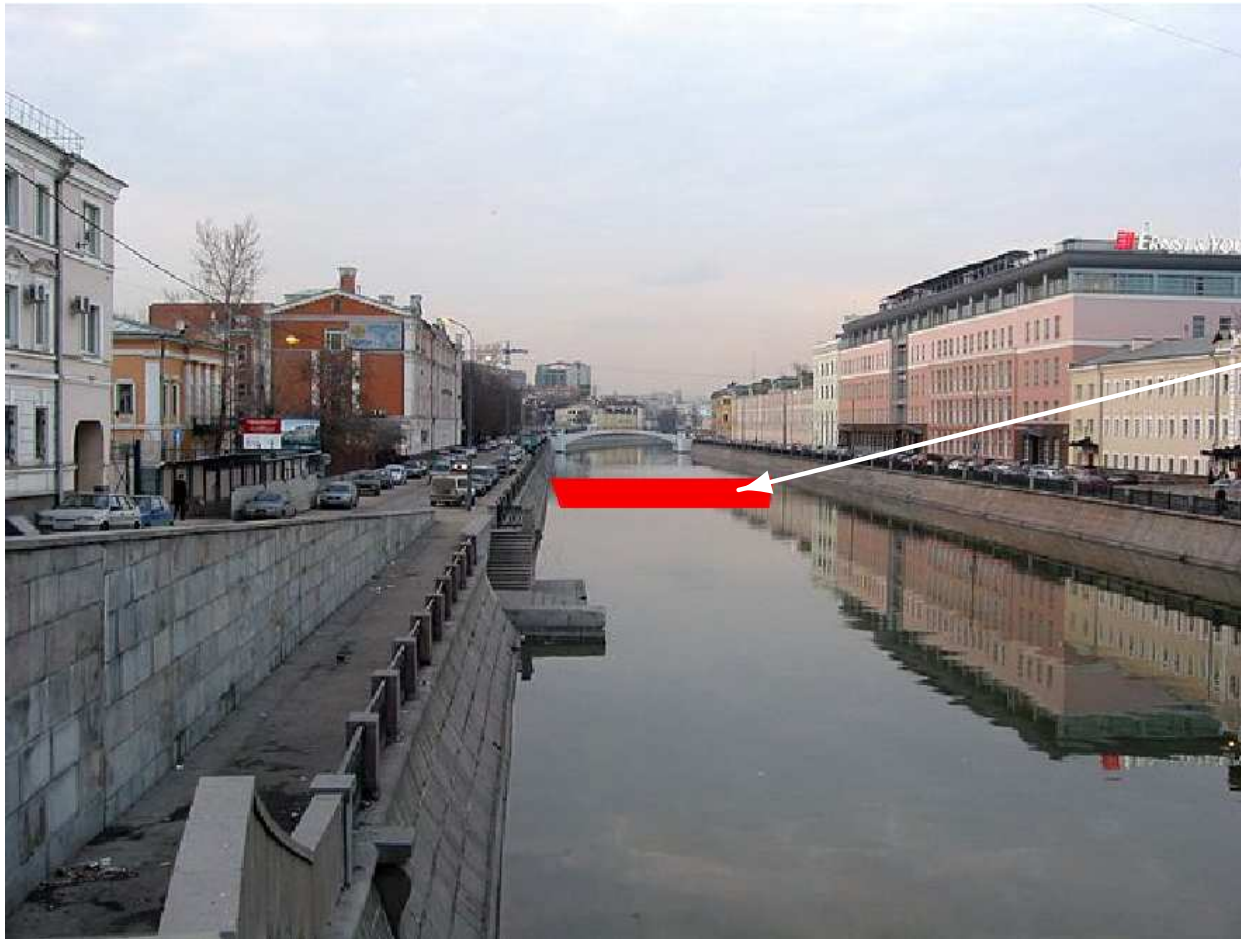
$$\begin{array}{l} \text{przekrój koryta:} \quad 6 \, m^2 \\ \text{średnia prędkość:} \quad 0.75 \, m/s \\ \text{PRZEPŁYW:} \quad \frac{4.5 \, m^3/s}{} \end{array}$$

Niezmienniki spoza informatyki



PRZEPŁYW: $\overline{4.5 \text{ m}^3/\text{s}}$

Niezmienniki spoza informatyki

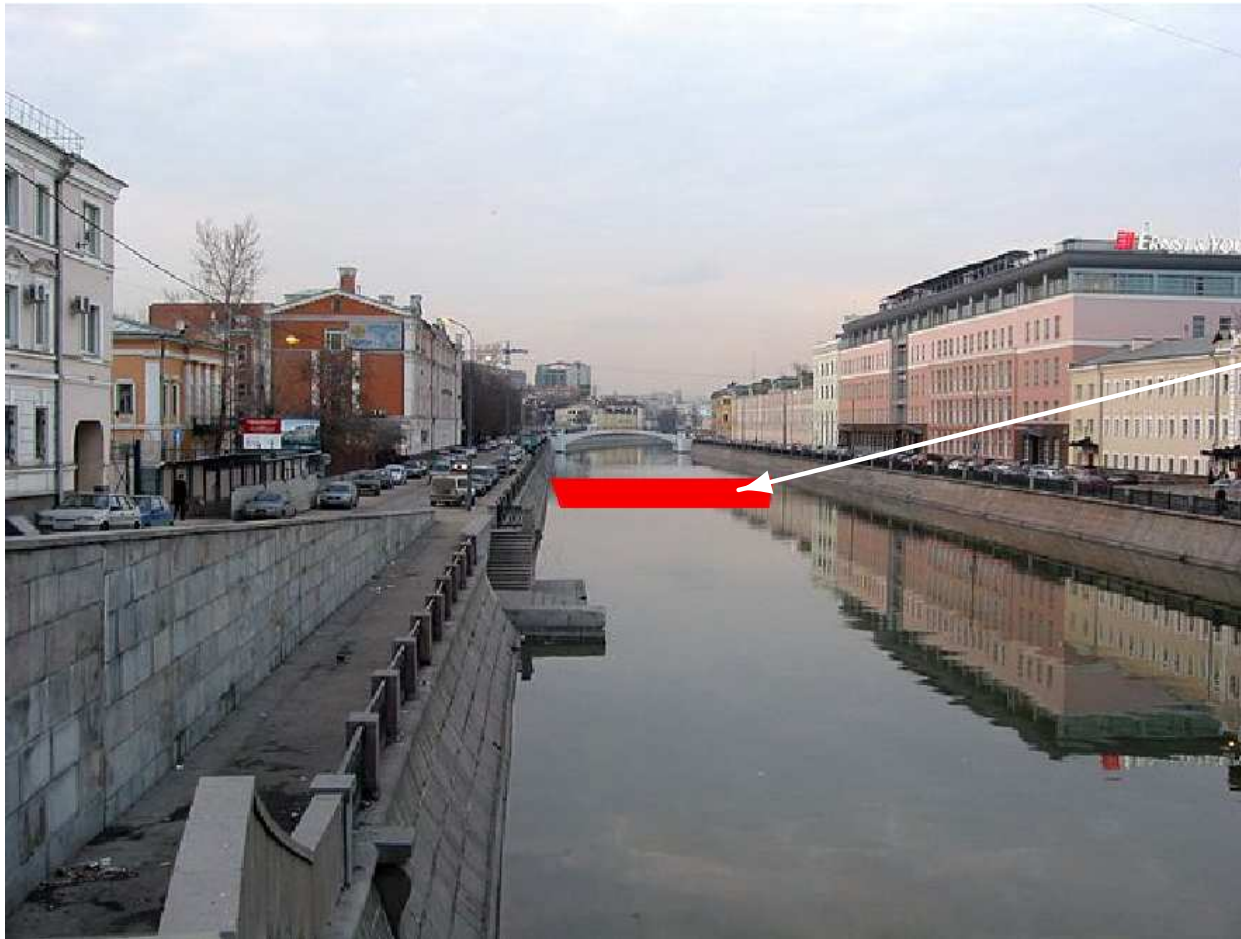


PRZEKRÓJ
KORYTA

przekrój koryta: 3 m^2

PRZEPŁYW: $\overline{4.5 \text{ m}^3/\text{s}}$

Niezmienniki spoza informatyki



PRZEKRÓJ
KORYTA

przekrój koryta: 3 m^2
średnia prędkość: 1.5 m/s
PRZEPŁYW: $4.5 \text{ m}^3/\text{s}$

Niezmienniki spoza informatyki

RZEKA DZIKA		UREGULOWANY KANAL	
przekrój koryta:	6 m^2	przekrój koryta:	3 m^2
średnia prędkość:	0.75 m/s	średnia prędkość:	1.5 m/s
PRZEPŁYW:	$4.5 \text{ m}^3/\text{s}$	PRZEPŁYW:	$4.5 \text{ m}^3/\text{s}$

Niezmienniki spoza informatyki

RZEKA DZIKA	UREGULOWANY KANAL
przekrój koryta: 6 m^2	przekrój koryta: 3 m^2
średnia prędkość: 0.75 m/s	średnia prędkość: 1.5 m/s
PRZEPŁYW: $4.5 \text{ m}^3/\text{s}$	PRZEPŁYW: $4.5 \text{ m}^3/\text{s}$

Zmienia się zarówno przekrój koryta jak prędkość prądu.

Niezmienniki spoza informatyki

RZEKA DZIKA		UREGULOWANY KANAL	
przekrój koryta:	6 m^2	przekrój koryta:	3 m^2
średnia prędkość:	0.75 m/s	średnia prędkość:	1.5 m/s
PRZEPŁYW:	$4.5 \text{ m}^3/\text{s}$	PRZEPŁYW:	$4.5 \text{ m}^3/\text{s}$

Zmienia się zarówno przekrój koryta jak prędkość prądu.
Ale zmieniają się w taki sposób, że ich **iloczyn** pozostaje stały.

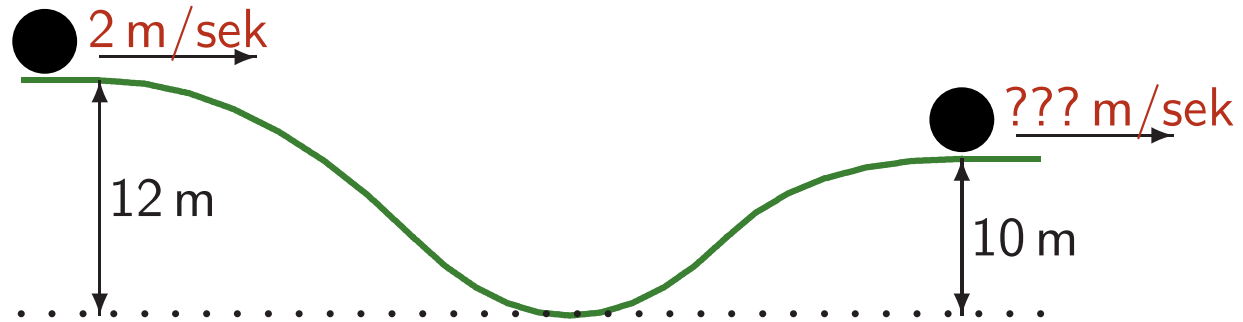
Niezmienniki spoza informatyki

RZEKA DZIKA		UREGULOWANY KANAL	
przekrój koryta:	6 m^2	przekrój koryta:	3 m^2
średnia prędkość:	0.75 m/s	średnia prędkość:	1.5 m/s
PRZEPŁYW:	$4.5 \text{ m}^3/\text{s}$	PRZEPŁYW:	$4.5 \text{ m}^3/\text{s}$

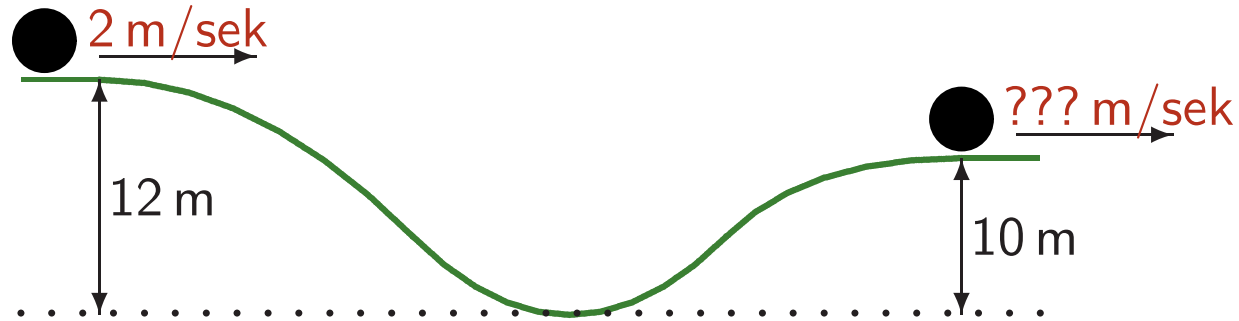
Zmienia się zarówno przekrój koryta jak prędkość prądu.
Ale zmieniają się w taki sposób, że ich **iloczyn** pozostaje stały.

Ten iloczyn jest **niezmiennikiem** — jest stały wzdłuż rzeki (jeśli nie ma dopływów), niezależnie od zmieniającego się kształtu koryta.

Niezmienniki spoza informatyki

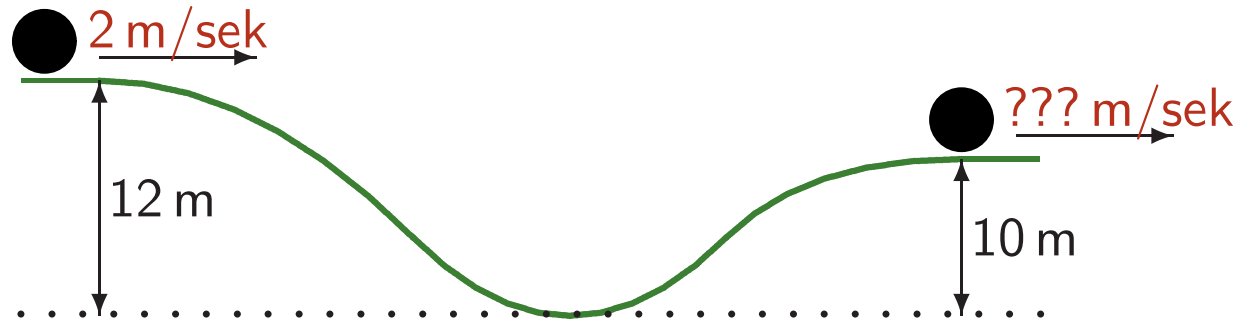


Niezmienniki spoza informatyki



energia = energia kinetyczna + energia potencjalna = $\frac{mv^2}{2} + mgh$
— jest stała.

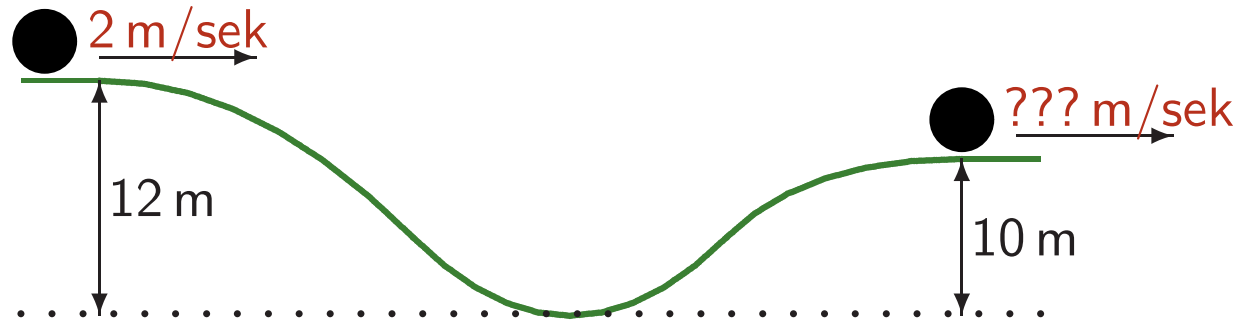
Niezmienniki spoza informatyki



energia = energia kinetyczna + energia potencjalna = $\frac{mv^2}{2} + mgh$
 — jest stała.

$$\frac{(2 \text{ m/sec})^2}{2} + 10 \text{ m/sek}^2 \cdot 12 \text{ m} = \frac{v^2}{2} + 10 \text{ m/sek}^2 \cdot 10 \text{ m}$$

Niezmienniki spoza informatyki

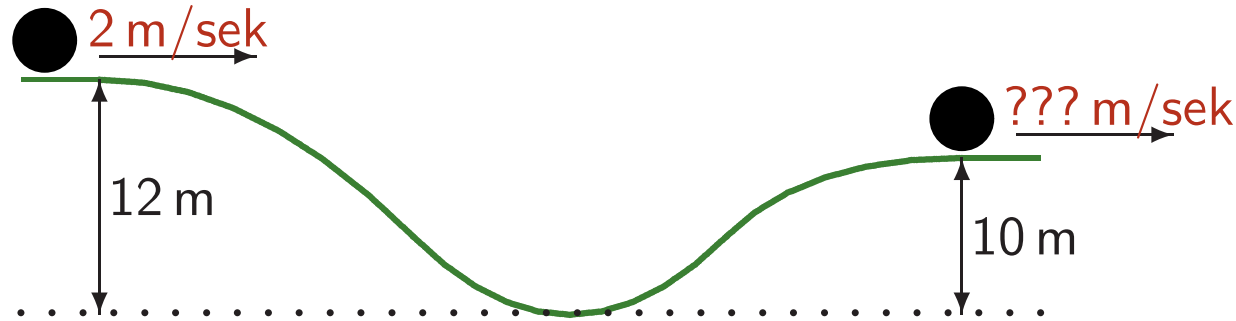


energia = energia kinetyczna + energia potencjalna = $\frac{mv^2}{2} + mgh$
 — jest stała.

$$\frac{(2 \text{ m/sec})^2}{2} + 10 \text{ m/sek}^2 \cdot 12 \text{ m} = \frac{v^2}{2} + 10 \text{ m/sek}^2 \cdot 10 \text{ m}$$

$$v = \sqrt{44} \text{ m/sek}$$

Niezmienniki spoza informatyki



energia = energia kinetyczna + energia potencjalna = $\frac{mv^2}{2} + mgh$
 — jest stała.

$$\frac{(2 \text{ m/sec})^2}{2} + 10 \text{ m/sek}^2 \cdot 12 \text{ m} = \frac{v^2}{2} + 10 \text{ m/sek}^2 \cdot 10 \text{ m}$$

$$v = \sqrt{44} \text{ m/sek} \approx 6.63 \text{ m/sek}$$

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają

Metoda niezmienników

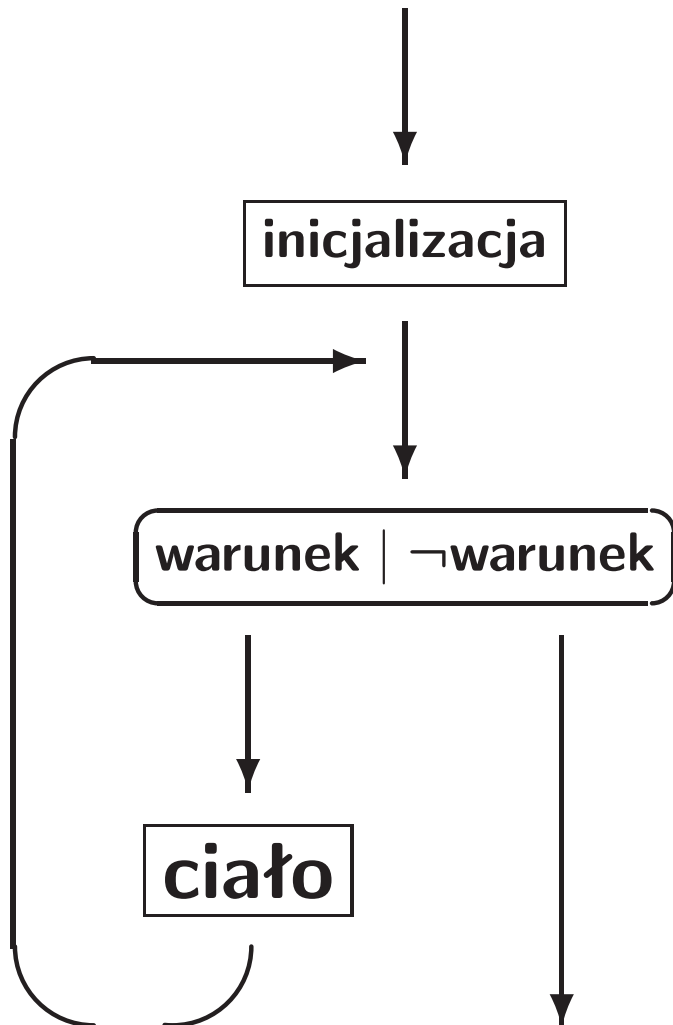
W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**.

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.

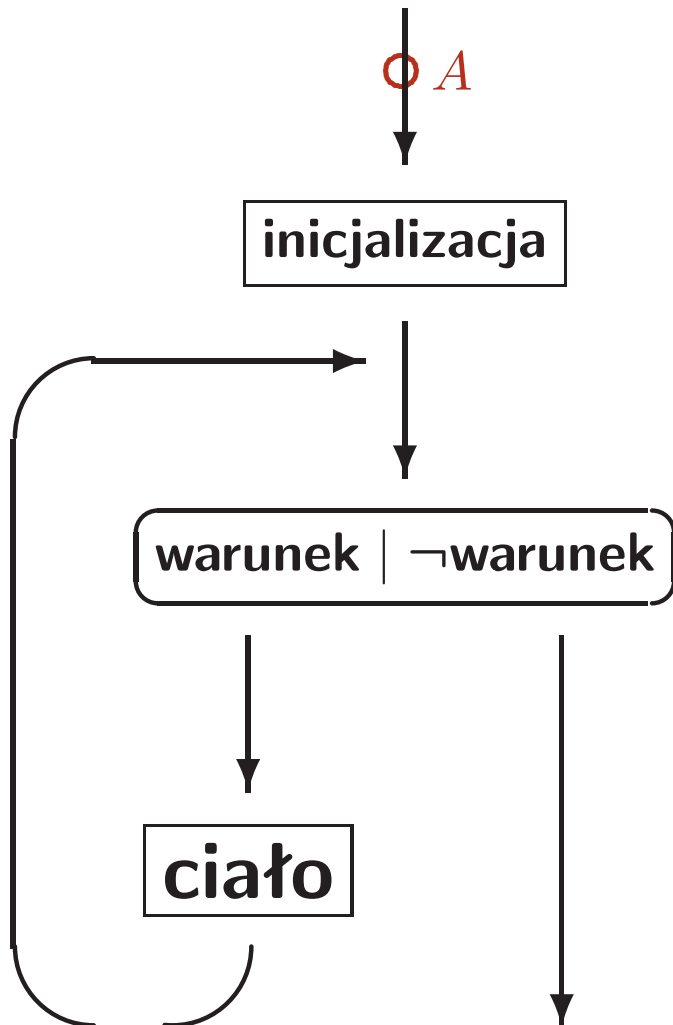
Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



Metoda niezmienników

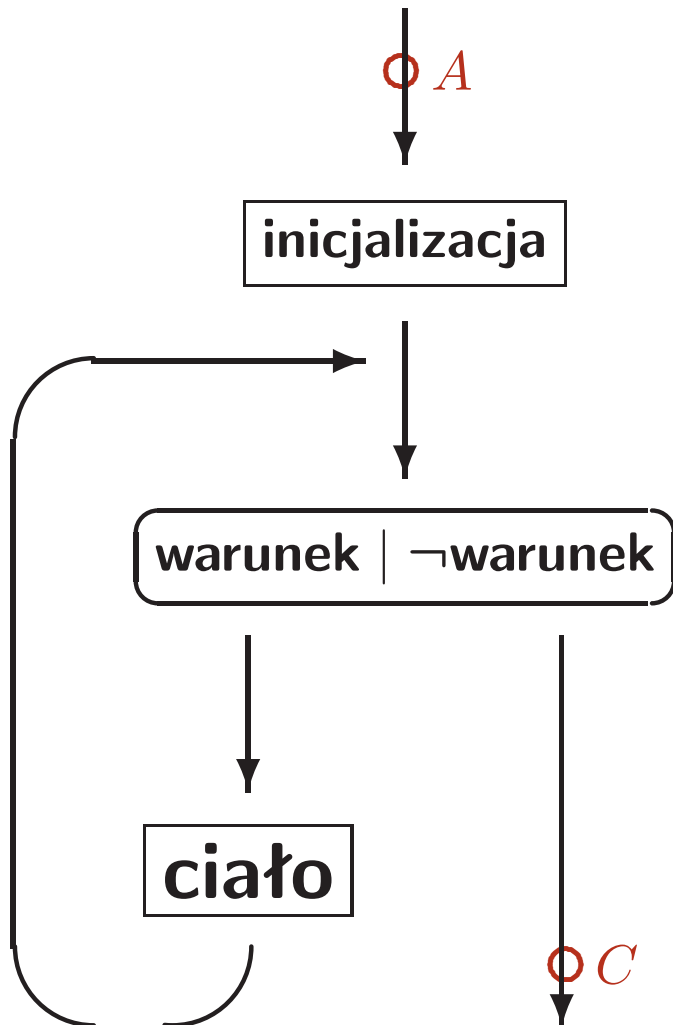
W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



A — założenia wstępne

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.

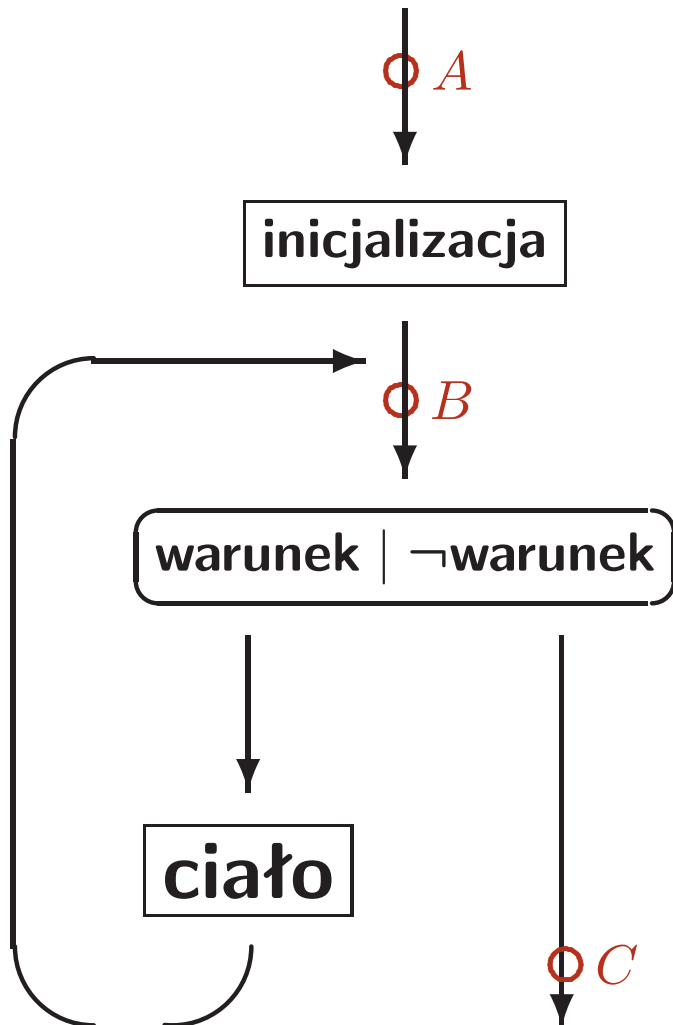


A — założenia wstępne

C — stan wynikowy

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



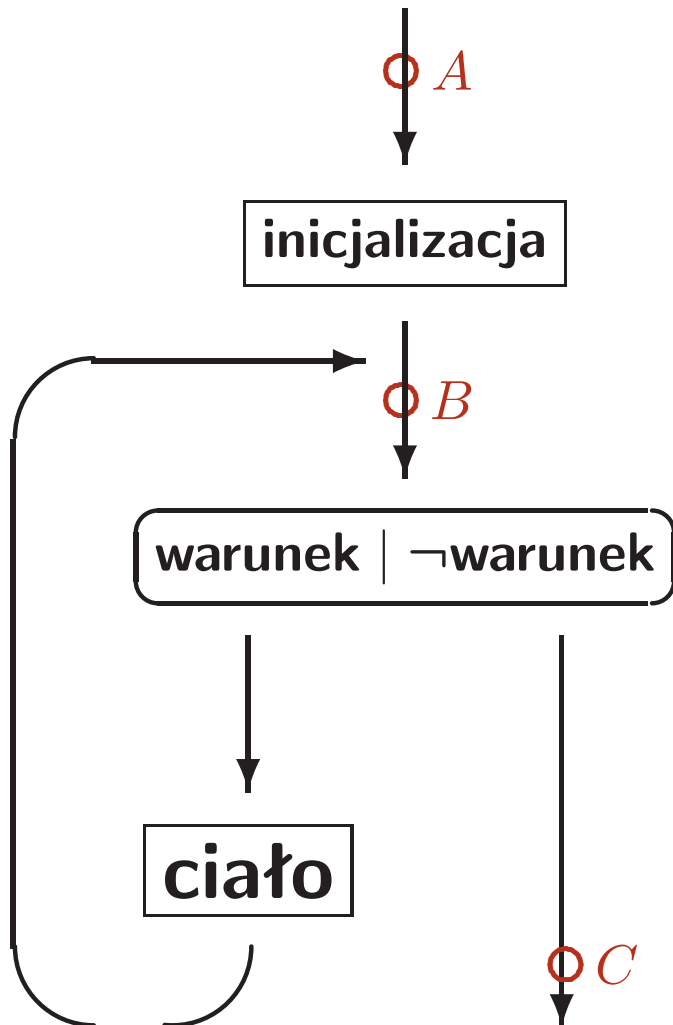
A — założenia wstępne

B — niezmiennik

C — stan wynikowy

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



A — założenia wstępne

B — niezmiennik

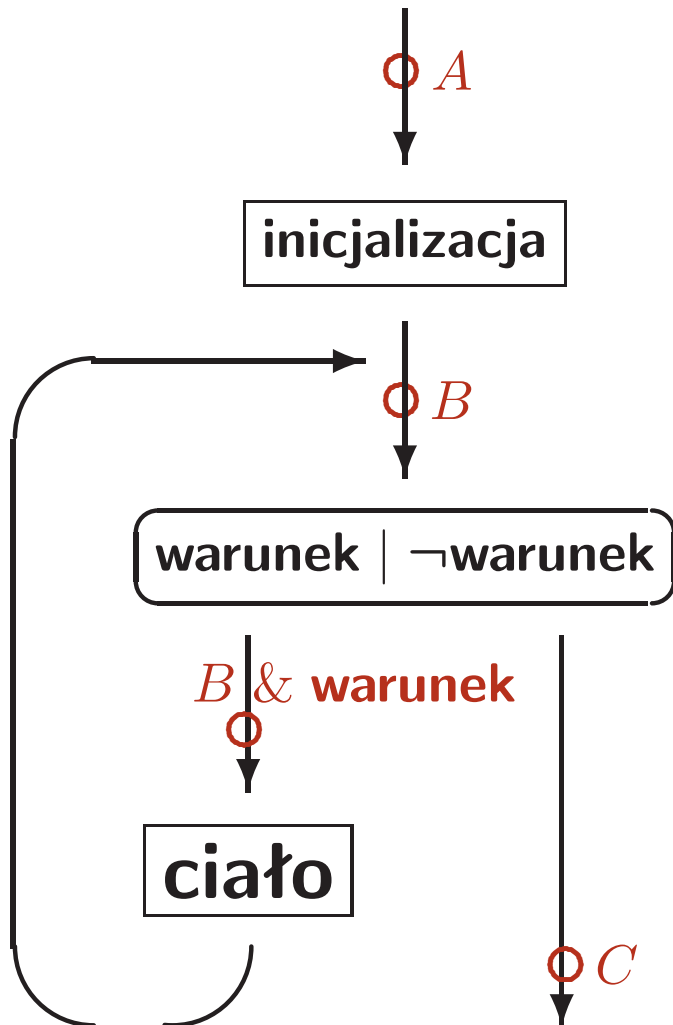
C — stan wynikowy

Trzeba zbadać:

1. przejście przez inicjalizację z A do B

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



A — założenia wstępne

B — niezmiennik

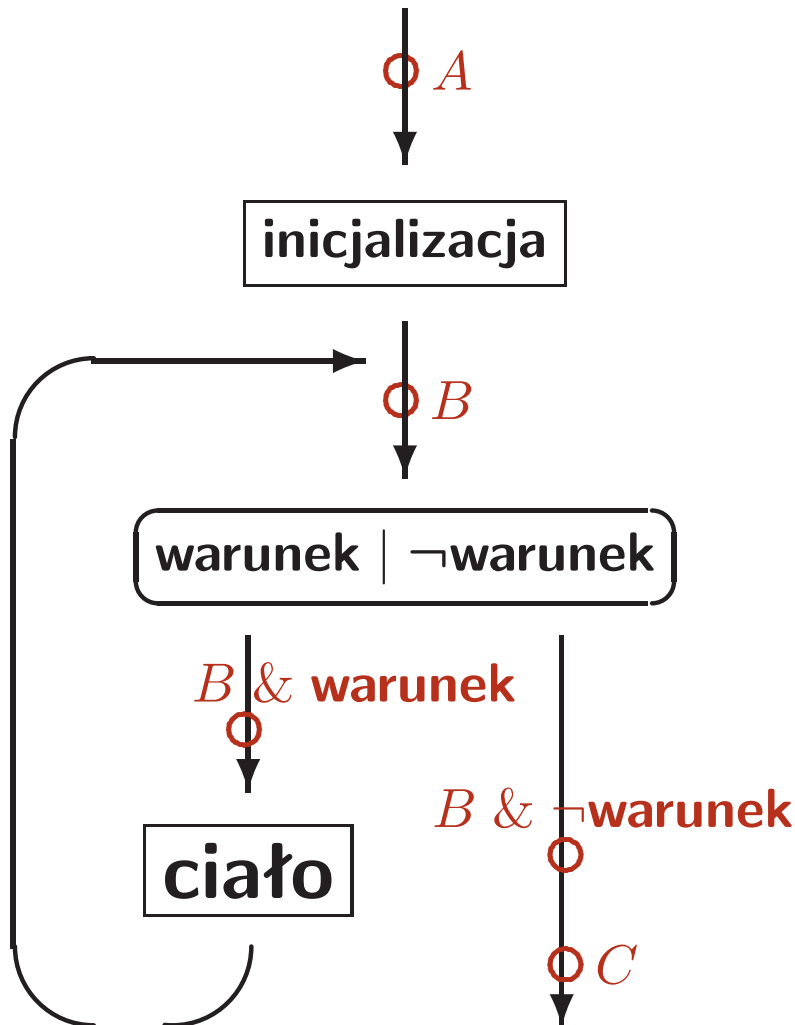
C — stan wynikowy

Trzeba zbadać:

1. przejście przez inicjalizację z A do B ,
2. przejście przez ciało z $B \& \text{warunek}$ do B

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



A — założenia wstępne

B — niezmiennik

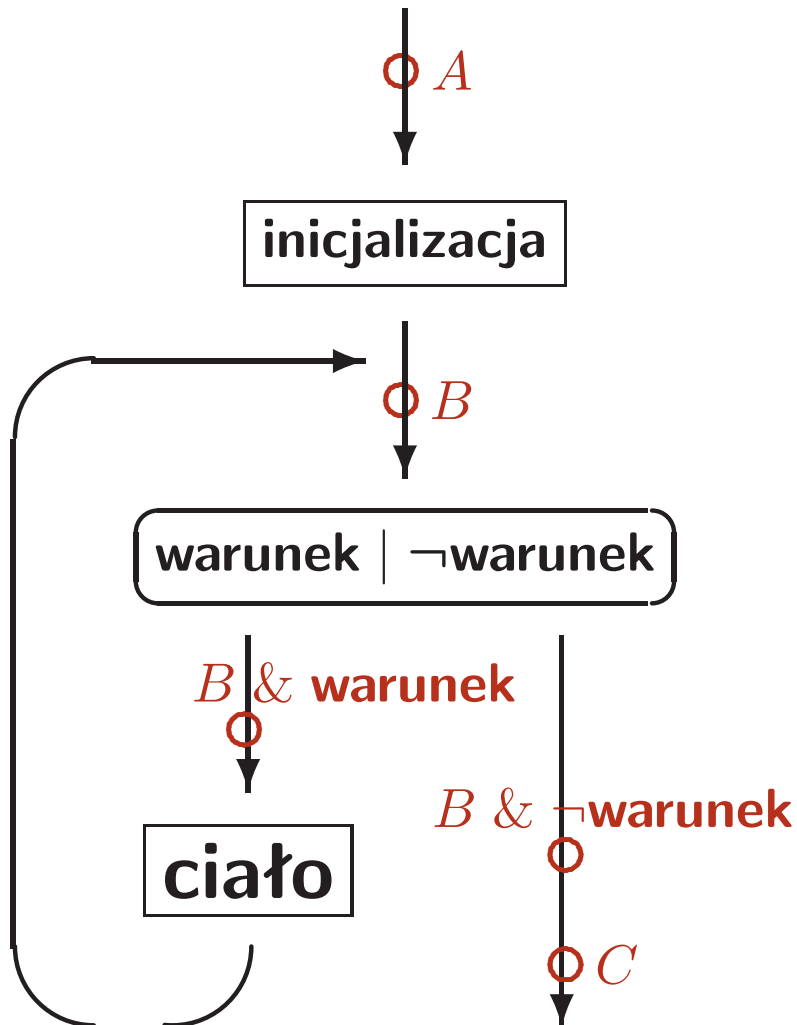
C — stan wynikowy

Trzeba zbadać:

1. przejście przez inicjalizację z A do B ,
2. przejście przez ciało z B & **warunek** do B ,
3. implikację B & \neg warunek $\Rightarrow C$.

Metoda niezmienników

W trakcie obliczeń pętli wartości zmiennych stale się zmieniają, ale pozostaje w mocy pewien związek między nimi — **niezmiennik**. Z tego faktu wynikają własności pętli.



A — założenia wstępne

B — niezmiennik

C — stan wynikowy

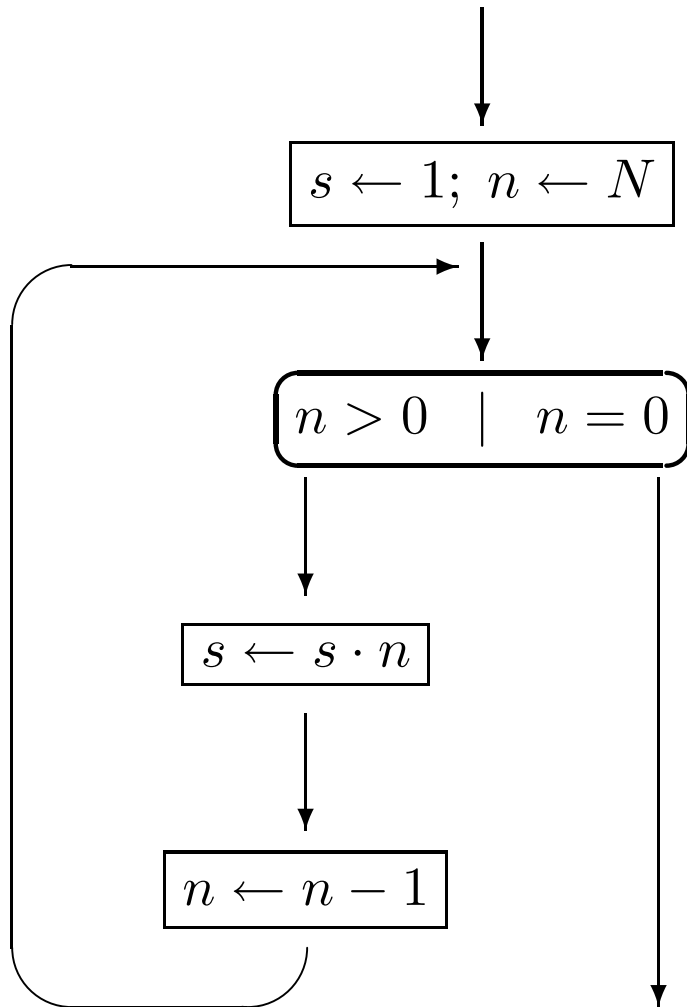
Trzeba zbadać:

1. przejście przez inicjalizację z A do B ,
2. przejście przez ciało z B & **warunek** do B ,
3. implikację B & \neg warunek $\Rightarrow C$.

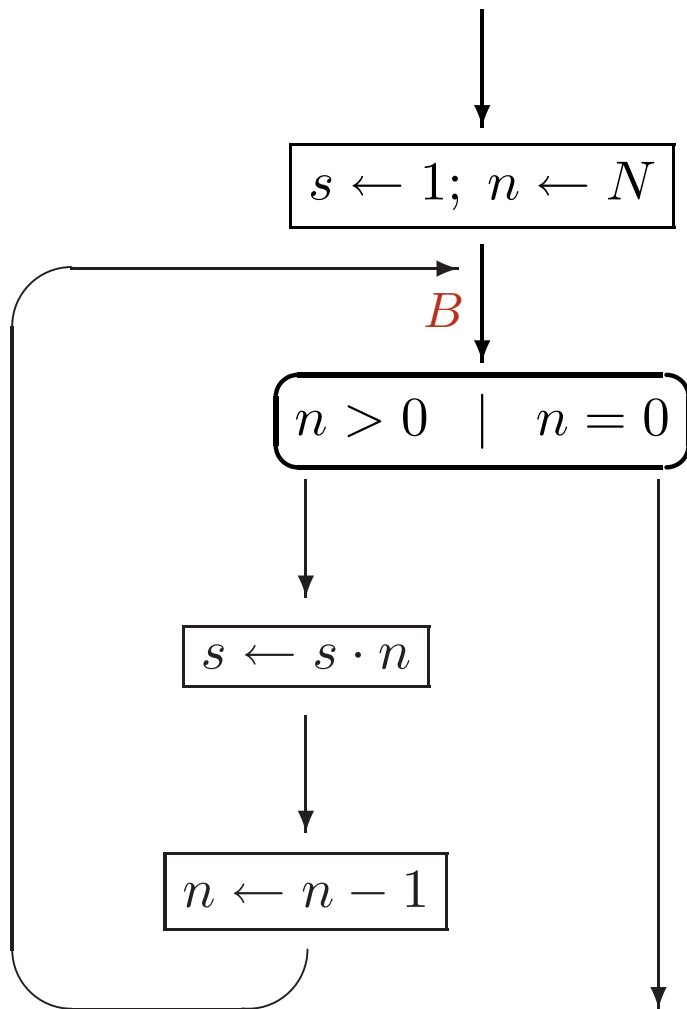
Wtedy wiemy, że

**dla każdych danych spełniających A
wyniki spełniają C**

Co liczy dany program?

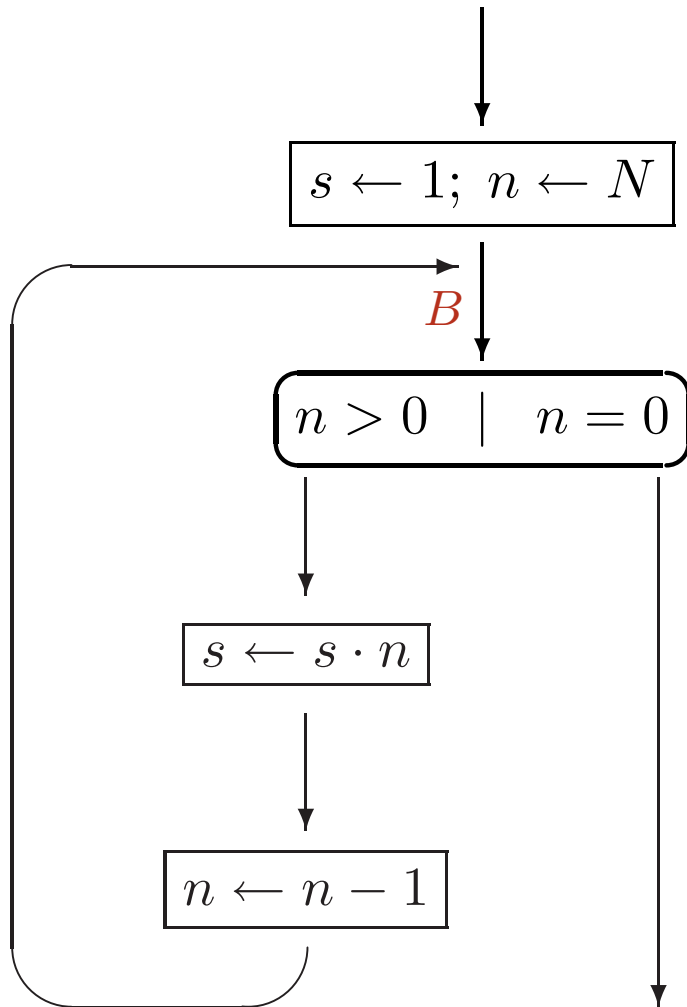


Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

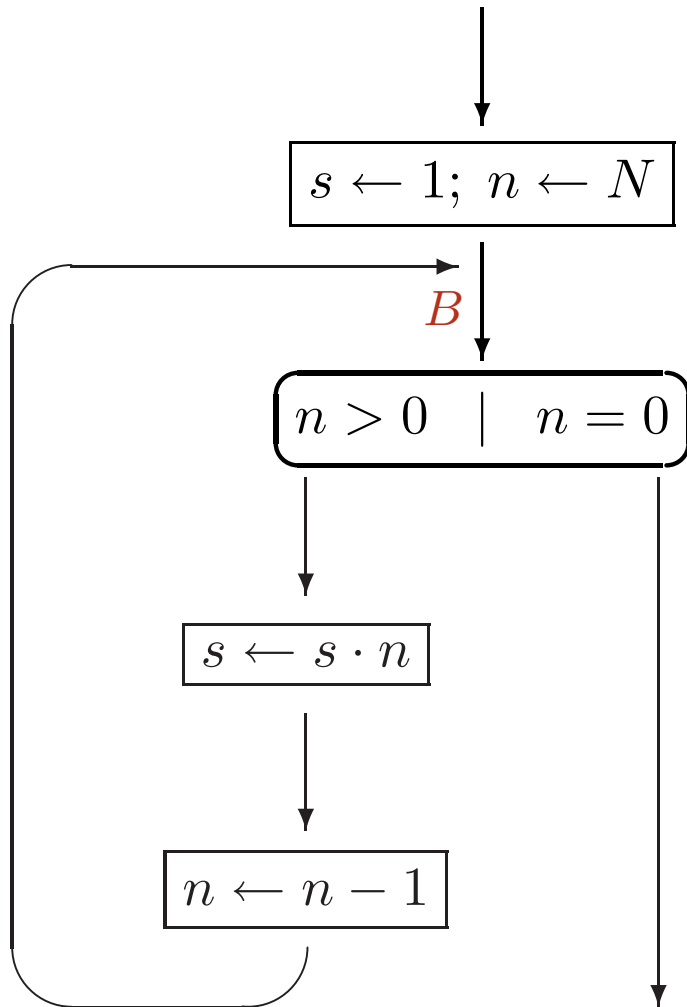
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

N	s	n
5		

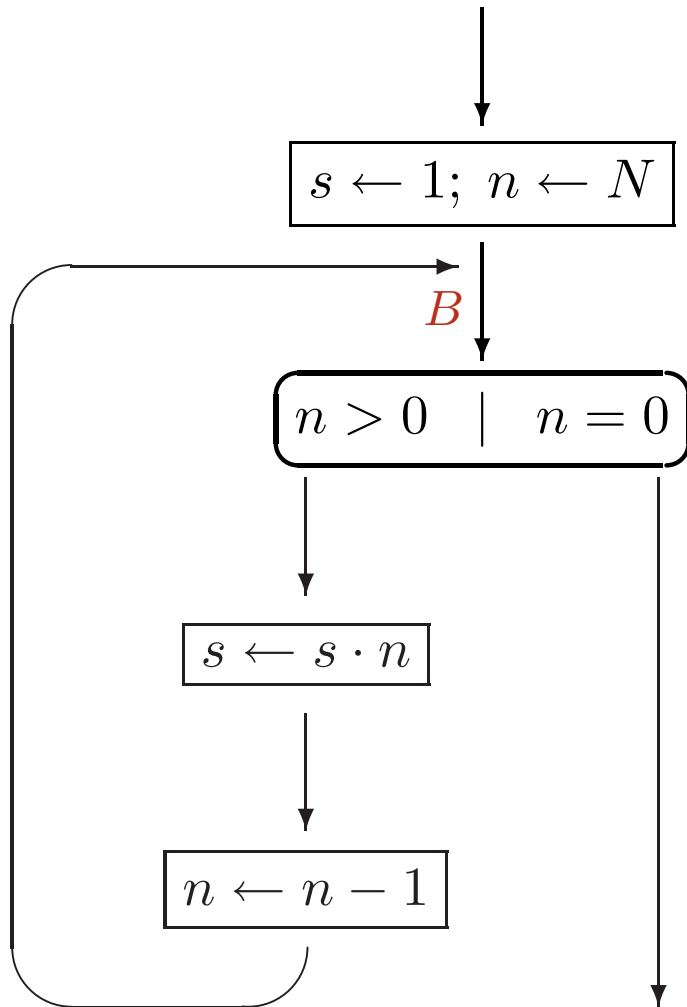
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

N	s	n
5	1	5

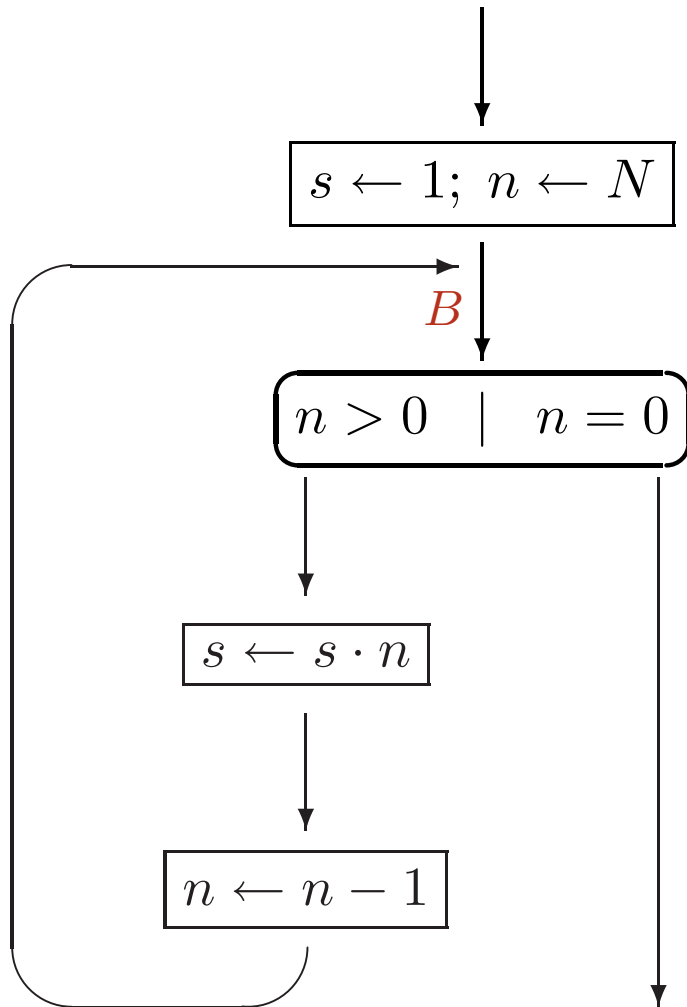
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

N	s	n
5	1	5
	5	4

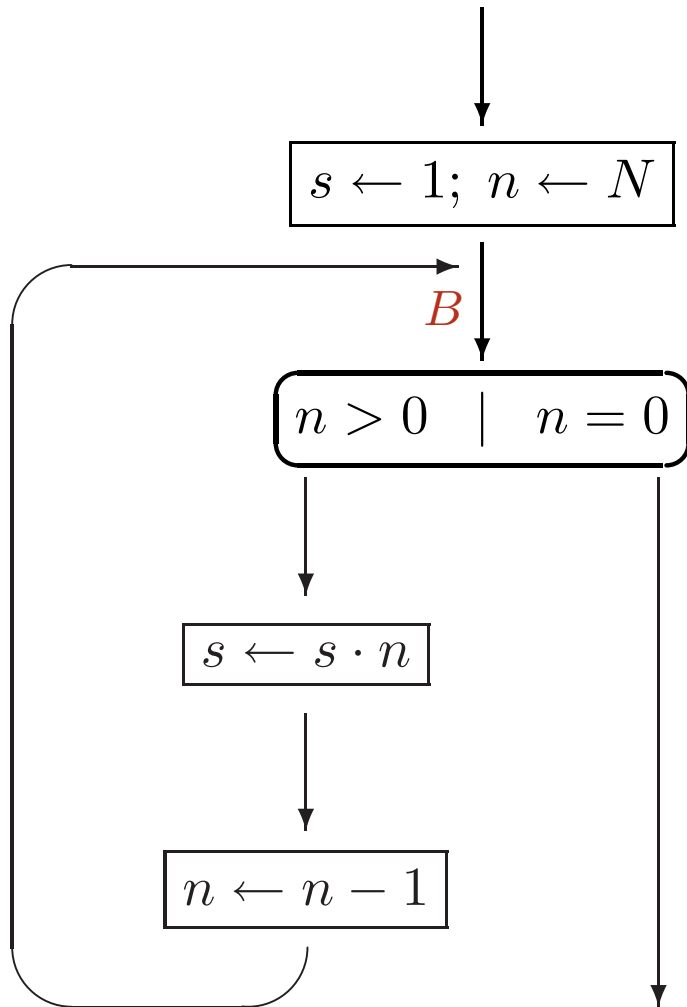
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

N	s	n
5	1	5
	5	4
	20	3

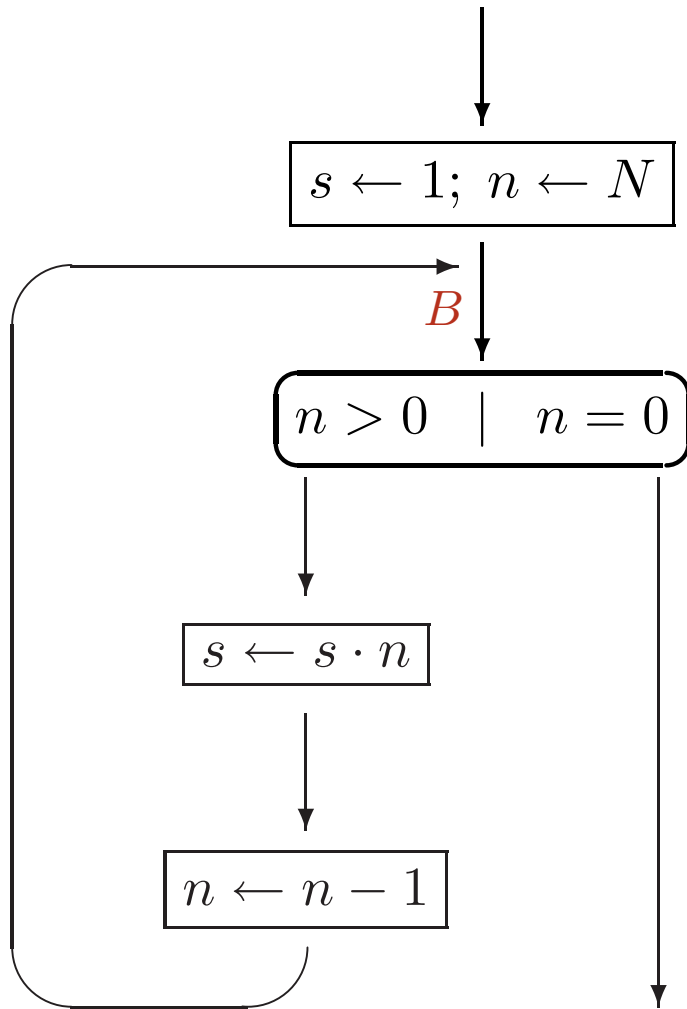
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>N</i>	<i>s</i>	<i>n</i>
5	1	5
	5	4
	20	3
	60	2

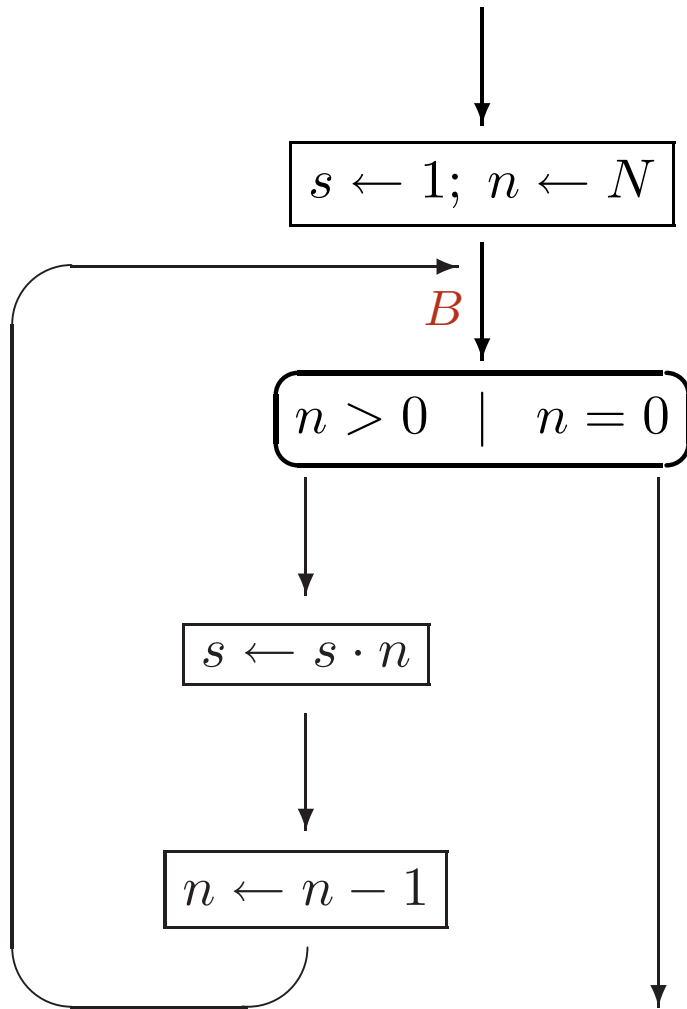
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

N	s	n
5	1	5
	5	4
	20	3
	60	2
	120	1

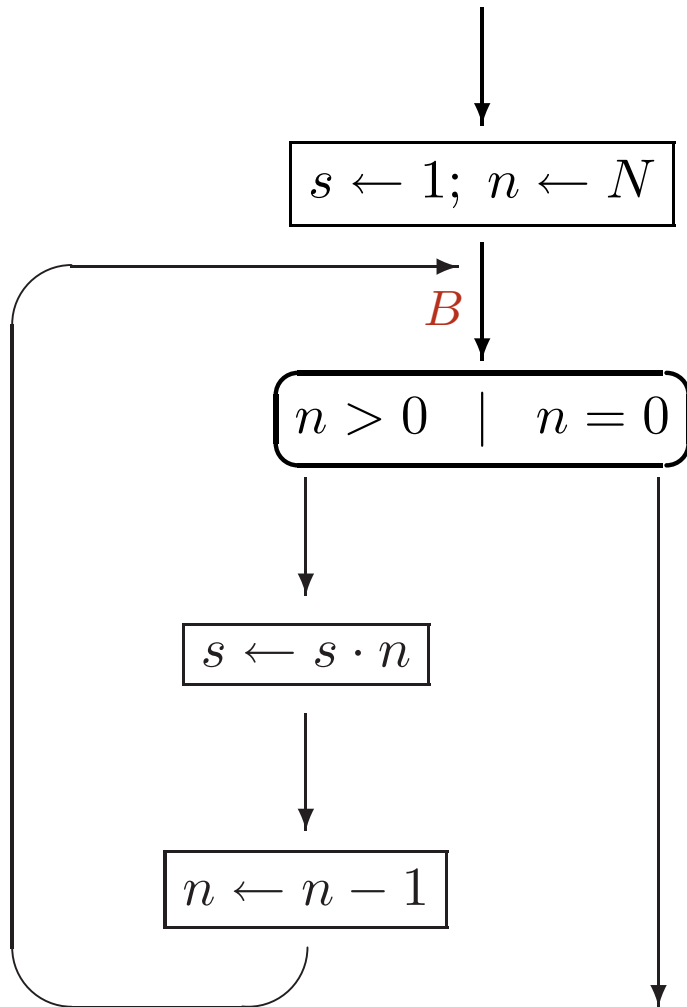
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

N	s	n
5	1	5
	5	4
	20	3
	60	2
	120	1
	120	0

Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

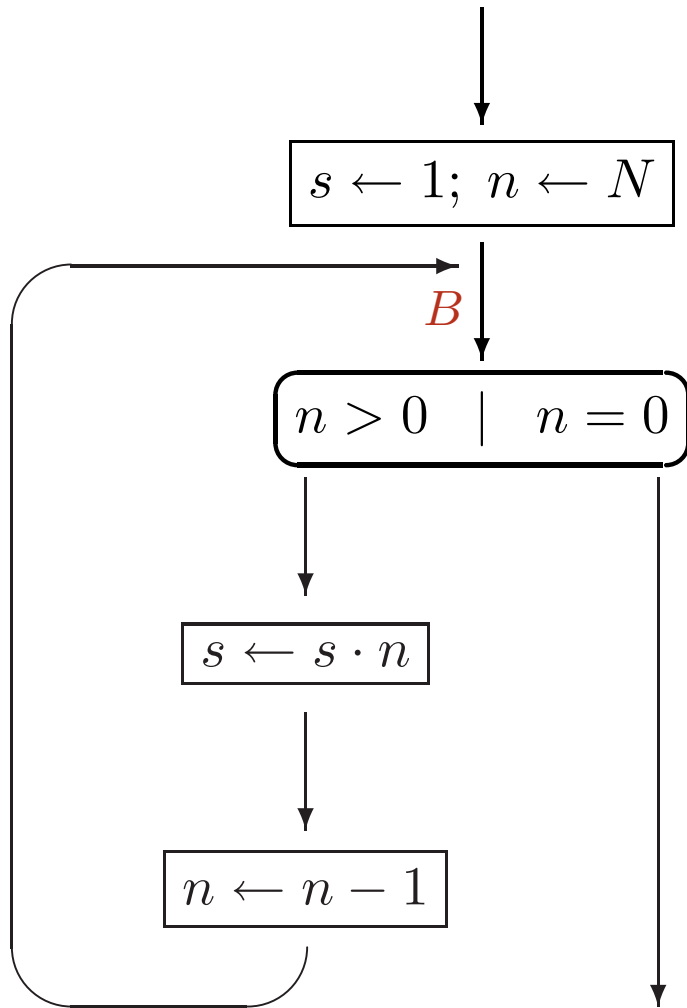
<i>N</i>	<i>s</i>	<i>n</i>
5	1	5
	5	4
	20	3
	60	2
	120	1
	120	0

Zależności między wartościami zmiennych:

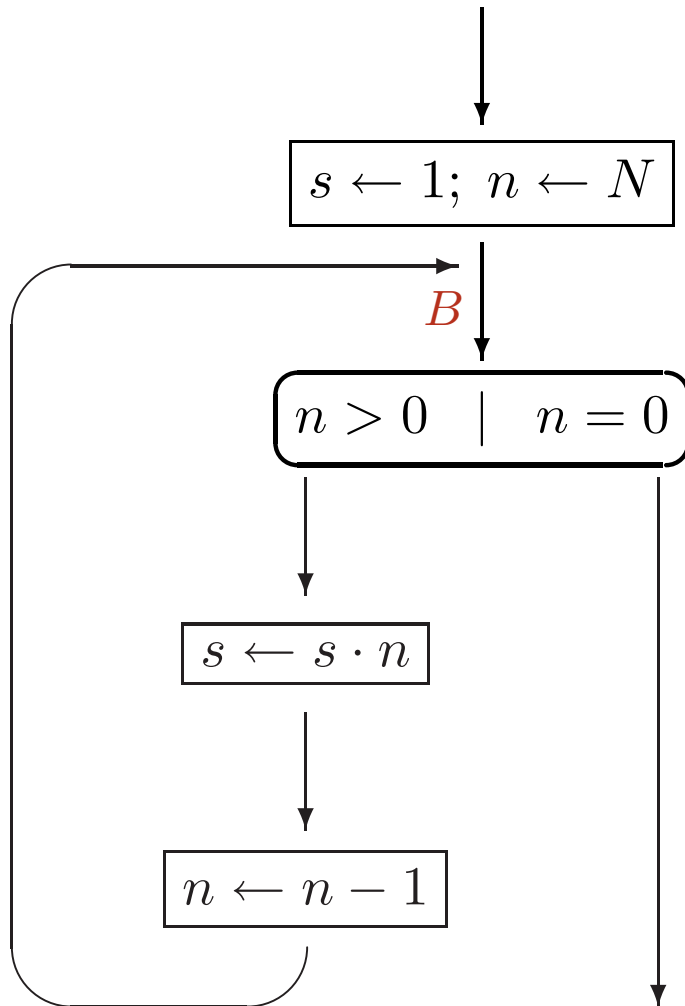
$$s \cdot n! = N!$$

Co liczy dany program?

Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?



Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

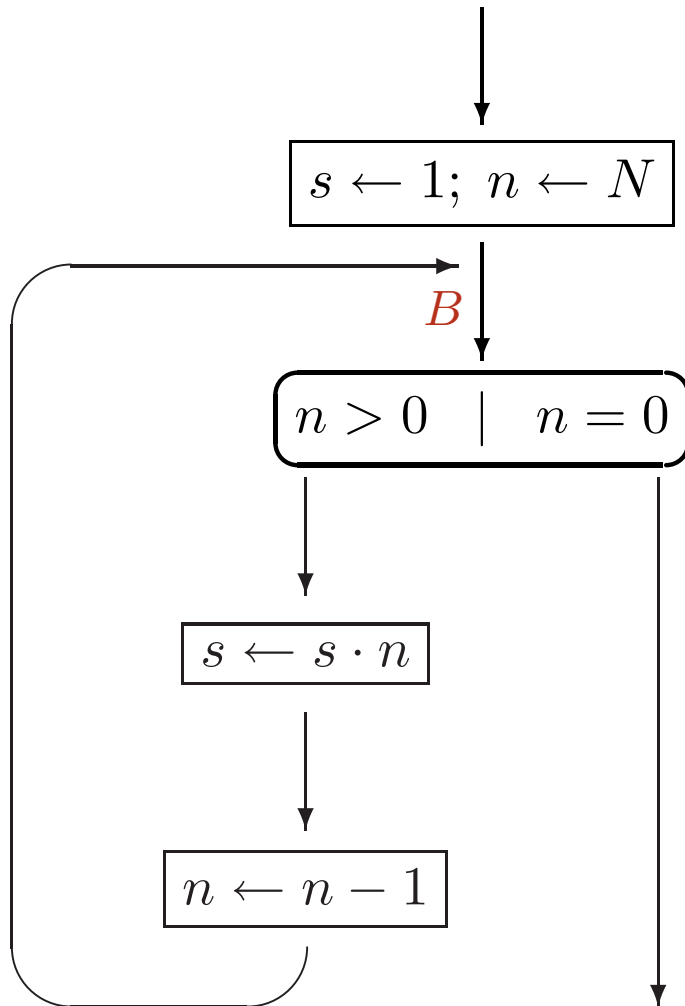
zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

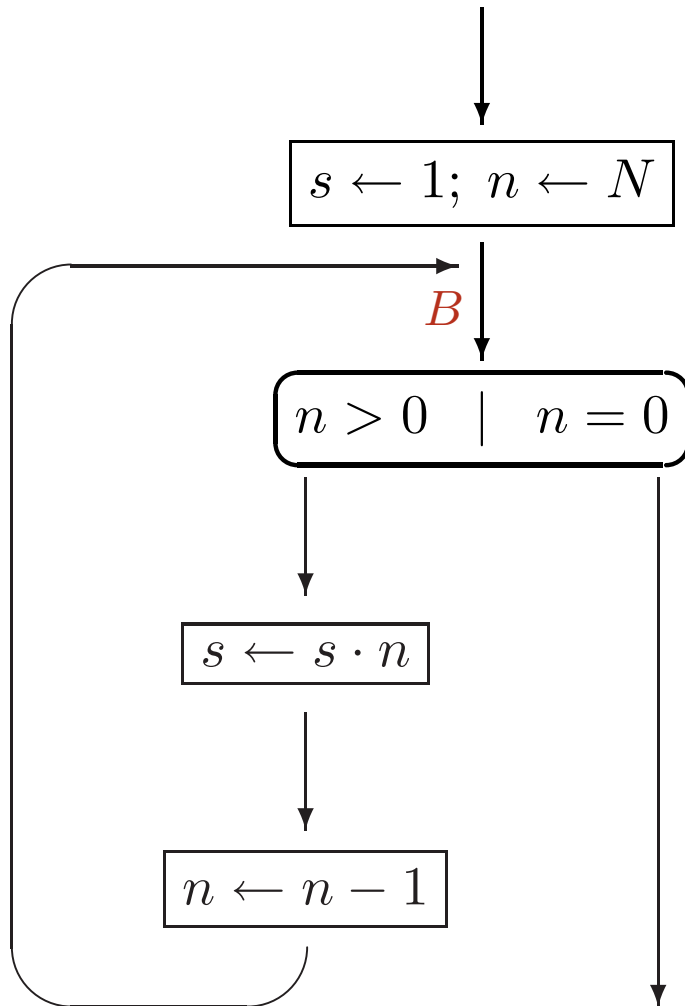
przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

Wiemy: $s \cdot n! = N! \ \& \ n > 0$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

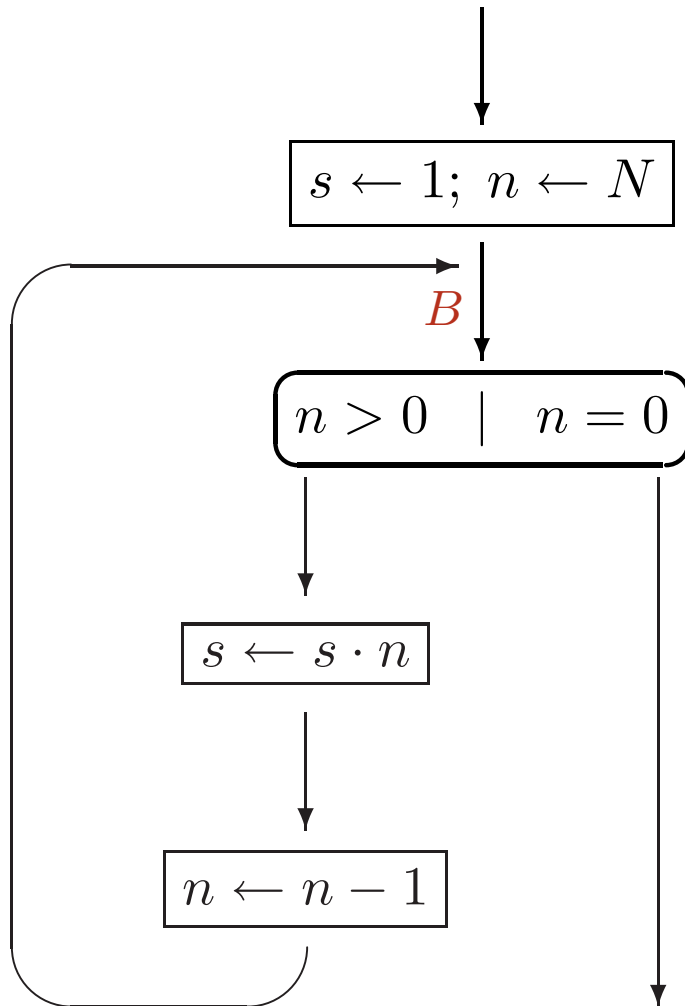
zmienne primowane —

po przejściu przez ciało pętli

Wiemy: $s \cdot n! = N! \ \& \ n > 0$

oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

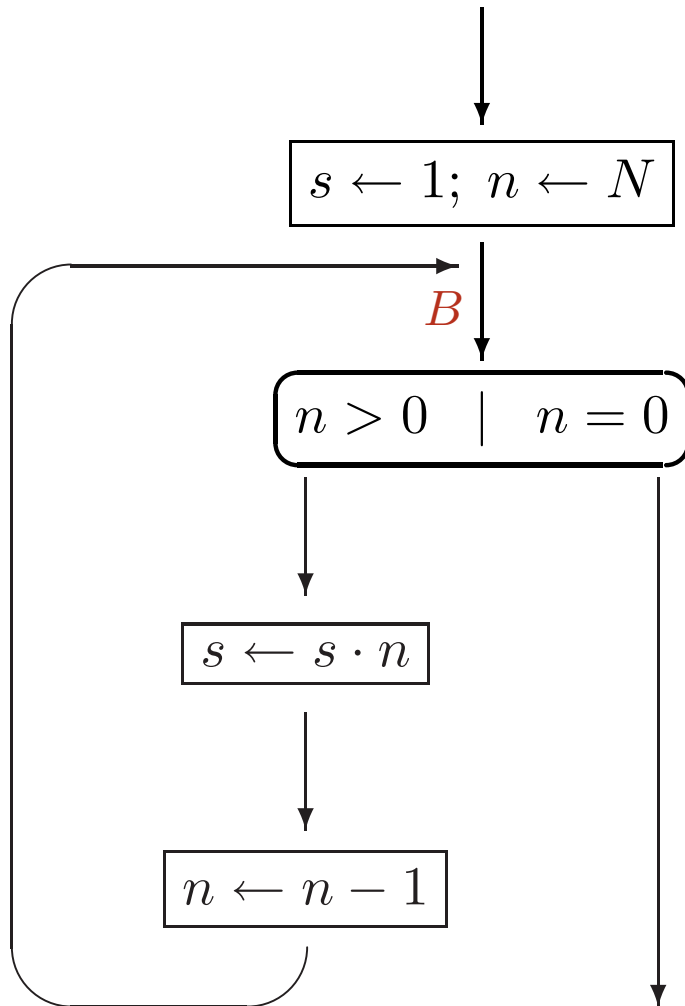
Wiemy: $s \cdot n! = N! \ \& \ n > 0$

oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Stąd wnioskujemy:

$$s' \cdot n'!$$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

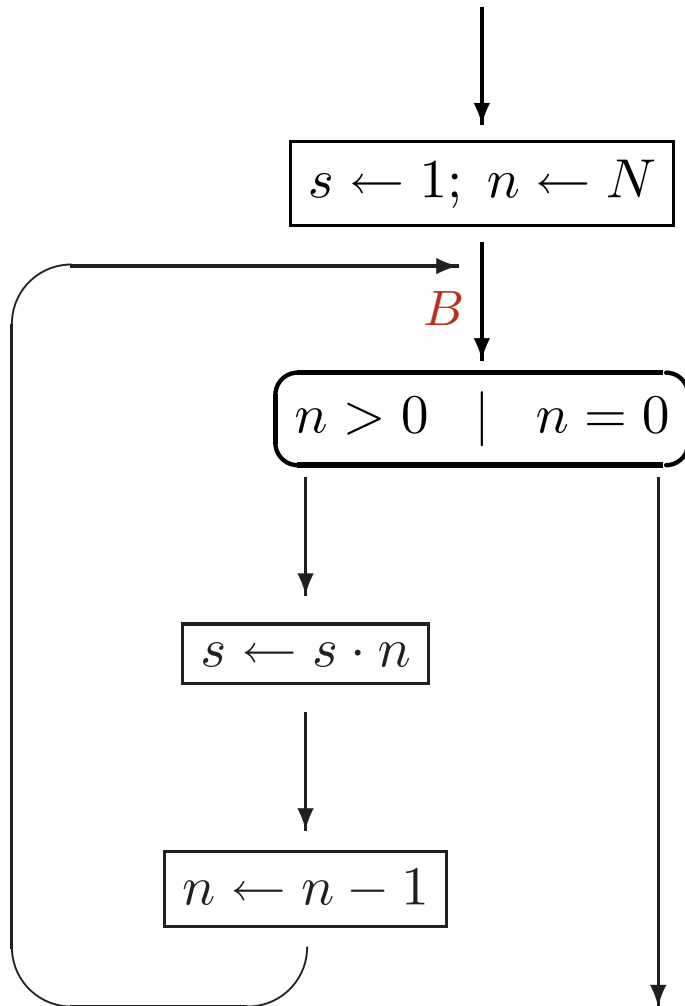
Wiemy: $s \cdot n! = N! \ \& \ n > 0$

oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Stąd wnioskujemy:

$$s' \cdot n'! = (s \cdot n) \cdot (n - 1)!$$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

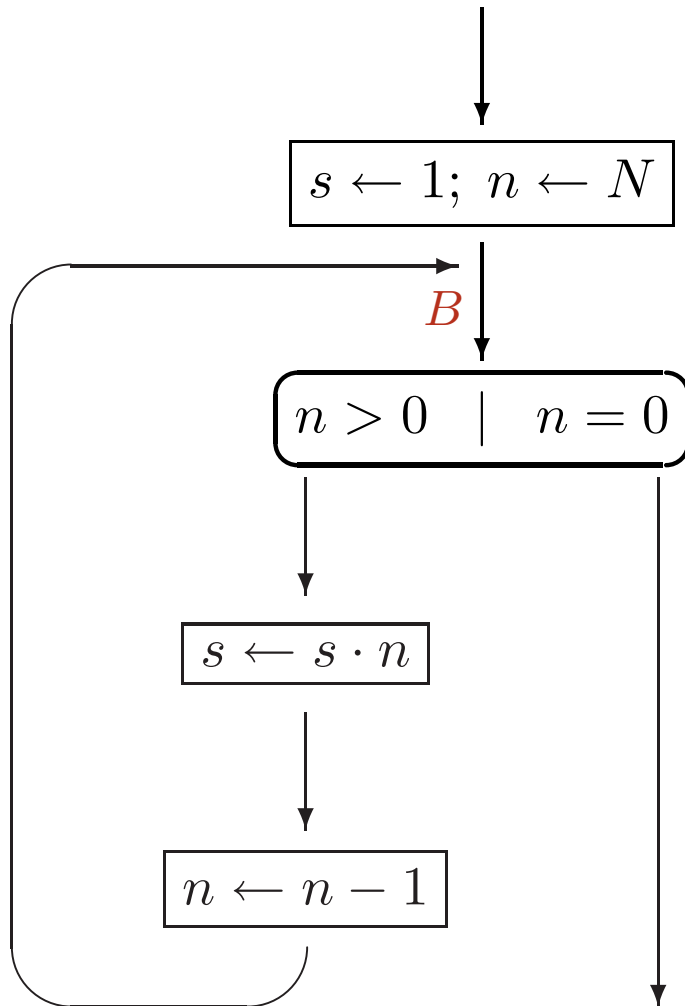
Wiemy: $s \cdot n! = N! \ \& \ n > 0$

oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Stąd wnioskujemy:

$$\begin{aligned}
 s' \cdot n'! &= (s \cdot n) \cdot (n - 1)! \\
 &= s \cdot (n \cdot (n - 1)!)
 \end{aligned}$$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

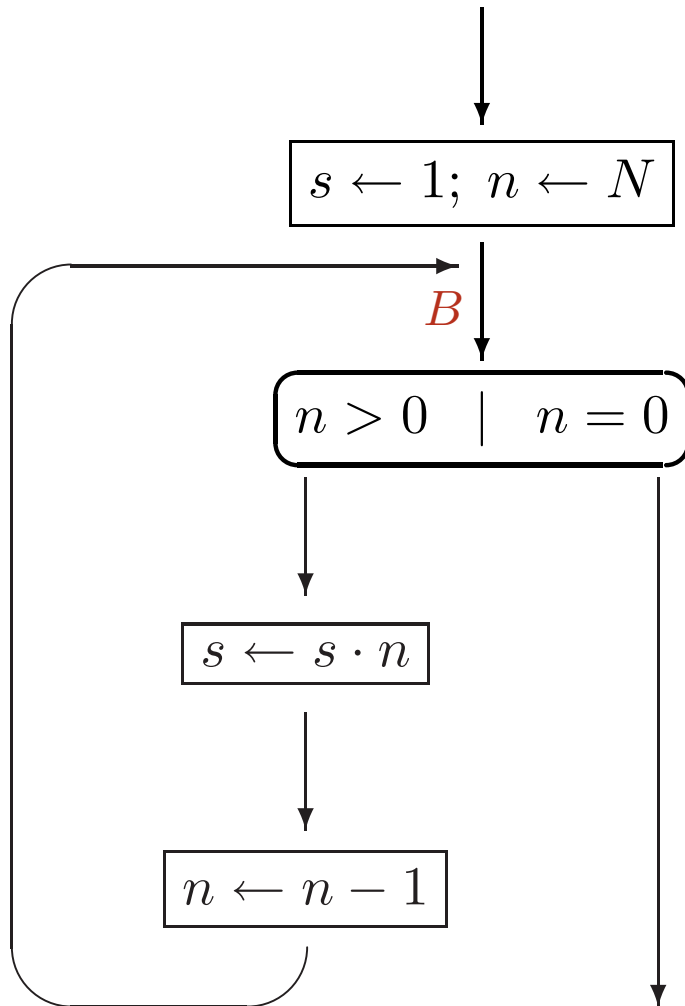
Wiemy: $s \cdot n! = N! \ \& \ n > 0$

oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Stąd wnioskujemy:

$$\begin{aligned}
 s' \cdot n'! &= (s \cdot n) \cdot (n - 1)! \\
 &= s \cdot (n \cdot (n - 1)!) \\
 &= s \cdot n!
 \end{aligned}$$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

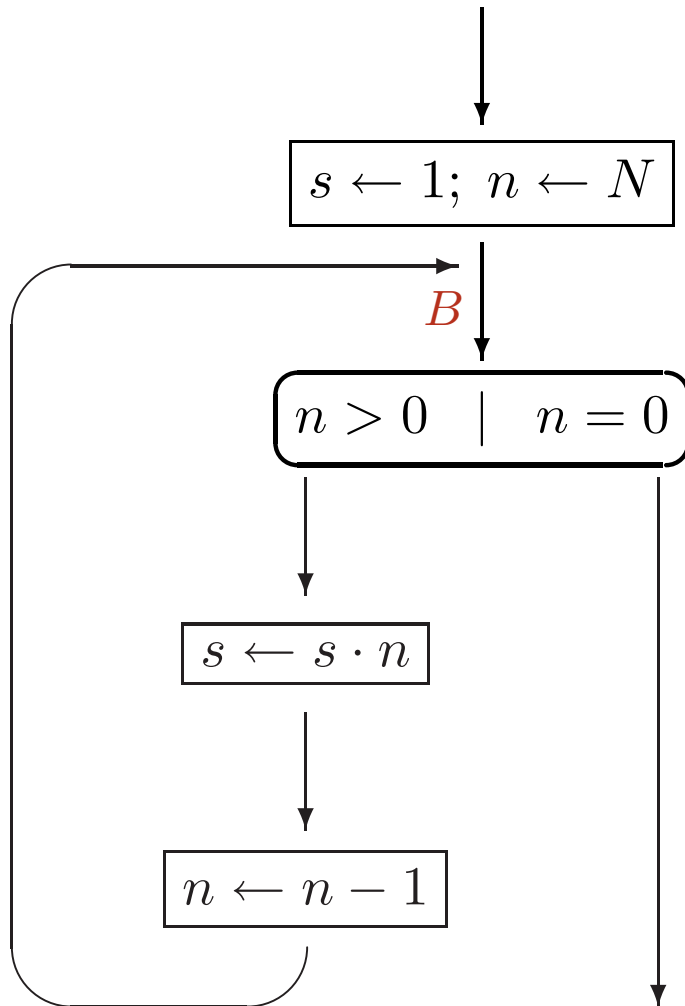
Wiemy: $s \cdot n! = N! \ \& \ n > 0$

oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Stąd wnioskujemy:

$$\begin{aligned}
 s' \cdot n'! &= (s \cdot n) \cdot (n - 1)! \\
 &= s \cdot (n \cdot (n - 1)!) \\
 &= s \cdot n! \\
 &= N!
 \end{aligned}$$

Co liczy dany program?



Czy $s \cdot n! = N!$ jest niezmiennikiem w B ?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

Wiemy: $s \cdot n! = N! \ \& \ n > 0$

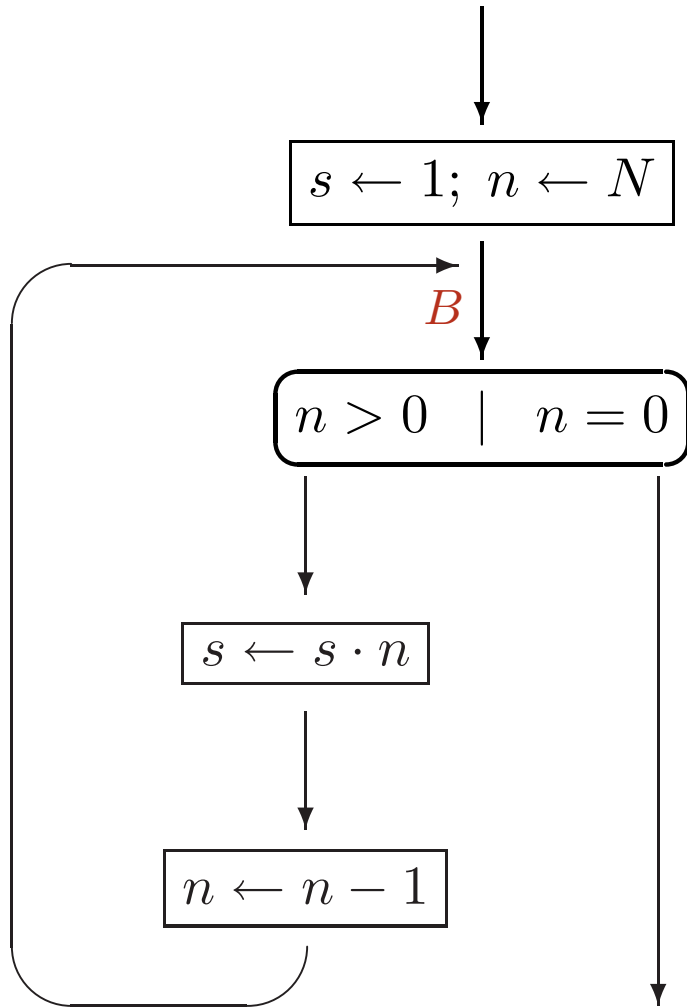
oraz $s' = s \cdot n \quad \& \quad n' = n - 1$

Stąd wnioskujemy:

$$\begin{aligned}
 s' \cdot n'! &= (s \cdot n) \cdot (n - 1)! \\
 &= s \cdot (n \cdot (n - 1)!) \\
 &= s \cdot n! \\
 &= N!
 \end{aligned}$$

To dowodzi niezmienniczości tej formuły.

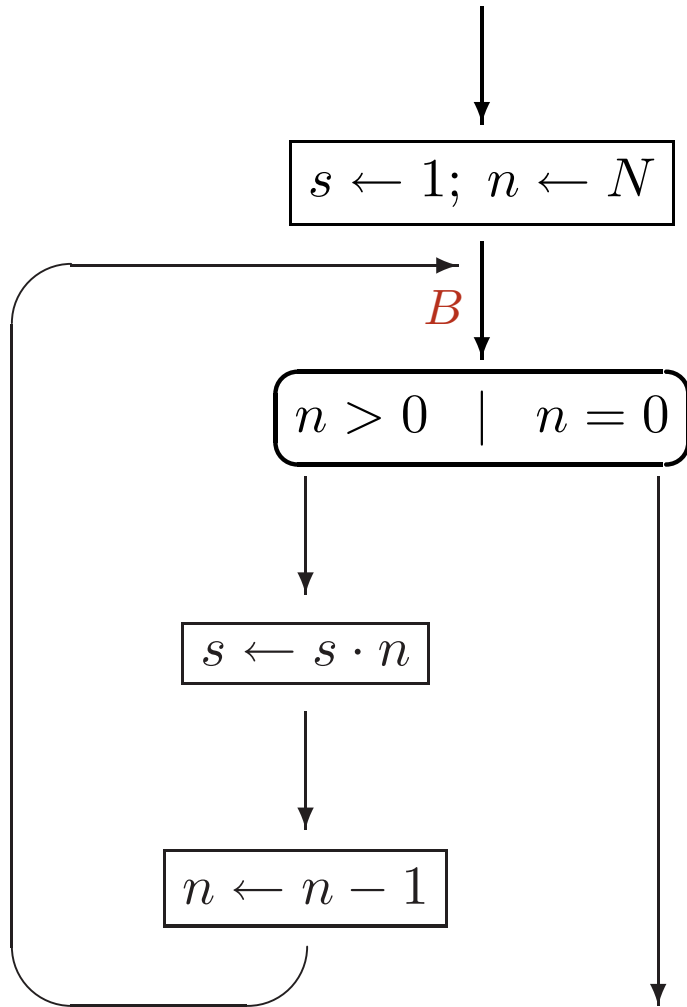
Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$

Co liczy dany program?

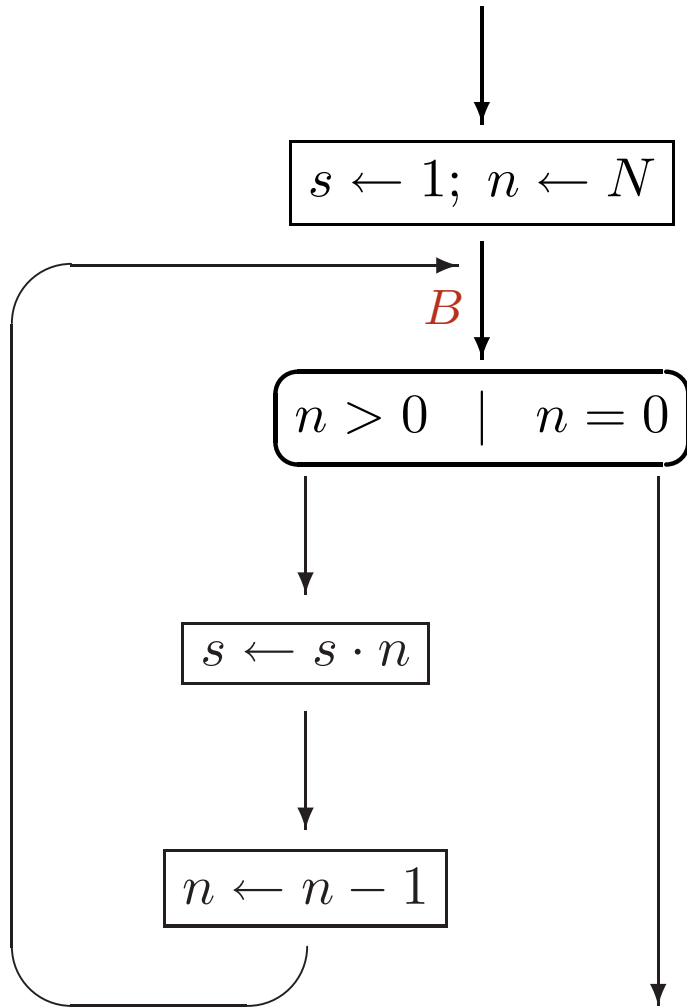


Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n!$$

Co liczy dany program?

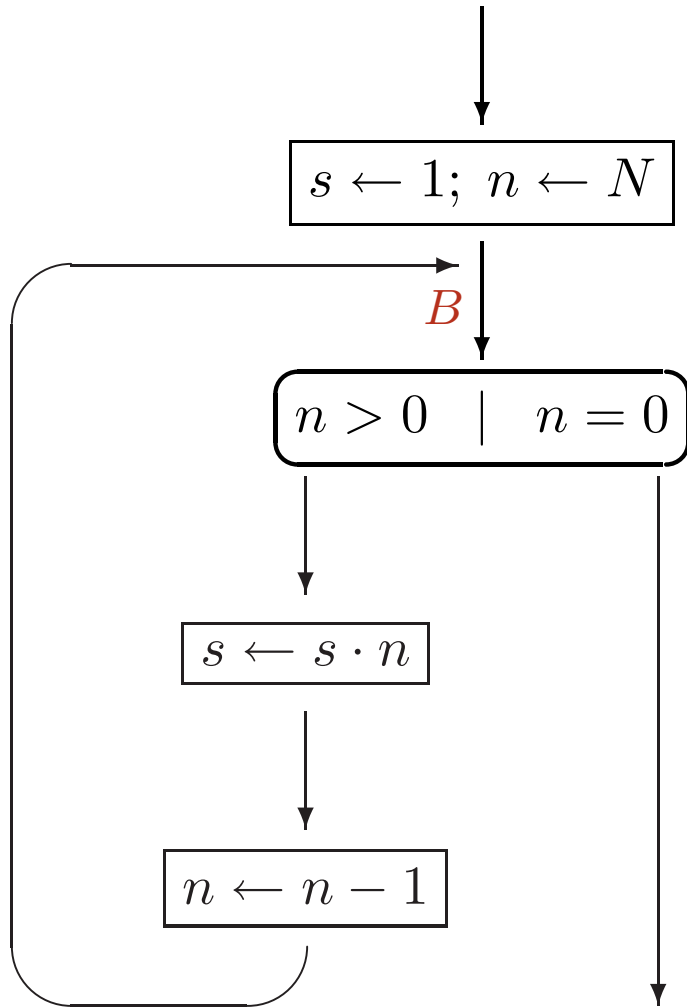


Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N!$$

Co liczy dany program?

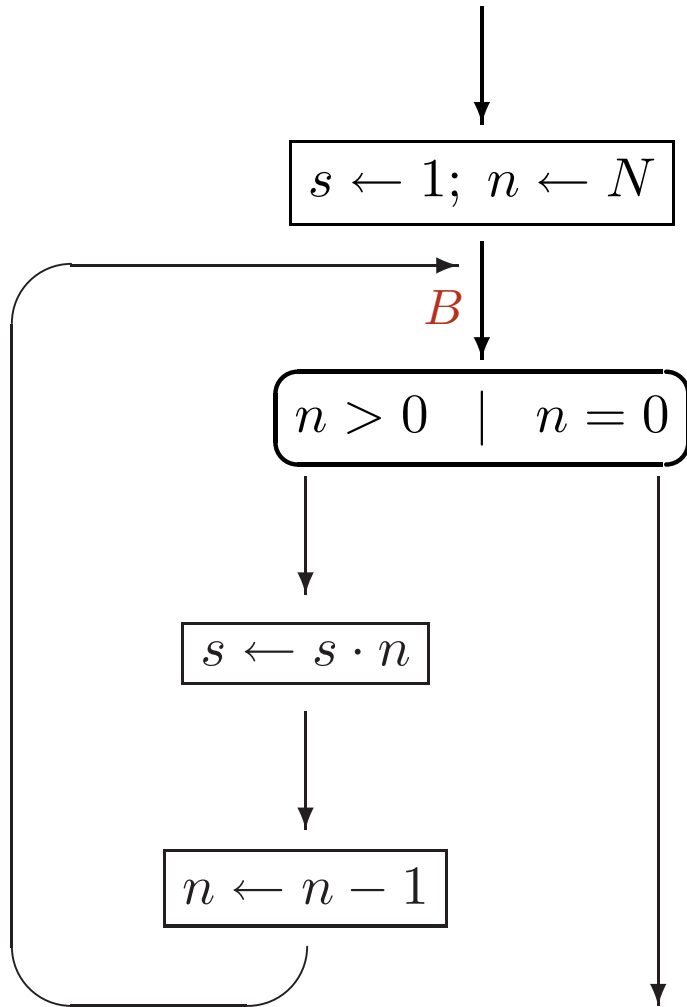


Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

Co liczy dany program?



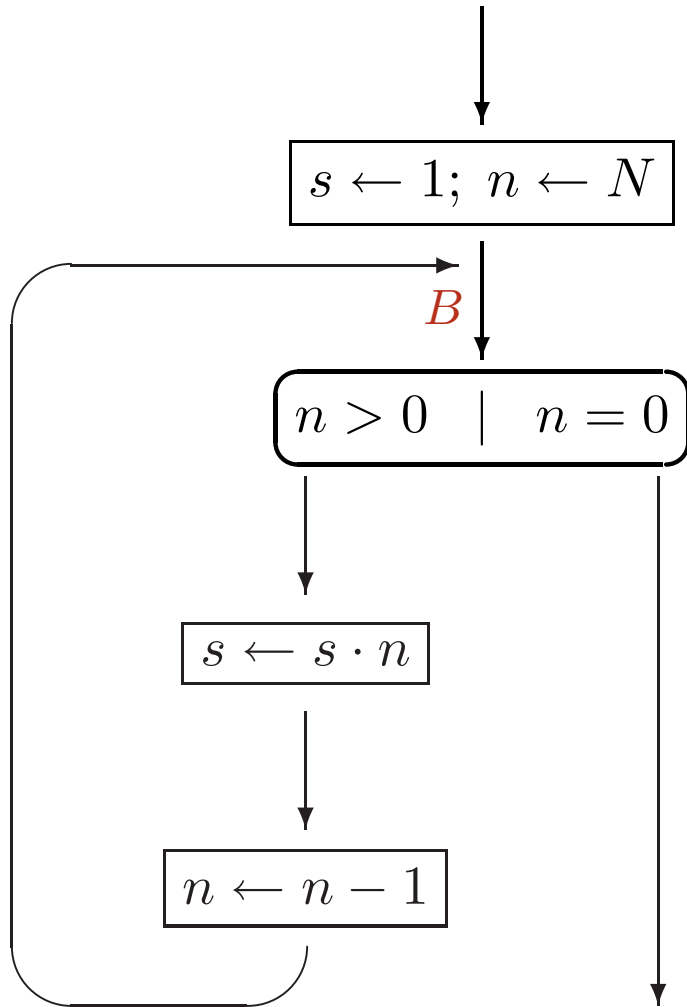
Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

więc niezmiennik jest wtedy spełniony.

Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

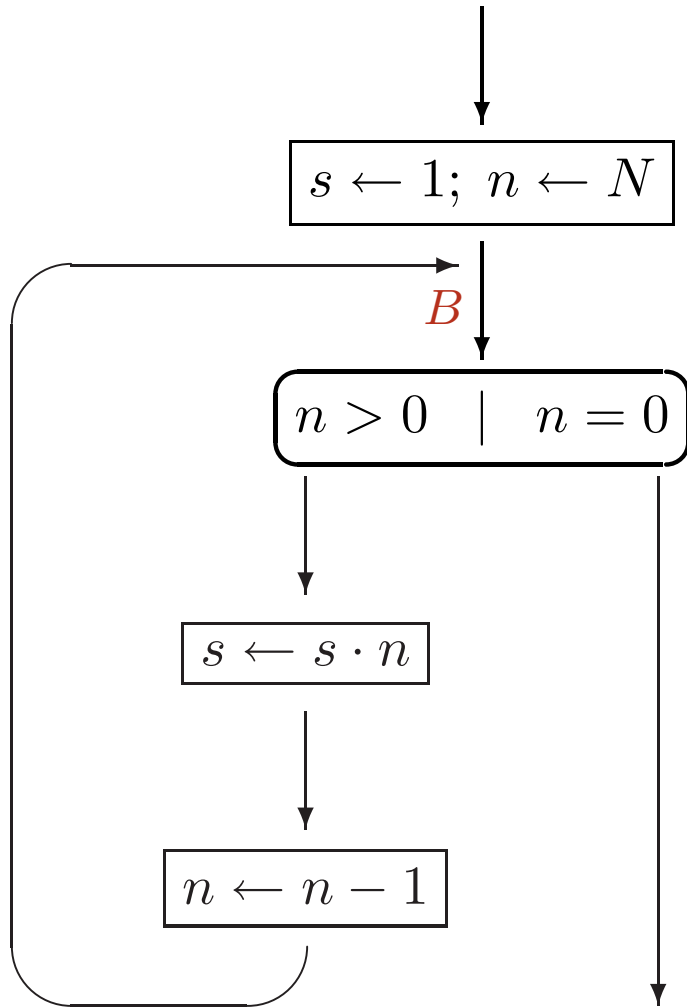
$$s \cdot n! = 1 \cdot N! = N!$$

więc niezmiennik jest wtedy spełniony.

Wyjście z pętli:

jeśli $s \cdot n! = N!$ i $n = 0$

Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

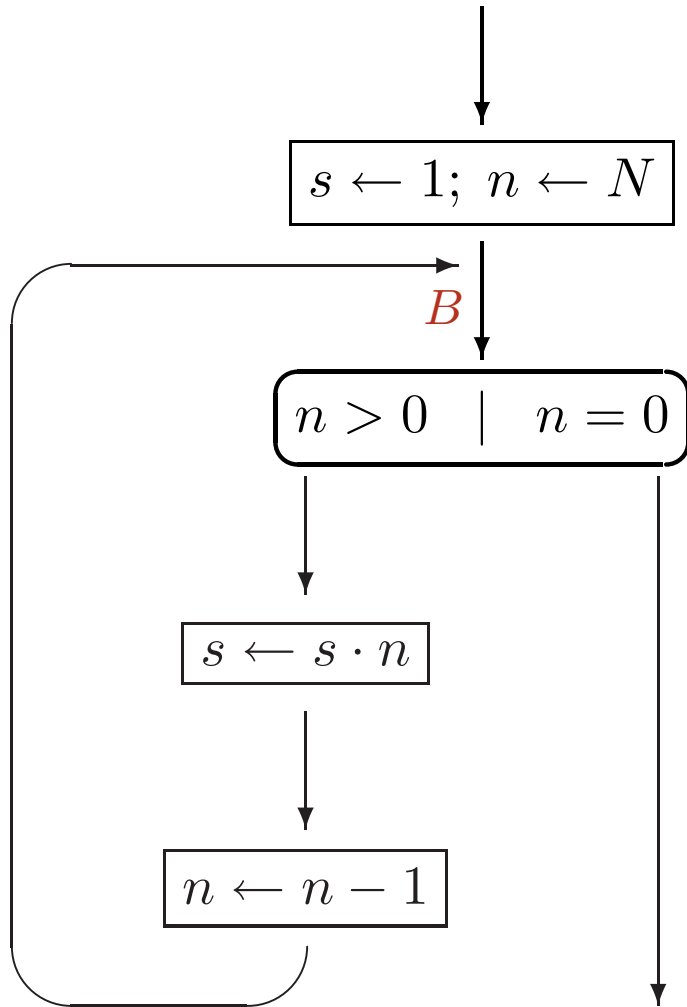
więc niezmiennik jest wtedy spełniony.

Wyjście z pętli:

jeśli $s \cdot n! = N!$ i $n = 0$, to

$$s = s \cdot 1$$

Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

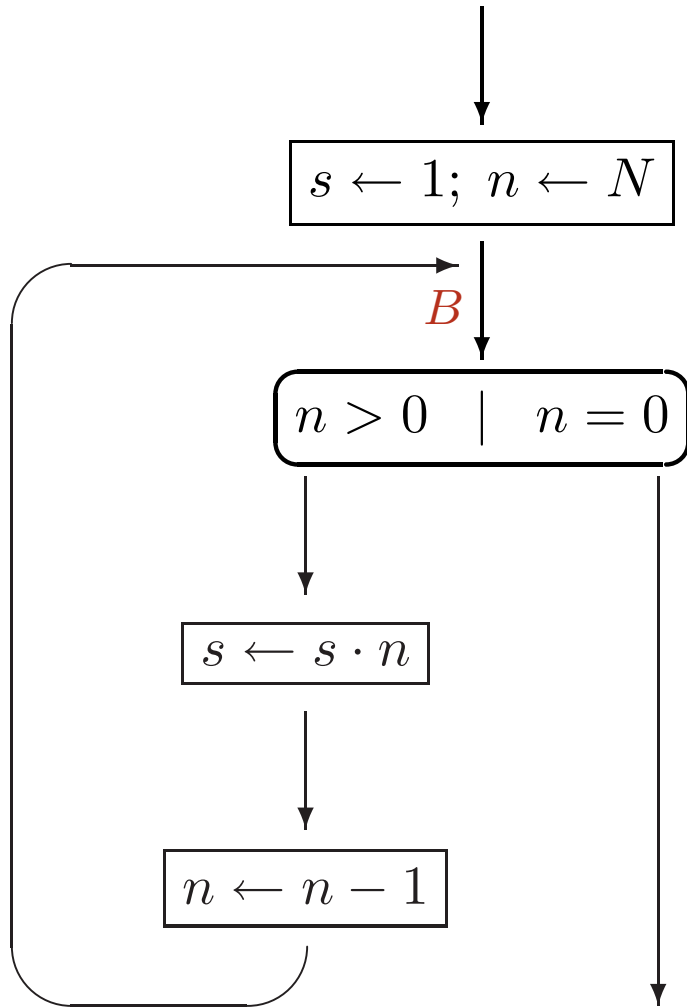
więc niezmiennik jest wtedy spełniony.

Wyjście z pętli:

jeśli $s \cdot n! = N!$ i $n = 0$, to

$$s = s \cdot 1 = s \cdot 0!$$

Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

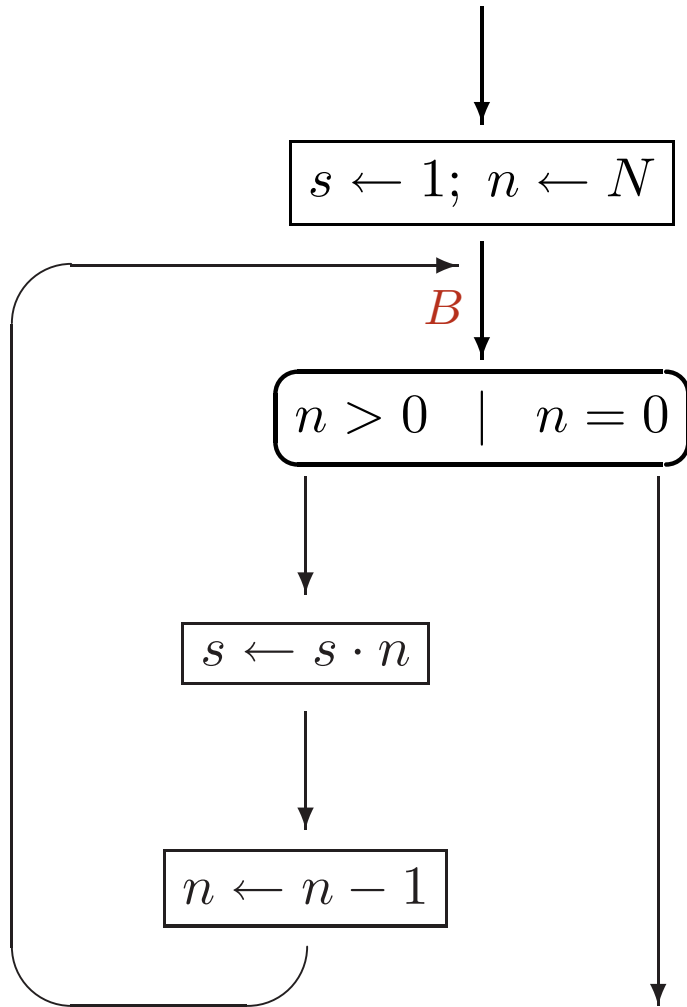
więc niezmiennik jest wtedy spełniony.

Wyjście z pętli:

jeśli $s \cdot n! = N!$ i $n = 0$, to

$$s = s \cdot 1 = s \cdot 0! = s \cdot n!$$

Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

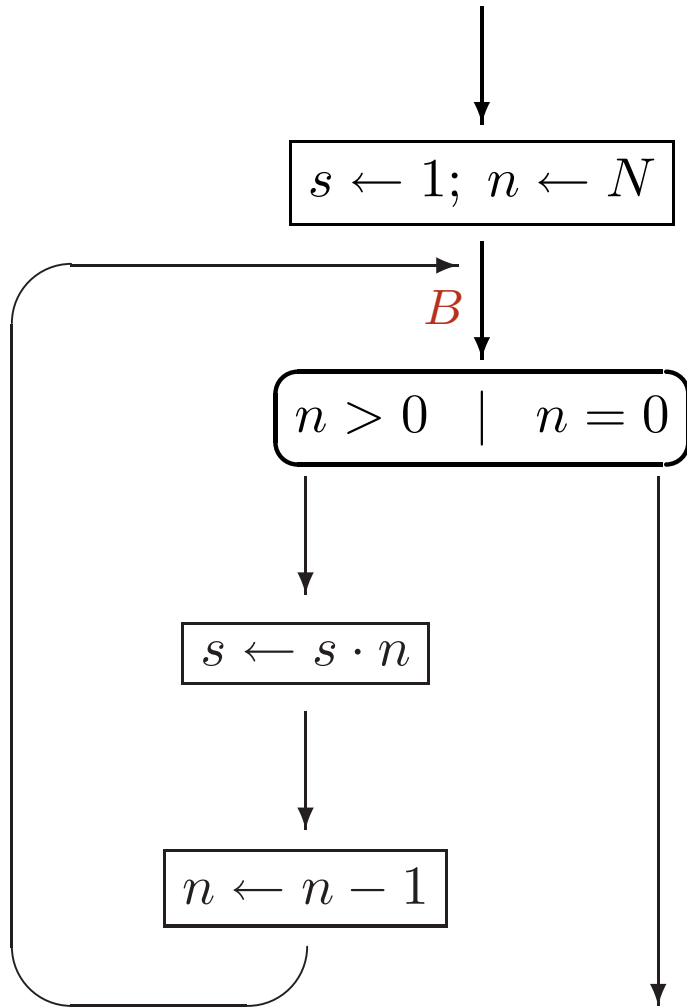
więc niezmiennik jest wtedy spełniony.

Wyjście z pętli:

jeśli $s \cdot n! = N!$ i $n = 0$, to

$$s = s \cdot 1 = s \cdot 0! = s \cdot n! = N!$$

Co liczy dany program?



Wejście do pętli:

jeśli $s = 1$ i $n = N$, to

$$s \cdot n! = 1 \cdot N! = N!$$

więc niezmiennik jest wtedy spełniony.

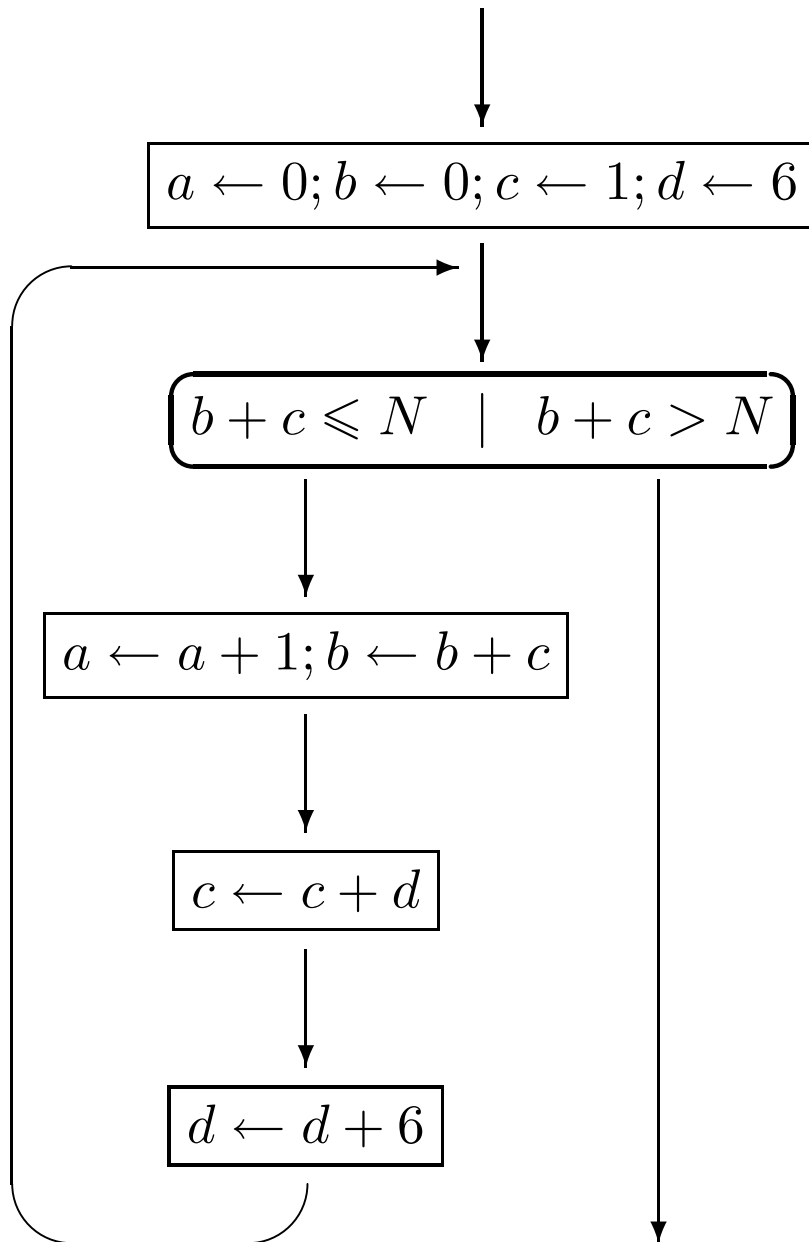
Wyjście z pętli:

jeśli $s \cdot n! = N!$ i $n = 0$, to

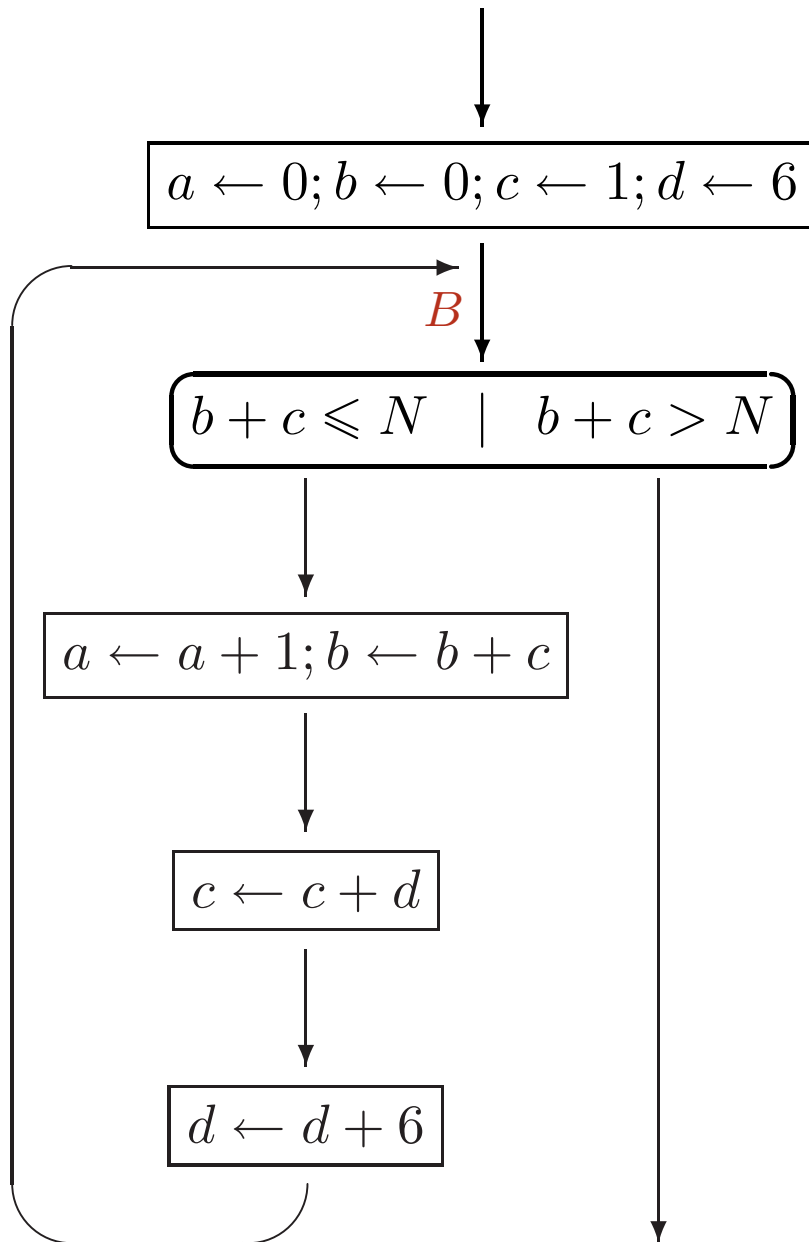
$$s = s \cdot 1 = s \cdot 0! = s \cdot n! = N!$$

więc przy wyjściu z pętli $s = N!$

Co liczy dany program?

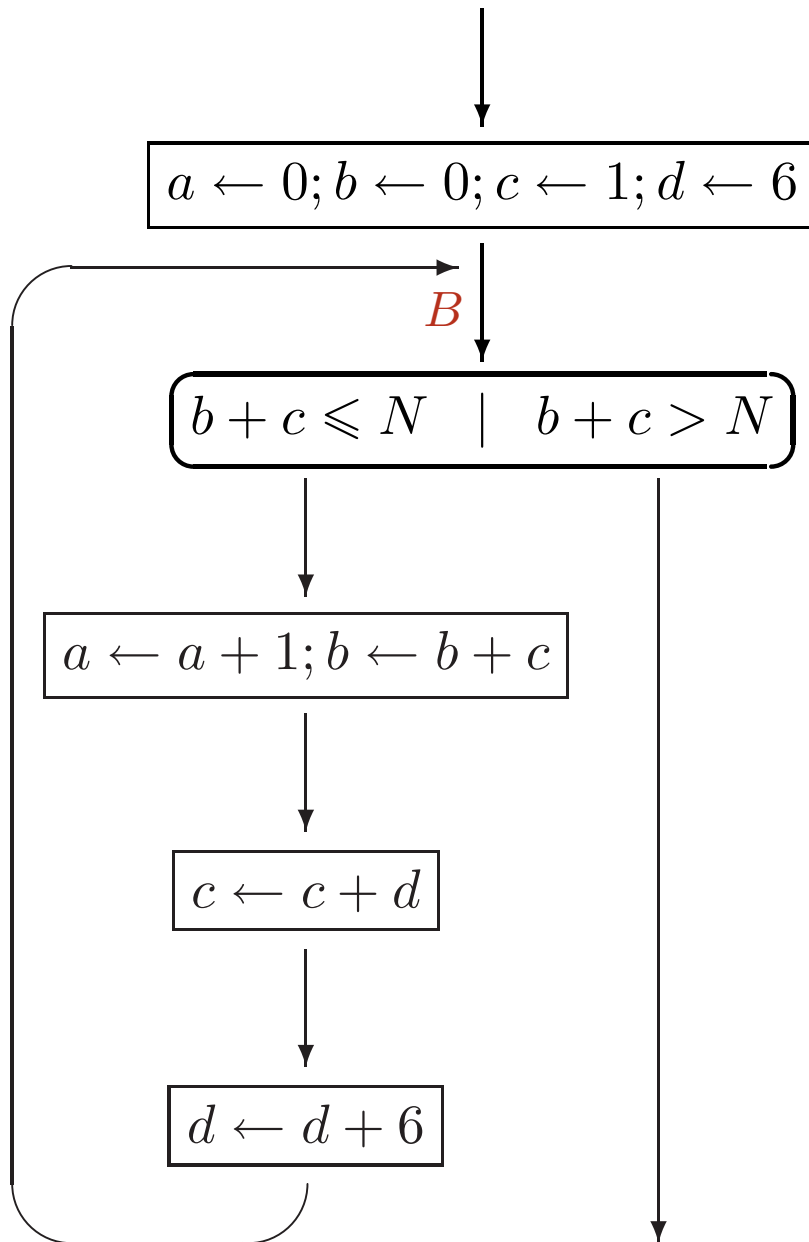


Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli

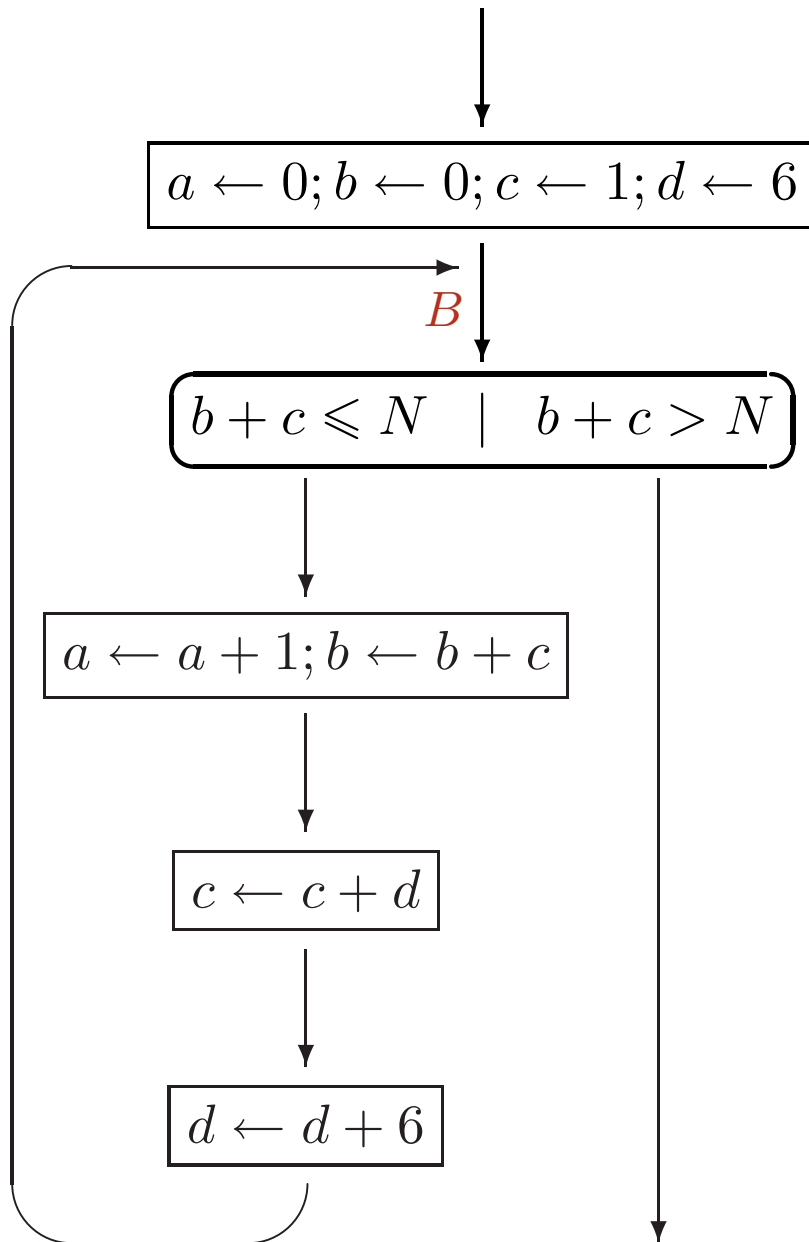
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w B w kilku pierwszych obrotach pętli:

a	b	c	d

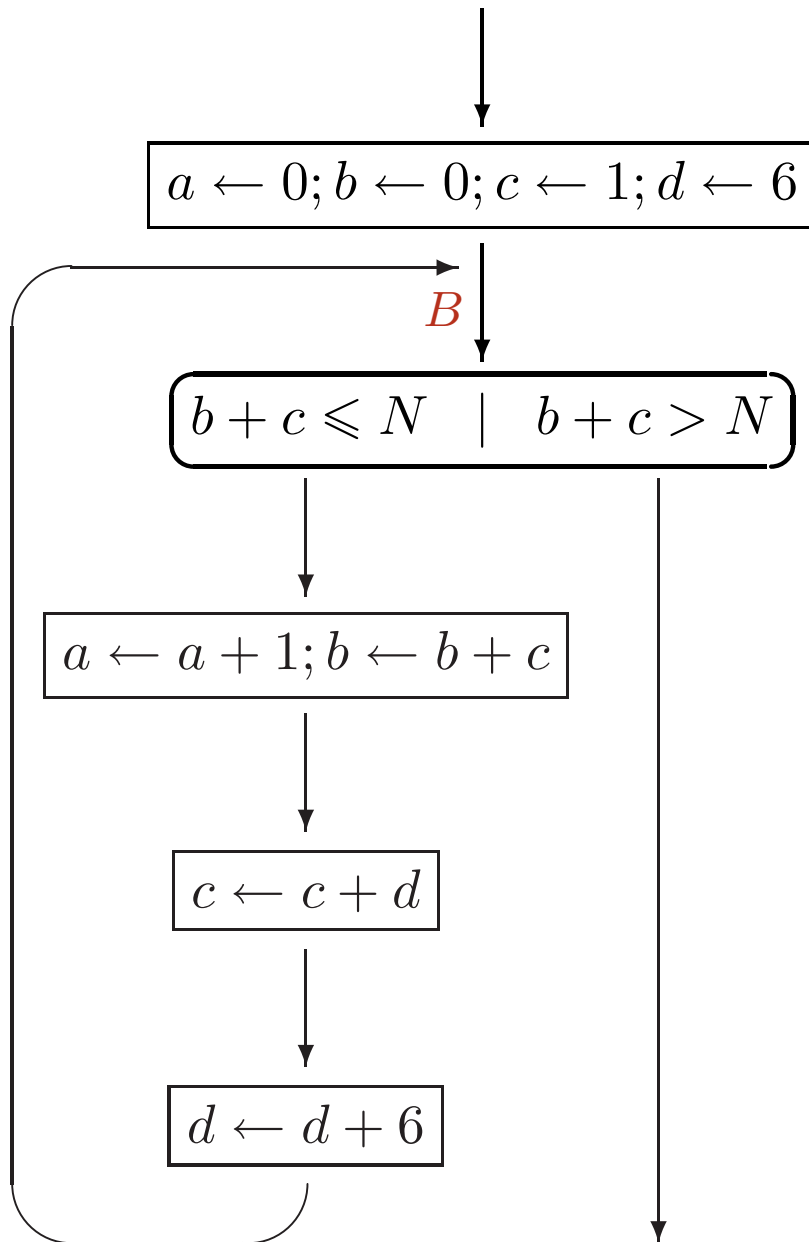
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6

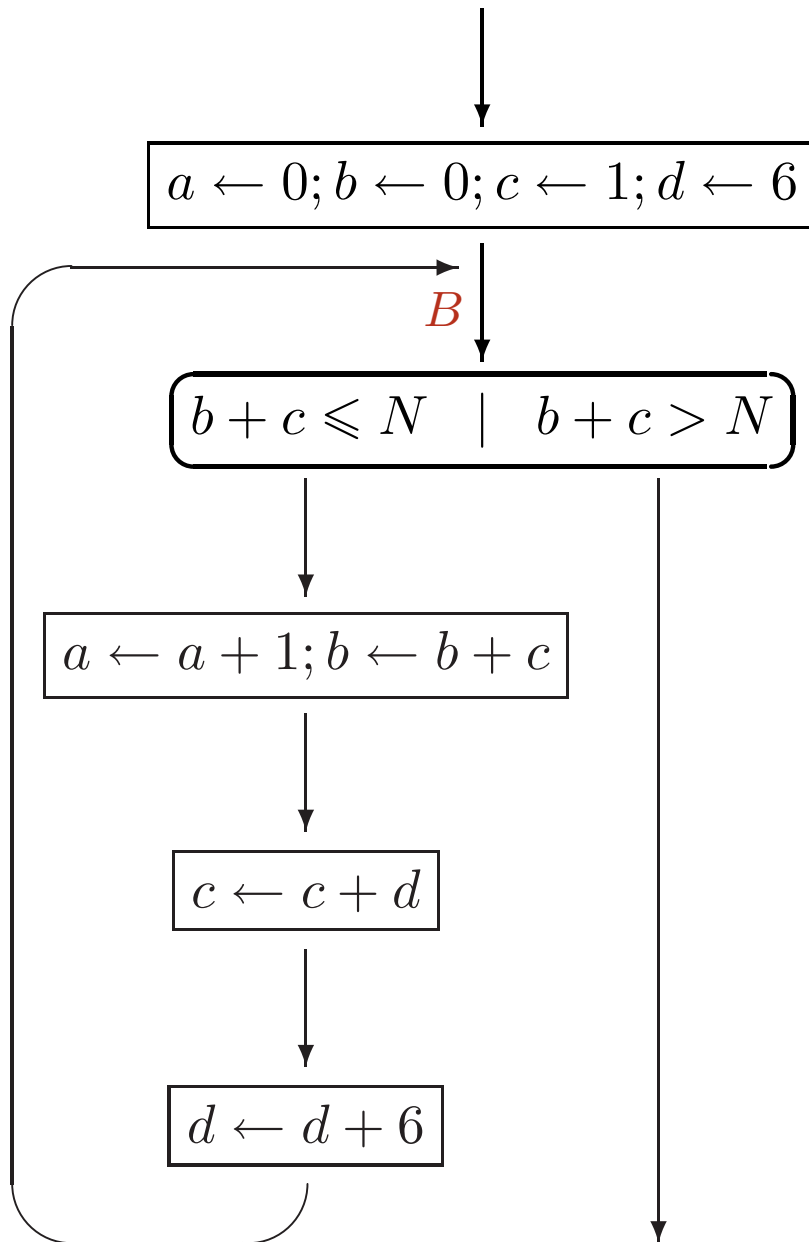
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12

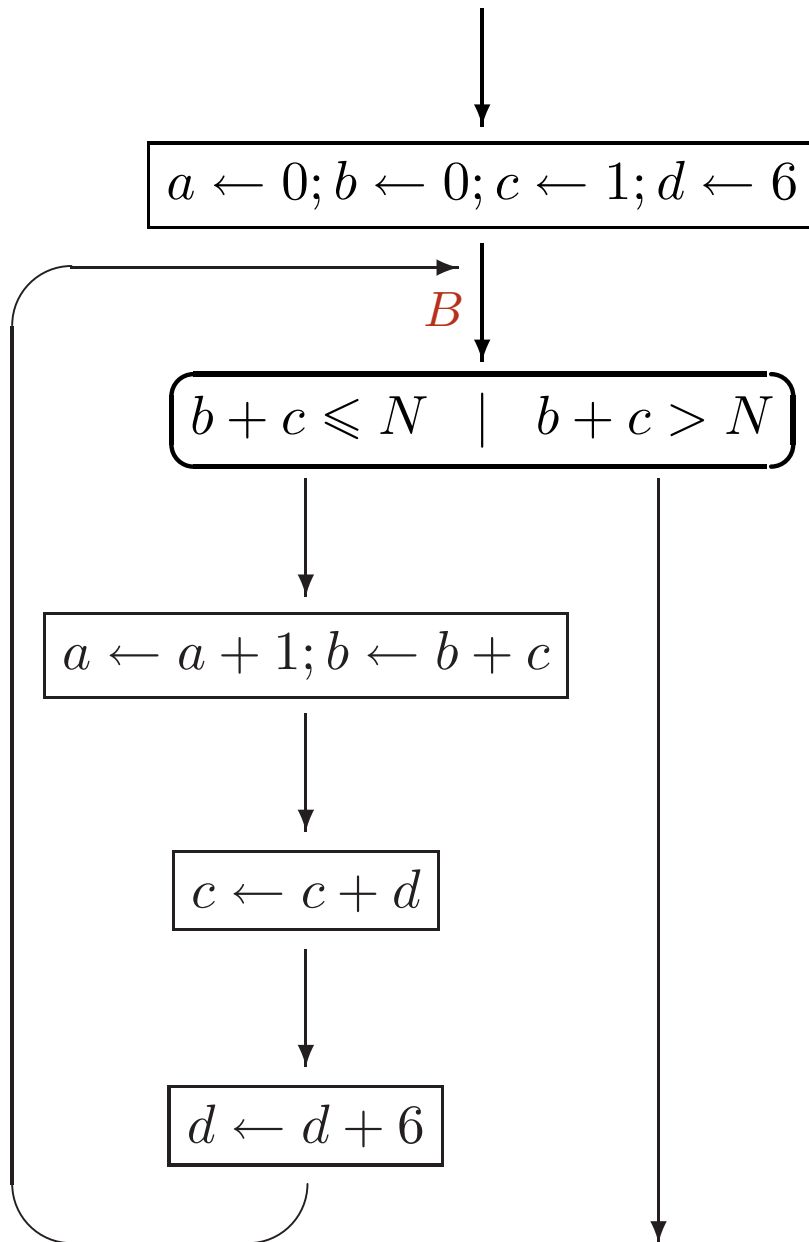
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18

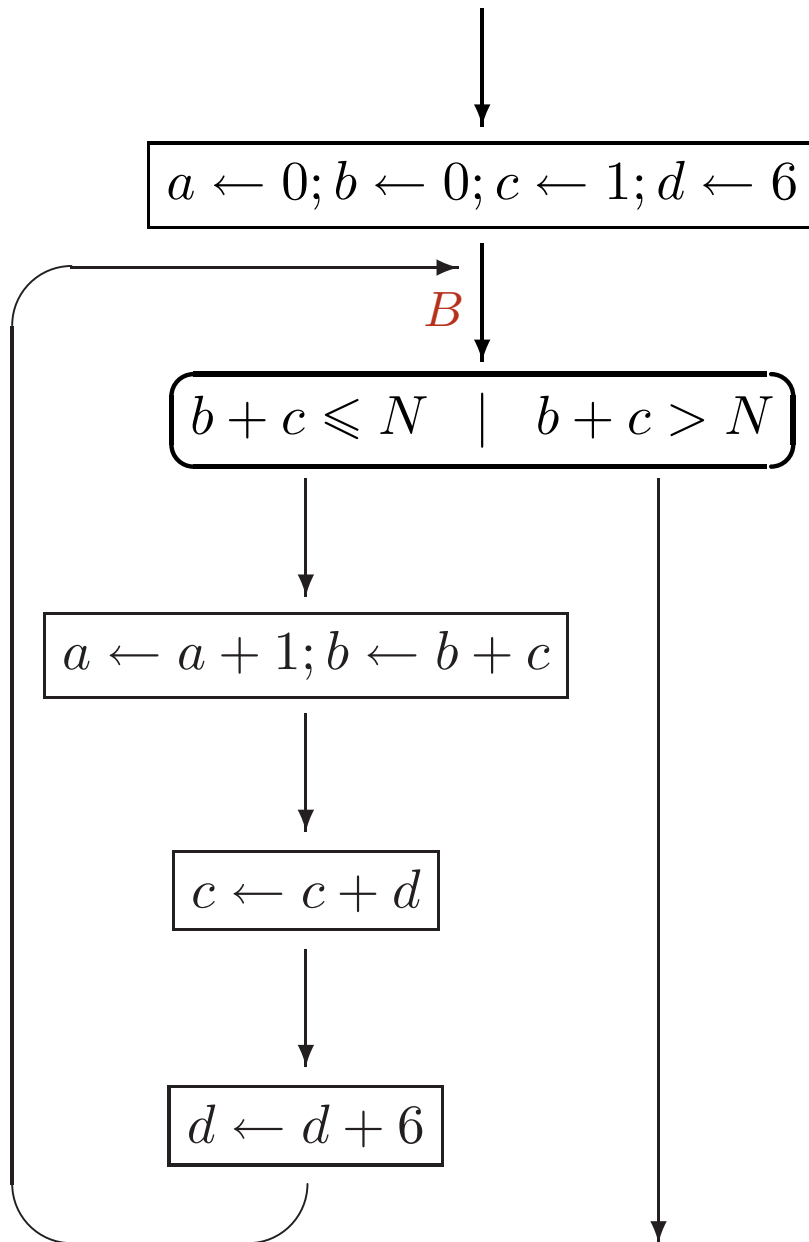
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24

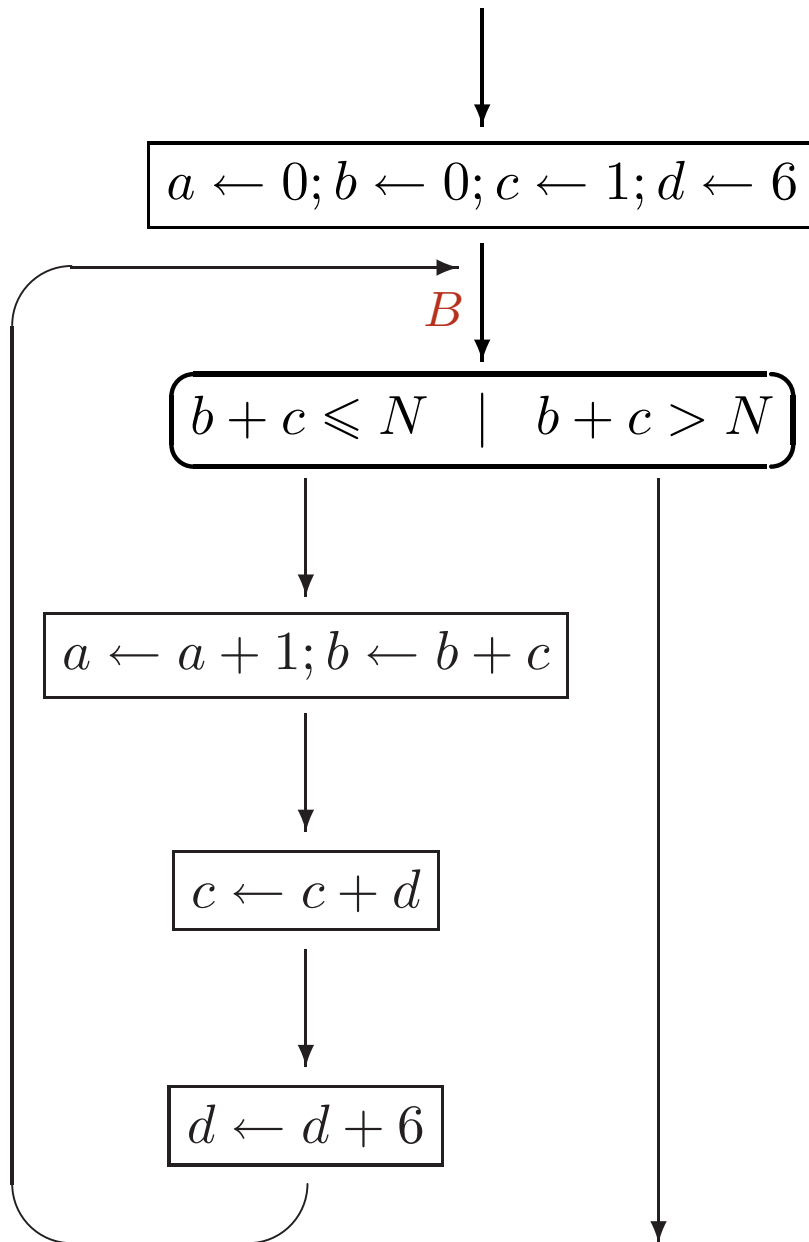
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30

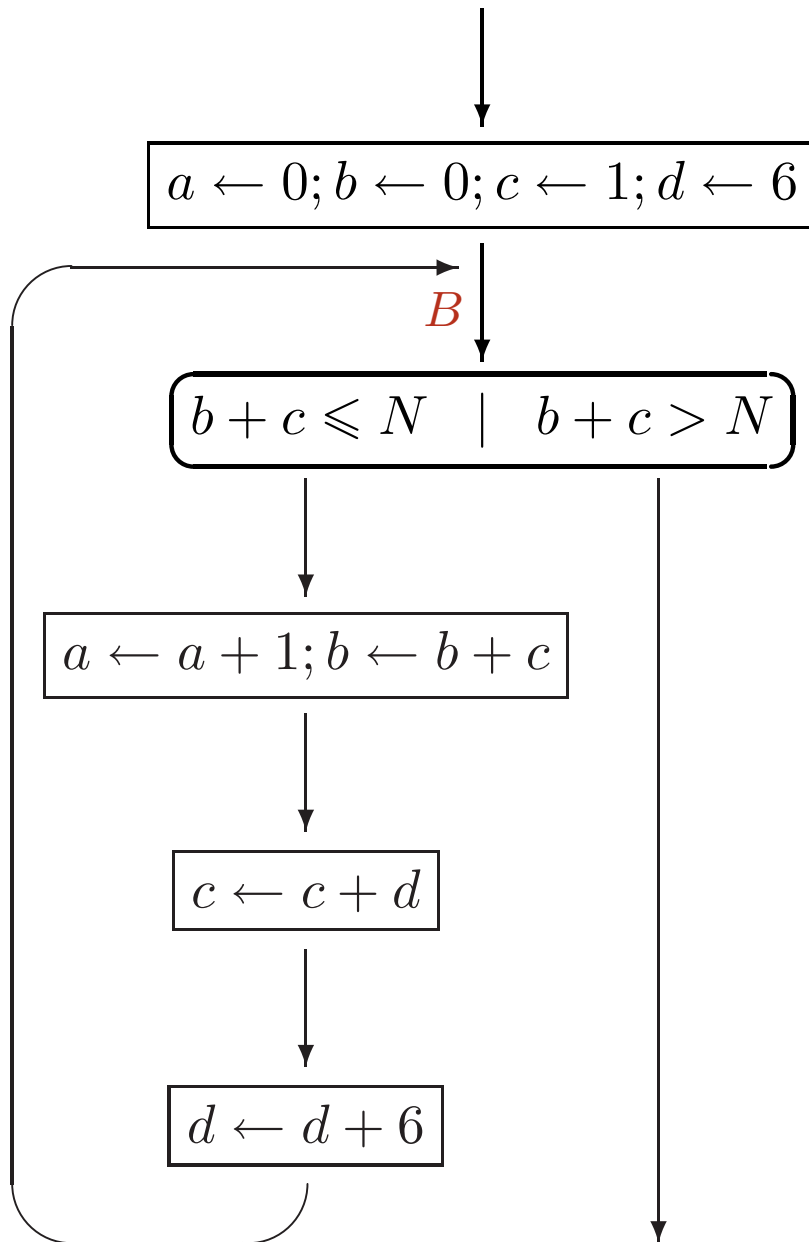
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36

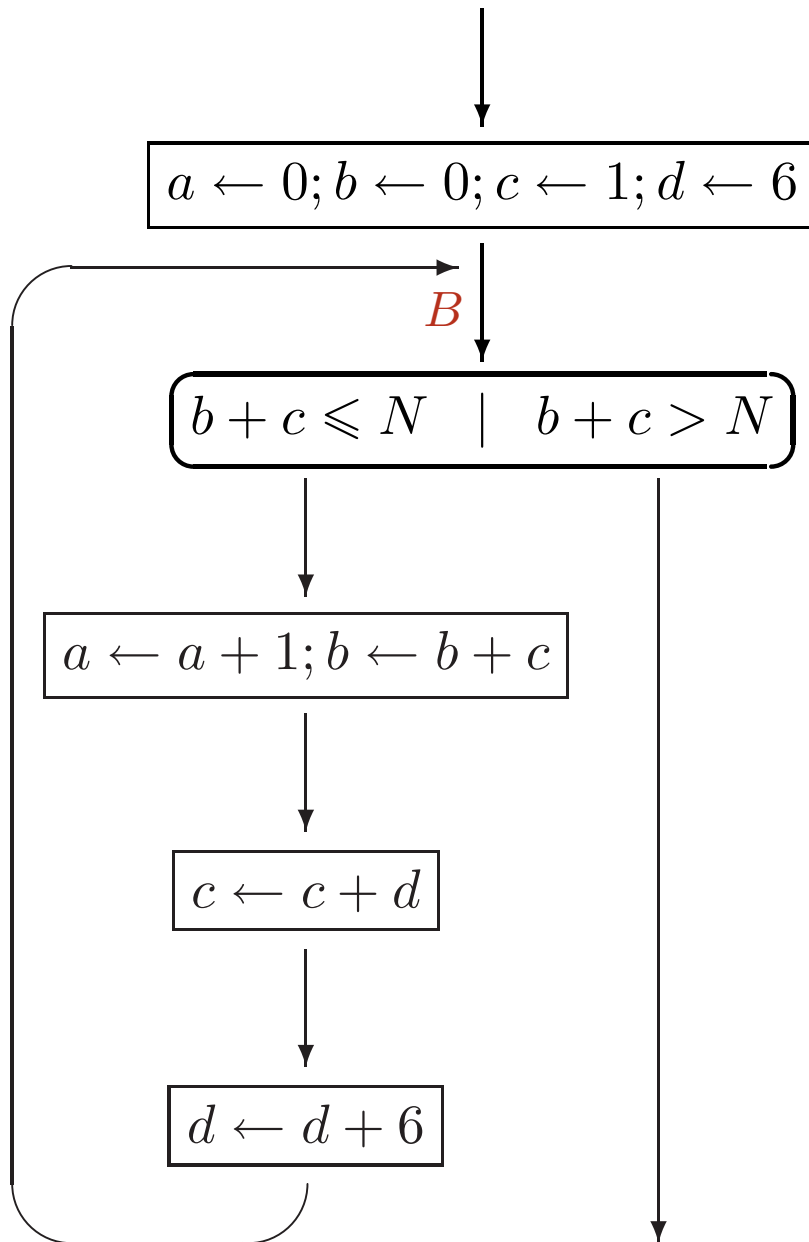
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36
6	216	127	42

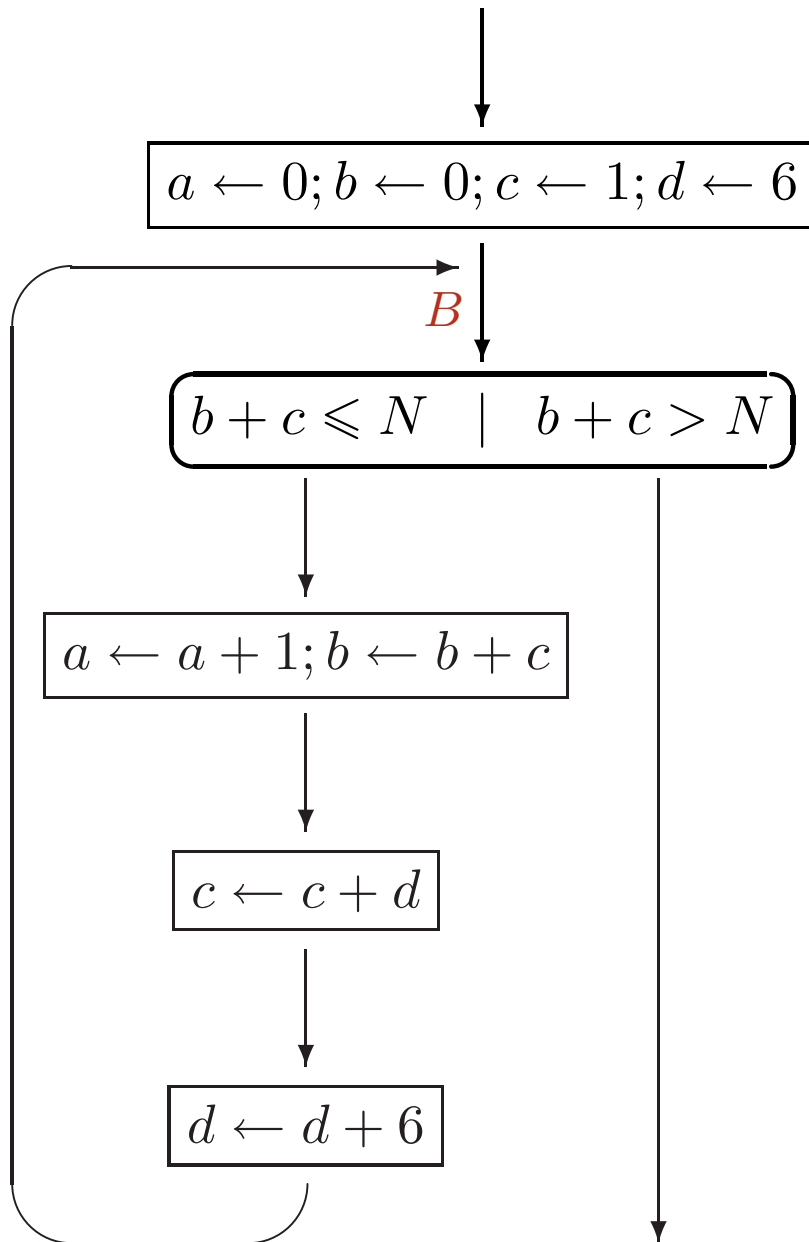
Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36
6	216	127	42
7	343	169	48

Co liczy dany program?

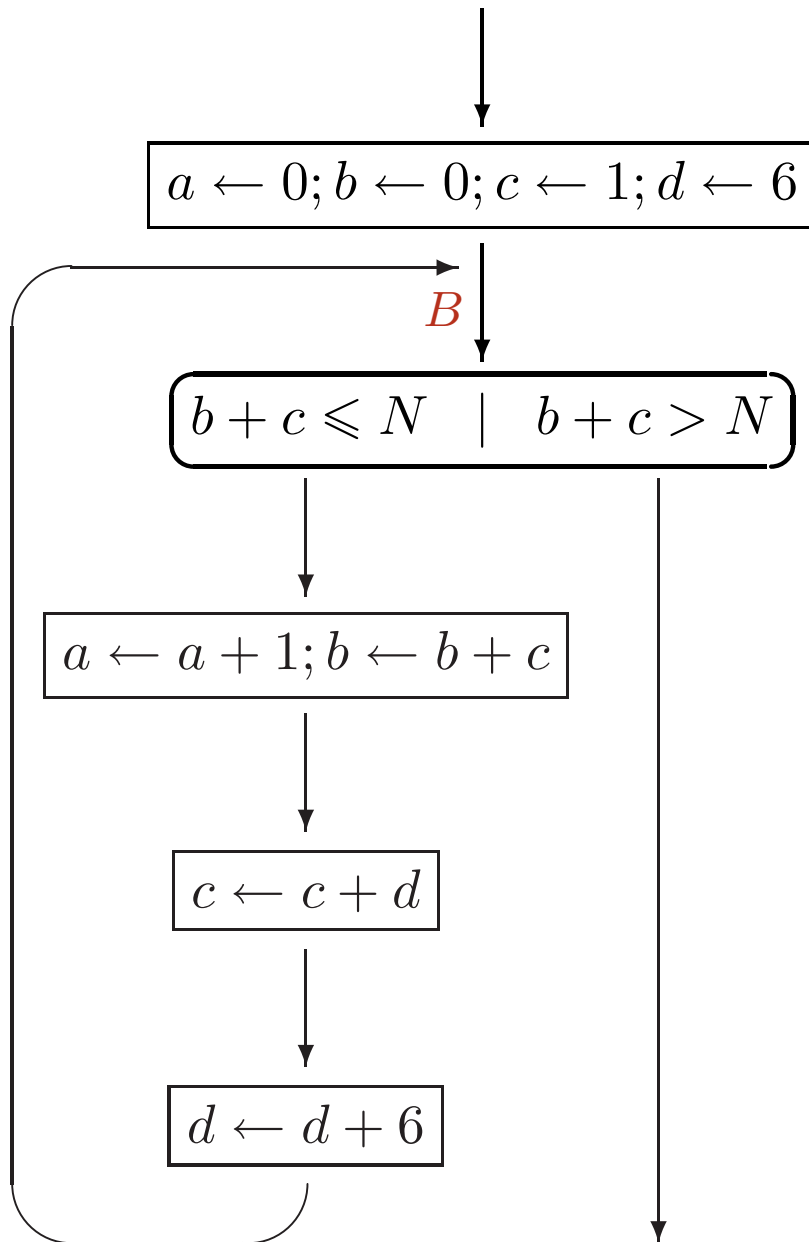


Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36
6	216	127	42
7	343	169	48

Zależności między wartościami zmiennych

Co liczy dany program?



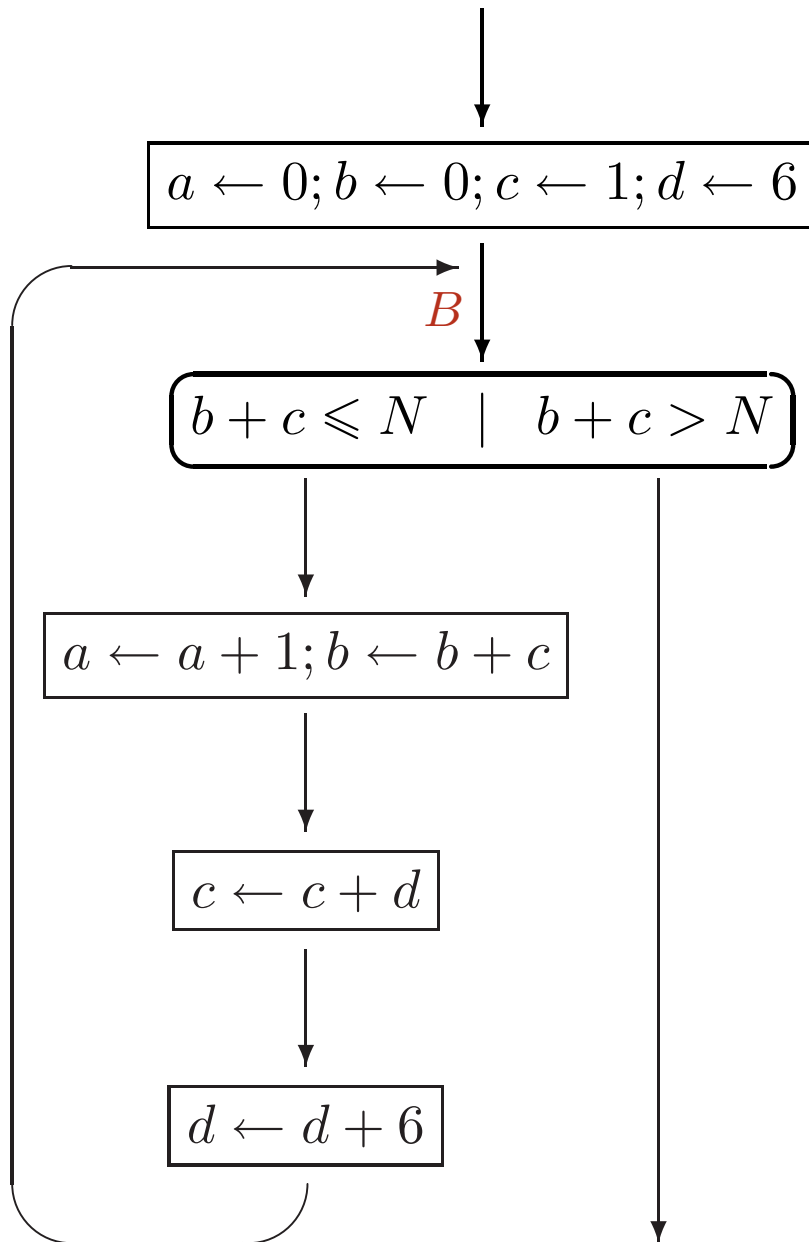
Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36
6	216	127	42
7	343	169	48

Zależności między wartościami zmiennych:

$$b = a^3$$

Co liczy dany program?



Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

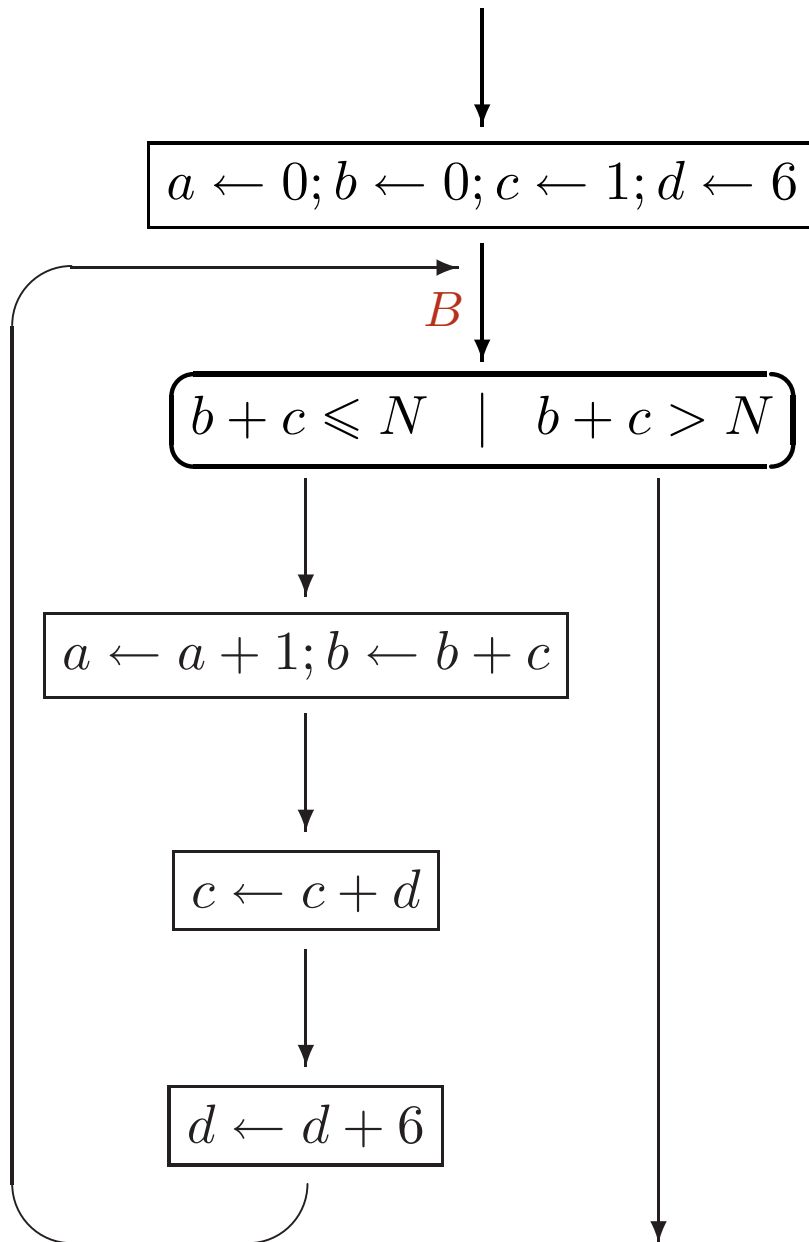
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36
6	216	127	42
7	343	169	48

Zależności między wartościami zmiennych:

$$b = a^3$$

$$d = 6 \cdot (a + 1)$$

Co liczy dany program?



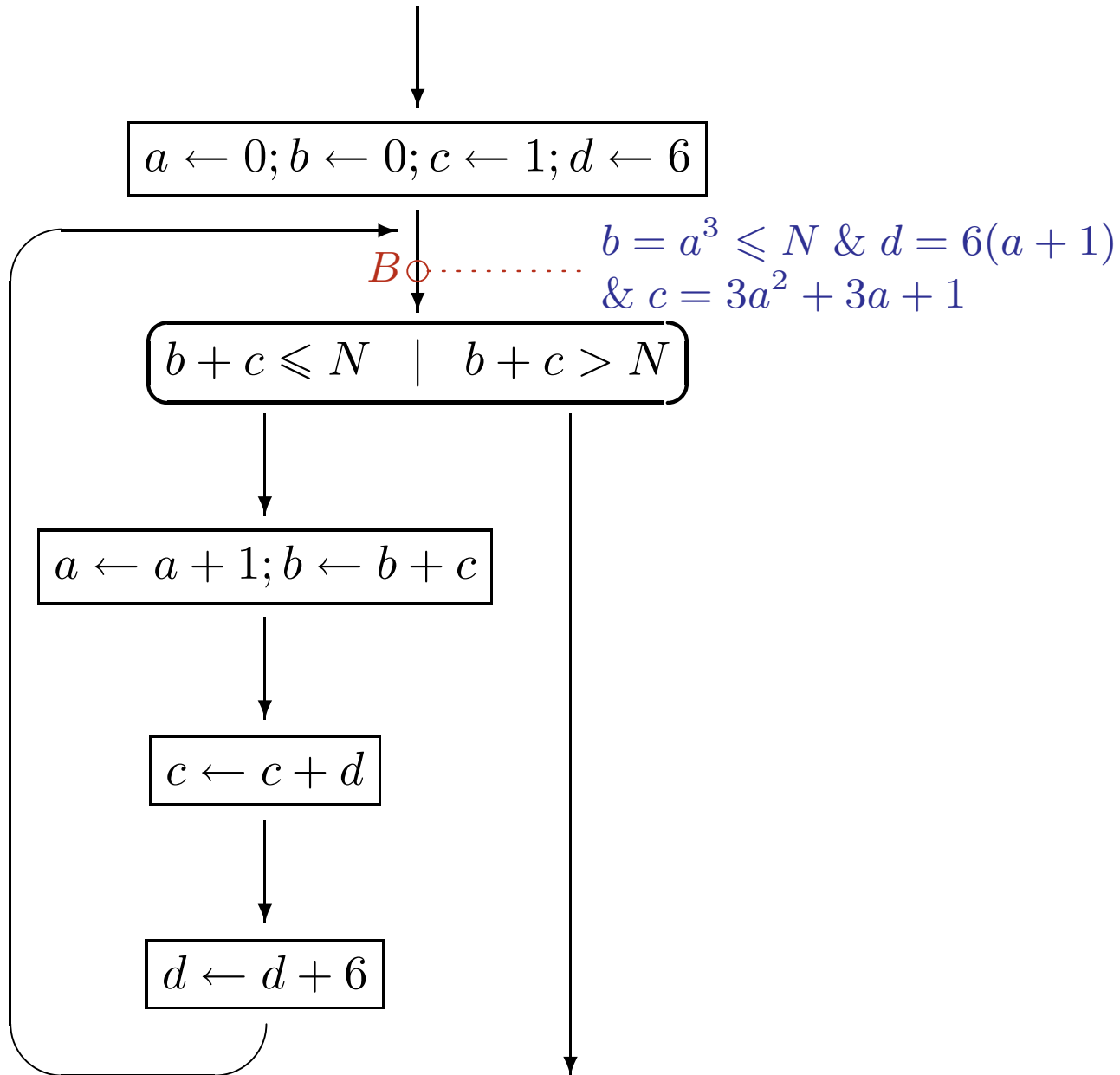
Ręczna symulacja: wartości zmiennych w *B* w kilku pierwszych obrotach pętli:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
0	0	1	6
1	1	7	12
2	8	19	18
3	27	37	24
4	64	61	30
5	125	91	36
6	216	127	42
7	343	169	48

Zależności między wartościami zmiennych:

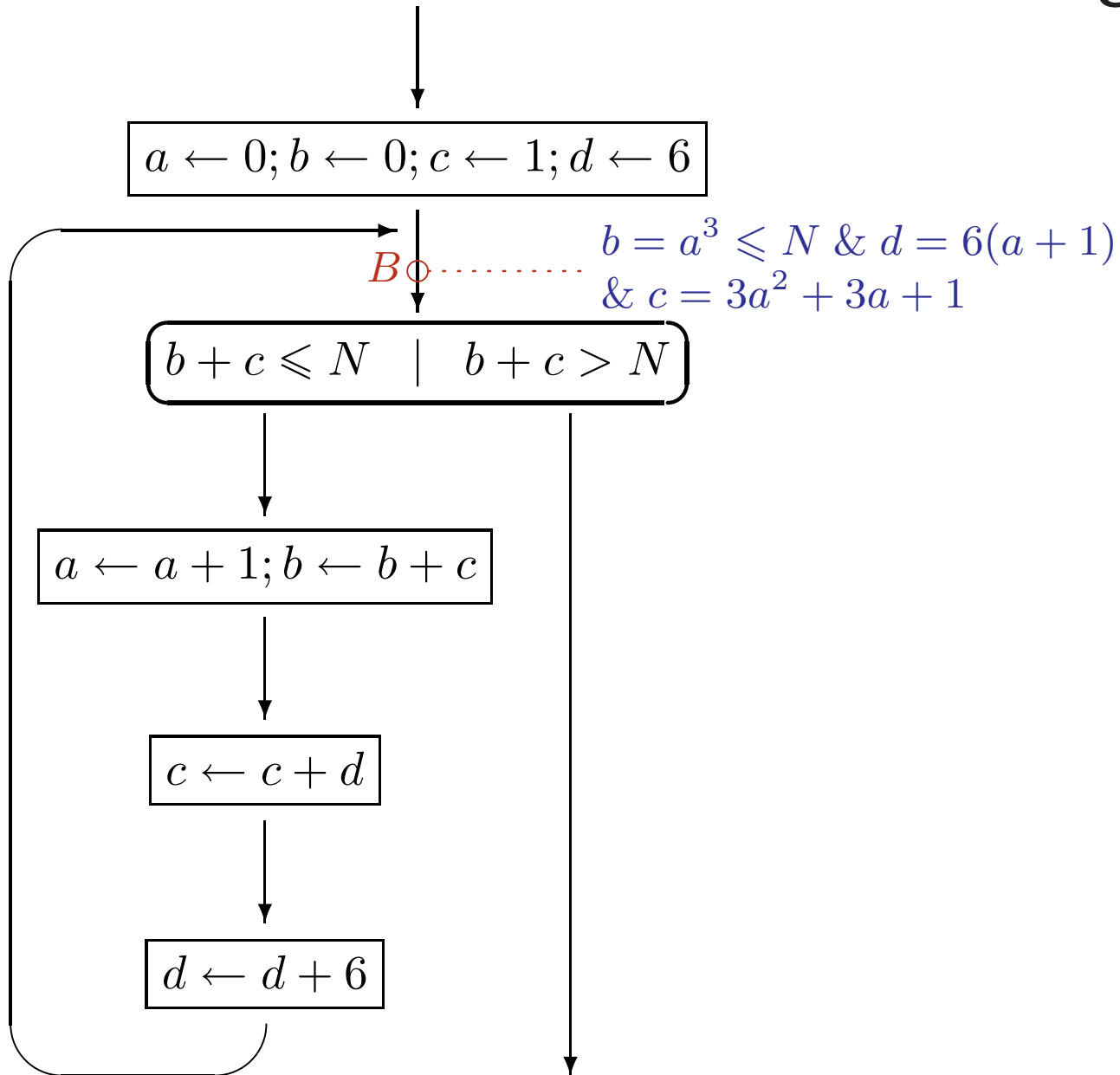
$$\begin{aligned}
 b &= a^3 & d &= 6 \cdot (a + 1) \\
 c &= (a + 1)^3 - a^3 = 3a^2 + 3a + 1
 \end{aligned}$$

Co liczy dany program?

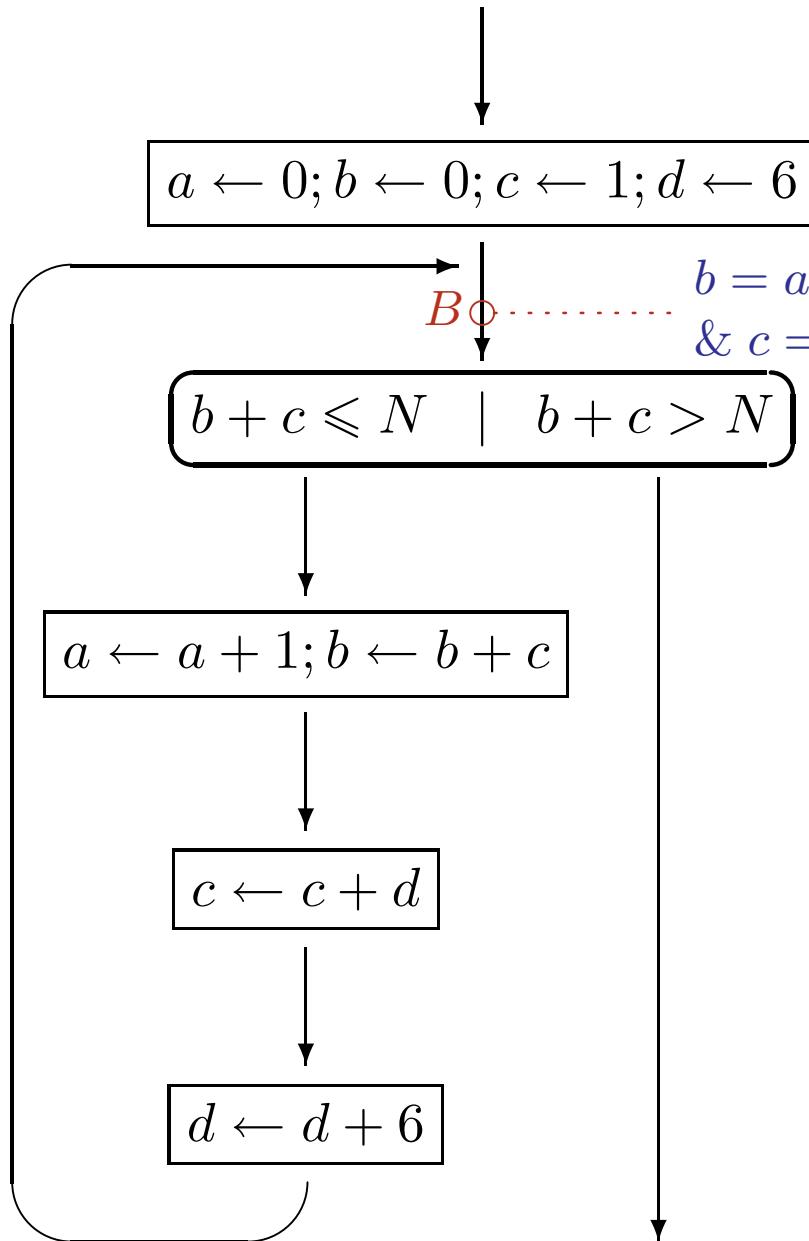


Co liczy dany program?

Czy B jest niezmiennikiem?



Co liczy dany program?



B

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \\ \& \ c = 3a^2 + 3a + 1$$

Czy B jest niezmiennikiem?

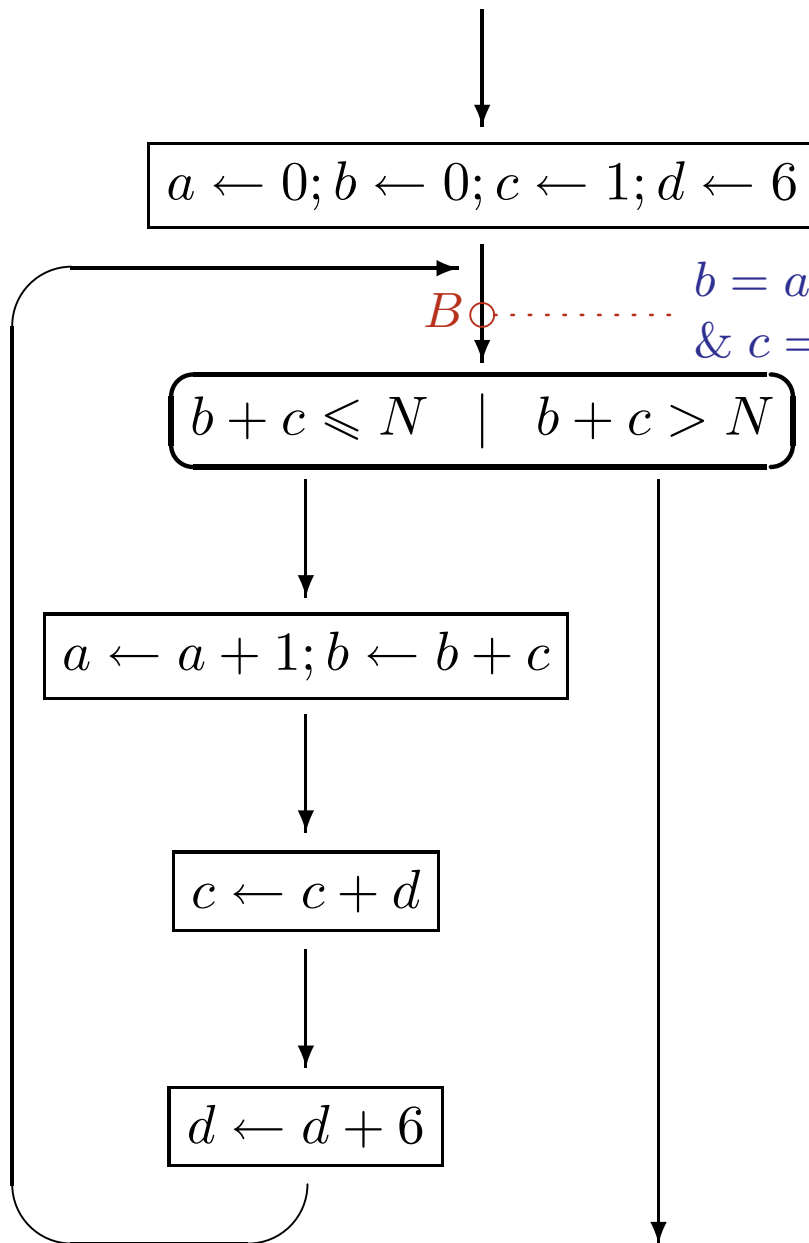
zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

Co liczy dany program?



B

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \\ \& \ c = 3a^2 + 3a + 1$$

Czy B jest niezmiennikiem?

zmienne nieprimowane —

przed przejściem przez ciało pętli

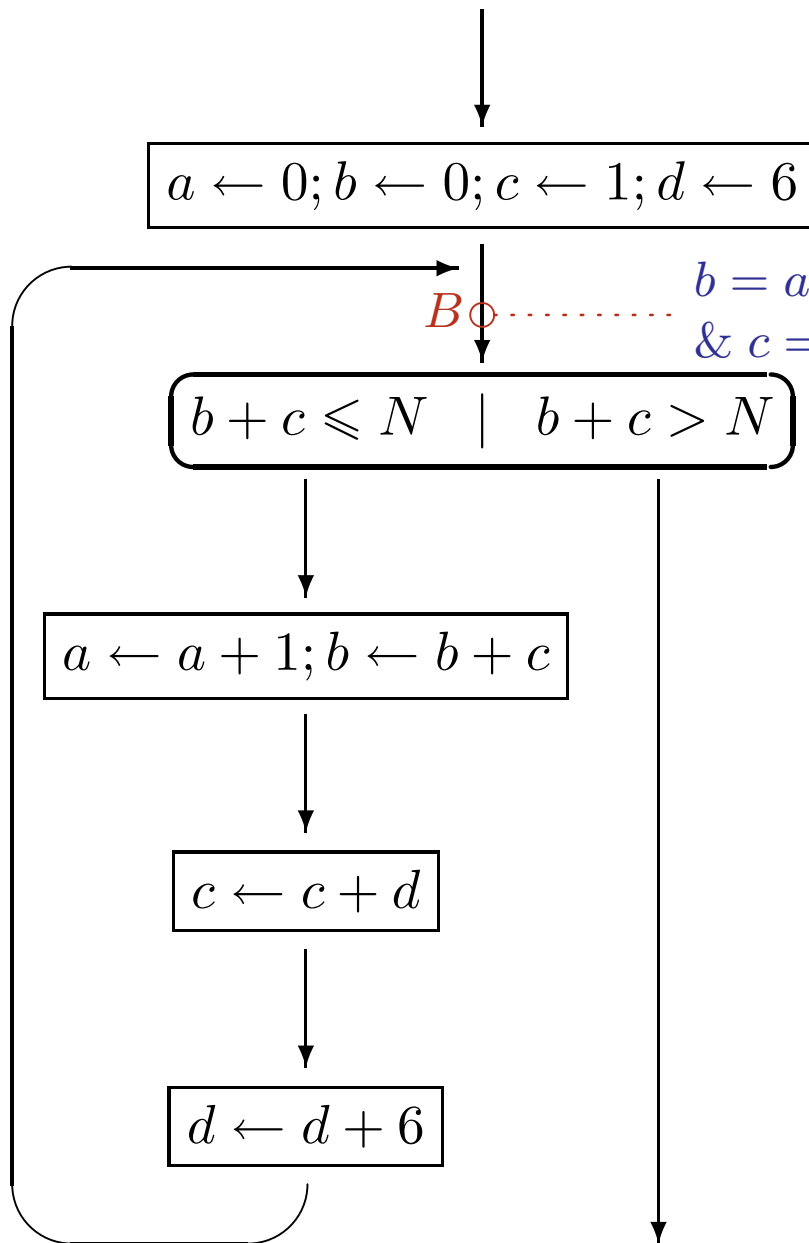
zmienne primowane —

po przejściu przez ciało pętli

Wiemy:

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \ \& \\ c = 3a^2 + 3a + 1 \ \& \ b + c \leq N$$

Co liczy dany program?



B

$b = a^3 \leq N \ \& \ d = 6(a + 1)$ po przejściu przez ciało pętli
 $\& \ c = 3a^2 + 3a + 1$

Czy B jest niezmiennikiem?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

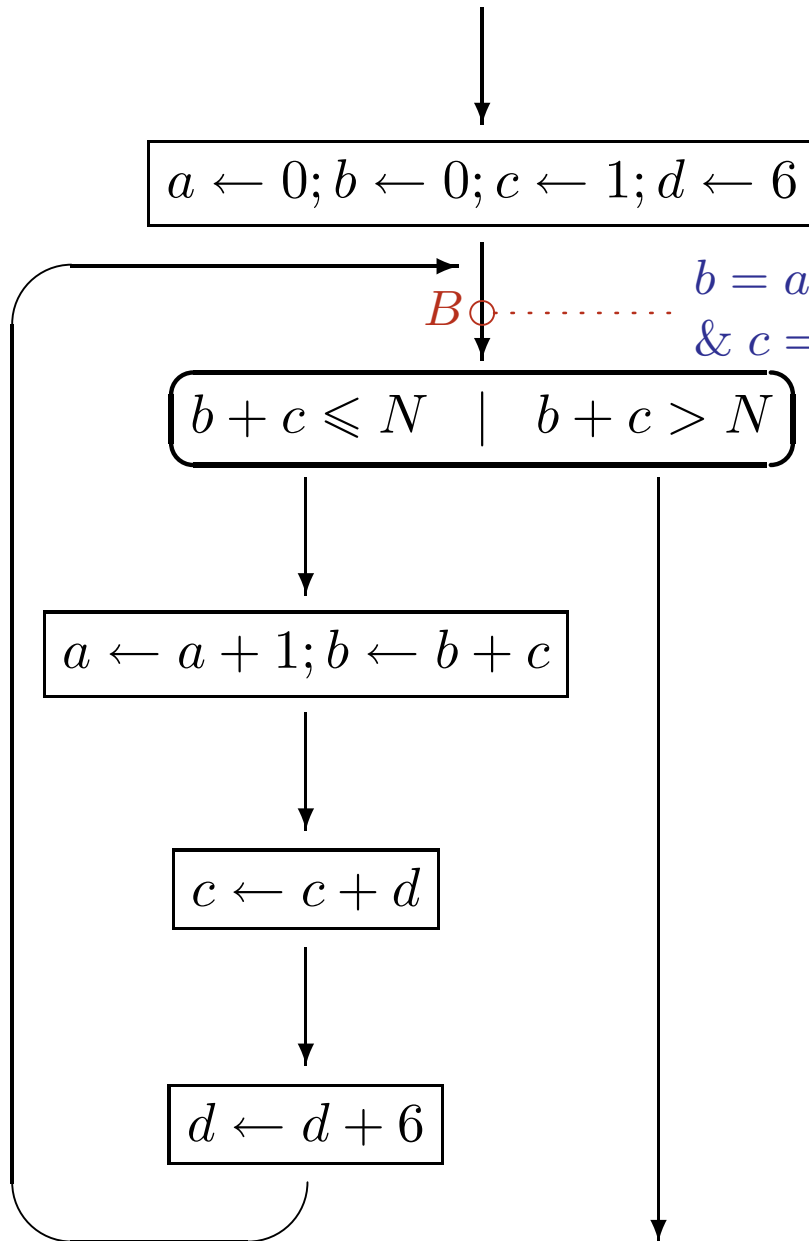
Wiemy:

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \ \& \ c = 3a^2 + 3a + 1 \ \& \ b + c \leq N$$

oraz

$$a' = a + 1$$

Co liczy dany program?



B

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \\ \& \ c = 3a^2 + 3a + 1$$

Czy B jest niezmiennikiem?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

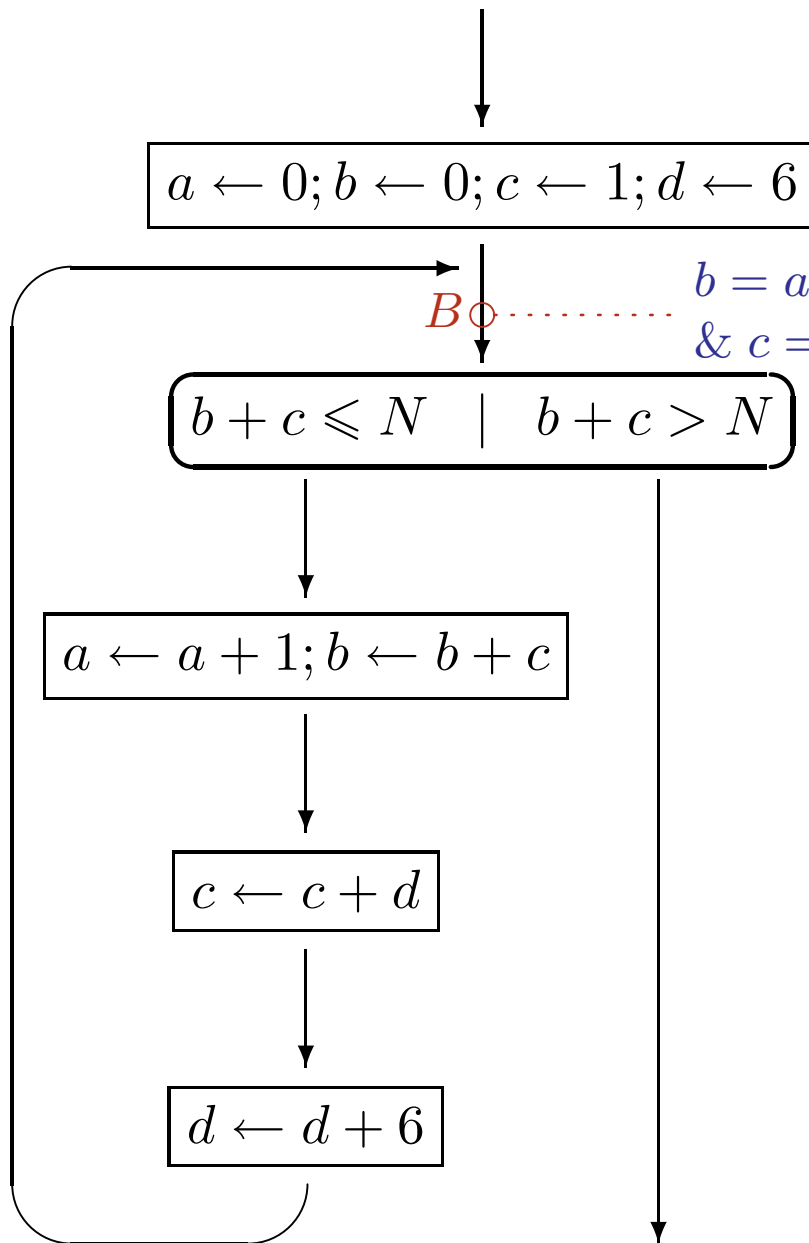
Wiemy:

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \ \& \\ c = 3a^2 + 3a + 1 \ \& \ b + c \leq N$$

oraz

$$a' = a + 1 \\ b' = b + c$$

Co liczy dany program?



B

$b = a^3 \leq N \ \& \ d = 6(a + 1)$ po przejściu przez ciało pętli
 $\& \ c = 3a^2 + 3a + 1$

Czy B jest niezmiennikiem?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

Wiemy:

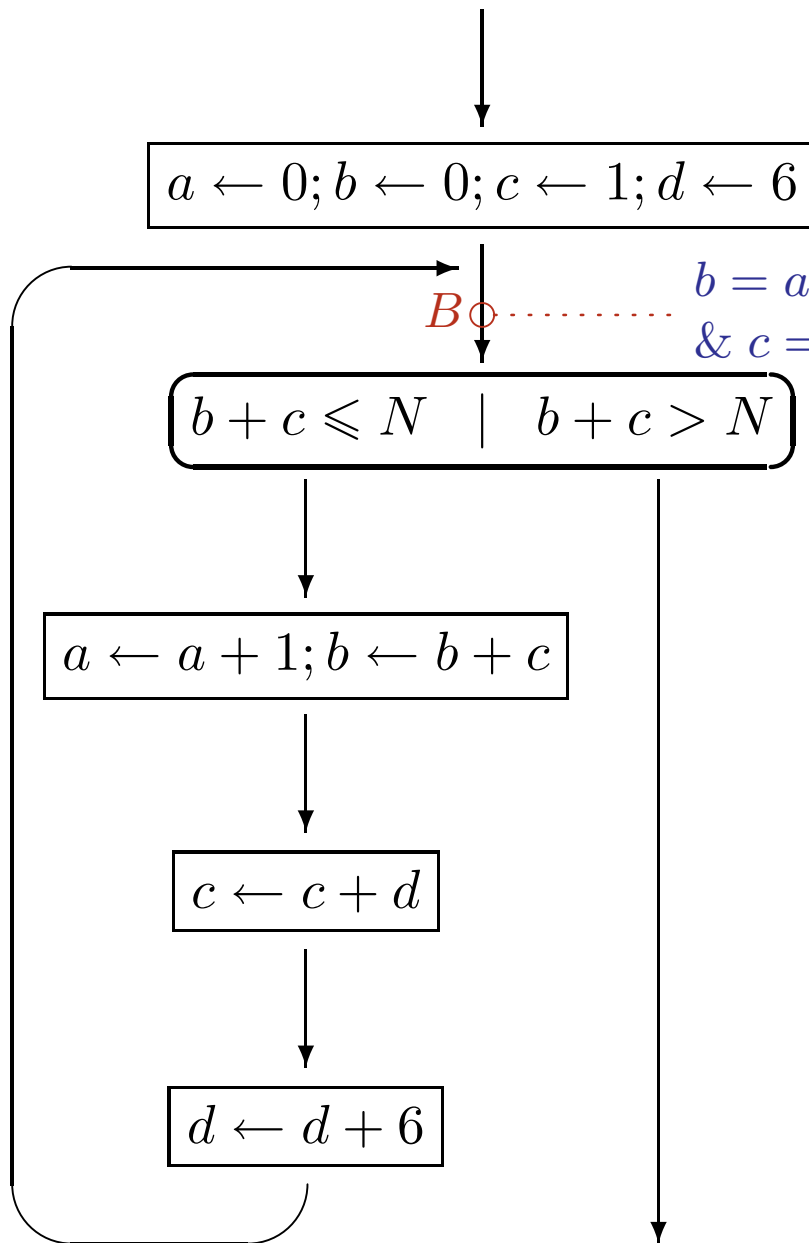
$$b = a^3 \leq N \ \& \ d = 6(a + 1) \ \& \ c = 3a^2 + 3a + 1 \ \& \ b + c \leq N$$

oraz

$$a' = a + 1 \quad c' = c + d$$

$$b' = b + c$$

Co liczy dany program?



$B \circ \dots b = a^3 \leq N \ \& \ d = 6(a + 1)$ po przejściu przez ciało pętli
 $\& \ c = 3a^2 + 3a + 1$

Czy B jest niezmiennikiem?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

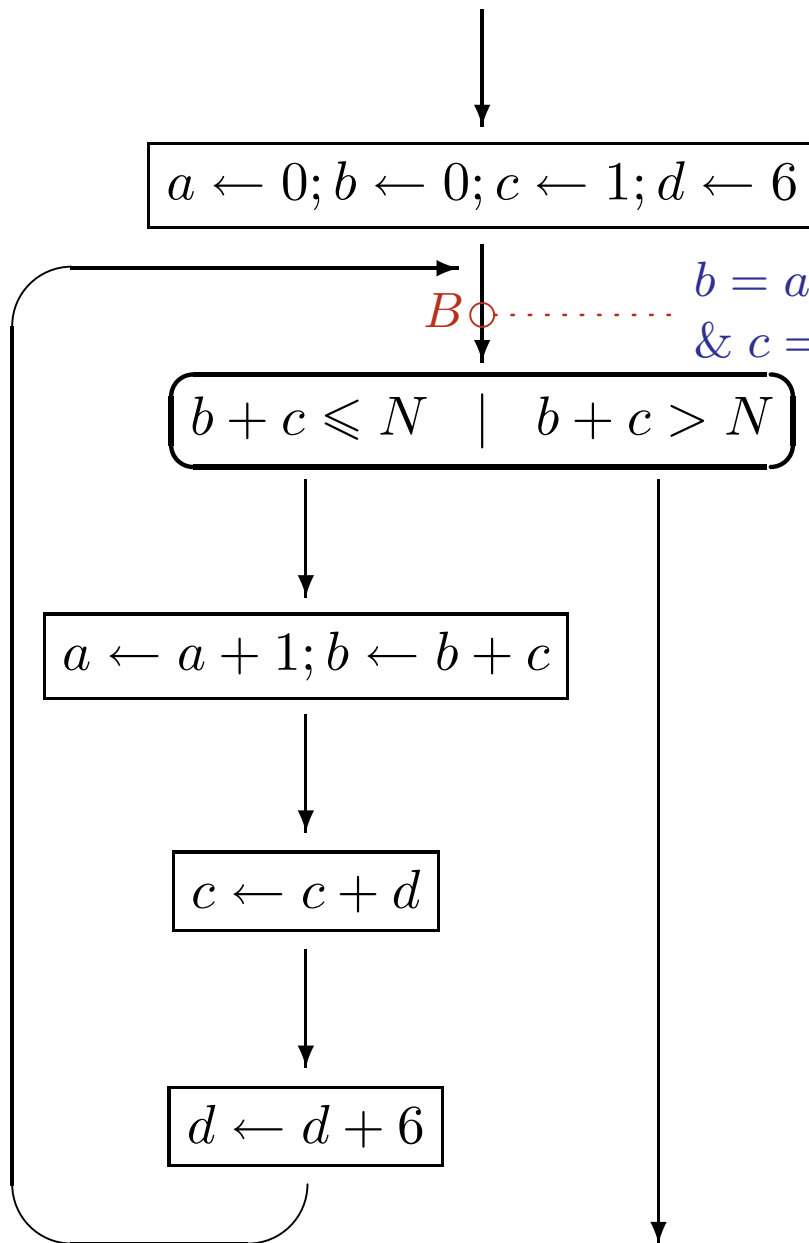
Wiemy:

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \ \& \\ c = 3a^2 + 3a + 1 \ \& \ b + c \leq N$$

oraz

$$\begin{array}{ll} a' = a + 1 & c' = c + d \\ b' = b + c & d' = d + 6 \end{array}$$

Co liczy dany program?



$B \circ \dots b = a^3 \leq N \ \& \ d = 6(a + 1)$ po przejściu przez ciało pętli
 $\& \ c = 3a^2 + 3a + 1$

Czy B jest niezmiennikiem?

zmienne nieprimowane —

przed przejściem przez ciało pętli

zmienne primowane —

po przejściu przez ciało pętli

Wiemy:

$$b = a^3 \leq N \ \& \ d = 6(a + 1) \ \& \\ c = 3a^2 + 3a + 1 \ \& \ b + c \leq N$$

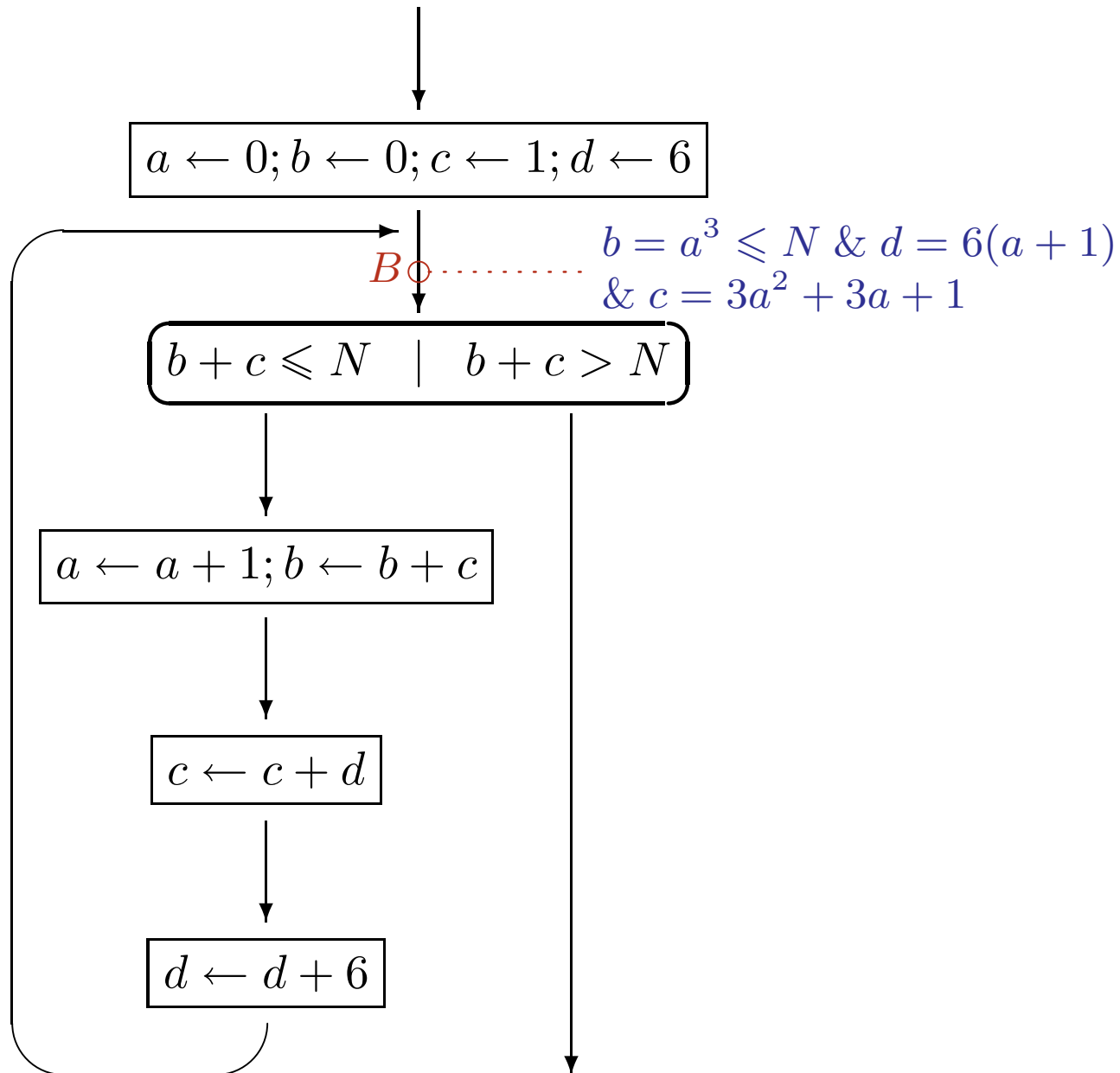
oraz

$$\begin{array}{ll} a' = a + 1 & c' = c + d \\ b' = b + c & d' = d + 6 \end{array}$$

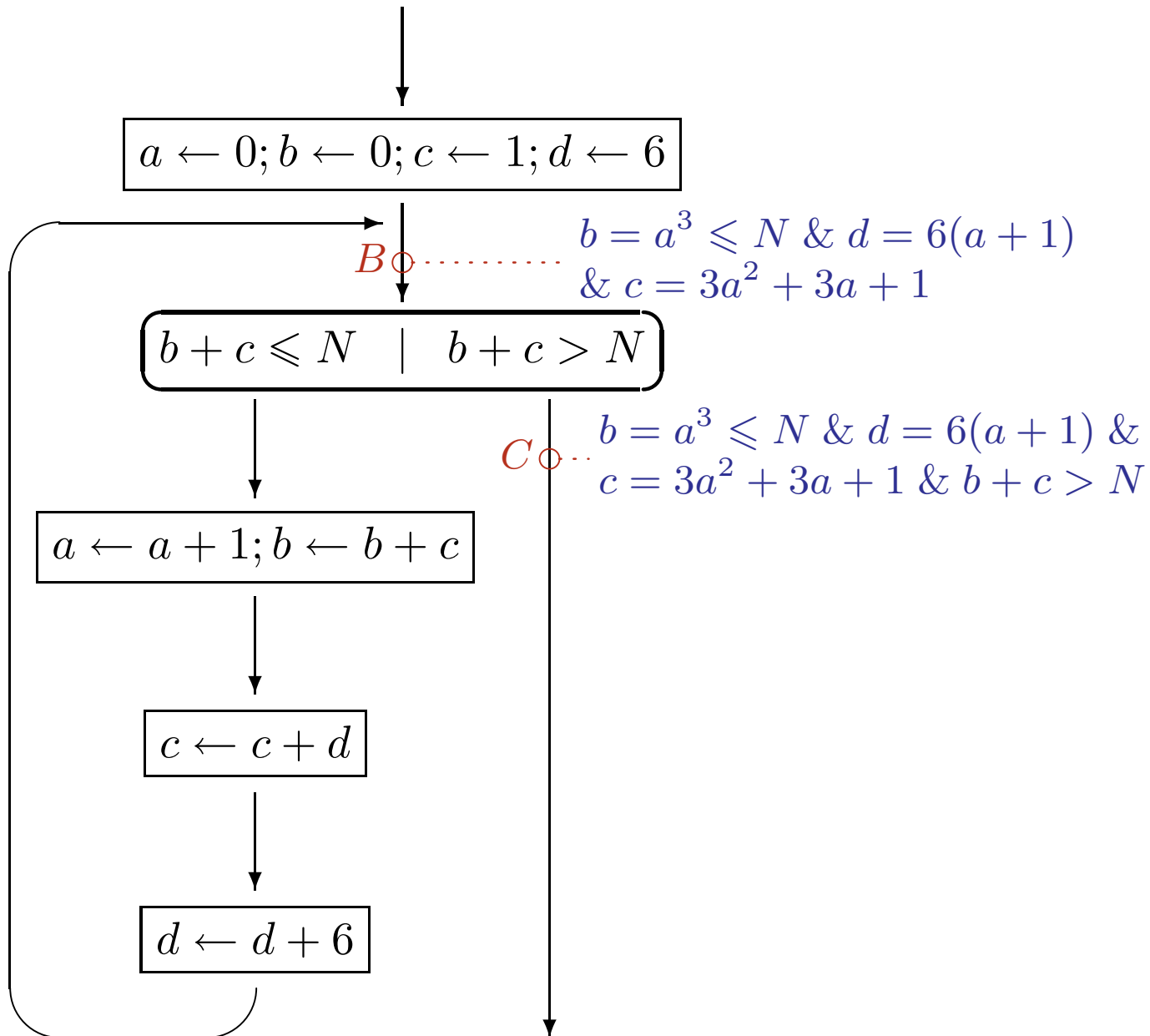
Stąd wnioskujemy:

$$\begin{array}{l} b' \stackrel{?}{=} a'^3 \stackrel{?}{\leq} N \ \& \ d' \stackrel{?}{=} 6(a' + 1) \ \& \\ c' \stackrel{?}{=} 3a'^2 + 3a' + 1 \end{array}$$

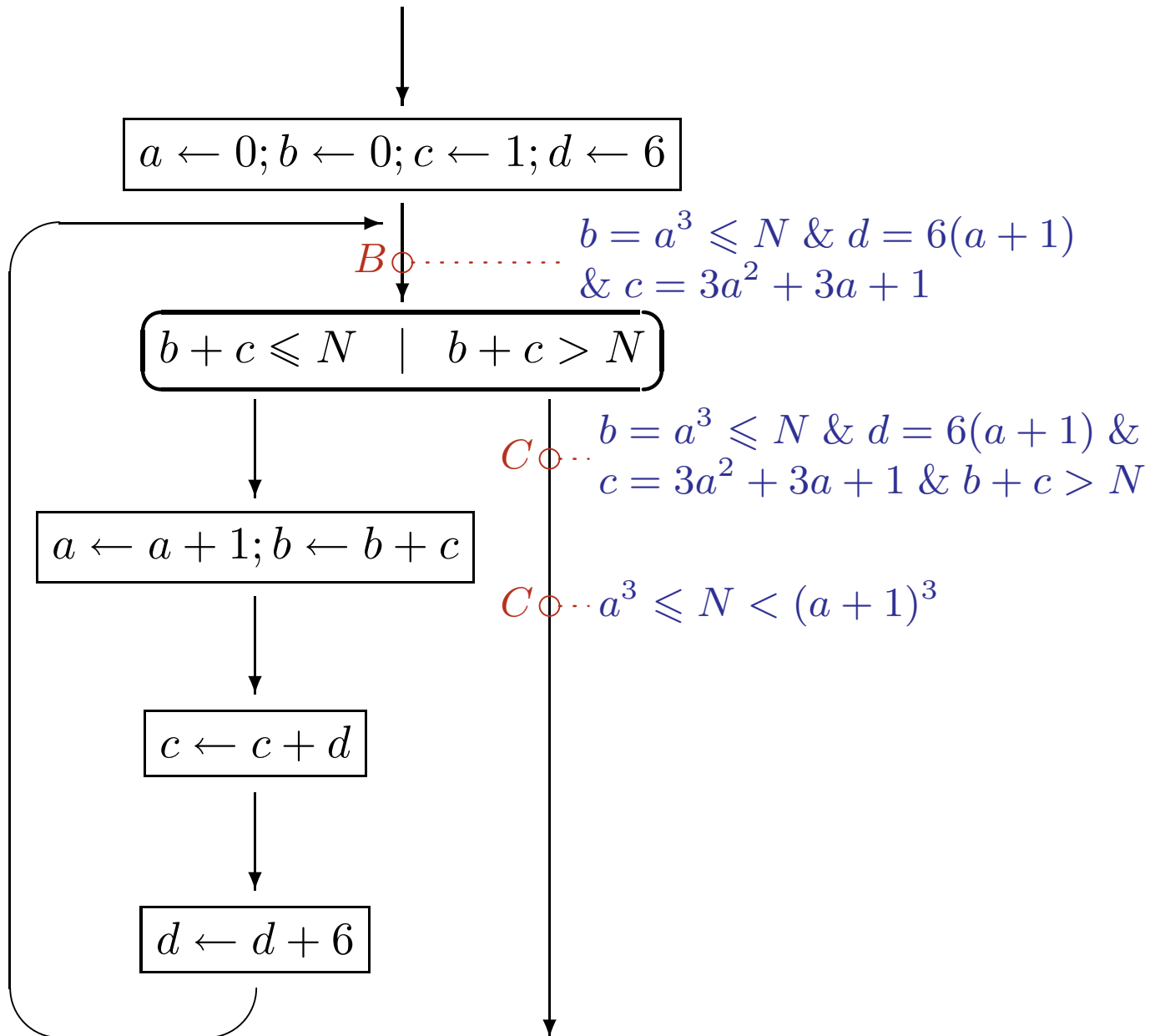
Co liczy dany program?



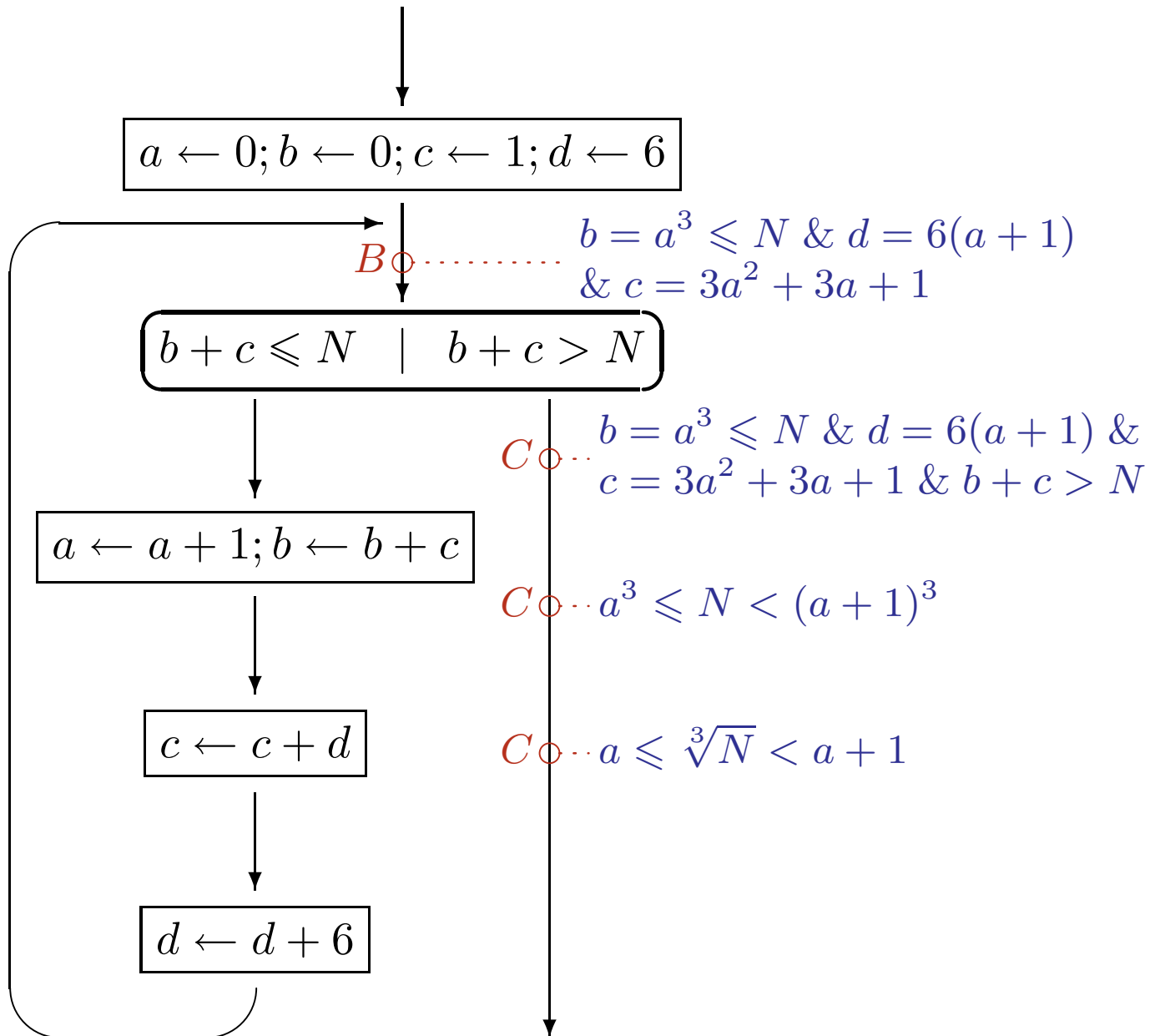
Co liczy dany program?



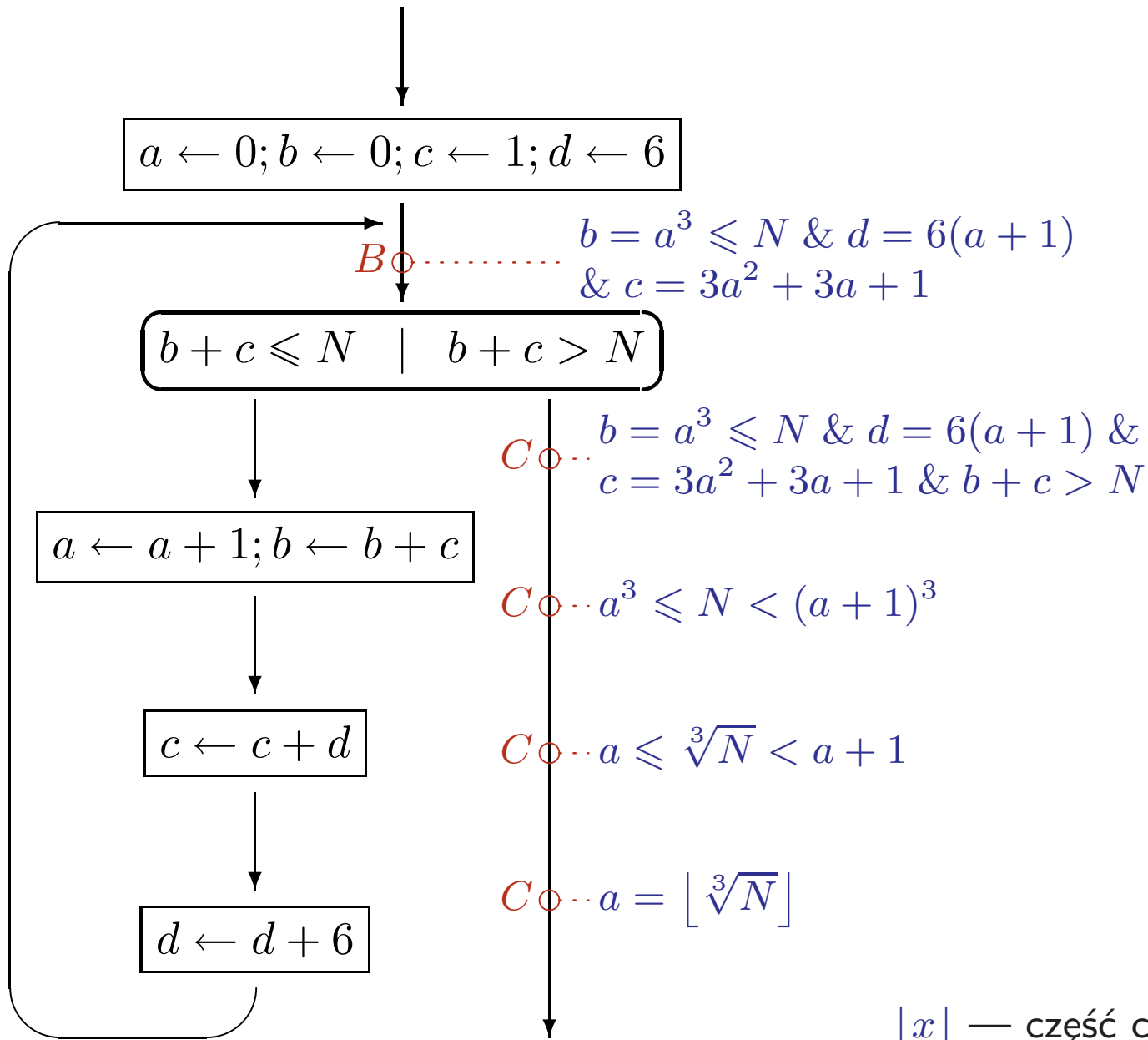
Co liczy dany program?



Co liczy dany program?

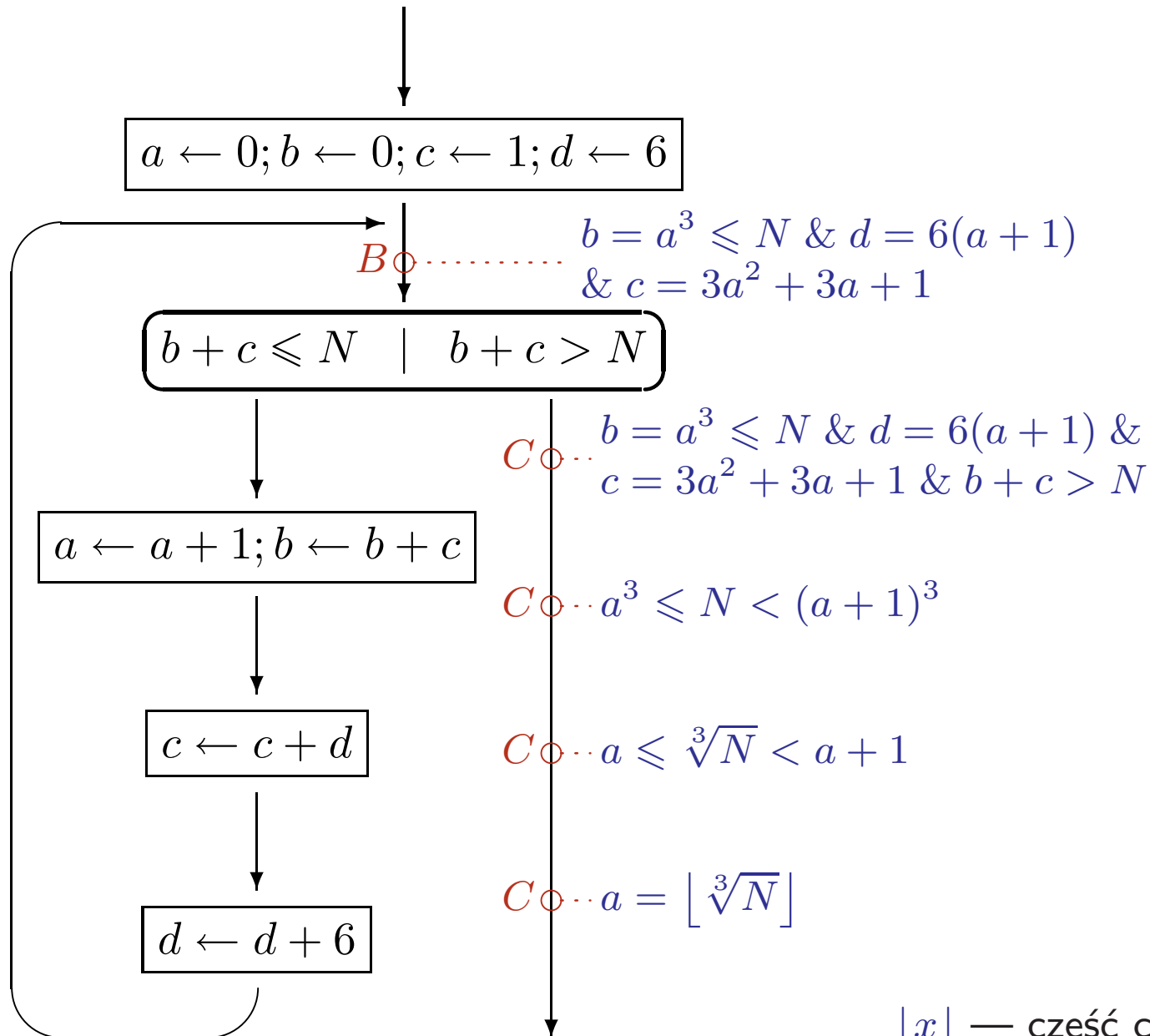


Co liczy dany program?



$\lfloor x \rfloor$ — część całkowita liczby rzeczywistej

Co liczy dany program?



**Program
oblicza**

$$a = \lfloor \sqrt[3]{N} \rfloor$$

$\lfloor x \rfloor$ — część całkowita liczby rzeczywistej

Przykłady programów

```
k=0; p=1; q=2;
```

```
while (p+q <= m) {  
    k=k+1; p=p+q; q=p+p;  
}
```

Przykłady programów

```
/*  $A : m \geq 1$  */
```

```
k=0; p=1; q=2;
```

```
while (p+q <= m) {  
    k=k+1; p=p+q; q=p+p;  
}
```

```
/*  $D : k = \lfloor \log_3 m \rfloor$  */
```


Przykłady programów

```
/*  $A : m \geq 1$  */  
k=0; p=1; q=2;  
/*  $B : p = 3^k \leq m \ \& \ q = 2 \cdot 3^k$  */  
while (p+q <= m) {  
    k=k+1; p=p+q; q=p+p;  
}  
/*  $D : k = \lfloor \log_3 m \rfloor$  */
```

Przykłady programów

```

/* A :  $m \geq 1$  */
k=0; p=1; q=2;
/* B :  $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k$  */
while (p+q <= m) {
    k=k+1; p=p+q; q=p+p;
}
/* D :  $k = \lfloor \log_3 m \rfloor$  */

```

Dowieść (zmienne primowane oznaczają *nowe* wartości)

Przykłady programów

```

/* A :  $m \geq 1$  */
k=0; p=1; q=2;
/* B :  $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k$  */
while (p+q <= m) {
    k=k+1; p=p+q; q=p+p;
}
/* D :  $k = \lfloor \log_3 m \rfloor$  */

```

Dowieść (zmienne primowane oznaczają *nowe* wartości):

$$\begin{aligned}
 1. \quad & m \geq 1 \ \& \ k' = 0 \ \& \ p' = 1 \ \& \ q' = 2 \quad \xRightarrow{?} \\
 & p' = 3^{k'} \leq m \ \& \ q' = 2 \cdot 3^{k'}
 \end{aligned}$$

Przykłady programów

```

/* A :  $m \geq 1$  */
k=0; p=1; q=2;
/* B :  $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k$  */
while (p+q <= m) {
    k=k+1; p=p+q; q=p+p;
}
/* D :  $k = \lfloor \log_3 m \rfloor$  */

```

Dowieść (zmienne primowane oznaczają *nowe* wartości):

1. $m \geq 1 \ \& \ k' = 0 \ \& \ p' = 1 \ \& \ q' = 2 \quad \xRightarrow{?}$
 $p' = 3^{k'} \leq m \ \& \ q' = 2 \cdot 3^{k'}$
2. $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k \ \& \ p + q \leq m \ \& \ k' = k + 1 \ \& \ p' = p + q \ \& \ q' = p' + p \quad \xRightarrow{?}$
 $p' = 3^{k'} \leq m \ \& \ q' = 2 \cdot 3^{k'}$

Przykłady programów

```

/* A :  $m \geq 1$  */
k=0; p=1; q=2;
/* B :  $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k$  */
while (p+q <= m) {
    k=k+1; p=p+q; q=p+p;
}
/* D :  $k = \lfloor \log_3 m \rfloor$  */

```

Dowieść (zmienne primowane oznaczają *nowe* wartości):

1. $m \geq 1 \ \& \ k' = 0 \ \& \ p' = 1 \ \& \ q' = 2 \quad \xRightarrow{?}$
 $p' = 3^{k'} \leq m \ \& \ q' = 2 \cdot 3^{k'}$
2. $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k \ \& \ p + q \leq m \ \&$
 $k' = k + 1 \ \& \ p' = p + q \ \& \ q' = p' + p' \quad \xRightarrow{?}$
 $p' = 3^{k'} \leq m \ \& \ q' = 2 \cdot 3^{k'}$
3. $p = 3^k \leq m \ \& \ q = 2 \cdot 3^k \ \& \ p + q > m \quad \xRightarrow{?} \quad k = \lfloor \log_3 m \rfloor$

Przykład programu: największy wspólny dzielnik

```
/*  $A > 0 \ \& \ B > 0$  */  
a=A; b=B;  
/*  $a > 0 \ \& \ b > 0 \ \& \ \text{nwd}(a, b) = \text{nwd}(A, B)$  */  
while (a != b) {  
    if (a > b) a = a-b;  
    else b = b-a;  
}  
/*  $a = \text{nwd}(A, B)$  */
```

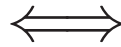
Przykłady programów

Dzielenie całkowite z resztą

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k

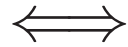


???

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k



$$k \cdot q = n$$

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k

\iff

$$k \cdot q = n$$

Dzielenie z resztą: q jest ilorazem całkowitym a r jest resztą n przez k

\iff

???

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k

$$\iff$$

$$k \cdot q = n$$

Dzielenie z resztą: q jest ilorazem całkowitym a r jest resztą n przez k

$$\iff$$

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

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k

$$\iff k \cdot q = n$$

Dzielenie z resztą: q jest ilorazem całkowitym a r jest resztą n przez k

$$\iff k \cdot q + r = n \ \& \ 0 \leq r < k$$

Dzielenie z resztą łatwo zorganizować przez *wielokrotne odejmowanie*

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k

$$\iff$$

$$k \cdot q = n$$

Dzielenie z resztą: q jest ilorazem całkowitym a r jest resztą n przez k

$$\iff$$

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

Dzielenie z resztą łatwo zorganizować przez *wielokrotne odejmowanie*:

- znaleźć jakiegokolwiek q i r spełniające $k \cdot q + r = n \ \& \ 0 \leq r$

Przykłady programów

Dzielenie całkowite z resztą:

Dzielenie dokładne: q jest ilorazem n przez k

$$\iff k \cdot q = n$$

Dzielenie z resztą: q jest ilorazem całkowitym a r jest resztą n przez k

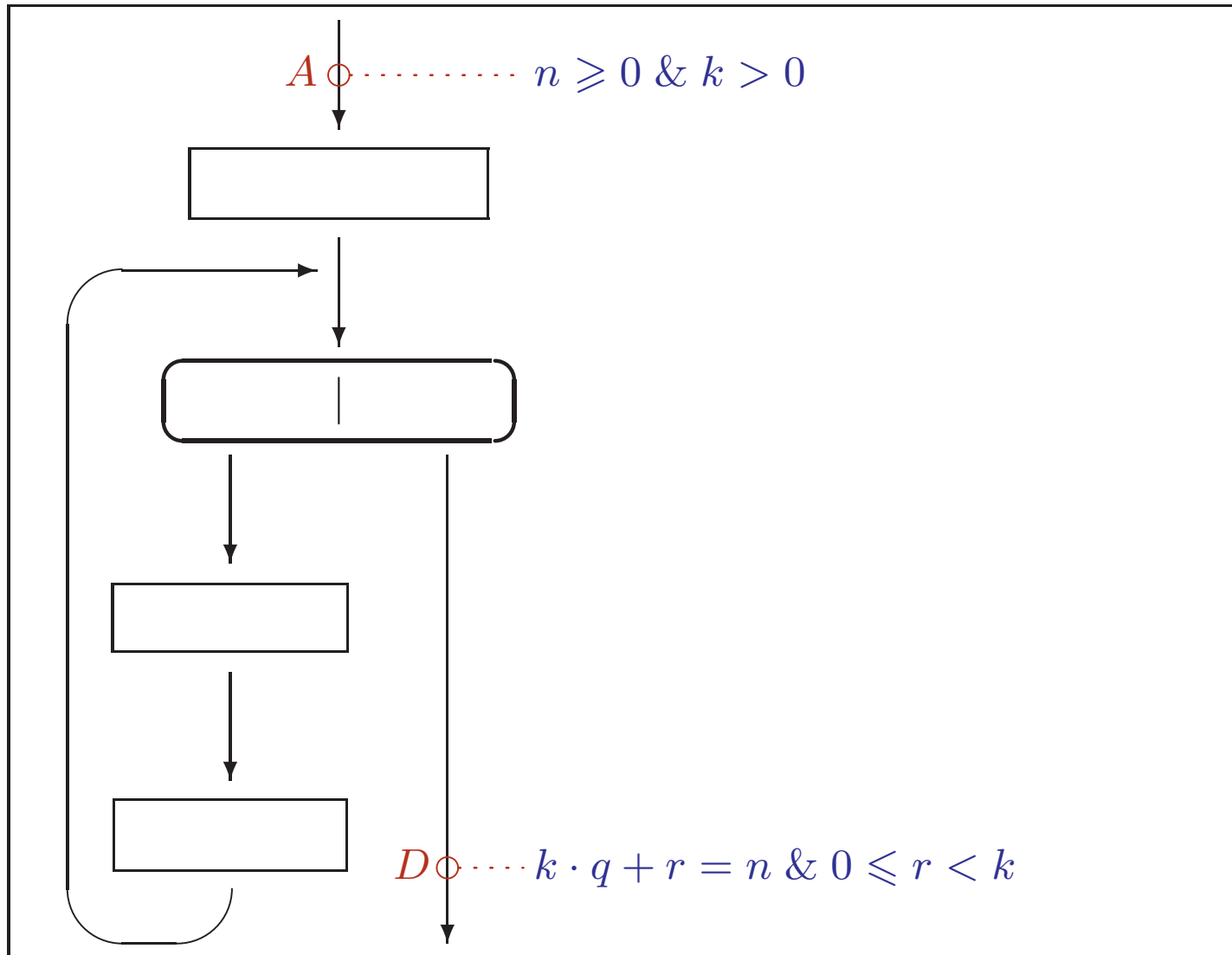
$$\iff k \cdot q + r = n \ \& \ 0 \leq r < k$$

Dzielenie z resztą łatwo zorganizować przez *wielokrotne odejmowanie*:

- znaleźć jakiegolwiek q i r spełniające $k \cdot q + r = n \ \& \ 0 \leq r$,
- tak długo odejmować dzielnik k od r , aż dodatkowo będzie spełnione $r < k$ (zakładamy $k > 0$).

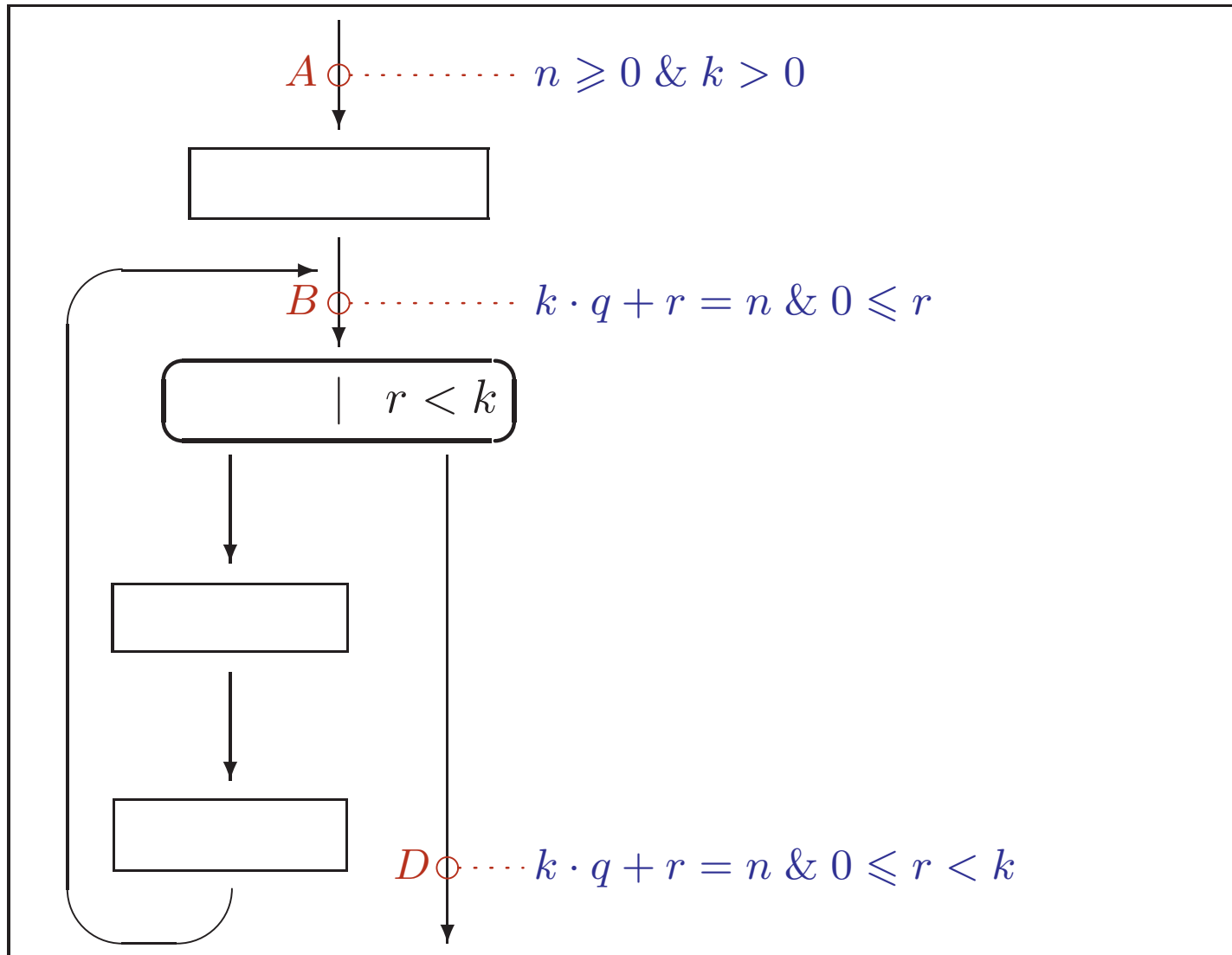
Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



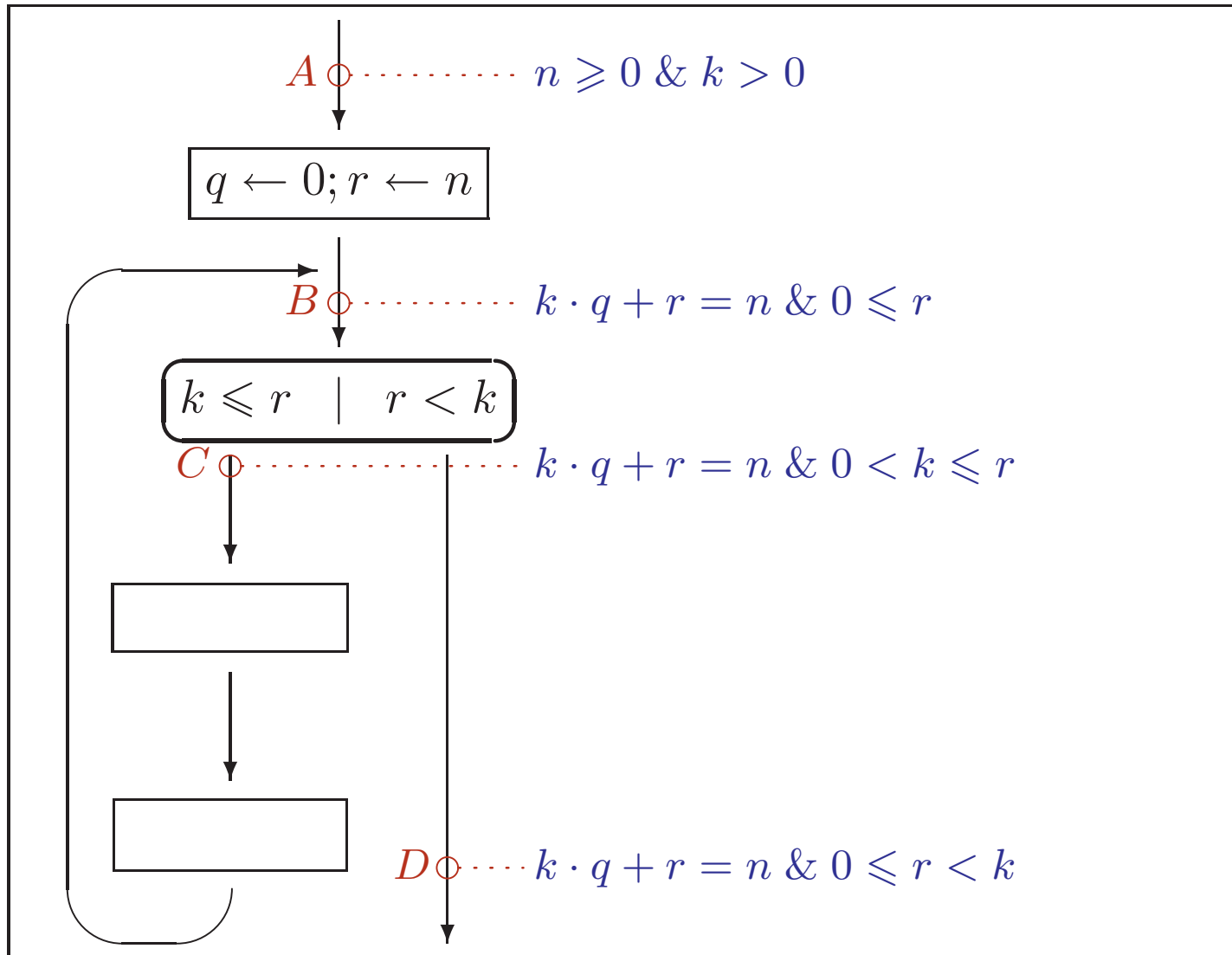
Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



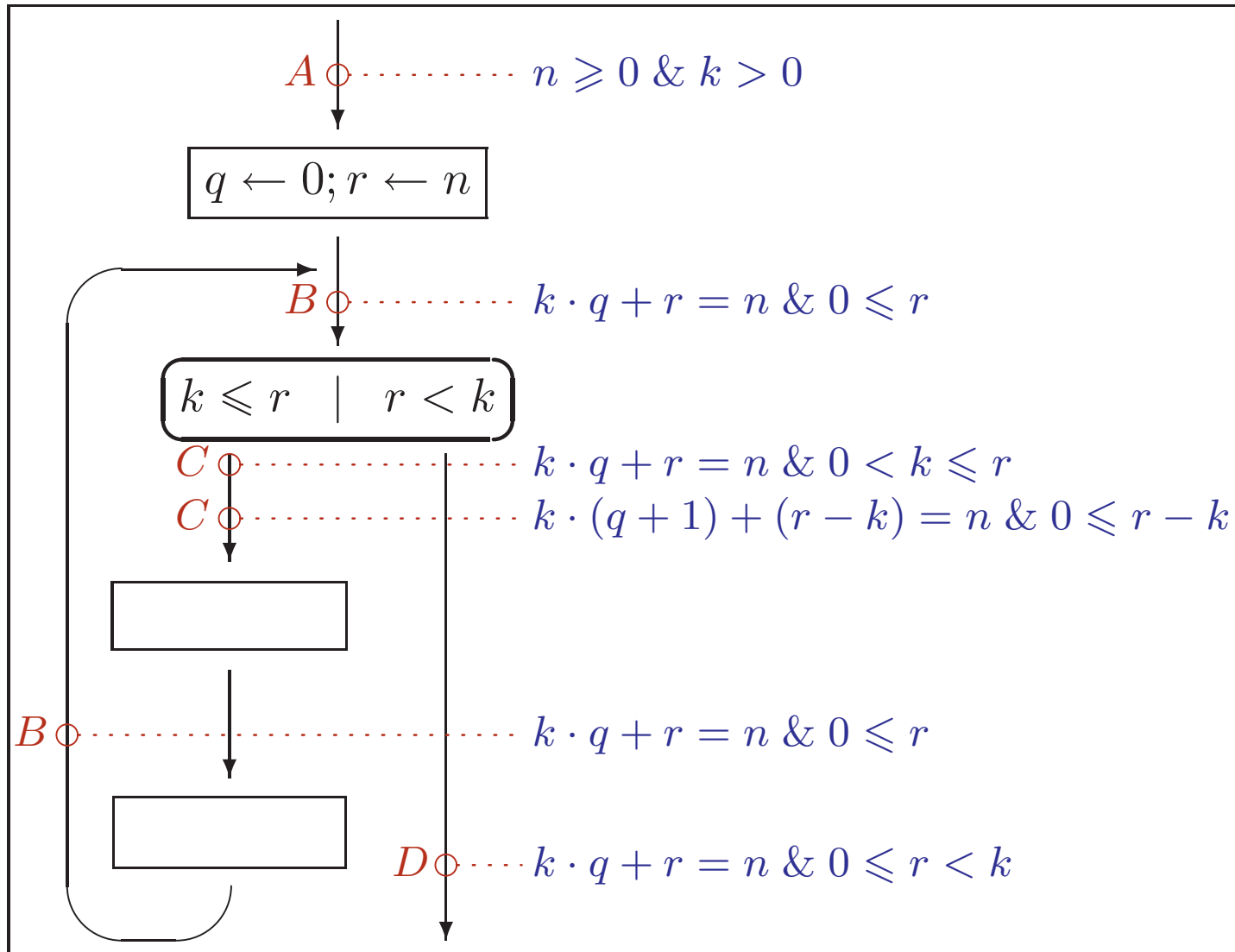
Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



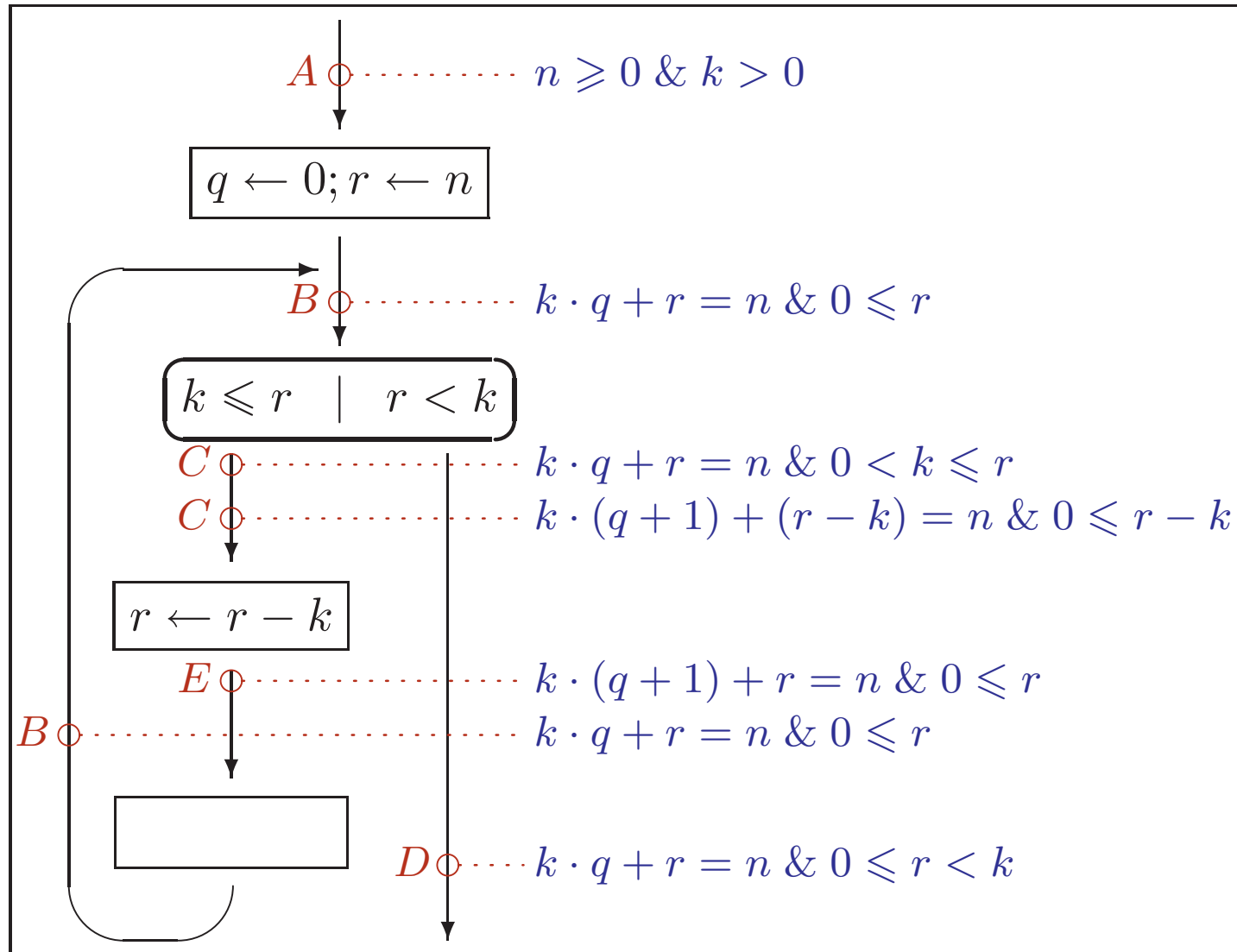
Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



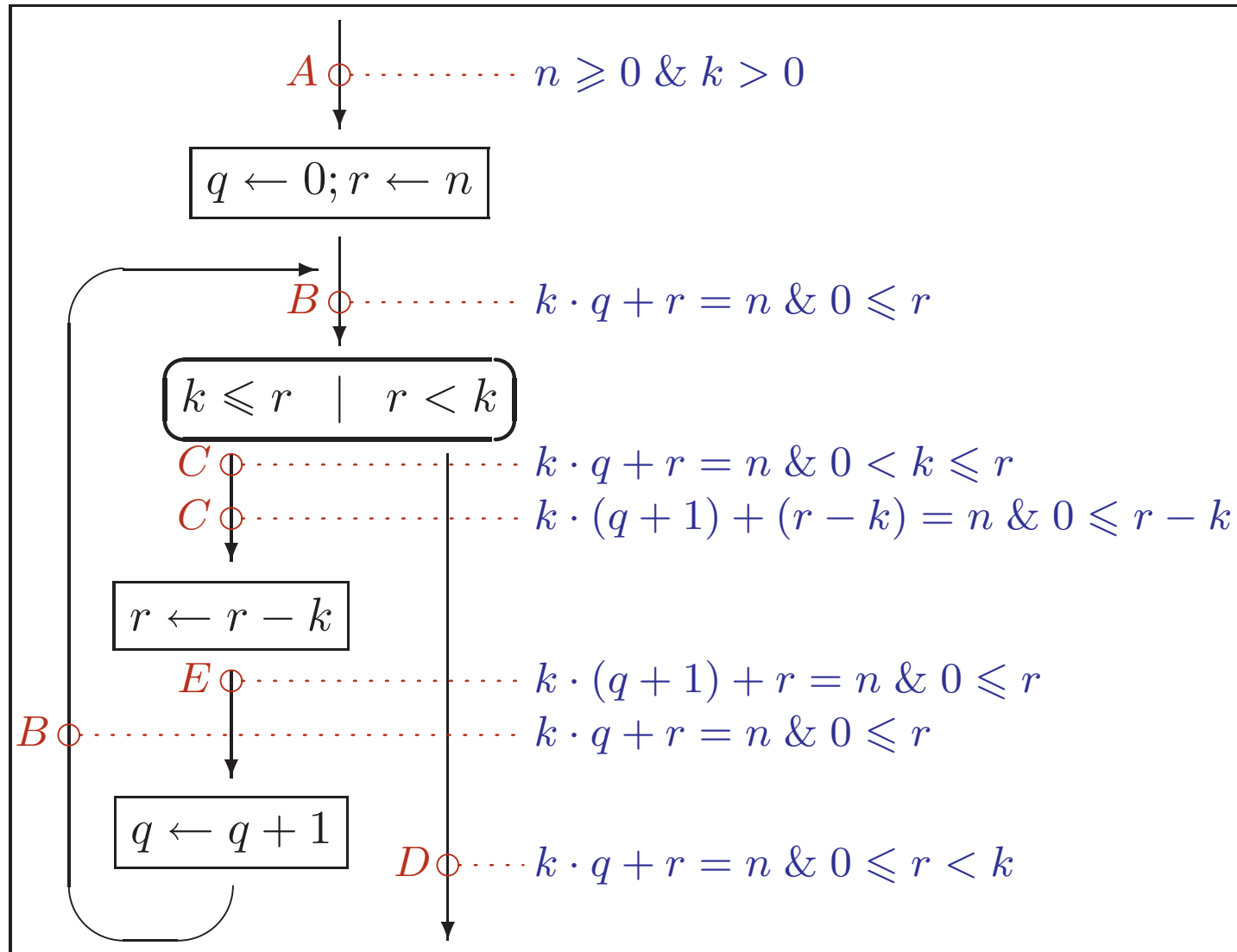
Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



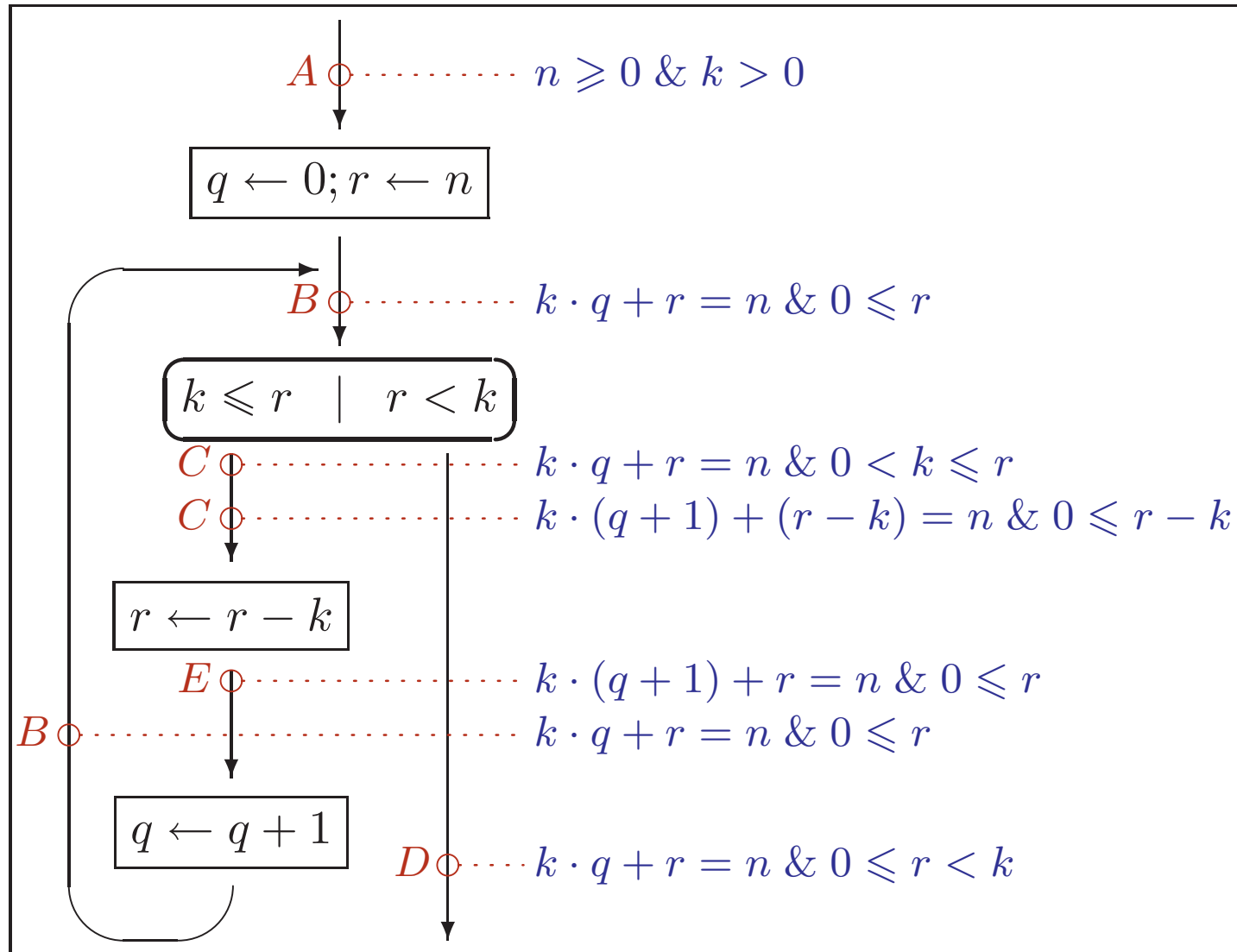
Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



Przykłady programów

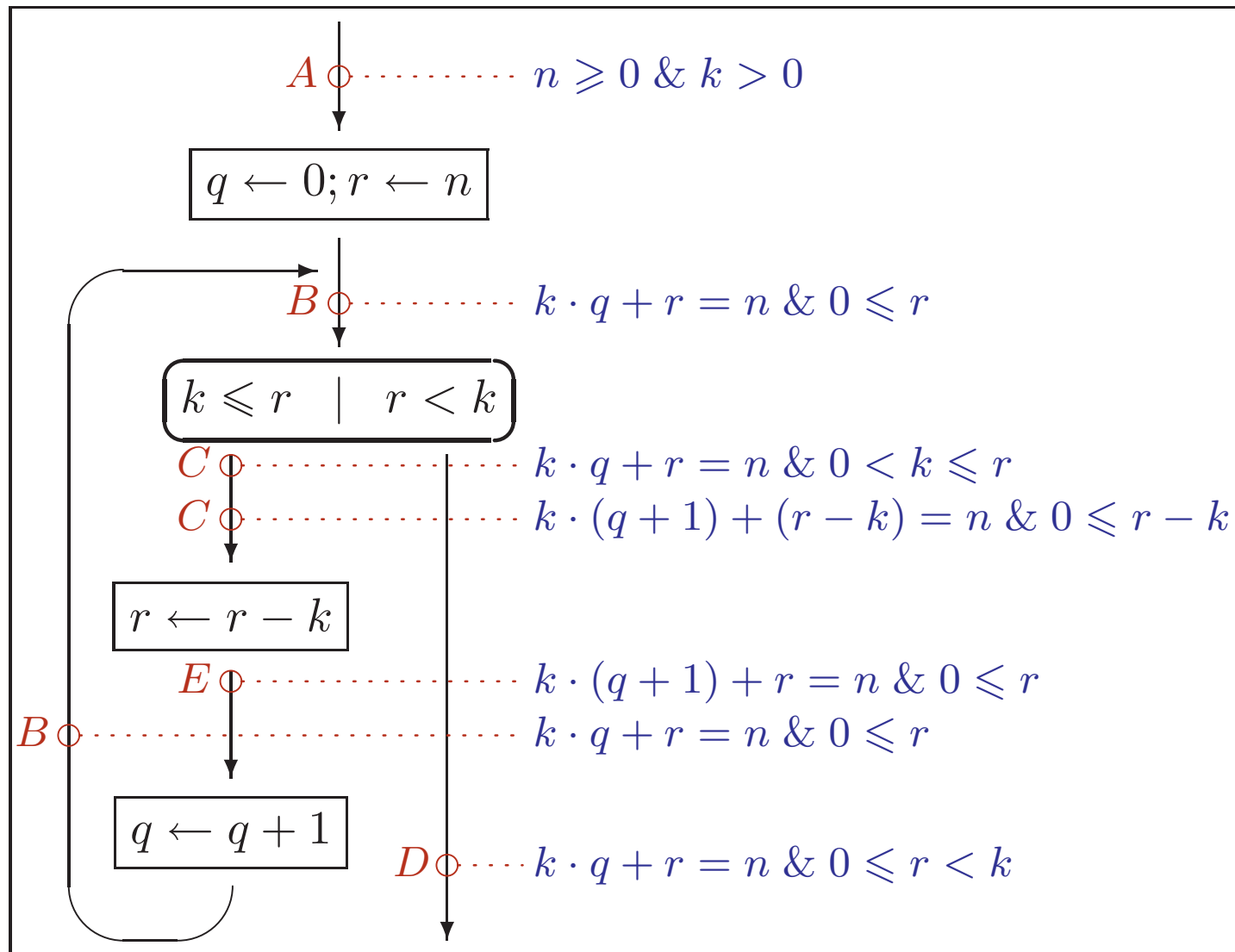
Dzielenie całkowite z resztą (zał.: $k > 0$):



1. Przejście od A do B

Przykłady programów

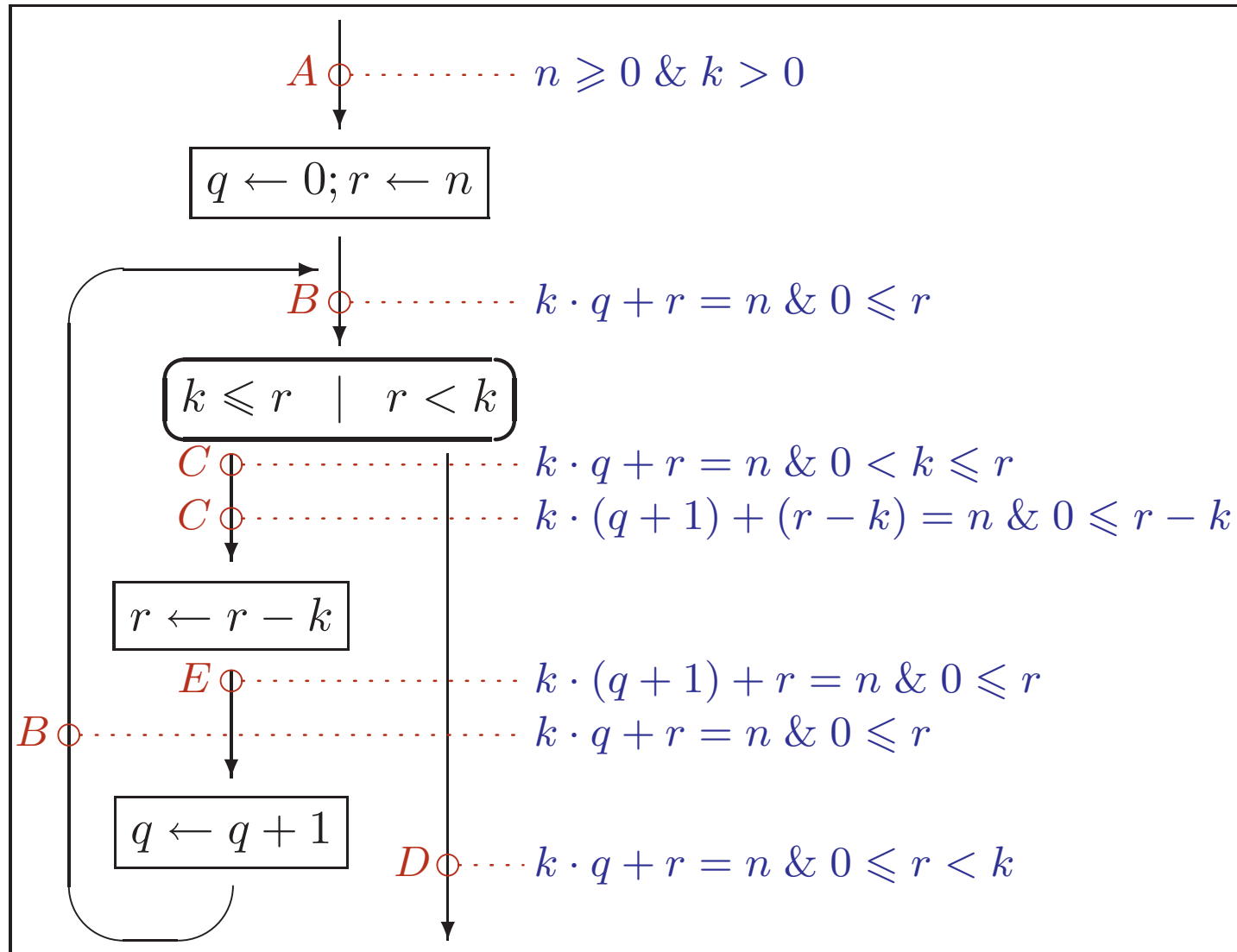
Dzielenie całkowite z resztą (zał.: $k > 0$):



1. Przejście od A do B
2. Wynikanie $B \ \& \ k \leq r \Rightarrow C$

Przykłady programów

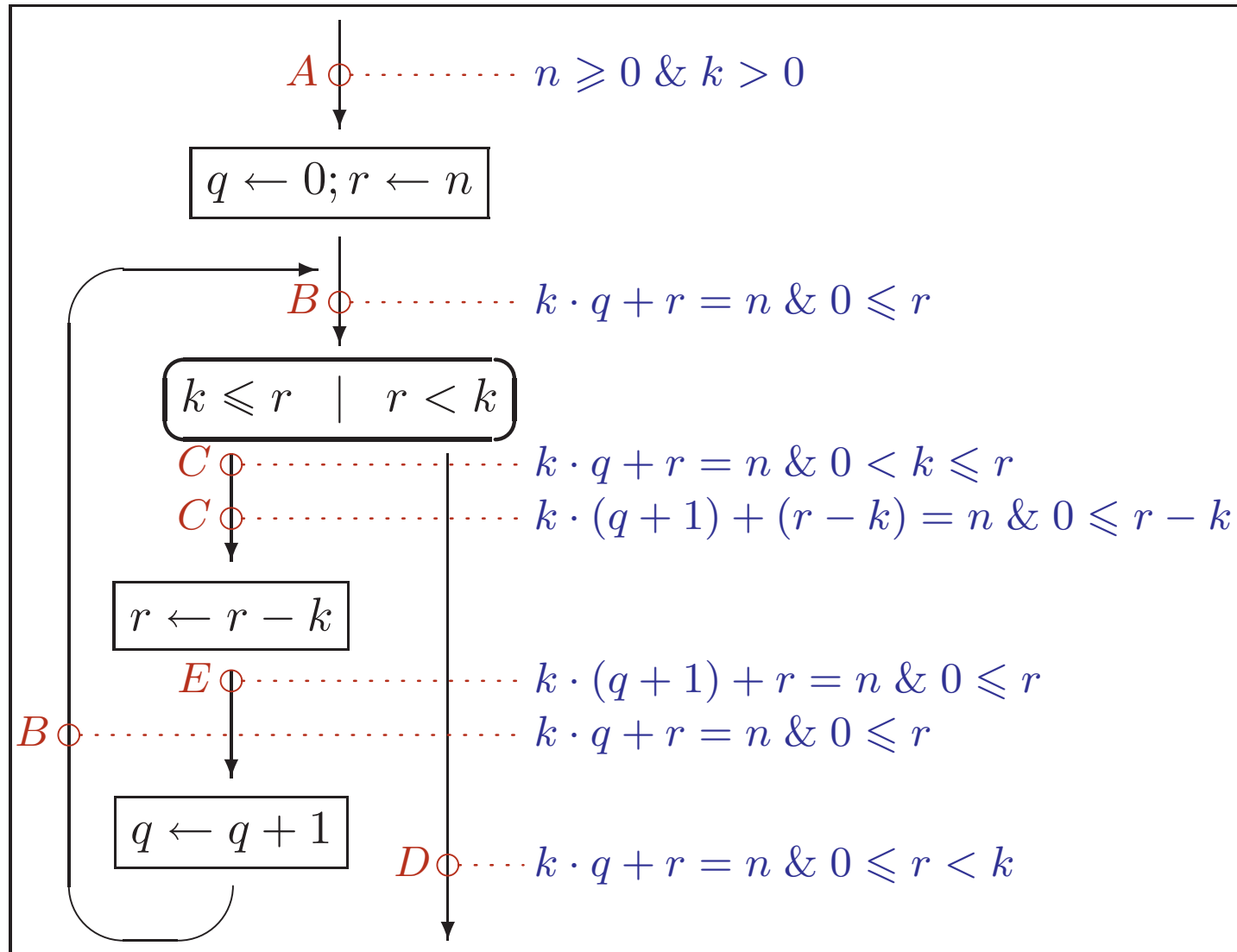
Dzielenie całkowite z resztą (zał.: $k > 0$):



1. Przejście od A do B
2. Wynikanie $B \ \& \ k \leq r \Rightarrow C$
3. Przejście od C do B (przez E)

Przykłady programów

Dzielenie całkowite z resztą (zał.: $k > 0$):



1. Przejście od A do B
2. Wynikanie $B \ \& \ k \leq r \Rightarrow C$
3. Przejście od C do B (przez E)
4. Wynikanie $B \ \& \ r < k \Rightarrow D$

Przykłady programów

Dzielenie całkowite z resztą:

```
/*  $n \geq 0$  &  $k > 0$  */
```

```
q = 0; r = n;
```

```
/*  $n = k \cdot q + r$  &  $0 \leq r$  */
```

```
while (r >= k) {
```

```
    q = q+1; r = r-k;
```

```
}
```

```
/*  $n = k \cdot q + r$  &  $0 \leq r < k$  */
```

Przykłady programów

Dzielenie całkowite z resztą:

```

/*  $n \geq 0 \ \& \ k > 0$  */
q = 0; r = n;
/*  $n = k \cdot q + r \ \& \ 0 \leq r$  */
while (r >= k) {
    q = q+1; r = r-k;
}
/*  $n = k \cdot q + r \ \& \ 0 \leq r < k$  */

```

Dla poprawności należy sprawdzić:

- jeśli $n \geq 0 \ \& \ k > 0$, to po wykonaniu $q = 0; r = n;$
jest $n = k \cdot q' + r' \ \& \ 0 \leq r'$

Przykłady programów

Dzielenie całkowite z resztą:

```

/*  $n \geq 0 \ \& \ k > 0$  */
q = 0; r = n;
/*  $n = k \cdot q + r \ \& \ 0 \leq r$  */
while (r >= k) {
    q = q+1; r = r-k;
}
/*  $n = k \cdot q + r \ \& \ 0 \leq r < k$  */

```

Dla poprawności należy sprawdzić:

1. jeśli $n \geq 0 \ \& \ k > 0$, to po wykonaniu $q = 0; r = n;$ jest $n = k \cdot q' + r' \ \& \ 0 \leq r'$
2. jeśli $n = k \cdot q + r \ \& \ 0 \leq r \ \& \ r \geq k$, to po wykonaniu $q = q+1; r = r-k;$ jest $n = k \cdot q' + r' \ \& \ 0 \leq r'$

Przykłady programów

Dzielenie całkowite z resztą:

```

/*  $n \geq 0 \ \& \ k > 0$  */
q = 0; r = n;
/*  $n = k \cdot q + r \ \& \ 0 \leq r$  */
while (r >= k) {
    q = q+1; r = r-k;
}
/*  $n = k \cdot q + r \ \& \ 0 \leq r < k$  */

```

Dla poprawności należy sprawdzić:

- jeśli $n \geq 0 \ \& \ k > 0$, to po wykonaniu $q = 0; r = n;$
jest $n = k \cdot q' + r' \ \& \ 0 \leq r'$
- jeśli $n = k \cdot q + r \ \& \ 0 \leq r \ \& \ r \geq k$, to po wykonaniu
 $q = q+1; r = r-k;$ jest $n = k \cdot q' + r' \ \& \ 0 \leq r'$
- jeśli $n = k \cdot q + r \ \& \ 0 \leq r \ \& \ r < k$,
to $n = k \cdot q' + r' \ \& \ 0 \leq r' < k$

Jak pisać programy z dowodami?

- Z tego, że asercje rozmieszczone w programie są niezmiennikami, wnioskujemy o jego poprawności.

Jak pisać programy z dowodami?

- Z tego, że asercje rozmieszczone w programie są niezmiennikami, wnioskujemy o jego poprawności.
- Konstruując program najpierw piszemy odpowiednie asercje; potem tak dobieramy komendy, żeby te asercje były niezmiennicze.

Jak pisać programy z dowodami?

- Z tego, że asercje rozmieszczone w programie są niezmiennikami, wnioskujemy o jego poprawności.
- Konstruując program najpierw piszemy odpowiednie asercje; potem tak dobieramy komendy, żeby te asercje były niezmiennicze.
- Program powstaje więc razem z dowodem swojej poprawności; dowód zawsze „o pół kroku wcześniej”.

Jak pisać programy z dowodami?

- Z tego, że asercje rozmieszczone w programie są niezmiennikami, wnioskujemy o jego poprawności.
- Konstruując program najpierw piszemy odpowiednie asercje; potem tak dobieramy komendy, żeby te asercje były niezmiennicze.
- Program powstaje więc razem z dowodem swojej poprawności; dowód zawsze „o pół kroku wcześniej”.
- Asercje mogą również pomóc w kontrolowanym transformowaniu programu do innej postaci.