

Pascal — Składnia i semantyka

Składnia przedstawiona w postaci reguł gramatycznych.

Semantyka operacyjna instrukcji podana w postaci schematów blokowych.

Instrukcje

$$\langle \text{Inst} \rangle ::= \langle \text{Inst_przyp} \rangle \mid$$
$$\langle \text{Inst_bloku} \rangle \mid$$
$$\langle \text{Inst_if_then_else} \rangle \mid$$
$$\langle \text{Inst_if_then} \rangle \mid$$
$$\langle \text{Inst_while} \rangle \mid$$
$$\langle \text{Inst_repeat} \rangle \mid$$
$$\langle \text{Inst_for} \rangle \mid$$
$$\dots$$

Instrukcja przypisania

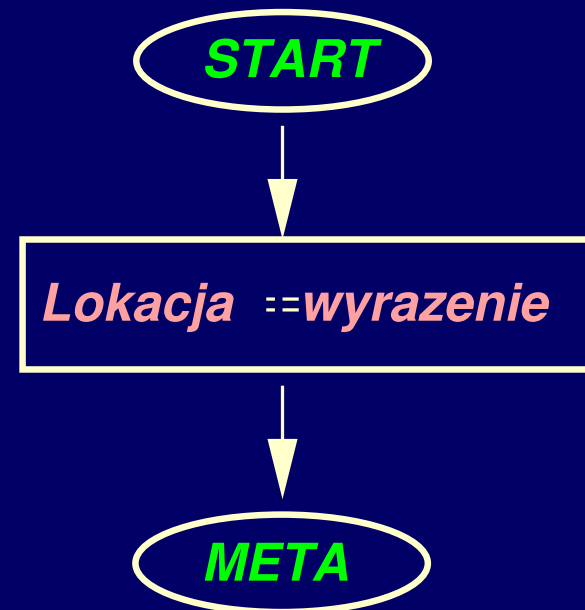
$\langle \text{Inst_przyp} \rangle ::=$
 $\langle \text{Lokacja} \rangle := \langle \text{Wyrażenie} \rangle$

Na przykład:

pom := pom - 15

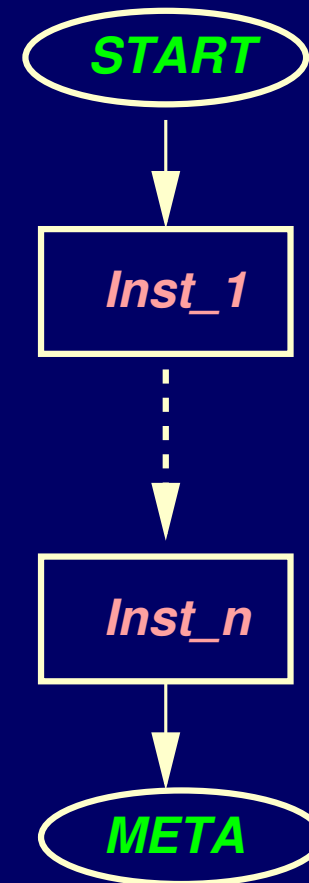
tab[i + 3] := tab[i]

rekord.pole := 3.1459



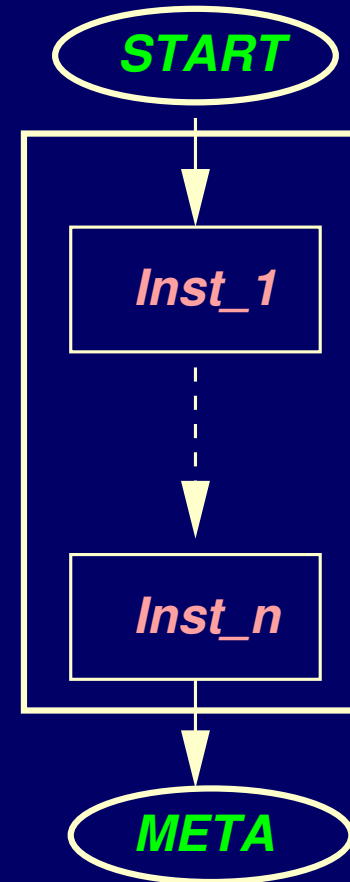
Ciąg instrukcji a instrukcja bloku

$\langle \text{Ciąg_Inst} \rangle ::=$
 $\langle \rangle \mid$
 $\langle \text{Inst} \rangle \mid$
 $\langle \text{Inst} \rangle ; \langle \text{Ciąg_Inst} \rangle$



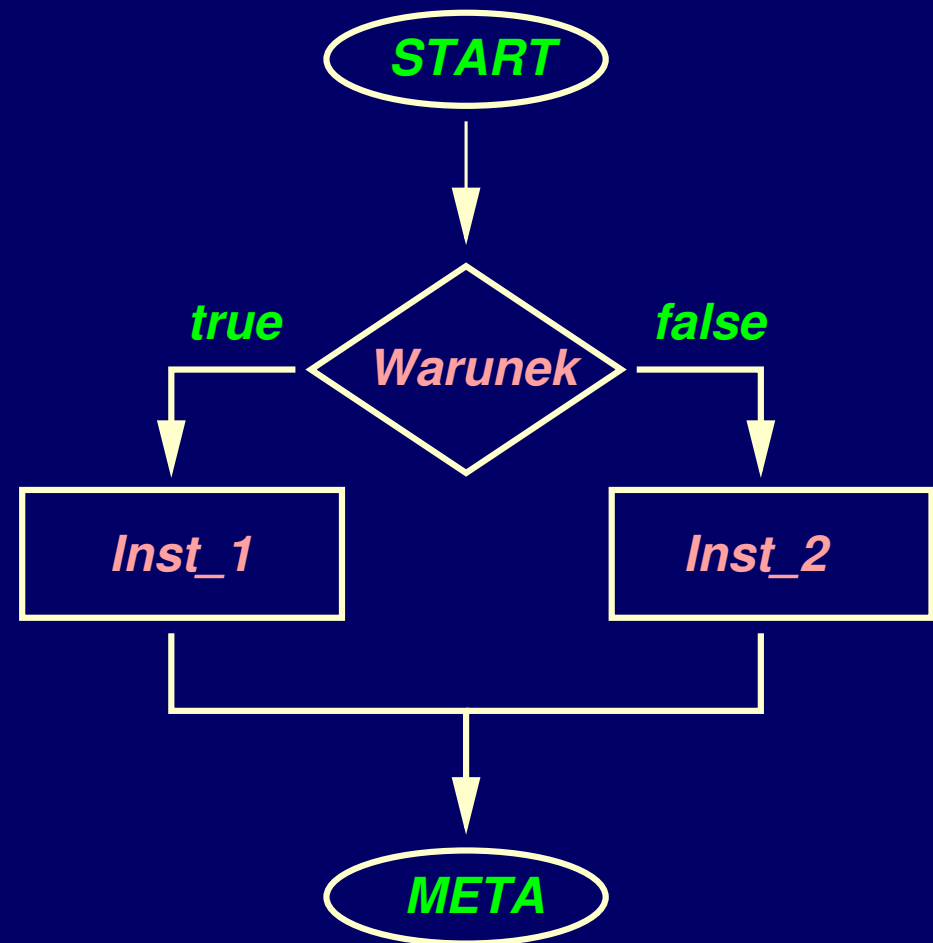
Ciąg instrukcji a instrukcja bloku

$\langle \text{Inst_bloku} \rangle ::=$
begin
 $\langle \text{Ciąg_Inst} \rangle$
end



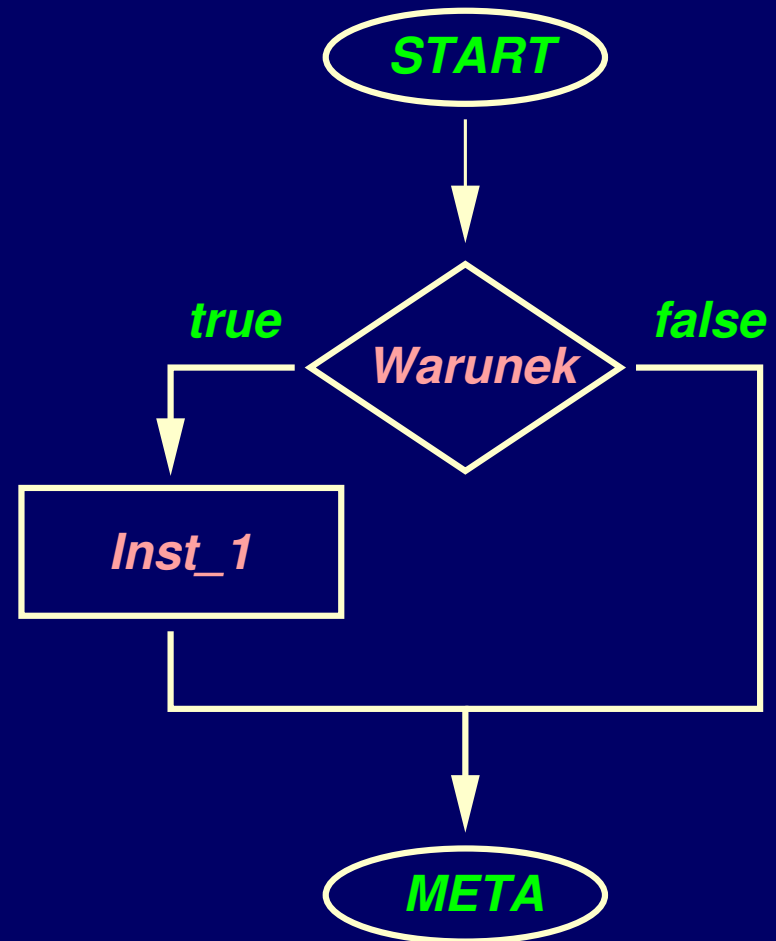
Instrukcja warunkowa if-then-else

$\langle \text{Inst_if_then_else} \rangle ::=$
if $\langle \text{Warunek} \rangle$ then
 $\langle \text{Inst_1} \rangle$
else
 $\langle \text{Inst_2} \rangle$



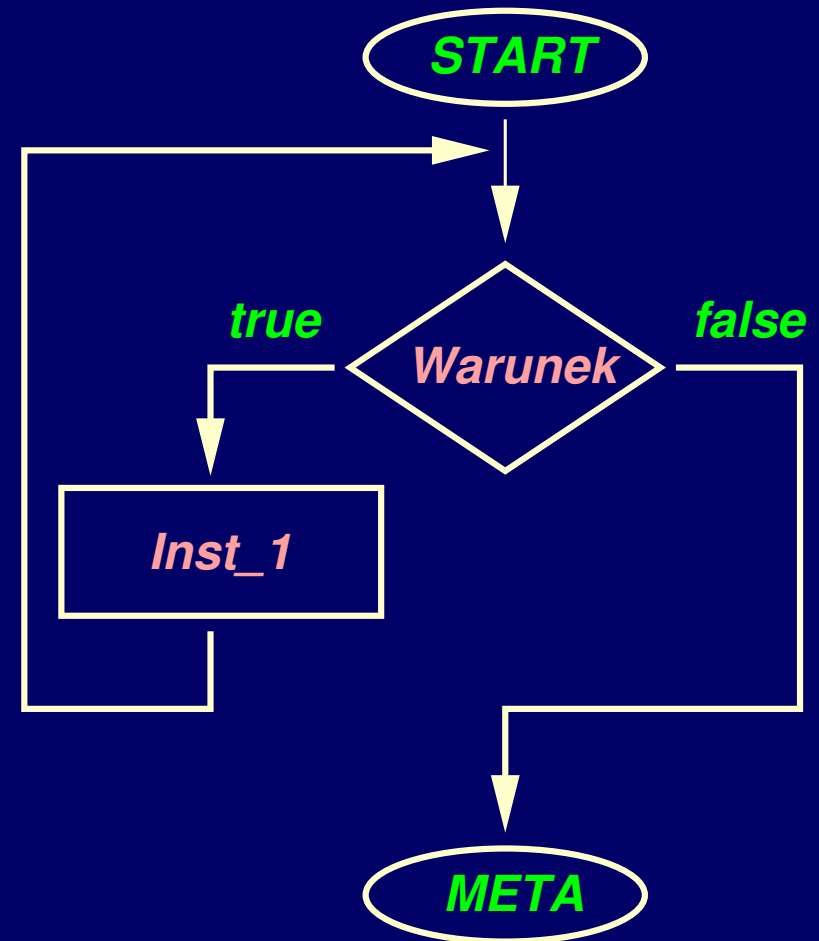
Instrukcja warunkowa if-then

$\langle \text{Inst_if_then} \rangle ::=$
if $\langle \text{Warunek} \rangle$ **then**
 $\langle \text{Inst_1} \rangle$



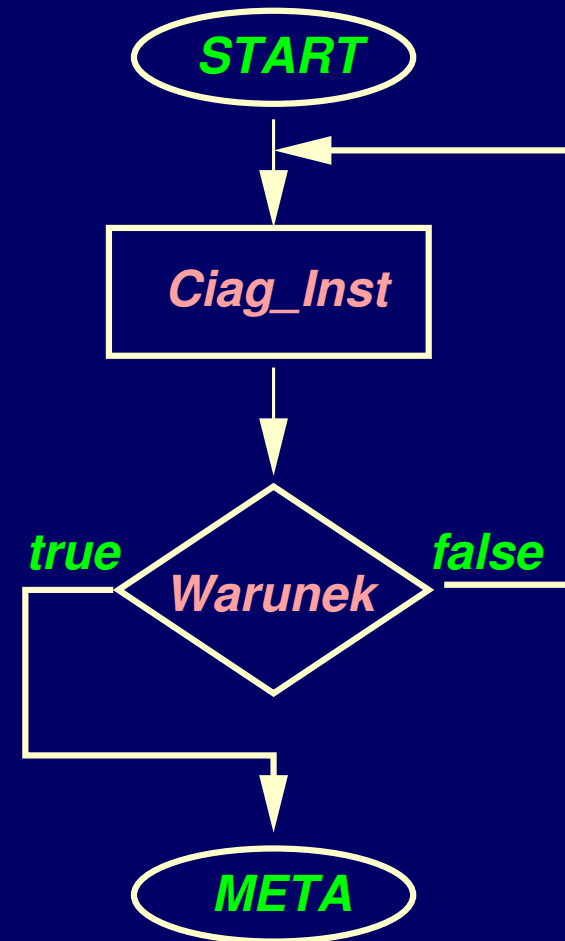
Instrukcja pętli **while-do**

$\langle \text{Inst_while} \rangle ::=$
while $\langle \text{Warunek} \rangle$ **do**
 $\langle \text{Inst_1} \rangle$



Instrukcja pętli repeat-until

$\langle \text{Inst_repeat} \rangle ::=$
repeat
 $\langle \text{Ciag_Inst} \rangle$
until $\langle \text{Warunek} \rangle$



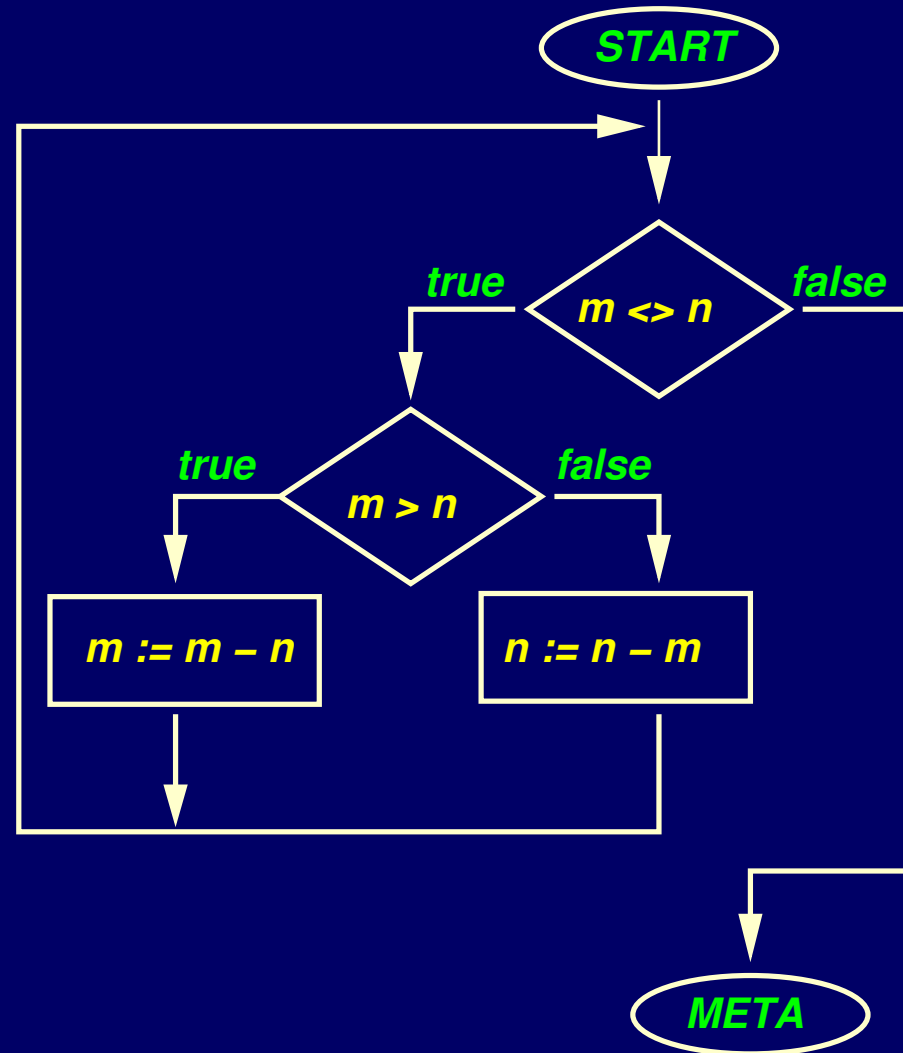
Instrukcja pętli for-do

$\langle \text{Inst_for} \rangle ::= \text{for } \langle \text{zmienna} \rangle := \langle \text{wyr_1} \rangle \text{ to } \langle \text{wyr_2} \rangle \text{ do}$
 $\quad \quad \quad \langle \text{Inst} \rangle$

jest równoważne

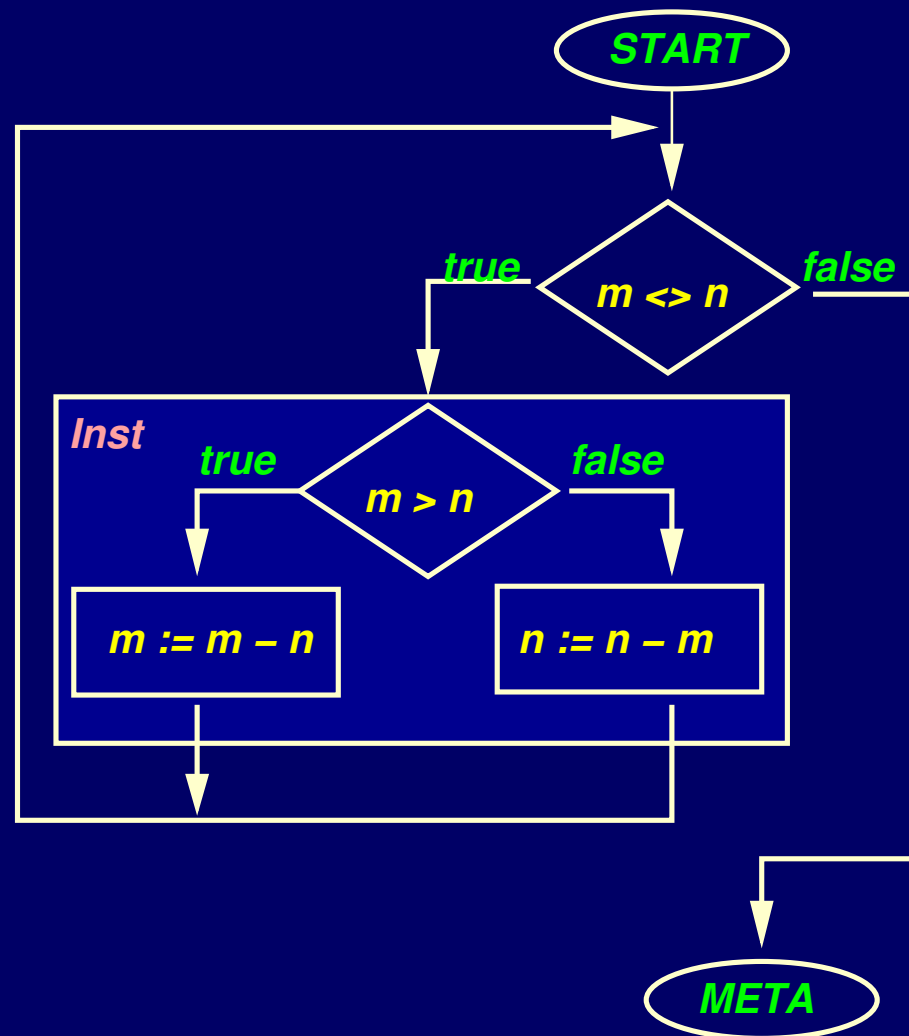
```
 $\langle \text{zmienna} \rangle := \langle \text{wyr\_1} \rangle - 1;$   
while  $\langle \text{zmienna} \rangle < \langle \text{wyr\_2} \rangle$  do  
  begin  
     $\langle \text{zmienna} \rangle := \langle \text{zmienna} \rangle + 1;$   
     $\langle \text{Inst} \rangle$   
  end
```

Algorytm Euklidesa



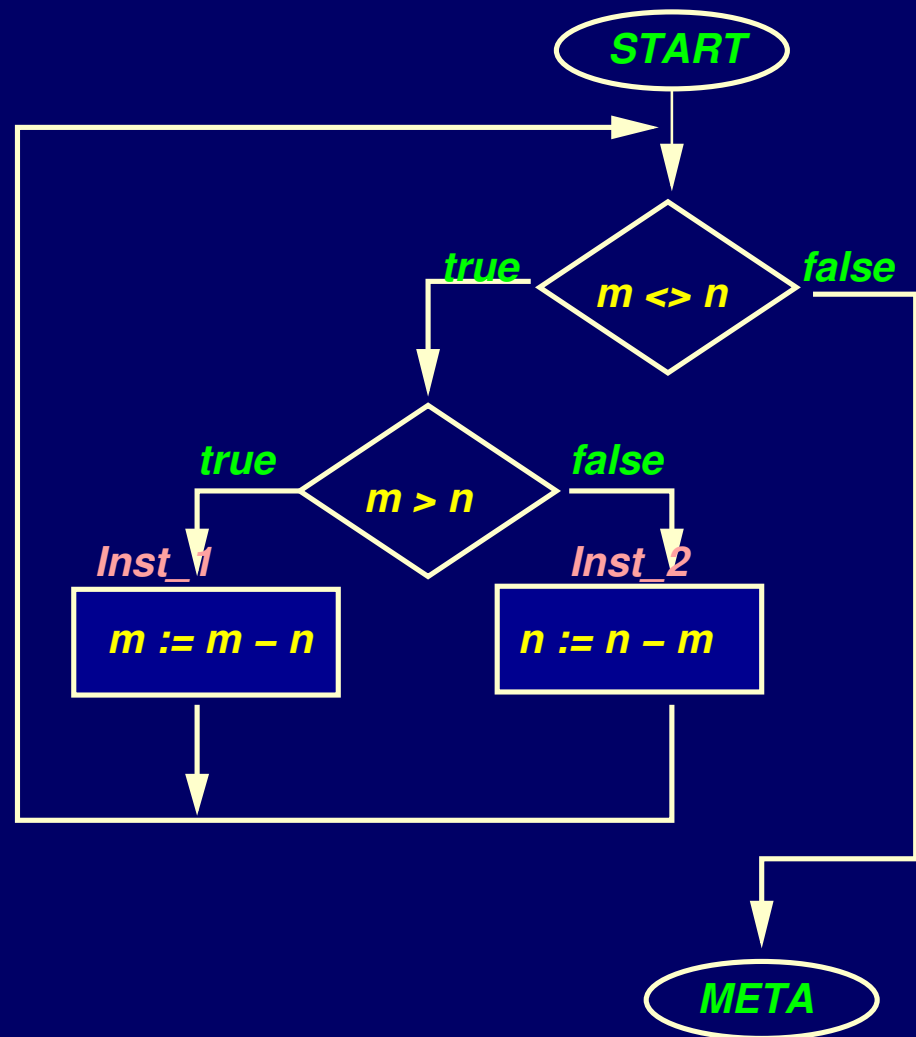
Algorytm Euklidesa

while $m \neq n$ do
 $\langle Inst \rangle$



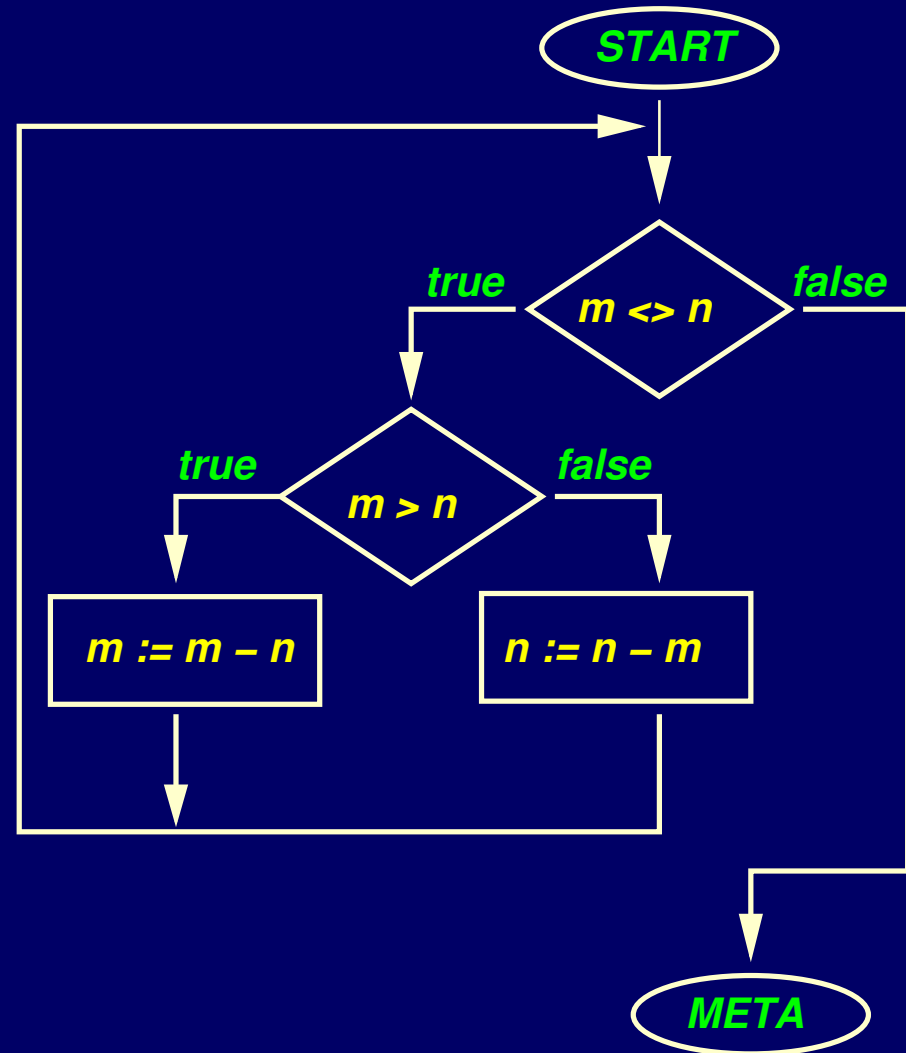
Algorytm Euklidesa

```
while  $m \neq n$  do  
  if  $m > n$  then  
     $\langle \text{Inst}_1 \rangle$   
  else  
     $\langle \text{Inst}_2 \rangle$ 
```

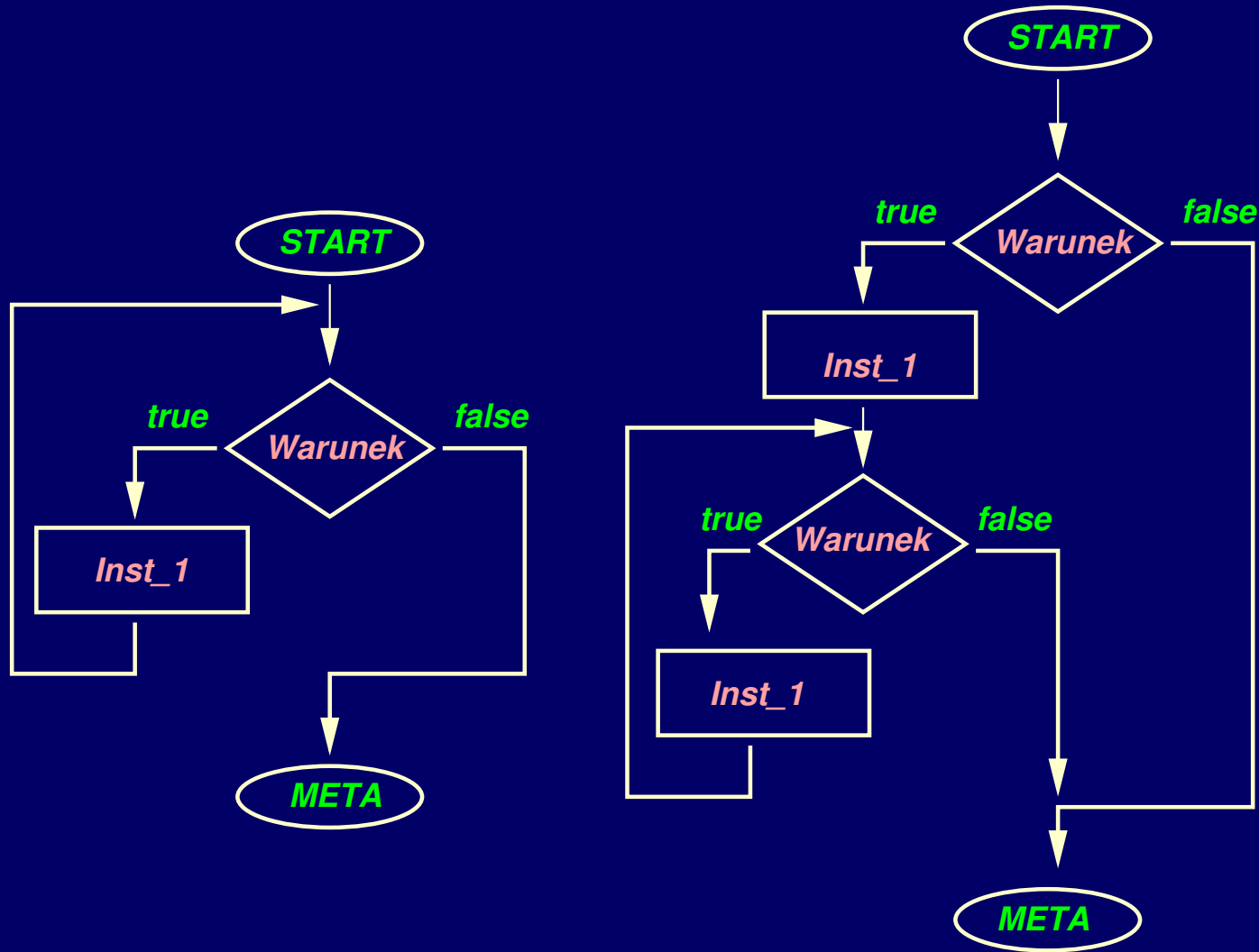


Algorytm Euklidesa

```
while  $m \neq n$  do  
  if  $m > n$  then  
     $m := m - n$   
  else  
     $n := n - m$ 
```



Pętla while-do a if



Pętla while-do a if

```
while <warunek> do
  <Inst_1>
==
if <warunek> then
  begin
    <Inst_1> ;
    while <warunek> do
      <Inst_1>
    end
  end
```