Struktura programu

```
program nazwa_programu;
uses
  modul1, modul2;
label
  etykieta1, etykieta2;
const
  nazwa_stalej = 2;
type
  nazwa_typu = definicja_typu;
var
  nazwa_zmiennej: typ_zmiennej;
procedure nazwa_procedury;
  ... {definicja procedury}
end;
function nazwa_funkcji ... : zwracany_typ;
  ... {definicja funkcji}
end;
begin
  ... {instrukcje – ciało programu}
  ...
end.
```

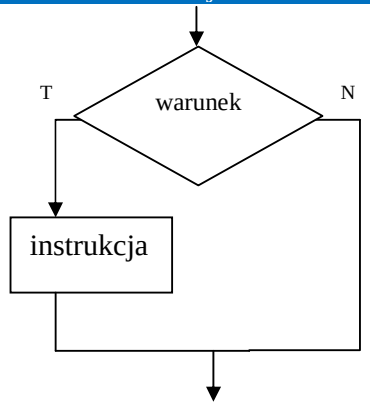
Typy danych

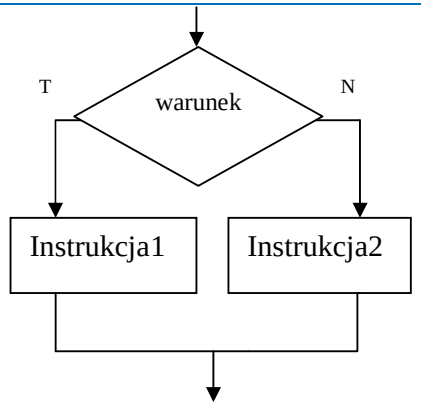
typ	opis
całkowity	
byte	od 0 do 255
word	od 0 do 65535
integer	od -32756 do 32767
longint	od -2147483648 do 2147483647
shortint	od -127 do 127
rzeczywisty	
Real	zakres 2.9e-39
single	zakres 1.5e-45
double	zakres 5.0e-324
extended	zakres 3.4e-4932
inne	
boolean	typ logiczny – wartości `true` lub `false` (prawda lub fałsz)
char	typ znakowy
1..7 'A'..'Z'	przykłady typów okrojonych
string	typ łańcuchowy

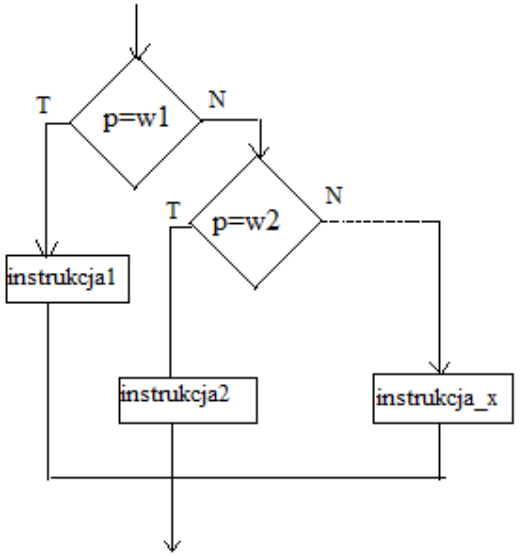
Operatory

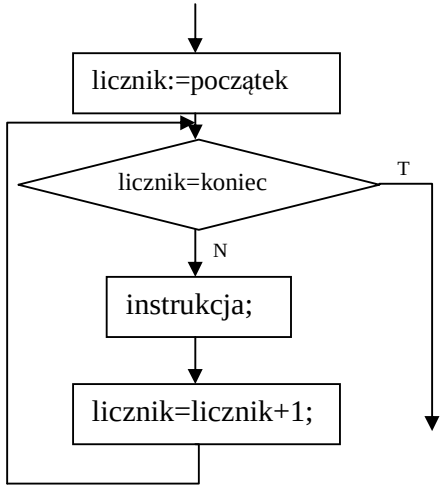
Operator	Przykład	Znaczenie
Arytmetyczne		
+	x + y	dodawanie
-	x – y	odejmowanie
*****	x * y	mnożenie
/	x / y	dzielenie
div	x div y	dzielenie całkowite
mod	x mod y	reszta z dzielenia
Logiczne		
not	**not** false = true	negacja
and	true **and** false = false	koniunkcja
or	true **or** false = true	alternatywa
xor	true **or** true = false	różnica symetryczna
shl	6 shl 1 = 12	przesunięcie w lewo
shr	14 shr 1 = 7	przesunięcie w prawo
Relacyjne		
<	x < y	mniejszy
<=	x <= y	mniejszy lub równy
>	x > y	większy
>=	x >= y	większy lub równy
<>	x <> y	różny
=	x = y	równy

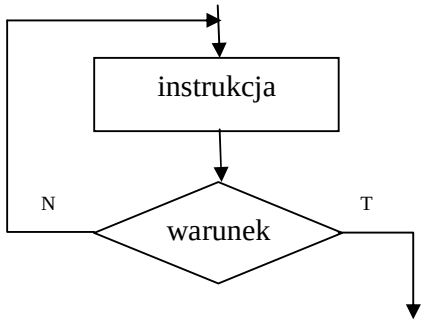
Instrukcje warunkowe

Instrukcja	Opis	Schemat blokowy
`if warunek then instrukcja;`	Jeśli został spełniony *warunek* to wykonaj: *instrukcja*	

`if warunek then`
`  instrukcja1`
`else`
`  instrukcja2;`
	Jeśli został spełniony *warunek* to wykonaj: *instrukcja1*, w przeciwnym wypadku wykonaj: *instrukcja2*.	

`case przełącznik of`
` wartosc1:`
`  instrukcja1;`
` wartosc2:`
`  instrukcja2;`
` ...`
`else`
`  instrukcja_x;`
`end;`
	W zależności od wartości *przełącznika* wykonaj odpowiednią *instrukcję*. Jeśli żadna wartość nie pasuje do przełącznika, wykonaj *instrukcję_x*;	

`for licznik:=początek to koniec` `do instrukcja;`	Przypisz *licznikowi* wartość *początek* i wykonuj *instrukcję*, aż wartość licznika będzie równa *koniec*. Po wykonaniu każdej *instrukcji* zwiększ *licznik* o 1.	

repeat instrukcja **until** warunek;	Powtrzaj *instrukcję*, aż zostanie spełniony *warunek*.	
while warunek **do** instrukcja;	Dopóki spełniony jest *warunek*, powtarzaj *instrukcję*.