

Instrukcje sterujące

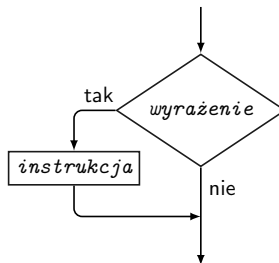
PRZYPOMNIENIE: OPERATORY

Operator przypisania			
=	przypisanie	$x = y$	$x \leftarrow y$
Operatory arytmetyczne			
*	mnożenie	$x * y$	$x \cdot y$
/	dzielenie	x / y	$\frac{x}{y}$
+	dodawanie	$x + y$	$x + y$
-	odejmowanie	$x - y$	$x - y$
%	reszta z dzielenia (modulo)	$x \% y$	$x \bmod y$
++	inkrementacja	$x++$	$x \leftarrow x + 1$
--	dekrementacja	$x--$	$x \leftarrow x - 1$
Operatory relacji			
<	mniej niż	$x < y$	$x < y$
>	większy niż	$x > y$	$x > y$
<=	mniej lub równy	$x <= y$	$x \leq y$
>=	większy lub równy	$x >= y$	$x \geq y$
==	równy	$x == y$	$x = y$
!=	różny	$x != y$	$x \neq y$
Operatory logiczne			
!	negacja (NOT)	$!x$	$\neg x$
&&	koniunkcja (AND)	$x > 1 \ \&\& \ y < 2$	$x > 1 \wedge y < 2$
	alternatywa (OR)	$x < 1 \ \ \ y > 2$	$x < 1 \vee y > 2$

INSTRUKCJA WARUNKOWA IF ELSE

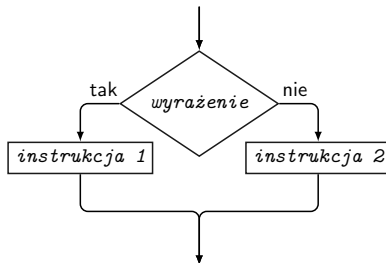
WARUNEK IF (JEŻELI)

```
if ( wyrażenie )  
    instrukcja
```



WARUNEK IF ELSE

```
if ( wyrażenie )  
    instrukcja 1  
else  
    instrukcja 2
```



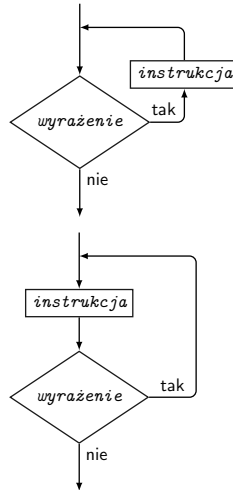
PĘTLA WHILE I DO WHILE

PĘTLA WHILE (DOPÓKI)

```
while ( wyrażenie )  
    instrukcja
```

PĘTLA DO WHILE

```
do  
    instrukcja  
while ( wyrażenie );
```



PRZYKŁAD: ALGORYTM HERONA

METODA BABILOŃSKA

Problem: wyznaczenie wartości pierwiastka $\sqrt{a}$

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right) \xrightarrow{n \rightarrow \infty} \sqrt{a}$$

Algorytm 1 Algorytm Herona

Dane wejściowe: liczba $a \geq 0$, wartość początkowa $x_0 > 0$, dokładność obliczeń $\epsilon > 0$

Wynik: wartość przybliżona $x \approx \sqrt{a}$ z dokładnością ϵ

- 1: $x \leftarrow x_0$
 - 2: **wykonuj**
 - 3: $x_0 \leftarrow x$
 - 4: $x \leftarrow \frac{1}{2} \left(x_0 + \frac{a}{x_0} \right)$
 - 5: **dopóki** $|x - x_0| > \epsilon$
 - 6: **wypisz** x
-

```

1  #include<stdio.h>
2
3  int main()
4  {
5      const float eps=1e-4;
6      float x,a,x0;
7
8      printf("a = "); scanf("%f",&a);
9
10     if( a < 0 ) printf("Zle dane: a < 0\n");
11     else
12     {
13         x0 = 1;
14         x = x0;
15         do
16         {
17             x0 = x;
18             x = (x0 + a/x0)/2;
19         }while(x - x0 > eps || x - x0 < -eps);
20
21         printf("pierwiastek z %f wynosi %f (eps= %f)\n",a,x,eps);
22     }
23     return 0;
24 }

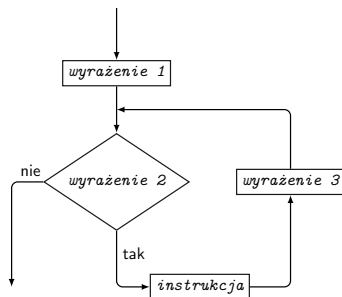
```

PĘTLA FOR

```
for ( wyrażenie 1 ; wyrażenie 2 ; wyrażenie 3 )
    instrukcja
```

```
int s=1, n=100, i;

for(i=1; i<n; i++)
{
    s = s * i;
}
```



WHILE

```
A
while ( B )
{
    instrukcja
    C
}
```

```
i=0;
while( i<n )
{
    printf( "%d\n", i );
    i=i+1;
}
```

FOR

```
for ( A ; B ; C )
{
    instrukcja
}
```

```
for(i=0; i<n; i=i+1)
    printf( "%d\n", i );
```

PRZEWANIE BREAK

Polecenie break przerywa działanie pętli while, do while, for.

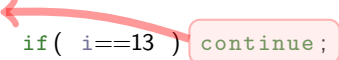
```
1  #include<stdio.h>
2
3  int main()
4  {
5      char c;
6
7      while(1)
8      {
9          printf("Czy przerwac [t/n]? ");
10         scanf(" %c", &c);
11         if( c == 't' || c == 'T' ) break;
12         printf("Przmysl to!\n");
13     }
14     printf("Koniec.\n");
15 }
16
```

Czy przerwac [t/n]? n
Przmysl to!
Czy przerwac [t/n]? N
Przmysl to!
Czy przerwac [t/n]? y
Przmysl to!
Czy przerwac [t/n]? t
Koniec.

KOLEJNA ITERACJA: CONTINUE

Polecenie `continue` przechodzi do następnej iteracji (pomija wszystkie instrukcje do końca bloku pętli).

```
1  #include<stdio.h>
2
3  int main()
4  {
5      int i, n=20;
6
7      for(i=1; i<=n; i++)
8      {
9          if( i==13 ) continue;
10         printf("%d\n",i);
11     }
12
13     return 0;
14 }
```



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

 `continue.c`

GOTO

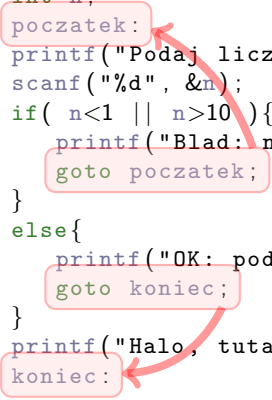
Przenosi sterowanie w miejsce kodu oznaczone etykietą.

```
etykieta:  
instrukcje  
...  
goto etykieta;
```

Główne przykazanie **programowania strukturalnego**:

Nie używaj instrukcji skoku!

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int n;
6      poczatek:
7      printf("Podaj liczbe z zakresu od 1 do 10\n");
8      scanf("%d", &n);
9      if( n<1 || n>10 ){
10         printf("Blad: niepoprawna wartosc\n");
11         goto poczatek;
12     }
13     else{
14         printf("OK: podales liczbe %d\n", n);
15         goto koniec;
16     }
17     printf("Halo, tutaj jestem!\n");
18     koniec:
19
20     return 0;
21 }
```



- Algorytm zawierający wiele instrukcji skoku, przenoszących sterowanie do przodu lub do tyłu, bardzo szybko staje się trudny do zrozumienia
- Trudności techniczne, niejednoznaczność, nieczytelność:
skoki do wnętrza pętli, z pętli do pętli, z podprogramu (funkcji) do podprogramu, itp. ...
- `break` i `continue` - podobne wątpliwości
- Program używający instrukcji skoku zawsze można przepisać do postaci nie zawierającej tej instrukcji
- Używanie instrukcji skoku to zły styl programowania

POPZREDNI PRZYKŁAD BEZ SKOKU

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int n;
6      int stop = 0;
7
8      while( stop == 0 )
9      {
10         printf("Podaj liczbe z zakresu od 1 do 10\n");
11         scanf("%d", &n);
12         if( n >= 1 && n<=10 ) stop=1;
13         else printf("Blad: niepoprawna wartosc\n");
14     }
15
16     printf("OK: podales liczbe %d\n", n);
17     return 0;
18 }
```

PĘTLE ZAGNIEŻDŻONE

```
1  #include <stdio.h>
2
3  int main ()
4  {
5      int i, j, n;
6
7      printf("Podaj zakres: ");
8      scanf("%d", &n);
9
10     printf("Liczby pierwsze w zakresie od 1 do %d\n", n);
11     for(i=2; i<n; i++) {
12         for(j=2; j <= (i/j); j++)
13             if(!(i%j)) break; /* nie jest l. pierwsza */
14         if(j > (i/j)) printf("%d\n", i);
15     }
16
17     return 0;
18 }
```

Instrukcja warunkowa `switch` porównuje wyrażenie z listą wartości i wykonuje instrukcje pasującego przypadku (`case`).

```
switch( wyrażenie )  
{  
    case wartość 1:  
        instrukcja 1  
        break;  
    case wartość 2:  
        instrukcja 1  
        break;  
    ...  
    default :  
        instrukcja n  
}
```

```

1  #include<stdio.h>
2
3  int main()
4  {
5      float x, y;
6      char op;
7
8      scanf("%f %c%f",&x, &op, &y);
9
10     switch(op)
11     {
12         case '+':
13             printf("%f\n", x+y);
14             break;
15         case '-':
16             printf("%f\n", x-y);
17             break;
18         case '*':
19             printf("%f\n", x*y);
20             break;
21         case '/':
22             printf("%f\n", x/y);
23             break;
24         default:
25             printf("Nieznana operacja: %c\n", op);
26     }
27     return 0;
28 }

```

3.14 + 5

8.140000

3.14+5

8.140000

3.14

+

5

8.140000

- Dopasowanie przypadków tylko dla zmiennych całkowitych (int, char, enum)
- break kończy działanie instrukcji switch
- instrukcje break i default są opcjonalne
- Po dopasowaniu właściwego przypadku wykonywane są WSZYSTKIE instrukcje aż do napotkania pierwszego wystąpienia break (lub do końca bloku switch). Pominięcie instrukcji break spowoduje więc wykonanie instrukcji dla kolejnego przypadku.

```
1  #include<stdio.h>
2
3  int main()
4  {
5      int n;
6
7      printf("Podaj liczbe od 1 do 4\n");
8      scanf("%d", &n);
9
10     switch(n)
11     {
12         case 1:
13             printf("Przypadek 1\n");
14         case 2 :
15             printf("Przypadek 2\n");
16         case 3 :
17             printf("Przypadek 3\n");
18             break;
19         case 4 :
20             printf("Przypadek 4\n");
21             break;
22         default:
23             printf("Nieznana operacja.\n");
24     }
25     return 0;
26 }
```

Podaj liczbe od 1 do 4
2
Przypadek 2
Przypadek 3

- Instrukcje warunkowe: `if`, `else`, `switch`
- Pętle: `while`, `do while`, `for`
- Operator inkrementacji `++` i dekrementacji `--`
- Instrukcja skoku `goto` (lepiej nie używać)
- Instrukcja przerwanie `break` oraz kontynuacji pętli `continue`
- Instrukcja wielokrotnego wyboru `switch`



David Griffiths, Dawn Griffiths „*Rusz głową! C.*”, Helion, Gliwice, 2013.



„Kurs programowania w C”, WikiBooks,
<http://pl.wikibooks.org/wiki/C>