



## Podstawy programowania (wykład)

## Instrukcje języka C

# Instrukcje języka C

- Wykaz instrukcji
- Instrukcja prosta i złożona
- Instrukcja pusta
- Instrukcje warunkowe
  - Instrukcja `if`
  - Instrukcja `if-else`
  - Instrukcja `switch`
- Instrukcje iteracyjne
  - Pętla `for`
  - Pętla `while`
  - Pętla `do-while`
  - Instrukcja `break`
  - Instrukcja `continue`
- Instrukcja skoku `goto`
- Instrukcja powrotu z funkcji `return`

# Wykaz instrukcji

- ❖ prosta
- ❖ złożona { }
  
- ❖ pusta ;
- ❖ warunkowe:
  - **if** (wariant podstawowy)
  - **if-else** (wariant rozszerzony)
  - wyboru **switch**
- ❖ iteracyjne (pętle):
  - **for** (pętla z licznikiem)
  - **while** (pętla warunkowa, z testem na wejściu)
  - **do-while** (pętla warunkowa, z testem na wyjściu)
- ❖ przerwania pętli **break**
- ❖ kontynuacji pętli **continue**
- ❖ skoku **goto**
- ❖ powrotu (wyjścia) z funkcji **return**

# Instrukcja prosta i złożona

Podział występujący w wielu językach programowania

**Instrukcja prosta**            `<instrukcja>;`

Pojedyncze polecenie zakończone średnikiem.

**Instrukcja złożona { }**

W języku C to instrukcja **blokowa** (ogranicza zakres widoczności zmiennych).

Blok (zestaw, grupa) instrukcji prostych zgrupowanych za pomocą nawiasów klamrowych.

```
{  
    <instrukcja 1>;  
    <instrukcja 2>;  
    <instrukcja 3>;  
    ...  
}
```

# Instrukcja pusta

Instrukcja pusta nie powoduje wykonania jakiejkolwiek czynności

## Instrukcja pusta

Polecenie „*nic nie wykonuj*”.

Oznaczana jest średnikiem ;

W procesorach rodziny Intel x86 odpowiada rozkazowi NOP.

NOP – *No Operation*

# Instrukcje warunkowe

Strona 6

## Instrukcja warunkowa **if**

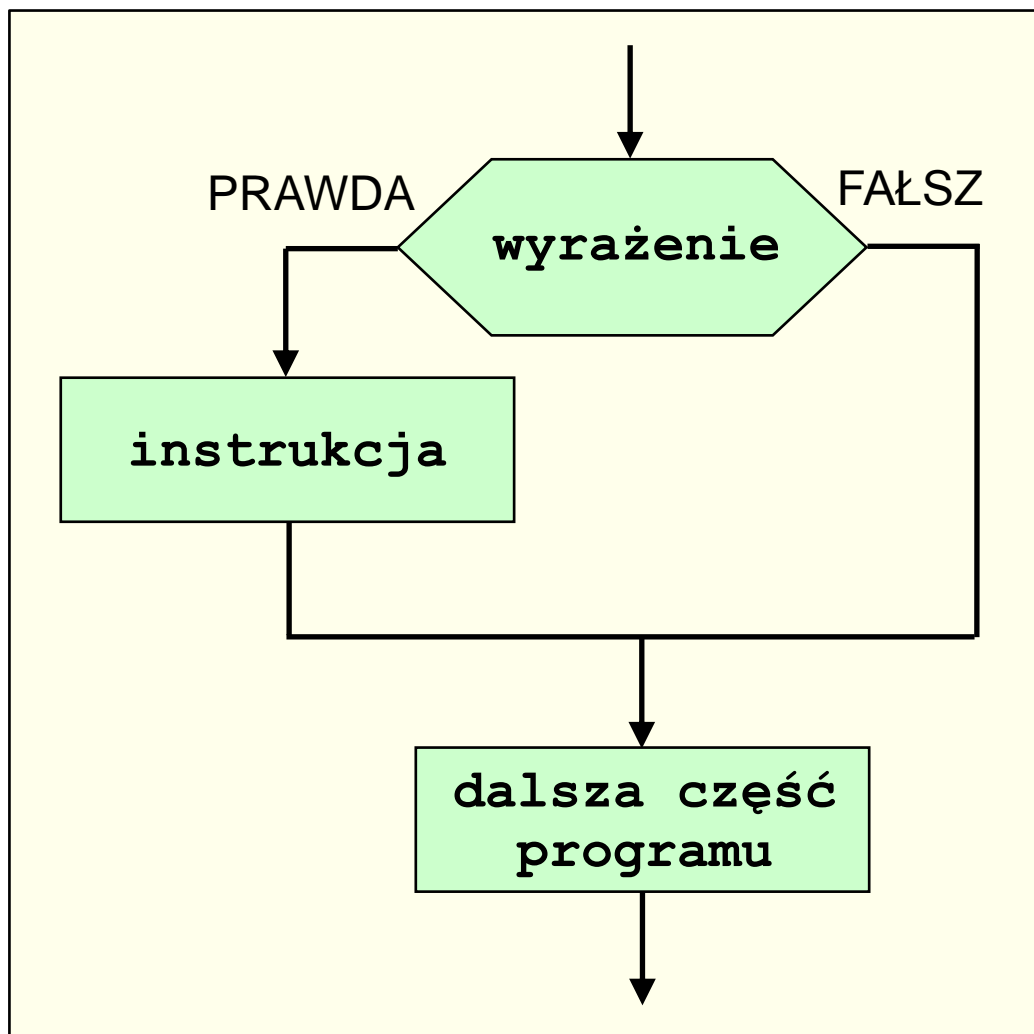
składnia:

```
if ( <wyrażenie> )  
  <instrukcja>;
```

```
if ( <wyrażenie> )  
{  
  <instrukcja 1>;  
  <instrukcja 2>;  
  <instrukcja 3>;  
  ...  
}
```

przykład:

```
if (ocena > 3)  
  printf("Dobrze") ;
```



# Instrukcje warunkowe

Strona 7

składnia:

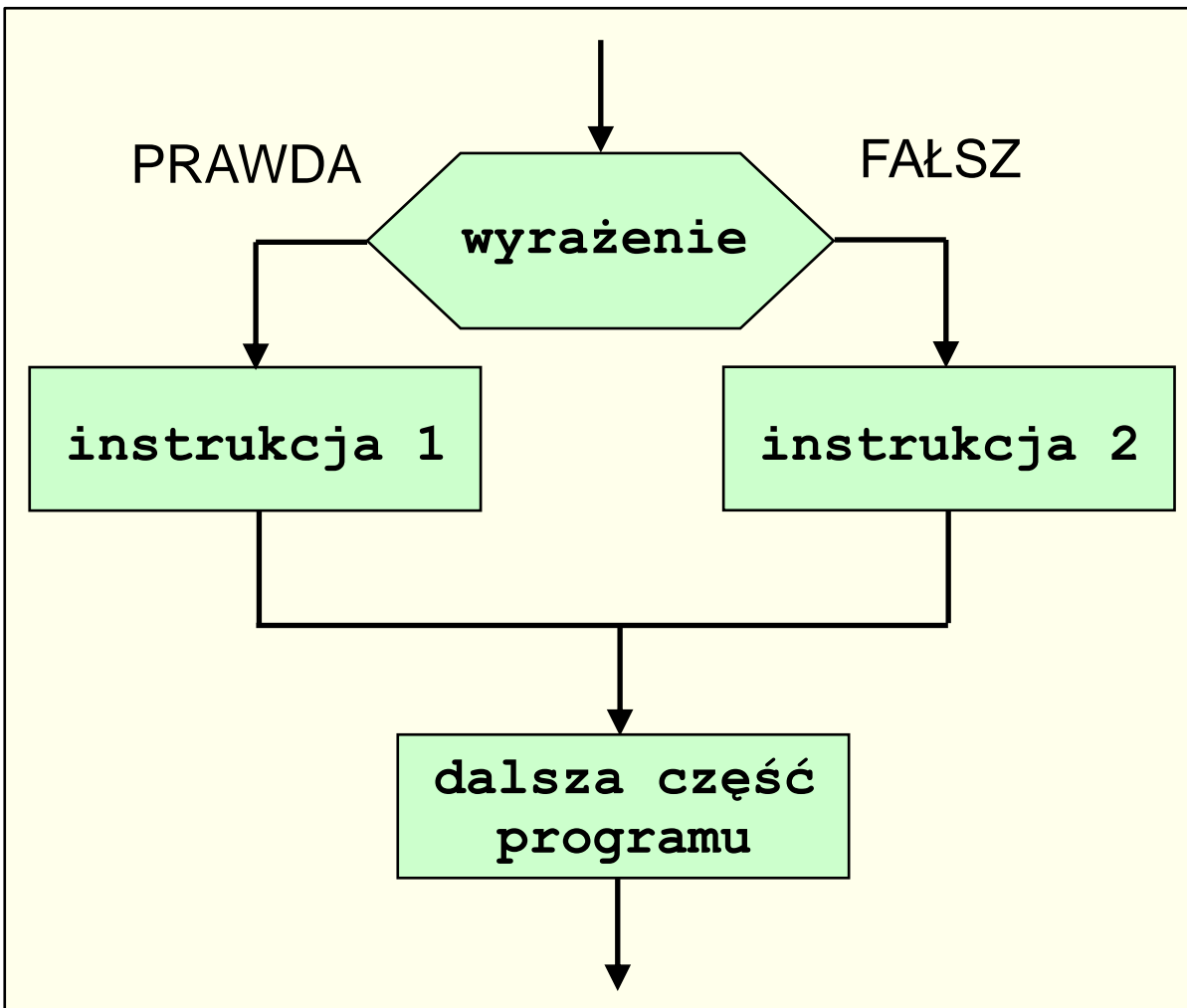
Instrukcja warunkowa **if-else**

```
if ( <wyrażenie> )  
    <instrukcja 1>;  
else  
    <instrukcja 2>;
```

```
if ( <wyrażenie> )  
{  
    <instrukcja 1>;  
    ...  
}  
else  
{  
    <instrukcja 2>;  
    ...  
}
```

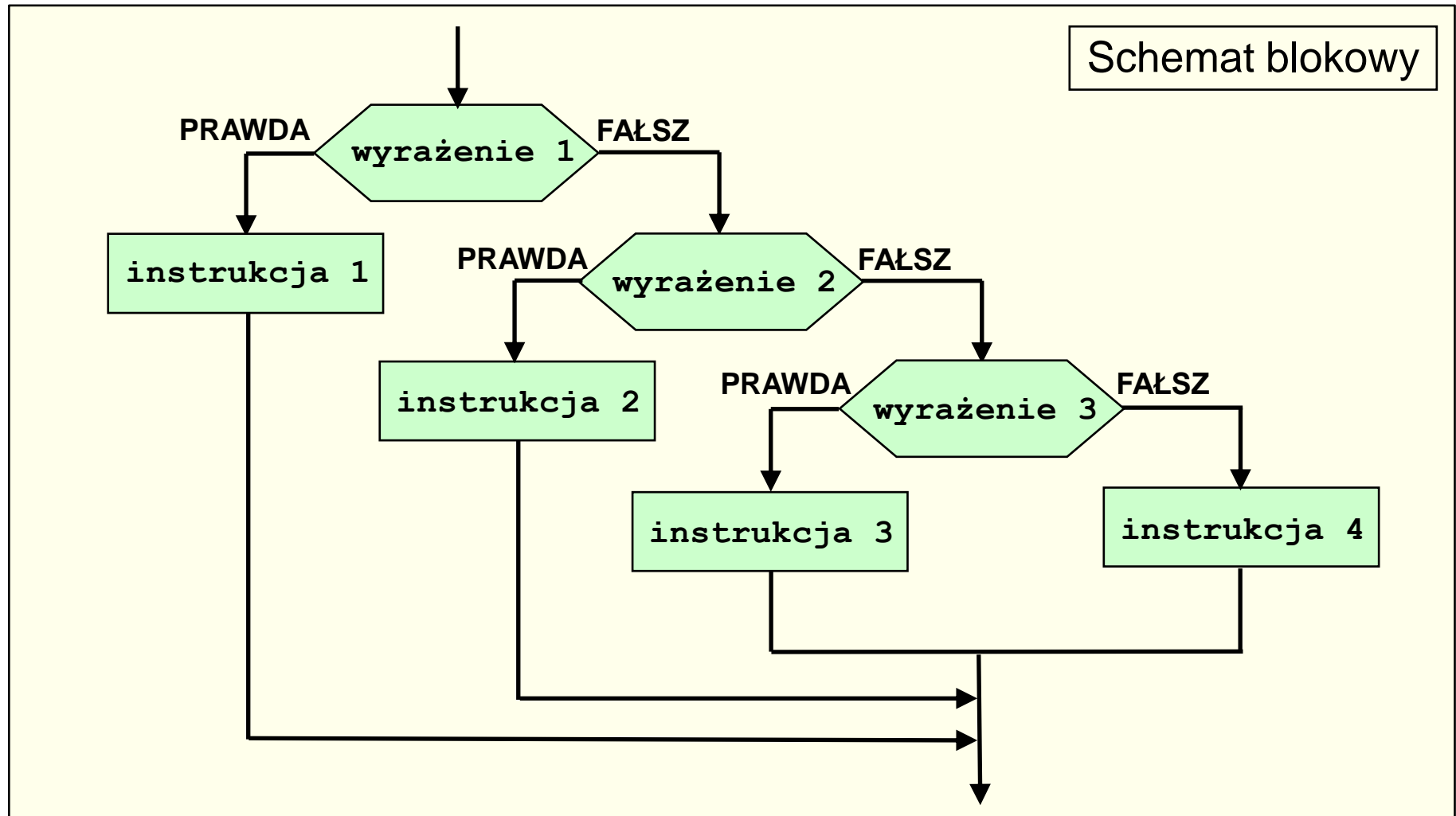
przykład:

```
if (ocena > 3)  
    printf("Dobrze");  
else  
    printf("Źle");
```



# Instrukcje warunkowe

Ciąg zagnieżdżonych instrukcji warunkowych **if-else**





# Instrukcje warunkowe

Ciąg zagnieżdżonych instrukcji warunkowych **if-else**

składnia

```
if (<wyrażenie 1>
    <instrukcja 1>;
else
    if (<wyrażenie 2>)
        <instrukcja 2>;
    else
        if (<wyrażenie 3>)
            <instrukcja 3>;
        else
            <instrukcja 4>;
```

kod źródłowy

```
if (ocena == 5)
    printf("Bardzo dobrze");
else
    if (ocena == 4)
        printf("Dobrze");
    else
        if (ocena == 3)
            printf("Dostatecznie");
        else
            printf("Niedostatecznie");
```

# Instrukcje warunkowe

Strona 10

Instrukcja wyboru **switch** (składnia)

wyrażenie stałoprzecinkowe

wartość stałoprzecinkowa

```
switch ( <wyrażenie> )  
{  
    case <wartość 1>: <instrukcja 1>; break;  
    case <wartość 2>: <instrukcja 2>; break;  
    case <wartość 3>: <instr3>; <instr4>; break;  
    /* . . . */  
    default: <instrukcja 5>; break;  
}
```

opcjonalne

# Instrukcje warunkowe

Instrukcja wyboru **switch** (przykład kodu)

```
switch (ocena)
{
    case 5:  printf("Bardzo dobrze"); break;
    case 4:  printf("Dobrze"); break;
    case 3:  printf("Dostatecznie"); break;
    case 2:  printf("Niedostatecznie"); break;
    default: printf("Ocena spoza skali"); break;
}
```

```
switch (znak)
{
    case 'a': printf("litera"); break;
    case '5': printf("cyfra"); break;
    case '+': printf("plus"); break;
}
```

# Instrukcje warunkowe

## Podsumowanie

Instrukcja warunkowa **if** (variant podstawowy):

pozwała określić czynności wykonywane, kiedy sprawdzany warunek jest spełniony.

Instrukcja warunkowa **if-else** (variant rozszerzony):

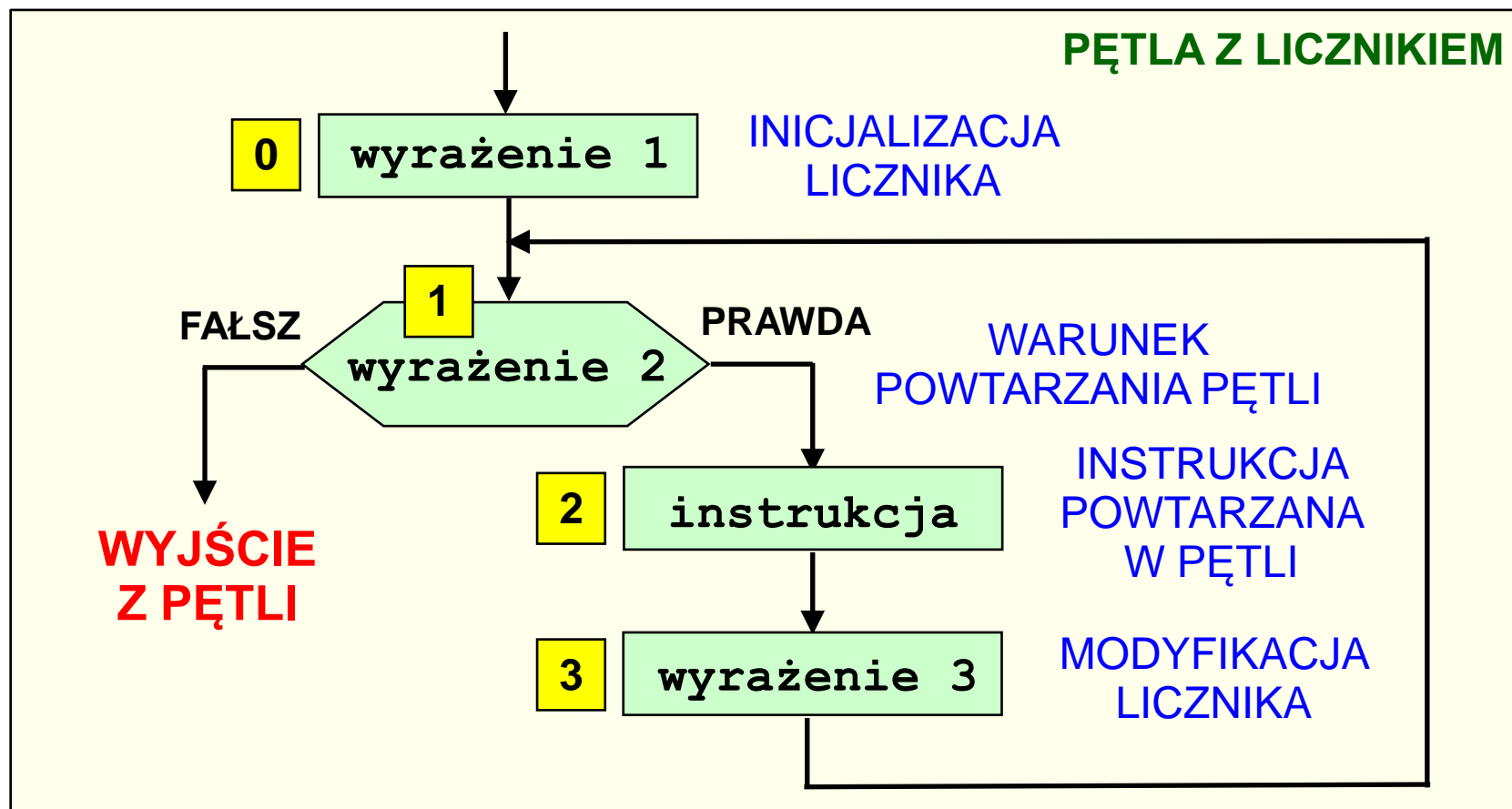
pozwała określić czynności wykonywane, kiedy sprawdzany warunek jest spełniony i czynności wykonywane kiedy ten sam warunek nie jest spełniony.

Instrukcja wyboru **switch**:

pozwała badać wartość stałoprzecinkową wyrażenia i w zależności od konkretnych wartości tego wyrażenia, wykonać różne czynności; w pewnych przypadkach pozwala zastąpić ciąg zagnieżdżonych instrukcji warunkowych.

# Instrukcje iteracyjne

**0**      Pętla z licznikiem **for**  
**for** ( <wyrażenie1>; <wyrażenie2>; <wyrażenie3> )  
    <instrukcja>;      **2**      **1**      **3**



# Instrukcje iteracyjne

## Pętla z licznikiem **for**

```
int i;  
for ( i = 1; i <= 5; i = i+1 )  
    printf("Przebieg %d pętli\n", i);
```

```
Przebieg 1 pętli  
Przebieg 2 pętli  
Przebieg 3 pętli  
Przebieg 4 pętli  
Przebieg 5 pętli  
—
```

```
int i;    /* licznik */  
int n;    /* liczba */  
int silnia = 1; /* wynik */  
  
scanf("%d", &n);  
for ( i = 1; i <= n; i = i+1 )  
    silnia = silnia*i;
```

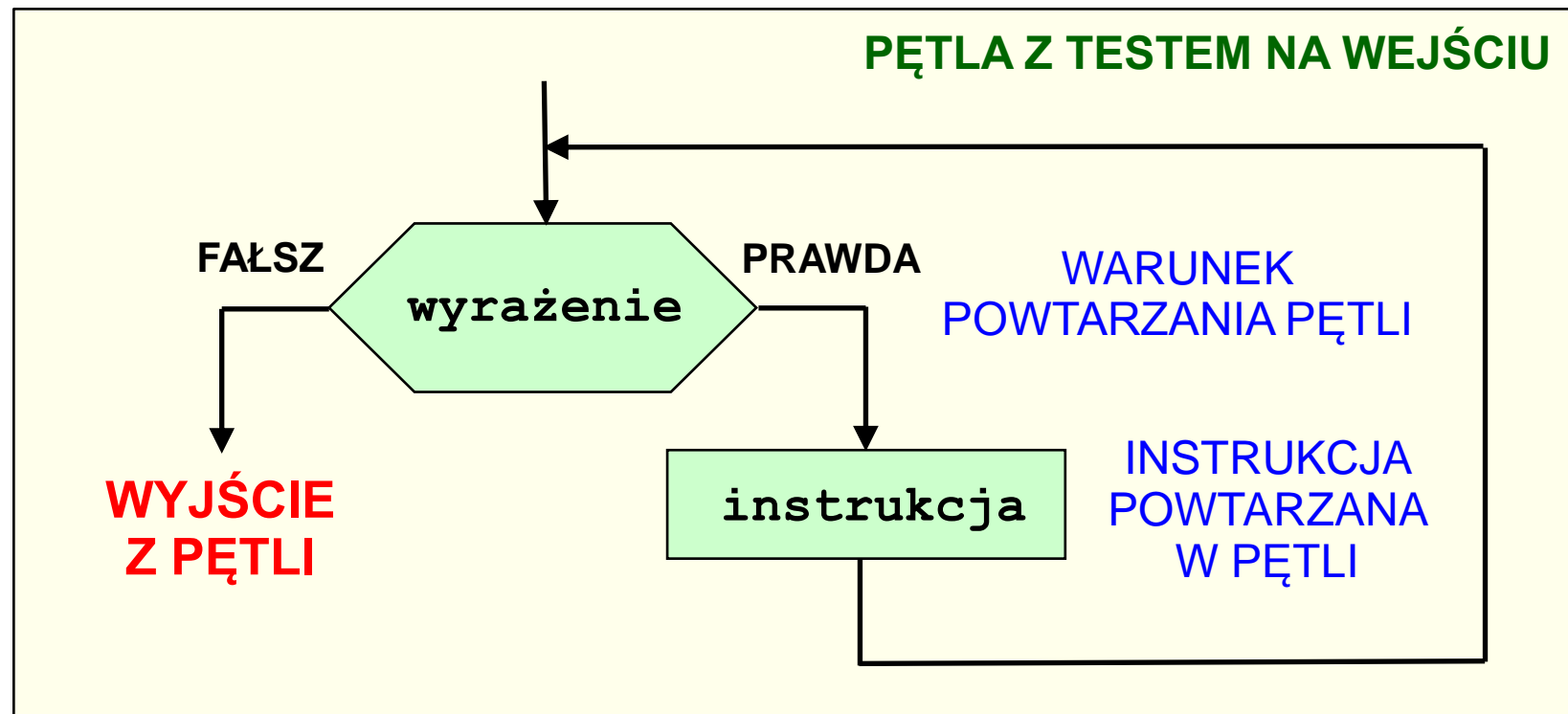
$$n! = 1 * 2 * \dots * n$$

# Instrukcje iteracyjne

Pętla warunkowa, z testem na wejściu **while**

```
while ( <wyrażenie> )  
  <instrukcja>;
```

```
while ( <wyrażenie> )  
{  
  <instrukcja 1>;  
  <instrukcja 2>;  
  ...  
}
```



# Instrukcje iteracyjne

Pętla warunkowa, z testem na wejściu **while**

```
while ( <wyrażenie> )  
{  
    <instrukcja1>;  
    ...  
}
```

```
int i;  
int n;  /* liczba */  
int silnia = 1;  /* wynik */  
scanf("%d", &n);  
  
i = 1;  
while ( i <= n )  
{  
    silnia = silnia*i;  
    i = i+1;  
}
```

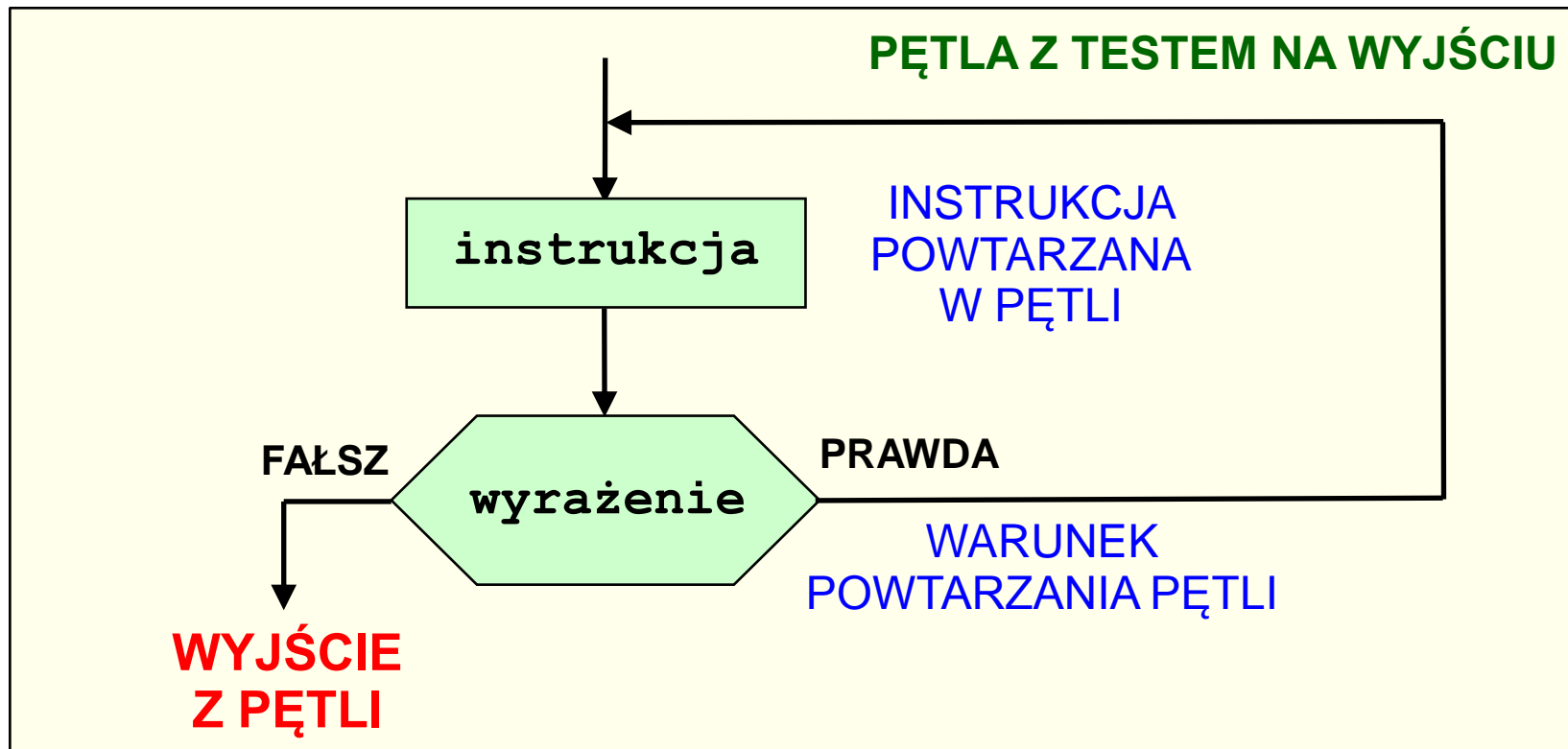


# Instrukcje iteracyjne

Pętla warunkowa, z testem na wyjściu **do-while**

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

```
do  
{  
    <instrukcja 1>;  
    <instrukcja 2>;  
    ...  
} while ( <wyrażenie> );
```



# Instrukcje iteracyjne

Pętla warunkowa, z testem na wyjściu **do-while**


```
int i;  
int n;  /* liczba */  
int silnia = 1;  /* wynik */  
scanf("%d", &n);  
  
i = 1;  
do  
{  
    silnia = silnia*i;  
    i = i+1;  
} while ( i <= n );
```

# Instrukcje iteracyjne

Strona 19

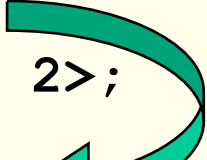
## Przerwanie pętli: instrukcja **break**

```
for ( <wyrażenie1>; <wyrażenie2>; <wyrażenie3> )  
{  
    <instrukcja 1>;  
    if ( <wyrażenie> )  
        break;  
    <instrukcja 2>;  
}
```




**Wyjście poza pętlę**

```
while ( <wyrażenie> )  
{  
    <instrukcja 1>;  
    if ( <wyrażenie> )  
        break;  
    <instrukcja 2>;  
}
```



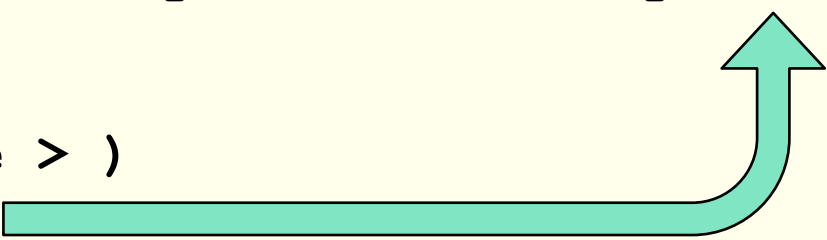
```
do  
{  
    <instrukcja 1>;  
    if ( <wyrażenie> )  
        break;  
    <instrukcja 2>;  
} while ( <wyrażenie> );
```



# Instrukcje iteracyjne

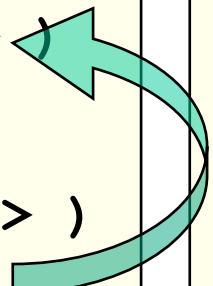
Kontynuacja pętli od następnego przebiegu: instrukcja **continue**

```
for ( <wyrażenie1>; <wyrażenie2>; <wyrażenie3> )  
{  
    <instrukcja 1>;  
    if ( <wyrażenie > )  
        continue;  
    <instrukcja 2>;  
}
```



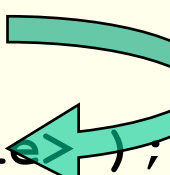
**Przejdźcie na koniec pętli**

```
while ( <wyrażenie> )  
{  
    <instrukcja 1>;  
    if ( <wyrażenie > )  
        continue;  
    <instrukcja 2>;  
}
```



**Przejdźcie na wejście pętli**

```
do  
{  
    <instrukcja 1>;  
    if ( <wyrażenie > )  
        continue;  
    <instrukcja 2>;  
} while ( <wyrażenie> );
```



**Przejdźcie na wyjście pętli**

# Instrukcje iteracyjne

## Podsumowanie

Pętla z licznikiem **for** jest stosowana, gdy pewne czynności należy powtarzać n-krotnie. Działaniem pętli steruje zmienna nazywana licznikiem.

Pętla warunkowa z testem na wejściu **while** jest stosowana, gdy liczba iteracji zależy od spełnienia warunku zdefiniowanego na wejściu pętli.

Pętla warunkowa z testem na wyjściu **do-while** jest stosowana, gdy liczba iteracji zależy od spełnienia warunku zdefiniowanego na wyjściu pętli. Czynności powtarzane wewnątrz pętli zawsze zostaną wykonane przynajmniej jeden raz (test dopiero na wyjściu).

Instrukcja **break** powoduje przerwanie bieżącej iteracji i wyjście poza pętlę.

Instrukcja **continue** powoduje przerwanie bieżącej iteracji i przejście do następnego przebiegu pętli.

# Instrukcja skoku

Strona 22

## Instrukcja skoku `goto`

Skok do etykiety (w przód lub w tył) w obrębie tej samej funkcji.

`goto <etykieta>;`

### PRZYKŁAD 1:

```
goto koniec;  
...  
koniec:           etykieta  
    <instrukcja 1>;  
    <instrukcja 2>;  
    ...
```

### PRZYKŁAD 2:

```
...  
powrot:           etykieta  
    <instrukcja 1>;  
    <instrukcja 2>;  
    ...  
goto powrot;
```

**NIE ZALECA SIĘ UŻYWANIA INSTRUKCJI SKOKU.**  
**NIE NALEŻY NADUŻYWAĆ INSTRUKCJI SKOKU.**

Zamiast instrukcji skoku należy stosować odpowiednio zbudowaną pętlę (na przykład `do-while` w sytuacji chęci powtórzenia pewnych czynności).

# Instrukcja powrotu z funkcji

Instrukcja powrotu (wyjścia) z funkcji **return**

Wyjście z funkcji z podaną wartością (lub bez wartości) i powrót do miejsca wywołania funkcji.

```
return <wartość>;
```

```
return <wyrażenie>;
```

```
return;
```

## PRZYKŁADY:

```
return 0;
```

```
return -1;
```

```
return a;
```

```
return sqrt(a);
```

```
return 2*M_PI*r;
```

```
return;
```