



UNIWERSYTET
EKONOMICZNY
we Wrocławiu



PROGRAMOWANIE W TECHNOLOGII .NET–WYKŁAD 5

INSTRUKCJE ITERACYJNE

DR RADOSŁAW WÓJTOWICZ

ALGORYTM Z POWTÓRZENIAMI (PĘTLE)

Występuje powtarzanie wykonania tych samych instrukcji wynikające z zastosowania przynajmniej jednego warunku.

Wyróżnia się algorytmy:

- **z cyklem**, tzn. z góry określona jest powtarzalność danego ciągu czynności, przed rozpoczęciem pętli; liczba powtórzeń jest znana przed rozpoczęciem cyklu (*iteracja deterministyczna*);
- **z iteracją**, tzn. dany ciąg czynności jest wykonywany tak długo, aż nastąpi prawdziwość postawionego warunku; liczba powtórzeń jest nieznana, zależy od spełnienia pewnego warunku (*iteracja niedeterministyczna*).

Opis algorytmu sumowania kolejnych 4 liczb (1/2)

1. Zaczynij algorytm
2. Suma jest równa 0
- 3. Wprowadź pierwszą liczbę**
- 4. Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
- 5. Wprowadź drugą liczbę**
- 6. Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
- 7. Wprowadź trzecią liczbę**
- 8. Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
- 9. Wprowadź czwartą liczbę**
- 10. Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
11. Wyprowadź wynik sumy
12. Zakończ algorytm

Opis algorytmu sumowania kolejnych 4 liczb (2/2)

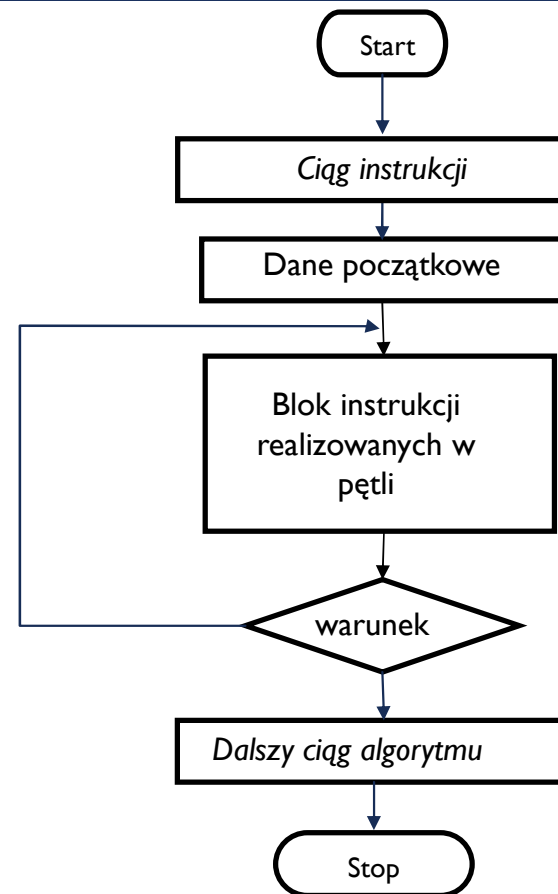
1. Zaczynij algorytm
2. Suma jest równa 0
3. **Licznik jest równy 0** *// Dodatkowa zmienna: Licznik*
4. **Do poprzedniej wartości licznika dodaj 1**
5. **Wprowadź liczbę**
6. **Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
7. **Sprawdź czy licznik równa się 4**
Jeśli NIE przejdź do punktu nr 4,
Jeśli TAK przejdź do następnego punktu.
8. Wyprowadź wynik sumy
9. Zakończ algorytm

Opis algorytmu sumowania N liczb (pętla)

1. Zacznij algorytm
2. Suma jest równa 0
3. Licznik jest równy 0
4. **Wprowadź wartość N** *// Dodatkowa zmienna: N*
5. **Do poprzedniej wartości licznika dodaj 1**
6. **Wprowadź liczbę**
7. **Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
8. **Sprawdź czy licznik równa się N**
Jeśli NIE przejdź do punktu nr 5,
Jeśli TAK przejdź do następnego punktu.
8. Wyprowadź wynik sumy
9. Zakończ algorytm

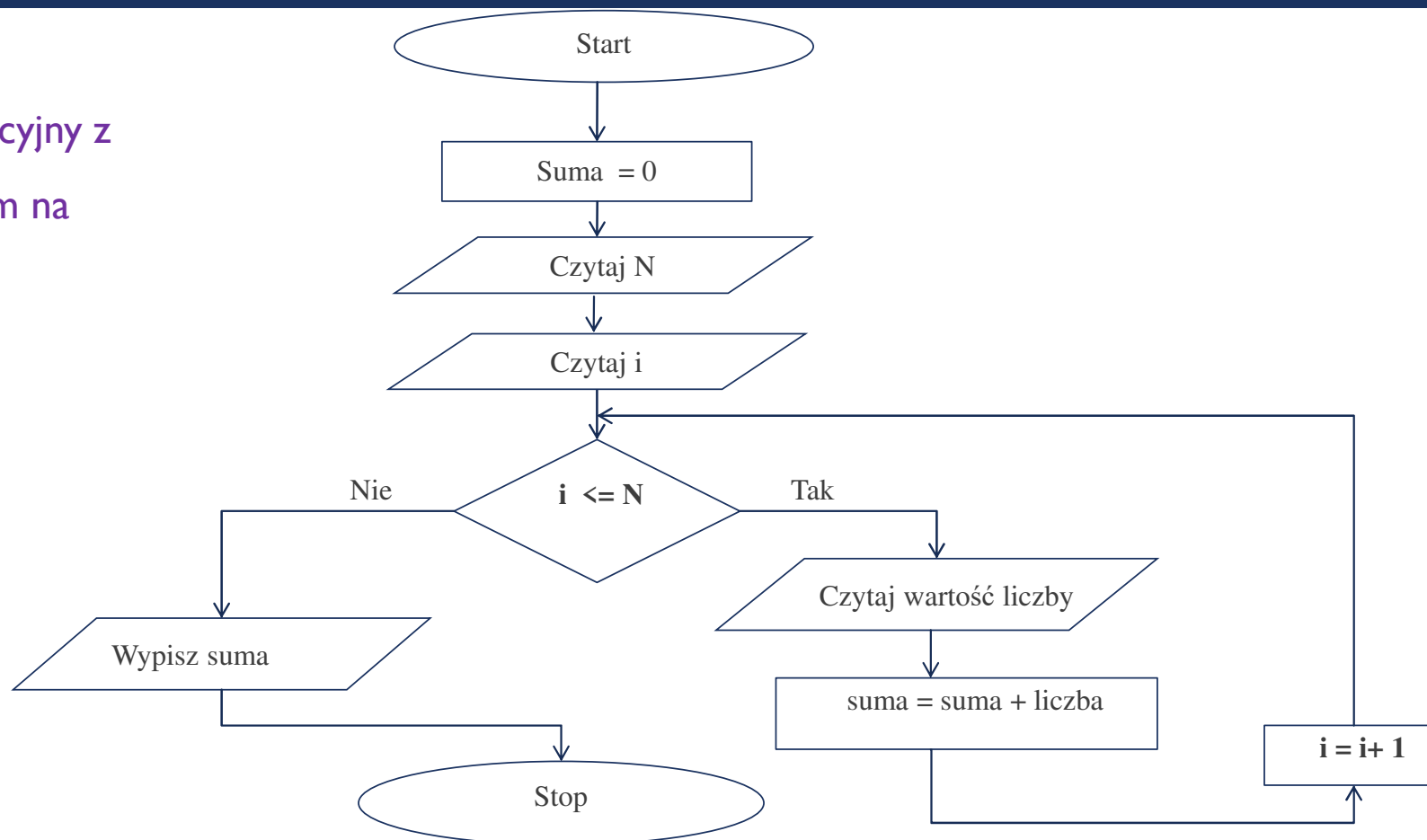
ALGORYTM Z PĘTLĄ (I/2)

- Cykl iteracyjny z warunkiem na końcu.



ALGORYTM Z PĘTLĄ (2/2) - PRZYKŁAD

- Cykl iteracyjny z warunkiem na początku.



OPERACJA PĘTLI (ITERACJI)

- **Algorytm zawierający pętlę (iterację)** - polega na wielokrotnym kolejnym zastosowaniu tego samego algorytmu postępowania, przy czym wynik poprzedniej operacji stanowi dane wejściowe dla kolejnej.
- **Instrukcja pętli (iteracji):**
 - jest to instrukcja, która powoduje wielokrotne wykonywanie pewnych czynności;
 - umożliwia zrealizować pętlę programową, tj. wielokrotnie wykonać określony ciąg instrukcji (operacji);
 - może być używana do różnych zadań (celów), takich jak czytanie danych z klawiatury, porządkowanie danych, wykonywanie obliczeń itp.

INSTRUKCJA ITERACYJNA *FOR* ... (1/2)

for (*wyrażenie początkowe ; wyrażenie logiczne; aktualizacja zmiennej sterującej*)

{

blok instrukcji

}

for (*inicjalizacja ; wyrażenie warunkowe ; wyrażenie modyfikujące*)

Przykład:

for (**int** k=0; k<10; k++)

{

Console.WriteLine (k);

}

INSTRUKCJA ITERACYJNA FOR ... (2/2)

```
for (inicjalizacja1, inicjalizacja2, ..., inicjalizacjaN; wyrażenie_logiczne;  
      aktualizacja_zmiennej_sterującej1, ... aktualizacja_zmiennej_sterującejN)  
{  
    blok instrukcji  
}
```

Przykład:

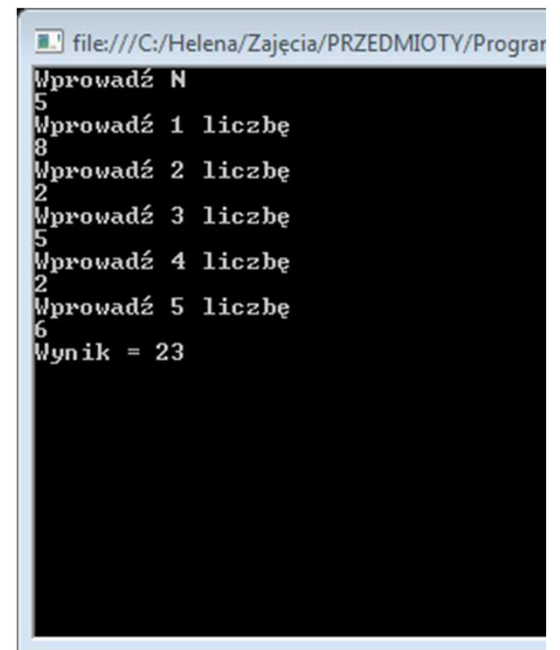
```
for ( int k=0, i= 10; k<i ; k++ , i - - )  
{  
    Console.WriteLine (k);  
}
```

Opis algorytmu sumowania N liczb (pętla)

1. Zacznij algorytm
2. Suma jest równa 0
3. Licznik jest równy 0
4. **Wprowadź wartość N** *// Dodatkowa zmienna: N*
5. **Do poprzedniej wartości licznika dodaj 1**
6. **Wprowadź liczbę**
7. **Do poprzedniego wyniku sumy dodaj wprowadzoną liczbę**
8. **Sprawdź czy licznik równa się N**
Jeśli NIE przejdź do punktu nr 5,
Jeśli TAK przejdź do następnego punktu.
8. Wyprowadź wynik sumy
9. Zakończ algorytm

INSTRUKCJA ITERACYJNA *FOR* ... - PRZYKŁAD

```
int N;  
int liczba, suma=0;  
Console.WriteLine("Wprowadź N");  
N = Int32.Parse(Console.ReadLine());  
for (int i = 1; i <= N; i++)  
{  
    Console.WriteLine("Wprowadź "+ i + " liczbę");  
    liczba = Int32.Parse(Console.ReadLine());  
    suma += liczba;  
}  
Console.WriteLine("Wynik = " + suma.ToString());
```



```
file:///C:/Helena/Zajęcia/PRZEDMIOTY/Program  
Wprowadź N  
5  
Wprowadź 1 liczbę  
8  
Wprowadź 2 liczbę  
2  
Wprowadź 3 liczbę  
5  
Wprowadź 4 liczbę  
2  
Wprowadź 5 liczbę  
6  
Wynik = 23
```

INSTRUKCJA ITERACYJNA *FOR* ... - *SILNIA*

Przykład 1.

```
int wynik = 1;
for (int i = 1; i <= 10; i++)
{
    wynik *= i;
    Console.WriteLine(" Licznik pętli: " + i.ToString() + " Silnia = " + wynik.ToString());
}
Console.WriteLine("Koniec pętli");
```

```
Licznik pętli: 1 Silnia = 1
Licznik pętli: 2 Silnia = 2
Licznik pętli: 3 Silnia = 6
Licznik pętli: 4 Silnia = 24
Licznik pętli: 5 Silnia = 120
Licznik pętli: 6 Silnia = 720
Licznik pętli: 7 Silnia = 5040
Licznik pętli: 8 Silnia = 40320
Licznik pętli: 9 Silnia = 362880
Licznik pętli: 10 Silnia = 3628800
Koniec pętli
```

INSTRUKCJA ITERACYJNA *FOR ... TO ...* (VB)

For *licznik* = *wartość_początkowa* **To** *wartość_końcowa* [**Step** *krok*]

Blok instrukcji

Next [*licznik*]

INSTRUKCJA ITERACYJNA *FOR ...* (PASCAL)

1. *for ... to ...*

{wartość zmiennej sterującej w każdej iteracji jest zwiększana o 1}

for *licznik* **:=** *wartość_początkowa* **to** *wartość_końcowa*
do *instrukcja*;

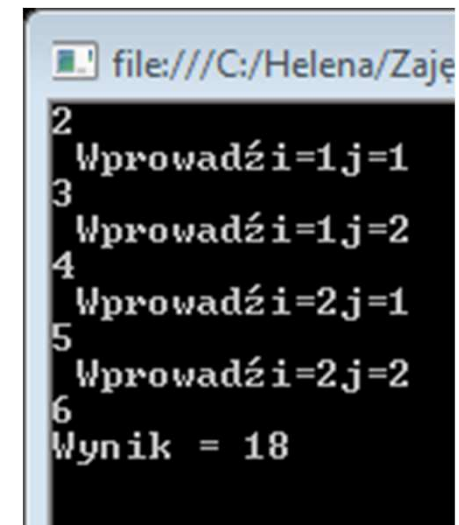
2. *for ... downto ...*

{wartość zmiennej sterującej w każdej iteracji jest zmniejszana o -1}

for *zmienna_sterująca* **:=** *wartość_początkowa* **downto** *wartość_końcowa* **do**
instrukcja;

ZAGNIEŹDŻENIE INSTRUKCJI ITERACYJNEJ FOR ... - PRZYKŁAD

```
double liczba, suma = 0;
int i, j, N;
N = System.Int32.Parse(Console.ReadLine());
for (i = 1; i <= N; i++)
{
    for (j = 1; j <= N; j++)
    {
        Console.WriteLine("Wprowadź i="+i.ToString()+"j="+j.ToString());
        liczba= System.Int32.Parse(Console.ReadLine());
        suma += liczba;
    }
}
Console.WriteLine ("Wynik = " + suma.ToString());
```



```
file:///C:/Helena/Zaję
2
  Wprowadź i=1j=1
3
  Wprowadź i=1j=2
4
  Wprowadź i=2j=1
5
  Wprowadź i=2j=2
6
Wynik = 18
```


ZAGNIEŹDZENIE INSTRUKCJI ITERACYJNEJ *FOR* ... - PRZYKŁAD

```
double suma = 0;
int i, j, N;
N = Int32.Parse(Console.ReadLine());
for (i = 1; i <= N; i++)
{
    for (j = 1; j <= N; j++)
    {
        Console.WriteLine(" Wprowadź i=" + i.ToString() + " j=" + j.ToString());
        suma += Int32.Parse(Console.ReadLine());
    }
}
Console.WriteLine ("Wynik = " + suma.ToString());
suma = 0;
for (i = 1, j = 1; (i <= N) && (j<= N); i++, j++ )
{
    Console.WriteLine("Wprowadź i= " + i.ToString() + " j=" + j.ToString());
    suma += Int32.Parse(Console.ReadLine());
}
Console.WriteLine("Wynik = " + suma.ToString());
```

```
2 Wprowadź i=1 j=1
3 Wprowadź i=1 j=2
4 Wprowadź i=2 j=1
5 Wprowadź i=2 j=2
6 Wynik = 18
```

INSTRUKCJA FOR

... —

MODYFIKACJE

(I/4)

for (*wyrażenie początkowe*; *wyrażenie logiczne*;)

{

blok instrukcji

aktualizacja zmiennej sterującej

}

Przykład

```
int j, suma = 0;
```

```
int N = System.Int32.Parse(Console.ReadLine());
```

```
for (j = 1; j <= N; )
```

```
{
```

```
    Console.WriteLine(" Wprowadź j= " + j.ToString());
```

```
    suma += System.Int32.Parse(Console.ReadLine());
```

```
    j++;           //aktualizacja zmiennej sterującej
```

```
}
```

```
Console.WriteLine("Wynik = " + suma.ToString());
```

INSTRUKCJA *FOR*

... —

MODYFIKACJE

(2/4)

wyrażenie początkowe;

for (; *wyrażenie logiczne*;)

{

blok instrukcji

aktualizacja zmiennej sterującej

}

Przykład

```
int j, N, suma = 0;
```

```
N = System.Int32.Parse(Console.ReadLine());
```

```
j = 1;
```

```
for ( ; j <= N ; )
```

```
{
```

```
    Console.WriteLine(" Wprowadź " + " j=" + j.ToString());
```

```
    suma += System.Int32.Parse(Console.ReadLine());
```

```
    j++;           //aktualizacja zmiennej sterującej
```

```
}
```

```
Console.WriteLine("Wynik = " + suma.ToString());
```

INSTRUKCJA FOR

... —

MODYFIKACJE

(3/4)

wyrażenie początkowe;

```
for ( ; wyrażenie logiczne; )    //połączenie wyrażenia modyfikującego z warunkiem
{
    blok instrukcji
}
```

Przykład

```
int j, N, suma = 0;
N = System.Int32.Parse(Console.ReadLine());
j = 0;
for ( ; j++ < N ; ) //zmiana warunku
{
    Console.WriteLine(" Wprowadź j=" + j.ToString());
    suma += System.Int32.Parse(Console.ReadLine());
}
Console.WriteLine("Wynik = " + suma.ToString());
```

INSTRUKCJA *FOR* ... – MODYFIKACJE (4/4)

```
4
Wprowadź j = 1
1
Wprowadź j = 2
2
Wprowadź j = 3
3
Wprowadź j = 4
4
Wynik = 10
```

```
4
Wprowadź j = 1
1
Wprowadź j = 2
2
Wprowadź j = 3
3
Wprowadź j = 4
4
Wprowadź j = 5
5
Wynik = 15
```

Przykład 1

```
for (j = 1; j <= N; )
{
    Console.WriteLine(" Wprowadź j = " + j.ToString());
    suma += System.Int32.Parse(Console.ReadLine());
    j++;
}
Console.WriteLine("Wynik = " + suma.ToString());
```

Przykład 2

```
j = 0;
for (; j++ <= N; )
{
    Console.WriteLine(" Wprowadź j = " + j.ToString());
    suma += System.Int32.Parse(Console.ReadLine());
}
Console.WriteLine("Wynik = " + suma.ToString());
```

PODSTAWOWE INSTRUKCJE STERUJĄCE PROGRAMEM

Instrukcje decyzyjne oraz wyboru:

- instrukcja *if* ...
- instrukcje *switch* ...

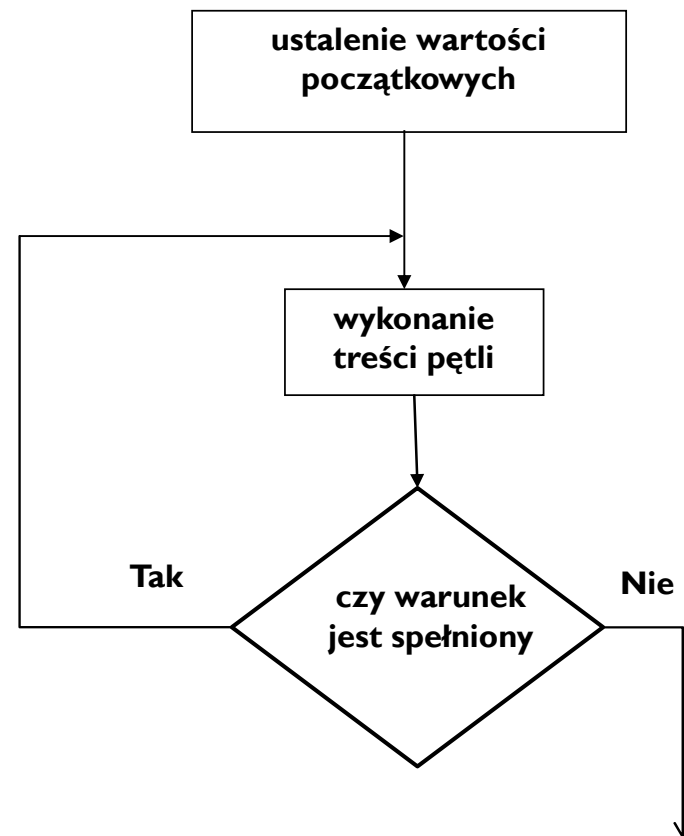
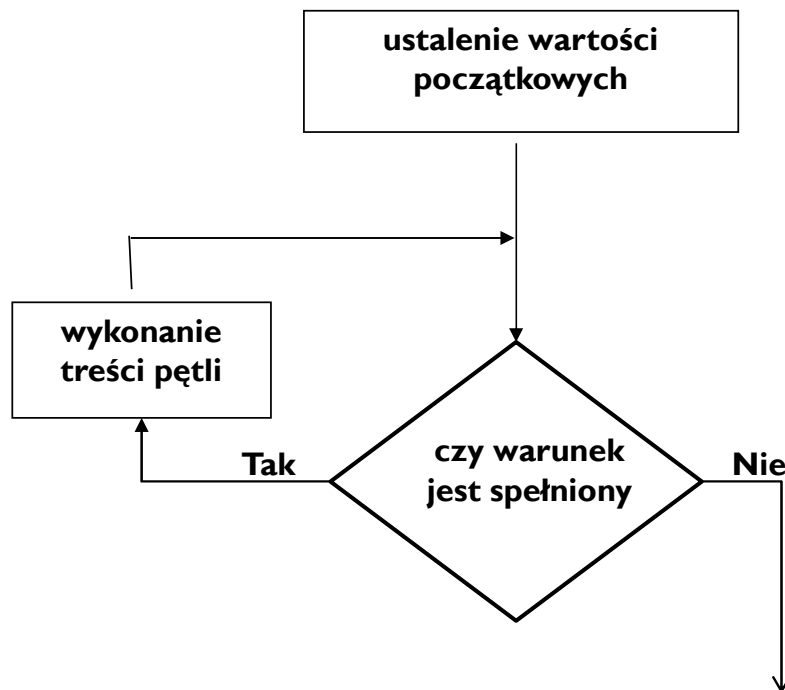
Instrukcje iteracyjne:

- instrukcja *for* ...
- instrukcja *while* ...
- instrukcja *do* ...

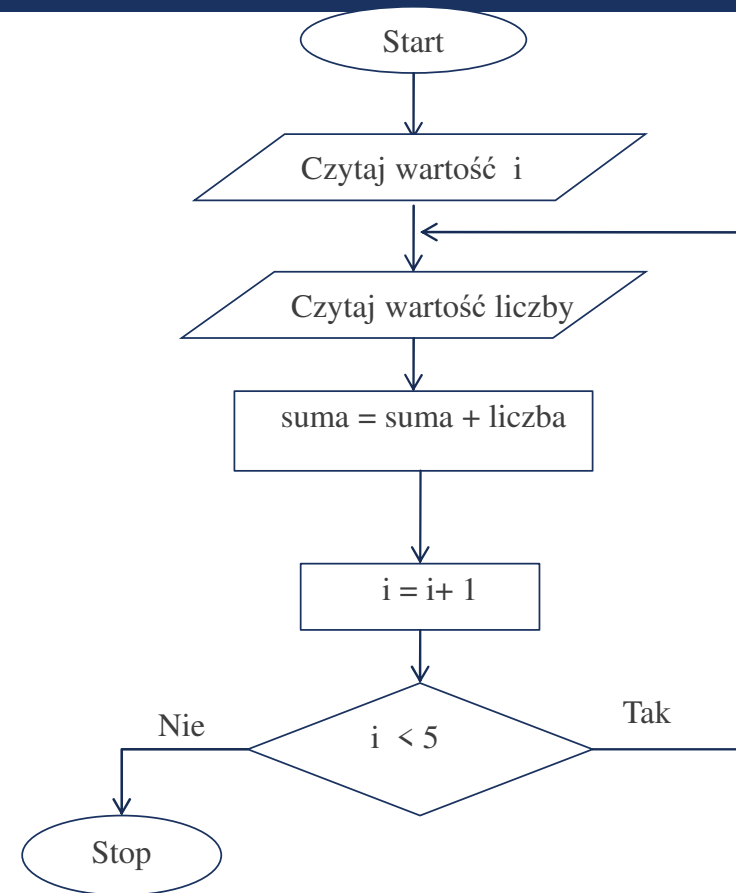
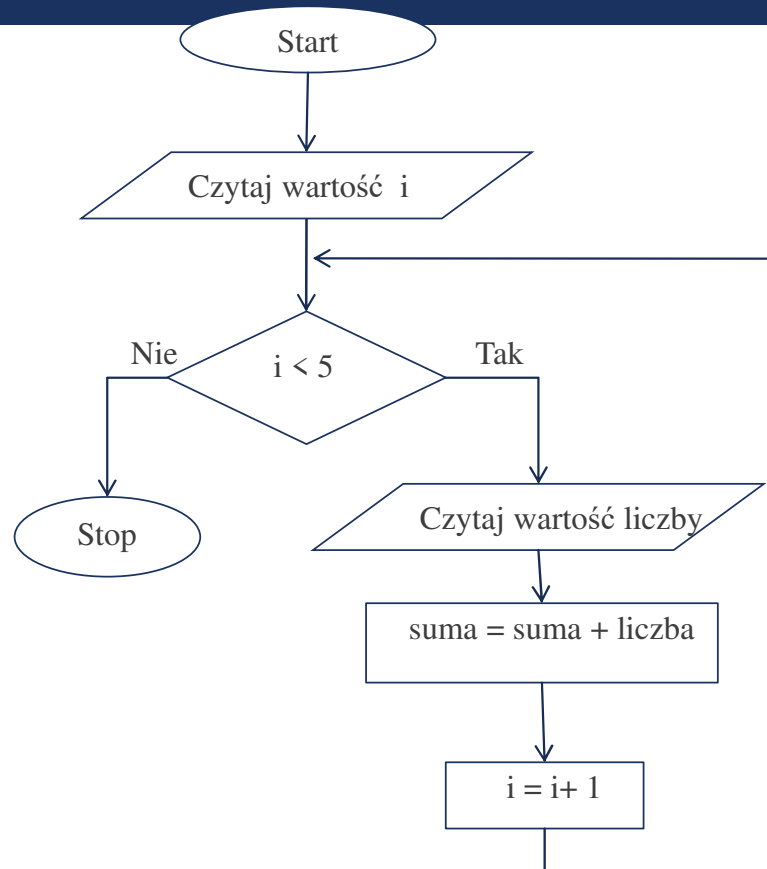
Instrukcja obsługi błędów i wyjątków *try* ...

Inne instrukcje sterujące przebiegiem programu (*continue*; *break*)

Pętle w algorytmach (I/2)

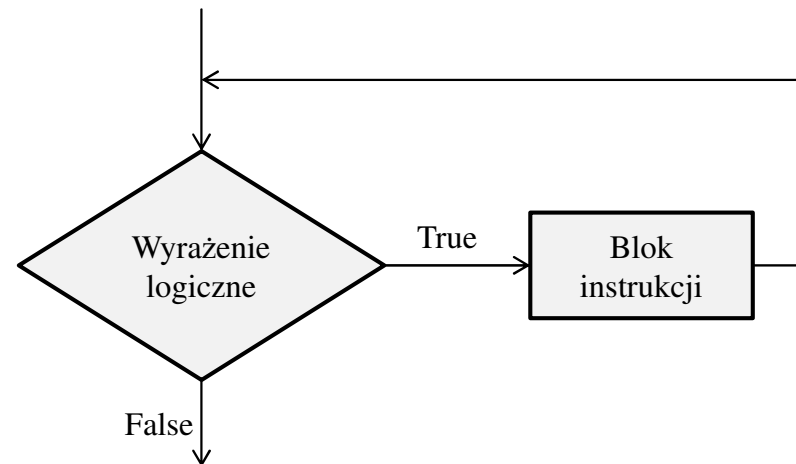


PĘTLE W ALGORYTMACH (2/2)



INSTRUKCJA WHILE ... (1/2)

```
while (wyrażenie_logiczne)  
{  
    blok instrukcji  
}
```



INSTRUKCJA WHILE ... (2/2)

```
int i = 0;
```

```
while (i < 10)
```

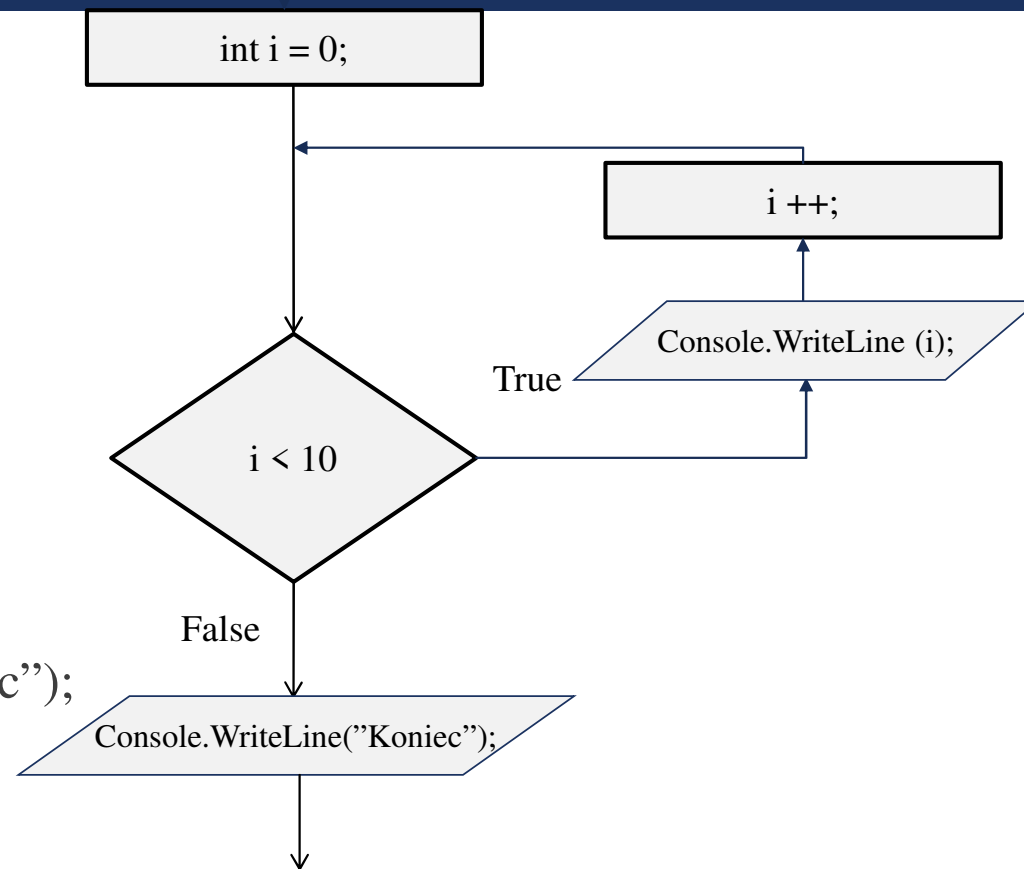
```
{
```

```
    Console.WriteLine (i);
```

```
    i ++;
```

```
}
```

```
Console.WriteLine("Koniec");
```



INSTRUKCJA *WHILE* ... - ZMIENNA POMOCNICZA (STERUJĄCA)

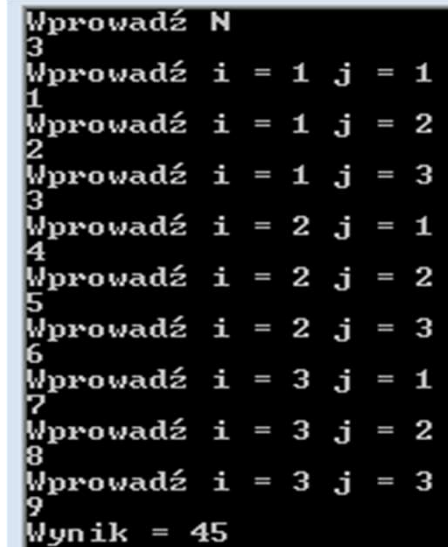
```
while (wyrażenie_logiczne)  
{  
    instrukcje;  
    aktualizacja zmiennej sterującej;  
}
```

INSTRUKCJA ITERACYJNA *WHILE* ... PRZYKŁAD PĘTLI NIEDETERMINISTYCZNEJ

```
int suma = 0, ile = 0;
while (suma < 100)
{
    suma += System.Int32.Parse(Console.ReadLine());
    ile++;
}
Console.WriteLine(„Liczba wprowadzonych jest "+ ile.ToString());
```

ZAGNIEŹDZENIE INSTRUKCJI ITERACYJNEJ WHILE ... - PRZYKŁAD

```
double liczba, suma = 0;
int i = 1, j, N;
Console.WriteLine("Wprowadź N");
N = System.Int32.Parse(Console.ReadLine());
while (i <= N)
{
    j = 1;
    while (j <= N)
    {
        Console.WriteLine("Wprowadź i = {0} j = {1}", i, j);
        //Console.WriteLine(" Wprowadź i= "+i.ToString()+" j= "+j.ToString());
        liczba = System.Int32.Parse(Console.ReadLine());
        suma += liczba;
        j++;
    }
    i++;
}
Console.WriteLine("Wynik = " + suma.ToString("#.##));
```



```
Wprowadź N
3
Wprowadź i = 1 j = 1
1
Wprowadź i = 1 j = 2
2
Wprowadź i = 1 j = 3
3
Wprowadź i = 2 j = 1
4
Wprowadź i = 2 j = 2
5
Wprowadź i = 2 j = 3
6
Wprowadź i = 3 j = 1
7
Wprowadź i = 3 j = 2
8
Wprowadź i = 3 j = 3
9
Wynik = 45
```

PORÓWNANIE *FOR* ORAZ *WHILE*

| Pętla <i>for</i> | Pętla <i>while</i> |
|---|--|
| <pre>for (int i = 1; i<10; i++) { Console.WriteLine (i); }</pre> | <pre>int i = 1; while (i < 10) { Console.WriteLine (i); i ++; }</pre> |

INSTRUKCJA WHILE ... – MODYFIKACJE

```
while ( wyrażenie logiczne ) //połączenie wyrażenia modyfikującego z warunkiem  
{  
    blok instrukcji  
}
```

Przykład

```
j = 0;  
while (j++ <= N)  
{  
    Console.WriteLine(" Wprowadź j= " + j.ToString());  
    suma += System.Int32.Parse(Console.ReadLine());  
}  
Console.WriteLine("Wynik = " + suma.ToString());
```

INSTRUKCJA *DO ... WHILE*

do

{

blok instrukcji

}

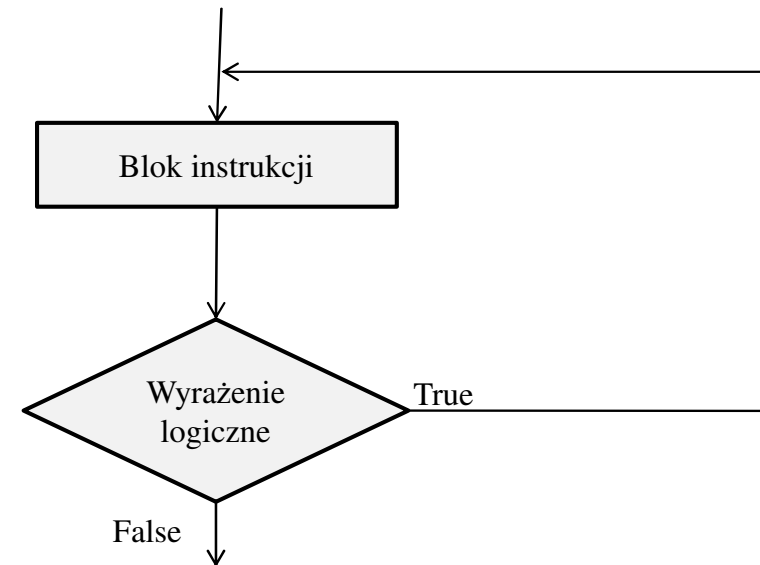
while (*wyrażenie_logiczne*);

Można również:

do {

blok instrukcji

} while (*wyrażenie_logiczne*);



INSTRUKCJA ITERACYJNA *WHILE / DO W VB*

While *warunek*

Blok instrukcji

End While

Do

Blok instrukcji

Loop While *warunek*

INSTRUKCJE NIEDETERMINISTYCZNE W JĘZYKU PASCAL

1. Warunek sprawdzany na początku pętli:

while *wyrażenie_warunkowe* **do** *instrukcja*;

2. Warunek sprawdzany na końcu pętli:

repeat

instrukcja_1;

instrukcja_2;

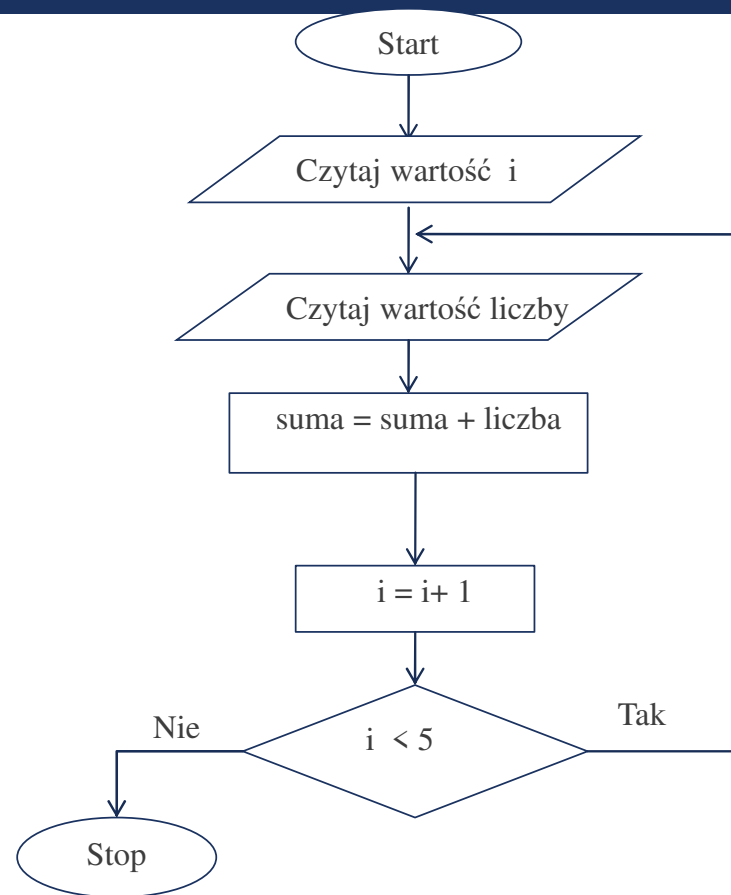
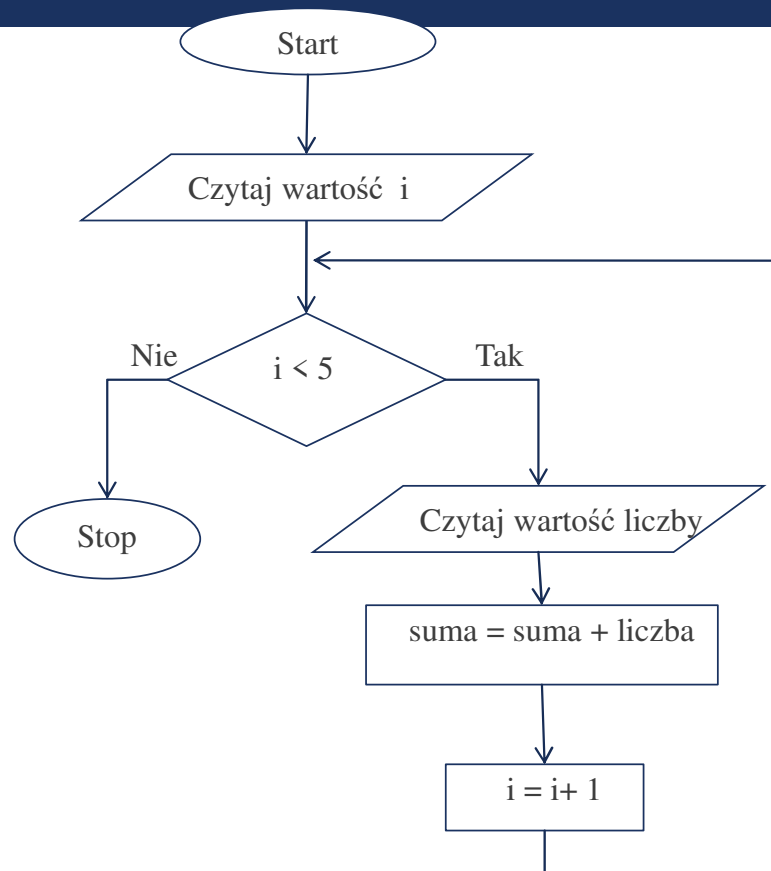
...

instrukcja_n;

until *wyrażenie_warunkowe*;

WHILE ...

DO ... WHILE



PORÓWNANIE: *WHILE* ... ORAZ *DO ... WHILE* (1/2)

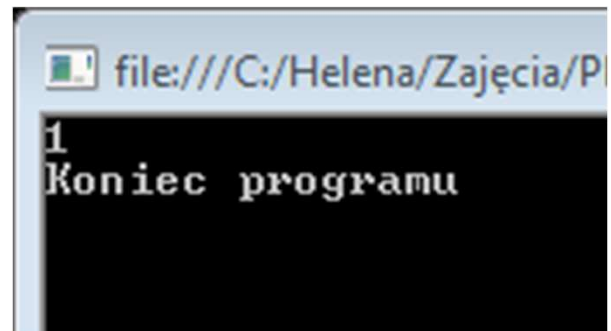
Czy pojawi się komunikat z wartością liczbową?

- Instrukcja *while*

```
int liczba = 1;
while (liczba < 0)
{
    Console.WriteLine (liczba);
}
Console.WriteLine ("Koniec programu");
```

- Instrukcja *do ... while*

```
int liczba= 1;
do
{
    Console.WriteLine (liczba);
} while (liczba < 0);
Console.WriteLine ("Koniec programu");
```



PORÓWNIANIE: *WHILE* ... ORAZ *DO ... WHILE* (2/2)

| Konstrukcja | Pętla jest wykonywana, gdy warunek | Sprawdzenie warunku następuje |
|--|------------------------------------|----------------------------------|
| while (warunek) { blok_instrukcji; } | jest prawdziwy | przed wykonaniem iteracji |
| do { blok_instrukcji; } while (warunek); | jest prawdziwy | po wykonaniu iteracji |

METODA TRYPARSE

Zadanie metody: przekształcenie danych z typu **string** na typ dla którego wywołujemy metodę (np. **int**) z równoczesnym zapisaniem w zmiennej typu logicznego jednej z dwóch wartości:

true – gdy w zmiennej typu *string* były same cyfry, oznacza to, że konwersja powiodła się, nastąpiła bez błędów, do zmiennej typu *integer* wstawiono wartość liczbową, która była zmiennej typu *string*;

false – gdy w zmiennej typu *string* znajdował się przynajmniej jeden dowolny znak inny niż cyfra, oznacza to, że konwersja nie powiodła się, natomiast do zmiennej typu *integer* wstawiono wartość zero.

METODA TRYPARSE - PRZYKŁAD

Przykład fragmentu programu ilustrujący zastosowanie metody TryParse:

```
string dane;  
dane = Console.ReadLine();  
int liczba;  
bool wynikKonwersji;  
wynikKonwersji = int.TryParse(dane, out liczba);  
if (wynikKonwersji)                                // if (wynikKonwersji == true)  
{  
    // wprowadzone liczbę  
}  
else  
{  
    // wprowadzone dowolny znak niebędący cyfrą (wartością liczbową)  
}
```

PRZYKŁAD FRAGMENTU PROGRAMU ILUSTRUJĄCY CZYTANIE DANYCH, DOPÓKI UŻYTKOWNIK NIE WPROWADZI DOWOLNEJ LITERY:

```
string dane;  
int liczba;  
bool wynikKonwersji;  
do  
{  
    Console.WriteLine ("Wprowadź liczbę:");  
    dane = Console.ReadLine();  
    wynikKonwersji = int.TryParse(dane, out liczba);  
    if (wynikKonwersji)  
    {  
        // wprowadzono cyfry - wykonujemy operacje wynikające z treści zadania  
    }  
}  
while (wynikKonwersji);
```


ZASTOSOWANIE INSTRUKCJI DO... ORAZ WHILE...

Wariant I:

```
string dane;  
int n, suma=0, ilosc=0;  
bool kon;  
do  
{  
    dane = Console.ReadLine();  
    kon=int.TryParse(dane, out n);  
    if (kon)  
    {  
        suma += n;  
        ilosc ++;  
    }  
}  
while (kon);
```

Wariant II:

```
string dane;  
int n, suma=0, ilosc=0;  
bool kon;  
dane = Console.ReadLine();  
kon=int.TryParse(dane, out n);  
while (kon);  
{  
    suma += n;  
    ilosc ++;  
    dane = Console.ReadLine();  
    kon=int.TryParse(dane, out n);  
}
```

PODSTAWOWE INSTRUKCJE STERUJĄCE PROGRAMEM

Instrukcje decyzyjne oraz wyboru:

- instrukcja *if* ...
- instrukcje *switch* ...

Instrukcje iteracyjne:

- instrukcja *for* ...
- instrukcja *while* ...
- instrukcja *do* ...

Instrukcja obsługi błędów i wyjątków *try* ...

Inne instrukcje sterujące przebiegiem programu (*continue*; *break*)

INNE INSTRUKCJE STERUJĄCE PRZEBIEGIEM PROGRAMU

break

continue

go to

INSTRUKCJE *BREAK* ORAZ *CONTINUE*

- Wyjście (opuszczenie danego bloku instrukcji):

break;

- Przejście do wykonania następnej iteracji:

continue;

INSTRUKCJE *BREAK – PRZYKŁAD* (1/3)

```
N = System.Int32.Parse(Console.ReadLine());  
for (j = 1; ; j++ )  
{  
    Console.WriteLine(" Wprowadź j = " + j.ToString());  
    suma += System.Int32.Parse(Console.ReadLine());  
    if (j == N)  
    {  
        break;  
    }  
}  
Console.WriteLine("Wynik = " + suma.ToString());
```

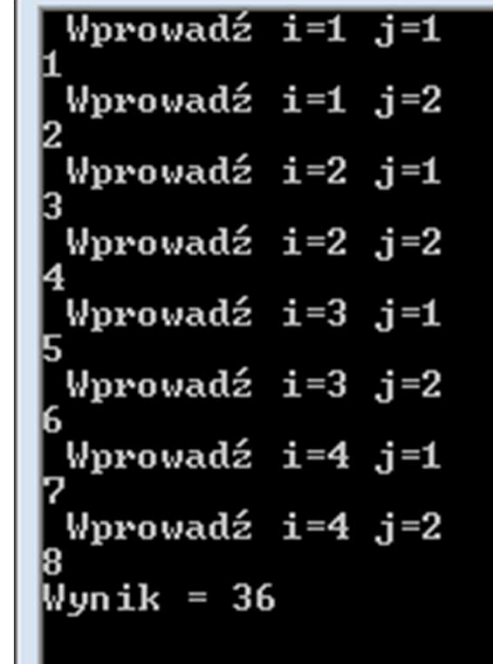
```
4  
Wprowadź j = 1  
1  
Wprowadź j = 2  
2  
Wprowadź j = 3  
3  
Wprowadź j = 4  
4  
Wynik = 10
```

INSTRUKCJE *IF* I FOR – BEZ WARUNKÓW, UŻYCIE *BREAK* (2/3)

```
N = System.Int32.Parse(Console.ReadLine());
j = 1;
for ( ; ; )
{
    Console.WriteLine(" Wprowadź j = " + j.ToString());
    suma += System.Int32.Parse(Console.ReadLine());
    if (j == N)
    {
        break;
    }
    j++;
}
Console.WriteLine("Wynik = " + suma.ToString());
```

ZAGNIEŹDZENIE INSTRUKCJI ITERACYJNEJ FOR - BREAK (3/3)

```
for (i = 1; i <= 4; i++)  
{  
    for (j = 1; j <= 4; j++)  
    {  
        Console.WriteLine("Wprowadź i =" + i.ToString() + " j=" + j.ToString());  
        liczba = System.Int32.Parse(Console.ReadLine());  
        suma += liczba;  
        if (j == 2)  
        {  
            break;  
        }  
    }  
}  
  
Console.WriteLine ("Wynik = " + suma.ToString());
```



```
Wprowadź i=1 j=1  
1 Wprowadź i=1 j=2  
2 Wprowadź i=2 j=1  
3 Wprowadź i=2 j=2  
4 Wprowadź i=3 j=1  
5 Wprowadź i=3 j=2  
6 Wprowadź i=4 j=1  
7 Wprowadź i=4 j=2  
8  
Wynik = 36
```

INSTRUKCJE *CONTINUE* – PRZYKŁAD

```
for (int i = 1; i <= 10; i++)  
{  
    if (i % 2 != 0)  
    {  
        continue;  
    }  
    Console.WriteLine("Liczba {0} parzysta" , i);  
}  
Console.WriteLine("Koniec");
```

```
-----  
for (int i = 1; i <= 10; i++)  
{  
    if (i % 2 != 0) continue;  
    Console.WriteLine("Liczba {0} parzysta" , i);  
}  
Console.WriteLine("Koniec");
```

```
Liczba 2 parzysta  
Liczba 4 parzysta  
Liczba 6 parzysta  
Liczba 8 parzysta  
Liczba 10 parzysta  
Koniec
```


INSTRUKCJA GOTO



goto *etykieta*
do etykiety (nazwaEtykiety)

– *przejdźcie*



goto case *wyrażenie Stałe* –
przejdźcie do wybranego bloku case



goto default
przejdźcie do wybranego bloku

–

INSTRUKCJA GOTO – PRZYKŁAD

```
Wprowadź liczbę -2
5
Wprowadź liczbę -2
7
Wprowadź liczbę -2
4
Wprowadź liczbę -2
0
Suma = 16
```

```
Wprowadź liczbę -1
5
Wprowadź liczbę -1
7
Wprowadź liczbę -1
4
Wprowadź liczbę -1
0
Suma = 16
```

Przykład 1:

```
do
{
    Console.WriteLine ("Wprowadź liczbę");
    liczba = System.Int32.Parse(Console.ReadLine());
    suma += liczba;
} while (liczba != 0);
Console.WriteLine ("Suma = " + suma.ToString());
```

Przykład 2:

```
czytaj: Console.WriteLine ("Wprowadź liczbę");
        liczba = System.Int32.Parse(Console.ReadLine());
        suma += liczba;
        if (liczba != 0) goto czytaj;
        Console.WriteLine ("Suma = " + suma.ToString());
```

PODSUMOWANIE

- pętli **while** używamy, gdy chcemy, aby blok kodu zawarty w pętli wykonywał się **dopóki warunek w niej zawarty będzie prawdziwy**.
- pętli **do while** używamy, gdy chcemy, aby blok kodu zawarty w pętli wykonał się **przynajmniej raz**.
- pętli **for** używamy, gdy chcemy, aby blok kodu zawarty w pętli wykonał się **określoną liczbę razy**.



DO OBEJRZENIA

- <https://www.youtube.com/watch?v=5t8Bzamyv3o>



DZIĘKUJĘ ZA
UWAGĘ!

RADOSLAW.WOJTOWICZ@UE.WROC.PL