

# Wykład 4: Instrukcje sterujące.

dr inż. Andrzej Stafiniak

*Wrocław 2023*



HR EXCELLENCE IN RESEARCH



Politechnika Wroclawska

# Wyrażenie vs. instrukcja

**Wyrażenie** to kombinacja różnych operatorów i operandów (argumentów) dająca wartość, którą można gdzieś przypisać, podstawić do funkcji, porównać itp...

```
x = 15
x + 10
z = x / y
(x > y) && (x == y)
```

**Instrukcja** jest to polecenie przekazane do wykonania, może nią być m. in. każde wyrażenie zakończone średnikiem.

```
x = 15;
y = x + 10;
z = x / y;
```

**if** (.....)

**Ale:** instrukcja warunkowa

..... ? x : y

wyrażenie warunkowe

# Instrukcje sterujące

## Instrukcje warunkowe:

➤ `if, if else`

➤ `switch`

## Pętle:

➤ `while`

➤ `for`

➤ `do while`

Instrukcje `break` i `continue`

Instrukcja  
`exit();`

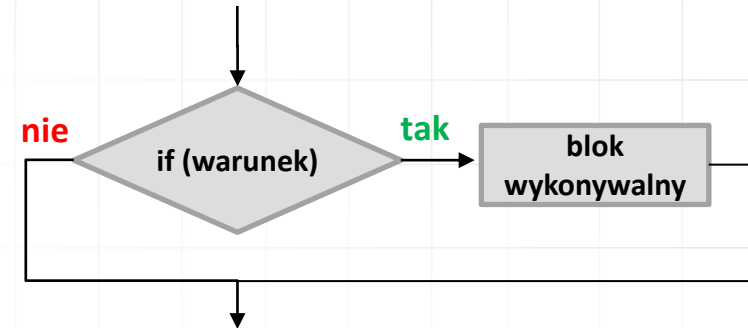
Instrukcja  
`goto`

z kodem wyjścia  
jako argumentem:  
`EXIT_SUCCESS` i  
`EXIT_FAILURE`

# Instrukcje warunkowe

## Instrukcja warunkowa **if**

Schemat blokowy

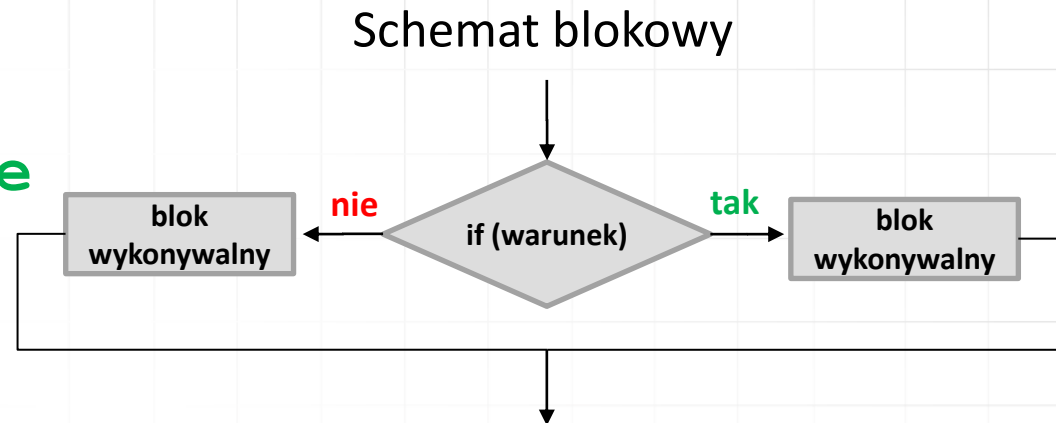


```
if (wyrażenie) {           //blok z instrukcjami do wykonania,  
    instrukcje;             //jeśli wyrażenie jest prawdziwe  
}
```

W języku C/C++ **wyrażenie** (np. **a>0** , **a** , **a==0** , **!a**) jest **prawdziwe**, jeżeli jego wartość jest różna od 0. Wyrażenie wykorzystujące operator przypisania ma wartość przypisania np. **a=2** ma wartość 2.

# Instrukcje warunkowe

## Instrukcja warunkowa **if else**

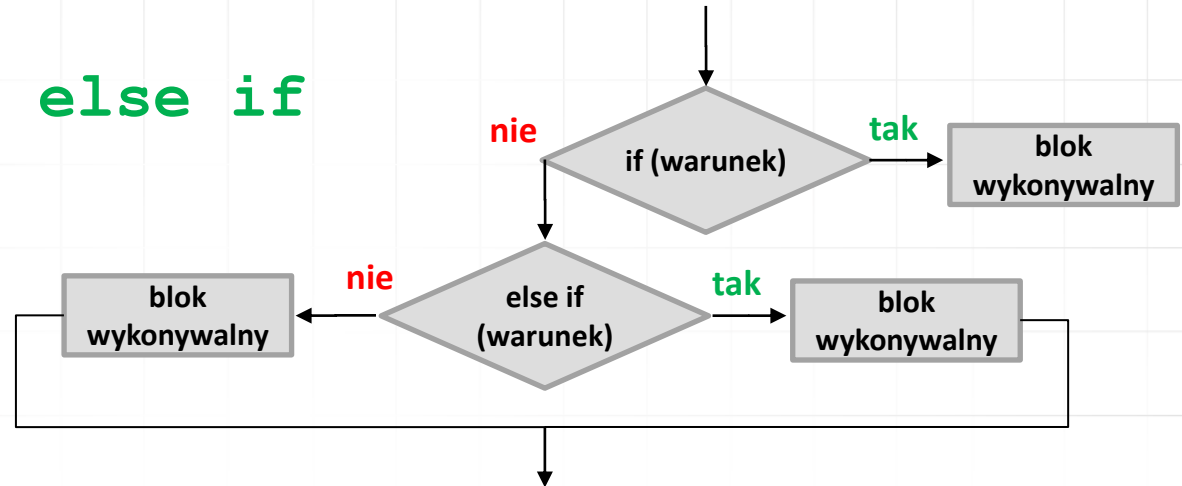


```
if (wyrażenie) {  
    instrukcje; //blok z instrukcjami do wykonania, jeśli  
                //wyrażenie jest prawdziwe  
}  
else {  
    instrukcje; //blok z instrukcjami do wykonania, jeśli  
                //wyrażenie nie jest prawdziwe  
}
```

# Instrukcje warunkowe

## Instrukcja warunkowa **if else if**

### Schemat blokowy



```
if (wyrażenie1) {  
    instrukcje;           //blok z instrukcjami do wykonania, jeśli  
                           //wyrażenie1 jest prawdziwe  
} else if (wyrażenie2) {  
    instrukcje;           //blok z instrukcjami do wykonania, jeśli  
                           //wyrażenie2 jest prawdziwe  
} else {  
    instrukcje;           //blok z instrukcjami do wykonania, jeśli  
                           //wyrażenia nie są prawdziwe  
}
```

# Instrukcje warunkowe

## Zagnieżdżanie instrukcji

```
#include <stdio.h>
#include <math.h> // dla instrukcji sqrt()
int main(){
    float a,b,c,delta; //wspolczynniki rownania kw.
    float x1,x2; //pierwiastki równania
    printf("ax^2 +bx +c , podaj a,b,c \n");
    scanf("%f %f %f",&a,&b,&c);
    delta = b*b - 4*a*c;
    if (delta<=0){
        if (delta==0){
            x1=x2=-b/(2*a);
            printf("x1 = x2 = %f \n",x1);
            break; // wyjście z if'a
        }
        printf("Delta ujemna \n");
    } else {
        x1 =( -b + sqrt(delta))/(2*a);
        x2 =( -b - sqrt(delta))/(2*a);
        printf("\nx1= %f x2= %f",x1,x2);
    }
    return 0;
}
```

# Instrukcje warunkowe

## Instrukcja wielokrotnego wyboru **switch**

```
switch (wyrażenie) {  
    case wartość1:    //instrukcje do wykonania,  
        instrukcje;    //dla wyrażenie==wartość1  
  
    case wartość2:    //instrukcje do wykonania,  
        instrukcje;    //dla wyrażenie==wartość2  
  
    case wartość3:    //instrukcje do wykonania,  
        instrukcje;    //dla wyrażenie==wartość3  
    ...  
}
```

**Wyrażenie** – argument instrukcji **switch**, w postaci wyrażenia typu **int** lub **char** (liczba całkowita). Natomiast z każdym słowem kluczowym **case** znajduje się **wartość** stałe tego samego typu co **wyrażenie**.



# Instrukcje warunkowe

## Instrukcja wielokrotnego wyboru **switch**

```
switch (wyrażenie) {  
    case wartość1:      //instrukcje do wykonania,  
        instrukcje;      //dla wyrażenie==wartość1  
        break;          // przerwij instrukcję switch  
  
    case wartość2:      //instrukcje do wykonania,  
        instrukcje;      //dla wyrażenie==wartość2  
        break;          // przerwij instrukcję switch  
  
    case wartość3:      //instrukcje do wykonania,  
        instrukcje;      //dla wyrażenie==wartość3  
        break;          // przerwij instrukcję switch  
  
    . . .  
}
```

# Instrukcje warunkowe

## Instrukcja wielokrotnego wyboru **switch**

```
switch (wyrażenie) {  
    case wartość1:           //instrukcje do wykonania,  
        instrukcje;           //dla wyrażenie==wartość1  
        break;               // przerwij instrukcję switch  
  
    case wartość2:           //instrukcje do wykonania,  
        instrukcje;           //dla wyrażenie==wartość2  
        break;               //przerwij instrukcję switch  
  
    ...  
  
    default:                 //w przypadku gdy wyrażenie nie  
        instrukcje;           //odpowiada żadnej wartości  
        break;  
}
```

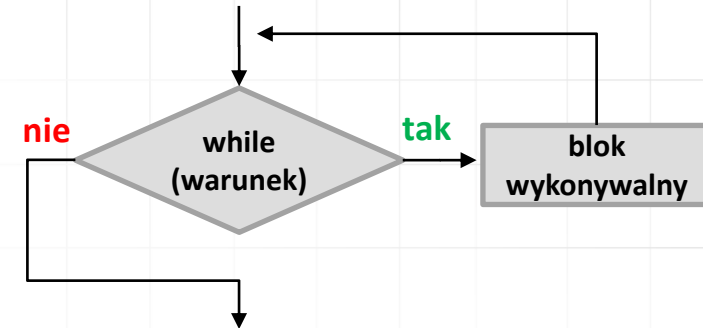
# Instrukcje powtarzania – pętle

## Instrukcja `while()`

```
while (wyrażenie) {  
    instrukcje; //instrukcje do wykonania,  
}
```

```
int i = 0;  
while (i<=10) {  
    printf("%d\n", i*i); //wypisz kolejne kwadraty liczby i  
    ++i;                //zwiększamy i o + 1  
}
```

## Schemat blokowy

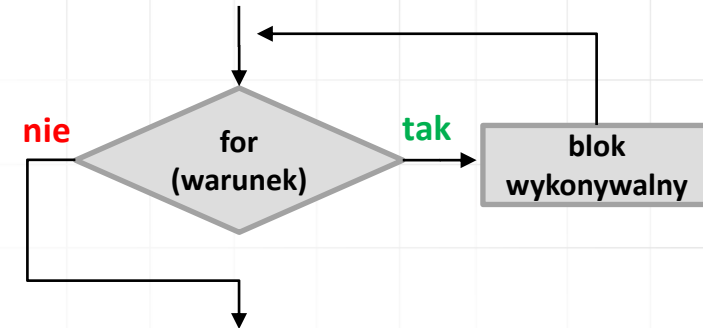


# Instrukcje powtarzania – pętle

## Instrukcja `for()`

```
for (wyr1; wyr2; wyr3) {  
    instrukcje; //instrukcje do wykonania,  
}
```

Schemat blokowy



(**wyr1** - ustawienia początkowej wartości zmiennej sterującej - licznika; **wyr2** – warunku wyjścia z pętli; **wyr3** - zmiana licznika w kolejnych iteracjach.)

(**a=b1; a<b2; a=a+da**)

C90

```
for (int i=10; i>0; --i) {  
    printf("%d\n", i*i); //wypisz kolejne kwadraty zmiennej i  
}
```

# Instrukcje powtarzania – pętle

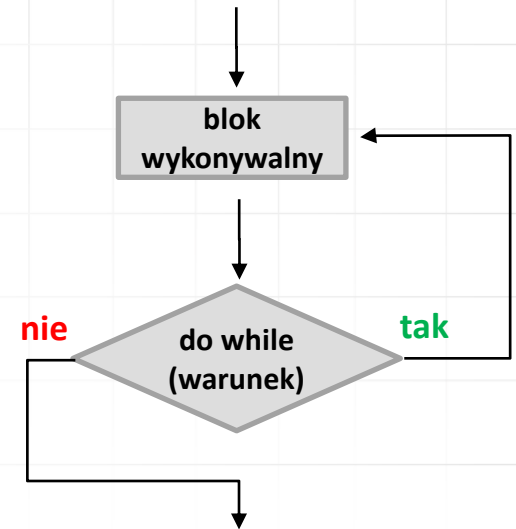
## Instrukcja `do while()`

```
do {  
    instrukcje; //instrukcje do wykonania  
} while (wyrażenie)
```

```
int i = 0;
```

```
do {  
    printf("%d\n", i*i); //wypisz kolejne kwadraty zmiennej i  
    ++i;                //zwiększamy i o + 1  
} while (i<=10)
```

## Schemat blokowy



# Instrukcje powtarzania – pętle

Pętle nieskończone – warunek cały czas spełniony

```
while (1) {           for (;;) { //(;1;), (a,a,a)           do {  
    instrukcje;        instrukcje;                instrukcje;  
}                       }                           } while (1)
```


Ważne aby w pętlach nieskończonych ustawić bezpiecznik wyjścia np. przez instrukcję przerwania pętli **break**.

```
for ( ; ; ++i ) {  
    instrukcje;  
    if (i>5)  
        break;  
}
```

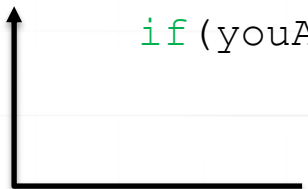
# Instrukcje powtarzania – pętle

## Pętle zagnieżdżone - **break** vs. **continue**

```
for (int i=1; i<=7; ++i) {  
    for (int j=0; j<=24; ++j) {  
        if (yourBattery>99) {  
            break;  
        }  
        sleep();  
    }  
    printf ("Next day \n");  
}
```



```
for (int i=0; i<=7; ++i) {  
    for (int j=0; j<=24; ++j) {  
        if (youAreHungry==1) {  
            eat();  
            continue;  
        }  
        sleep();  
    }  
    printf ("Next day \n");  
}
```



# Instrukcje powtarzania – pętle

## Pętla `for()` – popularne implementacje

```
for (int i=0; i<7; ++i)
```

```
for (int i=10; i>0; --i)
```

```
for (int i=5; i<=55; i+=5)
```

```
for (znak='a'; znak<='z'; ++znak)
```

```
for (int i=0; (2+i)*i<101; ++i)
```

```
for (int a=0, b=0; a+b<99; a++)  
    scanf("%d", &b)
```

```
for (int i=5; j<=85; j=20+(++i*5))
```

```
for (i=strlen(s)-1, j=0; i>=0; j++, i--)  
    r[i]=s[j];
```