

Programowanie w języku C

Politechnika Białostocka - Wydział Elektryczny

Elektronika i telekomunikacja, semestr III,
studia niestacjonarne II stopnia

Rok akademicki 2024/2025

Pracownia nr 4
(30.12.2024)

dr inż. Jarosław Forenc

Instrukcja switch

```
switch (wyrażenie)
{
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    case wyrażenie_stałe: instrukcje;
    ...
    default: instrukcje;
}
```

Instrukcja switch - wyrażenie stałe

- stała liczbowa, np. 3, 5, 9
- znak w apostrofach, np. 'a', 'z', '+'
- stała zdefiniowana przez #define lub const

```
#include <stdio.h>  
#define N 10
```

```
int main()  
{  
    const int n = 10;  
    ...  
}
```

Przykład - ocena

```
#include <stdio.h>

int main()
{
    int ocena;

    printf("Podaj ocene:");
    scanf("%d",&ocena);
```

Przykład - ocena

```
switch (ocena)
{
case 5: printf("Twoja ocena: bardzo dobry\n");
        break;
case 4: printf("Twoja ocena: dobry\n");
        break;
case 3: printf("Twoja ocena: dostateczny\n");
        break;
case 2: printf("Twoja ocena: niedostateczny\n");
        break;
default: printf("Bledna ocena\n");
}
```

Przykład - ocena

```
switch (ocena)
{
case 5: printf("Twoja ocena: bardzo dobry\n");
        break;
case 4: printf("Twoja ocena: dobry\n");
        break;
case 3: printf("Twoja ocena: dostateczny\n");
        break;
case 2: printf("Twoja ocena: niedostateczny\n");
        break;
default: printf("Bledna ocena\n");
}
```

Podaj ocene: 4
Twoja ocena: dobry

Przykład - ocena

```
switch (ocena)
{
case 5: case 4: case 3:
    printf("Ocena pozytywna\n");
    break;
case 2: printf("Ocena negatywna\n");
    break;
default: printf("Bledna ocena\n");
}
```

Przykład - ocena

```
switch (ocena)
{
case 5: case 4: case 3:
    printf("Ocena pozytywna\n");
    break;
case 2: printf("Ocena negatywna\n");
    break;
default: printf("Bledna ocena\n");
}
```

Podaj ocene: 4
Ocena pozytywna

Przykład - ocena (bez break)

```
switch (ocena)
{
case 5: printf("Twoja ocena: bardzo dobry\n");
case 4: printf("Twoja ocena: dobry\n");
case 3: printf("Twoja ocena: dostateczny\n");
case 2: printf("Twoja ocena: niedostateczny\n");
default: printf("Bledna ocena\n");
}
```

Przykład - ocena (bez break)

```
switch (ocena)
{
case 5: printf("Tw
case 4: printf("Tw
case 3: printf("Tw
case 2: printf("Tw
default: printf("Bledna ocena\n");
}
```

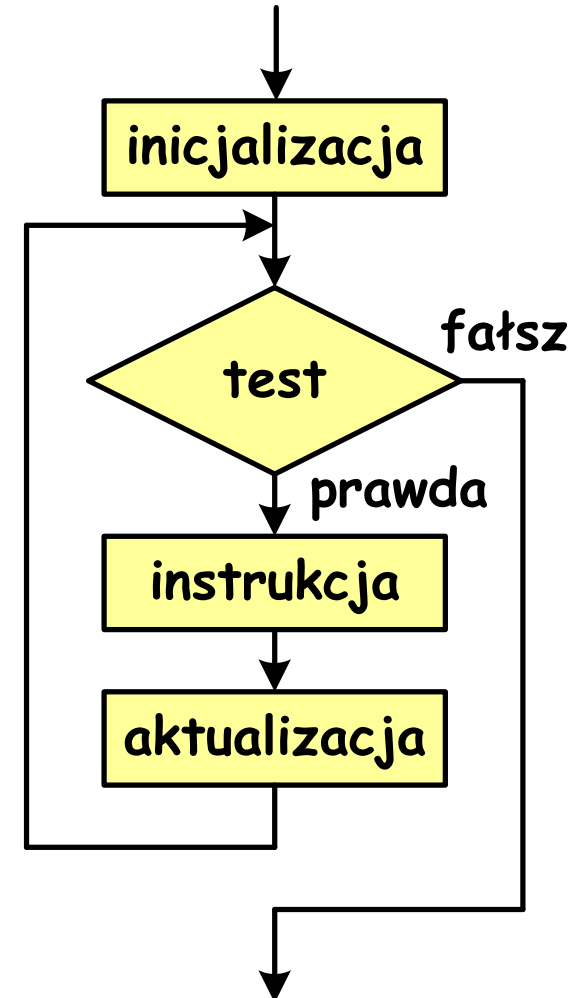
Podaj ocene: 4
Twoja ocena: dobry
Twoja ocena: dostateczny
Twoja ocena: niedostateczny
Bledna ocena

Pętla for

```
for (i=0; i<10; i=i+1)  
instrukcja;
```

- instrukcja zostanie wykonana 10 razy

```
for (inicjalizacja; test; aktualizacja)  
instrukcja;
```



Przykład - wyświetlenie tekstu

```
#include <stdio.h>

int main()
{
    int i;

    for (i=0; i<5; i=i+1)
        printf("Programowanie nie jest trudne\n");

    return 0;
}
```

Przykład - wyświetlenie tekstu

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int i;
```

```
    for (i=0; i<5; i=i+1)
```

```
        printf("Programowanie nie jest trudne\n");
```

```
    return 0;
```

```
}
```

Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne

Pętla for

```
for (i=0; i<10; i=i+1)  
    printf("%d ",i);
```

0 1 2 3 4 5 6 7 8 9

```
for (i=0; i<10; i=i+1)  
    printf("%d ",i+1);
```

1 2 3 4 5 6 7 8 9 10

```
for (i=1; i<=10; i=i+1)  
    printf("%d ",i);
```

1 2 3 4 5 6 7 8 9 10

Pętla for

```
for (i=1; i<=10; i=i+2)  
    printf("%d ",i);
```

1 3 5 7 9

```
for (i=10; i>0; i=i-1)  
    printf("%d ",i);
```

10 9 8 7 6 5 4 3 2 1

```
for (i=-9; i<=9; i=i+3)  
    printf("%d ",i);
```

-9 -6 -3 0 3 6 9

Przykład - suma liczb 1+2+...+10

```
#include <stdio.h>
#define N 10
```

```
int main()
{
```

```
    int i, suma=0;
```

```
    for (i=1; i<=N; i=i+1)
        suma = suma + i;
```

```
    printf("Suma %d liczb to %d\n", N, suma);
    return 0;
```

```
}
```

Suma 10 liczb to 55

Pętla for - instrukcja złożona

- wykonanie w pętli **for** więcej niż jednej instrukcji wymaga umieszczenia ich w dodatkowych nawiasach klamrowych

```
for (wyr1; wyr2; wyr3)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Pętla for - błędy

- średnik na końcu pętli **for**:

```
for (i=0; i<10; i++) ;  
printf("%d ",i);
```

10

- przecinki zamiast średników w pętli **for**:

```
for (i=0, i<10, i++)  
printf("%d ",i);
```

błąd kompilacji

Pętla for - błędy

- błędny warunek - brak wykonania instrukcji:

```
for (i=0; i>10; i++)  
    printf("%d ",i);
```



- błędny warunek - pętla nieskończona:

```
for (i=1; i>0; i++)  
    printf("%d ",i);
```

1 2 3 4 5 6 7 8 9 ...

Instrukcje break i continue

```
for (i=1; i<10; i++)  
{  
    if (i%2==0)  
        continue;  
    if (i%7==0)  
        break;  
    printf("%d\n",i);  
}
```

- **continue** - przerywa bieżącą iterację i przechodzi do kolejnej
- **break** - przerywa wykonywanie pętli for

Operator inkrementacji (++)

++X lub X++

- zwiększa wartość zmiennej o 1

```
++X;
```

=

```
X = X + 1;
```

```
X++;
```

- operator jednoargumentowy
- może być stosowany tylko do zmiennych (nie wolno stosować do wyrażeń)

Operator inkrementacji (++)

++X - operator **pre**inkrementacji

X++ - operator **post**inkrementacji

```
int x = 1, y;  
y = 2 * ++x;
```

++X $x = 2$
2 * ++X $2 * 2$
y = 2 * ++X $y = 4$

$x = 2, y = 4$

```
int x = 1, y;  
y = 2 * x++;
```

2 * x $2 * 1$
y = 2 * x $y = 2$
x++ $x = 2$

$x = 2, y = 2$

Operator dekrementacji (--)

- x - operator **pre**dekrementacji
- x-- - operator **post**dekrementacji

- zmniejsza wartość zmiennej o 1

```
--x;
```

=

```
x = x - 1;
```

```
x--;
```

Operatory ++ i --

```
x = x++;
```

```
x = ++x;
```

```
x = x--;
```

```
x = --x;
```

- wartość powyższych instrukcji jest **nieokreślona**
- nie należy stosować operatorów ++, -- do zmiennych pojawiających się w wyrażeniu więcej niż jeden raz