

Podstawy i języki programowania

Laboratorium 4 - wprowadzenie do pętli, instrukcje break i continue oraz etykietowanie

mgr inż. Krzysztof Szwarc

krzysztof@szwarc.net.pl

Sosnowiec, 25 października 2018

Po co nam pętle?

Pętle umożliwiają nam wykonanie danych instrukcji określoną liczbę razy. Możemy przykładowo zastąpić zapis:

```
System.out.print("tekst");  
System.out.print("tekst");  
System.out.print("tekst");  
System.out.print("tekst");  
System.out.print("tekst");
```

jedną pętlą:

```
for (int i=0; i<5; ++i)  
    System.out.print("tekst");
```

$$\sum_{i=1}^5 i$$

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

Opis

Pętla **while** ma następującą postać:

```
while (warunek)  
    instrukcja ;
```

Jedynie w momencie, gdy warunek ma wartość `true` następuje wykonanie instrukcji. Będą one wykonywane do momentu przyjęcia przez warunek wartości `false` (jeżeli przed przystąpieniem do wykonania pętli warunek ma wartość `false`, to instrukcje nigdy nie zostaną wykonane).

Przykład

```
int licznik = 0;
final int LICZBA_WYKONAN = 5;

while (licznik < LICZBA_WYKONAN)
{
    licznik++;
    System.out.println(licznik); // 1; 2;
    3; 4; 5
}
```

Przykład drugi

```
int licznik = 0;
final int LICZBA_WYKONAN = 5;

while (licznik++ < LICZBA_WYKONAN)
    System.out.println(licznik); // 1; 2;
    3; 4; 5
```

Przykład trzeci

```
int licznik = 5;
final int LICZBA_WYKONAN = 5;

while (licznik++ < LICZBA_WYKONAN)
    System.out.println(licznik);
```

Opis

Pętla **do while** ma następującą postać:

```
do  
    instrukcja;  
while (warunek);
```

W przeciwieństwie do pętli while wskazana konstrukcja wykonuje instrukcje przynajmniej raz, gdyż prawdziwość warunku sprawdzana jest po ich wystąpieniu.

Przykład

```
int licznik = 5;
final int LICZBA_WYKONAN = 5;

do
{
    System.out.println(licznik); // 5
    licznik++;
} while (licznik < LICZBA_WYKONAN);
```

Przykład drugi

```
int licznik = 0;
final int LICZBA_WYKONAN = 5;

do
{
    licznik++;
    System.out.println(licznik); // 1; 2;
    3; 4; 5
} while (licznik < LICZBA_WYKONAN);
```

Przykład trzeci

```
int licznik = 0;
final int LICZBA_WYKONAN = 5;

do
    System.out.println(licznik); // 0; 1;
    2; 3; 4
while (++licznik < LICZBA_WYKONAN);
```

Opis

Pętla **for** ma następującą postać:

```
for (wyrażenie początkowe; wyrażenie  
    warunkowe; wyrażenie modyfikujące)  
    instrukcja;
```

Wyrażenie początkowe wykorzystywane jest do inicjalizowania zmiennej używanej jako licznik wykonań pętli, która jest wykonywana do momentu określonego przez wyrażenie warunkowe. Wyrażenie modyfikujące stosowane jest w celu modyfikacji wartości licznika pętli.

Przykład

```
for (int i=0; i<5; ++i)
    System.out.println(i); // 0; 1; 2; 3;
    4
```

Zmienna licznik występuje jedynie w bloku pętli - można zadeklarować zmienną o tej nazwie poza pętlą.

Przykład drugi

```
int licznik = 0;
for (; licznik<5; ++licznik)
    System.out.println(licznik); // 0; 1;
    2; 3; 4
```

Zmienna licznik występuje także poza blokiem pętli.

Przykład trzeci

```
int licznik = 0;
for (; licznik<5; )
{
    licznik++;
    System.out.println(licznik); // 1; 2;
    3; 4; 5
}
```

Zmienna licznik występuje także poza blokiem pętli.

Przykład czwarty

```
int licznik = 0;
for (;;)
{
    licznik++;
    System.out.println(licznik); // 0; 1;
    2; 3; 4; ...
}
```

Pętla będzie się wykonywać w nieskończoność.

Przykład piąty

```
for (;;)
    System.out.println(1); // 1; 1; 1; 1;
    ...
```

Pętla będzie się wykonywać w nieskończoność.

- 1 Jeżeli mamy wykonać daną czynność określoną z góry liczbę razy, zastosujemy pętlę **for**.
- 2 W przeciwnym razie użyjemy pętli ze słowem kluczowym **while**.
- 3 Jeżeli instrukcje mają się wykonać przynajmniej raz zastosujemy **do** oraz **while**. W przeciwnym razie użyjemy samego **while**'a.

Każdy program powinien zawierać informację o przeznaczeniu oraz sprawdzać czy użytkownik wpisuje dobre dane (do momentu wpisania przez użytkownika poprawnej wartości ma zostać pobierana liczba - wykorzystaj metody hasNext... klasy Scanner).

- 1 Napisz program pobierający od użytkownika n liczb całkowitych (gdzie n jest podane przez użytkownika w pierwszym kroku), obliczający ich średnią arytmetyczną i wyświetlający wynik.
- 2 Napisz program wypisujący liczby podzielne przez 3 z wykorzystaniem pętli for, while oraz do while (zakres od 1 do 100).
- 3 Napisz program obliczający silnię wprowadzonej liczby.

Instrukcja break

Opis

Instrukcja **break** stosowana jest do przerywania wykonania pętli i opuszczenia jej bloku. Przykład:

```
int licznik = 0;
for (;;)
{
    licznik++;
    System.out.println(licznik); // 0; 1;
    2; 3; 4;
    if (licznik == 4)
        break;
}
```

Przykład

```
for (;;)
{
    for (int i=0; i<5; ++i)
    {
        System.out.println(i); // 0; 1; 2;
                                0; 1; 2...
        if (i == 2)
            break;
    }
}
```

Pętla będzie się wykonywać w nieskończoność - instrukcja `break` spowodowała opuszczenie wyłącznie zagnieżdżonej pętli.

Opis

Instrukcja `continue` stosowana jest do przejścia do kolejnej iteracji pętli. Przykład:

```
for (int i = 0; i<5 ; ++i)
{
    if (i == 1)
        continue;
    System.out.println(i); // 0; 2; 3; 4;
}
```

Opis

Etykieta umożliwia przeniesienie sterowania do innego miejsca. Ma ona następującą postać:

```
nazwa_etykiety: instrukcja
```

Przykład

```
petla:
for (;;)
{
    for (int j=0; j<5; ++j)
    {
        System.out.println(j); // 0; 1; 2;
        if (j == 2)
            break petla;
    }
}
```

Zostaną opuszczone obie pętle.

Przykład drugi

```
petla:
for (int i=0; i<2; ++i)
{
    for (int j=0; j<2; ++j)
    {
        if (j == 1)
            continue petla;
        System.out.println(i+" "+j);
        /* 0 0
           1 0 */
    }
}
```

Każdy program powinien zawierać informację o przeznaczeniu oraz sprawdzać czy użytkownik wpisuje dobre dane (do momentu wpisania przez użytkownika poprawnej wartości ma zostać pobierana liczba - wykorzystaj metody hasNext... klasy Scanner).

- 1 Napisz program wypisujący liczby z zakreśu od 1 do 100, z wyłączeniem liczb, które są podzielne przez 4 i jednocześnie są niepodzielne przez 16. Zastosuj instrukcję **continue**.
- 2 Napisz program pobierający od użytkownika n liczb całkowitych, gdzie n jest pobierane od użytkownika w pierwszym kroku. Wypisz największą i najmniejszą z wprowadzonych wartości.

- 3 Napisz program obliczający p^n dla p i n wprowadzonego przez użytkownika. Wynik powinien zostać wypisany.
- 4 Napisz program rysujący choinkę na podstawie pobranej od użytkownika wysokości.
- 5 Napisz program, który dla pobranej liczby całkowitej wyznaczy jej wszystkie dzielniki i wypisze je na ekran.
- 6 Napisz program, który wyznacza wartość, wprowadzonej przez użytkownika, liczby ułamkowej w naturalnym kodzie binarnym (do 7 reduktu rozwinięcia).

Dziękuję za uwagę