Pętla `while`

```
while (warunek)
{
      blok_instrukcji;
}
```

Zasada działania: ***blok_instrukcji*** wykonuje się dopóki ***warunek*** jest prawdziwy. Warunek pętli znajduje się na początku, może się zatem zdarzyć, że pętla nie wykona się ani razu.

```
do
{
      blok_instrukcji;
} while (warunek)
```

Zasada działania: ***blok_instrukcji*** wykonuje się dopóki ***warunek*** jest prawdziwy. Warunek pętli znajduje się na końcu, ergo: pętla wykona się przynajmniej raz.

Przykład 4.1 Program do obliczania średniej ocen studentów. Program kończy działanie, gdy użytkownik wprowadzi liczbę `-1`.

```
static void p41()
{
      int suma = 0;
      int ocena = 0;
      int licznik = 0;

      while (ocena != -1)
      {
            Console.Write("Podaj ocenę: ");
            ocena = Convert.ToInt32(Console.ReadLine());
            suma += ocena;
            licznik += 1;
      }

      if (licznik > 0)
      {
            double srednia = suma / (double)licznik;
            Console.WriteLine("Śrenia z ocen wynosi: {0:N}", srednia);
      }
      else
      {
            Console.WriteLine("Brak danych do obliczenia średniej");
      }
}
```

Zadanie 4.1 Zmienić pętlę `while(warunek)` z przykładu 4.1 na pętlę `do-while`.

Zadanie 4.2 Klient banku zdeponował na lokacie pewną kwotę. Lokata jest oprocentowana 6% rocznie. Kapitalizacja odsetek odbywa się na koniec każdego roku. Klient zdecydował, że zrezygnuje z lokaty w chwili, gdy zarobi 100% początkowej kwoty. Obliczyć, ile lat klient musi utrzymywać lokatę.

Zadanie 4.3 Napisać program do symulacji gry „Za dużo! Za mało! ”. Gra polega na tym, że „komputer” losuje liczbę całkowitą z góry określonego przedziału. Użytkownik próbuje odgadnąć tę liczbę. Jeśli liczba podana przez użytkownika jest większa od wylosowanej, program wypisuje

komunikat „za dużo”, jeśli mniejsza – „za mało”. Gra kończy się w chwili, gdy użytkownik odgadnie liczbę. Program wypisuje komunikat „Odgadłeś za XX razem”.
Wskazówka: Do generowania liczb psudolosowych należy wykorzystać klasę `Random`.

```
Random rnd = New Random();
```

Klasa ta posiada metodę `Next`, która generuje kolejne liczby losowe z zadanego przedziału.

```
int liczba = rnd.Next(2, 10);
```

Zadanie 4.4 Napisać program do sprawdzania, czy liczbę całkowitą `x` można przedstawić w postaci 2^n, gdzie n jest liczbą całkowitą. Jeśli okaże się, że można, to program powinien wypisać ile wynosi n.

Zadanie 4.5. Napisać program do znajdowania największego wspólnego dzielnika dwóch liczb całkowitych a i b.
Wskazówka: Dla liczb a i b odejmij mniejszą od większej, a różnicę podstaw pod większą z nich. Powtarzaj powyższą operację aż obydwie liczby staną się równe. (Algorytm Euklidesa) .

Zadanie 4.6 Napisać program, który oblicza liczbę `pi` poprzez rozwinięcie w szereg Leibniza

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

Z dokładnością do zadanego d, (tzn. sumowanie elementów ciągu powtarzane jest do momentu, kiedy różnica pomiędzy kolejnymi wyrazami jest większa od d)

Zadanie 4.7 Napisać program do sprawdzania, czy podana liczba całkowita jest doskonała.
Wskazówka: Liczba jest doskonała, jeśli jest równa sumie swoich dzielników bez niej samej. Np. `6 = 1 + 2 + 3`.