Lista 1

1. Napisz ***możliwie prosty*** program, który ma wyświetlać na ekranie wartości *n!* (n silnia) dla n = 0...40. Możesz posługiwać się wyłącznie typem całkowitym `int` i nie musisz przejmować się błędnymi wynikami dla dużych *n*. Skomentuj źródło błędnych wyników.

2. Oblicz wartości poniższych wyrażeń dla n = 1 000 000 i porównaj je z wartością liczby π:

$$\text{(a)}\ \ 4\sum_{j=1}^{n}\frac{(-1)^{j+1}}{2j-1} \qquad \text{(b)}\ \ 2\prod_{j=1}^{n}\frac{4j^2}{4j^2-1} \qquad \text{(c)}\ \ \sqrt{8\sum_{j=1}^{n}\frac{1}{(2j-1)^2}}$$

3. Archimedes wyznaczył przybliżoną wartość liczby π na podstawie długości obwodów wielokątów foremnych wpisanych i opisanych na kole promieniu 1. Rozpoczął od sześciokąta i kolejno podwajał liczbę boków wielokąta. Pomysł ten prowadzi do wzoru rekurencyjnego., który można zapisać w dwóch matematycznie równoważnych postaciach:

(a)

$$t_0 = \frac{1}{\sqrt{3}}, \qquad t_{i+1} = \frac{\sqrt{t_i^2+1}-1}{t_i}, \qquad \pi \approx 6\times 2^i \times t_i, \qquad i = 0,1,\ldots,$$

(b)

$$t_0 = \frac{1}{\sqrt{3}}, \qquad t_{i+1} = \frac{t_i}{\sqrt{t_i^2+1}+1}, \qquad \pi \approx 6\times 2^i \times t_i, \qquad i = 0,1,\ldots,$$

Sprawdź, jakie każda z tych metod daje przybliżenie liczby π dla i = 0, 1,…, 30. Czy błąd metody zawsze maleje wraz z *i*?

Zadania do rozwiązania zespołowego
(może być przy tablicy, ewentualnie jedna osoba sprawdza w komputerze)

1. Pomóż Bajtkowi znaleźć i usunąć błędy w programie, który ma obliczać sumę odwrotności kwadratów miliona kolejnych liczb całkowitych (tj. od 1 do 10^6).

```
#include <iostream>
int main()
{
  const int N = 1 000 000;
  auto suma = 0;
  for (int k = 1, k <= N, ++k)
    suma += 1/k*k;
  std::cout << suma << ”\n”;
}
```

2. Oblicz (bez komputera) wartości następujących wyrażeń:

   a) `0xa – 012`
   b) `13 % 3`
   c) `3,14  - 3`
   d) `1234 ^ 1234`
   e) `1 << 3`
   f) `0xF & 0xA`
   g) `3 > 2 > 1`
   h) `12345 + ~12345`
   i) `1 + 1e-40 – 1`
   j) `1 + 1e-10f – 1`
   k) `3 == 3 == 3`
   l) `1/4`
   m) `16 >> 1`
   n) `0xff ^ 0xf0`
   o) `~(-1)`
   p) `0xff | 0xaa`
   q) `1234567 & 1`
   r) `1234567 | 1`
   s) `1234567 ^ 1`
   t) `1 < 2 ? 1 : 2`
   u) `1, 2, 3, 4`
   v) `3 < 2 & 1 < 2`
   w) `1 < 2 && 2 < 1`
   x) `1 < 2 || 2 > 1`
   y) `0b1111`
   z) `-1 > 1u`
   aa) `0 – 1u > 0`

Wskazówka: Odpowiedź możesz sprawdzić np. taką instrukcją:

```
std::cout << (3,14 – 3) << "\n";
```

3. Niech

```
int x = 1;
int y = x++;
int z = --y + x;
int v = z += 2;
```

   Jaką wartość mają teraz zmienne `x`, `y`, `v` i `z`?