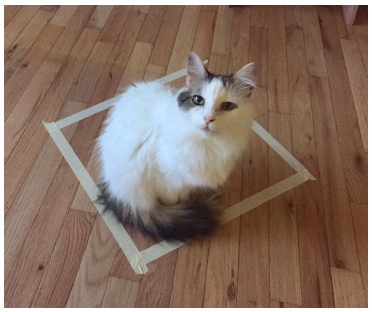

Kocur do kwadratu

#kwadrat
#potęga #pow
#liczba_zmiennoprzecinkowa #double
#operatory_porównania

Napiszemy teraz program, który oblicza kwadraty kolejnych liczb naturalnych, powiedzmy od 1 do 20. Nie jest to nic trudnego, ale jednak musimy przy tym powiedzieć o kilku ważnych sprawach, a w szczególności o obliczaniu potęg o wykładniku całkowitym. Kusiłoby nas, aby kwadrat liczby naturalnej n zapisać w postaci `n^2`, tak jak czynią to użytkownicy Excela. Niestety nic z tego: w języku C++ operator daszka `^` co prawda istnieje, ale nie ma nic wspólnego z potęgowaniem.*

Zatem kwadrat liczby n należy po prostu zapisać jako `n*n`. OK, nie jest to problem z potęgą o tak niewielkim wykładniku, ale jak wobec tego zapisać np. n^{15}? Z pomocą przychodzi tutaj biblioteczna funkcja `pow(n,k)` (od ang. **power**, wymawiaj: *pałer*), która oblicza wartość n^k (k może być niecałkowite, ale n powinno być dodatnie, aby nie było kłopotów).

Jednak funkcja `pow()` kryje w sobie pewien haczyk: jej rezultat jest typu `double`, czyli jest *liczbą zmiennoprzecinkową*. Operacje arytmetyczne na takich liczbach są z natury przybliżone. Owszem, są prowadzone z dużą ilością cyfr znaczących, ale jednak nie są idealnie dokładne, jak operacje na liczbach całkowitych (typ danych `int`). Dla przykładu, jeśli użyjemy tej funkcji do obliczenia pierwiastka kwadratowego z liczby 2, a następnie podniesiemy otrzymany pierwiastek do kwadratu, otrzymamy liczbę 2.0000000000000004441, a więc nie dokładnie 2. Oczywiście $\sqrt{2}$ jest liczbą niewymierną i można spodziewać się odstępstw od dokładnych wyników obliczeń, tym niemniej warto zachować ostrożność i nie używać typu `double`, jeśli wystarczy zwykły typ `int`.†

*Daszek oznacza bitową operację **xor**, czyli *alternatywę rozłączną*. Naukę operacji bitowych zostawimy sobie na później.

†Istnieje sposób dokładnego i efektywnego obliczania potęg o wykładniku naturalnym i myślę, że do niego jeszcze kiedyś wrócimy.

Skoro zamierzamy obliczać kwadraty liczb od 1 do 20, to nasz program może wyglądać na przykład tak:

```
#include <iostream>
using namespace std;

int main()
{
  int n = 1;
  while(n <= 20)
  {
    cout << n << " " << n * n << endl;
    n++;
  }
}
```

Rezultatem działania programu jest następująca podwójna kolumna liczb:

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
11 121
12 144
13 169
14 196
15 225
16 256
17 289
18 324
19 361
20 400
```

Jeszcze tylko jedną rzecz omówimy w miarę systematycznie – operatory porównania używane w wyrażeniach logicznych. Jest ich raptem kilka i łatwo je zapamiętać: `==` to „równe”, zaś `!=` to „różne od”. Nierówności ostre zapisujemy tak: `<` to „mniejsze niż”, a `>` to „większe niż”. Nierówności nieostre oznaczamy `<=`, czyli „mniejsze lub równe”, oraz `>=`, czyli „wieksze lub równe”. I tyle.