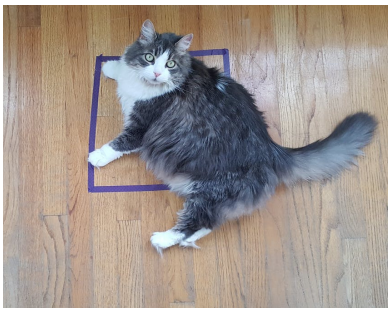
Znowu kwadrat



#funkcja
#instrukcja_skoku
#return

W tym przykładzie ponownie wrócimy do problemu podnoszenia do kwadratu liczby całkowitej, ale w nieco szerszym kontekście. Zauważmy, że samo obliczanie kwadratu jest dobrze określoną, niejako oddzielną operacją, która może wystąpić także w innych zadaniach. W takim przypadku zalecane jest zdefiniowanie własnej *funkcji*, której zadaniem będzie – w tym przypadku – obliczenie kwadratu danej liczby.

Tworzenie własnych funkcji jest jednym z najważniejszych elementów warsztatu programisty. Taka funkcja to coś w rodzaju czarnej skrzynki, do której przekazywane są dane – *argumenty funkcji* (nazywane też *parametrami funkcji*). Funkcja robi z nimi jakieś magiczne szacher-macher, po czym zwraca rezultat swojego działania przy użyciu instrukcji `return` (wymawiaj: *ritern*, z akcentem na *e*). W naszym przykładzie funkcja (nazwiemy ją z angielska `square()`, wymawiaj: *skłer*) pobierać będzie liczbę całkowitą – oznaczmy ją przez k – i zwracać będzie wartość jej kwadratu, czyli k^2 (również liczbę całkowitą). Oto definicja tej funkcji:

```
int square(int k)
{
  return k * k;
}
```

A tak powinien wyglądać program, który wypisuje kwadraty kolejnych liczb naturalnych od 1 do 20 – jak w podrozdziale *Kocur do kwadratu*:

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

int square(int k)
{
  return k * k;
}

int main()
{
  int n = 1;
  while(n <= 20)
  {
```

```
        cout << n << " " << square(n) << endl;
        n++;
    }
}
```

W funkcji `main()` mamy wywołanie funkcji `square()` z argumentem n. Na moment przenosimy się wtedy do wnętrza naszej funkcji (czyli wykonujemy skok), a po obliczeniu wartość n^2 jest wstawiana w miejsce wywołania funkcji (czyli wykonujemy skok z powrotem).

Zauważ, że definicję funkcji umieściliśmy przed jej użyciem w funkcji `main()`, tak aby kompilator „rozumiał", co znaczy słowo `square`.*

Zmienna k jest nazywana *argumentem formalnym* funkcji, bo jest wymieniona w definicji funkcji. Zmienna n jest z kolei nazywana *argumentem aktualnym* funkcji, bo funkcja jest wywoływana właśnie z takim argumentem.

*W rzeczywistości od tej reguły dopuszczalne są pewne odstępstwa, ale najczęściej kolejność zapisu jest właśnie taka.