Kot bankowy

#procent_składany
#dzielenie_zmiennoprzecinkowe
#round

Kontynuujemy naszą naukę operacji na liczbach zmiennoprzecinkowych (typ `double`). Tym razem zajmiemy się obserwacją stanu konta bankowego, na którym ktoś ulokował pewną kwotę, zaś bank co roku dopisuje do tego konta odsetki* – zatem w następnym roku odsetki będą już liczone od powiększonej kwoty. Taki sposób naliczania odsetek to tak zwany *procent składany*.

Użytkownik powinien wpisać trzy dane: początkową kwotę depozytu, stopę rocznego oprocentowania oraz ilość lat przechowywania pieniędzy w banku. Z wyjątkiem tej ostatniej, wprowadzone liczby powinny dopuszczać część ułamkową, a więc powinny być typu `double`:

```
double deposit, percent;
int years;
cin >> deposit >> percent >> years;
```

Teraz można obliczać stan konta na koniec każdego roku (czyli okresu rozrachunkowego). Tych obliczeń wykonamy tyle, ile wynosi wartość zmiennej `years`. Po każdym roku stan konta zwiększa się o czynnik `1 + percent / 100`, zatem nasza pętla może wygladać tak:

```
int y{ };
while(y < years)
{
  deposit *= 1 + percent / 100;
  y++;
  cout << y << ". " << deposit << '\n';
}
```

Operator `*=` daje w wyniku pomnożenie lewej strony przez wyrażenie stojące po prawej stronie.

Uruchamiamy program i wpisujemy przykładowe dane:

```
10000 4.5 6
```

(Depozyt dziesięć tysięcy złotych, oprocentowanie 4,5%, okres lokaty: 6 lat.)

*Fachowo nazywa się to *kapitalizacją odsetek*.

Program powinien wypisać:

```
1. 10450
2. 10920.2
3. 11411.7
4. 11925.2
5. 12461.8
6. 13022.6
```

Coś jest nie tak, bo nie widzimy pojedynczych groszy, czyli setnych części złotego. Możemy zwiększyć dokładność, dopisując przed pętlą instrukcję

```
cout.precision(10);
```

Liczbę wyświetlanych cyfr dobraliśmy dość arbitralnie na 10. Co teraz dostaniemy?

```
1. 10450
2. 10920.25
3. 11411.66125
4. 11925.18601
5. 12461.81938
6. 13022.60125
```

Masz babo placek! Teraz znowu za dokładnie, żaden bank nie oblicza stanu konta z taką dokładnością. Musimy ograniczyć się do pełnych groszy, co będzie łatwe, jeśli stan konta pomnożymy na początku przez 100 (czyli zamienimy złote na grosze) i po każdorazowej operacji będziemy kwotę zaokrąglać (ale nie obcinać) do pełnych groszy. W tym celu przyda nam się funkcja `round()` (wymawiaj: *raund*). Musimy tylko pamiętać, żeby wypisywać stan konta podzielony przez 100 lub pomnożony przez 0.01, abyśmy zobaczyli kwotę w złotych.

Oto kompletny program:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
  double deposit, percent;
  int years;
  cin >> deposit >> percent >> years;
  deposit *= 100;
  cout.precision(10);
  int y{ };
  while(y < years)
  {
    deposit = round(deposit * (1 + percent / 100));
    y++;
    cout << y << ". " << deposit*0.01 << '\n';
```

```
  }
  return 0;
}
```

Wynik tego programu jest już chyba akceptowalny:

```
1. 10450
2. 10920.25
3. 11411.66
4. 11925.18
5. 12461.81
6. 13022.59
```