Pisanie programów, czyli dużo małych nowości

Dziś napiszemy kilka programików, wykorzystując kilka przydatnych rzeczy, których nie jestem w stanie odpowiednio skategoryzować, albo są zbyt małe, żeby zasłużyć na oddzielnego PDFa.

Typy danych i ich rzutowanie

Na zajęciach rozmawialiśmy jakiś czas temu, że typ danych *bool* to tak w zasadzie 0 albo 1. To teraz magia. Litery to też tak naprawdę liczby. Mamy kilka możliwości, aby to sprawdzić.

```
char character = 65;
cout<<character;
```

char to typ znakowy, przechowujący pojedynczy znak ASCII - *American Standard Code for Information Interchange.* Standard ten przypisuje wartości 0-127 do liter, znaków przestankowych, symboli i poleceń sterujących. Pełen spis znajdziecie np. na Wikipedii. Powyższy kod wydrukuje nam literkę ‘A’. Ze swojej strony polecam stosowanie kodowania heksadecymalnego, trochę logiczniej się ta tabela układa. W zapisie HEX przed liczbą podajemy ‘0x’ do rozróżnienia od zapisu dziesiętnego.

```
char character1 = 65, character2 = 0x41;
cout<<character1<<character2;
```

Można też w drugą stronę:

```
int character1 = 'A';
cout<<character1;
```

Rzutowanie typów

Pojęcie może nieoczywiste, ale sprowadza się po prostu do konwersji/zmiany typu zmiennej. Robiliśmy to pośrednio powyżej – było to rzutowanie niejawne, czyli wykorzystanie w wyrażeniu lub przypisaniu wartości innego typu niż docelowy
Jest jeszcze klika rodzajów:

1. C – style, czyli

```
int character1 = 0x41;
cout<<char(character1);
```

Ponoć niezalecane w C++, ale moja miłość do czystego C nie pozwala mi o tym nie wspomnieć. Inną sprawą jest, że w C++ działa bez zarzutu pomiędzy wszystkimi typami.

2. Operator static_cast<typ>

```
int number = 21;
cout<<static_cast<float>(number*1.1);
```

Tylko te dwie metody rzutowania zapewnią nam 100% zgodności.

Biblioteka cmath

Biblioteka math.h zawiera zestaw operacji i stałych matematycznych, pozwalając tym samym na bardziej złożone obliczenia. Trygonometria, potęgi, funkcje logarytmiczne i wykładnicze, kilka stałych. Po pełną listę zapraszam do dokumentacji, na zajęciach kilka przykładów:

```
#include <iostream>
#include <cmath>
#define _USE_MATH_DEFINES
using namespace std;
int main(){

cout<<sqrt(81)<<endl;
cout<<round(21.37)<<endl;
cout<<pow(9, 3)<<endl;
cout<<cos(60.0*M_PI/180.0)<<endl;

return(0);
}
```

Na szczególną uwagę zasługuje tu M_PI – jest to stała definiowana w bibliotece, jej zastosowanie wymaga skorzystanie z dyrektywy #define:

```
#define _USE_MATH_DEFINES
```

Warto również zwrócić uwagę, że argument do funkcji cos() nie jest podany w stopniach, a w radianach.

$$\alpha_{rad} = \frac{\pi\alpha}{180}$$

Zadanie 1

Napisz program obliczający pole wycinka koła

$$F = \frac{R^2}{2}\left(\frac{\pi\alpha}{180} - \sin\alpha\right)$$

Zadanie 2

Rozszerz pisany na wcześniejsze zajęcia kalkulator, aby obsługiwał sinus, cosinus, tangens, cotangens, pierwiastek. Polecam rzucić okiem na możliwości formatowania cout, np.:

```
cout.setf(ios::fixed);

cout.setf(ios::showpoint);

cout.width(8);

cout.precision(3);
```