Programowanie Zaawansowane FM i NI (C++)
Ćwiczenia 1

Zadanie 1 (rama kodu, `cout`, kompilator)

W środowisku *programiz* zakoduj program, który wyświetla `Hello World` . Zadbaj o wcięcie w środku nawiasów `{ }`. Wykonaj kod.

Następnie zaloguj się do środowiska *repl.it* i przekopiuj swój kod. W terminalu (*"shell"*) skompiluj ten kod (np. `g++ mojkod.C -o mojkod`), a otrzymany *plik wykonywalny* włącz. Teraz skompiluj bez opcji `-o` i sprawdź nazwę utworzonego pliku.

Zadanie 2 (proste zmienne i ich zakres)

W środowisku *programiz* zadeklaruj zmienne `a` i `b`, nadając obu typ `int` i przypisz im wartości, np. `7` i `5`. Następnie zadeklaruj zmienną `c` typu `int` i przypisz jej wynik dzielenia `a` przez `b`. Wyświetl wartość zmiennej `c` na ekran.

Poprawmy otrzymany wynik. Wpierw typ zmiennej `c` przepraw na `float`. Czy to rozwiązało problem? Czy domyślasz się, dlaczego? Co zaproponujesz?

Przepisz kod w środowisku *repl.it* . Skompiluj go i wykonaj.

Zadanie 3 (`cin`)

Zmień powyższy kod tak, aby wartości do zmiennych podawał użytkownik z klawiatury. Sprawdź działanie kodu. Co się stanie, gdy do `b` użytkownik wpisze `0` ?

Zadanie 4 (`if/else`)

Popraw kod: wykorzystaj blok instrukcji warunkowych `if/else`, aby wykonywać dzielenie tylko wtedy, gdy `b` $\neq$ `0` . Operator logiczny $\neq$ w C++ ma postać: `!=`

Dla wariantu, gdy `b = 0` , zakoduj wyświetlanie ostrzeżenia, np:

```
Proba dzielenia przez 0...
```

Zadanie 5 (pętla `while`)

Zmień powyższy kod tak, aby w pętli komputer powtarzał pytanie o składniki i wyświetlanie wyniku. Najłatwiej użyć pętli `while`. Warunkiem zakończenia programu niech będzie sytuacja, w której użytkownik z klawiatury do obu zmiennych poda zera.

Na wypadek, gdyby już przy początkowej deklaracji zmiennych komputer wstawiał do nich zera, przypisz tam dowolne inne wartości "startowe".

Zadanie 6 (`if / else if / else`. Funkcja `sqrt` w bibliotece `cmath`)

Napisz program rozwiązujący równanie kwadratowe ($ax^2 + bx + c = 0$), jeśli użytkownik poda z klawiatury `a`, `b` i `c`. Kod powinien wyliczyć Δ trójmianu kwadratowego, a następnie uzależnić postępowanie od wartości tej Δ . W każdym z wariantów wypisz wyniki.

Zadanie 7 (pętla `do .. while`, funkcja `fmod` w `cmath`)

Bieg ParkRun na Polu Mokotowskim ma długość 5,0 km w 3 okrążeniach. Zaczynając we wspólnym miejscu, jeden zawodnik biegnie z prędkością v1 = 12 km/h, a drugi – idzie z prędkością v2 = 4 km/h. W efekcie, w pewnym miejscu zawodnik biegnący drugie okrążenie przekroczy idącego.

Napisz kod, w którym w pętli będziesz aktualizować położenie obu zawodników co dt = 10 s i wypisywać status: aktualny czas oraz dla każdego zawodnika: nr bieżącego okrążenia i odległość od startu.

Zakończ pętlę, gdy zawodnik biegnący przekroczy idącego.

Zadanie 8 (pętla `for`, funkcja `max` w `cmath`, znak vs napis)

Napisz program pobierający od użytkownika trzy nieujemne liczby całkowite i wypisujący na ekranie „histogram" dla tych danych, to znaczy trzy „słupki" złożone ze znaków `'*'`, wyrównane od dołu, o wysokościach równych wartościom trzech wczytanych liczb.

Wykonanie programu mogłoby zatem wyglądać tak:

```
$ ./histo
Enter three non-negative numbers: 3 1 8
  *
  *
  *
  *
  *
* *
* *
***
```

Zadanie 9 (zmienne, blok `if/else`)

Do oporu zastępczego złożonego z 2 oporników `R1` i `R2` przyłożono napięcie `U`. Napisz program, który pyta użytkownika o wartości oporów, przyłożone napięcie oraz maksymalne natężenie prądu w sieci `Imax` (zmienne typu `double`). Na koniec niech pyta o to, jak oporniki zostały połączone: gdy użytkownik wpisze `s`, to szeregowo, a gdy `r`, to równolegle.

W zależności od odpowiedzi, podaj wartość oporu zastępczego. Następnie podaj wartość łącznego natężenia prądu przechodzącego przez urządzenie. Na koniec, w zależności od tego, czy natężenie mieści się w normie, czy ją przekracza – niech komputer napisze adekwatny komunikat.

Wskazówka: do posługiwania się pojedynczym znakiem służy zmienna typu `char`. Np:

```
char znak;
cin >> znak;
if ( znak = 's') {
    ...
}
```