Programowanie Zaawansowane FM i NI (C++)
Ćwiczenia 2

Zadanie 1 (algorytmika, pętle, warunek if, break)

Napisz program, który wczytuje w pętli ciąg liczb całkowitych do momentu, gdy użytkownik poda liczbę 0, która jest tylko sygnałem końca danych i nie jest dalej brana pod uwagę.

Następnie program wypisuje wartości najmniejszego i największego elementu wczytanego ciągu oraz liczbę wystąpień tych wartości w całym ciągu.

Na przykład dla ciągu (7, 3, −24, 2, 7, −24, 7, 2, 0) program powinien wypisać:

```
Min = -24, 2 razy
Max = 7, 3 razy
```

Nie używaj tablic, napisów ani żadnych innych kolekcji.

Zadanie 2 (algorytmika, pętle, warunek if)

Napisz program wczytujący liczby całkowite (aż do pojawienia się zera, które kończy program). Kod ma wychwycić najdłuższą sekwencję kolejnych liczb o takich samych wartościach (wartość oraz ile razy wpisana), a na koniec wypisać odpowiedź.
Na przykład dla:

```
22 22 22 22 3 3 3 2 -6 -6 -6 0
```

wynik powinien być:

```
Longest sequence: 22 (4 times)
```

Dla:

```
-2 -2 -2 31 31 31 31 31 17 6 6 6 0
```

powinno być:

```
Longest sequence: 31 (5 times)
```

a dla:

```
-3 2 -3 -3 2 -3 61 61 61 0
```

powinno być:

```
Longest sequence: 61 (3 times)
```

Nie używaj tablic, napisów ani żadnych innych kolekcji.

Zadanie 3 (funkcje, cmath)

Kąt pomiędzy dowolnymi dwoma wektorami można wyznaczyć, korzystając z definicji i własności iloczynu skalarnego:

$$\cos \sphericalangle(\vec{v}_1, \vec{v}_2) = \frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1| \; |\vec{v}_2|}$$

$$\vec{v}_1 \cdot \vec{v}_2 = v_{1x} v_{2x} + v_{1y} v_{2y} \qquad \text{(przypadek 2D)}$$

Załóż przestrzeń 2-wymiarową i podziel zadanie wyliczenia kąta na trzy funkcje:
- `Length` – liczącą długość wektora,
- `DotProduct` – liczącą iloczyn skalarny oraz
- `Angle` – scalającą całość i zwracającą wynik (w stopniach).

Zakoduj te funkcje, przyjmując, że kąt podany ma zostać w stopniach. W funkcji `main` sprawdź działanie dla wektorów:

(a) $\overline{v}_1$ = `[ 2, 0]` i $\overline{v}_2$ = `[ 0, 2]`

(b) $\overline{v}_1$ = `[-2, 2]` i $\overline{v}_2$ = `[-2, 0]`

Zadanie 4 (funkcje, cmath)

W funkcji `double Fun (double x)` zakoduj funkcję `sin(x)`. Następnie w funkcji `double Integral (double x0, double x1, int Nsteps)` zakoduj numeryczne obliczanie całki oznaczonej metodą trapezów:

$$\int_a^b f(x)\, dx \approx \sum_i \frac{f(x_{L,i}) + f(x_{P,i})}{2} \Delta x \quad ,$$

gdzie przedział [a, b] dzieli się na Nsteps podprzedziałów o szerokości Δx = (b-a)/Nsteps , a w każdym i-tym podprzedziale mamy $x_i \in [\, x_{L,i} , x_{P,i} \,]$.

W funkcji `main` wywołaj Integral dla a = 0, b = 2, Nsteps = 10, a następnie 1000.

Zadanie 5 (funkcje)

Primorial nieujemnej liczby całkowitej N, oznaczany N#, jest zdefiniowany jako iloczyn wszystkich liczb pierwszych mniejszych lub równych N. Zakłada się przy tym, że z definicji 0# = 1# = 1. Napisz funkcję obliczającą N# . Np. następujący program:

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

bool isPrime (int n) {
  // ...
}

int primorial (int n) {
  // ...
}

int main() {
  cout << " N       N# \n";
  for (int n = 0; n <= 11; ++n)
     cout << n << '\t' << primorial(n) << '\n';
}
```

powinien wydrukować:

```
N   N#
0   1
1   1
2   2
3   6
4   6
5   30
6   30
7   210
8   210
9   210
10  210
11  2310
```

Zadanie 6 (funkcja rekurencyjna)

Zakoduj metodą rekurencji:

(a) funkcję `int SumDigits (int n)` , która po wszystkich wywołaniach zwróci sumę cyfr podanej liczby naturalnej (np. SumDigits (1234) = 10).

(b) funkcję `int NWDrec (int a, int b)` , która po wszystkich wywołaniach zwróci największy wspólny dzielnik metodą algorytmu Euklidesa.

(c) funkcję `int ExpRec (double x, int Nstep, int Nmax)`, która po wszystkich wywołaniach zwróci wartość exp (*x*) , policzoną do `Nmax` kroków. Zauważ, że:

$$\exp(x) = \sum_{n=0}^{N} \frac{x^n}{n!} = 1 + x \cdot \left(1 + \frac{x}{2} \cdot \left(1 + \frac{x}{3} \cdot (.....)\right)\right)$$

(d) funkcję `int FiboRec (int N)` , która po wszystkich wywołaniach zwróci N-ty wyraz ciągu Fibonacciego. Ciąg ten zdefiniowany jest jako:

$$F(N) = \begin{cases} 0 & \Leftrightarrow \quad N = 0 \\ 1 & \Leftrightarrow \quad N = 1 \\ F(N-1) + F(N-2) & \Leftrightarrow \quad N > 1 \end{cases}$$

<u>*Ciekawostka*</u>: w C++ istnieją zmienne statyczne, które raz zadeklarowane (np. w funkcji), zachowują wartość po jej końcu, jak i przy ponownym wykonaniu. Przykładowa deklaracja:

```
static int myValue = 5;
```

Spróbuj zakodować funkcję liczącą silnię lub ciąg Fibonacciego przez zmienne statyczne (zamiast rekurencji).

(e) funkcję `void printOddEven (int n)`, która po wszystkich wywołaniach wydrukuje w jednej linii:

- dla `n` parzystego: wszystkie liczby parzyste {2 .. n}
- dla `n` nieparzystego: wszystkie liczby nieparzyste {1 .. n}

(f) funkcję `void hailstone (int n)`, która po wywołaniach wydrukuje wszystkie liczby ciągu a_n, od `n` aż do 1.

$$a_{n+1} = \begin{cases} a_n / 2 & \text{jeśli } a_n \text{ jest parzyste} \\ 3a_n + 1 & \text{jeśli } a_n \text{ jest nieparzyste} \end{cases}$$