Elementy języka C++: funkcje, rekurencja, przeładowanie funkcji.

1. **Algorytm Euklidesa**
   Napisz funkcję znajdującą największy wspólny dzielnik dwóch liczb naturalnych metodą Euklidesa. Mając dwie liczby, większą z nich zastępujemy resztą z dzielenia przez mniejszą. Procedurę powtarzamy, aż jedna z liczb stanie się równa zero. Wtedy druga jest poszukiwanym wspólnym dzielnikiem wyjściowych liczb.

2. **Rekurencyjna silnia**
   Napisz program, który wczytuje ze standardowego wejścia liczbę naturalną i wypisuje na standardowe wyjście jej silnię. Użyj funkcji rekurencyjnej.

3. **Ciąg Fibonacciego**
   Ciąg Fibonacciego to ciąg liczb, w którym pierwszy wyraz jest równy 0, drugi jest równy 1 a każdy następny jest sumą dwóch poprzednich:

   $$F_0 = 1 \qquad F_1 = 1 \qquad F_n = F_{n-1} + F_{n-2}$$

   Napisz program wyznaczający n-ty wyraz ciągu najpierw przy użyciu funkcji iteracyjnej, a następnie rekurencji. Które podejście jest łatwiejsze?

4. **Czy posortowana?**
   Napisz funkcję typu *bool*, której argumentami są tablica oraz liczba jej elementów. Funkcja powinna ustalać, czy dane w tablicy są posortowane. Rozwiązanie powinno wymagać użycia tylko pojedynczej pętli (bez zagnieżdżania).

5. **Szukanie sekwencyjne**
   Napisz funkcję, która otrzymuje tablicę liczb całkowitych oraz wartość poszukiwaną i zwraca liczbę wystąpień tej wartości w podanej tablicy. Rozwiąż problem najpierw przy użyciu iteracji, a następnie rekurencji.

6. **Sortowanie przez scalanie ***
   Sortowanie przez scalanie to rekurencyjny algorytm sortowania danych, stosujący metodę *dziel i zwyciężaj*, o złożoności czasowej lepszej niż dla poznanych poprzednio algorytmów sortowania. Opis działania algorytmu jest przedstawiony np. na stronie:

   `http://main.edu.pl/pl/user.phtml?op=lesson&n=24&page=algorytmika`

   Korzystając z zamieszczonego tam opisu, postaraj się zaimplementować program sortujący, korzystając z funkcji dokonującej podziału oraz złączającej posortowane ciągi liczb całkowitych. W razie problemów spróbuj zrozumieć zamieszczone rozwiązanie.

7. **Wszystko jest funkcją**
   Użycie funkcji z jednej strony poprawia czytelność kodu programu, z drugiej pozwala na zastosowanie istniejącego kodu wprost do nowego problemu (np. występującego na kolokwium). Przepisz kilka z napisanych przez Ciebie programów tak, aby wykorzystywały funkcje.

8. **Pierwotne trójki Pitagorasa**
   Trójką pitagorejską nazywamy każde trzy liczby naturalne a, b, c takie, że $a^2 + b^2 = c^2$. Trójkę nazywamy pierwotną, jeżeli liczby a, b, c są względnie pierwsze. Można pokazać, że każda trójka pierwotna jest postaci:
   $$a = m^2 - n^2 \qquad b = 2mn \qquad c = m^2 + n^2$$
   gdzie m i n są względnie pierwszymi liczbami naturalnymi, z których jedna jest parzysta. Napisz program, który wczytuje ze standardowego wejścia liczbę naturalną C, a następnie wypisuje na standardowe wyjście wszystkie pierwotne trójki pitagorejskie, dla których $c < C$.

9. **Ocenianie kształtujące**
   W szkolnych klasach 1–3 popularną metodą oceniania sprawdzianów jest ich samodzielne sprawdzanie przez dzieci przez porównanie z odpowiedzią wzorcową na tablicy (małe dzieci nie oszukują). Napisz funkcję, która oblicza liczbę błędów popełnioną przez dziecko podczas pisania dyktanda. Funkcja powinna przyjmować jako argumenty dwa łańcuchy tekstowe – jeden napisany przez nauczyciela, drugi przez ucznia. Zweryfikuj działanie funkcji na przykładowym tekście wprowadzanym z klawiatury.

10. **Rekurencyjna potęga**
    Napisz program w sposób rekurencyjny obliczający funkcję potęgową x^y.

11. **Parzystość łańcucha binarnego ***
    Rozważ tablicę reprezentującą łańcuch binarny, w której elementy mają wartości 0 lub 1. Napisz funkcję typu *bool* ustalającą, czy łańcuch binarny jest nieparzysty (czyli posiada nieparzystą liczbę bitów równych 1). *Wskazówka*: pamiętaj, że funkcja rekurencyjna będzie zwracać true (nieparzystość) lub false (parzystość), a nie liczbę bitów równych 1. Rozwiąż problem najpierw przy użyciu iteracji, a następnie rekurencji.

12. **Rekurencyjne szukanie (binarne) ***
    Napisz funkcję rekurencyjną, która pobiera posortowaną tablicę, element docelowy oraz odszukuje ten element w tablicy (jeśli nie znajdzie elementu, powinna zwrócić -1). Jak szybkie może być takie szukanie? Czy można osiągnąć lepszy wynik bez potrzeby sprawdzania każdego elementu?

Pytania, a także rozwiązania zadań, można wysyłać na adres: MDABROWSKI@FUW.EDU.PL.