# Struktury i pliki

Przeanalizuj dokładnie problem opisany w każdym zadaniu. Następnie skonstruuj rozwiązanie opisanego problemu (algorytm) i zaimplementuj je w języku C++.

## Zadanie 1

Zdefiniuj strukturę `triangle` przechowującą długości boków trójkąta w postaci liczb całkowitych. Następnie zaimplementuj następujące funkcje:

1. Dla podanego parametru typu `triangle` funkcja `perimeter` oblicza obwód przekazanego trójkąta.
2. Dla podanego parametru typu `triangle` funkcja `isIsoscelesTriangle` sprawdza, czy jest to trójkąt równoramienny.
3. Dla podanego parametru typu `triangle` funkcja `isValidTriangle` sprawdza, czy jest to prawidłowy trójkąt, tzn. czy długości boków spełniają warunek trójkąta.
4. Dla podanego parametru typu `triangle` funkcja `area` oblicza pole przekazanego trójkąta (*wzór Herona*).
5. Funkcja `generateRandomTriangle` generuje i zwraca losowy, poprawny trójkąt.
6. Dla podanego parametru typu `triangle` oraz nazwy pliku funkcja `appendToFile` **dopisuje** podany trójkąt do pliku tekstowego.
7. Dla podanej tablicy elementów typu `triangle` oraz nazwy pliku funkcja `readTrianglesFromFile` wczytuje **wszystkie** trójkąty z podanego pliku tekstowego, zapisuje je do tablicy i zwraca liczbę wczytanych elementów.
8. Dla podanej tablicy elementów typu `triangle` funkcja `findTriangleWithSmallestPerimeter` znajduje trójkąt o najmniejszym obwodzie.
9. Dla podanej tablicy elementów typu `triangle` funkcja `findTriangleWithLargestPerimeter` znajduje trójkąt o największym obwodzie.
10. Dla podanej tablicy elementów typu `triangle` funkcja `findSmallestAndLargestPerimeterTriangles` znajduje i zwraca (**jednocześnie**) trójkąt o najmniejszym obwodzie oraz trójkąt o największym obwodzie.
11. Dla podanej tablicy elementów typu `triangle` funkcja `sortTrianglesByArea` sortuje tablicę rosnąco po polu trójkąta.

## Zadanie 2

Zdefiniuj poniższe struktury, dobierając odpowiednie typy danych do pól:

- `date` zawierającą pola: `year`, `month`, `day`.
- `location` zwierającą pola: `streetName`, `streetNumber`, `city`, `postalCode`.
- `student` zawierającą pola: `name`, `surname`, `idCardNumber`, `birthDate`, `address`.

Następnie zaimplementuj następujące funkcje:

1. Funkcja `readDateFromUser` wczytuje od użytkownika datę i zwraca ją jako wynik.
2. Funkcja `readLocationFromUser` wczytuje od użytkownika lokację i zwraca ją jako wynik.

3. Funkcja `readStudentFromUser` wczytuje od użytkownika dane studenta i zwraca jako wynik.
4. Dla podanego parametru typu `date` funkcja `printDate` wypisuje datę na ekranie.
5. Dla podanego parametru typu `location` funkcja `printLocation` wypisuje adres na ekranie.
6. Dla podanego parametru typu `student` funkcja `printStudentDetails` wypisuje dane studenta na ekranie.
7. Dla podanego parametru typu `date` i nazwy pliku funkcja `appendDateToFile` **dopisuje** datę do pliku tekstowego.
8. Dla podanego parametru typu `location` i nazwy pliku funkcja `appendLocationToFile` **dopisuje** lokację do pliku tekstowego.
9. Dla podanego parametru typu `student` i nazwy pliku funkcja `appendStudentToFile` **dopisuje** studenta do pliku tekstowego.
10. Funkcja `compareDates` przyjmuje dwa parametry typu `date` i określa ich relację: mniejsze ($-1$), większe ($1$), równe ($0$).
11. Dla podanej tablicy elementów typu `student` oraz nazwy pliku funkcja `loadStudentsFromFile` wczytuje dane **wszystkich** studentów, zapisuje ich do tablicy i zwraca liczbę wczytanych danych.
12. Dla podanej tablicy elementów typu `student` oraz parametru określającego pole struktury, funkcja `sortStudentsByField` sortuje tablicę rosnąco po wartości podanego pola.
13. Dla podanej tablicy elementów typu `student` oraz nazwy miasta funkcja `printStudentsFromCity` wypisuje wszystkich studentów z tego samego miasta.