IV. Enkapsulacja klas albo struktur (mechanizm zagnieżdżania), przyjaźń (*friend*), zastępczy typ zmiennej *auto*, funkcje stałe (*const*), konstruktor kopiujący, modyfikator *mutable*, obsługa obiektów stałych, użycie destruktorów.

Warunkiem wstępnym do realizacji tematów obecnego zajęcia laboratoryjnego jest kod z poprzedniego etapu (poprzedniego laboratorium). Ogólnym założeniem realizacji laboratoriów jest rozbudowa kodu albo jego modyfikacja nie powodująca redukcji jego funkcjonalności.

1. Zmodyfikuj albo rozszerz funkcjonalność kodu z etapu III w celu zademonstrowania mechanizmu klas zagnieżdżonych. *Szczegóły przykładu implementacji zagnieżdżania znajdują się w katalogu grupy wykładowej*), np.,

```
#include "Osoba.h"
class Karta {
   class Badanie {
      friend class Karta;
      bool chory{ false };
      bool stan_badan{ false };
      std::string choroba;
   public:
      Badanie() = default;
      Badanie(const Badanie&);
      explicit Badanie(std::string ch) : choroba{ ch } {}
   };
   size_t ilosc_badan{ 0 };
   Badanie** badania{ nullptr };
   Osoba* osoba{nullptr}; // właściciel karty

   void dodaj_badanie(std::string choroba);
public:
         Karta();
         Karta(const Karta&);
         ~Karta();
         void set_id_karty(size_t);
         void pokaz() const;
   void zlec_badanie(std::string choroba="angina");
};
```

W tym przykładzie klasa *Badanie* jest klasą zagnieżdżoną w polu prywatnym klasy *Karta*, przy czym ostatnia jest zaprzyjaźniona z klasą *Badanie*. Jest to typowy styl użycia zagnieżdżania.

2. Zmodyfikować kod z etapu III pod kątem użycia destruktorów. Zrealizować ten wymóg można drogą eliminacji tych funkcji globalnych, które zwalniają pamięć zasobów dynamicznych, będące atrybutami typów strukturalnych.

3. Zdefiniować we wszystkich klasach właściwych do tego celu konstruktorów kopiujących. Uzasadnić swój wybór.

4. Zmodyfikuj definicję wybranej klasy by potrafiła obsłużyć stały obiekt tejże klasy. Uzasadnić swoje rozwiązanie.

5. Po tych zmianach program powinien minimalnie realizować całą funkcjonalność z poprzedniego etapu (etap III), tzn. nie wolno redukować funkcjonalności, lecz tylko ją rozwijać (za to są naliczane dodatkowe punkty do oceny).

6. Podczas zajęcia zademonstrować działanie programu.

7. Pliki źródłowe wykonanego programu wgrać do tego zadania na Teams w określonym terminie.