

Instrukcja

W czasie pisania programu pamiętaj o:

1. dbaniu o czytelność kodu (odpowiednie formatowanie kodu, nazewnictwo zmiennych adekwatne do ich znaczenia, komentarze),
2. dbaniu o czytelność interfejsu z użytkownikiem (w sposób jawny pytaj użytkownika jakie dane ma podać oraz opisz wyniki, które zwracasz),
3. przed fragmentem implementującym poszczególne zadania umieść komentarz: `/*Zadanie X */` oraz wypisz na ekranie analogiczny komunikat (X jest numerem zadania): `std::cout << "Zadanie X"<< std::endl;`,
4. każde zadanie umieść w oddzielnej funkcji (w niej dopiero należy odwoływać się do zaimplementowanych funkcji i klas),
5. zaimplementuj menu wyboru zadania, a następnie wykorzystując pętle **do-while** oraz konstrukcję **switch** wykonaj odpowiedni fragment kodu,
6. w zadaniach wymagających udzielenia komentarza bądź odpowiedzi, należy umieścić go w kodzie programu (np. w postaci komentarza albo wydrukować na ekranie),
7. w zadaniach polegających na zaprojektowaniu klasy należy utworzyć jej instancję i wykorzystać zaimplementowaną funkcjonalność.

Wprowadzenie

Specyfikatory dostępu pozwalają na określenie poziomu dostępu do poszczególnych składowych klas. Specyfikator **private** oznacza, że elementy dostępne są tylko z wnętrza danej klasy (i klas/funkcji zaprzyjaźnionych). Specyfikator **public** powoduje, że składowe są publicznie dostępne. Jeśli deklaracja składowej (lub sekwencji składowych) klasy nie jest poprzedzona żadnym specyfikatorem, to domyślnym (dla kompilatora) jest **private**. Dlatego zapis:

```
class Point {  
    double x;  
    double y;  
}
```

jest równoważny:

```
class Point {  
private:  
    double x;  
    double y;  
}
```

5

Struktura różni się od klasy tym, że wszystkie jej elementy są publiczne. Dlatego zapis:

```
struct Point {  
    double x;  
    double y;  
}
```

jest równoważny:

```
class Point {  
public:  
    double x;  
    double y;  
}
```

5

Zadania

Zadanie 1

Zmodyfikuj deklarację klasy `TimeSpan` (listing 1), pozwalającą na reprezentację czasu w C++, w taki sposób, aby klasa gwarantowała poprawność danych (wartość sekund oraz minut musi być mniejsza od 60).

Listing 1: `TimeSpan.hpp`

```
class TimeSpan {  
public:  
    unsigned int hours;  
    unsigned int minutes; // [0 .. 59]  
    unsigned int seconds; // [0 .. 59]  
    TimeSpan(unsigned int seconds);  
};
```

W implementacji konstruktora, ustaw odpowiednio liczbę godzin, minut oraz sekund tak by ich suma odpowiadała podanej liczbie sekund (np. 3936s to 1h 6min i 36s). Deklaracje oraz implementacje umieść oddzielnie w odpowiednich plikach.

Następnie:

- Zaimplementuj metody umożliwiające dostęp do wszystkich pól klasy w trybie odczytu.
- Zaimplementuj metodę `print()` wyświetlającą na ekranie zawartość klasy `TimeSpan` w następujący sposób:

```
TimeSpan ts(3936);  
ts.print();      // wyświetla: 1:05:36
```

Wskazówka Aby wyświetlić liczby w formacie dwucyfrowym (np. 05) skorzystaj z odpowiednich funkcji z `iomanip`, np.:

```
cout << setfill('0') << setw(2) << 4;    // wyświetla: 04
```

Zadanie 2

Zaimplementuj klasę `BazaStudentow` do obsługi bazy danych osobowych studentów. Klasa powinna umożliwiać następujące operacje na bazie (funkcje publiczne):

- a) dodawanie nowego studenta (nr indeksu, imię oraz nazwisko),
- b) drukowanie całej listy studentów,
- c) wyszukiwanie studentów po numerze indeksu,
- d) usuwanie wybranych studentów (po numerze indeksu),
- e) zapis całej bazy do pliku,
- f) odczyt całej bazy z pliku,
- g) czyszczenie całej bazy studentów.

Napisz program, który wykorzystuje pełną funkcjonalność zaimplementowanej klasy i daje użytkownikowi możliwość wyboru czynności do wykonania.

Wskazówka 1 Możesz skorzystać z klasy `std::vector` do przechowywania listy studentów.

```
#include <vector>

/* ... */

5 // inicjalizacja listy studentów
std::vector<Student> bazaStudentow;

// dodawanie studenta do listy
Student s1 = wczytaj();
10 bazaStudentow.push_back(s1);

// pobieranie i-tego studenta (licząc od '0')
Student s2 = bazaStudentow[i];

15 // odczytywanie wielkości kolekcji (liczby elementów)
bazaStudentow.size()

// iterowanie po całej kolekcji
for (const Student& s : bazaStudentow) {
20     // ...
}

// usuwanie i-tego studenta (licząc od '0')
bazaStudentow.erase(bazaStudentow.begin() + i);

25 // czyszczenie bazy studentów
bazaStudentow.clear();
```

Zadanie 3*

Zaimplementuj obsługę drzewa typu BST przechowującego liczby całkowite w klasie `BSTree`. Klasa powinna posiadać następujące funkcje publiczne:

- konstruktor bezargumentowy – inicjalizujący obiekt,
- destruktor – zwalniający pamięć po obiekcie,
- funkcję `bool isEmpty()` – zwracającą `true` jeśli drzewo jest puste oraz `false` w przeciwnym wypadku,
- funkcję `void insert(int)` – wstawiającą nowy element do drzewa,
- funkcję `bool contains(int)` – zwracającą `true` jeśli drzewo jest zawiera podaną wartość oraz `false` w przeciwnym wypadku,
- funkcję `void clear()` – usuwającą wszystkie elementy z drzewa, oraz
- funkcję `void printInOrder()` – drukującą elementy drzewa w kolejności in-order.

Wszystkie pozostałe funkcje lub pola w klasie `BSTree` powinny być prywatne. Zastanów się, które z funkcji mogłyby być oznaczone modyfikatorem `const`. Swoją odpowiedź napisz w komentarzu.

Przetestuj działanie swojej implementacji z wykorzystaniem następującego fragmentu kodu:

```
BSTree bst;
cout << (bst.isEmpty()? "Drzewo jest puste" : "Drzewo nie jest puste") << endl
    ↪ ;
bst.insert(5);
bst.insert(3);
5 bst.insert(7);
bst.insert(4);
bst.insert(2);
cout << (bst.isEmpty()? "Drzewo jest puste" : "Drzewo nie jest puste") << endl
    ↪ ;
cout << "Drzewo zawiera element o wartosci 3: " << bst.contains(3) << endl;
10 cout << "Drzewo zawiera element o wartosci 9: " << bst.contains(9) << endl;
bst.printInOrder();
cout << endl;
bst.clear();
cout << (bst.isEmpty()? "Drzewo jest puste" : "Drzewo nie jest puste") << endl
    ↪ ;
```

Na następne zajęcia

- Dzierżyczenie
- Modyfikatory dostępu: **public**, **private** oraz **protected**.

Materialy:

- <http://www.cplusplus.com/doc/tutorial/inheritance/#inheritance>
- http://en.cppreference.com/w/cpp/language/derived_class