Lista zadań 7

Jeżeli w treści zadania nie podano inaczej – proszę używać strumieni plikowych (nie zapisywać danych do plików przez przekierowanie standardowych wyjść). Napisy można obsługiwać w dowolny sposób (tj. zarówno przez `char*` jak i `string`). Wszystkie funkcje w ramach danego zestawu powinny być zapisane w jednym pliku źródłowym oraz muszą umożliwić podanie argumentów, jako parametry uruchomienia programu. Należy również sprawdzać/diagnozować stan strumienia plikowego w trakcie odczytu/zapisu danych. Program ma być odporny na wystepowanie błędów (potocznie mówiąc ma być "idiotoodporny").

Proszę przeczytać treść wszystkich zadań przed implementacją któregokolwiek z nich. Jeżeli w treści zadania nie podano listy argumentów, nazwy funkcji i/lub typu zwracanego, to dany fragment sygnatury jest dowolny. Dozwolona jest implementacja dodatkowych funkcji. Każdy z Państwa może mieć swój własny pomysł na rozwiązanie zadań. Ważne jest, aby każdy program spełniał wytyczne opisane w treści zadania/zestawu.

Zestaw I

Zadanie 1

Zaimplementować i zaprojektować następujące jednostki danych:

- `SPiwo`, struktura z polami `wysokosc`, `promien_podstawy` i `marka`—,
- `SKarton`, struktura z polami: `piwa` (tablica obiektów typu `SPiwo`), `ile_piw` i `kod` (jako liczba całkowita).

W dalszych obliczeniach proszę przyjąć, że jednostką wysokości i promienia podstawy jest centymetr. Zakładamy, że każdy obiekt `SPiwo` reprezentuje walec wypełniony odpowiednim piwem.

Zadanie 2.

Napisać funkcję `wczytaj_piwa`, która zwraca dynamicznie utworzoną (wewnątrz funkcji) tablicę obiektów typu `SPiwo`. Argumentami tej funkcji są strumień wejściowy (`istream`) oraz `rozmiar` – zmiana tego ostatniego powinna być widoczna w programie po wywołaniu funkcji.
Zakładamy, że w pierwszym wierszu strumienia znajduje się liczba piw. Kolejne wiersze to wysokość, promień podstawy i marka. Na przykład plik postaci:

```
3
20 3.5 Klasyczne
22 3.0 Niepasteryzowane
21 3.65 Jasno-ciemne
```

reprezentuje trzy piwa: *Klasyczne* o wysokości 20 cm i promieniu 3.5 cm, *Niepasteryzowane* o wysokości 22 cm i promieniu 3.0 cm oraz *Jasno-ciemne* o wysokości 21 cm i promieniu 3.65 cm.

Zadanie 3.
Zaimplementować funkcję `pakuj_karton`, która przyjmuje jako argumenty:

- obiekt typu `SKarton` (będzie on modyfikowany wewnątrz funkcji, zmiana ma być widoczna na zewnątrz);
- tablica (jako wskaźnik) obiektów typu `SPiwo`;
- rozmiar tablicy `rozmiar`.

W funkcji tej należy zainicjalizować argument typu `SKarton` odpowiednimi danymi, tak aby były wypełnione pola `piwa` i `ile_piw`. Pole kod powinno zostać wypełnione:

- wartością stałą lub
- kolejną wartością całkowitą, tak aby każdy kolejny karton otrzymywał kod o jeden większy od poprzedniego, zaczynając od 1. Można wykorzystać zmienną globalną lub zmienną statyczną.

Zadanie 4.
Napisać funkcję, która wypisze na dany strumień wyjściowy informacje na temat obiektu typu `SKarton`, w tym:

- kod kartonu;
- listę wszystkich piw w kartonie.

Zadanie 5.
Zaimplementować funkcję, która dla danego obiektu typu `SPiwo` zwróci objętość danego piwa.

Zadanie 6.
Zaimplementować funkcję (lub funkcje), które w danym obiekcie typu `SKarton` posortują piwa według:

- wysokości;
- promienia podstawy;
- objętości piwa.

Sortowanie powinno odbywać się na oryginalnej tablicy obiektów typu `SPiwo` (bez kopiowania, z pominięciem stabilności sortowania itp.)

Zadanie 7.
Zaimplementować funkcję, która zapisze obiekt typu `SKarton` do strumienia wyjściowego (typu `ostream`). Format zapisu powinien być taki sam jak w zadaniu 2.

Zadanie 8.
Zaimplementować funkcję `daj_wysokosc_kartonu`, która dla danego obiektu typu `SKarton` zwróci jego wysokość. Załóżmy, że wysokość jest równa wysokości "najwyższego" piwa w kartonie.

Zadanie 9.
Zaimplementować funkcję `daj_objetosc_kartonu`, która dla danego obiektu typu `SKarton` zwróci sumę objętości wszystkich piw w kartonie (załóżmy milcząco, że opakowania są wykonane z gumy).

Zadanie 10.
Samodzielnie napisać kod testujący (program), który będzie wykorzystywać:

- wczytywanie i zapis obiektu typu `SKarton` z/do pliku/standardowego wejścia/wyjścia;
- wypisanie na ekran informacji o kartonie;
- sortowanie i zapis posortowanego obiektu `SKarton` do pliku;
- wypisanie na ekran wysokości i objętości kartonu.

Zestaw II

Zadanie 1
Zaimplementować i zaprojektować następujące jednostki danych:

- `SMieszkanie`, struktura z polami: `numer`, `powierzchnia`, `wysokosc`;
- `SPietro`, struktura z polami: `mieszkania` (tablica obiektów typu verb—SMieszkanie—), `ile_mieszkan` i `nr_pietra`.
- `SBlok`, struktura z polami: `pietra` (tablica obiektów typu `SPietro`), `ile_pieter` i `numer`.

Zadanie 2
Zaimplementować i zaprojektować funkcje:

- `wczytaj` i `zapisz`, umożliwiające odpowiednio wczytanie i zapisanie danych o obiekcie typu `SBlok` z/do strumienia (pliku lub `cout/cin`). Format zapisu jest dowolny, ale każdy obiekt typu `SBlok` musi posiadać informacje o numerze, liczbie pięter i opis każdego piętra. Podobnie z każdym piętrem: wymagana jest informacja o numerze piętra, liczbie mieszkań i opis każdego mieszkania na piętrze;

- `wysokosc_pietra`, która dla danego piętra obliczy wysokość piętra. Założono, że jest to największa wysokość z mieszkań na danym piętrze;

- `wysokosc_bloku`, która dla danego bloku policzy jego wysokość. Założono, że jest to suma wysokości wszystkich pięter (pod wysokością piętra rozumiemy wynik działania funkcji `wysokosc_pietra`);

Zadanie 3
Napisać funkcję `powierzchnia_calkowita`, która obliczy sumę powierzchni wszystkich mieszkań w całym bloku.

Zadanie 4
Napisać funkcję `numery_na_pietrze`, która zwróci numery od pierwszego do ostaniego mieszkania na wybranym przez użytkownika pietrze (numer podaje użytkownik).