ZMP - Lista 7

Marcin Michalski, WMAT PWr

Maj 2024

Zadania tradycyjnie należy rozwiązać zgodnie ze specyfikacją i udostępnić prowadzącemu laboratoria.

Deadline: 26.05.2024, 23:59

Ćwiczenie 1. Stwórz klasę `Biblioteka`, której z polami reprezentującymi książki (tytuł, rok wydania, etc. oraz liczba dostępnych sztuk) z metodami do ich wypożyczania, oddawania i sprawdzania dostępności.

Ćwiczenie 2. Dla kolejki i stosu (i innych poznanych struktur?) wykonaj istotne fragmenty Zadania 2.

Ćwiczenie 3. Stwórz klasę `Traffic_lights` z polem `light`, które może przyjąć jedną z wartości `green, yellow, red` (użyj do tego np. `enum`), oraz metodą `next` zwracającą kolor światła następującego po obecnym. Następnie użyj tej metody do przeciążenia operatora inkrementacji `++` dla tej klasy.

Rozwiązania zadań umieść w `l7/zi`, gdzie `i`, to nr zadania, w swoim repozytorium. Pliki nazwij `<nr_indeksu>zic.cpp`, gdzie `i`, to nr zadania, a znak `c` to podpunkt (jeśli zadanie nie ma podpunktów, to `c` należy pominąć). Przykład: `123456z1a.cpp`.

Zadanie 1 (5p.)**.** Zaimplementuj klasę `Zwierz` składającą się z pól `imie`, `rasa` i metody czysto wirtualnej `sound` oraz klasy `Kot` i `Pies` z niej dziedziczące. Dla klas potomnych przeciąż operatory `IO` do poprawnego wczytywania i wyświetlania obiektów tych klas. Wyświetlenie obiektu powinno pokazać imię i rasę zwierzęcia oraz informację o tym, jaki dźwięk wydaje (używając metody `sound`).

Dla testów przyjmijmy, że na standardowym wejściu przyjdą kolejno imię i rasa kota, a potem imię i rasa psa, program zainicjuje odpowiednie obiekty, a natępnie wypisze je na standardowym wyjściu.

Zadanie 2 (5p.)**.** Dla drzewa binarnego przechowującego w każdym węźle liczbę całkowitą przeciąż operatory arytmetyczne: unarny `-`, binarne `+, -` oraz operator `[]`. "Dodawanie" drzew powinno odbyć się po węzłach, tzn. wartości w korzeniach dodajemy do siebie i wynik stanowi wartość w korzniu drzewa wynikowego, suma wartości w lewych potomkach jest wartością w lewym potomku drzewa wynikowego, itd.[1]. Dla poprawnego działania operatora `[]` zaproponuj odpowiednie kodowanie węzłów drzewa za pomocą liczb naturalnych[2].

[1] Jeśli któreś z drzew się skończy, to dalej powinna być kopiowana reszta niepustego drzewa.

[2] Przydatne mogą być pewne modyfikacje implementacji drzew, np. można przechowywać dodatkowo wielkość drzewa lub nawet dla każdego węzła przechowywać liczbę węzłów potomnych.

Dla testów przyjmijmy, że na standardowym wejściu przyjdą liczby naturalne n i k oznaczająca wielkość nadchodzących drzew, a następnie $n + k$ liczb naturalnych, które powinny zostać rozsądnie[3] umieszczone w drzewach. Następnie program wyświetli drzewa `-t1`, `t1+t2`, `t2-t1` oraz wypisze kolejne wartości drzewa `t1` za pomocą `[]`.

[3]Np. tak, jak na liście 5.