Podstawy programowania

→ Zadania 3

Zadania

Zadanie III.1.

Napisz program, który losuje 1000 liczb całkowitych i zlicza, ile z nich jest parzystych.

Zadanie III.2.

Napisz funkcję, która dla podanej liczby całkowitej oblicza jej silnię.

Zadanie III.3.

Napisz funkcję, która na podstawie długości boków prostokąta oblicza i zwraca długość jego przekątnej.

Zadanie III.4.

Napisz funkcję, która przyjmuje dwie liczby całkowite dodatnie. Liczby te reprezentują godziny wyrażone w sekundach. Funkcja powinna zwracać napis "HH:MM:SS - HH:MM:SS", gdzie godziny są obliczone na podstawie podanych sekund i wcześniejsza godzina zawsze pojawia się pierwsza.

Zadanie III.5.

Napisz program, który oblicza pole trójkąta na podstawie podanych danych. Dane, które może przekazać użytkownik to: długości boków, miary kątów wewnętrznych, promień okręgu wpisanego i promień okręgu opisanego. Z podanych danych użytkownik może podać wszystkie lub tylko niektóre. Wszystkie dane wyrażone są jako liczby zmiennoprzecinkowe. Kąty podane są w stopniach.

Program powinien mieć zaimplementowane funkcje obliczania pola, realizujące wszystkie wzory podane w maturalnej karcie wzorów. Program rozpoznaje, czy podane dane są wystarczające do użycia któregoś z tych wzorów i oblicza pole. Jeśli dane są niewystarczajce, wypisuje stosowny komunikat.

Zadanie III.6.

Napisz program, który oblicza pole czworokąta. Użytkownik podaje długości wszystkich boków i miary wszystkich kątów (w stopniach) jako liczby zmiennoprzecinkowe. Program powinien posiadać funkcję, która na podstawie wskazanych danych rozpoznaje, czy czworokąt jest kwadratem, prostokątem, rombem, równoległobokiem, trapezem czy deltoidem (ale ma wskazywać tylko jedno z nich!). Powinien też posiadać funkcje obliczające pole dla każdego z podanych czworokątów na podstawie przekazanych danych. Jeżeli czworokąt nie jest żadnym z wymienionych, program wypisuje komunikat o niepowodzeniu.

Zadanie III.7.

Zmodyfikuj rozwiązania poprzednich dwóch zadań w ten sposób, by zabezpieczyć działanie programu przed błędem użytkownika. Jeżeli użytkownik wprowadzi coś innego niż liczbę (lub nic), program powinien wypisywać stosowny komunikat i prosić o ponowne podanie danych.

Następujące zadania będą stanowiły kolejne etapy implementacji gry *Super Farmer* stworzonej na podstawie koncepcji Karola Borsuka. Do listy zadań dołączona jest instrukcja tej gry, z którą należy się zapoznać.

Ponadto należy wiedzieć, że na jednej kostce znajdziemy 6 królików, 3 owce, 1 świnię, 1 krowę i wilka a na drugiej 6 królików, 2 owiece, 2 świnie, 1 konia i lisa.

Zadanie III.8.

Napisz funkcję, która przetwarza informacje o aktualnie posiadanym przez gracza stadzie oraz wartościach wyrzuconych na kostkach i na tej podstawie zwraca nowy stan stada gracza. Tymczasowo zignoruj ograniczenia na łączą liczbę zwierząt w grze.

Podpowiedź: Warto zdefiniować strukturę Gracz, w której będziemy przechowywać informacje o posiadanych zwierzętach.

Podpowiedź 2: Aby ułatwić operacje na typach zwierząt, warto zdefiniować enum z ich nazwami, by kod był bardziej przejrzysty.

Zadanie III.9.

Napisz funkcję do obsługi wymiany zwierząt. Funkcja ta przyjmuje informacje na temat aktualnego stanu stada gracza, typów zwierząt do wymiany (które na które) oraz liczby zwierząt, które chce się uzyskać (tj. zwierząt drugiego typu). Funkcja ma obsługiwać tylko proste wymiany zgodne z tabelą (czyli nie wymieniamy królików na krowy). Funkcja zwraca nowy stan stada gracza po wymianie. Jeżeli aktualne stado gracza nie pozwala na taką wymianę, funkcja zwraca stado bez żadnych zmian (nie robimy wymian częściowych). Tymczasowo zignoruj ograniczenia na łączą liczbę zwierząt w grze.

Podpowiedź: Tabele wymiany warto zakodować w postaci stałych (jakkolwiek rozumianych), by można było z nich korzystać wewnątrz różnych funkcji.

1 owca = 6 królików	1 koń = 2 krowy
1 świnia = 2 owce	1 mały pies = 1 owca
1 krowa = 3 świnie	1 duży pies = 1 krowa

Zadanie III.10.

Napisz funkcję do rzucania kośćmi. Funkcja nie przyjmuje argumentów i zwraca dwie wartości: wynik rzutu pierwszą kością i wynik rzutu drugą kością. Pamiętaj, że kości

różnią się od siebie. Zwróć uwagę, że wartości na kościach powtarzają się, więc prawdopodobieństwa określonych wartości są różne.

Zadanie III.11.

Napisz funkcję, która na podstawie stanu stada gracza zwraca wartość `true`, jeśli spełnia on wymagania do wygranej, oraz `false` w przeciwnym wypadku.

Zadanie III.12.

Napisz program, który będzie obsługiwał grę *Super Farmer*. Gra po uruchomieniu pyta o liczbę graczy (od 2 do 4). Każdemu z nich przypisuje stado. Następnie program pozwala kolejno każdemu graczowi wykonać swoją turę. Gra kończy się, gdy pojawi się zwycięzca. Wykorzystaj funkcje napisane w poprzednich zadaniach.

Zadanie III.13.

Zmodyfikuj program z poprzedniego zadania w taki sposób, by uwzględnić maksymalną liczbę zwierząt dostępnych w grze (tj. gra ma ograniczenie ze względu na liczbę posiadanych kartoników dla każdego gatunku zwierząt).