Motorola Science Cup^3
Zadania II Etapu

Programowanie w języku Rockstar

OMÓWIENIE

Rockstar jest ezoterycznym językiem programowania stworzonym przez Dylana Beattie. Kod w języku Rockstar przypomina tekst utworu rockowego.

Przydatne materiały:

- Wystąpienie Dylana Beattie na konferencji NDC w 2020, podczas którego zaprezentował język Rockstar: https://www.youtube.com/watch?v=6avJHaC3C2U
- Repozytorium i dokumentacja: https://github.com/RockstarLang/rockstar
- Interpreter języka: https://codewithrockstar.com

Prosimy o przygotowanie w języku Rockstar implementacji dwóch popularnych algorytmów:

- Sortowanie bąbelkowe
- Sito Eratostenesa

oraz rozwiązanie zadania polegającego na liczeniu ruchów piłki w grze typu Arkanoid.

Jako rozwiązanie zadania prosimy o przesłanie (dla każdego zadania osobno):

- kodu źródłowego (plik z rozszerzeniem .rock)
- zestawu danych na których testowaliście program oraz jego wynik działania
- pozostałej dokumentacji - wszystko, czym chcielibyście się pochwalić i na co Waszym zdaniem powinniśmy zwrócić uwagę, oceniając program

Źródła możecie udostępnić jako archiwum w serwisie chmurowym (Dysk Google, OneDrive, etc) lub projekt w systemie kontroli wersji (np. GitHub).

Nasz zespół oceniając Wasze prace będzie korzystał z interpretara dostępnego pod adresem: https://codewithrockstar.com/online.

Prace, które nie uruchomią się przy użyciu interpretera dostępnego na oficjalnej stronie Rockstar, nie będą oceniane.

CO OCENIAMY

Oceniać będziemy:

1. Zgodność rozwiązania ze specyfikacją (zobacz opisy poszczególnych etapów zadania)
2. Sposób rozwiązania, czystość kodu
3. Dokumentację projektu
4. Kreatywność - punkty za tekst utworu. Im lepiej będzie “brzmiał” Wasz kod, tym więcej punktów! Nie musicie ograniczać się do rocka, wszystkie gatunki muzyki są mile widziane.

Łączna liczba punktów do zdobycia - **120**

Jury obiecuje dodatkowy bonus za muzyczne wykonanie Waszego utworu (**0-15** punktów). Możecie stworzyć własny zespół i instrumenty, ale wykonanie innego zespołu jest również mile widziane.

ETAPY ZADANIA

Etap I - Sortowanie bąbelkowe

Za ten etap zadania uzyskacie do **30** punktów (implementacja algorytmu **0-15**, sposób przygotowania danych wejściowych **0-5,** kreatywność i warstwa liryczna utworu **0-10**).

Sortowanie bąbelkowe jest prostą metodą sortowania polegającą na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, jeśli zaburza ona porządek, w jakim się sortuje tablicę. Sortowanie kończy się, gdy podczas kolejnego przejścia nie dokonano żadnej zmiany (źródło: https://pl.wikipedia.org/wiki/Sortowanie_bąbelkowe).

Opis:

- Przygotujcie nie posortowaną tablicę min. 10 elementową zawierającą liczby całkowite z przedziału 0 - 1000 i wypiszcie jej zawartość na ekran. Tablica może być zainicjalizowana w kodzie programu lub podawana przez użytkownika.
- Napiszcie program sortowania bąbelkowego, który uporządkuje waszą tablicę i wypiszcie rezultat na ekran.

Etap II - Sito Eratostenesa

Ten etap wart jest do **30** punktów (implementacja algorytmu **0-15**, sposób przygotowania danych wejściowych **0-5,** kreatywność i poetyckość utworu **0-10**).

Sito Eratostenesa - metoda pozwalająca wyznaczyć wszystkie liczby pierwsze nie większe niż dana liczba naturalna n (źródło: https://encyklopedia.pwn.pl/haslo/;3898380).

Opis:

- Zaprogramujcie algorytm sita Eratostenesa, który znajdzie i wyświetli wszystkie liczby z przedziału od 2 do n, gdzie n jest zadaną liczbą naturalną większą od 2.
- Liczba n może być zainicjalizowana w kodzie programu lub podawana przez użytkownika.

Etap III - Arkanoid - liczba ruchów potrzebna na zbicie wszystkich bloków

Ten etap wart jest do **60** punktów (implementacja algorytmu **0-30**, sposób rozwiązania zadania i czystość kodu **0-10**, sposób wprowadzania danych wejściowych **0-10**, kreatywność i warstwa liryczna **0-10**).

Na czym polega gra?

Za pomocą małej ruchomej platformy odbija się piłeczkę. Piłka odbija się od ścian bocznych i górnej prostokąta oraz od bloków rozmieszczonych wewnątrz pola gry. Celem gry jest odbijanie piłki w taki sposób, aby zbić wszystkie rozmieszczone bloki. Blok jest rozbijany w momencie, kiedy piłeczka w niego uderzy. W naszej implementacji zakładamy, że piłka odbija się wewnątrz prostokąta - nie może wypaść poza pole gry.

Założenia (źródło: https://archive.algo.is/camps/mipt/2016/statement6a.pdf)

- Pole gry jest prostokątem o szerokości m i wysokości n, gdzie m jest liczbą nieparzystą oraz m i n są względnie pierwsze.
- Pole gry znajduje się w układzie współrzędnych, gdzie lewy dolny róg to punkt (0, 0), a prawy górny róg to punkt (m, n).
- Początkowo platforma z piłką znajduje się na środku dolnej krawędzi pola. Pozycja początkowa piłki to $(\frac{m}{2},\ 0)$.
- Pierwszy ruch piłki odbywa się w lewo i do góry, tj. zgodnie z wektorem $(-\frac{1}{2},\ \frac{1}{2})$.
- Wielkość piłki jest pomijalna.

- Liczba bloków nie może być większa niż rozmiar pola gry - 1. Miejsce, z którego startuje piłka, musi pozostać puste. Jeden blok zajmuje kwadrat o boku 1. Oznaczmy prawy górny róg bloku jako (x, y), wtedy lewy dolny róg znajduje się w punkcie (x - 1, y - 1).
- Piłka po uderzeniu w krawędź bloku zmienia kierunek ruchu, a blok jest zbijany - znika z planszy.
- Piłka po uderzeniu w krawędź pola zmienia kierunek ruchu.
- Kąt odbicia jest stały i wynosi 45°.
- Gracz nie przegrywa, tj. gra toczy się tak długo, aż wszystkie bloki zostaną zbite.
- Wymiar pola gry oraz liczba i rozmieszczenie bloków mogą być zdefiniowane w treści programu lub podawane dynamicznie przez użytkownika (można uzyskać do 10 punktów za dynamiczną implementację).

Opis

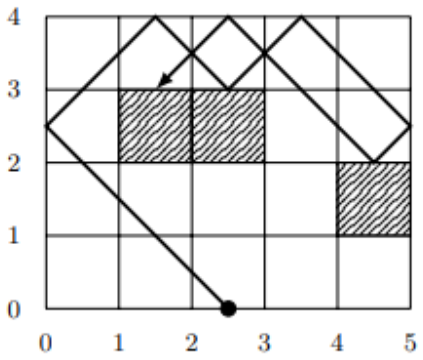
- Określcie wymiar pola gry oraz liczbę i rozmieszczenie bloków. Te dane mogą być zdefiniowane w algorytmie lub podawane dynamicznie przez użytkownika (można uzyskać do 10 punktów za dynamiczną implementację). Potrzebne będą dane:
    - m - szerokość pola gry
    - n - wysokość pola gry
    - rozmieszczenie bloków podane jako zbiór współrzędnych punktów (x, y) stanowiących prawy górny róg bloków
- Zasymulujcie ruch piłki zgodnie z podanymi założeniami, zliczając czas gry.
- Piłka porusza się ze stałą prędkością $V = \frac{\sqrt{2}}{2}$ bloków na sekundę.
- Zasymulujcie zbijanie bloków.
- Kontynuujcie “grę” aż do usunięcia wszystkich bloków z pola gry.
- Wynik to czas potrzebny do zbicia wszystkich bloków. Wypiszcie wynik na ekranie.

Przykład (źródło: https://archive.algo.is/camps/mipt/2016/statement6a.pdf)

- Dane wejściowe:

m	5
n	4
(x, y)	(2, 3) (3, 3) (5,2)

- Plansza i ruchy piłki



- Wynik: **22**