Programowanie Współbieżne

Zadanie PW-22

Semen Nachvynov

08.06.2025r.

WCY23IY2S1

mgr inż. Mierzejewski Krzysztof

Zadanie nr: PW-22 Język implementacji: Java: {Java, Kotlin, Scala}, .net {C#, F#}, Go, propozycja studenta Środowisko implementacyjne: JetBrains Termin wykonania: ostatnie zajęcia

Zasadnicze wymagania: a. liczba procesów sekwencyjnych powinna być dobrana z wyczuciem tak, aby zachować czytelność interfejsu i jednocześnie umożliwić zobrazowanie reprezentatywnych przykładów, b. kod źródłowy programu musi być tak skonstruowany, aby można było „swobodnie” modyfikować liczbę procesów sekwencyjnych (z wyjątkiem zadań o ściśle określonej liczbie procesów), c. graficzne zobrazowanie działania procesów współbieżnych, d. odczyt domyślnych danych wejściowych ze sformatowanego, tekstowego pliku danych (yaml, json, inne), e. [opcjonalnie] możliwość modyfikacji danych wejściowych poprzez GUI.

Sprawozdanie (w formie elektronicznej) powinno zawierać następujące elementy: 1) stronę tytułową, 2) numer i niniejszą treść zadania, 3) syntetyczny opis problemu – w tym wszystkie przyjęte założenia, 4) wykaz współdzielonych zasobów, 5) wykaz wyróżnionych punktów synchronizacji, 6) wykaz obiektów synchronizacji, 7) wykaz procesów sekwencyjnych, Powyższe ilustrować adekwatnymi sekcjami kodu źródłowego programu

Problem do rozwiązania: Załadunek koparkami. Założenia: Napisać program synchronizację podstawiania samochodów pod załadunek koparką. Należy założyć istnienie N koparek i K samochodów o małej ładowności i L samochodów o dużej ładowalności. Jedna koparka obsługuje równocześnie dwa małe pojazdy lub jeden duży. Dodatkowo należy przyjąć, że operatorzy koparek mają przerwę obiadową między godziną 1300 i 1400, a niecierpliwi kierowcy rezygnują z załadunku jeśli nie podjadą na stanowisko w ciągu założonego maksymalnego czasu oczekiwania np. 0.5 godz

Opis problemu oraz przyjęte założenia

Kolejka samochodów jest zawsze jedna, co pozwala umieścić ją w głównym wątku, więc wątkami potomnymi są koparki (1:1).

W programie zegar jest przyspieszony tak, aby w 1 sekundę mijało 10 minut.

Przydział samochodów do koparki odbywa się natychmiast, czyli jeżeli program sprawdza ile samochodów może obsłużyć koparka (patrzy czy są obok 2 małe samochody lub 1 duży), po czym wypisuje samochody z kolejki, a koparka się zawiesza na wyliczoną liczbę sekund, ale dla wprowadzenia różnorodności dodany jest element losowy. Można swobodnie to konfigurować.



Przerwa odbywa się od 13 do 14, przy tym jeżeli jakaś koparka obsługiwała samochód, to dokończy tę pracę.

Za niecierpliwego kierowcę przyjąłem 1 osobę w kolejce. Przykład: zaczęła się przerwa 1-godzinna, kierowca z przodu ma cierpliwość 20 minut. Wieć jak upłynie 20 minut – zrezygnuje i wyjdzie z kolejki. Zostało 40 minut przerwy, więc po 20 minutach kolejny kierowca zrezygnuje (bo stał się pierwszym w kolejce). I tak 3 samochody zrezygnują podczas przerwy.Podczas działania programu wyświetla się lista numerów samochodów, które zrezygnowały z załadunku.



Przerwa między 13 a 14

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

Liczba wątków zalezy od liczby koparek

Obraz zawierający tekst, Czcionka, zrzut ekranu

Zawartość wygenerowana przez AI może być niepoprawna.

Niecierpliwość kierowcy i dodawanie do listy nieobsłużonych

Wykazy

Program nie używa funkcjonalności pamięci współdzielonej – kolejka oraz listę nieobsłużonych samochodówprzechowują się w głównym wątku.

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Zawartość wygenerowana przez AI może być niepoprawna.

Punktem synchronizacji jest wątek główny – zarządza gdzie, ile i jakich samochodów podjedzie do załadunku.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez AI może być niepoprawna.

Nadzorca kolejki oraz licznik czasu. Jeżeli któryś samochód przekroczy próg niecierpliwości to odjeżdża, czyli przenosi się z kolejki carQueue do skippedCars. Po przerwie – podczas 85 sekundy (przy założeniu, że doba ma 144 sekundy) jest wysyłany komunikat do kazdego wątku o wznowieniu działania – usypiamy je na 0 sekund, co zmusza ich natychmiast zgłośić się po kolejne samochody.

Obraz zawierający tekst, zrzut ekranu

Zawartość wygenerowana przez AI może być niepoprawna.

Po zgłoszeniu gotowości przez wątek potomny – macierzysty odhacza go u siebie wewnętrzenie jako samochód gotowy do pracy i zmienia status z kopiącego na wolny. Następnie sprawdza, czy jest w stanie przydzielić do koparki 1 duży, 1 mały albo 2 małych samochody i zawiesza wątek na częściowo losowy okres czasu.

Obiektem synchronizacji są komunikaty – wątki potomne informują wątek główny o gotowości dalszej pracy i czekają na kolejne zadanie.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

Oczekują na komunikaty od wątku macierzystego z czasem zawieszenia się – symulują ładowanie piasku do samochodów. Wykorzysywane są tzw. porty, czyli określone kanały, na których wątek czeka na powiadomienia.

Do procesów sekwencyjnych można przydzielić inicjalizację, przydział samochodów do załadunku oraz dodawanie obiektów na ekran (nie chodzi o odświeżanie, tylko kolejność, w jakiej elementy pojawiają się na ekranie).

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez AI może być niepoprawna.

Sprawdzamy czy nie ma przerwy i czy kolejka nie pusta, po czym przydzielami samochody do koparki.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

Generowanie workerów (wątków)

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez AI może być niepoprawna.

Dodawanie samochodów na makiet ekranu