WOJSKOWA AKADEMIA TECHNICZNA

Wydział Cybernetyki

SPRAWOZDANIE

Z PROJEKTU

Programowanie Współbieżne

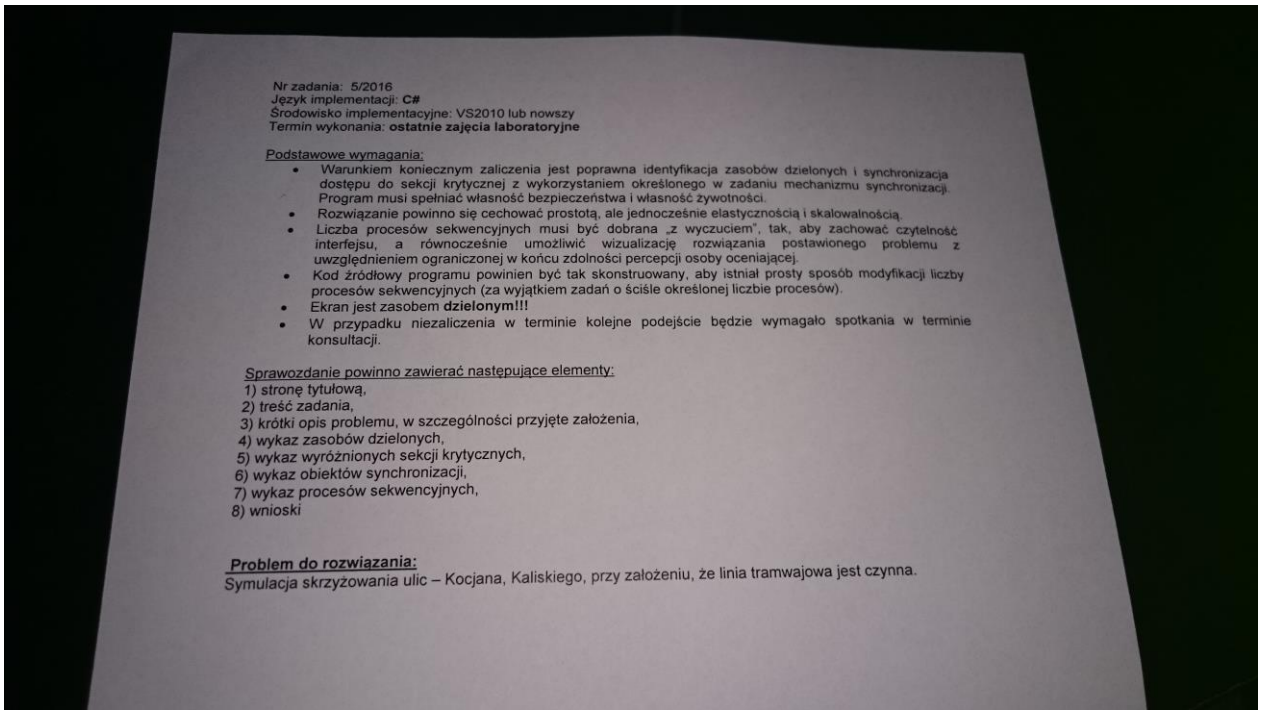
Prowadzący: mgr. inż. Marcin Paź

Wykonał: st. kpr. pchor. Damian KRATA (Nr albumu: 59223)

Grupa: I4X3S1

Data wykonania: 06.06.2016r.

1.Treść zadania:



Nr zadania: 5/2016
Język implementacji: **C#**
Środowisko implementacyjne: VS2010 lub nowszy
Termin wykonania: **ostatnie zajęcia laboratoryjne**

Podstawowe wymagania:

- Warunkiem koniecznym zaliczenia jest poprawna identyfikacja zasobów dzielonych i synchronizacja dostępu do sekcji krytycznej z wykorzystaniem określonego w zadaniu mechanizmu synchronizacji. Program musi spełniać własność bezpieczeństwa i własność żywotności.
- Rozwiązanie powinno się cechować prostotą, ale jednocześnie elastycznością i skalowalnością.
- Liczba procesów sekwencyjnych musi być dobrana „z wyczuciem", tak, aby zachować czytelność interfejsu, a równocześnie umożliwić wizualizację rozwiązania postawionego problemu z uwzględnieniem ograniczonej w końcu zdolności percepcji osoby oceniającej.
- Kod źródłowy programu powinien być tak skonstruowany, aby istniał prosty sposób modyfikacji liczby procesów sekwencyjnych (za wyjątkiem zadań o ściśle określonej liczbie procesów).
- Ekran jest zasobem **dzielonym!!!**
- W przypadku niezaliczenia w terminie kolejne podejście będzie wymagało spotkania w terminie konsultacji.

Sprawozdanie powinno zawierać następujące elementy:
1) stronę tytułową,
2) treść zadania,
3) krótki opis problemu, w szczególności przyjęte założenia,
4) wykaz zasobów dzielonych,
5) wykaz wyróżnionych sekcji krytycznych,
6) wykaz obiektów synchronizacji,
7) wykaz procesów sekwencyjnych,
8) wnioski

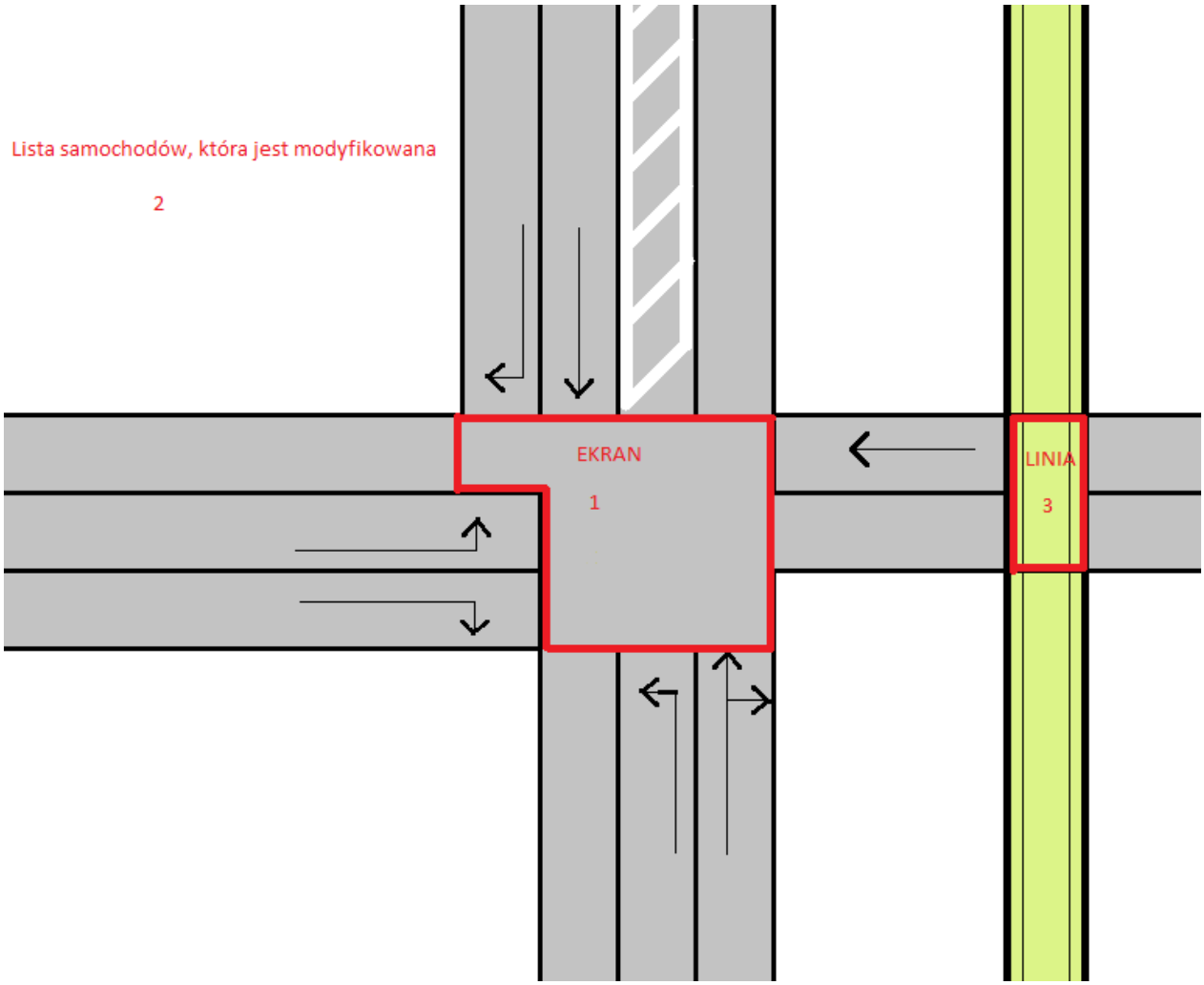
Problem do rozwiązania:
Symulacja skrzyżowania ulic – Kocjana, Kaliskiego, przy założeniu, że linia tramwajowa jest czynna.

2. Krótki opis problemu, w szczególności przyjęte założenia:

Moim problemem było graficzne pokazanie symulacji ruchy drogowego. Przy rozwiązaniu problemu nie rozważałem sytuacji, że np. tory jazdy samochodów mogą się przeciąć. Spowodowane jest to tym, że w realnym życiu skrzyżowania mają zapobiegać takim sytuacjom. W moim projekcie zakładam, że:

- Jest jeden tramwaj, który zawsze jedzie od „od dołu", czyli od strony autostrady (pokazuje tylko mechanizm).
- Samochody dodają się na przypadkowych trasach.
- Kolory samochodów są przypadkowe.
- Samochody zawsze startują od początku trasy.
- Nie ma światła pomarańczowego, tylko na pewien czas wszystkie światła się wyłączają.

3. Wykaz zasobów dzielonych:



4. Wykaz wyróżnionych sekcji krytycznych:

- Dodawanie samochodu do listy samochodów.
- Usuwanie samochodu z listy samochodów.
- Wjazd na skrzyżowanie.
- Opuszczenie skrzyżowania.

5. Wykaz obiektów synchronizacji:

- Samochody.
- Światła.

6. Wykaz procesów sekwencyjnych:

- Proces modyfikowania listy samochodów.
- Proces jazdy samochodów.
- Proces poprawnego załączania świateł.

7. Sposób realizacji danych problemów.

Na prawdziwym skrzyżowaniu takim gwarantem bezpiecznego opuszczenia skrzyżowania przez samochód jest pomarańczowe światło. Daje ono chwile czasu na to, aby nawet samochód który ostatni (regulaminowo) wjechał, opuścił skrzyżowanie nikomu nie wadząc. Tak samo jest u mnie, z tą jednak różnicą, że nie zapala się pomarańczowe światło, a jedynie na jakiś ułamek sekundy ponownie wszystkie światła stają się czerwone.

Aby wyświetlać samochody na ekranie, musimy je najpierw mieć. Dlatego przechowujemy wszystkie samochody w liście samochodów. Tam też przechowujemy wszystkie ich dane. Niestety w moim programie są trzy funkcje, które korzystają z tej wspólnej listy. Wywoływana jest ona dla operacji dodawania samochodów do listy (addCar), usuwania z listy (removeOutputCars), oraz jazdy samochodu (car_drive). Aby rozwiązać ten problem, zdecydowałem się na użycie semafora binarnego, który na początku przyjmuje wartość 1.

```
Semaphore carList_access = new Semaphore(1, 1);
```

Następnie po wejściu w którąkolwiek funkcję mającą dostęp do tego zasobu współdzielonego, semafor carList_access jest opuszczany, a po wyjściu podnoszony. Bez zastosowania tego mechanizmu pojawiał się błąd związany w „dostępem do kolekcji”.

Jeżeli chodzi o sytuację, w której samochód zostaje skasowany przez nadjeżdżający tramwaj, to niestety ale nie może do niej dojść. Wszystko przez warunek zapalania czerwonego światła (jeden if). To właśnie przez ten warunek, w funkcji, która powoduje poruszanie się samochodu po określonej trasie, możemy sprawdzać nie tylko czy samochód aby na pewno nie zatrzymał się na torach, ale także czy przez przypadek na nie nie wjedzie.

8.Wnioski:

Cel projektu uważam za osiągnięty, nauczyłem się korzystać z semaforów oraz przekładać problemy ogólne na język informatyki. Nauczyłem się na siłę nie stosować wyszukanych mechanizmów synchronizacji tam, gdzie wystarczy zwykły warunek, albo sprawę załatwia dobrze ustawiona flaga. Ponadto zapoznałem się z językiem c# oraz środowiskiem Visual Studio.