Programowanie Obiektowe

LABORATORIUM 1
PROJEKTOWANIE I PROGRAMOWANIE HIERARCHII KLAS OBIEKTÓW DYSKRETNYCH

Poruszane zagadnienia z zakresu programowania:

- Klasy i enkapsulacja.
- Klasa abstrakcyjna, klasa interfejsowa.
- Dziedziczenie i polimorfizm obiektowy dynamiczny.
- Generatory liczb losowych.
- Serializacja i deserializacja obiektu

Poruszane zagadnienia z zakresu modelowania matematycznego:

- Symulacja modelu ARX.
- Rozwiązywanie równania dyskretnego.

Zadanie do wykonania.

- Zdefiniuj klasę interfejsu o nazwie *ObiektWeWy, ObiektSISO* lub podobnej. Klasa będzie przeznaczona do dziedziczenia przez wszystkie klasy w programie, które będą modelowały obiekty o jednym wejściu i jednym wyjściu. Główną (jedyną?) metodą klasy powinna być metoda *symuluj()*, która powinna akceptować wartość typu *double* (wejście) i zwracać wartość tego samego typu (wyjście).
- Zdefiniuj klasę *ModelARX* (lub o podobnej nazwie) jako pochodną od klasy *ObiektSISO*. Klasa powinna zawierać pola (dane składowe) pozwalające na symulację modelu ARX (patrz wykład). W szczególności powinny się w niej znaleźć:
  - pamięci współczynników dla wielomianów A (mianownik) i B (licznik),
  - pamięć dla sygnałów wejściowych,
  - pamięć dla sygnałów wyjściowych,
  - pamięć dla opóźnienia transportowego,
  - zmienna pozwalająca określić rząd opóźnienia transportowego (>= 1),
  - „moc” zakłócenia (np. odchylenie standardowe jeżeli wybrany będzie rozkład normalny).
- Zdecyduj, czy warto zapisać osobno klasę wielomianu dyskretnego, zawierającą pamięci i obliczenia dla jednego wielomianu, a następnie użytą dwukrotnie: dla licznika i mianownika modelu ARX.
- Zdecyduj, jakie funkcje potrzebujesz. Pamiętaj o zasadzie YAGNI ("You aren't gonna need it") - nie definiuj funkcji których zastosowania nie przewidujesz.
- Napisz tymczasową wersję metody prywatnej do symulacji jednej próbki sygnału zakłócenia. Ta wersja powinna tylko zwracać 0.0, gdyż wprowadzenie losowości na tym etapie utrudniłoby sprawdzenie poprawności implementacji.
- Zdefiniuj implementację wirtualnej metody symuluj, tak aby poprawnie symulowała działanie modelu ARX (patrz równanie różnicowe na wykładzie). Uwaga: zabrania się używania pętli do implementacji algorytmu, w zamian należy użyć algorytmu standardowego.
- Wykonaj dla kasy ModelARX, testy jednostkowe dostarczone w pliku „lab1.cpp”, które sprawdzą poprawność symulacji. W razie niepowodzenia testów, poprawiaj kod symulacji, aż do osiągniecia sukcesu. Przyjmij, że weryfikacja w testach działa prawidłowo. Jeśli kod testów nie jest zgodny z interfejsem klasy, to go dostosuj.
  - Wskazówki: Zauważ, że przy wyniku negatywnym testy wypisują wynik spodziewany i faktyczny. Precyzję wyświetlania i tolerancję numeryczną testów można zmieniać stałymi PREC i TOL w kodzie funkcji pomocniczych.
  - Testy sprawdzają wartości sygnałów z pewną niewielką tolerancją. Jest to istotne przy porównywaniu liczb rzeczywistych, gdyż możliwe są drobne błędy numeryczne, nie oznaczające od razu błędów w implementacji. Wartości testowe także zostały zapisane z ograniczoną precyzją! Testy te zakładają zerową wariancję zakłócenia, gdyż w innym wypadku wartości sygnału wyjściowego są nieprzewidywalne i nie możliwe do tak prostego testowania.

- Dopiero gdy metoda symuluj „zaliczy” wszystkie testy na „OK!”, zaimplementuj metodę prywatną do symulacji jednej próbki sygnału zakłócenia. Zakłócenie powinno mieć zerową wartość oczekiwaną i ustawianą „moc” (np. odchylenie standardowe dla rozkładu normalnego).
- W klasie *ModelARX* zaimplementuj metodę użytkową do wykonania serializacji instancji obiektu. Serializacja to proces przekształcenia instancji w ciąg bajtów lub znaków, tak aby możliwy był ich zapis do plików, przesyłanie przez sieć itp. Samodzielnie dobierz sposób serializacji, np. skorzystaj z gotowych rozwiązań jak format klucz=wartość, JSON, boost::archive, lub napisz własny sposób serializacji.
- W klasie ModelARX dodaj konstruktor konwertujący dokonujący deserializacji instancji z formatu wybranego/opracowanego w poprzednim poleceniu. Konstruktor powinien odtworzyć cały stan obiektu.