НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп‘ютерних систем

**Лабораторна робота №2  
*З дисципліни: «Операційні системи»***

«Синхронізація процесів»

Студентка групи КВ-42

Брикалова Вікторія

Перевірив(ла):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2017

**Завдання**

1. Розробити програму, що моделює роботу заданого об’єкта, використовуючи для доступу процесів до подільних ресурсів засоби синхронізації. Об’єкт моделювання і засоби синхронізації процесів визначаються варіантом завдання (табл.2.1).

Вхідні дані студент задає самостійно з урахуванням особливостей індивідуального завдання.

2. Забезпечити візуалізацію роботи моделі у часі з наглядною демонстрацією результатів.

3. Проаналізувати та пояснити отримані результати. За результатами роботи надати висновки щодо використаних засобів синхронізації.

**Варіант 2:**

**Автомат для розміну грошей (монет)**

Автомат приймає монету та ідентифікує її (визначає номінал). Ідентифікація моделюється датчиком випадкових чисел. Приймаються монети номіналом 1, 2, 5, 10, 25, 50 коп. і 1 грн. Розмін монети здійснюється на монети номіналу, що вводиться з клавіатури.

Кількість монет різного номіналу (1, 2, 5, 10, 25, 50 коп.), що містить автомат для розміну та видачі здачі, наперед задається. Якщо розмін и видача монет можливі, вони виконуються. Інакше здійснюється відмова від виконання операції розміну і видається повідомлення про її причину.

Монети, що приймаються для розміну, додаються до наявних в автоматі, та, у свою чергу, можуть використовуватися для розміну.

Модель роботи автомату представити у вигляді двох взаємодіючих процесів  **А** і **В**: процес **А** визначає момент надходження монети та ідентифікує її, а процес **В** здійснює розмін і видає гроші або відмову від виконання операції розміну.

**Для організації доступу до подільних ресурсів використати *алгоритм Деккера***

**Текст програми**

**Coin.py**

class Coin:

def \_\_init\_\_(self, nominal):

self.nominal = nominal;

self.excange = [];

**Changer.py**

import threading

import random as rnd

import time

import coin as c

global NOMINALS

global BANK

global COIN

global FLAG\_1

global FLAG\_2

global TURN

def process\_1():

global FLAG\_1

global FLAG\_2

global TURN

global COIN

FLAG\_1 = True

while FLAG\_2 == True:

if TURN == 1 :

FLAG\_1 = False

while TURN == 1:

pass

FLAG\_1 = True

COIN = c.Coin(rnd.choice(NOMINALS\_FULL))

print("Coin nominal: ", COIN.nominal)

TURN = 1

FLAG\_1 = False

def process\_2():

global COIN

global FLAG\_2

global FLAG\_1

global TURN

global BANK

FLAG\_2 = True

while FLAG\_1 == True:

if TURN == 0:

FLAG\_2 = False

while TURN == 0:

pass

FLAG\_2 = True

print("Please, enter nominal for coin exchange: ")

while True:

entered\_nominal = int(input())

if (not entered\_nominal in NOMINALS or entered\_nominal > COIN.nominal):

print("Wrong nominal!!! Enter another: ")

elif (not entered\_nominal in BANK):

print("No such nominal in BANK! Enter another: ")

else:

break

summa = 0

bank\_copy = BANK

exchange\_list = []

exchange = True

bank\_summa = 0

for element in bank\_copy:

bank\_summa += element

if bank\_summa < COIN.nominal:

exchange = False

else:

while True:

if (summa < COIN.nominal):

if (entered\_nominal in bank\_copy and summa+entered\_nominal<=COIN.nominal):

exchange\_list.append(entered\_nominal)

bank\_copy.remove(entered\_nominal)

summa += entered\_nominal

else:

bank\_copy.reverse()

for element in bank\_copy:

if (summa+element<=COIN.nominal):

bank\_copy.remove(element)

summa += element

exchange\_list.append(element)

break

else:

break

if (summa < COIN.nominal):

exchange = False

while (entered\_nominal in exchange\_list):

exchange\_list.remove(element)

bank\_copy.append(element)

summa -= element

bank\_copy.sort()

bank\_copy.reverse()

break

for element in bank\_copy:

if (summa + element <= COIN.nominal):

summa += element

exchange\_list.append(element)

bank\_copy.remove(element)

if (summa == COIN.nominal):

exchange = True

break

if (exchange == True):

COIN.exchange = exchange\_list

BANK = bank\_copy

print("Exchanged list: ")

print(exchange\_list)

BANK.append(COIN.nominal)

else:

print("Cant exchange this coin!")

TURN = 0;

FLAG\_2 = False

if \_\_name\_\_ == '\_\_main\_\_':

NOMINALS\_FULL = [2, 5, 10, 25, 50, 100]

NOMINALS = [1, 2, 5, 10, 25, 50]

BANK = [2,2,2,2,5,5,5,10,10,10,25]

FLAG\_1 = False

FLAG\_2 = False

TURN = 0

COIN = 0

A = threading.Thread(target=process\_1)

B = threading.Thread(target=process\_2)

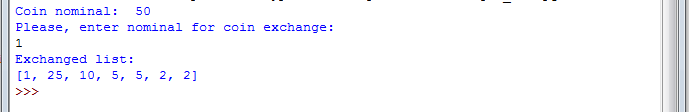
A.start()

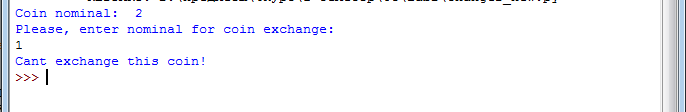
B.start()

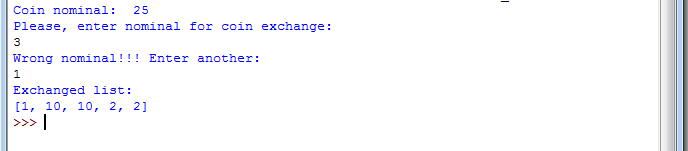
A.join()

B.join()

**Тести**







**Висновок**:

Перевагою цього алгоритму є те, що він не вимагає особливої команди Test-and-set (атомарного читання-і-зміни) і, таким чином, легко переноситься між різними мовами і машинними архітектурами. Недоліком є те, що в оригіналі він призначений тільки для двох процесів, а також використовує стан очікування(циклу) замість призупинення процесу. (Використання стану очікування означає, що процес має знаходитися якнайменше часу всередині критичної секції.)