**ЛАБОРАТОРНА РОБОТА #2**

З дисципліни «Операційні системи»

На тему «Синхронізація процесів»

|  |  |
| --- | --- |
| **Виконав**  Студент групи КВ-21  Комарницький О.Б. | **Перевірила**  Ст. викл. кафедри СПіСКС  Дробязко І.П. |

|  |  |  |
| --- | --- | --- |
| **Оцінка** | **Дата** | **Підпис** |
|  |  |  |

**Варіант #8**

**Завдання:**

Банкомат принимает заявку на выплату денег на сумму **X**, сверяет ее с количеством денег на счету клиента **Y** и, если **X<Y** , осуществляется попытка выплаты купюрами в 1,2,5,10,20,50,100 грн.

Величины **X** и **Y**- случайные числа. Первоначальное количество купюр каждого номинала, которые находятся в банкомате, задано.

Если сумма **X** подлежит выдаче, то программа определяет, можно ли затребованную сумму выплатить наличными в данный момент купюрами. Если это возможно, программа уменьшает количество соответствующих купюр в банкомате на число выданных и выдает деньги. Если наличность такова, что выдать затребованную сумму невозможно, банкомат сообщает об этом клиенту и отказывает в обслуживании.

Требования на выдачу денег поступают к банкомату после очередного обслуживания клиента (или отказа).

Модель работы банкомата представить в виде двух взаимодействующих процессов **A** и **B**.

Процесс **A** определяет момент поступления требования на оплату, возможность выплаты затребованной суммы, фиксирует возможность выплаты.

Процесс **B** ожидает момент появления необходимости сделать выплату. Если такая необходимость наступила, он расчитывает количество требуемых купюр каждого достоинства для выплаты и, соответственно, уменьшает наличность в банкомате или определяет невозможность выплаты, о чем печатает сообщение.

**Разработать программу работы автомата, используя почтовые ящики для организации доступа к разделяемым ресурсам.**

**Код програми мовою C++**

***Main.cpp***

#include "Functions.h"

#include <thread>

#include <Windows.h>

using namespace std;

int main()

{

thread p2(process2);

map<char \*, int> request;

while (true)

{

request["account"] = MIN\_ACCOUNT + rand() % (MAX\_ACCOUNT - MIN\_ACCOUNT) + 1;

request["request"] = MIN\_REQUEST + rand() % (MAX\_REQUEST - MIN\_REQUEST) + 1;

process1(request);

output();

//Sleep(300);

}

p2.join();

return 0;

}

***Functions.h***

#include <list>

#include <map>

const unsigned int MAX\_ACCOUNT = 10000;

const unsigned int MIN\_ACCOUNT = 1000;

const unsigned int MAX\_REQUEST = 11000;

const unsigned int MIN\_REQUEST = 10;

struct Mailbox

{

const unsigned int N\_MAX = 100;

std::list<int> buffer;

};

void process1(std::map <char \*, unsigned int> & request);

void process2();

void output();

***Functions.cpp***

#include "Functions.h"

#include <iostream>

#include <Windows.h>

using namespace std;

const unsigned int CURR[7] = { 1, 2, 5, 10, 20, 50, 100 }; // Possible currency in ua

unsigned int cur\_number[7] = { 100, 100, 100, 100, 100, 100, 10000 }; // Different currency money

Mailbox mailbox; // Mailbox to synchonize

void process1(map <char \*, unsigned int> & request)

{

cout << "Process1: request = " << request["request"] << endl;

if (request["request"] < request["account"])

cout << "Process1: Denied. Not enough money on account." << endl;

else

{

if (mailbox.buffer.size() >= mailbox.N\_MAX)

cout << "Process1: Buffer is already full." << endl;

else

{

mailbox.buffer.push\_back(request["request"]);

cout << "Process1: Request added to buffer." << endl;

}

}

}

void process2()

{

while (true)

{

if (mailbox.buffer.size())

{

unsigned int money = mailbox.buffer.front(), money\_counted = 0;// Money to give

int tmp[7] = { 0, 0, 0, 0, 0, 0, 0 }; // Array of currency to give

for (int i = 6; i >= 0 && money - money\_counted; --i) // Find currency

{

unsigned int n = (money - money\_counted) / CURR[i];

tmp[i] += min(n, cur\_number[i]);

money\_counted += min(n, cur\_number[i]) \* CURR[i];

}

if (money == money\_counted)

{

for (int i = 0; i < 7; i++)

cur\_number[i] -= tmp[i];

cout << "Process2: succeed outcome " << money << endl;

}

else

cout << "Process2: no currency" << endl;

mailbox.buffer.pop\_front();

}

//Sleep(500);

}

}

void output()

{

cout << "Buffer: " << mailbox.buffer.size() << endl;

for (int i = 6; i >= 0; --i)

cout << '\t' << cur\_number[i] << " x " << CURR[i] << " ua" << endl << endl;

}

**Скриншот програми**

