

Міністерство освіти і науки України

Національний технічний університет України

„КПІ імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки Кафедра інформатики і програмної інженерії

**Звіт**

Комп’ютерний практикум №3  
«побудова імітаційної моделі системи з використанням  
формалізму моделі масового обслуговування»

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*Горобець Олександр ІП-96*

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2022

Зміст

[1 Завдання 3](#_Toc121072720)

[2 Виконання 6](#_Toc121072721)

[2.1 Програмна реалізація алгоритму 6](#_Toc121072722)

[2.2 Хід роботи 7](#_Toc121072723)

[2.2.1 **Загальні зміни алгоритму** 7](#_Toc121072724)

[2.2.2 **Задача 1 – автомобільний банк** 8](#_Toc121072725)

[2.2.3 **Задача 2 – лікарня** 12](#_Toc121072726)

[Висновок 18](#_Toc121072727)

# Завдання

1. Реалізувати універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю (30 балів).

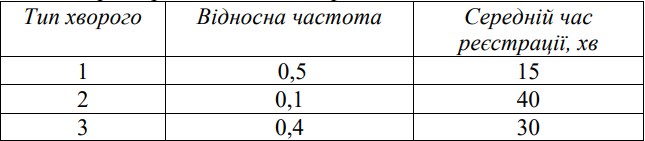
2. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму (30 балів):

У банку для автомобілістів є два віконця, кожне з яких обслуговується одним касиром і має окрему під'їзну смугу. Обидві смуги розташовані поруч. З попередніх спостережень відомо, що інтервали часу між прибуттям клієнтів у годину пік розподілені експоненційно з математичним очікуванням, рівним 0,5 од. часу. Через те, що банк буває переобтяжений тільки в годину пік, то аналізується тільки цей період. Тривалість обслуговування в обох касирів однакова і розподілена експоненційно з математичним очікуванням, рівним 0,3 од. часу. Відомо також, що при рівній довжині черг, а також при відсутності черг, клієнти віддають перевагу першій смузі. В усіх інших випадках клієнти вибирають більш коротку чергу. Після того, як клієнт в'їхав у банк, він не може залишити його, доки не буде обслугований. Проте він може перемінити чергу, якщо стоїть останнім і різниця в довжині черг при цьому складає не менше двох автомобілів. Через обмежене місце на кожній смузі може знаходитися не більш трьох автомобілів. У банку, таким чином, не може знаходитися більш восьми автомобілів, включаючи автомобілі двох клієнтів, що обслуговуються в поточний момент касиром. Якщо місце перед банком заповнено до границі, то клієнт, що прибув, вважається втраченим, тому що він відразу ж виїжджає. Початкові умови такі: 1) обидва касири зайняті, тривалість обслуговування для кожного касира нормально розподілена з математичним очікуванням, рівним 1 од. часу, і середньоквадратичним відхиленням, рівним 0,3 од. часу; 2) прибуття першого клієнта заплановано на момент часу 0,1 од. часу; 3) у кожній черзі очікують по два автомобіля.

Визначити такі величини: 1) середнє завантаження кожного касира; 2) середнє число клієнтів у банку; 3) середній інтервал часу між від'їздами клієнтів від вікон; 4) середній час перебування клієнта в банку; 5) середнє число клієнтів у кожній черзі; 6) відсоток клієнтів, яким відмовлено в обслуговуванні; 7) число змін під'їзних смуг.

3. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму (40 балів):

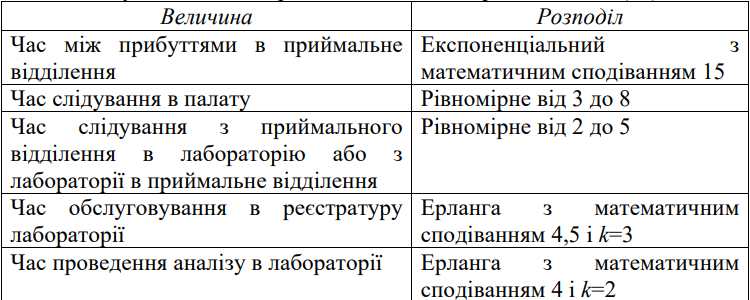
У лікарню поступають хворі таких трьох типів: 1) хворі, що пройшли попереднє обстеження і направлені на лікування; 2) хворі, що бажають потрапити в лікарню, але не пройшли повністю попереднє обстеження; 3) хворі, які тільки що поступили на попереднє обстеження. Чисельні характеристики типів хворих наведені в таблиці:



Тип хворого Відносна частота Середній час реєстрації, хв 1 0,5 15 2 0,1 40 3 0,4 30 При надходженні в приймальне відділення хворий стає в чергу, якщо обидва чергових лікарі зайняті. Лікар, який звільнився, вибирає в першу чергу тих хворих, що вже пройшли попереднє обстеження. Після заповнення різноманітних форм у приймальне відділення хворі 1 типу ідуть прямо в палату, а хворі типів 2 і 3 направляються в лабораторію. Троє супровідних розводять хворих по палатах. Хворим не дозволяється направлятися в палату без супровідного. Якщо всі супровідні зайняті, хворі очікують їхнього звільнення в приймальному відділенні. Як тільки хворий доставлений у палату, він вважається таким, що завершив процес прийому до лікарні.

Хворі, що спрямовуються в лабораторію, не потребують супроводу. Після прибуття в лабораторію хворі стають у чергу в реєстратуру. Після реєстрації вони ідуть у кімнату очікування, де чекають виклику до одного з двох лаборантів. Після здачі аналізів хворі або повертаються в приймальне відділення (якщо їх приймають у лікарню), або залишають лікарню (якщо їм було призначено тільки попереднє обстеження). Після повернення в приймальне відділення хворий, що здав аналізи, розглядається як хворий типу 1.

У наступній таблиці приводяться дані по тривалості дій (хв):



Величина Розподіл Час між прибуттями в приймальне відділення Експоненціальний з математичним сподіванням 15 Час слідування в палату Рівномірне від 3 до 8 Час слідування з приймального відділення в лабораторію або з лабораторії в приймальне відділення Рівномірне від 2 до 5 Час обслуговування в реєстратуру лабораторії Ерланга з математичним сподіванням 4,5 і k=3 Час проведення аналізу в лабораторії Ерланга з математичним сподіванням 4 і k=2 Визначити час, проведений хворим у системі, тобто інтервал часу, починаючи з надходження і закінчуючи доставкою в палату (для хворих типу 1 і 2) або виходом із лабораторії (для хворих типу 3). Визначити також інтервал між прибуттями хворих у лабораторію.

# Виконання

## Програмна реалізація алгоритму

Через те, що програмна реалізація даного алгоритму виявилась досить великою (19 файлів та майже 2 тис. рядків коду), то код розміщено на платформі GitHub.

Посилання на вихідний код програми: <https://github.com/ztAlexGor/SystemModeling>

## Хід роботи

### **Загальні зміни алгоритму**

При виконання даної ЛР за основу було взято алгоритм розробленій у попередній лабораторній роботі (ЛР №2). Далі наведено список змін та оновлень.

* Rand Generator:
  + Додано генератор випадкових чисел відповідно до закону розподілення Ерланга.
  + Додано генератор, який може бути налаштований користувачем і генеруватиме числа відповідно до конкретної задачі.
* Задачі:
  + Тепер усі елементи працюють із класом Task, який зберігає інформацію про задачу, що оброблюється.
* Element:
  + Додано подію перевірки змін. Вона корисна, наприклад, для реалізації події зміни черги.
  + Можливість умовного переходу до наступного елементу
* Creator Element:
  + Можливість задати відносну частоту генерації задач різних видів.
* Process Element:
  + Можливість задати приорітет оброблюваних задач.
  + Можливість вносити зміни до черги (наприклад задача переходить до іншої черги).
  + Додано можливість задати початковий набір даних для моделювання.
* Додано новий елемент «Transition Element». Потрібен для імітації часової затримки при переході.
* Додано новий елемент «Work Group Element». За допомогою нього зручно моделювати ситуації коли декілька процесів працюють із спільною чергою.
* Статистика
  + Майже уся інформація що не стосується безпосередньо моделювання винесена до окремого класу Info, що займається її збором, обробкою та презентацією.
  + Повністю перероблено вивід інформації та загальну статистику.

Окрім перелічених основних змін алгоритм зазнав ще дуже багато менших модифікацій та покращень.

### **Задача 1 – автомобільний банк**

У другому завданні лабораторної роботи потрібно реалізувати програму, яка буде моделювати роботу автомобільного банку. Далі наведено схему поставленої задачі (рис 2.1).

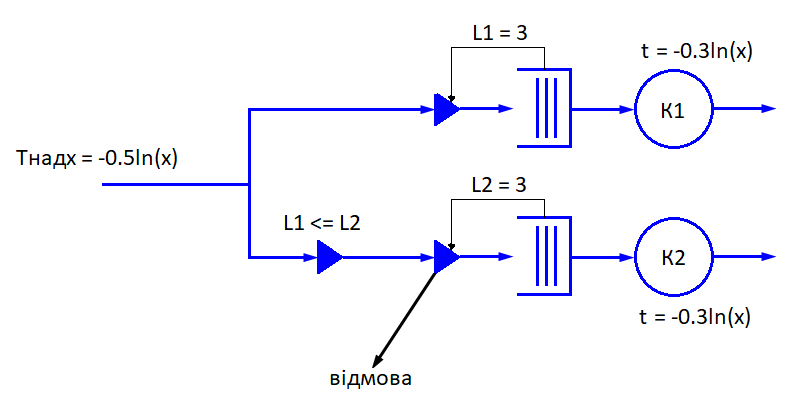


Рис 2.1 – схематичнее зображення задачі «автомобільний банк».

Далі наведено лістинг функції у якій створюється модель задачі та розпочинається її моделювання.

void bank\_second\_way() {

//create scheme

Create\* c = new Create("CREATOR", new Exponential(0.5));

Process\* p1 = new Process("CASHIER\_1", new Gaussian(1, 0.3), 3);

Process\* p2 = new Process("CASHIER\_2", new Gaussian(1, 0.3), 3);

WorkGroup\* wg = new WorkGroup(std::vector<Process\*>{p1, p2}, 0);

wg->setName("CASSIERS");

wg->setChangeQueueFunction([](std::vector<Process\*> processes, std::vector<Task\*> queue)->int {

for (int p = 0; p < processes.size(); p++) {

if (processes[p]->getQueueSize() + 1 < queue.size())

return int(queue.size()) - 1;

else return -1;

}

});

//init model

std::vector<Task\*>q1;

q1.push\_back(new Task());

q1.push\_back(new Task());

p1->init(new Task(), q1);

std::vector<Task\*>q2;

q2.push\_back(new Task());

q2.push\_back(new Task());

p2->init(new Task(), q2);

c->setTnext(0.1);

//create model

c->addNextElement(wg);

std::vector<Element\*>list;

list.push\_back(c);

list.push\_back(wg);

Model model(list);

model.simulate(500.0);

}

Як видно із наведеного фрагменту коду – моделювання тривало 500 одиниць часу. Процес роботи програми можна побачити на рис 2.2. А на рис 2.3 – зображено результат моделювання із наведеною статистикою як по кожному окремому елементу, так і загальний результат.

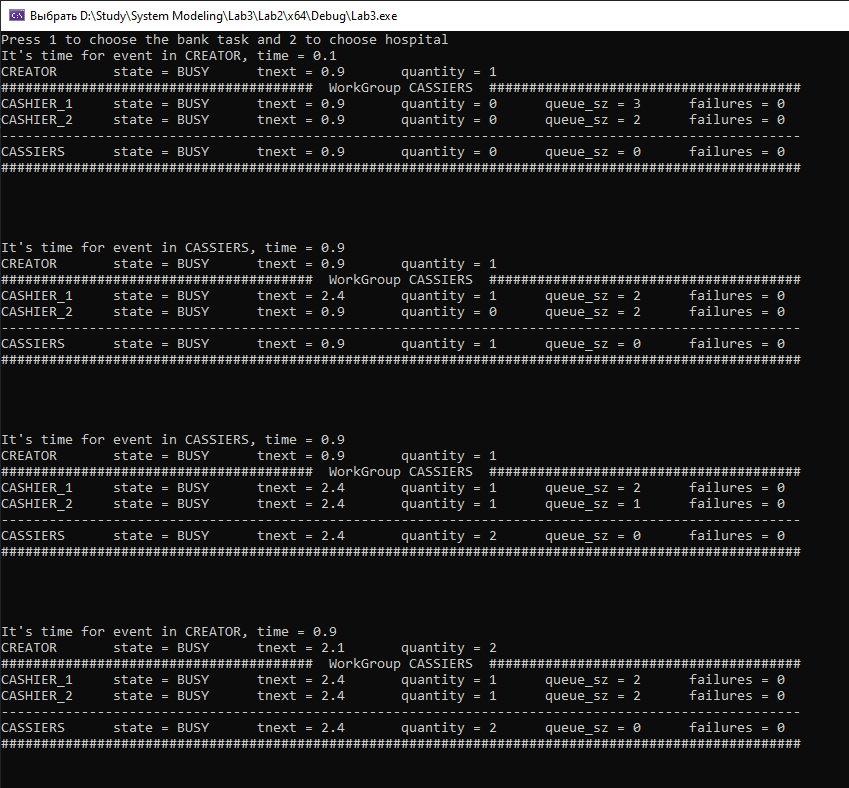


Рис 2.2 – фрагмент виводу програми на початку моделювання

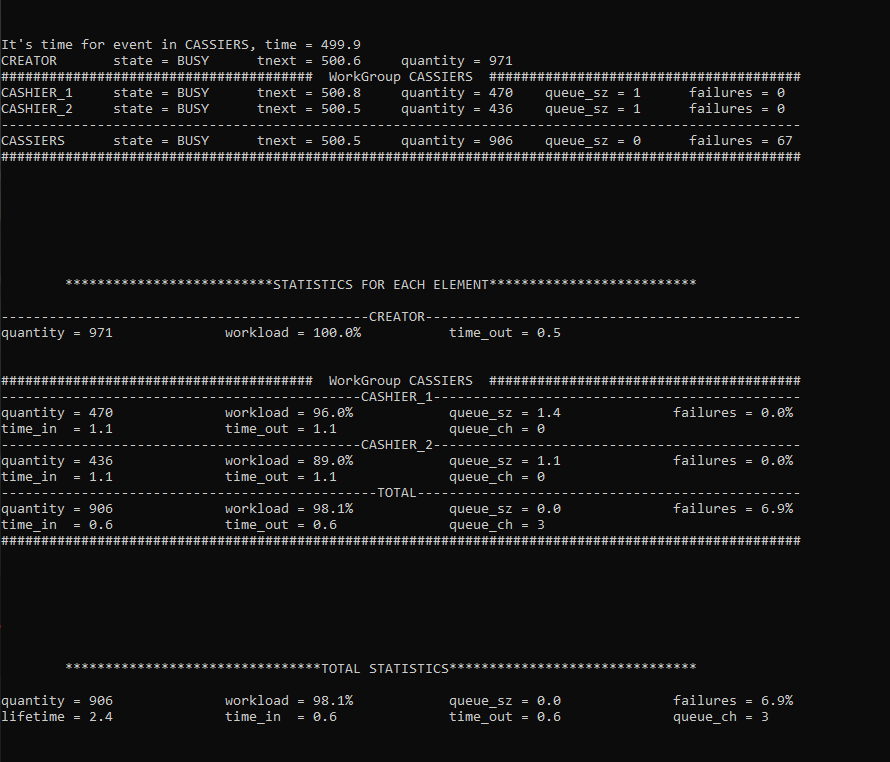


Рис 2.3 – результат моделювання СМО «автомобільний банк»

Перевіримо правильність моделювання порахувавши кількість вхідних та вихідних задач.

Вхідні:

* За умовою, на час початку моделювання касири були зайняті, а у їх чергах знаходилось по 2 автомобілі. Тобто загалом – 6 автомобілів.
* Під час моделювання у Creator було створено ще 971 подія надходження автомобілю.
* Відмов за час моделювання – 67.

Вихідні:

* Загальна кількість оброблених задач – 906
* На момент завершення моделювання у банку знаходилось 4 автомобіля

Оскільки 6 + 971 – 67 = 906 + 4, то можна стверджувати, що моделювання пройшло правильно.

**Отримані результати:**

1. Середнє завантаження: 1 касир – 96 %, 2 касир – 89%;
2. Середня кількість клієнтів у банку: 4.4 клієнти на одиницю часу;
3. Середній інтервал від’їзду клієнтів після обслуговування: 0.6 часу;
4. Середній час перебування клієнту у банку: 2.4 часу;
5. Середнє число клієнтів у чергах: 1 черга – 1.4, 2 черга – 1.1
6. Відсоток вибувших клієнтів: 6.9%
7. Число змін черги клієнтами: 3

### **Задача 2 – лікарня**

У другому завданні лабораторної роботи потрібно реалізувати програму, яка буде моделювати роботу лікарні. Далі наведено схему поставленої задачі (рис 2.4).

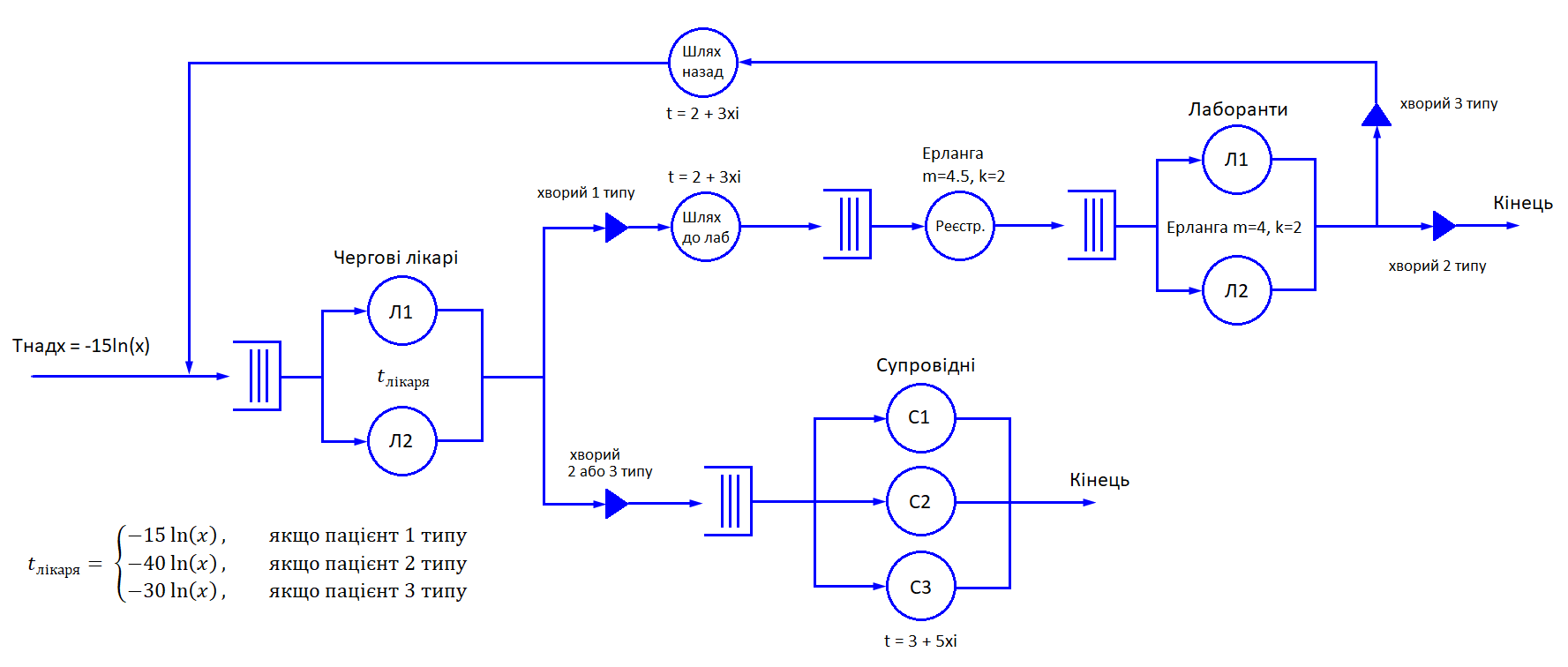


Рис 2.4 – схематичнее зображення задачі «лікарня».

Далі наведено лістинг функції у якій створюється модель задачі та розпочинається її моделювання.

void hospital() {

//patient generator

std::vector<std::pair<Task\*, double>> distribution = {

std::pair<Task\*, double>(new Task("1"), 0.5), std::pair<Task\*, double>(new Task("2"), 0.1), std::pair<Task\*, double>(new Task("3"), 0.4) };

Create\* creator = new Create("CREATOR", new Exponential(15), distribution);

//admission department

CustomGenerator\* delayGenerator = new CustomGenerator([](Task\* task)->double {

if (task->getType() == "1")return Exponential{ 15.0 }.generate();

if (task->getType() == "2")return Exponential{ 40.0 }.generate();

if (task->getType() == "3")return Exponential{ 30.0 }.generate();

throw "Unknown task type";

});

Process\* doctor1 = new Process("DOCTOR1", delayGenerator, 0);

Process\* doctor2 = new Process("DOCTOR2", delayGenerator, 0);

Element\* (\*transitionConditionForDoctors) (std::vector<Element\*>, Task\*) = [](std::vector<Element\*>list, Task\* task) {

if (task->getType() == "1") {

for (auto e : list) {

if (e->getName() == "GUIDES")return e;

}

}

else {

for (auto e : list) {

if (e->getName() == "WAY\_TO\_LAB")return e;

}

}

};

doctor1->setTransitionCondition(transitionConditionForDoctors);

doctor2->setTransitionCondition(transitionConditionForDoctors);

std::vector<Process\*> doctorsList;

doctorsList.push\_back(doctor1);

doctorsList.push\_back(doctor2);

WorkGroup\* doctors = new WorkGroup(doctorsList, INT\_MAX);

doctors->setName("DOCTORS");

doctors->setSelectTaskFunction([](std::vector<Task\*> queue)->int {

for (int i = 0; i < queue.size(); i++) {

if (queue[i]->getType() == "1")return i;

}

return 0;

});

//guides for patient 1

Process\* guide1 = new Process("GUIDE1", new Uniform(3, 8), 0);

Process\* guide2 = new Process("GUIDE2", new Uniform(3, 8), 0);

Process\* guide3 = new Process("GUIDE3", new Uniform(3, 8), 0);

std::vector<Process\*> guidesList;

guidesList.push\_back(guide1);

guidesList.push\_back(guide2);

guidesList.push\_back(guide3);

WorkGroup\* guides = new WorkGroup(guidesList);

guides->setName("GUIDES");

//laboratory way

Transition\* labWay = new Transition("WAY\_TO\_LAB", new Uniform(2, 5));

Transition\* admissionWay = new Transition("WAY\_TO\_ADMISSION", new Uniform(2, 5));

//laboratory

Process\* labRegistry = new Process("REGISTRY", new Erlang(4.5, 3));

Process\* laborant1 = new Process("LABORANT1", new Erlang(4, 2), 0);

Process\* laborant2 = new Process("LABORANT2", new Erlang(4, 2), 0);

Element\* (\*transitionConditionForLaborants) (std::vector<Element\*>, Task\*) = [](std::vector<Element\*>list, Task\* task)->Element\* {

if (task->getType() == "2") {

task->setType("1");

return list[0];

}

else {

return nullptr;

}

};

laborant1->setTransitionCondition(transitionConditionForLaborants);

laborant2->setTransitionCondition(transitionConditionForLaborants);

std::vector<Process\*> laborantList;

laborantList.push\_back(laborant1);

laborantList.push\_back(laborant2);

WorkGroup\* laborants = new WorkGroup(laborantList);

laborants->setName("LABORANTS");

//create a scheme

creator->addNextElement(doctors);

doctors->addNextElement(guides);

doctors->addNextElement(labWay);

labWay->addNextElement(labRegistry);

labRegistry->addNextElement(laborants);

laborants->addNextElement(admissionWay);

admissionWay->addNextElement(doctors);

std::vector<Element\*>list;

list.push\_back(creator);

list.push\_back(doctors);

list.push\_back(guides);

list.push\_back(labWay);

list.push\_back(labRegistry);

list.push\_back(laborants);

list.push\_back(admissionWay);

Model model(list);

model.simulate(1000.0);

}

Як видно із наведеного фрагменту коду – моделювання тривало 1000 хвилин. Процес роботи програми можна побачити на рис 2.5. А на рис 2.6 – зображено результат моделювання із наведеною статистикою як по кожному окремому елементу, так і загальний результат.

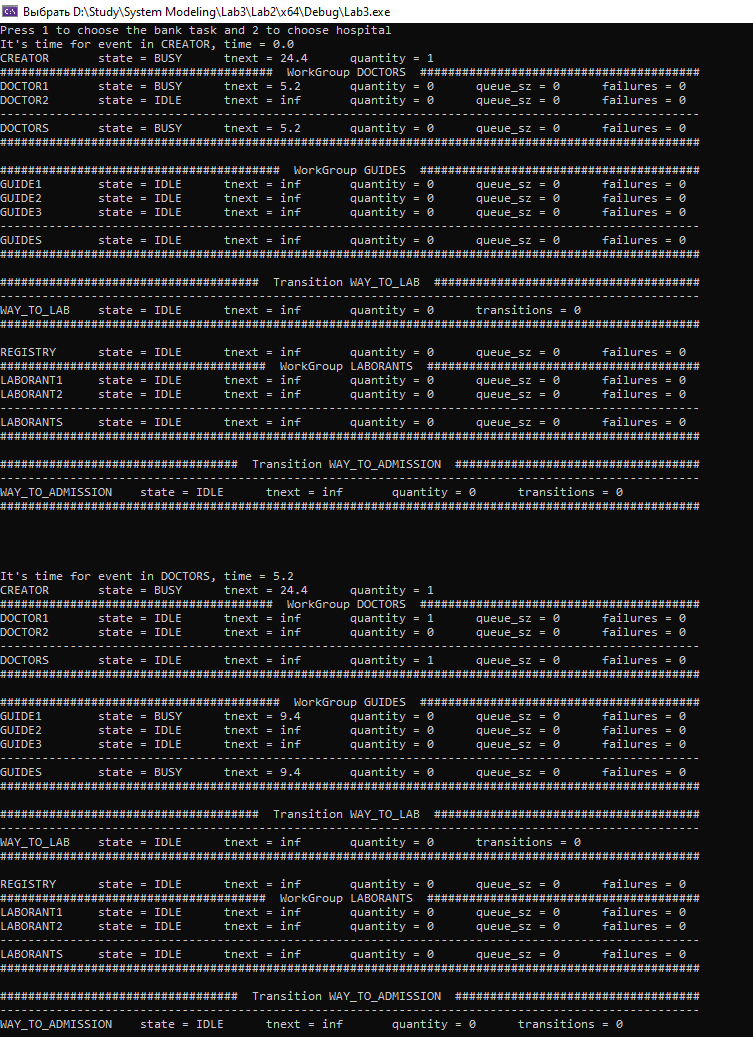


Рис 2.5 – фрагмент виводу програми на початку моделювання

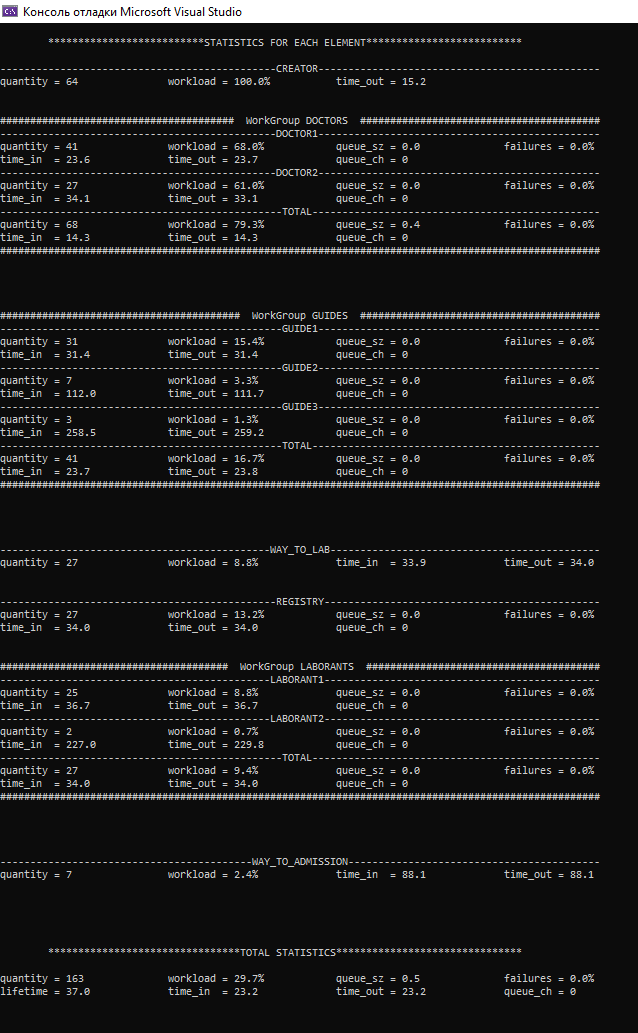


Рис 2.6 – результат моделювання СМО «лікарня»

**Перевірка правильності результатів**

За допомогою елементу CREATOR було згенеровано 64 пацієнти.

У лабораторію було направлено 27 хворих, із них 7 повернулись назад вже як хворі 1го типу.

У приймальному відділенні чергові лікарі 68 раз зареєстрували хворих (ще 3 не встигли закінчити до завершення моделювання)

Ці числа досить сильно співпадають із заданим розподілом хворих:

1 типу – 34, це становить 53%

2 типу – 7, це становить 11%

3 типу – 23, це становить 36%

І в результаті, супроводжуючими було розведено по палатах 41 хворий, що в точності дорівнює кількості хворих 1 та 2 типів. Отже побудована модель – вірна!

**Отримані результати:**

Середній час проведений хворим у системі становить 37 хвилин;

Середній інтервал між прибуттям хворих у лабораторію дорівнює 34 хв.

Висновок

В рамках даної лабораторної роботи згідно із завданням було вдосконалено алгоритм побудови імітаційних моделей дискретно-подійних систем. Процес модифікації торкнувся абсолютно кожного структурного елементу, а також було розроблено два нових: Transition Element та WorkGroup Element. Також було повністю змінено принцип переходу між елементами та створено окрему структуру для збору статистичних даних.

Новий алгоритм дозволяє моделювати майже будь-які види СМО завдяки високому рівню абстракції та модульності елементів. Також користувач в процесі моделювання та по його завершенню має доступ до усієї статистичної інформації, яку можна легко вивести на екран за допомогою розробленого зручного інтерфейсу.

Алгоритм було використано для виконання поставлених у лабораторній задач по моделюванню автомобільного банку та лікарні. Після створення робочої схеми її було перевірено під час моделювання. Особливо детально перевірялись специфічні моменти кожної із задач. Наприклад, зміна черги автомобілем, умовні переходи, відмови, часові затримки, використання декількома процесами спільної черги та інше.

Отже, за допомогою створеного алгоритму вдалося побудувати модель заданої предметної області та експериментально дослідити бажані параметри.