**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №2

з дисципліни «Моделювання та Аналіз Програмного Забезпечення» на тему:

«Моделювання систем масового обслуговування з одним обслуговуючим приладом та чергою»

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Викладач: Ж. К. Камінська

2024

МОДЕЛЮВАННЯ СИСТЕМ МАСОВОГО ОБСЛУГОВУВАННЯ З ОДНИМ ОБСЛУГОВУЮЧИМ ПРИЛАДОМ ТА ЧЕРГОЮ

Мета роботи

Вивчити методи моделювання різних дисциплін обслуговування в СМО з одним приладом та чергою, аналізувати вихідні дані моделювання з метою вибору оптимального варіанту реалізації системи, що моделюється.

Індивідуальне завдання

У механічний цех, відразу ж з початком робочого дня, кожні 3хв надходять червноі деталі з масою 2кг, кожні 5хв червоні деталі з масою 2,4кг і кожні 20хв зелені з масою 3кг та температурою 85С і двома отворами. Червоні деталі надходять для обробки на свердлильному верстаті. У кожній з них свердлять по одному отвору. Час обслуговування: червоні – 15хв. Зелені поступають на токарний верстат і обробляються 20хв. Час встановлення на верстат та зняття з нього становить 4хв для кожної деталі. Промоделювати роботу цеху протягом 8 годин робочого дня. Визначити чергу перед свердлильним верстатом.

Опис термінів

Прилад - динамічний об'єкт для моделювання апарату, що обслуговує транзакти

Черга - динамічний об'єкт для просування транзактів на ділянках моделі та статистики

Результати виконання

Код програми

*Загальне завдання*

#include "../simc/simc.h"

#include <iostream>

using namespace std;

int TOTAL\_HOURS = 28800;

int FIRST\_INTERVAL = 510;

int FIRST\_DURATION = 300;

int SECOND\_INTERVAL = 420;

int SECOND\_DURATION = 100;

pair<double, double> model(bool givePriorityToYellow = false)

{

  pqueue partsQueue;

  pfacility warehousePerson;

  int totalPeopleInQueue = 0;

  int totalIterations = 0;

  initlist(TOTAL\_HOURS);

  initcreate(1, 0);

  initcreate(8, 0);

  newqueue(partsQueue, "\"Orders Order\"");

  newfac(warehousePerson, "\"Warehouse\"");

  while (systime < TOTAL\_HOURS)

  {

    plan();

    switch (sysevent)

    {

    case 1:

      create(FIRST\_INTERVAL);

      break;

    case 2:

      inqueue(partsQueue);

      trans->prty = 1;

      totalPeopleInQueue += 1;

      break;

    case 3:

      seize(warehousePerson);

      break;

    case 4:

      outqueue(partsQueue);

      totalPeopleInQueue -= 1;

      break;

    case 5:

      delayt(FIRST\_DURATION);

      break;

    case 6:

      outfac(warehousePerson);

      break;

    case 7:

      destroy();

      break;

    case 8:

      create(SECOND\_INTERVAL);

      break;

    case 9:

      inqueue(partsQueue);

      if (givePriorityToYellow == true)

      {

        trans->prty = 2;

      }

      else

      {

        trans->prty = 1;

      }

      totalPeopleInQueue += 1;

      break;

    case 10:

      seize(warehousePerson);

      break;

    case 11:

      outqueue(partsQueue);

      totalPeopleInQueue -= 1;

      break;

    case 12:

      delayt(SECOND\_DURATION);

      break;

    case 13:

      outfac(warehousePerson);

      break;

    case 14:

      destroy();

      break;

    }

    totalIterations += 1;

  }

  printall();

  double averagePeopleInQueue = (double)totalPeopleInQueue / totalIterations;

  double totalMoneyLost = totalPeopleInQueue \* 0.25 \* FIRST\_DURATION + totalPeopleInQueue \* 0.25 \* SECOND\_DURATION;

  return make\_pair(averagePeopleInQueue, totalMoneyLost);

}

int main()

{

  pair<double, double> resOne = model(false);

  pair<double, double> resTwo = model(true);

  cout << endl

       << "Average people in queue without prioritization: " << resOne.first << endl;

  cout << "Average people in queue with prioritization: " << resTwo.first << endl;

  cout << "Total money lost without prioritization: " << resOne.second << endl;

  cout << "Total money lost with prioritization: " << resTwo.second << endl;

  return 0;

}

*Індивідуальне завдання*

#include "../simc/simc.h"

#include <iostream>

using namespace std;

auto STARTING\_TIME = 0;

auto TIME\_TO\_TAKE\_PARTS = 4;

auto TIME\_TO\_WORK = 480;

auto RED\_MASS = 2;

auto RED\_INTERVAL = 3;

auto RED\_HOLES = 0;

auto HEFTY\_RED\_MASS = 2.4;

auto HEFTY\_RED\_INTERVAL = 5;

auto HEFTY\_RED\_HOLES = 0;

auto GREEN\_MASS = 3;

auto GREEN\_INTERVAL = 20;

auto GREEN\_TEMPERATURE = 85;

auto GREEN\_HOLES = 2;

auto DRILLING\_MACHINE\_DURATION = 15;

auto LATHE\_DURATION = 20;

auto DRILLING\_MACHINE\_NAME = "\"Drilling Machine\"";

auto LATHE\_NAME = "\"Lathe\"";

auto TOTAL\_WORKING\_TIME = 0;

auto TOTAL\_DRILLING\_MACHINE\_TIME = 0;

auto TOTAL\_LATHE\_TIME = 0;

auto TOTAL\_HOLES = 0;

void model()

{

  pstorage drillingMachine;

  pstorage lathe;

  initlist(TIME\_TO\_WORK);

  // red drill

  initcreate(1, STARTING\_TIME);

  // hefty red drill

  initcreate(4, STARTING\_TIME);

  // green lathe

  initcreate(7, STARTING\_TIME);

  newstorage(drillingMachine, DRILLING\_MACHINE\_NAME, HEFTY\_RED\_INTERVAL);

  newstorage(lathe, LATHE\_NAME, GREEN\_INTERVAL);

  const int drillDelay = TIME\_TO\_TAKE\_PARTS + DRILLING\_MACHINE\_DURATION;

  const int latheDelay = TIME\_TO\_TAKE\_PARTS + LATHE\_DURATION;

  cout << "Drilling machine queue: " << endl;

  while (systime < TIME\_TO\_WORK)

  {

    plan();

    switch (sysevent)

    {

    case 1:

      enter(drillingMachine, 1);

      TOTAL\_HOLES += 1;

      cout << "Transact " << trans->nom << " in queue" << endl;

      break;

    case 2:

      delayt(drillDelay);

      TOTAL\_WORKING\_TIME += drillDelay;

      TOTAL\_DRILLING\_MACHINE\_TIME += drillDelay;

      break;

    case 3:

      leave(drillingMachine, 1);

      cout << "Transact " << trans->nom << " out of queue" << endl;

      break;

    case 4:

      enter(drillingMachine, 1);

      TOTAL\_HOLES += 1;

      cout << "Transact " << trans->nom << " in queue" << endl;

      break;

    case 5:

      delayt(drillDelay);

      TOTAL\_WORKING\_TIME += drillDelay;

      TOTAL\_DRILLING\_MACHINE\_TIME += drillDelay;

      break;

    case 6:

      leave(drillingMachine, 1);

      cout << "Transact " << trans->nom << " out of queue" << endl;

      break;

    case 7:

      enter(lathe, 1);

      cout << "Transact " << trans->nom << " in queue" << endl;

      break;

    case 8:

      delayt(latheDelay);

      TOTAL\_WORKING\_TIME += latheDelay;

      TOTAL\_LATHE\_TIME += latheDelay;

      break;

    case 9:

      leave(lathe, 1);

      cout << "Transact " << trans->nom << " out of queue" << endl;

      break;

    case 10:

      next(1);

      cout << endl;

      break;

    }

  }

  printall();

  clear();

  destrs(drillingMachine);

  cout << endl

       << "Modeling finished, praise Jesus Christ our Holy Lord Almighty Living GOD Most High" << endl

       << "  - Total time the machines worked: " << TOTAL\_WORKING\_TIME << " minutes" << endl

       << "  - Total time the drill worked: " << TOTAL\_DRILLING\_MACHINE\_TIME << " minutes" << endl

       << "  - Total time the lathe worked: " << TOTAL\_LATHE\_TIME << " minutes" << endl

       << "  - Total number of holes: " << TOTAL\_HOLES << endl;

}

int main()

{

  model();

  return 0;

}

// g++ ind.cpp ../simc/simc.cpp -o ind

Виконання програми

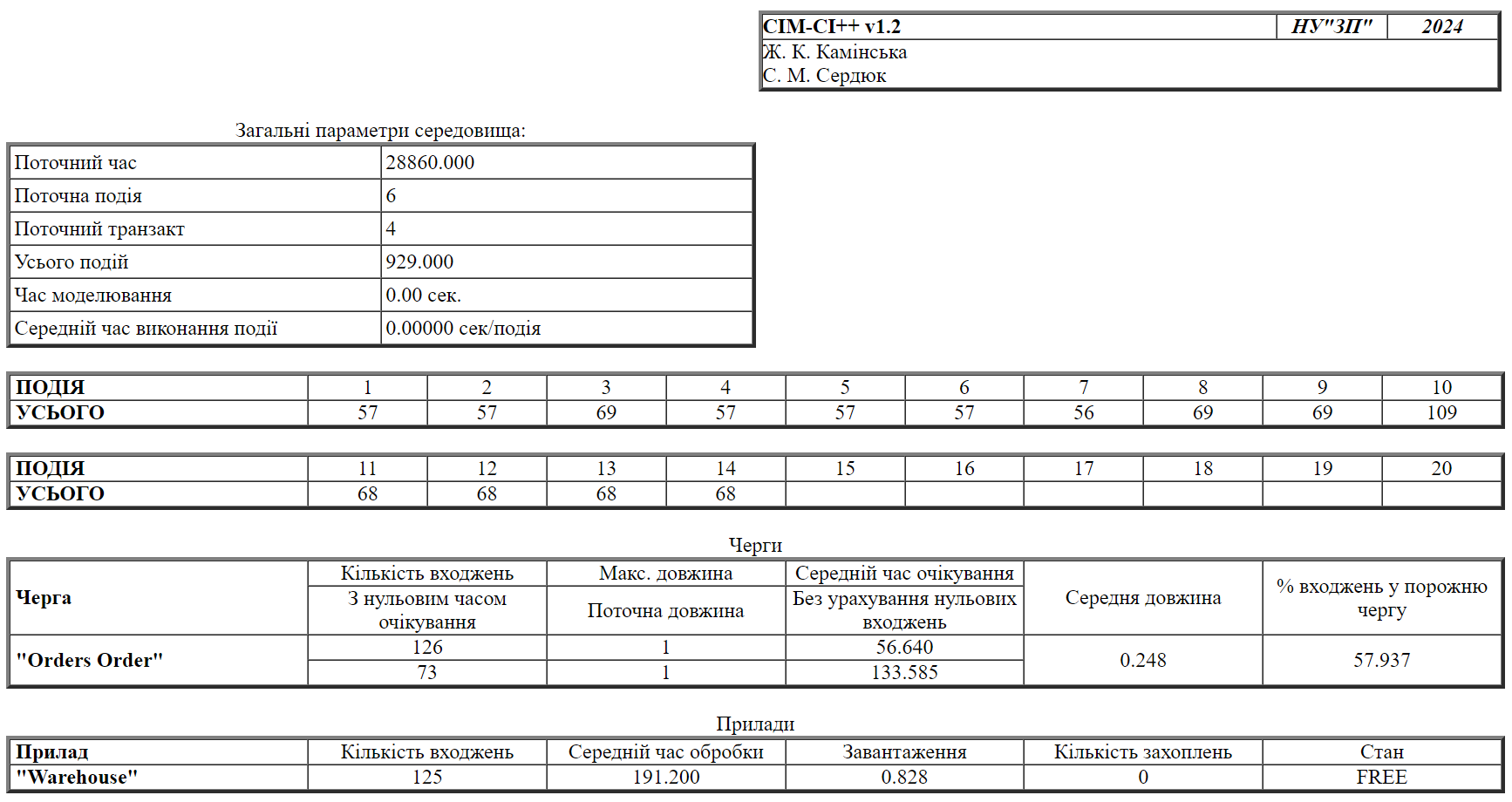


Рисунок 1.1 – Загальне завдання – браузер

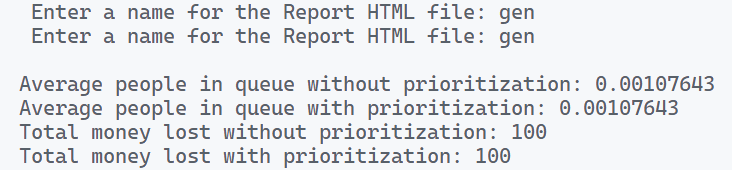


Рисунок 1.2 – Загальне завдання – консолька ♰

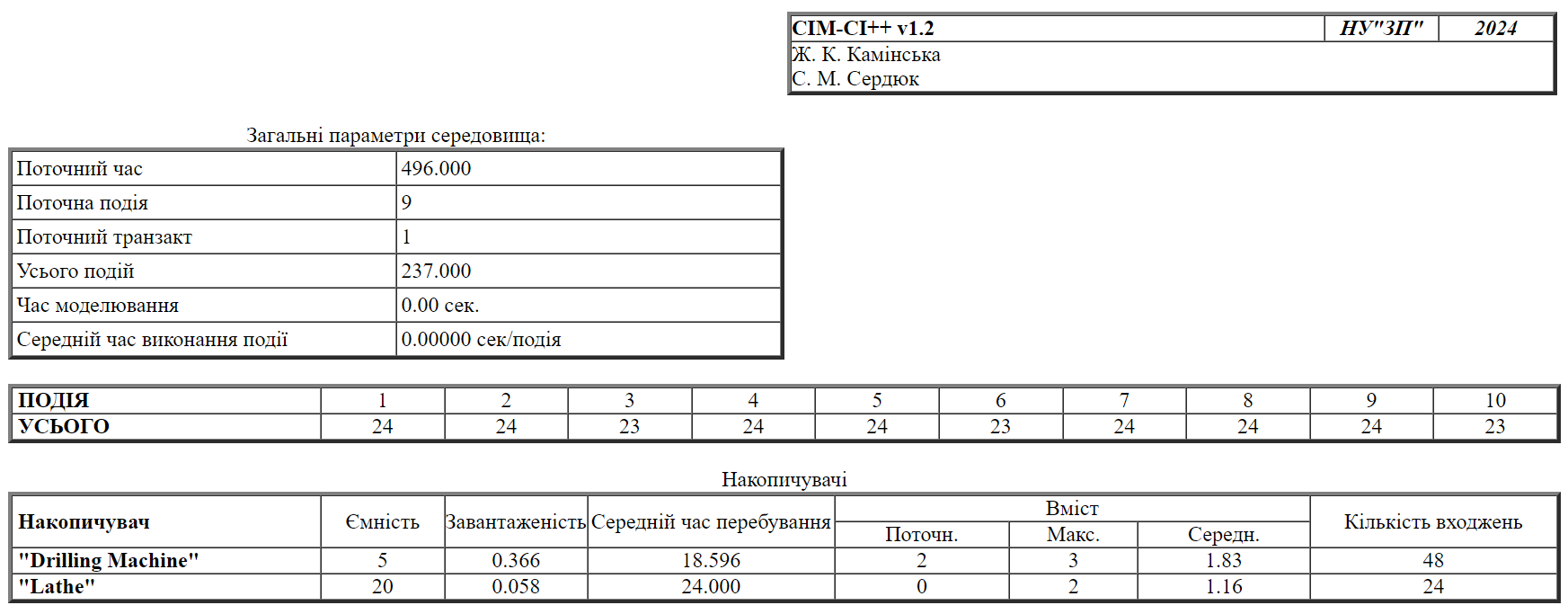


Рисунок 1.3 – Індивідуальне завдання – браузер

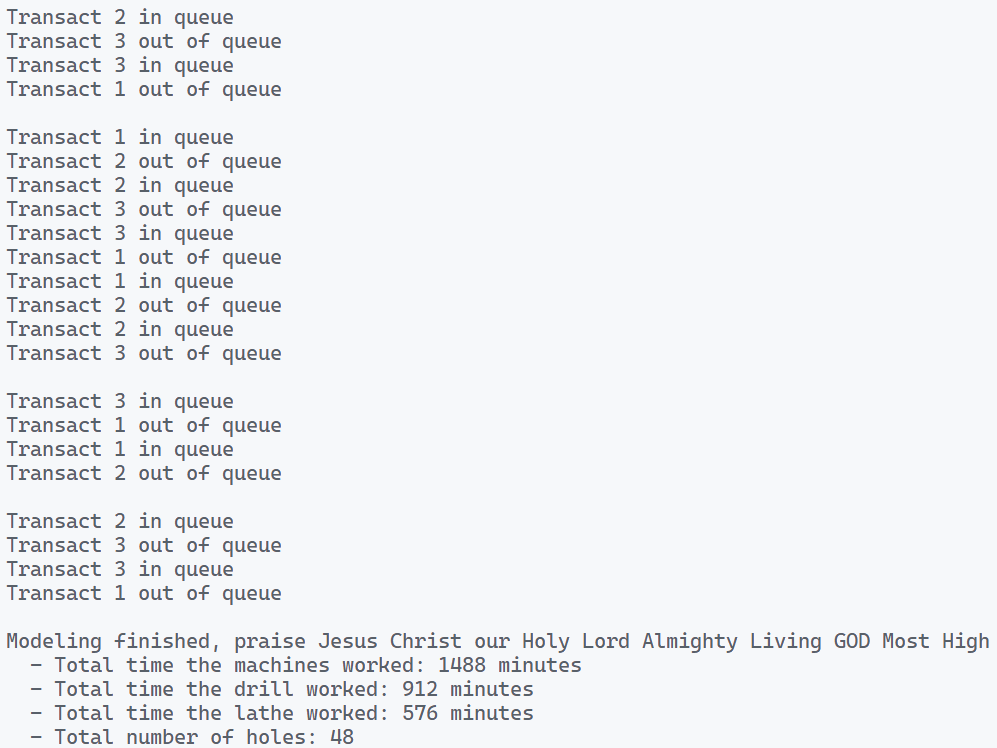


Рисунок 1.4 – Індивідуальне завдання – консолька

Висновки

Таким чином, ми вивчили методи моделювання різних дисциплін обслуговування в СМО з одним приладом та чергою; а також проаналізували вихідні дані моделювання з метою вибору оптимального варіанту реалізації системи, що моделюється.

Контрольні питання

Основні дисципліни обслуговування в СМО

Основними дисциплінами обслуговування в Системах Масового Обслуговування є First in First out (FIFO) та Last in First out (LIFO). Також існують інші нестандартні дисципліни.

Множинні типи даних "ПРИЛАД" та "ЧЕРГА" СІМ

Прилад -динамічний об'єкт для моделювання апарату, що обслуговує транзакти. Транзакт може очікувати входу до приладу, займати чи звільняти його.

Черга -динамічний об'єкт для просування транзактів на ділянках моделі та статистики.

Процедури створення, знищення, реєстрації черги

Для створення черги можна використати void newqueue(посилання на чергу, ймення черги); для реєстрації черги або void inqueue(pqueue), або void outqueue(pqueue); для знищення черги можна використати void destrq(pqueue).