**Міністерство освіти і науки України**

**Національний університет «Запорізька Політехніка»**

Кафедра програмних засобів

**ЗВІТ**

з лабораторної роботи №5

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

«Моделювання довільних дисциплін обслуговування з використанням лангюгів користувача; вивчення принципів побудови гістограм»

**Виконав:**

Студент групи КНТ-122 О. А. Онищенко

**Прийняли:**

Викладач: Ж. К. Камінська

2024

Моделювання довільних дисциплін обслуговування з використанням лангюгів користувача; вивчення принципів побудови гістограм

Мета роботи

Метою роботи є освоєння процедур будування вибором випадкових величин та графічного виводу інформації в системі імітаційного моделювання SIMC; вивчення принципів моделювання різних дисциплін, які обслуговуються на основі списків користувача.

Результати виконання

Код програми

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

#include <iostream>

using namespace std;

void one() {

  auto Total\_Modeling\_Time = 8\*60\*60;

  auto Last\_Time = NULL;

  auto Interval\_Min=300-20;

  auto Interval\_Max=300+20;

  auto First\_Delay\_Min=100-20;

  auto First\_Delay\_Max=100+20;

  auto Second\_Delay\_Min=110-25;

  auto Second\_Delay\_Max=110+25;

  /\*

  first: first, second

  second: first, second

  third: first

  \*/

  pfacility First\_Worker;

  pfacility Second\_Worker;

  signalp Signal;

  phistogram Interval\_Graphics;

  initlist(Total\_Modeling\_Time);

  initcreate(1, 0);

  newfac(First\_Worker, "\"First Worker\"");

  newfac(Second\_Worker, "\"Second Worker\"");

  newhist(Interval\_Graphics, 177,277, 10, true, "\"Intervals\"");

  while (systime < Total\_Modeling\_Time) {

    plan();

    switch (sysevent) {

      case 1: create(randab(Interval\_Min,Interval\_Max,v1)); Last\_Time = systime; break;

      case 2: split(1,8); break;

      case 3: seize(First\_Worker); break;

      case 4: delayt(randab(First\_Delay\_Min,First\_Delay\_Max,v1)); break;

      case 5: outfac(First\_Worker); break;

      case 6: if (Second\_Worker->status == facility::seized) accept(Signal); else send(Signal); break;

      case 7: next(13); break;

      case 8: seize(Second\_Worker); break;

      case 9: delayt(randab(Second\_Delay\_Min,Second\_Delay\_Max,v1)); break;

      case 10: outfac(Second\_Worker); break;

      case 11: if (First\_Worker->status == facility::seized) accept(Signal); else send(Signal); break;

      case 12: next(18); break;

      case 13: seize(First\_Worker); break;

      case 14: delayt(randab(First\_Delay\_Min,First\_Delay\_Max,v1)); break;

      case 15: outfac(First\_Worker); break;

      case 16: if (Second\_Worker->status == facility::seized) accept(Signal); else send(Signal); break;

      case 17: next(22); break;

      case 18: seize(Second\_Worker); break;

      case 19: delayt(randab(Second\_Delay\_Min,Second\_Delay\_Max,v1)); break;

      case 20: outfac(Second\_Worker); break;

      case 21: if (First\_Worker->status == facility::seized) accept(Signal); else send(Signal); break;

      case 22: assemble(2); break;

      case 23: tabulate(Interval\_Graphics, systime - Last\_Time); destroy(); break;

    }

  }

  printall();

  cout << "Modeling finished, glory to Jesus Christ our Holy Lord GOD Almighty King of Kings and Lord of Lords" << endl << endl;

}

void two() {

  auto Patient\_Interval\_Min=6-1;

  auto Patient\_Interval\_Max=6+1;

  auto Doctor\_Interval\_Min=7-2;

  auto Doctor\_Interval\_Max=7+2;

  auto Total\_Modeling\_Time = 3\*60;

  auto Total\_Patients = 0;

  pfacility Doctors\_Office;

  plistt Doctors\_Queue;

  initlist(Total\_Modeling\_Time);

  initcreate(1, 0);

  newfac(Doctors\_Office, "\"Doctors Office\"");

  newuserlt(Doctors\_Queue, "\"Doctor Queue\"");

  while (systime < Total\_Modeling\_Time) {

    plan();

    switch (sysevent) {

      case 1: create(randab(Patient\_Interval\_Min, Patient\_Interval\_Max, v1)); break;

      case 2: trans->testprty = false; inlfifo(Doctors\_Queue); outuserlt(Doctors\_Queue); break;

      case 3: seize(Doctors\_Office); break;

      case 4: delayt(randab(Doctor\_Interval\_Min, Doctor\_Interval\_Max, v1)); break;

      case 5: outfac(Doctors\_Office); break;

      case 6: Total\_Patients+=1; destroy(); break;

    }

  }

  cout << "Total Patients: " << Total\_Patients << endl;

  printall();

}

int main()

{

  one();

  two();

  return 0;

}

Виконання програми

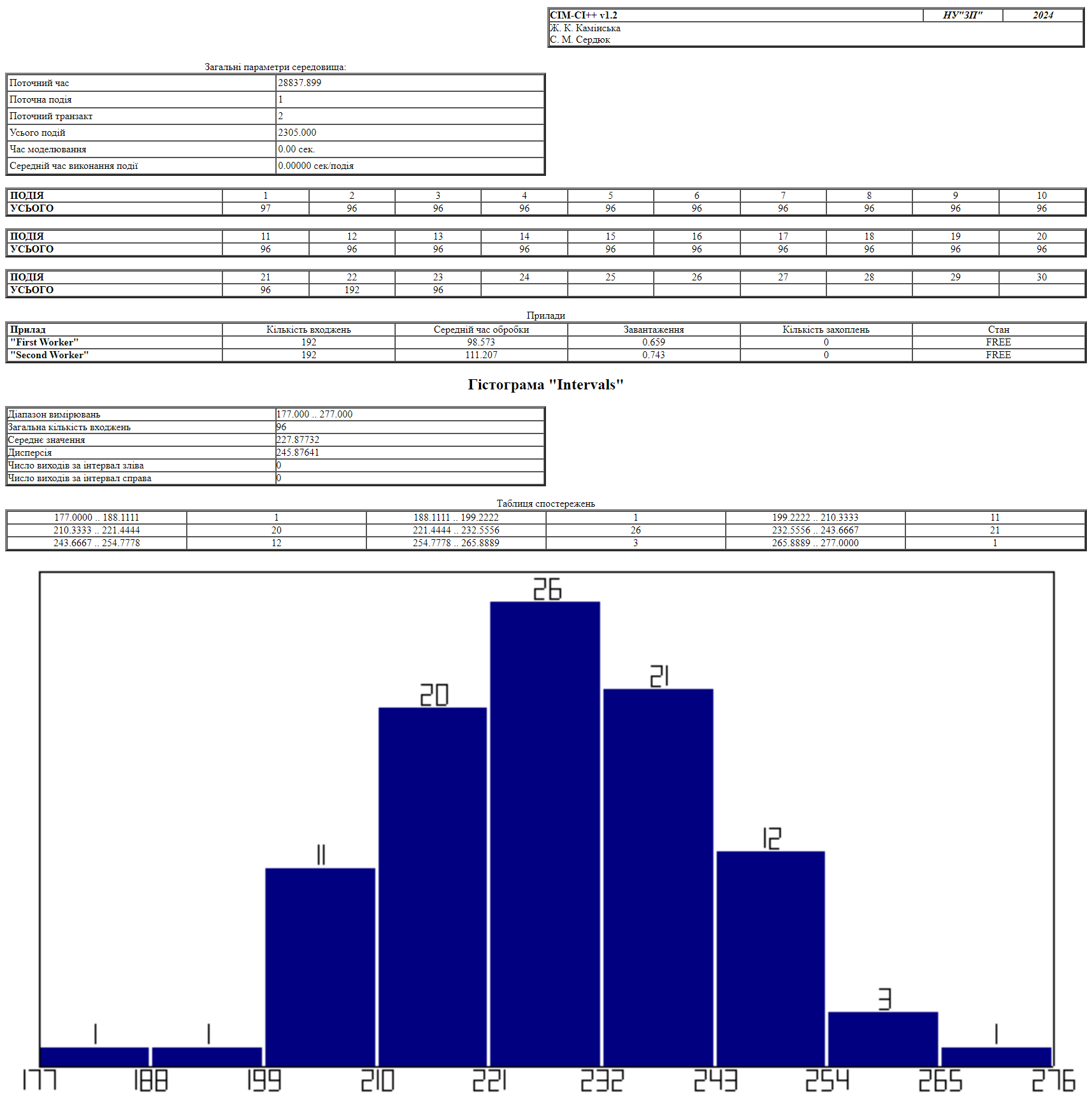


Рисунок 1.1 – Загальне завдання – браузер

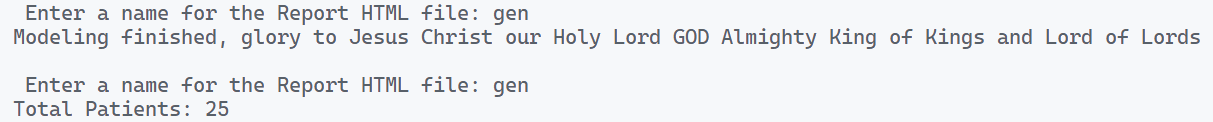


Рисунок 1.2 – Загальне завдання – консолька ♰

Висновки

Таким чином ми освоїли процедури будування вибором випадкових величин та графічного виводу інформації в системі імітаційного моделювання SIMC, а також вивчили принципи моделювання різних дисциплін, які обслуговуються на основі списків користувача

Контрольні питання

Процедури створення гістограм в SIMC

Побудова гістограм у системі імітаційного моделювання SIMC відбувається за допомогою користувацького типу phistogram. У самій моделі посилатися на нього можна черезе phistogram h. Для ініціалізації нової гістограми використовується функція void newhist(phistogram, double, double hint2, bool, alfa); де перший параметр - посилання на гістограму, другий - нижня межа, третій - верхня межа, четвертий - число інтервалів, п'ятий - ключ печатки, шостий - навзва гістограми. Метод табулювання має наступний формат: void tabulate(phistogram hist, double r), де hist - посилання на гістограму, а r - табульоване значення

Процедури блокування транзактів в SIMC

У системі імітаційного моделювання SIMC існують два способи блокування транзактів: через функцію wait або через методи accept/send. Функція wait очікуватиме поки у системі не відбудеться зазначена подія. Функція має наступний вигляд: void wait(event e), де e - номер події на яку чекаємо. Методи accept/send мають наступний вигляд: void accept(signal s), void send(signal s). У SIMC визначений тип сигнал signal, який приймає значненя від 1 до signmax.

Процедури збору статистики в SIMC

У системі імітаційного моделювання SIMC імплементовано автоматичний збір статистики під час виконання програми, однак існують способи виводу та видалення статистики вручну через відповідні методи. Такі методи включають: void resetall() для скидання всієї статистики, що була накопичена під час моделювання, а також void clear() аби повернути усі транзакти у delist та скинути всю поточну статистику.