Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №4

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

«Використання розподілу ймовірності в системі імітаційного моделювання SIMC; Генератор випадкових чисел»

Виконав:

Студент групи КНТ-122 О. А. Онищенко

Прийняли:

Викладач: Ж. К. Камінська

2024

ВИКОРИСТАННЯ РОЗПОДІЛУ ЙМОВІРНОСТІ В СИСТЕМІ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ SIMC; ГЕНЕРАТОР ВИПАДКОВИХ ЧИСЕЛ

Мета роботи

Метою роботи є вивчення методів описання рівномірного та нерівномірного розподілу безперервних та дискретних випадкових величин в SIMC та способів їх практичного використання при моделюванні систем масового обслуговування

Результати виконання

Код програми

```
#include "../simc/simc.h"
#include <iostream>
using namespace std;

void one()
{
  auto Modeling_Hours=40;
  auto Total_Modeling_Time=Modeling_Hours*60;

  auto Assembly_Delay=30;
  auto Firing_Delay=8;

  auto Worker_Hourly_Salary=50;
  auto Firing_Daily_Price=200;
  auto Material_Price=2;
  auto Product_Price=7;

  auto Workers_Min=4;
  auto Workers_Max=6;

  auto Best_Result=0;
  auto Best_Count=0;

  pqueue Assembly_Queue;
  pstorage Assembly_Facility;
  pqueue Firing_Queue;
```

```
  pfacility Firing_Facility;

  table Assembly_Table;
  table Firing_Table;

  Assembly_Table.x[1] = 25;
  Assembly_Table.p[1] = 0.01;
  Assembly_Table.x[2] = 26;
  Assembly_Table.p[2] = 0.04;
  Assembly_Table.x[3] = 27;
  Assembly_Table.p[3] = 0.09;
  Assembly_Table.x[4] = 28;
  Assembly_Table.p[4] = 0.19;
  Assembly_Table.x[5] = 29;
  Assembly_Table.p[5] = 0.37;
  Assembly_Table.x[6] = 30;
  Assembly_Table.p[6] = 0.63;
  Assembly_Table.x[7] = 31;
  Assembly_Table.p[7] = 0.81;
  Assembly_Table.x[8] = 32;
  Assembly_Table.p[8] = 0.91;
  Assembly_Table.x[9] = 33;
  Assembly_Table.p[9] = 0.96;
  Assembly_Table.x[10] = 34;
  Assembly_Table.p[10] = 0.99;
  Assembly_Table.x[11] = 35;
  Assembly_Table.p[11] = 1.0;

  Firing_Table.x[1] = 6;
  Firing_Table.p[1] = 0.05;
  Firing_Table.x[2] = 7;
  Firing_Table.p[2] = 0.3;
  Firing_Table.x[3] = 8;
  Firing_Table.p[3] = 0.7;
  Firing_Table.x[4] = 9;
  Firing_Table.p[4] = 0.95;
  Firing_Table.x[5] = 10;
  Firing_Table.p[5] = 1.0;

  initlist(Total_Modeling_Time);
  initcreate(1, 0);

  newqueue(Assembly_Queue, "\"Assembly Queue\"");
  newqueue(Firing_Queue, "\"Firing Queue\"");
  newfac(Firing_Facility, "\"Firing Facility\"");

  for (auto
Workers_Count=Workers_Min;Workers_Count<=Workers_Max;Workers_Count++) {
```

```
    for (auto j=Workers_Min;j<=Workers_Count;j++) initcreate(1,0);
    newstorage(Assembly_Facility, "\"Assembly Facility\"", 3);
    auto Parts_Assembled=0;
    while (systime<Total_Modeling_Time) {
      plan();
      switch (sysevent) {
        case 1: inqueue(Assembly_Queue); break;
        case 2: enter(Assembly_Facility, 1); break;
        case 3: outqueue(Assembly_Queue); break;
        case 4: delayt(randdtable(Assembly_Table, v1)); break;
        case 5: leave(Assembly_Facility, 1); break;

        case 6: inqueue(Firing_Queue); break;
        case 7: seize(Firing_Facility); break;
        case 8: outqueue(Firing_Queue); break;
        case 9: delayt(randdtable(Firing_Table, v1)); break;
        case 10: outfac(Firing_Facility); Parts_Assembled+=1; break;
        case 11: next(1); break;
      }
    }
    auto
Workers_Salary=Worker_Hourly_Salary*Modeling_Hours*Workers_Count;
    auto Firing_Facility_Cost=Firing_Daily_Price/8*Modeling_Hours;
    auto Materials_Cost=Parts_Assembled*Material_Price;
    auto Parts_Cost=Parts_Assembled*Product_Price;
    auto
Total_Expenses=Workers_Salary+Firing_Facility_Cost+Materials_Cost;
    auto Profit=Parts_Cost-Total_Expenses;
    if (abs(Profit)>abs(Best_Result)) { Best_Result=Profit;
Best_Count=Workers_Count; }
    cout << "Workers: " << Workers_Count << " Profit: " << Profit <<
endl;
  }

  cout << "\nBest Count: " << Best_Count << " Best Result: " <<
Best_Result << endl << endl;
  printall();
}

void two() {
  pfacility Packing_Facility;

  // pack = 12x
  auto Packs_Count=0;

  auto Modeling_Time=1*60*60;

  initlist(Modeling_Time);
```

```
  initcreate(1, 0);
  newfac(Packing_Facility, "\"Packing Facility\"");

  while (systime < Modeling_Time) {
    plan();
    switch (sysevent) {
      case 1: split(12, 2); break;
      case 2: seize(Packing_Facility); break;
      case 3: delayt(randab(13,19,v1)); break;
      case 4: outfac(Packing_Facility); break;
      case 5: assemble(12); break;
      case 6: Packs_Count+=1; next(1); break;
    }
  }
  printall();
  clear();
  cout << "Total Packs: " << Packs_Count << endl << endl;
}

void three()
{
  auto Modeling_Time = 8 * 60 * 60;
  auto Maximum_Queue = 0;
  auto Maximum_Bags = 0;

  pfacility Cash_Register;
  pstorage Store_Baskets;
  pqueue Store_Queue;

  initlist(Modeling_Time);
  initcreate(1, 0);
  newfac(Cash_Register, "\"Cash Register\"");
  newstorage(Store_Baskets, "\"Store Rows\"", 100);
  newqueue(Store_Queue, "\"Store Queue\"");

  while (systime < Modeling_Time) {
    plan();
    switch (sysevent) {
      case 1: create(randexp(75, v1)); break;
      case 2: enter(Store_Baskets, 1); break;
      case 3: if (rand01(v1) > 0.75) { trans->pi[1] += randab(2, 4, v1);
delayt(randab(60, 180, v1)); } else next(4); break;
      case 4: if (rand01(v1) > 0.55) { trans->pi[1] += randab(3, 5, v1);
delayt(randab(120, 180, v1)); } else next(5); break;
      case 5: if (rand01(v1) > 0.82) { trans->pi[1] += randab(4, 6, v1);
delayt(randab(75, 165, v1)); } else next(6); break;
      case 6: inqueue(Store_Queue); trans->pi[1] += randab(1,3,v1);
break;
```

```
      case 7: seize(Cash_Register); break;
      case 8: outqueue(Store_Queue); break;
      case 9: delayt(3*trans->pi[1]); break;
      case 10: outfac(Cash_Register); break;
      case 11: leave(Store_Baskets, 1); break;
      case 12: destroy(); break;
    }

    if (Store_Baskets->sm > Maximum_Bags) Maximum_Bags = Store_Baskets-
>sm;
    if (Store_Queue->mq > Maximum_Queue) Maximum_Queue = Store_Queue->mq;
  }

  cout << "Maximum Queue: " << Maximum_Queue << endl
       << "Maximum Bags: " << Maximum_Bags << endl << endl;
  printall();
  clear();
}

int main()
{
  one();
  two();
  three();
  return 0;
}
```

Виконання програми

CIM-CH++ v1.2 | *НУ"ЗП"* | *2024*

Ж. К. Камінська
С. М. Сердюк

Загальні параметри середовища:

Поточний час	2404.000
Поточна подія	5
Поточний транзакт	2
Усього подій	1399.000
Час моделювання	0.00 сек.
Середній час виконання події	0.00000 сек/подія

ПОДІЯ	1	2	3	4	5	6	7	8	9	10
УСЬОГО	127	127	127	127	126	125	140	125	125	125

ПОДІЯ	11	12	13	14	15	16	17	18	19	20
УСЬОГО	125									

Черги

Черга	Кількість входжень	Макс. довжина	Середній час очікування	Середня довжина	% входжень у порожню чергу
	З нульовим часом очікування	Поточна довжина	Без урахування нульових входжень		
"Assembly Queue"	127	1	0.000	0.000	100.000
	127	0	0.000		
"Firing Queue"	125	1	0.200	0.010	91.200
	114	0	2.273		

Прилади

Прилад	Кількість входжень	Середній час обробки	Завантаження	Кількість захоплень	Стан
"Firing Facility"	125	7.992	0.417	0	FREE

Накопичувачі

Накопичувач	Ємність	Завантаженість	Середній час перебування	Вміст			Кількість входжень
				Поточн.	Макс.	Середн.	
"Assembly Facility"	3	0.525	29.795	1	2	1.57	127

Рисунок 1.1 – Загальне завдання – браузер

```
 Enter a name for the Report HTML file: gen
Workers: 4 Profit: -8375
Workers: 5 Profit: -11000
Workers: 6 Profit: -13000

Best Count: 6 Best Result: -13000

 Enter a name for the Report HTML file: gen
Total Packs: 18

 Enter a name for the Report HTML file: gen
Maximum Queue: 3
Maximum Bags: 8
```

Рисунок 1.2 – Загальне завдання – консолька ✝

Висновки

Таким чином ми вивчили методи описання рівномірного та нерівномірного розподілу безперервних та дискретних випадкових величин в SIMC та способи їх практичного використання при моделюванні систем масового обслуговування

Контрольні питання

Генерація випадкових величин

В процесі моделювання коли потрібен генератор чисел то використовуємо такі функції:

rand01(v) - генерує числа рівномірно в інтервалі [0,1)

randab(a,b,v) - генерує числа рівномірно в інтервалі [a,b)

randexp(lambda,v) - генерує числа експоненційно з інтенсивністю lambda

randnorm(xmean,disp,v) - генерує числа за нормальним законом з середнім xmean і дисперсії disp

Процедури роботи з ансамблями в CIM SIMC

Для роботи з ансамблями в SIMC існують такі процедури:

split(n,e) - де n то є число транзактів на створення, а e то є номер події де транзакти направляються

assemble(n) - де n то є число транзактів на збір

priority(p) - де p то є нове значення пріоритету

params(p) - параметр p застосовується до всіх транзактів

Реалізація безперервних та дискретних випадкових величин заданих в SIMC вигляді таблиці

Для безперервного розподілу є функція double randtable(table t, long v) де t то є таблиця а v то є число джерело

Для дискретного розподілу є функція double randdtable(table t, long v) де t то є таблиця а v то є число джерело