5. PRAKTIKA

1. Lehenengo ataza, simulatzailea martxan jartzea izango da. Zuen esku utziko dut IDE-aren aukeraketa.Horrez gain, eGelan C-n programatzearen inguruko ataltxo bat duzue laguntza edo material gehigarriabehar izanez gero.

Simuladorea martxan jartzeko nire IDE aukera *Visual Studio* izan da, bere interfaza oso erabilerraza suertatu zait.

2. Identifikatu simulatzailea osatzen duten elementu desberdinak. Zeintzuk dira horiek? Nola interakzio-natzen dute haien artean? Interakzioen-diagrama bat egitea interesgarria izan daiteke azalpena errazteko

Parameters

Hemen simulatzaileak beharko dituen parametroak definitzen dira gero beste programek erabili ahal izateko. Ala nola *arrival()*, *service()* edo *simulation_time*. *Arrival* eta *service* funtzioetan aldaketak eginez ilara mota aldatu daiteke.

Main

Programa honek simulazio guztia martxan jartzen du *event* bakoitza exekutatuz simulazio denborara iritxi arte. Honetarako Simulator-i deitzen dio. Gainera simulazioa denbora kalkulatzen du.

Simulator

Event-ak ilara batera sartu eta bertatik ateratzen ditu eta beraiek prozesatzen ditu. Gainera System-i deiak egiten dizkio prozesua martxan jar dezan. Azkenik klase honetan zerbitzaria simulatzeko beharrezko elementuak daude.

System

Prozesua martxan jartzeaz enkargatzen da, ala nola ilara mota zeaztu (honetarako parameters.h erabiliz), gainera event-ak hasieratu, ilaran sartu eta bertatik ateratzen ditu. Azkenik azken emaitzak pantailaratzeaz ere bera enkargatzen da.

Event

Event motako objetuak sortzen dira hemen beste programek erabil ditzaten.

3. Zein ilara-eredu inplementatzen du? Erabili Kendall-en notazioa lagungarria bazaizu.

G/D/1 motako ilara eredua inplementatzen du, *parameters.h* dokumentuan definitzen diren `double arrival() { return 2 * udistr(rengine) * 14.0; }` eta `double service() { return 10; }` ikusita ikusten da zerbitzua determinatua dela eta iritsera distribuzio uniforme baten baitan ematen direla.

4. Zeintzuk dira sistemaren defektuzko sarrera-parametroak? Zeintzuk estatistiko kalkulatzen ditu? Deskribatu ateratzen duen emaitza (grafikaren bat egitea interesgarria izan daiteke).

Ikus daitekeen moduan sarrera-parametroak hemengo hauek dira. Simulazio denbora, ilara-mota definitzeko beharrezkoak diren *arrival* eta *service* eta beste ilara batzuk simulatzeko erabili daitezkeen *seed* eta *lambda* (baina laborategi honetan **EZ** dira erabiltzen). Gainera *Time_intervals* ezberdinak kalkulatzen dira distribuzio ezberdinak erazarriz.

```
const bool print_trace = true;         // print simulation trace
const double simulation_time = 200.0;    // model simulation time
const int seed = 123123;
const double lambda= 2.0;
class Time_intervals {
    default_random_engine rengine {seed};  // random numbers generator
    uniform_real_distribution<double> udistr {};  // default [0,1] interval
    exponential_distribution<double> edistr {lambda};  // default lambda=1
    normal_distribution<double> ndistr{20,5};
public:
    double arrival() { return 2 * udistr(rengine) * 14.0; }
    double service() { return 10; }
};
```

Gainera C++ dokumentu bakoitzak eraikitzaile bana daukate beharrezkoak diren beste eragigaiak sortzeko.

Azkenik simulazioa gauzatu ondoren lortzen ditun emaitzak beheko taulan ikus ditzazkegunak dira.

STATISTICS	
Simulation time	200
In clients / Out clients	14 / 14
Client inter-arrival time	14.2857
Served client throughput	0.07
Mean service time	10
Utilization	0.7

Emaitzak aztertzeko Batazbesteko erantzun denbora ikusten da interesgarrien moduan.

BB erantzun denbora = In clients / Client inter-arrival time

BBed = 14 / 14.2857 = 0,98 s

Eskaera bakoitza tratzeko ia segundu bat behar du sistemak, konparaketarako datu gehiago gabe ezin esan liteke positiboa ote den. Ala ere ez du erabiltzailerik galtzen eta hau azparragarria da.

6. PRAKTIKA

Ilara modelo berriak inplementatzeko hurrengo aldaketak gauzatu dira parameters.h fitxategian:

```
public:
   double arrival() { return 2 * udistr(rengine) * 14.0; } //G
   double arrival() {return 2 * edistr(rengine) * 14.0; } //M
   double service() { return 10; } //D
   double service() { return 2 * edistr(rengine) * 14.0; } //M
};
```

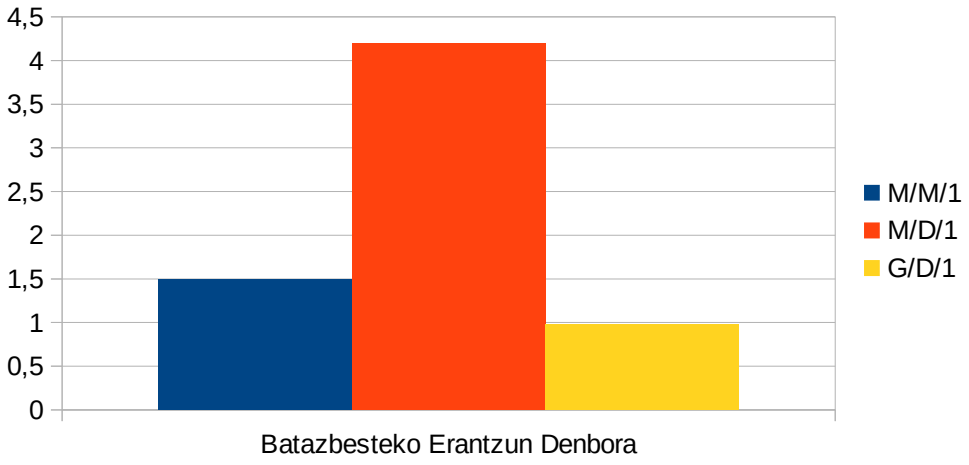
Komentario bakoitzean ikus daiteke zein motatakoa osatzen duen eta hauen konbinaketen bitartez sor daitezke ilara mota ezberdinak batzuk komentatu edo deskomentatuz. Adibidez M/M/1 motako ilara bat exekutatzeko lehen eta hirugarren lerroak komentatzea nahikoa izango da (G eta D jartzen dutenak).

Modu honetan exekuzio bakoitzaren ondoren hurrengo emaitzak bildu ditut:

M/M/1	M/D/1	G/D/1
Simulation time 200 In clients 20 Out clients 14 Client inter-arrival time 10 Served client throughput 0.07 Mean service time 13.3888 Utilization 0.937216	Simulation time 200 In clients 29 Out clients 19 Client inter-arrival time 6.89655 Served client throughput 0.095 Mean service time 10 Utilization 0.95	Simulation time 200 In clients 14 Out clients 14 Client inter-arrival time 14.2857 Served client throughput 0.07 Mean service time 10 Utilization 0.7

5. praktikako kasuan bezala aztertzeko eredu bakoitzaren batazbesteko erantzun denbora erabiliko dugu:

M/M/1	M/D/1	G/D/1
20 / 13.3888 = 1,49	29 / 6,89655 = 4,2	14 / 14.2857 = 0,98 s



Grafikoan argi ikusten den moduan G/D/1 motako ilarak dira batazbesteko erantzun denbora baxueana dutenak, ala ere beraiek dira sarrera gutxien (14) dituztenak eta client inter-arrival time altuena dutenak (14.2857) beraz normala da balio oso baxuak izatea mota honetako ilarak.

K tamainako bufferra

Atal honetan k tamainako buffer batekin bezero kopurua mugatuko da. Honetarako parameters.h eta system.cpp fitxategietako kodea aldatu behar da, behean ikusten den moduan:

Parameters.h

```
const int k = 10; //2 5
```

Hasierako definizioetan zehaztu da k-ren tamaina, frogetan k = 2, k = 5 eta k = 10 balioekin egin dira.

System.cpp

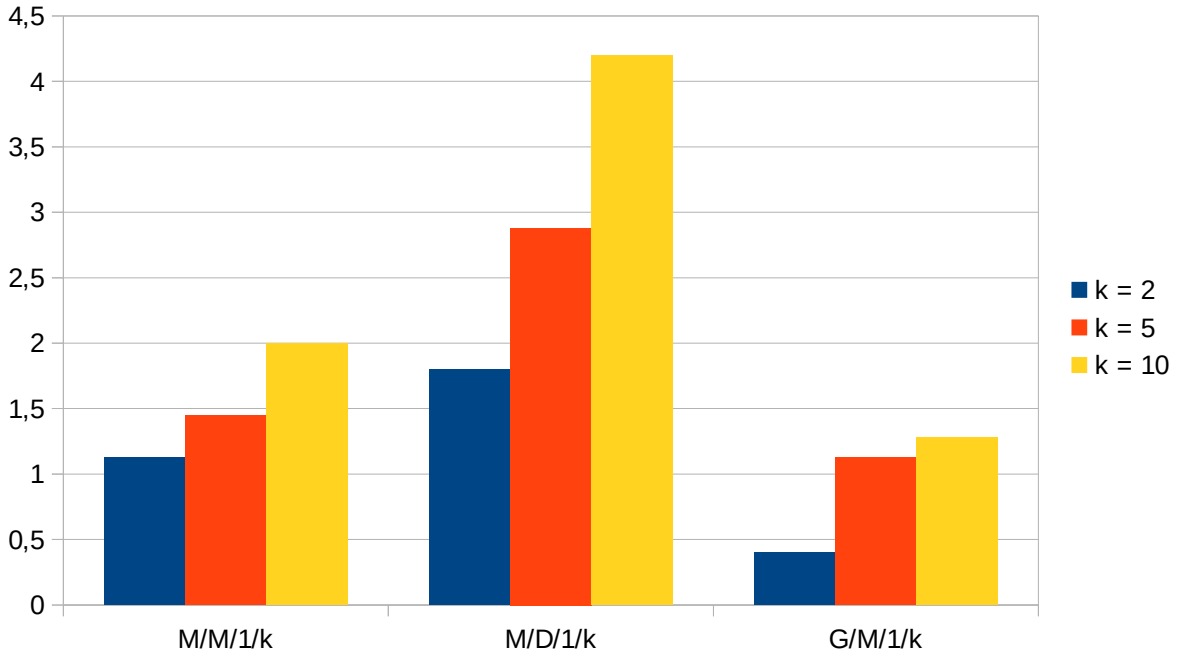
```
if (m_num_clients < k) {
        m_num_clients++;
        m_in_clients++;
        event->Set_time(event_time+m_tm_intervals.arrival());
        Event * client_arrival_event= new
Event(event_time+m_tm_intervals.arrival(),client_arrival);
        //System_event client_arrival_event {event_time + tm_intervals.arrival(),
client_arrival};
        new_events.push_back(client_arrival_event);
        break;
}
else {
        event->Set_time(event_time + m_tm_intervals.arrival());
        Event * client_arrival_event = new Event(event_time + m_tm_intervals.arrival(),
client_arrival);
        //System_event client_arrival_event {event_time + tm_intervals.arrival(),
client_arrival};
        new_events.push_back(client_arrival_event);
        break;
}
```

Horiz azparraturik ikus daiteke kode originalari gehituriko kode zatiak. Eskaera kopurua k mugara iritsi bada *else* klausula exekutatuko du, *event* berri bat sortuz. Aldiz iritsi ez bada ohiko exekuzioa egingo du.

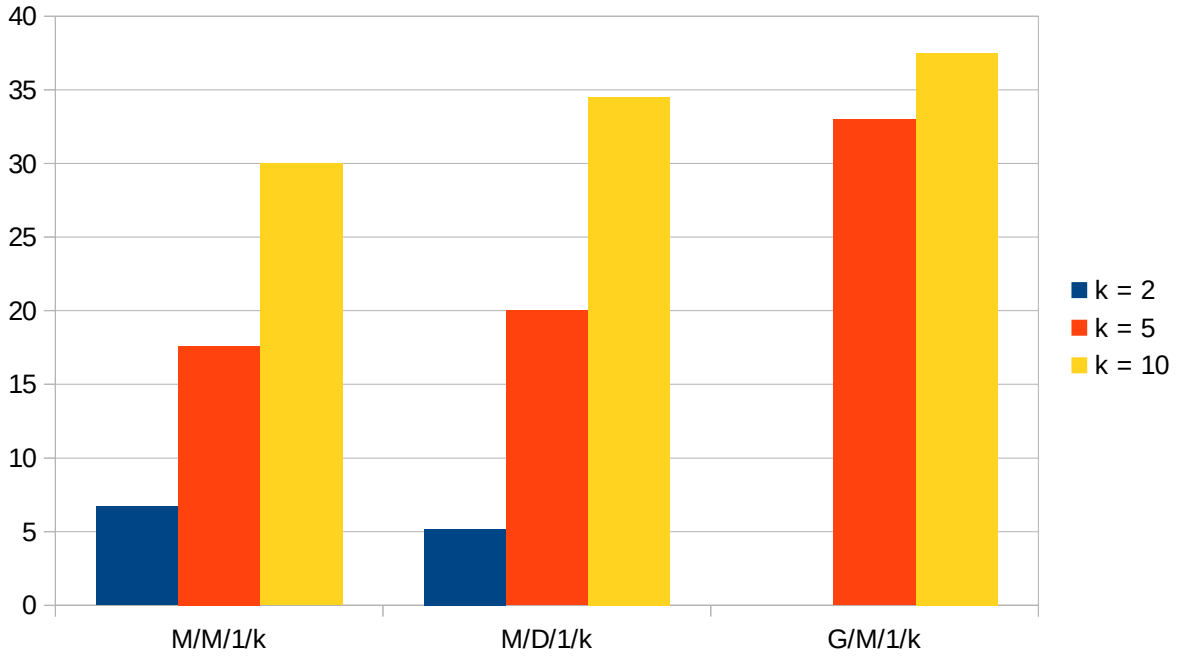
Bufferraren tamaina ezberdinak erabiliz (2, 5 eta 10) eta ilara mota ezberdinak (M/M/1/k, M/D/1/k eta G/M/1/k) ondorengo emaitzak lortu dira:

	M/M/1/k	**M/D/1/k**	**G/M/1/k**
K = 2	Simulation time 200 In clients 15 Out clients 14 Client inter-arrival time 13.3333 Served client throughput 0.07 Mean service time 7.58695 Utilization 0.531087	Simulation time 200 In clients 19 Out clients 18 Client inter-arrival time 10.5263 Served client throughput 0.09 Mean service time 10 Utilization 0.9	Simulation time 200 In clients 9 Out clients 9 Client inter-arrival time 22.2222 Served client throughput 0.045 Mean service time 16.2288 Utilization 0.730295
BBed	1,125	1,8	0,4
Galdu%	0,067 (%6,7)	0,052 (%5,2)	0 (%0)
K = 5	Simulation time 200 In clients 17 Out clients 14 Client inter-arrival time 11.7647 Served client throughput 0.07 Mean service time 13.3888 Utilization 0.937216	Simulation time 200 In clients 24 Out clients 19 Client inter-arrival time 8.33333 Served client throughput 0.095 Mean service time 10 Utilization 0.95	Simulation time 200 In clients 15 Out clients 10 Client inter-arrival time 13.3333 Served client throughput 0.05 Mean service time 18.6889 Utilization 0.934443
BBed	1,445	2,88	1,125
Galdu%	0,176 (%17,6)	0,2 (%20)	0,33 (%33)
K = 10	Simulation time 200 In clients 20 Out clients 14 Client inter-arrival time 10 Served client throughput 0.07 Mean service time 13.3888 Utilization 0.937216	Simulation time 200 In clients 29 Out clients 19 Client inter-arrival time 6.89655 Served client throughput 0.095 Mean service time 10 Utilization 0.95	Simulation time 200 In clients 16 Out clients 10 Client inter-arrival time 12.5 Served client throughput 0.05 Mean service time 18.6889 Utilization 0.934443
BBed	2	4,2	1,28
Galdu%	0,3 (%30)	0,345 (%34,5)	0,375 (%37,5)

Batazbesteko erantzun denboretan oinarritutako grafikoa:



Galdutako erabiltzaileen portzentaiaren araberako grafikoa:



Lehen grafikan argi ikus daiteke denbora txikienak G/M/1/k motakoek dituztela, k guztietan, aldiz bigarren grafikoan argi ikusten da pakete gehien ere mota honetako ilaretan galtzen dirala k handituz doan einean. Bi grafikoak kontutan hartuz gero ondorio argia da M/M/1/k motako ilarak direla aukera egokiena, denbora G/M/1/k-k baino kaxkarragoa izan arren M/D/1/k-ena baina hobea da eta galduen kopuru baxuena du nabarmen.