Osnovne akademske studije

PREDMET: Objektno-orijentisana simulacija

TEMA: Strategija raspoređivanja događaja 3

Predmetni nastavnik: Prof. dr Milorad Stanojević
Asistent: mr Marko Đogatović

Primer 3. U pošti postoje dva univerzalna šaltera i jedan paketski šalter.Na univerzalne šaltere klijenti dolaze po puasonovom toku sa srednjim vremenom od 60 sek.

Na paketske šaltere klijenti dolaze po uniformnom zakonu 200±80 sek.

Univerzalni šalteri imaju zajednički red čekanja. Vreme opsluge je eksponecijalno raspodeljeno sa srednjim vremenom koje zavisi od vrste usluge, a prema tabeli 1.

Tabela 1.

Vrsta usluge	Verovatnoća izbora usluge	Srednje vreme opsluge (sek)
1	0.20	40
2	0.40	50
3	0.15	15
4	0.25	70

Vreme opsluge na paketskom šalteru je normalno raspodeljeno sa srednjim vremenom od 150 s i standardnom devijacijom od 60 s sa tim da to vreme ne može da bude manje od 20 sek.

Simulirati rad pošte u toku jednog dvanestočasovnog radnog dana. Vremenska jedinica je 1 sekunda. Za rešavanje zadatka koristiti strategiju raspoređivanja događaja. Napisati bezuslovne događaje korišćenjem pseudokoda, realizovati model u programskom jeziku C++ i izračunati statistike redova čekanja i resursa.

Bezuslovni događaji – Primer 3

```
procedure DolazakKlijentaUUniverzalniSalter
begin
        Postavi kljenta u red čekanja.
        if šalter je raspoloživ then
        begin
                Izvadi prvog klijenta iz reda čekanja.
                Zauzmi šalter.
                Izbor vrste usluge i dodeljivanje vremena opsluge.
                Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera.
        end
        Napravi novog klijenta.
        Rasporedi klijenta na događaj DolazakKlijentaUUniverzalniSalter.
end
```

```
procedure OdlazakKlijentaIzUniverzalnogSaltera
begin
        Oslobodi šalter.
        Uništi klijenta.
        if red čekanja nije prazan then
        begin
                Izvadi prvog klijenta iz reda čekanja.
                Zauzmi šalter.
                Izbor vrste usluge i dodeljivanje vremena opsluge.
                Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera.
        end
end
```

```
procedure DolazakKlijentaUPaketskiSalter
begin
        Postavi kljenta u red čekanja.
        if šalter je raspoloživ then
        begin
                Izvadi prvog klijenta iz reda čekanja.
                Zauzmi šalter.
                repeat
                        Dodeljivanje vremena opsluge.
                until vreme opsluge > 20
                Rasporedi klijenta na događaj OdlazakKlijentaIzPaketskogSaltera.
        end
        Napravi novog klijenta.
        Rasporedi klijenta na događaj DolazakKlijentaUPaketskiSalter.
end
```

```
procedure OdlazakKlijentaIzPaketskogSaltera
begin
        Oslobodi šalter.
        Uništi klijenta.
        if red čekanja nije prazan then
        begin
                Izvadi prvog klijenta iz reda čekanja.
                Zauzmi šalter.
                repeat
                        Dodeljivanje vremena opsluge.
                until vreme opsluge > 20
                Rasporedi klijenta na događaj OdlazakKlijentaIzPaketskogSaltera.
        end
end
```

Izvorna datoteka simu3.cpp

```
// Datototeka simu2.cpp
#include "simu2.h"
#include <iostream>

// Makro naredbe
#define ID(e) (e)->vrati_id()
#define VREME simu.vrati_vreme_simulacije()
#define VRTR(vreme) simu.vrati_vreme_simulacije()+(vreme)

// Ukljucujemo prostore imena simu2 i std
using namespace simu2;
using namespace std;

// Redni broj dogadjaja DolazakKlijentaUUniverzalniSalter
const int DOLAZAK_UNI_KLIJENTA = 0;
// Redni broj dogadjaja OdlazakKlijentaIzUniverzalnogSaltera
const int ODLAZAK_UNI_KLIJENTA = 1;
// Redni broj dogadjaja DolazakKlijentaUPaketskiSalter
const int DOLAZAK_PAK_KLIJENTA = 2;
// Redni broj dogadjaja OdlazakKlijentaIzPaketskogSaltera
const int ODLAZAK_PAK_KLIJENTA = 3;
// Redni broj dogadjaja KrajSimulacije
const int KRAJ_SIMULACIJE      = 4;

// Potpis funkcije bezuslovnog dogadjaja DolazakKlijentaUUniverzalniSalter
void dolazak_uni_klijenta(entitet *e);
// Potpis funkcije bezuslovnog dogadjaja OdlazakKlijentaIzUniverzalnogSaltera
void odlazak_uni_klijenta(entitet *e);
// Potpis funkcije bezuslovnog dogadjaja DolazakKlijentaUPaketskiSalter
void dolazak_pak_klijenta(entitet *e);
// Potpis funkcije bezuslovnog dogadjaja OdlazakKlijentaIzPaketskogSaltera
void odlazak_pak_klijenta(entitet *e);
// Potpis funkcije bezuslovnog dogadjaja KrajSimulacije
```

```
void kraj_simulacije(entitet *e);

// Izvrsavanje dogadjaja
void simulacija::izvrsavanje_dogadjaja(int dog, entitet* e) {
  switch(dog) {
    case DOLAZAK_UNI_KLIJENTA:
      dolazak_uni_klijenta(e);
      break;
    case ODLAZAK_UNI_KLIJENTA:
      odlazak_uni_klijenta(e);
      break;
    case DOLAZAK_PAK_KLIJENTA:
      dolazak_pak_klijenta(e);
      break;
    case ODLAZAK_PAK_KLIJENTA:
      odlazak_pak_klijenta(e);
      break;
    case KRAJ_SIMULACIJE:
      kraj_simulacije(e);
  }
}
// Redovi cekanja
fifo_red red1,red2;
// Salteri
resurs uni_salt(2),pak_salt(1);
// Objekat simulacije
simulacija simu;
// Raspodele
raspodela rni(76531), rn1(11615), rn2(36517), rn3(61521), rn4(98737);

// Srednje vreme dolaska entiteta u univerzalni salter
const double SRVRDOLUNI  = 60;
// Srednja vremena opsluge entiteta u univerzalnom salteru u zavisnosti od usluge
const double SRVROPSUNI[] = {40, 50, 15, 70};
// Srednje vreme dolaska entiteta u paketski salter
const double SRVRDOLPAK  = 200;
// Modifikator vremena dolaska entiteta u paketski salter
```

```
const double MODDOLPAK   = 80;
// Srednje vreme opsluge entiteta u paketskom salteru
const double SRVROPSPAK  = 150;
// Standardna devijacija vremena opsluge entiteta u paketskom salteru
const double STDEVOPSPAK = 60;
// Minimalno vreme opsluge entiteta u paketskom salteru
const double MINOPSPAK = 20;

// Dogadjaj dolazak klijenta na univerzalni salter
void dolazak_uni_klijenta(entitet *e) {
  entitet *fe, *ne;
  double pi,vreme_opsluge;
  cout << "ured_uni" << "," << ID(e) << "," << VREME << endl;
  // Postavi klijenta u red cekanja
  red1.ured(e);
  // Ukoliko je salter raspoloziv
  if(uni_salt.raspoloziv()) {
    // Izvadi prvog klijenta iz reda
    fe = red1.izred();
    cout << "izred_uni" << "," << ID(fe) << "," << VREME << endl;
    // Zauzmi jedno mesto u salteru
    uni_salt.zauzmi();
    cout << "usal_uni" << "," << ID(fe) << "," << VREME << endl;
    // Verovatnoca izbora usluge
    pi = rni();
    // Vreme opsluge na osnovu izabrane vrste usluge
    if(pi<0.20)
      vreme_opsluge = rn2.expo(SRVROPSUNI[0]);
    else if(pi<0.60)
      vreme_opsluge = rn2.expo(SRVROPSUNI[1]);
    else if(pi<0.75)
      vreme_opsluge = rn2.expo(SRVROPSUNI[2]);
    else if(pi<1.00)
      vreme_opsluge = rn2.expo(SRVROPSUNI[3]);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_UNI_KLIJENTA,VRTR(vreme_opsluge));
  }
```

```
  // Stvaramo narednog klijenta
  ne = simu.napravi_entitet();
  // Postavljamo vreme narednog dolaska
  simu.rasporedi(ne,DOLAZAK_UNI_KLIJENTA,VRTR(rn1.expo(SRVRDOLUNI)));
}
// Dogadjaj odlazak klijenta sa univerzalnog saltera
void odlazak_uni_klijenta(entitet *e) {
  entitet *fe;
  double pi,vreme_opsluge;
  // Vrati salter;
  uni_salt.oslobodi();
  cout << "izsal_uni" << "," << ID(e) << "," << VREME << endl;
  // Klijent odlazi iz poste - unistavamo tekuceg klijenta
  simu.unisti_entitet(e,0);
  // Ukoliko red nije prazan
  if(red1.velicina()>0) {
    // Izvadi prvog klijenta iz reda
    fe = red1.izred();
    cout << "izred_uni" << "," << ID(fe) << "," << VREME << endl;
    // Zauzmi jedno mesto u salteru
    uni_salt.zauzmi();
    cout << "usal_uni" << "," << ID(fe) << "," << VREME << endl;
    // Verovatnoca izbora usluge
    pi = rni();
    // Vreme opsluge na osnovu izabrane vrste usluge
    if(pi<0.20)
      vreme_opsluge = rn2.expo(SRVROPSUNI[0]);
    else if(pi<0.60)
      vreme_opsluge = rn2.expo(SRVROPSUNI[1]);
    else if(pi<0.75)
      vreme_opsluge = rn2.expo(SRVROPSUNI[2]);
    else if(pi<1.00)
      vreme_opsluge = rn2.expo(SRVROPSUNI[3]);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_UNI_KLIJENTA,VRTR(vreme_opsluge));
  }
}
```

```
// Dogadjaj dolazak klijenta na paketski salter
void dolazak_pak_klijenta(entitet *e) {
  entitet *fe, *ne;
  double vreme_opsluge;
  cout << "ured_pak" << "," << ID(e) << "," << VREME << endl;
  // Postavi klijenta u red cekanja
  red2.ured(e);
  // Ukoliko je salter raspoloziv
  if(pak_salt.raspoloziv()) {
    // Izvadi prvog klijenta iz reda
    fe = red2.izred();
    cout << "izred_pak" << "," << ID(fe) << "," << VREME << endl;
    // Zauzmi jedno mesto u salteru
    pak_salt.zauzmi();
    cout << "usal_pak" << "," << ID(fe) << "," << VREME << endl;
    do {
      vreme_opsluge = rn4.norm(SRVROPSPAK,STDEVOPSPAK);
    } while(vreme_opsluge<MINOPSPAK);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_PAK_KLIJENTA,VRTR(vreme_opsluge));
  }
  // Stvaramo narednog klijenta
  ne = simu.napravi_entitet();
  // Postavljamo vreme narednog dolaska
  simu.rasporedi(ne,DOLAZAK_PAK_KLIJENTA,
        VRTR(rn3.unif(SRVRDOLPAK-MODDOLPAK,SRVRDOLPAK+MODDOLPAK)));
}
// Dogadjaj odlazak klijenta sa paketskog saltera
void odlazak_pak_klijenta(entitet *e) {
  entitet *fe;
  double vreme_opsluge;
  // Vrati paketski salter;
  pak_salt.oslobodi();
  cout << "izsal_pak" << "," << ID(e) << "," << VREME << endl;
  // Klijent odlazi iz poste - unistavamo tekuceg klijenta
  simu.unisti_entitet(e,0);
  // Ukoliko red ispred paketsko saltera nije prazan
```

```
  if(red2.velicina()>0) {
    // Izvadi prvog klijenta iz reda
    fe = red2.izred();
    cout << "izred_pak" << "," << ID(fe) << "," << VREME << endl;
    // Zauzmi jedno mesto u paketskom salteru
    pak_salt.zauzmi();
    cout << "usal_pak" << "," << ID(fe) << "," << VREME << endl;
    do {
      vreme_opsluge = rn4.norm(SRVROPSPAK,STDEVOPSPAK);
    } while(vreme_opsluge<MINOPSPAK);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_PAK_KLIJENTA,VRTR(vreme_opsluge));
  }
}
// Dogadjaj kraj simulacije
void kraj_simulacije(entitet* e) {
  simu.unisti_entitet(e,1);
  // Stampamo vreme simulacije
  cout << "vsim" << "," << -1 << "," << VREME << endl;
}
int main() {
  // Stvaramo prvog klijenta
  entitet *e1 = simu.napravi_entitet();
  // Postavljamo vreme dolaska prvog klijenta
  simu.rasporedi(e1,DOLAZAK_UNI_KLIJENTA,rn1.expo(SRVRDOLUNI));
  // Stvaramo prvog klijenta
  entitet *e2 = simu.napravi_entitet();
  // Postavljamo vreme dolaska prvog klijenta
  simu.rasporedi(e2,DOLAZAK_PAK_KLIJENTA,
                 rn3.unif(SRVRDOLPAK-MODDOLPAK,SRVRDOLPAK+MODDOLPAK));
  // Stvaramo entitet za kraj simulacije
  entitet *e3 = simu.napravi_entitet();
  // Postavljamo vreme zavrsetka simulacije
  simu.rasporedi(e3,KRAJ_SIMULACIJE,43200);
  // Izvrsavamo simulaciju
  simu.izvrsi(1);
}
```

Listing simulacionog programa realizovanog u GPSS/H

```
Student GPSS/H Release 3.70 (UN072)     10 Feb 2013   01:50:31     File: SIMU3.gps

   Line# Stmt#  If Do  Block#  *Loc Operation      A,B,C,D,E,F,G  Comments

       1     1                        SIMULATE
       2     2                 USLUGA FUNCTION RN4,D4
       3     3                 0.20,1/0.60,2/0.75,3/1.00,4
       4     4                 SRVRO  FUNCTION P1,D4
       5     5                 1,40/2,50/3,15/4,70
       6     6                        STORAGE S(1),2
       7     7             1          GENERATE RVEXPO(1,60)
       8     8             2          ASSIGN 1,FN(USLUGA)
       9     9             3          QUEUE 1
      10    10             4          ENTER 1
      11    11             5          DEPART 1
      12    12             6          ADVANCE RVEXPO(2,FN(SRVRO))
      13    13             7          LEAVE 1
      14    14             8          TERMINATE
      15    15                *
      16    16             9          GENERATE RVUNI(3,200,80)
      17    17            10          QUEUE 2
      18    18            11          SEIZE 2
      19    19            12          DEPART 2
      20    20            13   PON    ASSIGN 1,RVNORM(5,150,60)
      21    21            14          TEST G P1,20,PON
      22    22            15          ADVANCE P1
      23    23            16          RELEASE 2
      24    24            17          TERMINATE
      25    25                *
      26    26            18          GENERATE 43200
      27    27            19          TERMINATE 1
```

```
       28    28                        START 1
       29    29                        END

Entity Dictionary (in ascending order by entity number; "*" => value conflict.)

       Facilities: 2

           Queues: 1               2

         Storages: 1

        Functions: 1=USLUGA        2=SRVRO

       Parameters: 1

   Random Numbers: 1               2               3               4               5

Symbol    Value    EQU Defns  Context      References by Statement Number

PON          13           20  Block           21

2             2               Facility        18    23

1             1               Queue            9    11
2             2               Queue           17    19

1             1            6  Storage         10    13

SRVRO         2            4  Function        12
USLUGA        1            2  Function         8

1             1               Parameter        4     8    20    21    22

1             1               Random Nmbr      7
2             2               Random Nmbr     12
3             3               Random Nmbr     16
4             4               Random Nmbr      2
```

```
5            5              Random Nmbr     20


Storage Requirements (Bytes)

Compiled Code:       848
Compiled Data:       248
Miscellaneous:         0
Entities:            954
Common:            10000
-----------------------
Total:             12050


GPSS/H Model Size:

Control Statements       6
Blocks                  19


Simulation begins.

Relative Clock: 43200.0000   Absolute Clock: 43200.0000


Block Current     Total  Block Current     Total
1                   724  11                  209
2                   724  12                  209
3                   724  PON                 213
4                   724  14                  213
5                   724  15          1       209
6           1       724  16                  208
7                   723  17                  208
```

8	723	18	1
9	209	19	1
10	209		

Facility	--Avg-Util-During-- Total Time	Avail Time	Unavl Time	Entries	Average Time/Xact	Current Status	Percent Avail	Seizing Xact	Preempting Xact
2	0.747			209	154.415	AVAIL		928	

Storage	--Avg-Util-During-- Total Time	Avail Time	Unavl Time	Entries	Average Time/Unit	Current Status	Percent Avail	Capacity	Average Contents	Current Contents	Maximum Contents
1	0.413			724	49.247	AVAIL	100.0	2	0.825	1	2

Queue	Maximum Contents	Average Contents	Total Entries	Zero Entries	Percent Zeros	Average Time/Unit	$Average Time/Unit	Qtable Number	Current Contents
1	6	0.177	724	558	77.1	10.562	46.066		0
2	2	0.082	209	139	66.5	16.878	50.392		0

Random Stream	Antithetic Variates	Initial Position	Current Position	Sample Count	Chi-Square Uniformity
1	OFF	100000	100725	725	0.83
2	OFF	200000	200724	724	0.67
3	OFF	300000	300210	210	0.89
4	OFF	400000	400724	724	0.83
5	OFF	500000	500426	426	0.70

```
Status of Common Storage

    9224 bytes available
     776 in use
    1656 used (max)

Simulation complete.  Absolute Clock: 43200.0000


Total Block Executions: 7679

Blocks / second:        5607549

Microseconds / Block:   0.18


Elapsed Time Used (Sec)

Pass1:           0.00
Sym/Xref         0.00
Pass2:           0.00
Load/Ctrl:       0.00
Execution:       0.00
Output:          0.00
---------------------
Total:           0.00
```

Rezultati dobijeni u programskom jeziku C++

```
Vreme simulacije: 43200.000

STATISTIKE REDA CEKANJA - UNIVERZALNI
Broj entiteta koji je usao u red cekanja: 718
Preostali broj entiteta u redu cekanja: 0
Maksimalni broj entiteta u redu: 5
Srednje vreme cekanja u redu: 9.745
Srednji broj entiteta u redu: 0.162
Broj entiteta koji je prosao kroz red bez zadrzavanja: 545
Procenat ulaza bez zadrzavanja: 75.905
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 40.446

STATISTIKE REDA CEKANJA - PAKETSKI
Broj entiteta koji je usao u red cekanja: 212
Preostali broj entiteta u redu cekanja: 1
Maksimalni broj entiteta u redu: 2
Srednje vreme cekanja u redu: 21.823
Srednji broj entiteta u redu: 0.110
Broj entiteta koji je prosao kroz red bez zadrzavanja: 129
Procenat ulaza bez zadrzavanja: 60.849
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 56.155

STATISTIKE RESURSA - UNIVERZALNI
Broj entiteta koji je usao u resurs: 718
Preostali broj entiteta u resursu: 1
Maksimalni broj zauzetih kanala opsluge: 2
Srednje vreme opsluge: 50.098
Srednji broj zauzetih kanala: 0.836
```

Iskoriscenost: 0.418

STATISTIKE RESURSA - PAKETSKI
Broj entiteta koji je usao u resurs: 211
Preostali broj entiteta u resursu: 1
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 155.051
Srednji broj zauzetih kanala: 0.757
Iskoriscenost: 0.757

Uporedne statistike reda čekanja i resursa (skladišta) dobijene simulacionim programom realizovanim u GPSS/H i C++

Uporedne statistike reda čekanja ispred univerzalnih šaltera

Statistike reda čekanja	GPSS/H	C++
Broj entiteta koji je ušao u red čekanja	724	718
Preostali broj entiteta u redu čekanja	0	0
Maksimalni broj entiteta u redu	6	5
Srednje vreme čekanja u redu	10.562	9.744
Srednji broj entiteta u redu	0.177	0.162
Broj entiteta koji je prošao kroz red bez zadržavanja	558	545
Procenat ulaza bez zadržavanja	77.1	75.905
Srednje vreme čekanja u redu (isključ. ent. koji se nisu zadržali)	46.066	40.446

Uporedne statistike reda čekanja ispred paketskog šaltera

Statistike reda čekanja	GPSS/H	C++
Broj entiteta koji je ušao u red čekanja	209	212
Preostali broj entiteta u redu čekanja	0	1
Maksimalni broj entiteta u redu	2	2
Srednje vreme čekanja u redu	16.878	21.823
Srednji broj entiteta u redu	0.082	0.110
Broj entiteta koji je prošao kroz red bez zadržavanja	139	129
Procenat ulaza bez zadržavanja	66.5	60.849
Srednje vreme čekanja u redu (isključ. ent. koji se nisu zadržali)	50.392	56.155

Uporedne statistike resursa - univerzalni šalteri

Statistike resursa (skladišta)	GPSS/H	C++
Broj entiteta koji je ušao u resurs	724	718
Preostali broj entiteta u resursu	1	1
Maksimalni broj zauzetih kanala opsluge	2	2
Srednje vreme opsluge	49.247	50.098
Srednji broj zauzetih kanala	0.825	0.836
Iskorišćenost	0.413	0.418

Uporedne statistike resursa - paketski šalter

Statistike resursa (skladišta)	GPSS/H	C++
Broj entiteta koji je ušao u resurs	209	211
Preostali broj entiteta u resursu	0	1
Maksimalni broj zauzetih kanala opsluge	1	1
Srednje vreme opsluge	154.415	155.051
Srednji broj zauzetih kanala	0.747	0.757
Iskorišćenost	0.747	0.757

Kompajliranje i izvršenje programa

```
cl stat_simu3.cpp /EHsc

stat_simu3
```

Program stat_simu3.cpp

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

const int max_id_num = 20;

class resurs_stat {
  protected:
    // Ukupni, tekuci i maksimalni broj entiteta u resursu
    int broj, tek_broj, max_broj;
    // Ukupno vreme za koje je resur bio zauzet
    double ukupno_vreme;
    // Vreme za koje se entitet zadrzava u resursu
    double povrsina_cekanja;
    // Poslednji trenutak promene stanja entiteta u resurs
    double vreme_promene;
    // Ukupno vreme simulacije
    double vreme_sim;
    // Polje rednih brojeva entiteta koji se nalaze u resursu
    int id[max_id_num];
    // Vremenski trenutak promene stanja resursa od strane entiteta
    double vreme[max_id_num];
    // Broj entiteta u polju id i polju vreme
    int broj_id;
  public:
    resurs_stat(): broj(0),tek_broj(0),max_broj(0),
                   ukupno_vreme(0),povrsina_cekanja(0),
                   vreme_promene(0), broj_id(0) {}
    void ulaz(int idv,double vremev) {
      if(idv!=-1) {
        // Uvecamo broj entiteta koji su usli resurs
        broj++;
```

```
      //
      povrsina_cekanja += tek_broj*(vremev-vreme_promene);
      // Pamtimo trenutak poslednje promene broja zauzetih mesta
      vreme_promene = vremev;
      //  Uvecavamo tekuci broj zauzetih mesta u salteru
      tek_broj++;
      // Ukoliko je tekuci broj zauzetih mesta veci od maksimalnog
      // tekuci broj postaje maksimalni broj
      if(tek_broj>max_broj)
        max_broj = tek_broj;
      // Pamtimo redni broj entiteta u resursu i trenutak ulaska entiteta u resurs
      dodaj_id(idv,vremev);
    }
    else
      povrsina_cekanja += tek_broj*(vremev-vreme_promene);
  }
  void izlaz(int idv,double vremev) {
    // Nalazimo poziciju entiteta u resursu
    int n = nadji_id(idv);
    // Ukupno vreme koje su entiteti proveli u salteru
    ukupno_vreme += vremev-vreme[n];
    //
    povrsina_cekanja += tek_broj*(vremev-vreme_promene);
    // Pamtimo vreme ulaska entiteta u resurs
    vreme_promene = vremev;
    // Uklanjamo redni broj entiteta sa pozicije n
    ukloni_id(n);
    //  Umanjujemo tekuci broj entiteta u resursu
    tek_broj--;
  }
protected:
  // Dodajemo redni broj entiteta u polje id
  void dodaj_id(int novi_id,double novo_vreme) {
    if(broj_id<max_id_num) {
      id[broj_id] = novi_id;
      vreme[broj_id] = novo_vreme;
      broj_id++;
```

```
      }
      else {
        cerr << "Greska dodaj id" << endl;
        exit(1);
      }
    }
    // Nalazimo redni broj entiteta u polju id
    int nadji_id(int trazi_id) {
      int i;
      for(i=0; i<broj_id; i++)
        if(id[i]==trazi_id)
          break;
      return i;
    }
    // Uklanjamo redni broj i tekuce vreme entiteta iz polja id i vreme sa pozicije pos
    void ukloni_id(int pos) {
      for(int i=pos+1; i<broj_id; i++) {
        id[i-1] = id[i];
        vreme[i-1] = vreme[i];
      }
      broj_id--;
    }
  public:
    void vreme_simulacije(double vs) { vreme_sim = vs; }
    inline int ukupno_uslih_entiteta() { return broj; }
    inline int tekuci_broj_entiteta() { return tek_broj; }
    inline int maksimalni_broj_entiteta() { return max_broj; }
    inline double srednje_vreme() {
      return ukupno_vreme/(broj-tek_broj);
    }
    inline double srednji_broj_entiteta() {
      return povrsina_cekanja/vreme_sim;
    }
    inline double iskoriscenost(int broj_kanala) {
      return srednji_broj_entiteta()/broj_kanala;
    }
};
```

```
// Klasa statistika reda cekanja
class fifo_red_stat: public resurs_stat {
  public:
    // Broj klijenata koji nisu cekali
    int broj_bezcekanja;
  public:
    fifo_red_stat():resurs_stat(),broj_bezcekanja(0) {}
    void izlaz(int idv,double vremev) {
      // Ukoliko je vreme koje je entitet proveo u redu 0
      if(vremev-vreme_promene==0.0)
        // Uvecavamo broj entiteta koji nisu cekali u redu
        broj_bezcekanja++;
      resurs_stat::izlaz(idv,vremev);
    }
  public:
    inline int broj_entiteta_bez_zadrzavanja() { return broj_bezcekanja; }
    inline double procenat_entiteta_bez_zadrzavanja() {
      return (100.0*broj_bezcekanja)/broj;
    }
    inline double srednji_broj_entiteta_bez_zadrzavanja() {
      return ukupno_vreme/(broj-tek_broj-broj_bezcekanja);
    }
};
void print_statistike_reda_cekanja(fifo_red_stat& red,const char* opis) {
  // Statistike reda cekanja
  cout << "STATISTIKE REDA CEKANJA - " << opis << endl;
  cout << "Broj entiteta koji je usao u red cekanja: " << red.ukupno_uslih_entiteta() << endl;
  cout << "Preostali broj entiteta u redu cekanja: " << red.tekuci_broj_entiteta() << endl;
  cout << "Maksimalni broj entiteta u redu: " << red.maksimalni_broj_entiteta() << endl;
  cout << "Srednje vreme cekanja u redu: " << red.srednje_vreme() << endl;
  cout << "Srednji broj entiteta u redu: " << red.srednji_broj_entiteta() << endl;
  cout << "Broj entiteta koji je prosao kroz red bez zadrzavanja: " <<
                                        red.broj_entiteta_bez_zadrzavanja() << endl;
  cout << "Procenat ulaza bez zadrzavanja: " <<
                                        red.procenat_entiteta_bez_zadrzavanja() << endl;
  cout << "Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): " <<
                                        red.srednji_broj_entiteta_bez_zadrzavanja() << endl;
```

```
  cout << endl;
}
void print_statistike_resursa(resurs_stat& salt,int n,const char* opis) {
  // Statistike resursa
  cout << "STATISTIKE RESURSA - " << opis << endl;
  cout << "Broj entiteta koji je usao u resurs: " << salt.ukupno_uslih_entiteta() << endl;
  cout << "Preostali broj entiteta u resursu: " << salt.tekuci_broj_entiteta() << endl;
  cout << "Maksimalni broj zauzetih kanala opsluge: " <<
                                          salt.maksimalni_broj_entiteta() << endl;
  cout << "Srednje vreme opsluge: " << salt.srednje_vreme() << endl;
  cout << "Srednji broj zauzetih kanala: " << salt.srednji_broj_entiteta() << endl;
  cout << "Iskoriscenost: " << salt.iskoriscenost(n) << endl;
  cout << endl;
}
int main() {
  int pos,posn,idv;
  string str,tip,id,vreme;
  char line[1024];
  resurs_stat salt1, salt2;
  fifo_red_stat red1, red2;
  double vremev, vreme_sim;
  // Ucitavamo izvestaj
  ifstream in("simu3.txt",ifstream::in);
  // Ucitavamo podatke iz izvestaja sve dok ne dodjemo do kraja datoteke
  while(in.good()) {
    // Ucitavamo liniju iz izvestaja
    in.getline(line,1024);
    // Dodlejujemo liniju stringu
    str = line;
    // Izdvajamo tip obavestenja
    posn = str.find_first_of(',');
    tip = str.substr(0,posn);
    pos = posn;
    // Izdvajamo vrednost rednog broja entiteta
    posn = str.find_first_of(',',pos+1);
    id = str.substr(pos+1,posn-pos-1);
    pos = posn;
```

```
  // Izdvajamo vremenski trenutak
  vreme = str.substr(pos+1);
  // Prevodimo id i vremenski trenutak u broj
  idv = atoi(id.c_str());
  vremev = atof(vreme.c_str());
  // Proveravamo tip obavestenja
  if(!tip.compare("ured_uni"))
    red1.ulaz(idv,vremev);
  else if(!tip.compare("izred_uni"))
    red1.izlaz(idv,vremev);
  else if(!tip.compare("ured_pak"))
    red2.ulaz(idv,vremev);
  else if(!tip.compare("izred_pak"))
    red2.izlaz(idv,vremev);
  else if(!tip.compare("usal_uni"))
    salt1.ulaz(idv,vremev);
  else if(!tip.compare("izsal_uni"))
    salt1.izlaz(idv,vremev);
  else if(!tip.compare("usal_pak"))
    salt2.ulaz(idv,vremev);
  else if(!tip.compare("izsal_pak"))
    salt2.izlaz(idv,vremev);
  else if(!tip.compare("vsim")) {
    //  Pamtimo vreme simulacije
    vreme_sim = vremev;
    // Postavljamo vreme simulacije
    red1.vreme_simulacije(vremev);
    red2.vreme_simulacije(vremev);
    salt1.vreme_simulacije(vremev);
    salt2.vreme_simulacije(vremev);
    //
    red1.ulaz(idv,vremev);
    red2.ulaz(idv,vremev);
    salt1.ulaz(idv,vremev);
    salt2.ulaz(idv,vremev);
  }
}
```

```
  // Zatvaramo datoteku
  in.close();
  // Podesavamo preciznost izlaznih rezulata
  cout.precision(3);
  cout.setf(ios_base::fixed);
  // ISPISUJEMO STATISTIKE
  // Ispisujemo vreme simulacije
  cout << "Vreme simulacije: " << vreme_sim << endl;
  cout << endl;
  // Statistike reda cekanja
  print_statistike_reda_cekanja(red1,"UNIVERZALNI");
  print_statistike_reda_cekanja(red2,"PAKETSKI");
  // Statistike resursa
  print_statistike_resursa(salt1,2,"UNIVERZALNI");
  print_statistike_resursa(salt2,1,"PAKETSKI");
  return 0;
}
```