Osnovne akademske studije

PREDMET: Objektno-orijentisana simulacija

TEMA: Strategija raspoređivanja događaja 3

Predmetni nastavnik: Doc. dr Marko Đogatović

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

Na paketske šaltere korisnici 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 100 s i standardnom devijacijom od 40 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 šaltera i redova čekanja.

Bezuslovni događaji – Primer 3

procedure *DolazaKorisnikaUUniverzalniSalter*
begin
    Postavi korisnika u red čekanja.
    if *šalter je raspoloživ* **then**
    begin
        Izvadi prvog korisnika iz reda čekanja.
        Zauzmi šalter.
        Izbor vrste usluge i dodeljivanje vremena opsluge.
        Rasporedi korisnika na događaj OdlazakKorisnikaSaUniverzalnogSaltera.
    end
    Napravi novog korisnika.
    Rasporedi korisnika na događaj DolazakKorisnikaUUniverzalniSalter.
end

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

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

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

Izvorna datoteka zad3.cpp

```
// Datototeka zad3.cpp
#include "aes.h"
#include "aes proc.h"
#include "aes dist.h"
#include <iostream>

// Ukljucujemo prostore imena esa i std
using namespace aes;
using namespace std;

// Redovi cekanja
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   = 100;
// Standardna devijacija vremena opsluge entiteta u paketskom salteru
const double STDEVOPSPAK  = 40;
// Minimalno vreme opsluge entiteta u paketskom salteru
const double MINOPSPAK    = 20;

// Klasa korisnika
class korisnik:public agent {
  // Brojac korisnika
  static size_t _idc;
  // Redni broj korisnika
  size_t _id;
public:
```

```
  // Konstruktor
  korisnik():agent() { _id = ++_idc; }
public:
  // Vraca id korisnika
  size_t vrati_id() { return _id; }
};
size_t korisnik::_idc = 0;

// Klasa dogadjaja dolazak korisnika na univerzalne saltere
class dolazak_uni_korisnika:public dogadjaj {
public:
  // Konstruktor
  dolazak_uni_korisnika():dogadjaj(1) {}
  // Akcija
  void akcija();
  // Pomocna funkcija za pravljenje dogadjaja
  static dolazak_uni_korisnika* napravi(korisnik* kor,vreme vn) {
    dolazak_uni_korisnika* dk = new dolazak_uni_korisnika;
    dk->postavi_agenta(0,kor);
    dk->postavi_vreme(simu.vrati_vreme()+vn);
    return dk;
  }
};
// Klasa dogadjaja odlazak korisnika sa univerzalnih saltera
class odlazak_uni_korisnika:public dogadjaj {
public:
  // Konstruktor
  odlazak_uni_korisnika():dogadjaj(1) {}
  // Akcija dogadjaja
  void akcija();
  // Pomocna funkcija za pravljenje dogadjaja
  static odlazak_uni_korisnika* napravi(korisnik* kor,vreme vn) {
    odlazak_uni_korisnika* ok = new odlazak_uni_korisnika;
    ok->postavi_agenta(0,kor);
    ok->postavi_vreme(simu.vrati_vreme()+vn);
    return ok;
  }
};

// Klasa dogadjaja dolazak korisnika na paketski salter
class dolazak_pak_korisnika:public dogadjaj {
public:
  // Konstruktor
  dolazak_pak_korisnika():dogadjaj(1) {}
  // Akcija
```

```
  void akcija();
  // Pomocna funkcija za pravljenje dogadjaja
  static dolazak_pak_korisnika* napravi(korisnik* kor,vreme vn) {
    dolazak_pak_korisnika* dk = new dolazak_pak_korisnika;
    dk->postavi_agenta(0,kor);
    dk->postavi_vreme(simu.vrati_vreme()+vn);
    return dk;
  }
};
// Klasa dogadjaja odlazak korisnika sa paketskog saltera
class odlazak_pak_korisnika:public dogadjaj {
public:
  // Konstruktor
  odlazak_pak_korisnika():dogadjaj(1) {}
  // Akcija dogadjaja
  void akcija();
  // Pomocna funkcija za pravljenje dogadjaja
  static odlazak_pak_korisnika* napravi(korisnik* kor,vreme vn) {
    odlazak_pak_korisnika* ok = new odlazak_pak_korisnika;
    ok->postavi_agenta(0,kor);
    ok->postavi_vreme(simu.vrati_vreme()+vn);
    return ok;
  }
};

// Klasa dogadjaja zavrsetka simulacije
class kraj: public dogadjaj {
public:
  // Konstruktor dogadjaja kraj
  kraj():dogadjaj(0) {}
public:
  // Akcija dogadjaja
  void akcija() { simu.zaustavi(); }
  // Pomocna funkcija za pravljenje dogadjaja
  static kraj* napravi(vreme vn) {
    kraj* k = new kraj;
    k->postavi_vreme(simu.vrati_vreme()+vn);
    return k;
  }
};

// Akcija
void dolazak_uni_korisnika::akcija() {
  korisnik *kor,*prvi_kor,*novi_kor;
  double vi, vreme_opsluge;
```

```
  // Korisnik koji dolazi u postu
  kor = dynamic_cast<korisnik*>(vrati_agenta(0));
  #ifdef STAMPA
  cout << "ured_uni" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
  #endif
  // Postavi korisnika u red cekanja
  red1.smesti(kor);
  // Ukoliko su univerzalni salteri raspolozivi
  if(uni_salt.raspoloziv()) {
    // Izvadi prvog klijenta iz reda
    prvi_kor = dynamic_cast<korisnik*>(red1.prednji());
    // Prvi korisnik izlazi iz reda
    red1.vadi();
    #ifdef STAMPA
    cout << "izred_uni" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    // Zauzmi jedno mesto u salteru
    uni_salt.zauzmi();
    #ifdef STAMPA
    cout << "usal_uni" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    // Verovatnoca izbora usluge
    vi = rni();
    // Vreme opsluge na osnovu izabrane vrste usluge
    if(vi<0.20)
      vreme_opsluge = rn2.expo(SRVROPSUNI[0]);
    else if(vi<0.60)
      vreme_opsluge = rn2.expo(SRVROPSUNI[1]);
    else if(vi<0.75)
      vreme_opsluge = rn2.expo(SRVROPSUNI[2]);
    else if(vi<1.00)
      vreme_opsluge = rn2.expo(SRVROPSUNI[3]);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(odlazak_uni_korisnika::napravi(prvi_kor,vreme_opsluge));
  }
  // Stvaramo narednog korisnika
  novi_kor = new korisnik;
  // Postavljamo vreme narednog dolaska
  simu.rasporedi(dolazak_uni_korisnika::napravi(novi_kor,rn1.expo(SRVRDOLUNI)));
}

// Akcija dogadjaja
void odlazak_uni_korisnika::akcija() {
  korisnik *kor, *prvi_kor;
  double vi, vreme_opsluge;
```

```
  // Korisnik koji odlazi
  kor = dynamic_cast<korisnik*>(vrati_agenta(0));
  // Vrati salter
  uni_salt.oslobodi();
  #ifdef STAMPA
  cout << "izsal_uni" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
  #endif
  // Korisnik odlazi iz poste - unistavamo tekuceg korisnik
  delete kor;
  // Ukoliko red nije prazan
  if(red1.vrati_velicinu()>0) {
    // Izvadi prvog korisnika iz reda
    prvi_kor = dynamic_cast<korisnik*>(red1.prednji());
    red1.vadi();
    #ifdef STAMPA
    cout << "izred_uni" << "," << prvi_kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    // Zauzmi jedno mesto u salteru
    uni_salt.zauzmi();
    #ifdef STAMPA
    cout << "usal_uni" << "," << prvi_kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    // Verovatnoca izbora usluge
    vi = rni();
    // Vreme opsluge na osnovu izabrane vrste usluge
    if(vi<0.20)
      vreme_opsluge = rn2.expo(SRVROPSUNI[0]);
    else if(vi<0.60)
      vreme_opsluge = rn2.expo(SRVROPSUNI[1]);
    else if(vi<0.75)
      vreme_opsluge = rn2.expo(SRVROPSUNI[2]);
    else if(vi<1.00)
      vreme_opsluge = rn2.expo(SRVROPSUNI[3]);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(odlazak_uni_korisnika::napravi(prvi_kor,vreme_opsluge));
  }
}

// Akcija
void dolazak_pak_korisnika::akcija() {
  korisnik *kor,*prvi_kor,*novi_kor;
  double vreme_opsluge;
  // Korisnik koji dolazi u postu
  kor = dynamic_cast<korisnik*>(vrati_agenta(0));
  #ifdef STAMPA
```

```
  cout << "ured_pak" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
  #endif
  // Postavi korisnika u red cekanja
  red2.smesti(kor);
  // Ukoliko je salter raspoloziv
  if(pak_salt.raspoloziv()) {
    // Izvadi prvog korisnika iz reda
    prvi_kor = dynamic_cast<korisnik*>(red2.prednji());
    // Prvi korisnik izlazi iz reda
    red2.vadi();
    #ifdef STAMPA
    cout << "izred_pak" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    // Zauzmi jedno mesto u salteru
    pak_salt.zauzmi();
    #ifdef STAMPA
    cout << "usal_pak" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    do {
      vreme_opsluge = rn4.norm(SRVROPSPAK,STDEVOPSPAK);
    } while(vreme_opsluge<MINOPSPAK);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(odlazak_pak_korisnika::napravi(prvi_kor,vreme_opsluge));
  }
  // Stvaramo narednog korisnika
  novi_kor = new korisnik;
  // Postavljamo vreme narednog dolaska
  simu.rasporedi(dolazak_pak_korisnika::napravi(novi_kor,rn3.unif(SRVRDOLPAK-MODDOLPAK,SRVRDOLPAK+MODDOLPAK)));
}

// Akcija
void odlazak_pak_korisnika::akcija() {
  korisnik *kor, *prvi_kor;
  double vreme_opsluge;
  // Korisnik koji odlazi
  kor = dynamic_cast<korisnik*>(vrati_agenta(0));
  // Vrati salter
  pak_salt.oslobodi();
  #ifdef STAMPA
  cout << "izsal_pak" << "," << kor->vrati_id() << "," << simu.vrati_vreme() << endl;
  #endif
  // Klijent odlazi iz poste - unistavamo tekuceg klijenta
  delete kor;
  // Ukoliko red nije prazan
  if(red2.vrati_velicinu()>0) {
```

```
    // Izvadi prvog korisnika iz reda
    prvi_kor = dynamic_cast<korisnik*>(red2.prednji());
    red2.vadi();
    #ifdef STAMPA
    cout << "izred_pak" << "," << prvi_kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    // Zauzmi jedno mesto u salteru
    pak_salt.zauzmi();
    #ifdef STAMPA
    cout << "usal_pak" << "," << prvi_kor->vrati_id() << "," << simu.vrati_vreme() << endl;
    #endif
    do {
      vreme_opsluge = rn4.norm(SRVROPSPAK,STDEVOPSPAK);
    } while(vreme_opsluge<MINOPSPAK);
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(odlazak_pak_korisnika::napravi(prvi_kor,vreme_opsluge));
  }
}

int main() {
  // Stvaramo prvog korisnika
  korisnik *kor1 = new korisnik, *kor2 = new korisnik;
  // Postavljamo prvi dogadja u listu sto je odgovara dolasku prvog klijenta
  simu.rasporedi(dolazak_uni_korisnika::napravi(kor1,rn1.expo(SRVRDOLUNI)));
  // Postavljamo prvi dogadja u listu sto je odgovara dolasku prvog klijenta
  simu.rasporedi(dolazak_pak_korisnika::napravi(kor2,rn3.unif(SRVRDOLPAK-MODDOLPAK,SRVRDOLPAK+MODDOLPAK)));
  // Rasporedi dogadjaj zavrsetka simulacije
  simu.rasporedi(kraj::napravi(43200));
  // Izvrsavamo simulaciju
  simu.izvrsi();
  #ifdef STAMPA
  // Stampamo vreme simulacije
  cout << "vsim" << "," << -1 << "," << simu.vrati_vreme() << endl;
  #endif
  return 0;
}
```

Listing simulacionog programa realizovanog u GPSS/H

```
    Student GPSS/H Release 3.70 (AY015)     21 Jan 2018   22:11:53     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(5,100,40)
       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	215
4	724	14	215
5	724	15	209
6	1 724	16	209
7	723	17	209
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	0.501			209	103.617	AVAIL			

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	1	0.005	209	200	95.7	0.959	22.280		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	500430	430	0.73

Status of Common Storage

9336 bytes available
664 in use
1544 used (max)

Simulation complete. Absolute Clock: 43200.0000

```
Total Block Executions: 7685

Blocks / second:        5846596

Microseconds / Block:   0.17


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: 6
Srednje vreme cekanja u redu: 12.496
Srednji broj entiteta u redu: 0.324
Broj entiteta koji je prosao kroz red bez zadrzavanja: 546
Procenat ulaza bez zadrzavanja: 76.045
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 52.165

STATISTIKE REDA CEKANJA - PAKETSKI
Broj entiteta koji je usao u red cekanja: 213
Preostali broj entiteta u redu cekanja: 0
Maksimalni broj entiteta u redu: 1
Srednje vreme cekanja u redu: 0.442
Srednji broj entiteta u redu: 0.002
Broj entiteta koji je prosao kroz red bez zadrzavanja: 206
Procenat ulaza bez zadrzavanja: 96.714
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 13.463

STATISTIKE RESURSA - UNIVERZALNI
Broj entiteta koji je usao u resurs: 718
Preostali broj entiteta u resursu: 2
Maksimalni broj zauzetih kanala opsluge: 2
Srednje vreme opsluge: 48.686
Srednji broj zauzetih kanala: 0.808
Iskoriscenost: 0.404

STATISTIKE RESURSA - PAKETSKI
Broj entiteta koji je usao u resurs: 213
Preostali broj entiteta u resursu: 0
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 101.475
Srednji broj zauzetih kanala: 0.500
Iskoriscenost: 0.500
```

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	6
Srednje vreme čekanja u redu	10.562	12.496
Srednji broj entiteta u redu	0.177	0.324
Broj entiteta koji je prošao kroz red bez zadržavanja	558	546
Procenat ulaza bez zadržavanja	77.1	76.045
Srednje vreme čekanja u redu (isključ. ent. koji se nisu zadržali)	46.066	52.165

Uporedne statistike reda čekanja ispred paketskog šaltera

Statistike reda čekanja	GPSS/H	C++
Broj entiteta koji je ušao u red čekanja	209	213
Preostali broj entiteta u redu čekanja	0	0
Maksimalni broj entiteta u redu	1	1
Srednje vreme čekanja u redu	0.959	0.442
Srednji broj entiteta u redu	0.005	0.002
Broj entiteta koji je prošao kroz red bez zadržavanja	200	206
Procenat ulaza bez zadržavanja	95.7	96.714
Srednje vreme čekanja u redu (isključ. ent.koji se nisu zadržali)	22.280	13.463

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	2
Maksimalni broj zauzetih kanala opsluge	2	2
Srednje vreme opsluge	49.247	48.686
Srednji broj zauzetih kanala	0.825	0.808
Iskorišćenost	0.413	0.404

Uporedne statistike resursa - paketski šalter

Statistike resursa (skladišta)	GPSS/H	C++
Broj entiteta koji je ušao u resurs	209	213
Preostali broj entiteta u resursu	0	0
Maksimalni broj zauzetih kanala opsluge	1	1
Srednje vreme opsluge	103.617	101.475
Srednji broj zauzetih kanala	0.501	0.500
Iskorišćenost	0.501	0.500

Kompajliranje i izvršenje programa

Microsoft Visual C++

```
cl zad3.cpp /DSTAMPA -O2 /EHsc
```

```
cl stat_zad3.cpp /EHsc
```

MingW g++

```
g++ zad3.cpp -DSTAMPA -O3 -o zad3.exe -static
```

```
g++ stat_zad3.cpp -o stat_zad3.exe -static
```

Izvršenje

```
zad3 > out.txt
```

```
stat_zad3 < out.txt
```

Program stat_zad3.cpp

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

const int max_id_num = 20;

class resurs_stat {
  public:
    // 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];
  private:
    // 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) {}
    // 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--;
    }
    void vreme_simulacije(double vs) { vreme_sim = vs; }
    double srednje_vreme() {
      return ukupno_vreme/(broj-tek_broj);
    }
    double srednji_broj_entiteta() {
      return povrsina_cekanja/vreme_sim;
    }
    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) {}
    double procenat_entiteta_bez_zadrzavanja() {
      return (100.0*broj_bezcekanja)/broj;
    }
    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.broj << endl;
  cout << "Preostali broj entiteta u redu cekanja: " << red.tek_broj << endl;
  cout << "Maksimalni broj entiteta u redu: " << red.max_broj << 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_bezcekanja << 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.broj << endl;
  cout << "Preostali broj entiteta u resursu: " << salt.tek_broj << endl;
  cout << "Maksimalni broj zauzetih kanala opsluge: " << salt.max_broj << 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,
         vreme_u_redu, vreme_u_res;
  // Ucitavamo podatke iz izvestaja sve dok ne dodjemo do kraja datoteke
  while(cin.good()) {
    // Ucitavamo liniju iz izvestaja
    cin.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")) {
  // Broj entiteta koji su usli u red cekanja
  red1.broj++;
  // Uvecavamo ukupno cekanje u redu
  red1.povrsina_cekanja += red1.tek_broj*(vremev-red1.vreme_promene);
  // Pamtimo trenutak dolaska entiteta u red
  red1.vreme_promene = vremev;
  //  Uvecavamo broj entiteta u redu
  red1.tek_broj++;
  // Ukoliko je tekuci broj klijenata u redu  veci od maksimalnog
  // tekuci broj postaje maksimalni broj
  if(red1.tek_broj>red1.max_broj)
    red1.max_broj = red1.tek_broj;
  // Pamtimo redni broj i trenutak ulaska entiteta u redu
  red1.dodaj_id(idv,vremev);
}
else if(!tip.compare("izred_uni")) {
  // Nalazimo poziciju entiteta u redu
  int n = red1.nadji_id(idv);
  // Izracunavamo vreme cekanja u redu
  vreme_u_redu = vremev-red1.vreme[n];
  // Ukupno vreme koje su entiteti proveli u redu
  red1.ukupno_vreme += vreme_u_redu;
  //
  red1.povrsina_cekanja += red1.tek_broj*(vremev-red1.vreme_promene);
  // Ukoliko je vreme koje je entitet proveo u redu 0
  if(vreme_u_redu==0.0)
    // Uvecavamo broj entiteta koji nisu cekali u redu
    red1.broj_bezcekanja++;
  // Uklanjamo redni broj entiteta sa pozicije n
  red1.ukloni_id(n);
  //  Umanjujemo tekuci broj entiteta u redu
  red1.tek_broj--;
}
else if(!tip.compare("ured_pak")) {
  // Broj entiteta koji su usli u red cekanja
  red2.broj++;
  // Uvecavamo ukupno cekanje u redu
  red2.povrsina_cekanja += red2.tek_broj*(vremev-red2.vreme_promene);
```

```
  // Pamtimo trenutak dolaska entiteta u red
  red2.vreme_promene = vremev;
  //  Uvecavamo broj entiteta u redu
  red2.tek_broj++;
  // Ukoliko je tekuci broj klijenata u redu  veci od maksimalnog
  // tekuci broj postaje maksimalni broj
  if(red2.tek_broj>red2.max_broj)
    red2.max_broj = red2.tek_broj;
  // Pamtimo redni broj i trenutak ulaska entiteta u redu
  red2.dodaj_id(idv,vremev);
}
else if(!tip.compare("izred_pak")) {
  // Nalazimo poziciju entiteta u redu
  int n = red2.nadji_id(idv);
  // Izracunavamo vreme cekanja u redu
  vreme_u_redu = vremev-red2.vreme[n];
  // Ukupno vreme koje su entiteti proveli u redu
  red2.ukupno_vreme += vreme_u_redu;
  //
  red2.povrsina_cekanja += red2.tek_broj*(vremev-red2.vreme_promene);
  // Ukoliko je vreme koje je entitet proveo u redu 0
  if(vreme_u_redu==0.0)
    // Uvecavamo broj entiteta koji nisu cekali u redu
    red2.broj_bezcekanja++;
  // Uklanjamo redni broj entiteta sa pozicije n
  red2.ukloni_id(n);
  //  Umanjujemo tekuci broj entiteta u redu
  red2.tek_broj--;
}
else if(!tip.compare("usal_uni")){
  // Uvecamo broj entiteta koji su usli resurs
  salt1.broj++;
  //
  salt1.povrsina_cekanja += salt1.tek_broj*(vremev-salt1.vreme_promene);
  // Pamtimo trenutak poslednje promene broja zauzetih mesta
  salt1.vreme_promene = vremev;
  //  Uvecavamo tekuci broj zauzetih mesta u salteru
  salt1.tek_broj++;
  // Ukoliko je tekuci broj zauzetih mesta veci od maksimalnog
  // tekuci broj postaje maksimalni broj
  if(salt1.tek_broj>salt1.max_broj)
    salt1.max_broj = salt1.tek_broj;
  // Pamtimo redni broj entiteta u resursu i trenutak ulaska entiteta u resurs
  salt1.dodaj_id(idv,vremev);
}
```

```
else if(!tip.compare("izsal_uni")) {
  // Nalazimo poziciju entiteta u resursu
  int n = salt1.nadji_id(idv);
  // Vreme koje entitet provodi u resursu
  vreme_u_res = vremev-salt1.vreme[n];
  // Ukupno vreme koje su entiteti proveli u salteru
  salt1.ukupno_vreme += vreme_u_res;
  //
  salt1.povrsina_cekanja += salt1.tek_broj*(vremev-salt1.vreme_promene);
  // Pamtimo vreme ulaska entiteta u resurs
  salt1.vreme_promene = vremev;
  // Uklanjamo redni broj entiteta sa pozicije n
  salt1.ukloni_id(n);
  //  Umanjujemo tekuci broj entiteta u resursu
  salt1.tek_broj--;
}
else if(!tip.compare("usal_pak")) {
  // Uvecamo broj entiteta koji su usli resurs
  salt2.broj++;
  //
  salt2.povrsina_cekanja += salt2.tek_broj*(vremev-salt2.vreme_promene);
  // Pamtimo trenutak poslednje promene broja zauzetih mesta
  salt2.vreme_promene = vremev;
  //  Uvecavamo tekuci broj zauzetih mesta u salteru
  salt2.tek_broj++;
  // Ukoliko je tekuci broj zauzetih mesta veci od maksimalnog
  // tekuci broj postaje maksimalni broj
  if(salt2.tek_broj>salt2.max_broj)
    salt2.max_broj = salt2.tek_broj;
  // Pamtimo redni broj entiteta u resursu i trenutak ulaska entiteta u resurs
  salt2.dodaj_id(idv,vremev);
}
else if(!tip.compare("izsal_pak")) {
  // Nalazimo poziciju entiteta u resursu
  int n = salt2.nadji_id(idv);
  // Vreme koje entitet provodi u resursu
  vreme_u_res = vremev-salt2.vreme[n];
  // Ukupno vreme koje su entiteti proveli u salteru
  salt2.ukupno_vreme += vreme_u_res;
  //
  salt2.povrsina_cekanja += salt2.tek_broj*(vremev-salt2.vreme_promene);
  // Pamtimo vreme ulaska entiteta u resurs
  salt2.vreme_promene = vremev;
  // Uklanjamo redni broj entiteta sa pozicije n
  salt2.ukloni_id(n);
```

```
      //  Umanjujemo tekuci broj entiteta u resursu
      salt2.tek_broj--;
    }
    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.dodaj_id(idv,vremev);
      red2.dodaj_id(idv,vremev);
      salt1.dodaj_id(idv,vremev);
      salt2.dodaj_id(idv,vremev);
      //
      salt1.povrsina_cekanja += salt1.tek_broj*(vremev-salt1.vreme_promene);
      salt2.povrsina_cekanja += salt2.tek_broj*(vremev-salt2.vreme_promene);
      red1.povrsina_cekanja += red1.tek_broj*(vremev-red1.vreme_promene);
      red2.povrsina_cekanja += red2.tek_broj*(vremev-red2.vreme_promene);
    }
  }
  // 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;
}
```