Osnovne akademske studije

PREDMET: Objektno-orijentisana simulacija

TEMA: Strategija raspoređivanja događaja 4

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

Primer 4. U pošti postoje dva univerzalna šaltera i jedan paketski šalter.Na univerzalne šaltere klijenti dolaze po puasonovom toku sa srednjim vremenom koje se menja u zavisnosti od vremena u toku dana, a prema tabeli 1.

Tabela 1.

Vreme u toku dana	Srednje vreme između dolazaka (s)
prvih 5 sati	50
naredna 4 sata	35
naredna 2 sata	25

Na paketske šaltere klijenti dolaze po erlangovom zakonu sa srednjim vremenom od 240 sek.

Univerzalni šalteri imaju nezavisne redove čekanja. Korisnici biraju prvi slobodan šalter ili kraći red čekanja. Vreme opsluge je eksponecijalno raspodeljeno sa srednjim vremenom koje zavisi od vrste usluge, a prema tabeli 2.

Tabela 2.

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. Paketski šalter ne radi prvi i poslednji sat radnog vremena.

U zadatku je potrebno napraviti histograme:

1. vremena čekanja na univerzalnim šalterima,
2. dužinu reda čekanja ispred paketskog šaltera.

Simulirati rad pošte u toku dva jedanestočasovna radna 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 prikazati statistike redova čekanja i resursa za svaki radni dan.

Bezuslovni događaji – Primer 4

```
procedure DolazakKlijentaUUniverzalniSalter
begin
    if šalter 1 je raspoloživ then
        Postavi kljenta u red čekanja 1.
        if šalter 1 je raspoloživ then
        begin
            Izvadi prvog klijenta iz reda čekanja 1.
            Zauzmi šalter 1.
            Izbor vrste usluge i dodeljivanje vremena opsluge.
            Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera1.
        end
    else if šalter 2 je raspoloživ then
        Postavi kljenta u red čekanja 2.
        if šalter 2 je raspoloživ then
        begin
```

```
            Izvadi prvog klijenta iz reda čekanja 2.
            Zauzmi šalter 2.
            Izbor vrste usluge i dodeljivanje vremena opsluge.
            Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera2.
        end
else if red čekanja 1 < red čekanja 2 then
        Postavi kljenta u red čekanja 1.
        if šalter 1 je raspoloživ then
        begin
            Izvadi prvog klijenta iz reda čekanja 1.
            Zauzmi šalter 1.
            Izbor vrste usluge i dodeljivanje vremena opsluge.
            Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera1.
        end
else
        Postavi kljenta u red čekanja 2.
        if šalter 2 je raspoloživ then
        begin
            Izvadi prvog klijenta iz reda čekanja 2.
```

```
                Zauzmi šalter 2.
                Izbor vrste usluge i dodeljivanje vremena opsluge.
                Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera2.
            end
        end
        Napravi novog klijenta.
        Rasporedi klijenta na događaj DolazakKlijentaUUniverzalniSalter.
end
```

```
procedure OdlazakKlijentaIzUniverzalnogSaltera1
begin
    Oslobodi šalter 1.
    Uništi klijenta.
    if red čekanja 1 nije prazan then
    begin
        Izvadi prvog klijenta iz reda čekanja 1.
        Zauzmi šalter 1.
        Izbor vrste usluge i dodeljivanje vremena opsluge.
        Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera1.
    end
end
```

```
procedure OdlazakKlijentaIzUniverzalnogSaltera2
begin
    Oslobodi šalter 2.
    Uništi klijenta.
    if red čekanja 2 nije prazan then
    begin
        Izvadi prvog klijenta iz reda čekanja 2.
        Zauzmi šalter 2.
        Izbor vrste usluge i dodeljivanje vremena opsluge.
        Rasporedi klijenta na događaj OdlazakKlijentaIzUniverzalnogSaltera2.
    end
end
```

```
procedure DolazakKlijentaUPaketskiSalter
begin
    if vreme simulacije >= 3600 and vreme simulacije <= 36000 then
        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
    else
        Uništi entitet.
    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
```

```
procedure KrajSimulacije
begin
    Uništi entitet i umanji TB.
end
```

Zaglavlje simul.h

```
// Zaglavlje simul.h
#include <iostream>
#include <cstdlib>
#include <map>
#include <deque>
#include <cmath>
#include <algorithm>
#include <limits>
#include <cstring>

// Koristimo prostor imena STL biblioteke
using namespace std;

// Prostor imena simu4
namespace simul {
  // Klasa raspodela
  class raspodela {
    // Seme generatora slucajnih brojeva (GSB)
    unsigned int seme;
    // Multiplikator GSB
    unsigned int a;
    // Modulus GSB
    unsigned int m;
  public:
    // Konstruktor
    raspodela(unsigned int s = 1) :seme(s) {
      // Postavljamo vrednost multiplikatora
      a = 16807;
      // Postavljamo vrednost modulusa
      m = 2147483647;
    }
```

```
// Lehmerov multiplikativni kongruentni generator
double operator() () {
  seme = (a*seme) % (m + 1);
  return seme / (1.0*m);
}
// Uniformna raspodela
double unif(double a, double b) {
  if (b>a)
    return a + (b - a)*(*this)();
  else {
    cerr << "Gornja granica mora da bude veca od donje" << endl;
    exit(1);
  }
}
// Eksponencijalna raspodela
double expo(double m) {
  if (m>0)
    return -m*log((*this)());
  else {
    cerr << "Ocekivanje mora da bude vece od nule" << endl;
    exit(1);
  }
}
// Normalna raspodela
double norm(double m, double s) {
  double z = 0.0;
  if (s>0) {
    for (int i = 0; i<12; i++)
      z += (*this)();
    return m + (z - 6)*s;
  }
  else {
    cerr << "Devijacija ne sme da bude manja ili jednaka nuli" << endl;
    exit(1);
  }
```

```
  }
};
// Klasa funkcija
struct funkcija {
  // Diskretna raspodela
  static double diskretna(double val, const double x[], const double y[], int n) {
    if(val<x[0]) return y[0];
    if(val>x[n-1]) {
      cerr << "Trazite vrednost funkcije van zadatog intervala" << endl;
      exit(1);
    }
    for(int i=0;i<n-1;++i) {
      if(val>=x[i] && val<x[i+1])
        return y[i+1];
    }
    return 0.0;
  }
  // Kontinualna raspodela
  static double kontinualna(double val, const double x[], const double y[], int n) {
    if (val >= x[0] && val<x[n-1]) {
      for (int i=0;i<n-1;++i)
        if (x[i]==val) return y[i];
      for (int i=0;i<n-1;++i) {
        if (val>=x[i] && val<x[i+1])
          return y[i]+(y[i + 1]-y[i])/(x[i+1]-x[i])*(val-x[i]);
      }
    }
    else {
      cerr << "Trazite vrednost funkcije van zadatog intervala" << endl;
      exit(1);
    }
    return 0.0;
  }
};
class histogram {
```

```
  double ggpi;  // Gornja granica prvog intervala
  double si;    // Sirina intervala
  int bi;       // Broj intervala
  int *frekv;   // Frekvencije
  int frekv_of; // Frekvencija overflowa
  double sumaa; // Suma argumenata
  double sumaao;// Suma argumenata overflow vrednosti
  double srvr;  // Srednja vrednost
  double m2;
  int broj;     // Broj snimljenih vrednosti
public:
  histogram(double ggpi, double si, int bi) :ggpi(ggpi),si(si),bi(bi),broj(0),
                                             sumaa(0.0),srvr(0.0),m2(0.0),sumaao(0.0),frekv_of(0) {
    if (bi<=1) {
      cerr << "Histogram mora da ima 2 ili vise intervala" << endl;
      exit(1);
    }
    if (si<=0) {
      cerr << "Sirina intervala mora da bude pozitivni realni broj" << endl;
      exit(1);
    }
    if (ggpi<0) {
      cerr << "Gornja granica prvog intervala mora da bude pozitivni realni broj" << endl;
      exit(1);
    }
    // Alociramo fekvencije po intervalima
    frekv = new int[bi];
    // Inicijalizujemo frekvencije
    memset(frekv,0,bi*sizeof(int));
  }
  ~histogram() {
    delete[] frekv;
  }
  void operator()(double v) {
    // Brojimo ulaske u histogram
```

```
    broj++;
    // Nalazimo sumu argumenata
    sumaa += v;
    // Online srednja vrednost i devijacija
    double d = v - srvr;
    srvr += d / broj;
    m2 += d*(v-srvr);
    // Uvecavamo frekvencije za
    // prvi interval
    if (v>=0.0&&v<=ggpi) {
      ++frekv[0];
    }
    // overflow
    else if(v>ggpi+(bi-1)*si) {
      ++frekv_of;
      sumaao += v;
    }
    // ostale intervale
    else {
      for (int i=0;i<(bi-1);i++) {
        if (v>ggpi+i*si && v<=ggpi+(i+1)*si) {
          ++frekv[i+1];
          break;
        }
      }
    }
}
inline void postavi_gornju_granicu_prvog_intervala(const double gg) {
    if (gg < 0) {
      cerr << "Gornja granica prvog intervala mora da bude pozitivni realni broj" << endl;
      exit(1);
    }
    ggpi = gg;
}
inline void postavi_sirinu_intervala(const double s) {
```

```
  if (s <= 0) {
    cerr << "Sirina intervala mora da bude pozitivni realni broj" << endl;
    exit(1);
  }
  si = s;
}
inline void postavi_broj_intervala(const int b) {
  if (b<=1) {
    cerr << "Histogram mora da ima 1 ili vise intervala" << endl;
    exit(1);
  }
  bi = b;
  frekv = new int[bi];
  memset(frekv,0,bi*sizeof(int));
}
inline double vrati_sirinu_intervala() const {
  return si;
}
inline int vrati_broj_intervala() const {
  return bi;
}
inline double vrati_gornju_granicu_prvog_intervala() const {
  return ggpi;
}
inline double vrati_broj_ulaza_u_histogram() const {
  return broj;
}
inline double vrati_srednju_vrednost_histograma() const {
  return srvr;
}
inline double vrati_standardnu_devijaciju_histograma() {
  return sqrt(m2/(broj-1));
}
inline double vrati_sumu_argumenata_histograma() const {
  return sumaa;
```

```
}
double vrati_gornju_granicu_intervala(int n) {
  if(n<0||n>bi-1) {
    cerr << "Gornju granicu intervala mozete dobiti za sve intervale" << endl;
    exit(1);
  }
  return ggpi + n*si;
}
double vrati_frekvenciju_intervala(int n) {
  if(n<0||n>bi) {
    cerr << "Frekvenciju intervala mozete dobiti za sve intervale i overflow" << endl;
    exit(1);
  }
  if(n==bi)
    return frekv_of;
  else
    return frekv[n];
}
double vrati_verovatnocu_intervala(int n) {
  if(n<0||n>bi) {
    cerr << "Verovatnocu intervala mozete dobiti za sve intervale i overflow" << endl;
    exit(1);
  }
  if(n==bi)
    return frekv_of/(1.0*broj);
  else
    return frekv[n]/(1.0*broj);
}
double vrati_umnozak_od_srednje_vrednosti_intervala(int n) {
  if(n<0||n>bi-1) {
    cerr << "Umnozak od srednje vrednosti intervala mozete dobiti za sve intervale" << endl;
    exit(1);
  }
  return vrati_gornju_granicu_intervala(n)/vrati_srednju_vrednost_histograma();
}
```

```
    double vrati_devijaciju_od_srednje_vrednosti_intervala(int n) {
      if(n<0||n>bi-1) {
        cerr << "Devijaciju od srednje vrednosti intervala mozete dobiti za sve intervale" << endl;
        exit(1);
      }
      return (vrati_gornju_granicu_intervala(n)-
vrati_srednju_vrednost_histograma())/vrati_standardnu_devijaciju_histograma();
    }
    inline double vrati_prosecnu_vrednost_overflowa() {
      return sumaao/frekv[bi-1];
    }
    void resetuj() {
      memset(frekv,0,bi*sizeof(int));
      frekv_of = 0;
      broj     = 0;
      sumaa    = 0.0;
      sumaao   = 0.0;
      srvr     = 0.0;
      m2       = 0.0;
    }
  };
  // Klasa entiteta
  class entitet {
    // Omogucavamo klasi simulacija da pristupi privatnim clanovima klase entitet
    friend class simulacija;
    // Omogucavamo klasi red_cekanja da pristupi privatnim clanovima klase entitet
    friend class red_cekanja;
    // Omogucavamo klasi resurs da pristupi privatnim clanovima klase entitet
    friend class resurs;
    // Redni broj entiteta
    int id;
    // Buduci dogadjaj
    int naredni_dogadjaj;
    // Vreme nastupanja buduceg dogadjaja
    double vreme_nastupanja_dogadjaja;
```

```
  // Broj parametara entiteta
  int npar;
  // Parametri entiteta
  double *parametri;
public:
  // Konstruktor
  entitet(int eid,int npar=10) :id(eid),npar(npar) {
    // Alociramo parametre
    parametri = new double[npar];
    // Inicijalizujemo parametre entiteta
    memset(parametri,npar,sizeof(double));
  }
  // Destruktor
  ~entitet() {
    // Uklanjamo parametre entiteta iz memorije
    delete [] parametri;
  }
  // Vraca id entiteta
  inline int vrati_id() { return id; }
public:
  // Postavlja i vraca parametar entiteta
  double& operator[](int n) {
    if(n>=0 || n<npar)
      return parametri[n];
    else {
      cerr << "Neodgovarajuci broj parametra (<200)" << endl;
      exit(1);
    }
  }
};
//
class statistike {
private:
  // Ukupni, tekuci i maksimalni broj entiteta u resursu
  int broj, tekuci_broj, maks_broj;
  // Broj klijenata koji nisu cekali
```

```
  int broj_bez_cekanja;
  // Ukupno vreme za koje je resurs bio zauzet
  double ukupno_vreme;
  // Vreme za koje se entitet zadrzava u resursu
  double povrsina;
  // Poslednji trenutak promene stanja entiteta u resurs
  double vremenski_trenutak;
  // Vremenski trenuci promene stanja
  map<int, double> idvreme;
public:
  statistike() : broj(0), tekuci_broj(0), maks_broj(0), broj_bez_cekanja(0),
    ukupno_vreme(0.0), povrsina(0.0), vremenski_trenutak(0.0){}
  void ulaz(const int id, double ti) {
    // Uvecamo broj entiteta koji su usli resurs
    broj++;
    // Prebrisana povrsina
    povrsina += (tekuci_broj++)*(ti - vremenski_trenutak);
    // Pamtimo trenutak poslednje promene broja zauzetih mesta
    vremenski_trenutak = ti;
    // Ukoliko je tekuci broj zauzetih mesta veci od maksimalnog
    // tekuci broj postaje maksimalni broj
    maks_broj = (tekuci_broj>maks_broj) ? tekuci_broj : maks_broj;
    // Pamtimo redni broj entiteta u resursu i trenutak ulaska entiteta u resurs
    idvreme[id] = ti;
  }
  void izlaz(const int id, double ti) {
    if (idvreme.find(id) != idvreme.end()) {
      double dt = ti - idvreme[id];
      // Ukoliko je vreme koje je entitet proveo u redu 0
      // uvecavamo broj entiteta koji nisu cekali u redu
      if (abs(dt)<numeric_limits<double>::epsilon()) broj_bez_cekanja++;
      // Ukupno vreme koje su entiteti proveli u salteru
      ukupno_vreme += dt;
      // Prebrisana povrsina
      povrsina += (tekuci_broj--)*(ti - vremenski_trenutak);
      // Pamtimo vreme ulaska entiteta u resurs
```

```
      vremenski_trenutak = ti;
      // Uklanjamo redni broj entiteta
      idvreme.erase(id);
    }
  }
  // Finisiramo statistike na kraju simulacije
  inline void finisiraj(double ti) {
    povrsina += tekuci_broj*(ti - vremenski_trenutak);
  }
  // Ocisti statistike
  void ocisti() {
    broj = 0;
    tekuci_broj = 0;
    maks_broj = 0;
    broj_bez_cekanja = 0;
    ukupno_vreme = 0.0;
    povrsina = 0.0;
    vremenski_trenutak = 0.0;
    idvreme.clear();
  }
public:
  inline int vrati_broj() const { return broj; }
  inline int vrati_tekuci_broj() const { return tekuci_broj; }
  inline int vrati_maksimalni_broj() const { return maks_broj; }
  inline int vrati_broj_bez_cekanja() const { return broj_bez_cekanja; }
  inline double vrati_ukupno_vreme() const { return ukupno_vreme; }
  inline double vrati_srednje_vreme() {
    return ukupno_vreme / (broj - tekuci_broj);
  }
  inline double vrati_srednje_vreme_bez_cekanja() {
    return ukupno_vreme / (broj - tekuci_broj - broj_bez_cekanja);
  }
  inline double vrati_srednji_broj(double sim_time) {
    return povrsina / sim_time;
  }
```

```
  inline double vrati_iskoriscenost(double sim_time, int broj_of_servers = 1) {
    return vrati_srednji_broj(sim_time) / broj_of_servers;
  }
  inline double vrati_ucesce_bez_cekanja() {
    return (100.0*broj_bez_cekanja) / broj;
  }
};
// Klasa simulacija
class simulacija {
  // Brojac entiteta
  int eid;
  // Terminacioni brojac
  int tb;
  // Vreme simulacije
  double vreme_simulacije, apsolutno_vreme_simulacije;
  // Polje entiteta u LBD listi
  deque<entitet*> lista_nastupanja_entiteta;
  // Sortiranje LBD - Sortiranje izborom
  void sortiraj_listu_nastupanja() {
    stable_sort(lista_nastupanja_entiteta.begin(), lista_nastupanja_entiteta.end(), simulacija::uporedi_entitete);
  }
public:
  // Konstruktor
  simulacija() :eid(1), tb(0), vreme_simulacije(0.0), apsolutno_vreme_simulacije(0.0) {}
  // Destruktor
  ~simulacija() {
    // Uklanjamo presotale entitete iz LBD na kraju simulacije
    ocisti();
  }
  // Pravljenje entiteta
  entitet* napravi_entitet(const int npar=20) {
    // Pravimo novi entitet
    return new entitet(eid++,npar);
  }
  // Unistavanje entiteta
  void unisti_entitet(entitet *e, int b) {
```

```
  // Unistavamo entitet
  delete e;
  // Umanjujemo terminacioni brojac
  tb -= b;
}
// Rasporedjivanje dogadjaja
void rasporedi(entitet* e, int dog, double vreme) {
  // Postavljamo dogadjaj
  e->naredni_dogadjaj = dog;
  // Postavljamo vreme nastupanja dogadjaja
  e->vreme_nastupanja_dogadjaja = vreme;
  // Smestamo entitet u listu
  lista_nastupanja_entiteta.push_back(e);
  // Sortiramo LBD
  sortiraj_listu_nastupanja();
}
// Izvrsavanje simulacije
void izvrsi(int b) {
  entitet * tekuci;
  // Postavljamo terminacioni brojac
  tb = b;
  // Izvrsavamo simulaciju. Simulacija se zavrsava ukoliko
  // nema entiteta u LBD ili ukoliko je terminacioni brojac 0.
  do {
    // Vadimo prvi entitet iz liste. Taj entitet postaje tekuci entitet.
    tekuci = lista_nastupanja_entiteta.front();
    lista_nastupanja_entiteta.pop_front();
    // Faza 1: Azurirarmo vreme simulacije
    vreme_simulacije = tekuci->vreme_nastupanja_dogadjaja;
    // Faza 2: Izvrsavamo dogadjaj
    izvrsavanje_dogadjaja(tekuci->naredni_dogadjaj, tekuci);
  } while (!lista_nastupanja_entiteta.empty() && tb>0);
}
// Uklanjamo entitete iz LBD
void ocisti() {
  // Uvecavamo apsolutno vreme simulacije za vrednost relativnog casovnika
```

```
    apsolutno_vreme_simulacije += vreme_simulacije;
    // Resetujemo vreme simulacije
    vreme_simulacije = 0.0;
    // Uklanjamo sve preostale entitete iz memorije
    for(auto it = lista_nastupanja_entiteta.begin();
        it != lista_nastupanja_entiteta.end(); it++)
        delete *it;
    // Cistimo listu
    lista_nastupanja_entiteta.clear();
  }
  // Izvrsava dogadjaj
  void izvrsavanje_dogadjaja(int dog, entitet* e);
  // Vraca vreme simulacije
  double vrati_vreme_simulacije() const { return vreme_simulacije; }
  // Vraca apsolutno vreme simulacije
  double vrati_apsolutno_vreme_simulacije() { return apsolutno_vreme_simulacije + vreme_simulacije; }
  // Komparacija vremena nastupanja entiteta
  static bool uporedi_entitete(entitet* ea, entitet* eb) {
    return ea->vreme_nastupanja_dogadjaja < eb->vreme_nastupanja_dogadjaja;
  }
};
// Klasa FIFO red cekanja
class red_cekanja {
  enum tip_reda {
    fifo, lifo
  };
private:
  // Tip reda cekanja
  tip_reda tip;
  // Simulacija
  simulacija* psim;
  // Statistike
  statistike stat;
  // Polje pokazivaca na entitete
  deque<entitet*> red;
public:
```

```
// Konstruktor
red_cekanja(tip_reda tip) : tip(tip), psim(nullptr) {}
red_cekanja(tip_reda tip, simulacija* s) : tip(tip), psim(s) {}
red_cekanja() : tip(fifo), psim(nullptr) {}
red_cekanja(simulacija* s) : tip(fifo), psim(s) {}

// Smestamo entitet u red cekanja
void ured(entitet* e) {
  // Smestamo entitet na kraj reda cekanja
  red.push_back(e);
  // Azuriramo statistike dolaskom entiteta
  if (e&&psim)
    stat.ulaz(e->id, psim->vrati_vreme_simulacije());
  else {
    cerr << "Nema objekta simulacije" << endl;
    exit(1);
  }
}
// Vadimo entitet iz reda cekanja
entitet* izred() {
  entitet *e = nullptr;
  if (!red.empty()) {
    if (tip == fifo) {
      // Vracamo prvi entitet iz reda cekanja
      e = red.front();
      // Vadimo entitet iz reda cekanja
      red.pop_front();
    }
    else if (tip == lifo){
      // Vracamo prvi entitet iz reda cekanja
      e = red.back();
      // Vadimo entitet iz reda cekanja
      red.pop_back();
    }
    // Azuriramo statistike odlaskom entiteta
    if (e&&psim)
```

```
      stat.izlaz(e->id, psim->vrati_vreme_simulacije());
    else {
      cerr << "Nema objekta simulacije" << endl;
      exit(1);
    }
  }
  else {
    cerr << "Nema entiteta u redu cekanja" << endl;
    exit(1);
  }
  // Vraca entitet
  return e;
}
// Finisiraj statistike
void finisiraj() {
  if (psim)
    stat.finisiraj(psim->vrati_vreme_simulacije());
  else {
    cerr << "Nema objekta simulacije" << endl;
    exit(1);
  }
}
// Uklanjamo entitete i resetujemo statistike
void ocisti() {
  stat.ocisti();
  for (deque<entitet*>::iterator it = red.begin(); it != red.end(); it++)
    delete *it;
  red.clear();
}
// Postavljamo simulaciju
void postavi_simulaciju(simulacija* s) { psim = s; }
// Vracamo velicinu reda cekanja
int velicina() { return stat.vrati_tekuci_broj(); };
// Vrati statistike
const statistike& vrati_statistike() const { return stat; }
```

```
};
// Klasa resursa
class resurs {
  // Simulacija
  simulacija* psim;
  // Statistike
  statistike stat;
  // Maksimalni broj  mesta u resursu
  int ukupni_broj_mesta;
public:
  // Konstruktori
  resurs() :ukupni_broj_mesta(0), psim(nullptr) {}
  resurs(int b) :ukupni_broj_mesta(b), psim(nullptr) {}
  resurs(simulacija* s, int b) :ukupni_broj_mesta(b), psim(s) {}
  // Zauzima mesto u resursu
  void zauzmi(entitet* e) {
    if (stat.vrati_tekuci_broj()<ukupni_broj_mesta) {
      // Azuriramo statistike dolaskom entiteta
      if (psim)
        stat.ulaz(e->id, psim->vrati_vreme_simulacije());
      else {
        cerr << "Nema objekta simulacije" << endl;
        exit(1);
      }
    }
    else {
      cerr << "Sva mesta u resursu su zauzeta" << endl;
      exit(1);
    }
  }
  // Oslobadja mesto u resursu
  void oslobodi(entitet* e) {
    if (e&&stat.vrati_tekuci_broj()>0) {
      // Azuriramo statistike odlaskom entiteta
      if (e&&psim)
        stat.izlaz(e->id, psim->vrati_vreme_simulacije());
```

```
        else {
          cerr << "Nema objekta simulacije" << endl;
          exit(1);
        }
      }
      else {
        cerr << "Nema entiteta u resursu" << endl;
        exit(1);
      }
    }
    // Finisiraj statistike
    void finisiraj() {
      if (psim)
        stat.finisiraj(psim->vrati_vreme_simulacije());
      else {
        cerr << "Nema entiteta u redu cekanja" << endl;
        exit(1);
      }
    }
    // Uklanjamo entitete i resetujemo statistike
    void ocisti() {
      stat.ocisti();
    }
    // Postavljamo simulaciju
    void postavi_simulaciju(simulacija* s) { psim = s; }
    // Postavlja i vraca broj mesta u resursu
    void postavi_broj_mesta(const int b) { ukupni_broj_mesta = b; }
    const int vrati_broj_mesta() const { return ukupni_broj_mesta; }
    // Raspolozivost resursa (ima makar jedno slobodno mesto u resursu)
    bool raspoloziv() { return (stat.vrati_tekuci_broj()<ukupni_broj_mesta); }
    // Vrati statistike
    const statistike& vrati_statistike() const { return stat; }
  };
}
```

Izvorna datoteka simu4.cpp

```
// Datototeka simu4.cpp
#include "simul.h"
#include <iostream>
#include <iomanip>
#include <string>

// 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 simul i std
using namespace simul;
using namespace std;

// Redni broj dogadjaja DolazakKlijentaUUniverzalniSalter
const int DOLAZAK_UNI_KLIJENTA   = 0;
// Redni broj dogadjaja OdlazakKlijentaIzUniverzalnogSaltera1
const int ODLAZAK_UNI_KLIJENTA_1 = 1;
// Redni broj dogadjaja OdlazakKlijentaIzUniverzalnogSaltera2
const int ODLAZAK_UNI_KLIJENTA_2 = 2;
// Redni broj dogadjaja DolazakKlijentaUPaketskiSalter
const int DOLAZAK_PAK_KLIJENTA   = 3;
// Redni broj dogadjaja OdlazakKlijentaIzPaketskogSaltera
const int ODLAZAK_PAK_KLIJENTA   = 4;
// Redni broj dogadjaja KrajSimulacije
const int KRAJ_SIMULACIJE        = 5;

// Potpis funkcije bezuslovnog dogadjaja DolazakKlijentaUUniverzalniSalter
```

```
void dolazak_uni_klijenta(entitet *e);
// Potpis funkcije bezuslovnog dogadjaja OdlazakKlijentaIzUniverzalnogSaltera1
void odlazak_uni_klijenta_1(entitet *e);
// Potpis funkcije bezuslovnog dogadjaja OdlazakKlijentaIzUniverzalnogSaltera2
void odlazak_uni_klijenta_2(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_1:
      odlazak_uni_klijenta_1(e);
      break;
    case ODLAZAK_UNI_KLIJENTA_2:
      odlazak_uni_klijenta_2(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);
  }
}
// Objekat simulacije
```

```
simulacija simu;
// Redovi cekanja
red_cekanja ured1(&simu),ured2(&simu),pred(&simu);
// Salteri
resurs uni_salt1(&simu,1),uni_salt2(&simu,1),pak_salt(&simu,1);
// Histogrami
histogram histo1(50,50,20), histo2(1,2,10);
// Raspodele
raspodela rni(76531), rn1(11615), rn2(36517), rn3(61521), rn4(98737), rn5(12341);

// Vremenski interval dolaska entiteta u univerzalni salter
const double VIDOLUNI[]   = {18000, 32400, 39600};
// Srednje vreme dolaska entiteta u univerzalni salter
const double SRVRDOLUNI[] = {50.0, 35.0, 25.0};
// Verovatnoca izbora srednjeg vremena opsluge entiteta u univerzalnom salteru u zavisnosti od usluge
const double PIOPSUNI[]   = {0.20, 0.60, 0.75, 1.0};
// 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   = 240;
// 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;
  int u_salter;
  double pi,vreme_opsluge;
  // Ukoliko je salter raspoloziv
  if(uni_salt1.raspoloziv())
    u_salter = 1;
  else if(uni_salt2.raspoloziv())
```

```
  u_salter = 2;
else if(ured1.velicina()<=ured2.velicina())
  u_salter = 1;
else
  u_salter = 2;
if(u_salter == 1) {
  // U parametar jedan upisujemo vremenski trenutak ulaska u red
  (*e)[0] = VREME;
  // Postavi klijenta u red cekanja
  ured1.ured(e);
  // Ukoliko red nije prazan
  if (uni_salt1.raspoloziv()) {
    // Izvadi prvog klijenta iz reda
    fe = ured1.izred();
    // Tabeliramo vreme boravka u redu
    histo1(VREME-(*fe)[0]);
    // Zauzmi jedno mesto u salteru
    uni_salt1.zauzmi(fe);
    // Vreme opsluge
    vreme_opsluge = rn2.expo(funkcija::diskretna(rni(),PIOPSUNI,SRVROPSUNI,4));
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_UNI_KLIJENTA_1,VRTR(vreme_opsluge));
  }
}
else {
  // U parametar jedan upisujemo vremenski trenutak ulaska u red
  (*e)[0] = VREME;
  // Postavi klijenta u red cekanja
  ured2.ured(e);
  // Ukoliko red nije prazan
  if (uni_salt2.raspoloziv()) {
    // Izvadi prvog klijenta iz reda
    fe = ured2.izred();
    // Tabeliramo vreme boravka u redu
    histo1(VREME-(*fe)[0]);
    // Zauzmi jedno mesto u salteru
```

```
      uni_salt2.zauzmi(fe);
      // Vreme opsluge
      vreme_opsluge = rn2.expo(funkcija::diskretna(rni(),PIOPSUNI,SRVROPSUNI,4));
      // Rasporedi klijenta za kraj opsluge
      simu.rasporedi(fe,ODLAZAK_UNI_KLIJENTA_2,VRTR(vreme_opsluge));
    }
  }
  // Stvaramo narednog klijenta
  ne = simu.napravi_entitet();
  // Postavljamo vreme narednog dolaska
  simu.rasporedi(ne,DOLAZAK_UNI_KLIJENTA,VRTR(rn1.expo(funkcija::diskretna(VREME,VIDOLUNI,SRVRDOLUNI,3))));
}
// Dogadjaj odlazak klijenta sa univerzalnog saltera 1
void odlazak_uni_klijenta_1(entitet *e) {
  entitet *fe;
  double pi,vreme_opsluge;
  // Vrati salter;
  uni_salt1.oslobodi(e);
  // Klijent odlazi iz poste - unistavamo tekuceg klijenta
  simu.unisti_entitet(e,0);
  // Ukoliko red nije prazan
  if(ured1.velicina()>0) {
    // Izvadi prvog klijenta iz reda
    fe = ured1.izred();
    // Tabeliramo vreme boravka u redu
    histo1(VREME-(*fe)[0]);
    // Zauzmi jedno mesto u salteru
    uni_salt1.zauzmi(fe);
    // Vreme opsluge
    vreme_opsluge = rn2.expo(funkcija::diskretna(rni(),PIOPSUNI,SRVROPSUNI,4));
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_UNI_KLIJENTA_1,VRTR(vreme_opsluge));
  }
}
// Dogadjaj odlazak klijenta sa univerzalnog saltera 2
void odlazak_uni_klijenta_2(entitet *e) {
```

```
  entitet *fe;
  double pi,vreme_opsluge;
  // Vrati salter;
  uni_salt2.oslobodi(e);
  // Klijent odlazi iz poste - unistavamo tekuceg klijenta
  simu.unisti_entitet(e,0);
  // Ukoliko red nije prazan
  if(ured2.velicina()>0) {
    // Izvadi prvog klijenta iz reda
    fe = ured2.izred();
    // Tabeliramo vreme boravka u redu
    histo1(VREME-(*fe)[0]);
    // Zauzmi jedno mesto u salteru
    uni_salt2.zauzmi(fe);
    // Vreme opsluge
    vreme_opsluge = rn2.expo(funkcija::diskretna(rni(),PIOPSUNI,SRVROPSUNI,4));
    // Rasporedi klijenta za kraj opsluge
    simu.rasporedi(fe,ODLAZAK_UNI_KLIJENTA_2,VRTR(vreme_opsluge));
  }
}
// Dogadjaj dolazak klijenta na paketski salter
void dolazak_pak_klijenta(entitet *e) {
  entitet *fe, *ne;
  double vreme_opsluge;
  if(VREME>=3600 && VREME<=36000) {
    // Postavi klijenta u red cekanja
    pred.ured(e);
    // Ukoliko je salter raspoloziv
    if(pak_salt.raspoloziv()) {
      // Tabeliramo duzinu reda cekanja
      histo2(pred.velicina());
      // Izvadi prvog klijenta iz reda
      fe = pred.izred();
      // Zauzmi jedno mesto u salteru
      pak_salt.zauzmi(fe);
      do {
```

```
        vreme_opsluge = rn5.norm(SRVROPSPAK,STDEVOPSPAK);
      } while(vreme_opsluge<MINOPSPAK);
      // Rasporedi klijenta za kraj opsluge
      simu.rasporedi(fe,ODLAZAK_PAK_KLIJENTA,VRTR(vreme_opsluge));
    }
  }
  else {
    simu.unisti_entitet(e,0);
  }
  // Stvaramo narednog klijenta
  ne = simu.napravi_entitet();
  // Postavljamo vreme narednog dolaska
  simu.rasporedi(ne,DOLAZAK_PAK_KLIJENTA,
                    VRTR(rn3.expo(SRVRDOLPAK/2)+rn4.expo(SRVRDOLPAK/2)));
}
// Dogadjaj odlazak klijenta sa paketskog saltera
void odlazak_pak_klijenta(entitet *e) {
  entitet *fe;
  double vreme_opsluge;
  // Vrati paketski salter;
  pak_salt.oslobodi(e);
  // Klijent odlazi iz poste - unistavamo tekuceg klijenta
  simu.unisti_entitet(e,0);
  // Ukoliko red ispred paketsko saltera nije prazan
  if(pred.velicina()>0) {
    // Tabeliramo duzinu reda cekanja
    histo2(pred.velicina());
    // Izvadi prvog klijenta iz reda
    fe = pred.izred();
    // Zauzmi jedno mesto u paketskom salteru
    pak_salt.zauzmi(fe);
    do {
      vreme_opsluge = rn5.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);
  // Stvaramo naredni entitet
  entitet* ne = simu.napravi_entitet();
  simu.rasporedi(ne,KRAJ_SIMULACIJE,VRTR(3600.0));
}
void ocisti() {
  ured1.ocisti();
  ured2.ocisti();
  pred.ocisti();
  uni_salt1.ocisti();
  uni_salt2.ocisti();
  pak_salt.ocisti();
  histo1.resetuj();
  histo2.resetuj();
}
void stampaj_statistike_resursa(const string& ime,const resurs& res) {
       statistike stat = res.vrati_statistike();
       // Statistike resursa
       cout << "STATISTIKE RESURSA - " << ime << endl;
       cout << "Broj entiteta koji je usao u resurs: " << stat.vrati_broj() << endl;
       cout << "Preostali broj entiteta u resursu: " << stat.vrati_tekuci_broj() << endl;
       cout << "Maksimalni broj zauzetih kanala opsluge: " << stat.vrati_maksimalni_broj() << endl;
       cout << "Srednje vreme opsluge: " << stat.vrati_srednje_vreme() << endl;
       cout << "Srednji broj zauzetih kanala: " << stat.vrati_srednji_broj(VREME) << endl;
       cout << "Iskoriscenost: " << stat.vrati_iskoriscenost(VREME, res.vrati_broj_mesta()) << endl;
       cout << endl;
}
void stampaj_statistike_reda_cekanja(const string& ime,const red_cekanja& red) {
       statistike stat = red.vrati_statistike();
       // Statistike reda cekanja
       cout << "STATISTIKE REDA CEKANJA - " << ime << endl;
```

```
	cout << "Broj entiteta koji je usao u red cekanja: " << stat.vrati_broj() << endl;
	cout << "Preostali broj entiteta u redu cekanja: " << stat.vrati_tekuci_broj() << endl;
	cout << "Maksimalni broj entiteta u redu: " << stat.vrati_maksimalni_broj() << endl;
	cout << "Srednje vreme cekanja u redu: " << stat.vrati_srednje_vreme() << endl;
	cout << "Srednji broj entiteta u redu: " << stat.vrati_srednji_broj(VREME) << endl;
	cout << "Broj entiteta koji je prosao kroz red bez zadrzavanja: " << stat.vrati_broj_bez_cekanja() << endl;
	cout << "Procenat ulaza bez zadrzavanja: " << stat.vrati_ucesce_bez_cekanja() << endl;
	cout << "Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): " <<
stat.vrati_srednje_vreme_bez_cekanja() << endl;
	cout << endl;
}
void stampaj_statistike_histograma(const string& ime,histogram& histo) {
	// Statistike histograma
	cout << "HISTOGRAM - " << ime << endl;
	cout << "Broj ulaza u histogram: " << histo.vrati_broj_ulaza_u_histogram() << endl;
	if (histo.vrati_frekvenciju_intervala(histo.vrati_broj_intervala() - 1) > 0){
		cout << "Prosecna vrednost overflowa: " << histo.vrati_prosecnu_vrednost_overflowa() << endl;
		cout << "Frekvencija overflowa: " << histo.vrati_frekvenciju_intervala(histo.vrati_broj_intervala() -
1) << endl;
		cout << "Verovatnoca overflowa: " << histo.vrati_verovatnocu_intervala(histo.vrati_broj_intervala() -
1) << endl;
	}
	cout << "Srednja vrednost histograma: " << histo.vrati_srednju_vrednost_histograma() << endl;
	cout << "Standardna devijacija histograma: " << histo.vrati_standardnu_devijaciju_histograma() << endl;
	cout << "Suma argumenata histograma: " << histo.vrati_sumu_argumenata_histograma() << endl;
	cout << setw(10) << "int   " << setw(10) << "frek  " << setw(10) << "vero  " << setw(10) << "umno  " <<
setw(10) << "dev   " << endl;
	for (int i = 0; i < histo.vrati_broj_intervala() - 1; i++){
		cout << setw(10) << histo.vrati_gornju_granicu_intervala(i) << " "
			<< setw(10) << histo.vrati_frekvenciju_intervala(i) << " "
			<< setw(10) << histo.vrati_verovatnocu_intervala(i) << " "
			<< setw(10) << histo.vrati_umnozak_od_srednje_vrednosti_intervala(i) << " "
			<< setw(10) << histo.vrati_devijaciju_od_srednje_vrednosti_intervala(i) << endl;
	}
	cout << endl;
```

```
}
void finisiraj_statistike() {
  ured1.finisiraj();
  ured2.finisiraj();
  pred.finisiraj();
  uni_salt1.finisiraj();
  uni_salt2.finisiraj();
  pak_salt.finisiraj();
}
void stampaj_statistike() {
  stampaj_statistike_resursa("uni_salt1",uni_salt1);
  stampaj_statistike_resursa("uni_salt2",uni_salt2);
  stampaj_statistike_resursa("pak_salt" ,pak_salt);

  stampaj_statistike_reda_cekanja("ured1",ured1);
  stampaj_statistike_reda_cekanja("ured2",ured2);
  stampaj_statistike_reda_cekanja("pred",pred);

  stampaj_statistike_histograma("histo1",histo1);
  stampaj_statistike_histograma("histo2",histo2);
}
void izvrsi_dan() {
  // Stvaramo prvog klijenta
  entitet *e1 = simu.napravi_entitet();
  // Postavljamo vreme dolaska prvog klijenta
  simu.rasporedi(e1,DOLAZAK_UNI_KLIJENTA,rn1.expo(SRVRDOLUNI[0]));
  // Stvaramo prvog klijenta
  entitet *e2 = simu.napravi_entitet();
  // Postavljamo vreme dolaska prvog klijenta
  simu.rasporedi(e2,DOLAZAK_PAK_KLIJENTA,rn3.expo(SRVRDOLPAK/2)+rn4.expo(SRVRDOLPAK/2));
  // Stvaramo entitet za kraj simulacije
  entitet *e3 = simu.napravi_entitet();
  // Postavljamo vreme zavrsetka simulacije
  simu.rasporedi(e3,KRAJ_SIMULACIJE,3600.0);
  // Izvrsavamo simulaciju
```

```
  simu.izvrsi(11);
  // Finisiramo statistike
  finisiraj_statistike();
  // Stampamo statistike
  stampaj_statistike();
  // Cistimo statistike
  ocisti();
  // Brisemo preostale entitete iz LBD
  simu.ocisti();
}
int main() {
  cout << "Prvi dan" << endl;
  // Izvrsavamo prvi radni dan
  izvrsi_dan();
  cout << "Drugi dan" << endl;
  // Izvrsavamo drugi radni dan
  izvrsi_dan();
  return 0;
}
```

Listing simulacionog programa realizovanog u GPSS/H

```
Student GPSS/H Release 3.70 (AY015)     28 Dec 2015   11:17:32     File: simu4.gps

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

    1     1                         SIMULATE
    2     2                         REALLOCATE COM,32720
    3     3                  DOLS   FUNCTION C1,D3
    4     4                  18000,50/32400,35/39600,25
    5     5                  ERLAN  FVARIABLE RVEXPO(2,120)+RVEXPO(2,120)
    6     6                  SVUSL  FUNCTION RN5,D4
    7     7                  0.20,40/0.60,50/0.75,15/1.0,70
    8     8                 *
    9     9                 * UNIVERZALNI SALTERI
   10    10                 *
   11    11             1           GENERATE RVEXPO(4,FN(DOLS)),,,,,1PH
   12    12             2           SELECT E (SALT)PH,1,2,0,F,AAA
   13    13             3   NAZD    QUEUE PH(SALT)
   14    14             4           SEIZE PH(SALT)
   15    15             5           DEPART PH(SALT)
   16    16             6           TABULATE 1
   17    17             7           ADVANCE RVEXPO(5,FN(SVUSL))
   18    18             8           RELEASE PH(SALT)
   19    19             9           TERMINATE
```

```
20    20              10   AAA    SELECT MIN (SALT)PH,1,2,,Q
21    21              11          TRANSFER ,NAZD
22    22                  1       TABLE M1,50,50,20
23    23                  *
24    24                  * PAKETSKI SALTER
25    25                  *
26    26              12          GENERATE V$ERLAN,,,,,1PL
27    27              13          TEST GE C1,3600,VAN
28    28              14          TEST LE C1,36000,VAN
29    29              15          QUEUE 3
30    30              16          SEIZE 3
31    31              17          TABULATE 2
32    32              18          DEPART 3
33    33              19   PON    ASSIGN VROPS,RVNORM(6,150,60),PL
34    34              20          TEST GE PL(VROPS),20,PON
35    35              21          ADVANCE PL(VROPS)
36    36              22          RELEASE 3
37    37              23   VAN    TERMINATE
38    38                   2      TABLE Q3,1,2,10
39    39                  *
40    40                  * SEGMENT TAJMERA
41    41                  *
42    42              24          GENERATE 3600
43    43              25          TERMINATE 1
44    44                          START 11
45    45                          CLEAR
46    46                          START 11
47    47                          END
```

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

       Facilities: 1               2               3

           Queues: 1               2               3

           Tables: 1               2

        Functions: 1=DOLS          2=SVUSL

     (F)variables: 1=ERLAN

   Halfword Parms: 1=SALT

      Float Parms: 1=VROPS

   Random Numbers: 2               4               5               6

Symbol    Value   EQU Defns  Context      References by Statement Number

AAA          10          20  Block           12
NAZD          3          13  Block           21
PON          19          33  Block           34
VAN          23          37  Block           27    28

1             1              Facility        12
2             2              Facility        12
3             3              Facility        30    36
```

1	1		Queue	20					
2	2		Queue	20					
3	3		Queue	29	32	38			
1	1	22	Table	16					
2	2	38	Table	31					
DOLS	1	3	Function	11					
SVUSL	2	6	Function	17					
ERLAN	1	5	(F)variable	26					
SALT	1		Halfword Par	12	13	14	15	18	20
VROPS	1		Float Par	33	34	35			
2	2		Random Nmbr	5	5				
4	4		Random Nmbr	11					
5	5		Random Nmbr	6	17				
6	6		Random Nmbr	33					

Storage Requirements (Bytes)

Compiled Code:	1844
Compiled Data:	236
Miscellaneous:	0
Entities:	3680

```
Common:            32720
-----------------------
Total:             38480


GPSS/H Model Size:

Control Statements     10
Blocks                 25

Simulation begins.

Relative Clock: 39600.0000   Absolute Clock: 39600.0000


Block Current     Total  Block Current     Total  Block Current     Total
1                  1056  11                  555  21                  139
2                  1056  12                  162  22                  139
NAZD       40      1056  13                  162  VAN                 162
4                  1016  14                  151  24                   11
5                  1016  15                  139  25                   11
6                  1016  16                  139
7           2      1016  17                  139
8                  1014  18                  139
9                  1014  PON                 140
AAA                 555  20                  140
```

Facility	--Avg-Util-During-- Total Time	Avail Time	Unavl Time	Entries	Average Time/Xact	Current Status	Percent Avail	Seizing Xact	Preempting Xact
1	0.697			570	48.426	AVAIL		1193	
2	0.526			446	46.741	AVAIL		1172	
3	0.524			139	149.269	AVAIL			

Queue	Maximum Contents	Average Contents	Total Entries	Zero Entries	Percent Zeros	Average Time/Unit	$Average Time/Unit	Qtable Number	Current Contents
1	34	3.528	590	282	47.8	236.762	453.537	20	
2	33	3.285	466	219	47.0	279.127	526.612	20	
3	3	0.240	139	67	48.2	68.429	132.106	0	

Table 1

Entries in Table	Mean Argument	Standard Deviation	Sum of Arguments	
1016.0000	233.5624	445.0633	2.3730E+05	Non-Weighted

Upper Limit	Observed Frequency	Percent of Total	Cumulative Percentage	Cumulative Remainder	Multiple of Mean	Deviation From Mean
50.0000	646.0000	63.5827	63.58	36.42	0.2141	-0.4124
100.0000	93.0000	9.1535	72.74	27.26	0.4282	-0.3001
150.0000	42.0000	4.1339	76.87	23.13	0.6422	-0.1878
200.0000	19.0000	1.8701	78.74	21.26	0.8563	-0.0754
250.0000	10.0000	0.9843	79.72	20.28	1.0704	0.0369
300.0000	6.0000	0.5906	80.31	19.69	1.2845	0.1493
350.0000	2.0000	0.1969	80.51	19.49	1.4985	0.2616
400.0000	8.0000	0.7874	81.30	18.70	1.7126	0.3740
450.0000	3.0000	0.2953	81.59	18.41	1.9267	0.4863
500.0000	6.0000	0.5906	82.19	17.81	2.1408	0.5987
550.0000	4.0000	0.3937	82.58	17.42	2.3548	0.7110
600.0000	3.0000	0.2953	82.87	17.13	2.5689	0.8233
650.0000	2.0000	0.1969	83.07	16.93	2.7830	0.9357
700.0000	2.0000	0.1969	83.27	16.73	2.9971	1.0480
750.0000	5.0000	0.4921	83.76	16.24	3.2111	1.1604
800.0000	2.0000	0.1969	83.96	16.04	3.4252	1.2727
850.0000	3.0000	0.2953	84.25	15.75	3.6393	1.3851
900.0000	6.0000	0.5906	84.84	15.16	3.8534	1.4974
950.0000	10.0000	0.9843	85.83	14.17	4.0674	1.6097
Overflow	144.0000	14.17	100.00	0.00		

Average value of overflow is 1238.8866

Table 2

Entries in Table	Mean Argument	Standard Deviation	Sum of Arguments	
139.0000	1.1727	0.4329	163.0000	Non-Weighted

Upper Limit	Observed Frequency	Percent of Total	Cumulative Percentage	Cumulative Remainder	Multiple of Mean	Deviation From Mean
1.0000	118.0000	84.8921	84.89	15.11	0.8528	-0.3989
3.0000	21.0000	15.1079	100.00	-0.00	2.5583	4.2216

Random Stream	Antithetic Variates	Initial Position	Current Position	Sample Count	Chi-Square Uniformity
2	OFF	200000	200326	326	0.24
4	OFF	400000	401057	1057	0.44
5	OFF	500000	502032	2032	0.09
6	OFF	600000	600280	280	0.35

Status of Common Storage

27528 bytes available
5192 in use
8216 used (max)

Relative Clock: 39600.0000 Absolute Clock: 39600.0000

Block	Current	Total	Block	Current	Total	Block	Current	Total
1		1051	11		597	21		126
2		1051	12		164	22		126
NAZD	2	1051	13		164	VAN		164
4		1049	14		147	24		11
5		1049	15		126	25		11
6		1049	16		126			
7	2	1049	17		126			
8		1047	18		126			
9		1047	PON		126			
AAA		597	20		126			

	--Avg-Util-During--							
Facility	Total Time	Avail Time	Unavl Time	Entries	Average Time/Xact	Current Status	Percent Avail	Seizing Xact	Preempting Xact
1	0.714			598	47.302	AVAIL		2457	
2	0.535			451	46.940	AVAIL		2455	
3	0.489			126	153.670	AVAIL			

Queue	Maximum Contents	Average Contents	Total Entries	Zero Entries	Percent Zeros	Average Time/Unit	$Average Time/Unit	Qtable Number	Current Contents
1	9	1.094	599	264	44.1	72.300	129.277	1	

2	8	0.903	452	190	42.0	79.091	136.447
1							
3	4	0.234	126	71	56.3	73.565	168.530
0							

Table 1

Entries in Table	Mean Argument	Standard Deviation	Sum of Arguments	
1049.0000	75.3347	105.8402	79026.0539	Non-Weighted

Upper Limit	Observed Frequency	Percent of Total	Cumulative Percentage	Cumulative Remainder	Multiple of Mean	Deviation From Mean
50.0000	638.0000	60.8198	60.82	39.18	0.6637	-0.2394
100.0000	98.0000	9.3422	70.16	29.84	1.3274	0.2330
150.0000	82.0000	7.8170	77.98	22.02	1.9911	0.7055
200.0000	71.0000	6.7684	84.75	15.25	2.6548	1.1779
250.0000	74.0000	7.0543	91.80	8.20	3.3185	1.6503
300.0000	34.0000	3.2412	95.04	4.96	3.9822	2.1227
350.0000	18.0000	1.7159	96.76	3.24	4.6459	2.5951
400.0000	20.0000	1.9066	98.67	1.33	5.3096	3.0675
450.0000	10.0000	0.9533	99.62	0.38	5.9733	3.5399
500.0000	4.0000	0.3813	100.00	0.00	6.6371	4.0123

Table 2

Entries in Table	Mean Argument	Standard Deviation	Sum of Arguments	
126.0000	1.2302	0.5819	155.0000	Non-Weighted

Upper Limit	Observed Frequency	Percent of Total	Cumulative Percentage	Cumulative Remainder	Multiple of Mean	Deviation From Mean
1.0000	106.0000	84.1270	84.13	15.87	0.8129	-0.3955
3.0000	19.0000	15.0794	99.21	0.79	2.4387	3.0415
5.0000	1.0000	0.7937	100.00	0.00	4.0645	6.4785

Random Stream	Antithetic Variates	Initial Position	Current Position	Sample Count	Chi-Square Uniformity
2	OFF	200326	200656	330	0.92
4	OFF	401057	402109	1052	0.43
5	OFF	502032	504130	2098	0.12
6	OFF	600280	600532	252	0.86

Status of Common Storage

31784 bytes available
936 in use
8216 used (max)

```
Simulation complete.  Absolute Clock: 39600.0000


Total Block Executions: 24449

Blocks / second:        3364163

Microseconds / Block:   0.30


Elapsed Time Used (Sec)

Pass1:            0.00
Sym/Xref          0.00
Pass2:            0.00
Load/Ctrl:        0.00
Execution:        0.01
Output:           0.00
--------------------
Total:            0.01
```

Rezultati dobijeni u programskom jeziku C++

```
Prvi dan
STATISTIKE RESURSA - uni_salt1
Broj entiteta koji je usao u resurs: 594
Preostali broj entiteta u resursu: 1
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 47.3409
Srednji broj zauzetih kanala: 0.70903
Iskoriscenost: 0.70903

STATISTIKE RESURSA - uni_salt2
Broj entiteta koji je usao u resurs: 439
Preostali broj entiteta u resursu: 1
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 51.2768
Srednji broj zauzetih kanala: 0.567274
Iskoriscenost: 0.567274

STATISTIKE RESURSA - pak_salt
Broj entiteta koji je usao u resurs: 138
Preostali broj entiteta u resursu: 0
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 154.873
Srednji broj zauzetih kanala: 0.539708
Iskoriscenost: 0.539708

STATISTIKE REDA CEKANJA - ured1
```

```
Broj entiteta koji je usao u red cekanja: 596
Preostali broj entiteta u redu cekanja: 2
Maksimalni broj entiteta u redu: 16
Srednje vreme cekanja u redu: 65.2142
Srednji broj entiteta u redu: 0.980775
Broj entiteta koji je prosao kroz red bez zadrzavanja: 279
Procenat ulaza bez zadrzavanja: 46.8121
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 122.975

STATISTIKE REDA CEKANJA - ured2
Broj entiteta koji je usao u red cekanja: 440
Preostali broj entiteta u redu cekanja: 1
Maksimalni broj entiteta u redu: 15
Srednje vreme cekanja u redu: 70.1086
Srednji broj entiteta u redu: 0.778511
Broj entiteta koji je prosao kroz red bez zadrzavanja: 230
Procenat ulaza bez zadrzavanja: 52.2727
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 147.262

STATISTIKE REDA CEKANJA - pred
Broj entiteta koji je usao u red cekanja: 138
Preostali broj entiteta u redu cekanja: 0
Maksimalni broj entiteta u redu: 3
Srednje vreme cekanja u redu: 85.0481
Srednji broj entiteta u redu: 0.29638
Broj entiteta koji je prosao kroz red bez zadrzavanja: 63
Procenat ulaza bez zadrzavanja: 45.6522
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 156.489

HISTOGRAM - histo1
Broj ulaza u histogram: 1033
```

```
Srednja vrednost histograma: 67.2942
Standardna devijacija histograma: 128.805
Suma argumenata histograma: 69514.9
    int       frek      vero      umno      dev
     50        686   0.664085   0.743006  -0.134266
    100        129   0.124879    1.48601   0.253916
    150         76  0.0735721    2.22902   0.642099
    200         51  0.0493708    2.97202    1.03028
    250         15  0.0145208    3.71503    1.41846
    300         12  0.0116167    4.45804    1.80665
    350          8 0.00774443    5.20104    2.19483
    400          7 0.00677638    5.94405    2.58301
    450         12  0.0116167    6.68705    2.97119
    500          4 0.00387222    7.43006    3.35938
    550         13  0.0125847    8.17307    3.74756
    600          8 0.00774443    8.91607    4.13574
    650          5 0.00484027    9.65908    4.52392
    700          4 0.00387222    10.4021    4.91211
    750          0          0    11.1451    5.30029
    800          2 0.00193611    11.8881    5.68847
    850          1 0.000968054    12.6311    6.07665
    900          0          0    13.3741    6.46484
    950          0          0    14.1171    6.85302

HISTOGRAM - histo2
Broj ulaza u histogram: 138
Srednja vrednost histograma: 1.22464
Standardna devijacija histograma: 0.452376
Suma argumenata histograma: 169
    int       frek      vero      umno      dev
      1        109   0.789855   0.816568  -0.496573
```

3	29	0.210145	2.4497	3.92453
5	0	0	4.08284	8.34563
7	0	0	5.71598	12.7667
9	0	0	7.34911	17.1878
11	0	0	8.98225	21.6089
13	0	0	10.6154	26.03
15	0	0	12.2485	30.4511
17	0	0	13.8817	34.8722

Drugi dan
STATISTIKE RESURSA - uni_salt1
Broj entiteta koji je usao u resurs: 595
Preostali broj entiteta u resursu: 1
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 46.6523
Srednji broj zauzetih kanala: 0.701054
Iskoriscenost: 0.701054

STATISTIKE RESURSA - uni_salt2
Broj entiteta koji je usao u resurs: 451
Preostali broj entiteta u resursu: 1
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 47.4433
Srednji broj zauzetih kanala: 0.54085
Iskoriscenost: 0.54085

STATISTIKE RESURSA - pak_salt
Broj entiteta koji je usao u resurs: 131
Preostali broj entiteta u resursu: 0
Maksimalni broj zauzetih kanala opsluge: 1
Srednje vreme opsluge: 163.627

```
Srednji broj zauzetih kanala: 0.541291
Iskoriscenost: 0.541291

STATISTIKE REDA CEKANJA - ured1
Broj entiteta koji je usao u red cekanja: 600
Preostali broj entiteta u redu cekanja: 5
Maksimalni broj entiteta u redu: 10
Srednje vreme cekanja u redu: 56.1796
Srednji broj entiteta u redu: 0.882393
Broj entiteta koji je prosao kroz red bez zadrzavanja: 291
Procenat ulaza bez zadrzavanja: 48.5
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 109.957

STATISTIKE REDA CEKANJA - ured2
Broj entiteta koji je usao u red cekanja: 458
Preostali broj entiteta u redu cekanja: 7
Maksimalni broj entiteta u redu: 9
Srednje vreme cekanja u redu: 61.5555
Srednji broj entiteta u redu: 0.725211
Broj entiteta koji je prosao kroz red bez zadrzavanja: 210
Procenat ulaza bez zadrzavanja: 45.8515
Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 115.193

STATISTIKE REDA CEKANJA - pred
Broj entiteta koji je usao u red cekanja: 131
Preostali broj entiteta u redu cekanja: 0
Maksimalni broj entiteta u redu: 4
Srednje vreme cekanja u redu: 80.4306
Srednji broj entiteta u redu: 0.266071
Broj entiteta koji je prosao kroz red bez zadrzavanja: 65
Procenat ulaza bez zadrzavanja: 49.6183
```

Srednje vreme cekanja u redu (iskljuc. ent. koji se nisu zadrzali): 159.643

HISTOGRAM - histo1
Broj ulaza u histogram: 1046
Srednja vrednost histograma: 58.4975
Standardna devijacija histograma: 93.5899
Suma argumenata histograma: 61188.4

int	frek	vero	umno	dev
50	707	0.675908	0.854737	-0.0907955
100	101	0.0965583	1.70947	0.44345
150	65	0.0621415	2.56421	0.977696
200	74	0.0707457	3.41895	1.51194
250	50	0.0478011	4.27368	2.04619
300	21	0.0200765	5.12842	2.58043
350	10	0.00956023	5.98316	3.11468
400	5	0.00478011	6.83789	3.64892
450	4	0.00382409	7.69263	4.18317
500	6	0.00573614	8.54737	4.71742
550	2	0.00191205	9.4021	5.25166
600	1	0.000956023	10.2568	5.78591
650	0	0	11.1116	6.32015
700	0	0	11.9663	6.8544
750	0	0	12.8211	7.38865
800	0	0	13.6758	7.92289
850	0	0	14.5305	8.45714
900	0	0	15.3853	8.99138
950	0	0	16.24	9.52563

HISTOGRAM - histo2
Broj ulaza u histogram: 131
Srednja vrednost histograma: 1.23664

Standardna devijacija histograma: 0.53826
Suma argumenata histograma: 162

int	frek	vero	umno	dev
1	106	0.80916	0.808642	-0.439641
3	24	0.183206	2.42593	3.27603
5	1	0.00763359	4.04321	6.99171
7	0	0	5.66049	10.7074
9	0	0	7.27778	14.4231
11	0	0	8.89506	18.1387
13	0	0	10.5123	21.8544
15	0	0	12.1296	25.5701
17	0	0	13.7469	29.2858

Kompajliranje i izvršenje programa

```
cl simu4.cpp /EHsc
simu4
```