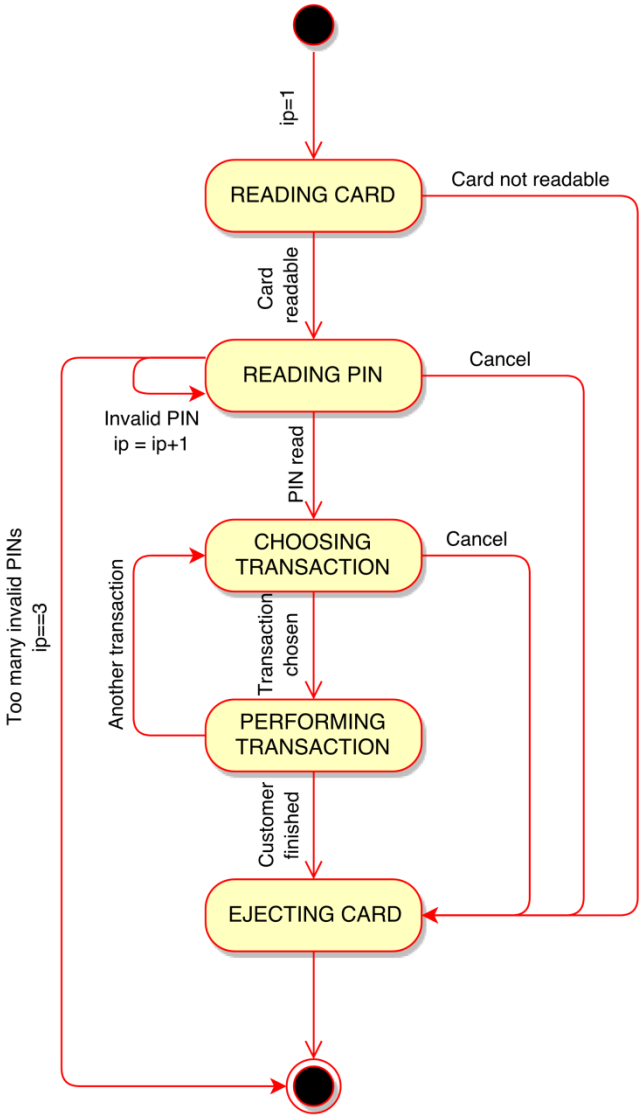
Simulirati rad bankomata čiji je rad opisan sledećim dijagramom stanja i tabelom prelaska iz stanje u stanje.



Stanje u / Stanje iz	READING CARD	READING PIN	CHOOSING TRANSACTION	PERFORMING TRANSACTION	EJECTING CARD	Final
Initial	*Start*, ip = 1 1.0	-	-	-	-	-
READING CARD	-	*Card readable* 0.95	-	-	*Card not readable* 0.05	-
READING PIN	-	*Invalid pin, ip++* 0.05	*PIN read* 0.70	-	Cancel 0.20	*Too many invalid PINs* ip==3
CHOOSING TRANSACTION	-	-	-	*Transaction chosen* 0.80	*Cancel* 0.20	-
PERFORMING TRANSACTION	-	-	*Another transaction* 0.40	-	*Customer finished* 0.60	-
EJECTING CARD	-	-	-	-	-	*Finish* 1.0

Rešenje:

Kod ovog rešenja stanja i događaji su deklarisani kao nabrojivi tipovi (enum). Promena stanja u klasi dijagrama stanja (statechart) se odigrava kroz funkciju tranzicija koja za prosleđeni događaj dovodi do promene stanja u kome se bankomat nalazi.

```
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

// Funkcija vraca slucajan broj na intervalu od 0 do 1
double random() {
  return rand()/(1.0*RAND_MAX);
}

// Stanja
enum state {
  reading_card,              // Citam kartu
  reading pin,               // Citam PIN
  choosing transaction,      // Izabiram transakciju
  performing transaction,    // Transakcija
  ejecting_card              // Izbacujem karticu
};

// Dogadjaji
enum event {
  card readable,             // Kartica moze da se procita
  card_not_readable,         // Kartica ne moze da se procita
  cancel,                    // Otkazivanje
  pin read,                  // Ucitan PIN
  invalid pin,               // Neispravan PIN
  transaction chosen,        // Izabrana transakcija
  another_transaction,       // Druga transakcija
  customer_finished          // Korisnik zavrsava
};

// Dijagram stanja
class statechart {
  // Zastavica prelaska u zavrsno stanje
  bool _final;
  // Tekuce stanje
  state _current;
public:
  // Konstruktor
  statechart(): _final(false) {}
  // Postavljamo pocetno stanje
  void initial(const state s) {
    _current = s;
  }
  // Tranzicija iz stanja u stanje na osnovu okinutog dogadjaja
  void transition(const event e) {
    // Proveravamo koji je dogadjaj okinut i na osnovu okinutog dogadjaja
    // i stanja u kome se dijagram nalazi sprovodimo promenu stanja.
    switch (e){
    case card readable:
      if(_current==reading_card) {
        _current = reading_pin;
        cout << "READING_PIN" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case card not readable:
      if(_current==reading_card) {
        _current = ejecting_card;
        cout << "EJECTING_CARD" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
```

```
      }
      break;
    case cancel:
      if(_current==reading_pin) {
        _current = ejecting_card;
        cout << "EJECTING_CARD" << endl;
      }
      else if( current==choosing transaction) {
        _current = ejecting_card;
        cout << "EJECTING_CARD" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case pin_read:
      if( current==reading pin) {
         current = choosing transaction;
        cout << "CHOOSING_TRANSACTION" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case invalid_pin:
      if( current==reading pin) {
         current = reading pin;
        cout << "READING_PIN" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case transaction_chosen:
      if(_current==choosing_transaction) {
         current = performing transaction;
        cout << "PERFORMING_TRANSACTION" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case another transaction:
      if(_current==performing_transaction) {
        _current = choosing_transaction;
        cout << "CHOOSING_TRANSACTION" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case customer finished:
      if( current==performing transaction) {
        _current = ejecting_card;
        cout << "EJECTING_CARD" << endl;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    default:
      break;
    }
  }
public:
  // Postavi u finalno stanje i vrati da li je u finalnom stanju
  bool get_final()                    { return _final;  }
  void set_final(const bool f)        { _final = f;     }
  // Postavi i vrati tekuce stanje
  void   set_current(const state s)   { _current = s;    }
  state  get_current() const          { return _current; }
};
```

```
int main() {
  // Verovatnoca
  double p;
  // Broj pogresno unetih pinova
  int ip = 1;
  // Objekat dijagrama stanja
  statechart sc;
  // Pocetno seme generatora slucajnih brojeva
  srand(time(NULL));
  // Postavljamo pocetno stanje
  sc.initial(reading_card);
  cout << "READING CARD" << endl;
  // Prolazimo kroz sva stanja dijagrama stanja dok ne dodjemo do finalnog
  while(!sc.get_final()) {
    if(sc.get_current()==reading_card) {
      p = random();
      if(p<=0.05) {
        sc.transition(card not_readable);
        sc.set_final(true);
      }
      else {
        sc.transition(card_readable);
      }
    }
    else if(sc.get current()==reading_pin) {
      p = random();
      if(p<=0.05) {
        if(ip<3) {
          sc.transition(invalid_pin);
          ++ip;
        }
        else {
          sc.set_final(true);
        }
      }
      if(p<=0.25) {
        sc.transition(cancel);
        sc.set_final(true);
      }
      else {
        sc.transition(pin_read);
      }
    }
    else if(sc.get_current()==choosing_transaction) {
      p = random();
      if(p<=0.20) {
        sc.transition(cancel);
        sc.set_final(true);
      }
      else {
        sc.transition(transaction_chosen);
      }
    }
    else if(sc.get_current()==performing_transaction) {
      p = random();
      if(p<=0.40) {
        sc.transition(another_transaction);
      }
      else {
        sc.transition(customer_finished);
        sc.set_final(true);
      }
    }
  }
  return 0;
}
```