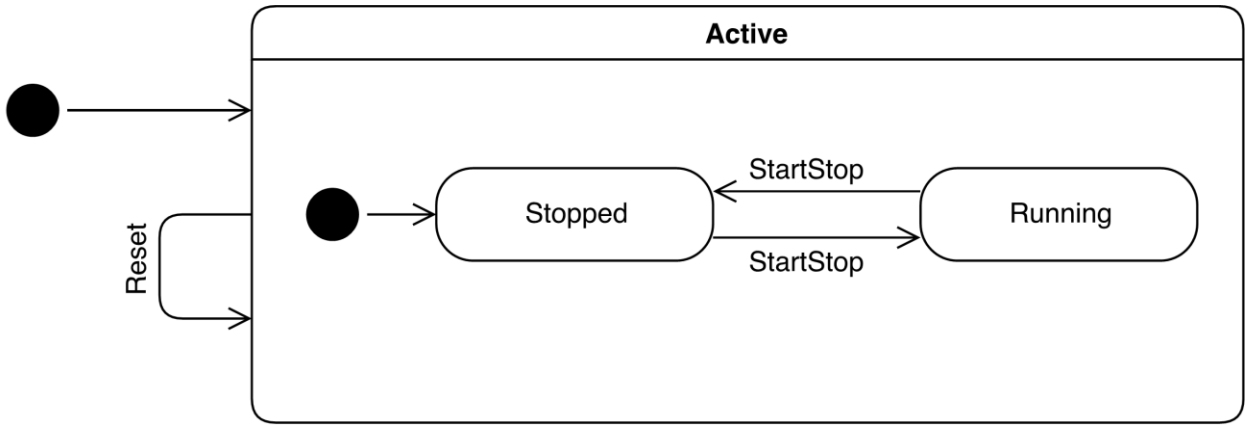
Primer: Na sledećem dijagramu stanja je prikazan rad štoperice. Inicijalno, nakon uključivanja štoperice ona prelazi u aktivno stanje (Active). Kada uđe u aktivno stanje štoperica prelazi u interno stanje zaustavljena (Stopped). Pritiskom na Start/Stop dugme štoperica iz zaustavljenog stanja prelazi u stanje merenja vremena (Running). Ponovnim pritiskom na Start/Stop dugme štoperica se vraća u zaustavljeno stanje. Pritiskom na dugme Reset štoperica iz aktivnog stanja vraća se u aktivno stanje sa resetovanjem merenja vremena na 0.



Rešenje:

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

using namespace std;

// Dijagram stanja
class statechart;

// Dogadjaji
enum event {
  reset,
  startstop
};

// Klasa stoperice
class stopwatch {
  // Pokayivac na objekat dijagrama stanja
  statechart *_sc;
public:
  // Konstruktor
  stopwatch();
  // Destruktor
  ~stopwatch();
public:
  // Ukljuci stopericu
  void turn_on();
  // Resetuj stopericu
  void reset();
  // Start/Stop dugme
  void start_stop();
  // Vrati vreme
  double get_time() const;
};

// Interni dijagram stanja
```

```
class istatechart {
public:
  // Stanja internog dijagrama stanja
  enum state{
    stopped,
    running
  };
private:
  // Tekuce stanje
  state _current;
public:
  // Pocetno stanje
  void initial(const state s) {
    _current = s;
  }
  // Tranzicija stanja
  void transition(const event e) {
    switch (e){
    case startstop:
      if(_current==stopped) {
        _current = running;
      }
      else if(_current==running) {
        _current = stopped;
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
    default:
      break;
    }
  }
public:
  state get_current() const { return _current; }
};

// Dijagram stanja
class statechart {
public:
  enum state {
    active
  };
private:
  // Pokazivac na interni dijagram stanja
  istatechart *_isc;
  state _current;
  double _time;
  time_t _start;
public:
  // Konstruktor dijagrama stanja
  statechart() {
    _isc = new istatechart;
    _isc->initial(istatechart::stopped);
  }
  // Destruktor dijagrama stanja
  ~statechart() {
    delete _isc;
  }
  // Inicijalno stanje
  void initial(const state s) {
    _current = s;
  }
  // Tranzicija stanja na osnovu okidanja dogadjaja
  void transition(event e) {
    switch (e){
    case reset:
      if( current==active) {
        _current = active;
        _time = 0.0;
      }
```

```
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
      break;
    case startstop:
      if(_current==active) {
         current = active;
        _isc->transition(e);
        if( isc->get current()==istatechart::running) {
          _start = std::time(nullptr);
        }
        else if(_isc->get_current()==istatechart::stopped) {
          time t stop = time(nullptr);
          _time += difftime(stop,_start);
        }
      }
      else {
        cerr << "Error" << std::endl;
        exit(1);
      }
    default:
      break;
    }
  }
public:
  state get current() const { return  current; }
  double get_time() const { return _time; }
};

// Konstruktor stoperice
stopwatch::stopwatch() {
  _sc = new statechart;
}
// Destruktor stoperice
stopwatch::~stopwatch() {
  delete _sc;
}
// Ukljuci stopericu
void stopwatch::turn_on() {
  _sc->initial(statechart::active);
}
// Resetuj stopericu
void stopwatch::reset() {
  _sc->transition(event::reset);
}
// Start/Stop dugme na stoperici
void stopwatch::start_stop() {
  _sc->transition(startstop);
}
// Vrati vreme stoperice
double stopwatch::get time() const {
  return _sc->get_time();
}

int main() {
  char d;
  // Objekat stoperice
  stopwatch sw;
  // Ukljucujemo stopericu
  sw.turn_on();
  while(true) {
    cout << "Pritisnite dugme na stoperici: ";
    cin >> d;
    if(d=='S' || d=='s') {
      // Pritisak na Start/Stop dugme
      sw.start_stop();
    }
    else if(d=='R' || d=='r') {
      // Pritisak na Reset dugme
      sw.reset();
```

```
    }
    cout << "Vreme: " << sw.get_time() << " sec" << endl;
  }
  return 0;
}
```