Obrada izuzetaka

```
enum DeviceException {deviceFaulty, communicationFailed};

Temperature readTemperature ();
Pressure readPressure ();
Humidiry readHumidity ();
…

void readMeteo () {
   try {
      Temperature temp = readTemperature();
      Pressure press = readPressure();
      Humidity hum = readHumidity();

      // Finally, do the job with temp, press, and hum
   }
   catch (DeviceException& de) {
     // Handle exceptions
   }
}
```

try blok uokviruje regularan tok u kom se može baciti izuzetak

Ukoliko se *try* blok završi bez izuzetka, završava se ceo konstrukt *try-catch* (*catch* blokovi se preskaču)

Ukoliko se tokom izvršavanja *try* bloka baci izuzetak, prekida se izvršavanje tog *try* bloka i kontrola prebacuje na *catch* blok (*exception handler*) koji prihvata izuzetak tog tipa kao svoj argument

Obrada izuzetaka

Bacanje izuzetka:

- Operatorom *throw*:

```
template<typename T>
T& vector<T>::at (int pos) {
  if (pos<0 || pos>=this->size) throw out_of_range;

  return this->array[pos];
}
```

Unutar *catch* bloka, tekući izuzetak koji se obrađuje može biti ponovo bačen operatorom *throw* bez operanda:

```
try {
  …
}
catch (…) {
  device->reset();
  throw;
}
```

- Mnoge funkcije iz standardne biblioteke jezika C++ mogu da bace izuzetke, npr. `vector::at`, `string::substr` itd, čime signaliziraju greške (nemogućnost da urade zahtevano npr. zbog nekorektnosti argumenta)
- Neki operatori ugrađeni u jezik mogu da bace izuzetke, npr. `dynamic_cast` (ako rezultat izraza koji se konvertuje nije zahtevanog odredišnog tipa za reference) ili `new` (ako ne može da alocira prostor u memoriji za objekat koji se kreira), opet signalizirajući grešku