Obrada izuzetaka

- Zbog ovog redosleda, *catch* blok koji prima objekat izvedene klase (po vrednosti ili preko reference ili pokazivača na izvedenu klasu) mora da bude ispred onog koji hvata objekat osnovne klase (po vrednosti ili preko reference ili pokazivača na osnovnu klasu), jer u suprotnom nikada neće biti aktiviran:

```
try {
  …
}
catch (Derived& e) {
  …
}
catch (Base& e) {
  …
}
```

Ovaj hvatač biće aktiviran za izuzetke tipa *Derived*…

… a ovaj za ostale koji su tipa *Base*, ali nisu *Derived*. Da je ovaj ispred prvog, uvek bi on bio aktiviran (jer je svaki *Derived* ujedno i *Base*)

- *catch(…)* hvata izuzetke bilo kog tipa i, shodno tome, mora uvek biti poslednji u nizu (inače ostali nikada ne bi bili aktivirani); ovakav hvatač može poslužiti kao obezbeđenje za to da nijedan izuzetak ne bude bačen iz date funkcije:

```
try {
  …
}
catch (ThermometerException& e) {
  …
}
catch (ManometerException& e) {
  …
}
catch (...) {
  …
}
```

Hvata izuzetke bilo kog tipa

Obrada izuzetaka

- *try-catch* konstrukt može da uokviri celo telo funkcije, uključujući i inicijalizatore podobjekta osnovne klase i članova klase za konstruktore te klase (u kojima se ovo tipično i koristi); svaki *catch* za ovakav *try* treba da se završi bacanjem izuzetka; ukoliko toga nema, isti izuzetak koji se obrađuje se implicitno baca na kraju *catch* bloka (kao sa *throw;*)

```
template <typename T>
class huge_vector : public vector<T> {
  …
}

template <typename T>
huge_vector<T>::huge_vector<T> () try : vector<T>(0x100000000) {
}
catch (bad_alloc& e) {
  cerr<<"Cannot allocate a huge vector: “<< e.what() << “\n";
}
```

- Ako se ne pronađe odgovarajući hvatač za bačeni izuzetak sve do dna steka poziva za tekuću nit, izvršava se bibliotečna funkcija *terminate* koja podrazumevano prekida izvršavanje programa (ali se to može promeniti)
- Ako se funkcija označi kao *noexcept*, onda ona implicitno hvata sve izuzetke koji eventualno nisu obrađeni unutar nje i poziva funkciju *terminate*; takva funkcija tako nikada ne baca (ne prosleđuje) izuzetke:

```
void transaction () noexcept {
  …
}
```

isto ovo se može postići i eksplicitno:

```
void transaction () noexcept {
  try {
    …
  }
  catch (...) {
    std::terminate();
  }
}
```