Introduktion till undantagshantering

Undantagshanteringen bygger på objekt och tre språkkomponenter:

- undantag är *objekt* – kan vara av godtycklig typ men normalt av *klasstyp*.
- **try**-block
  - en speciell blocksats som används för att avgränsa kod där undantag kan kastas, antingen uttryckligen av blockets satser eller indirekt genom funktioner som anropas i blocket.
- **throw**-uttryck
  - avbryter den normala programexekveringen genom att ”kasta” ett undantagsobjekt

```
throw out_of_range{"utanför tillåtet intervall"};
```

  - objektets typ avgör vad för slags undantag det är – i detta fall `out_of_range`.
- **catch**-hanterare

```
catch (const out_of_range& e)
{
    …
}
```

  - placeras efter **try**-block
  - kan fånga undantag som har kastats, direkt eller indirekt, i **try**-blocket
  - vilken typ av undantag som ska fångas anges i en *undantagsspecifikation* – `out_of_range` i exemplet
  - ett **try**-block kan ha flera hanterare

Exempel

```
…
try
{
    …
    if (value < 1 or value > MAX_VALUE)
        throw out_of_range{”otillåtet värde på ’value’!”};
    …
}
catch (const out_of_range& e)
{
    cout << e.what() << endl;
}
…
```

Standardundantagen är klassobjekt som har en medlemsfunktion `what`() – returnerar det meddelande som undantaget bär med sig.

Konvention för att definiera egna undantagsklasser

Standardundantaget `out_of_range` definieras på följande sätt (`logic_error` är ett annat standardundantag):

```
#include <stdexcept>

class out_of_range : public logic_error
{
public:
    explicit out_of_range(const string& what_arg) noexcept
        : logic_error{what_arg} {}

    explicit out_of_range(const char* what_arg) noexcept
        : logic_error{what_arg} {}
};
```

Definiera egna undantag på samma sätt.

- byt `out_of_range` mot namnet på det egna undantaget – allt annat identiskt
- all funktionalitet ärvs från `std::logic_error` – endast två egna konstruktor behöver definieras
- egna undantag kan genom detta hanteras på samma sätt som standardundantag

Funktioner och undantag

En funktion som kastar ett undantag avbryts direkt och undantaget kastas till anroparen.

- `string`-klassens `at`-funktion kastar `out_of_range` om en otillåten position anges:

```
char& string::at(size_type pos)
{
    if (pos >= size_)
        throw out_of_range{”otillåtet index”};

    return buffer[pos];
}
```

- funktionen avbryts ”mitt i”
- undantaget kastas vidare i anropet av at() – funktionsanropet avbryts

```
try
{
    …
    c = line.at(i);                 // anropet avbryts om at() kastar undantag – c tilldelas inte …
    …
}
catch (const out_of_range& e)       // … i stället hamnar vi här …
{
    cout << e.what() << endl;
}
…                                   // … och med tiden här, oavsett om undantag kastats eller inte
```