

HOOFDSTUK 6

EXCEPTION HANDLING

Helga Naessens

Voorbeeld: faculteit (versie 1)

```
int fac(int getal) {  
    int res = 1;  
    for (int i = 2 ; i <= getal ; i++)  
        res *= i;  
    return res;  
}  
  
int main() {  
    //wat komt er op het scherm?  
    cout << fac(3) << " " << fac(-3);  
}
```



RM: “ongeldige invoer”

Voorbeeld: faculteit (versie 2)

```
int fac(int getal) {  
    int res = 1;  
    for (int i = 2 ; i <= getal ; i++)  
        res *= i;  
    return getal >= 0 ? res : -1;  
}  
  
int main() {  
    int res = fac(-3);  
    if (res == -1) cout << "fout";  
    else cout << res;  
}
```

Is dit beter?
(= C-stijl)

Opmerkingen

- Als de gebruiker het resultaat niet controleert, dan gaat het programma verder met de speciale waarde (hier -1).
- Het is niet altijd mogelijk een speciale returnwaarde te vinden.
- Bij het voorgaande voorbeeld kan men eventueel voor het oproepen van de functie het getal controleren, maar dit is niet altijd mogelijk.

⇒ betere oplossing: **exception handling**

Exception handling: opwerpen van een exceptie

```
int fac(int getal) {  
    if (getal < 0)  
        throw "exceptie: negatief getal!";  
    int res = 1;  
    for (int i = 2 ; i <= getal ; i++)  
        res *= i;  
    return res;  
}
```

via throw (*type*) kan aangegeven worden welke exceptie de functie opwerpt, maar dit wordt sterk afgeraden

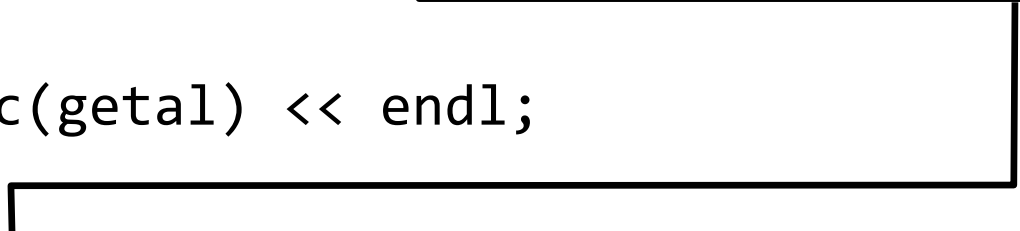
Elk type (dus ook int, char, string, ...) kan opgeworpen worden

Voorbeeld: **vb_excep1.cpp**

Opvangen van 1/meerdere specifieke exceptie(s)

```
int main() {  
    int getal;  
    cin >> getal;  
    try {  
        cout << fac(getal) << endl;  
    }  
    catch (const char *s) { cout << s << endl; }  
    return 0;  
}
```

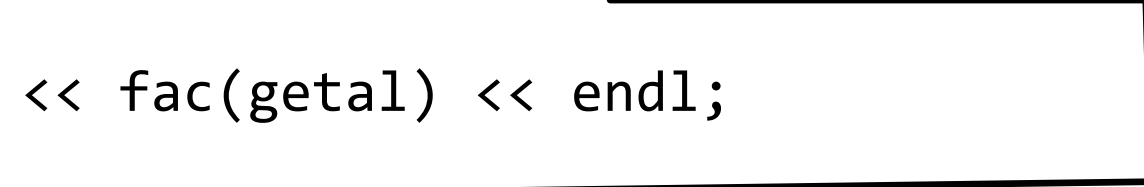
vang excepties van het
type C-string op



Voorbeeld: **vb_excep2.cpp**

Opvangen van alle excepties

```
int main() {  
    int getal;  
    cin >> getal;  
    try {  
        cout << fac(getal) << endl;  
    }  
    catch (...) { cout << "oei, een exceptie" << endl; }  
    return 0;  
}
```



catch-all handler
(vangt alle excepties op)
=> steeds laatste catch-blok

Opmerkingen

- De exceptie-declaratie in de catch-clausule hoeft enkel het exceptie-type te bevatten (niet noodzakelijk een variabele).

Voorbeeld:

```
catch (const char *) { cout << "ai ai ai"; }
```

- Sinds C++11 kan met behulp van het sleutelwoord **noexcept** aangegeven worden dat een functie géén exceptie zal opwerpen.

Voorbeeld:

```
int functie() noexcept;
```


Voorbeeld

Gevraagd

Schrijf een functie `gemidde1de(s)` waarbij `s` een gegeven standaardstring is met de *naam* van een bestand, dat een onbepaald aantal gehele getallen bevat.

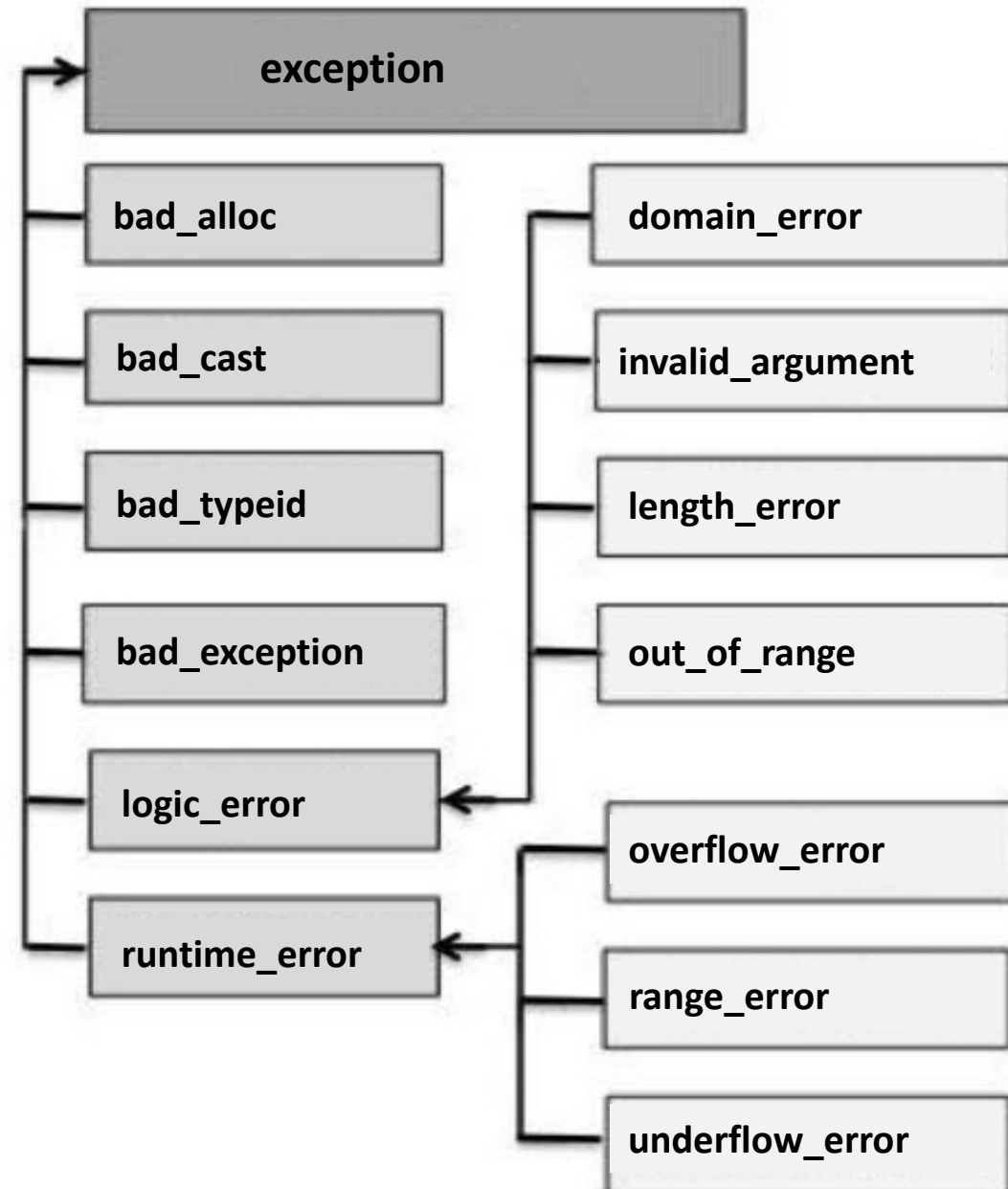
De functie bepaalt het gemiddelde van alle getallen in het bestand. De functie werpt een exceptie op indien het bestand niet kan geopend worden, indien het bestand leeg is of indien het bestand fouten bevat.

Oplossing `vb_excep3.cpp`

Exception classes

Overzicht exceptieklassen uit de standard library

- Sommige types worden door bestaande methodes opgeworpen (**out_of_range**, **invalid_argument**,...)
- Sommige types zijn bedoeld om zelf op te werpen (vb: faculteit berekenen van een negatief getal => **domain_error**)



Voorbeeld gebruik bestaande exceptieklasse

```
#include <stdexcept>    // nodig voor excepties
using namespace std;

int main() {
    vector<int> v = {8,10,12};
    try {
        int i = v.at(v.size());    // 1tje te ver
    }
    catch (const out_of_range& e) {
        cout << "Exceptie: " << e.what();
    }
}
```

Zelfgemaakte exceptieklasse

file_error.h

```
#include <stdexcept>
using namespace std;

class file_error : public runtime_error {
public:
    file_error() : runtime_error("can't open file") {}
    file_error(const string &what) :
        runtime_error(what) {}
};
```

Opmerking:

klasse exception heeft geen constructor met std/c-string als parameter \Rightarrow klasse beter niet afleiden van exception

```
#include "file_error.h"

void openFile(string fname, ifstream& in) {
    in.open(fname);
    if (!in)
        throw file_error("Can't open file " + fname);
}

int main() {
    string file_name = "test.txt"; ifstream inv;
    try {
        openFile(file_name, inv); ...
    }
    catch (const file_error& fe) { cout << fe.what(); }
}
```