## *Vad är ett undantag?*

- Oväntade fel
- Ej användarfel
  - ska kollas och promptas för nytt värde
- Fel på enheter
- Internt fel; ”bugg”
- Olika krav internt program/styrsystem





## *Problem med felhantering*

- Den som upptäcker felet vet inte vad som ska göras och den som vet vad som ska göras vet inte att det är fel.





## *Rapportering av fel*

- Avsluta programmet med textmeddelande
  - Vi kan bättre!
- Returnera värde med error-värde
  - -1 eller NULL - som sedan används felaktigt
- Returnera med fixat värde och sätt felflagga
  - i math.h: global errno - som inte kollas
- Anropa errorfunktion
  - och sen?





## *C++ Exceptions*

- Bara synkrona (typ range-check)
- Ej asynkrona (avbrott från t ex tangentbord)
- Icke-lokal returmekanism
- Avsedd för felhantering och för feltolerans
- Separerar felhanteringen från koden
  - ökar läsbarheten
- Default: Avsluta programmet
  - istf för att hoppas på det bästa
- Tar endast prestanda då fel uppträder





## *Ett exempel-Vektor med range-check*

```
class V
{
  int *p; // Stroustrup-ordning
  int sz;
public:
  class Range {} // exception class
  int& operator[](int i);
}
```





## *operator[] med kontroll*

```
int& V::operator[](int i)
{
  if (0 <= i && i < sz) return p[i];
  else throw Range();
}
```

## *Användning av vektorn*

```
main()
{
  V vek(10);
  try
  {
     vek[12] = 27; // orsakar range-error
  } catch(V::Range)
  {  // exception handler
    cerr << ”Bad index;
  }
}
```




## *Kontrollen kan kopplas bort för att höja prestandan*

```
#define RANGECHECK 0
inline int& V::operator[](int i)
{
#if RANGECHECK
  if (0 <= i && i < sz)
#end
 return p[i];
#if RANGECHECK
  else throw Range();
#end
}
```




## *Throwing-catching*

- Sök i anropskedjan efter hanterare
- Avallokera lokala variabler via destruktorer på vägen
- Ingen hanterare gör att programmet avslutas
- Bara den närmaste hanteraren hanterar




## *Fler olika feltyper*

```
class V
{ public:
  V(int sz);
  class Range {};
  class Size {};
  int& operator[](int i);
};
V::V(int sz) // konstruktor
{
  if (sz<0 || sz>max) throw Size();
}
```




## *Fångning av flera feltyper*

```
void f()
{
  try
  {
     use_vectors();
  } catch (V::Range)
  {  cerr << ”Bad index”;
  } catch (V::Size)
  {  cerr << ”Bad size to allocate”;
  }
}
```




## *Man behöver inte fånga alla*

```
void f()
{
  try
  {
    use_vectors();
  }
  catch (V::Range)
  {
  } // Trillar förbi om det är andra fel
    // eller helt rätt
}
```





2

## *Parametrar med felen*

```
class V
{ public:
  class Range // exception class
  { public:
    int index;
    Range(int i) :index(i) {} // ctor
  };
  int& operator[](int i);
}
```





## *Exception Objekt*

```
int& V::operator[](int i)
{
  if(0<=i && i< sz) return p[i];
  throw Range(i);
}

try
{
  use_vect();
} catch (V::Range r) // tar emot ett objekt
{
  cerr << ”Bad index: ” << r.index << ’\n’;
} catch(...) // matchar alla klasser
{
  cerr << ”Okänt fel i vektoranvändning”;
}
```





## *Resursallokering*

```
void use_file(const char* fn)
{
  FILE *f = fopen(fn,”w”);
  use_vector()
  fclose(f); // kanske aldrig anropas
}
```

---

```
void use_file(const char *fn)//bättre version
{
  ostream f(fn); // anrop destr vid fel
  use_vector();
}
```





## *Gränssnitt*

```
void f(int a) throw(x2,x3,x4);
// kan kasta exception x2,x3 eller x4
// annars anropas unexpected() och sedan
// terminate() och sedan abort()
```

---

```
void f2()
{
  catch(x2)
  {
    throw; // re-throw;
  }
}
```





## *Att använda Kodregler*

- Layout
  - Ordning på definitioner
  - Kommentarer
  - Indentering
  - Klamrarnas positioner
- Namnsättnnig
  - Byt inte namn
  - Versaler, gemena, kapitalisering
  - Understrykningstecken
  - Språk (Engelska/Svenska)





## *Allmänt*

- Optimera ej i början
- Undvik varningar
- Undvik #define
- Använd explicit typkonvertering (cast)
- Använd parenteser







3

## *Biblioteksdesign*

- Trädprincipen
  - Samtliga ärver från rotobjekt
- Skogprincipen
  - Grupper, ”cluster”

158



## *Ericsson-regler*

- 1992: 51 regler, 61 rekommendationer, 18 porteringstips
- r0: Regelbrott måste dokumenteras
- namnsättning på filer (.hh, .cc, inga kataloger)
- introkommentar
- kommentarer på engelska
- Copyright
- Separata include-filer för klasser

159



## *forts Ericsson-regler*

- globaler börjar med fil-prefix
- Inga medlemsfunktioner i klassdeklarationen
- Inga public eller protected datamedlemmar
- En medlem som inte ändrar klassen ska deklareras const
- ...
- r49: använd inte goto
- använd ej malloc,free

160



## *Kodregler från andra håll*

- Effective C++ av Scott Myers (ny utgåva)
- More - ” -
- C++ FAQs av Cline, Lomow
  - Copy-construktor, tilldelningsoperator, virtuell destruktor alltid tillsammans
- Advanced C++, Programming styles and idioms J.Coplien
  - Smarta pekare, Kuvert
- Trevliga lärare
- Sunt förnuft

161