TDP004 - Objektorienterad programmering

Klasser, operatoröverlagring och strängströmmar

Pontus Haglund & Rasmus Jonsson

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

1. Mål med föreläsning(arna)
2. Parameteröverföring och referenser
3. Grundläggande klasser
4. Filuppdelning
5. Operatoröverlagring
6. Fler strömmar
7. Testning

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

Mål med föreläsning(arna)

Efter föreläsningen skall studenten kunna:

- Korrekt och effektiv överföring av parameterar genom att använda const och & (referens).
- Utveckla sitt program efter TDD och Catch.
- Samla representation av data och funktioner i en klass. Samt korrekt användning av åtkomstnivå.
- Styra skapandet av klasser med den speciella medlemsfunktionen “konstruktor”
- Dela upp koden i .h och .cc filer

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

Referenser

Referenser

```
string concatenate_name(string const fn,
                        string const sn)
{
  return fn + sn;
}

int main()
{
  string first_name{"Pontus"};
  string surname{"Haglund"};
  first_name = concatenate_name(first_name,
                                surname);
}
```

Referenser

Referenser

```
void concatenate_name(string &fn,
                      string const sn)
{
    fn += sn;
}

int main()
{
  string first_name{"Pontus"};
  string surname{"Haglund"};
  concatenate_name(first_name, surname);
}
```

Referenser

Referenser

```
string concatenate_name(string const& fn,
                        string const& sn)
{
  return fn + sn;
}

int main()
{
  string first_name{"Pontus"};
  string surname{"Haglund"};
  first_name = concatenate_name(first_name,
                                surname);
}
```

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

Grundläggande klasser

Vad är en klass?

```
class Person
{
public:
    string first_name{};

private:

};
```

Grundläggande klasser

Klassmetod och datamedlem

```
class Person
{
public:
  string first_name{};

  void print()
  {
    cout << first_name << endl;
  }

private:

};
```

Grundläggande klasser

Regler för åtkomster

```
class Person
{
public:
  void print()
  {
    cout << first_name << endl;
  }

private:
  string first_name{};
};
```

Grundläggande klasser

Setters och Getters

```
class Person
{
public:
  string get_first_name() { return first_name; }

  void set_first_name(string const& fn)
  {
    first_name = fn;
  }
  void print()
  {
    cout << first_name << endl;
  }
private:
  string first_name{};
};
```

Grundläggande klasser

Konstruktor

```
class Person
{
public:
  Person(string const& fn)
  : first_name{fn}
  {}

  string get_first_name() { ... }
  void set_first_name(string const& fn) { ... }
  void print() { ... }

private:
  string first_name;
};
```

Grundläggande klasser

Konstanta metoder

```
class Person
{
public:
  Person(string const& fn)
  : first_name{fn}
  {}

  string get_first_name() { ... }
  void set_first_name(string const& fn) { ... }
  void print() const { ... }

private:
  string first_name;
};
```

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

LINKÖPING UNIVERSITY

Filuppdelning

definition och deklaration

```
//Person.h - deklaration
#ifndef PERSON_H
#define PERSON_H
class Person
{
public:
  Person(string const& fn,
         int const age);
  // ...
  void print() const;

private:
  string first_name;
  int age;
};
#endif
```

```
//Person.cc - definition
#include "Person.h"

Person::Person(string const& fn,
               int const a)
: first_name{fn}, age{a}
{}

void Person::print() const
{
  cout << first_name << endl;
}
```

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

Operatoröverlagring

Kort om operatorer

```
string lhs{"How are you "};
string rhs{"world?"};

cout << lhs + rhs;
```

Operatoröverlagring

Hur funkar en operator

```
string lhs{"How are you "};
string rhs{"world?"};

cout << lhs.operator+( rhs );
```

Operatoröverlagring

Prioritet

```
string lhs{"How are you "};
string rhs{"world?"};

cout.operator<<( lhs.operator+( rhs ) );
```

Operatoröverlagring

Utvärderas steg för steg

```
string lhs{"How are you "};
string rhs{"world?"};

cout.operator<<( "How are you world?" );
```

Operatoröverlagring

Vad gör den här jämförelsen

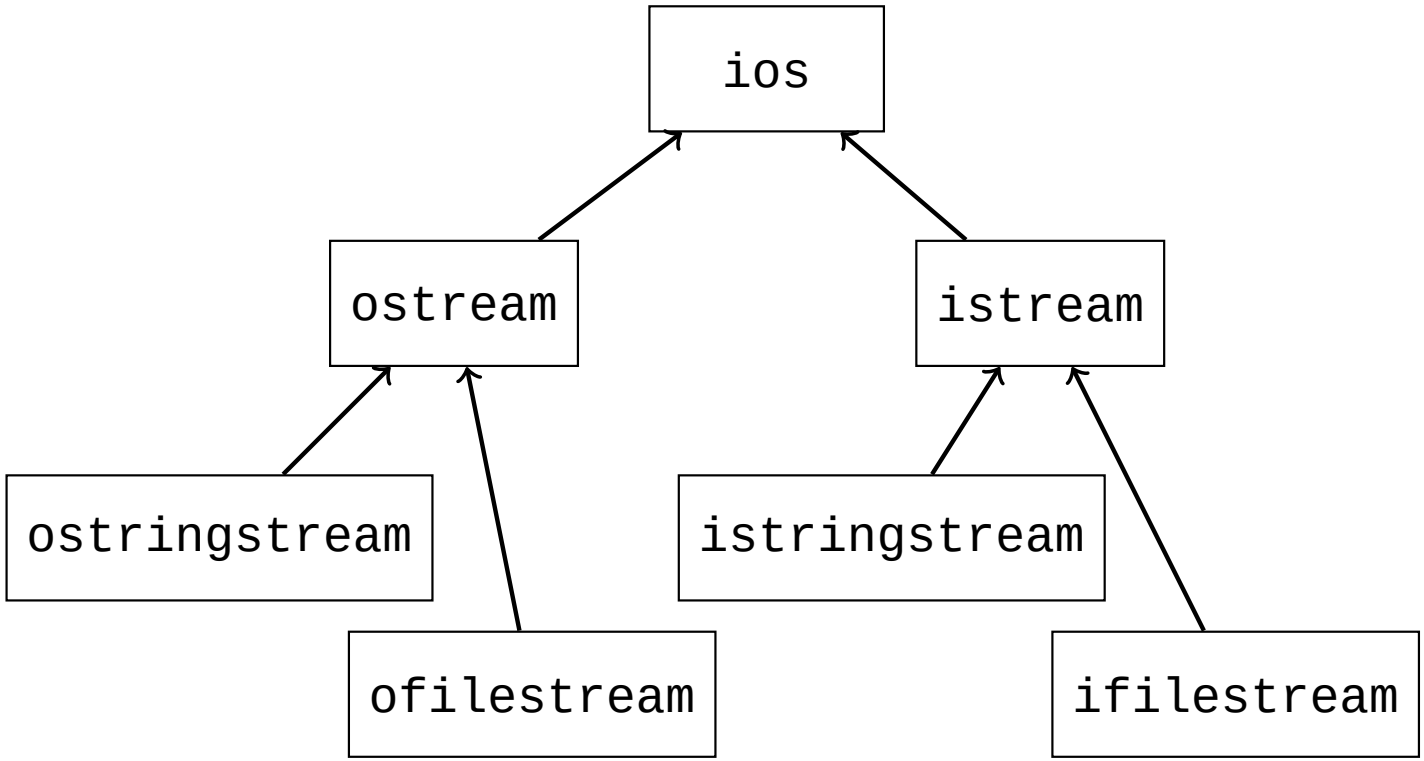
```
int main()
{
  Person p1{"Pontus", 30};
  Person p2{"János", 25};

  if (p1 < p2)
  {
    p2.print();
  }
  else
  {
    p1.print();
  }
}
```

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

LINKÖPING UNIVERSITY

Fler strömmar

Fler strömmar



Vilken TYP är cin, cout, cerr...

- https://en.cppreference.com/w/cpp/io/basic_istream
  - cin är av denna typ
- https://en.cppreference.com/w/cpp/io/basic_ostream
  - cout är av denna typ

Titta i dokumentationen ovan för att se vad man kan göra med cin och cout.

iostate

ios_base::iostate flags			basic_ios accessors					
eofbit	failbit	badbit	good()	fail()	bad()	eof()	operator bool	operator!
false	false	false	true	false	false	false	true	false
false	false	true	false	true	true	false	false	true
false	true	false	false	true	false	false	false	true
false	true	true	false	true	true	false	false	true
true	false	false	false	false	false	true	true	false
true	false	true	false	true	true	true	false	true
true	true	false	false	true	false	true	false	true
true	true	true	false	true	true	true	false	true

1 Mål med föreläsning(arna)
2 Parameteröverföring och referenser
3 Grundläggande klasser
4 Filuppdelning
5 Operatoröverlagring
6 Fler strömmar
7 Testning

LINKÖPING UNIVERSITY

Catch

```
//person_test.cc
#include "catch.hpp"
#include "Person.h"

TEST_CASE( "Test constructor" ) {
  Person p {"Pontus"};
  REQUIRE( p.get_first_name() == "Pontus");
}

TEST_CASE( "Test setter" ) {
  Person p {"Pontus"};
  p.set_first_name("Janos");
  REQUIRE( p.get_first_name() == "Janos");
}
```

Filer för att testa med Catch2

- `catch.hpp`
- `test_main.cc`
- Testfilen och tillhörande filer.

Snabbare kompilering

Kompilera en .o fil först, detta bör bara göras en gång

```
$ w++17 -c test_main.cc
```

Därefter kan ni kompilera med följande

```
$ w++17 test_main.o person_test.cc Person.cc
```

www.liu.se

LINKÖPING UNIVERSITY