TDP004 - Objektorienterad programmering

Arv, polymorfi, argument och felhantering

Pontus Haglund & Rasmus Jonsson

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

1. Mål med föreläsningen
2. Vector
3. Arv
4. Polymorfi
5. Argument till main
6. Felhantering och arv
7. Interface

1 Mål med föreläsningen
2 Vector
3 Arv
4 Polymorfi
5 Argument till main
6 Felhantering och arv
7 Interface

LINKÖPING UNIVERSITY

Mål med föreläsningen

Efter föreläsningen skall studenten kunna:

- Använda `vector`.
- Förstå och arbeta med arv.
- Korrekt användning av polymorfi.
- Hantera kommandoradsargument.
- Mer filseparering.
- Konvertera mellan datatyper.
- Felhantering.

1 Mål med föreläsningen
2 Vector
3 Arv
4 Polymorfi
5 Argument till main
6 Felhantering och arv
7 Interface

Vector

vector

```
#include <vector>

int main()
{
  vector<int> empty_vector{};
  vector<int> my_vector{1,2,3,4,5};

  my_vector.push_back(6);
  int my_int{ my_vector.pop_back() };

  cout << "The size is: "
       << my_vector.size()
       << ", the element on pos 3 is :"
       << my_vector.at(3) << endl;
}
```

Vector

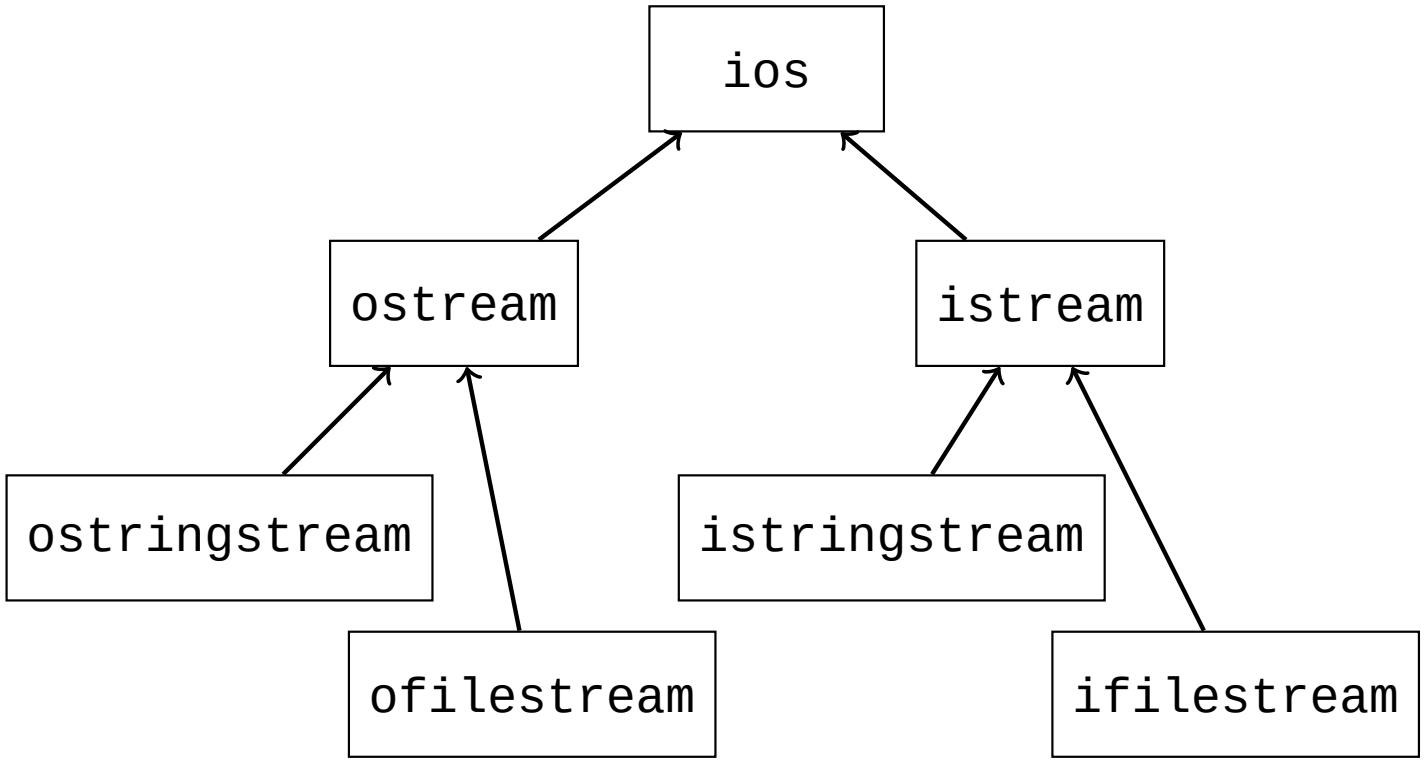
Range baserad for-loop

```
#include <vector>

int main()
{
  vector<int> my_vector{1,2,3,4,5};

  for( int i : my_vector )
  {
    cout << i << ", ";
  }
  cout << endl;
}
```

LINKÖPING UNIVERSITY

Arv

Kommer ni ihåg strömmar



Arv

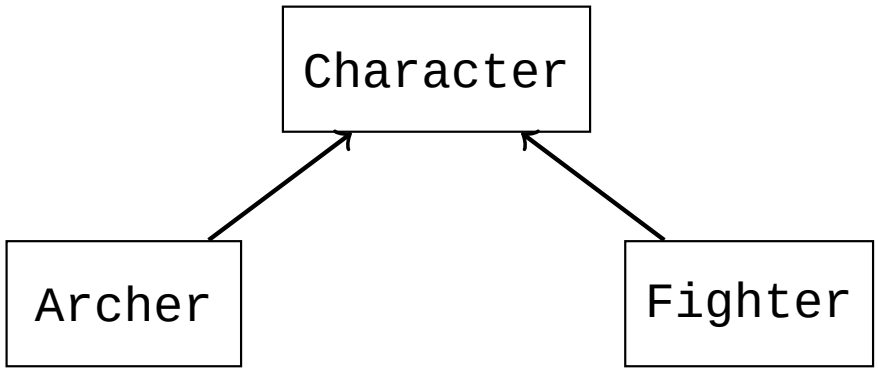
Spelobjekt

`Archer`

`Fighter`

Arv

Gemensam basklass



Arv

Nyckelord

- Basklass
- Specialisering
- `virtual`
- `override`
- pure virtual
- virtual destructor

1 Mål med föreläsningen
2 Vector
3 Arv
4 Polymorfi
5 Argument till main
6 Felhantering och arv
7 Interface

LINKÖPING UNIVERSITY

Polymorfi

Delade typer

```
int main()
{
  Archer my_archer{};
  Fighter my_fighter{};

  Character  my_character{my_fighter};
  Character* character_ptr{&my_archer};
}
```

Polymorfi

Pekare i en vector

```
int main()
{
  Archer my_archer{};
  Fighter my_fighter{};

  vector<Character*> characters{};
  characters.push_back(&my_archer);
  characters.push_back(&my_fighter);
}
```

Polymorfi

Spelloop

```
for (Character* character: characters)
{
  character -> update();
}
```

1 Mål med föreläsningen
2 Vector
3 Arv
4 Polymorfi
5 Argument till main
6 Felhantering och arv
7 Interface

Argument till main

Argument till main

```
int main(int argc, char* argv[])
{
  cout << "nr of arguments: " << argc << endl;
  cout << argv[0] << " : " << argv[1] << endl;
}
```

Argument till main

Typkonvertera argumenten

```
int main(int argc, char* argv[])
{
  cout << "nr of arguments: " << argc << endl;

  istringstream iss{argv[1]};
  double argument1{};
  iss >> argument1;
  cout << argument1 << endl;

  int argument2{ stoi(argv[2]) };
  cout << argument2 << endl;
}
```

Argument till main

Felhantering

```
int main(int argc, char* argv[])
{
  cout << "nr of arguments: " << argc << endl;
  try
  {
    int argument1{ stoi(argv[1]) };
  }
  catch (invalid_argument const& ia_error)
  {
    cerr << "Error: " << ia_error.what() << endl;
    return 1;
  }
  cout << argument1 << endl;
}
```

LINKÖPING UNIVERSITY

Skapa eget fel

https://en.cppreference.com/w/cpp/header/exception

```
class My_Error : std::logic_error
{
  public:
  using logic_error::logic_error;
  //Överlagra lämpliga delar

  const char* what () const noexcept override {
    return "My_Error: ";
  }

};
```

LINKÖPING UNIVERSITY

Interface (gränssnitt)

```
class Stack_Interface
{
  public:
  virtual int top() const = 0;
  virtual int& top() = 0;
  virtual void pop() = 0;
  virtual void push(int) = 0;
  virtual int size() const = 0;
  virtual bool empty() const = 0;
};
```

www.liu.se

LINKÖPING UNIVERSITY