TDDC76

Kursupplägg och Intro till C++

Eric Ekström & Klas Arvidsson

Institutionen för datavetenskap

LINKÖPING UNIVERSITY

LINKÖPING UNIVERSITY

TDDC76: Syfte

Lära sig

- grunderna i C++
- testdriven utveckling
- objektorientering i C++
- datastrukturer och algoritmer
- att jobba i större programmeringsprojekt

TDDC76: Kursmoment

- 8 Föreläsningar
- 4 Laborationer (egen tid + 11 schemalagda pass)
- 4 Seminarier
- Datastrukturer och algoritmer genom OpenDSA (egen tid)
- Projekt (HT2)

TDDC76: Examination

- **PRA1** - 3hp Projekt
- **LAB2** - 3.5hp Laborationer
- **UPG3** - 1.5hp Datorbaserade inlämningsuppgifter (OpenDSA)

För betyg 3 i kursen ska alla tre moment vara godkända.

Högre betyg fås genom att samla bonuspoäng under kursens gång.

TDDC76: Högre betyg

- Betyg 3: alla moment godkända
- Betyg 4: 20 poäng
- Betyg 5: 40 poäng

Poäng samlas genom att ligga i fas i kursen och göra extrauppgifter.

Fullständig lista med möjliga poäng finns på kurshemsidan.

TDDC76: Personal

- Examinator - Klas Arvidsson (klas.arvidsson@liu.se)
- Kursledare - Eric Ekström (eric.ekstrom@liu.se)
- Assistenter - Maria, Nils, Lorenz, Isak, Eric
- Prata med oss!
- Kontaktuppgifter på hemsidan!

TDDC76: Utvärdering och Återkoppling

Evaliuate (svarsfrekvens 22%):

- Helkursbetyg: 4.07
- Rimlig arbetsbelastning: 75% (42% 2021)
- Otydliga labinstruktioner*
- OpenDSA känns bortkopplat från resten av kursen
- För många föreläsare!

TDDC76: Utvärdering och Återkoppling

Åtgärder:

- Innehållet på kurshemsidan ses över
- Större ändringar skjuts till 2024 då vi uppdaterar examinationsmoment och inför bedömningskiterier.
- En föreläsare per föreläsning!

TDDC76: Verktyg och hemsida

Kursen använder inte lisam. Istället används ett antal verktyg:

- Kurshemsida: https://www.ida.liu.se/~TDDC76/
- WebReg - Resultat och labb-feedback
- Sendlab - Inlämning av labbar

1 Kursinformation

2 Grunderna i C++

- Utströmmar
- Kompilering
- Variabler
- Inströmmar
- Villkor

3 Avslut

Vad är C++?

- Ett programspråk baserat på C
- Designas av en kommitté
- Kompilerat språk
- Skapat av Bjarne Stroustrup på 80-talet
- Uppdateras regelbundet
  (c++11, c++14, **c++17**, c++20, c++23)

Varför C++?

- C++ används ofta när prestanda är viktigt (operativsystem, spel, inbyggda system, etc)

Varför lära ut C++?

- Populärt språk med med bred användning
- Inkluderar många olika koncept
- Finns mycket resurser att tillgå
- Både lågnivå och stöd för abstraktion

Varför C++?

- C++ används ofta när prestanda är viktigt (operativsystem, spel, inbyggda system, etc)

Varför lära ut C++?

- Populärt språk med med bred användning
- Inkluderar många olika koncept
- Finns mycket resurser att tillgå
- Både lågnivå och stöd för abstraktion
- Efter C++ kommer andra språk vara lätta att lära sig

Ett enkelt program i C++

C++

```cpp
#include <iostream>

int main()
{
  std::cout << "Hello World" << std::endl;
}
```

Python

```python
print("Hello World")
```

Ett enkelt program i C++

C++

```cpp
#include <iostream>

int main()
{
  std::cout << "Hello World" << std::endl;
}
```

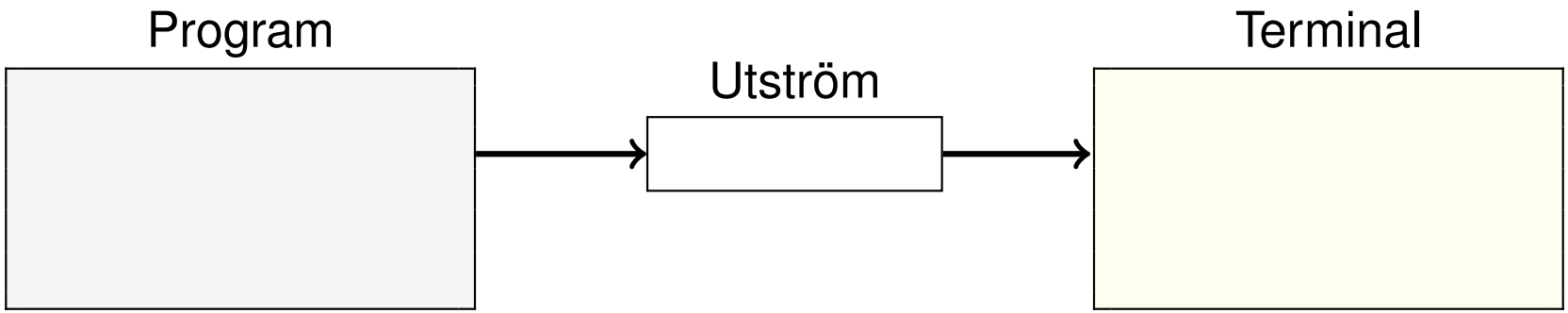
Python

```python
print("Hello World")
```

- Instruktioner avslutas med **;**
- Alla instruktioner skrivs inom main()
- Mer explicit än Python
- Indentering i Python == Måsvingar i C++

Utströmmar

Det finns en utström från programmet till terminalen som den kan kommunicera genom



Varför har vi en buffer (utström)?

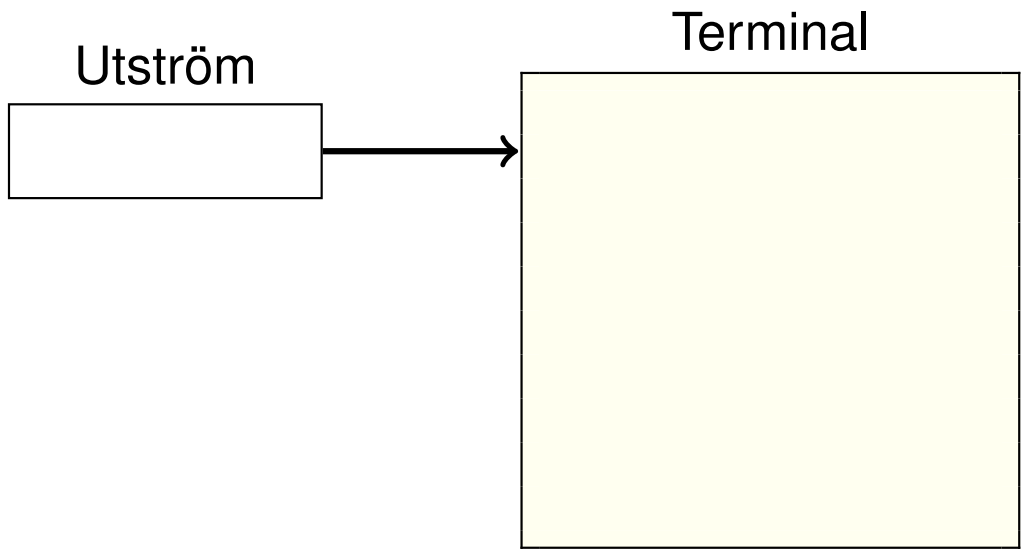
Utströmmar

- `std::cout` kan vi använda för att kommunicera med utströmmen.
- med `operator<<` skickar vi data till strömmen.
- `std::flush` tömmer strömmen till terminalen.
- `std::endl` lägger en nyrad i strömmen och gör en `std::flush`.
- Ex. `std::cout << "Hello World" << std::endl;`

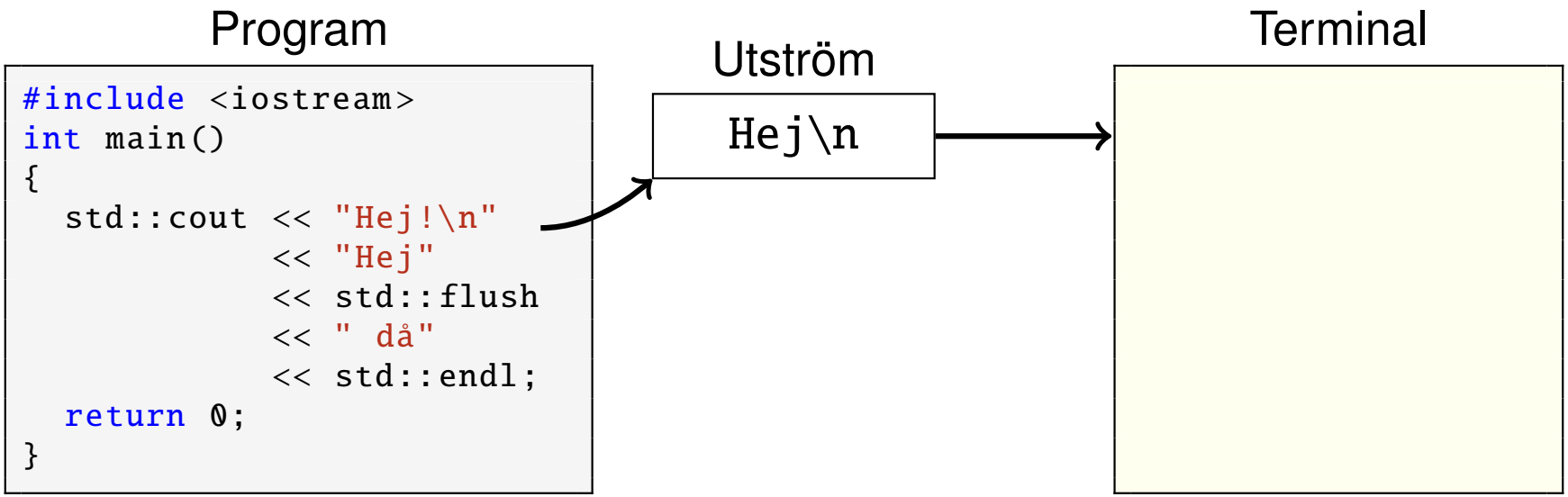
Utströmmar

Program

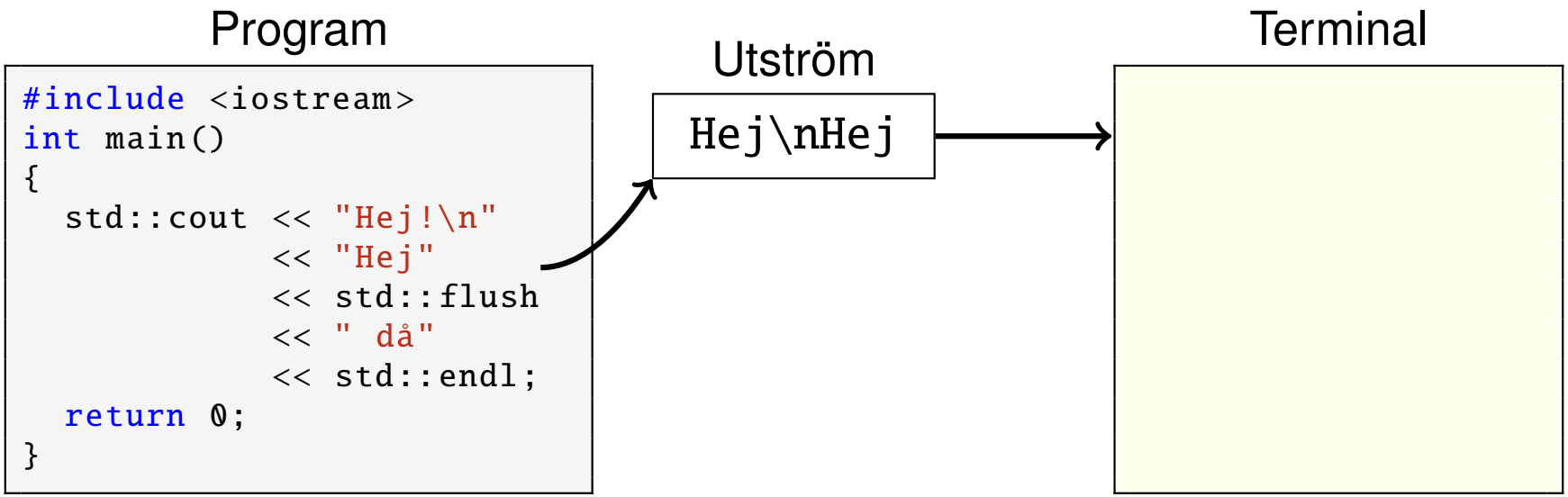
```
#include <iostream>
int main()
{
  std::cout << "Hej!\n"
            << "Hej"
            << std::flush
            << " då"
            << std::endl;
  return 0;
}
```



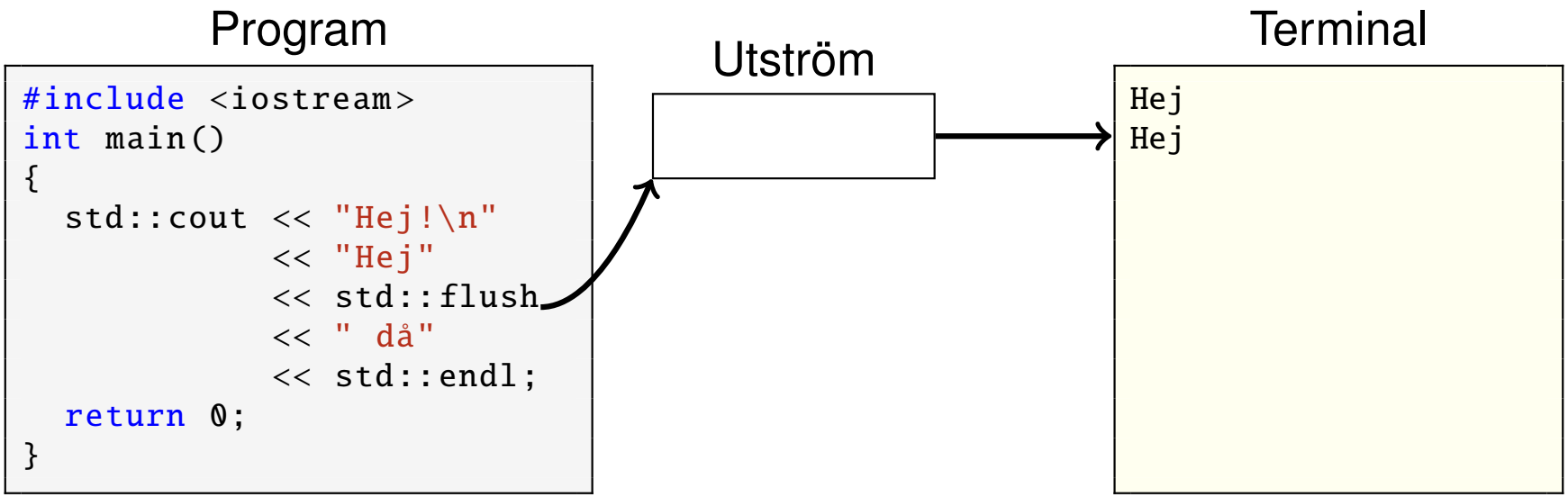
Utströmmar


Program
#include <iostream>
int main()
{
std::cout << "Hej!\n"
<< "Hej"
<< std::flush
<< " då"
<< std::endl;
return 0;
}
Utström
Hej\n
Terminal

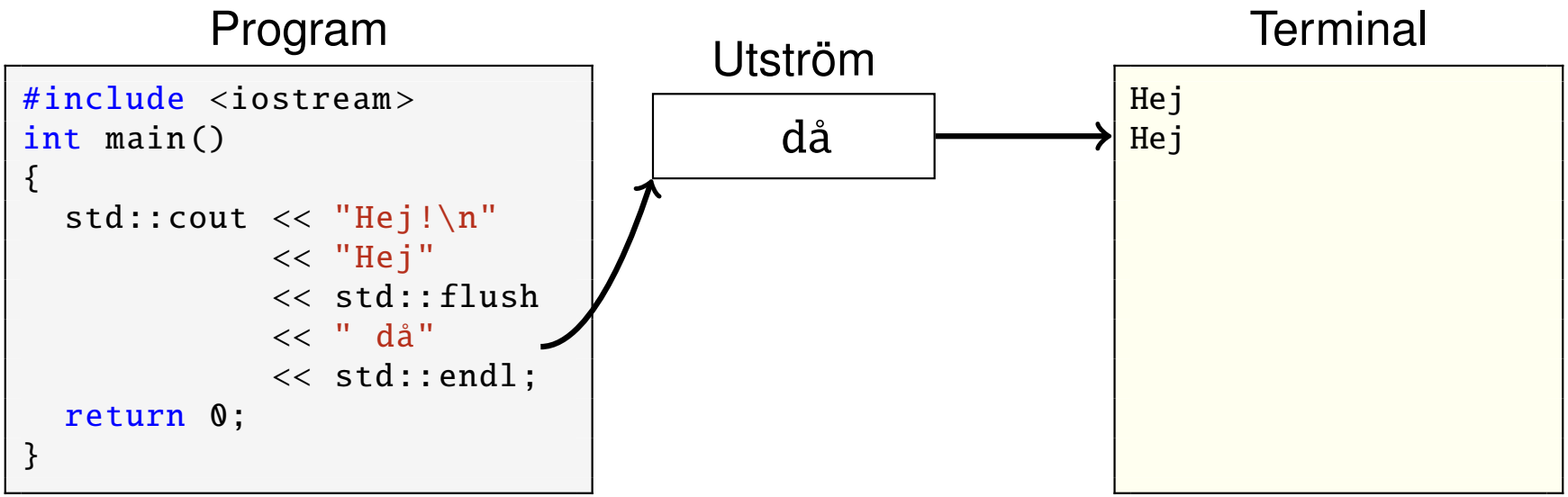
Utströmmar


Program
#include <iostream>
int main()
{
std::cout << "Hej!\n"
<< "Hej"
<< std::flush
<< " då"
<< std::endl;
return 0;
}
Utström
Hej\nHej
Terminal

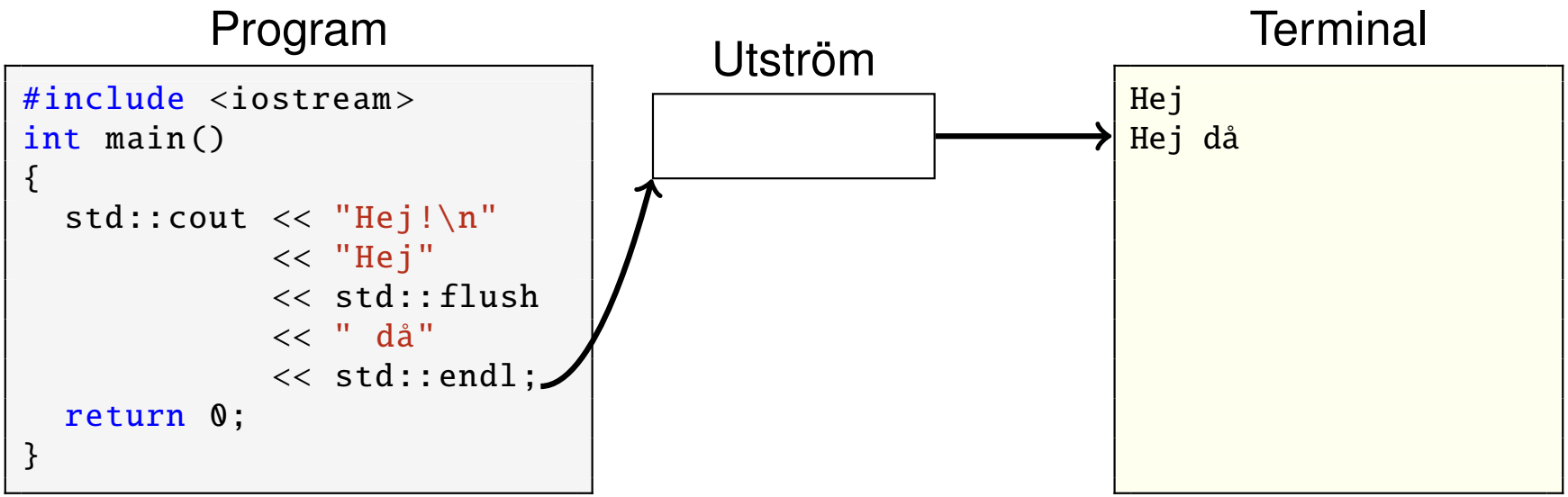
Utströmmar


Program
Utström
Terminal
#include <iostream>
int main()
{
std::cout << "Hej!\n"
<< "Hej"
<< std::flush
<< " då"
<< std::endl;
return 0;
}
Hej
Hej

Utströmmar


Program
#include <iostream>
int main()
{
std::cout << "Hej!\n"
<< "Hej"
<< std::flush
<< " då"
<< std::endl;
return 0;
}
Utström
då
Terminal
Hej
Hej

Utströmmar


Program
Utström
Terminal
#include <iostream>
int main()
{
std::cout << "Hej!\n"
<< "Hej"
<< std::flush
<< " då"
<< std::endl;
return 0;
}
Hej
Hej då

Utströmmar

Vi kan formatera utskriften m.h.a funktioner från `std::iostream` och `std::iomanip`.

- `fixed` - Använd decimalpunkt och N=6 decimaler
- `setprecision(N)` - Använd N decimaler
- `setw(N)` - Använd minst N tecken för nästa utskrift
- `setfill(C)` - Använd tecknet C som utfyllnadstecken
- `left` - Vänsterjustera utskriften (inom `setw`)
- `right` - Högerjustera utskriften (inom `setw`)

Utströmmar

Vi kan formatera utskriften m.h.a funktioner från `std::iostream` och `std::iomanip`.

```
#include <iostream>
#include <iomanip>

int main()
{
  std::cout << std::setw(6)
            << std::setfill('.')
            << 10 << std::endl;
  return 0;
}
```

```
....10
```

Hur kör vi vårt program?

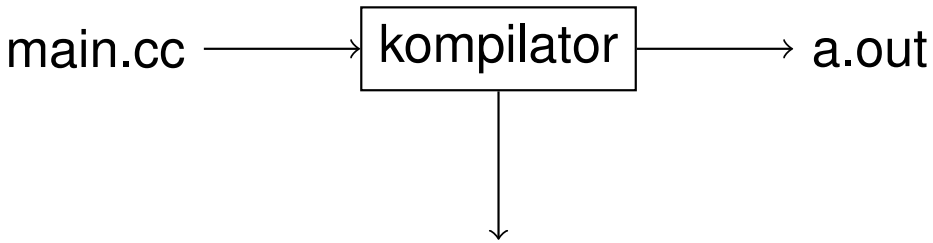
C++ är ett kompilerat och typat språk.

Hur kör vi vårt program?

C++ är ett kompilerat och typat språk.

Hur kör vi vårt program?

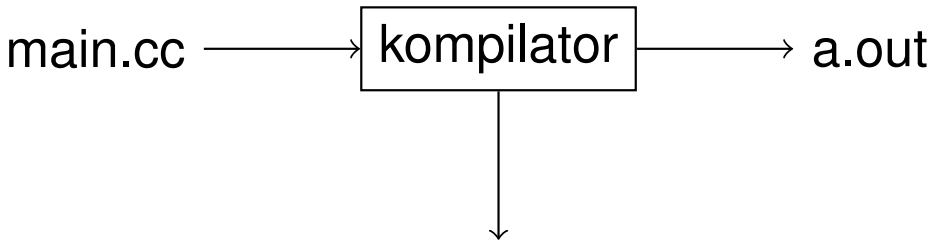
C++ är ett kompilerat och typat språk.




```
main.cc: In function int main():
main.cc:8:9: error: cannot convert std::string to int in assignment
    8 |      x = num;
      |          ^~~
      |          |
      |          std::string
```

Hur kör vi vårt program?

C++ är ett kompilerat och typat språk.




```
main.cc: In function int mian():
main.cc:6:9: warning: unused variable x [-Wunused-variable]
    6 |     int x {};
      |         ^
/usr/bin/ld: /usr/lib/gcc/.../Scrt1.o: in function `_start':
(.text+0x1b): undefined reference to `main'
collect2: error: ld returned 1 exit status
```

Hur kör vi vårt program?

Kompilatorn vi använder heter `g++`.

```
$ ls
hello.cc

$ g++ -std=c++17 hello.cc

$ ls
hello.cc a.out

$ ./a.out
Hello World!
```

Jobbigt att skriva `-std=c++17` varje gång? Skapa ett alias!

Ett *lite* mer komplicerat program i C++

C++

```cpp
#include <iostream>
int main()
{
  std::cout << "Vilket heltal är bäst? "
            << std::flush;
  int number {};
  std::cin >> number;

  if (number == 76)
  {
    std::cout << "Korrekt" << std::endl;
  }
  else
  {
    std::cout << "Fel" << std::endl;
  }
  return 0;
}
```

Python

```python
number = input("Vilket heltal är bäst? ")

if number == 76:

  print("Korrekt")

else:

  print("Fel")
```

Datatyper och Variabler

För att spara data under programkörning behöver vi lista ut ett par saker:

- Var i minnet placerar vi data?
- Hur hittar vi den datan senare?
- Hur mycket plats behövs i minnet?
- Hur tolkas datan?

Datatyper och Variabler

Variabler i C++ har alltid

- ett unikt namn (a-Z, 0-9, _)
- ett värde (även om vi inte har angett något!)
- en specifik datatyp. De vanligaste är

`int`	Heltal	(-7, 10, 2147483647)
`float, double`	Flyttal	(14.7, -7.1, 0.99)
`char`	Tecken	(’j’, ’7’, ’\n’)
`std::string`	Text	(“Hej”, “jag”, “heter...”)
`bool`	Sanningsvärde	(true, false)

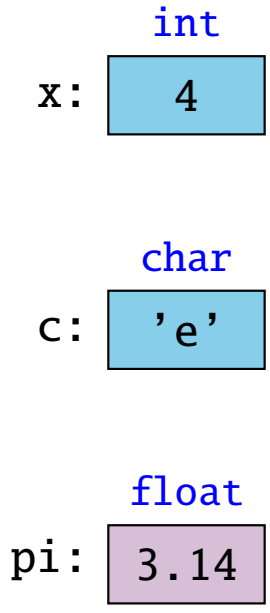
Datatyper och Variabler

```cpp
#include <string>

int main()
{
  int x { 4 };
  char c { 'e' };
  float const pi { 3.14 };
  std::string erics_passwd { "c++ärbäst" };




  return 0;
}
```



- Datatypen måste alltid anges när vi skapar variabler.
- Vi initialiserar variabler med {}

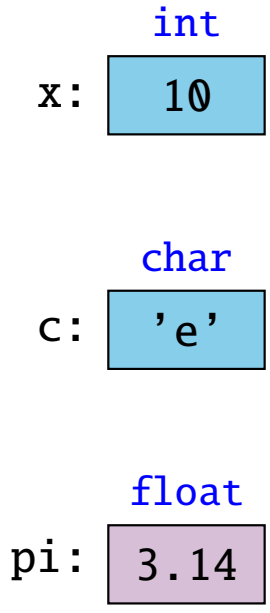
Datatyper och Variabler

```
#include <string>

int main()
{
  int x { 4 };
  char c { 'e' };
  float const pi { 3.14 };
  std::string erics_passwd { "c++ärbäst" };

  x = 10;


  return 0;
}
```



- Datatypen måste alltid anges när vi skapar variabler.
- Vi initialiserar variabler med {}

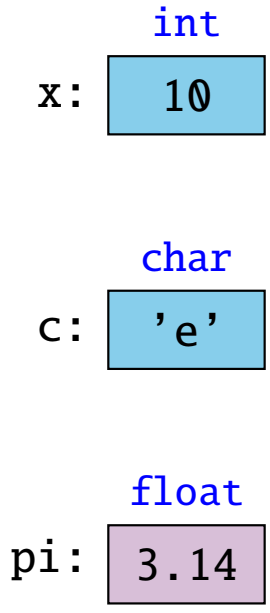
Datatyper och Variabler

```
#include <string>

int main()
{
  int x { 4 };
  char c { 'e' };
  float const pi { 3.14 };
  std::string erics_passwd { "c++ärbäst" };

  x = 10;
  pi = 9;

  return 0;
}
```



- Datatypen måste alltid anges när vi skapar variabler.
- Vi initialiserar variabler med {}

Datatyper och Variabler

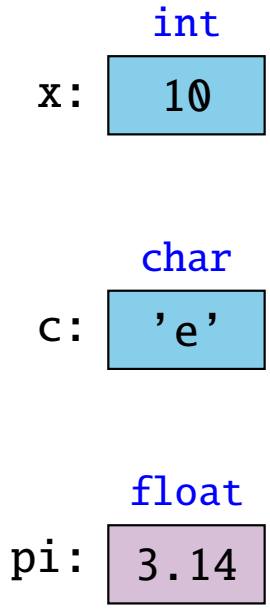
```
#include <string>

int main()
{
  int x { 4 };
  char c { 'e' };
  float const pi { 3.14 };
  std::string erics_passwd { "c++ärbäst" };

  x = 10;
  pi = 9;

  return 0;
}
```

KOMPILERAR EJ!



- Datatypen måste alltid anges när vi skapar variabler.
- Vi initialiserar variabler med {}

Datatyper och Variabler

C++

```cpp
#include <string>

int main()
{
  double diam {4};
  std::string pi { "3.14" };
  float c { pi * diam };
  return 0;
}
```

KOMPILERAR EJ!

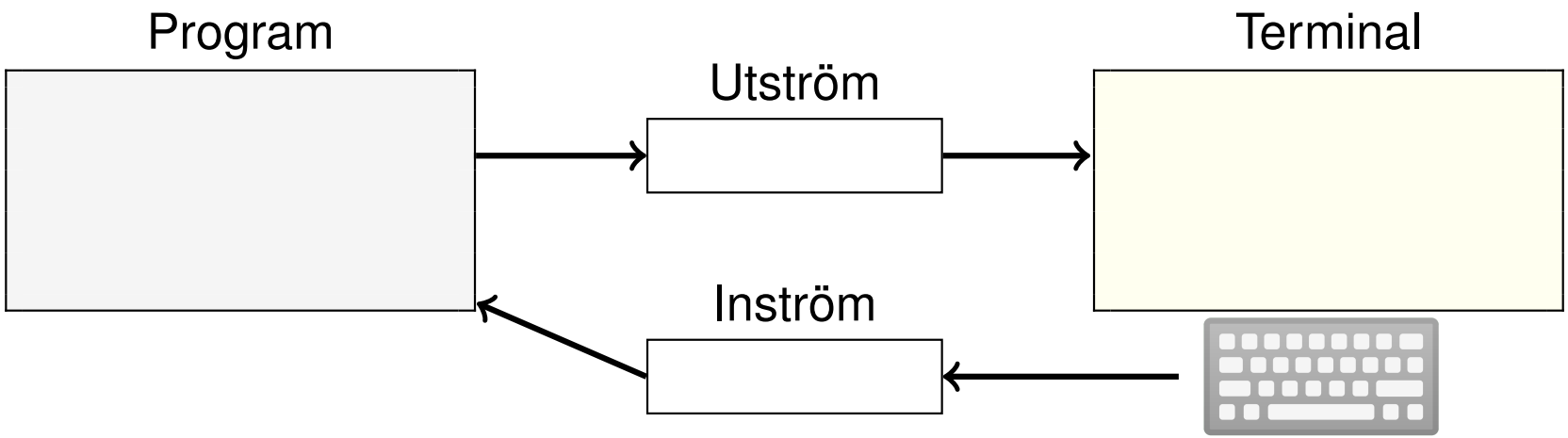
Python

```python
diam = 4
pi = "3.14"
c = pi * diam
print(c) # = "3.143.143.143.14"
```

- I C++ kan vi inte tilldela en datatyp till en annan datatyp.
- Det går inte heller att byta datatyp på en variabel.
- Varför är detta “bättre” än python?

Inströmmar

Det finns även en inström från terminalen till programmet som vi kan kommunicera genom.



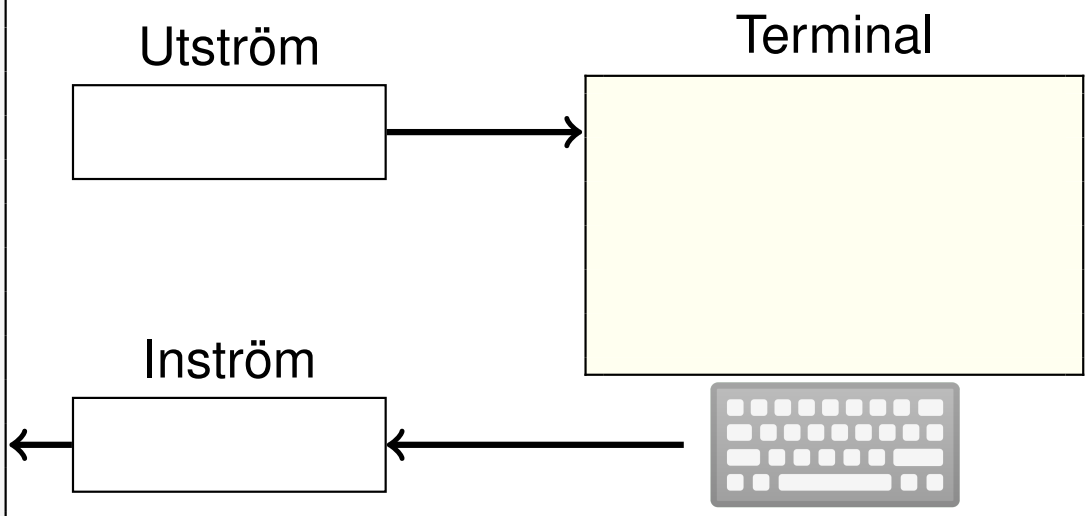
Inströmmar

- `std::cin` kan vi använda för att kommunicera med inströmmen.
- med `operator>>` läser vi data från strömmen.
- Ex. `std::cin >> x;`
- Vad kan stå på höger sida om »?

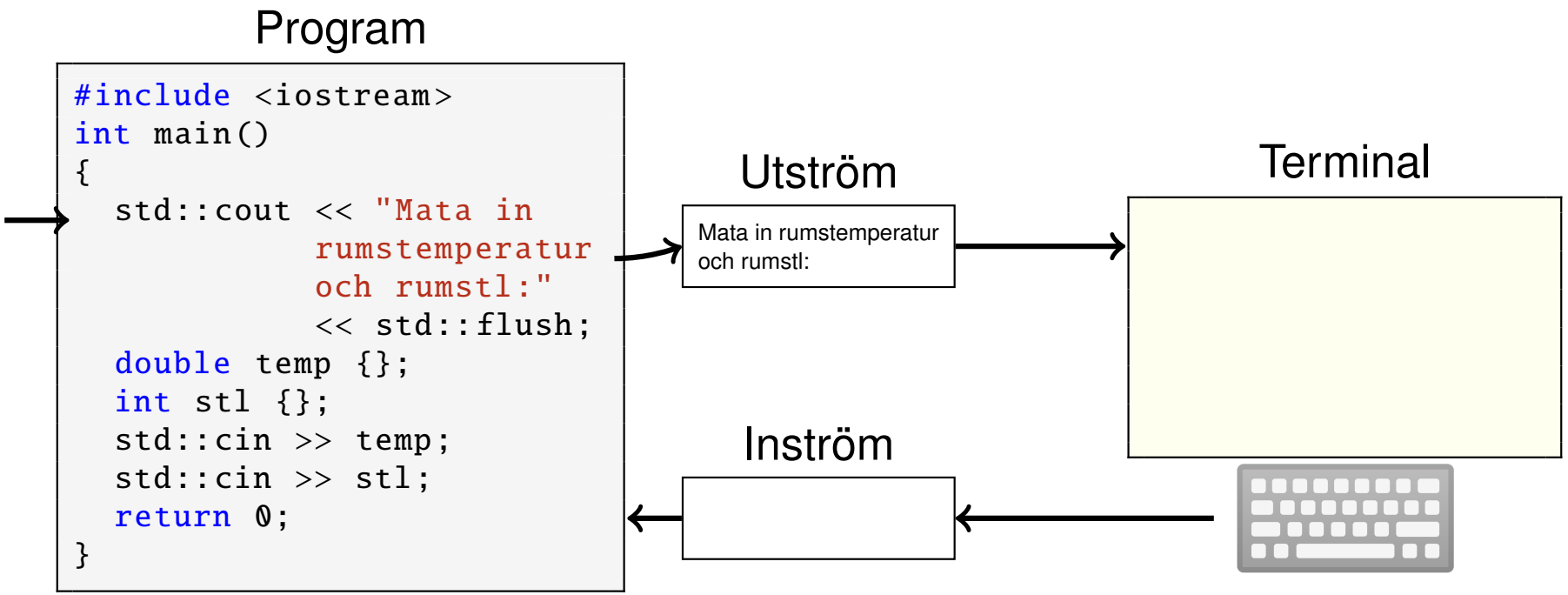
Inströmmar

Program

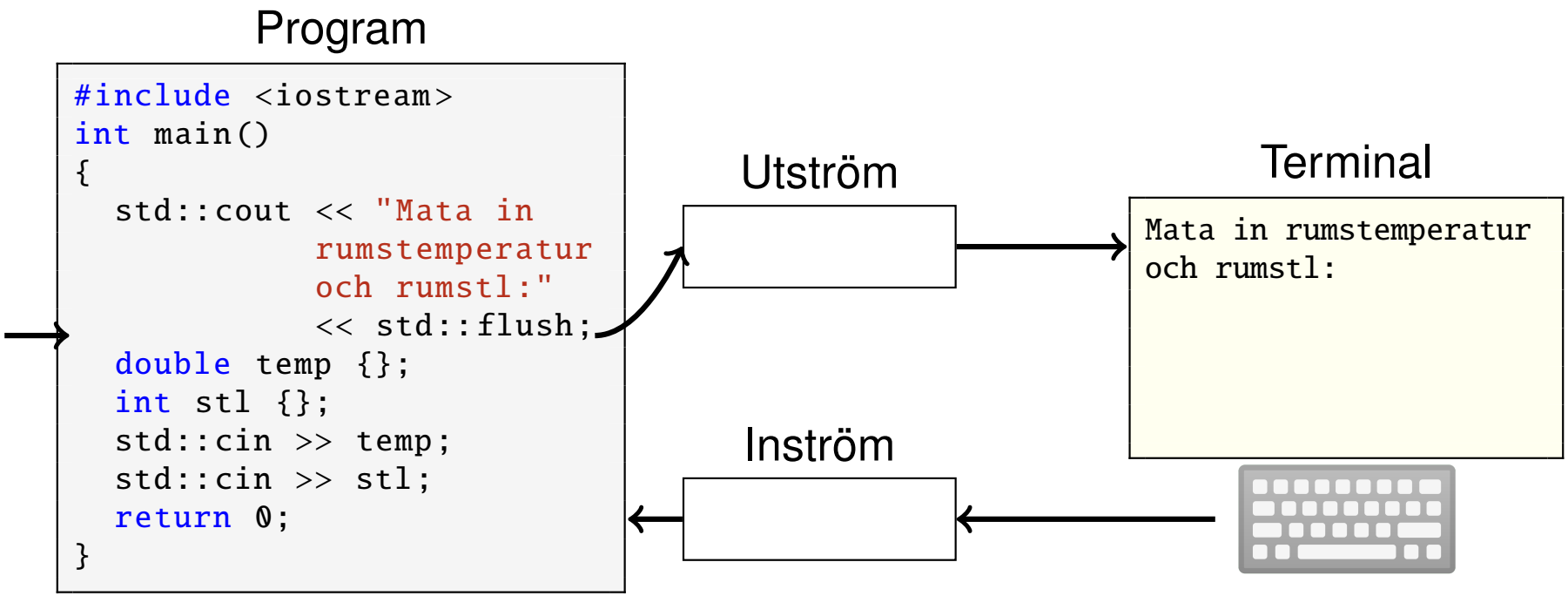
```
#include <iostream>
int main()
{
  std::cout << "Mata in
            rumstemperatur
            och rumstl:"
            << std::flush;
  double temp {};
  int stl {};
  std::cin >> temp;
  std::cin >> stl;
  return 0;
}
```



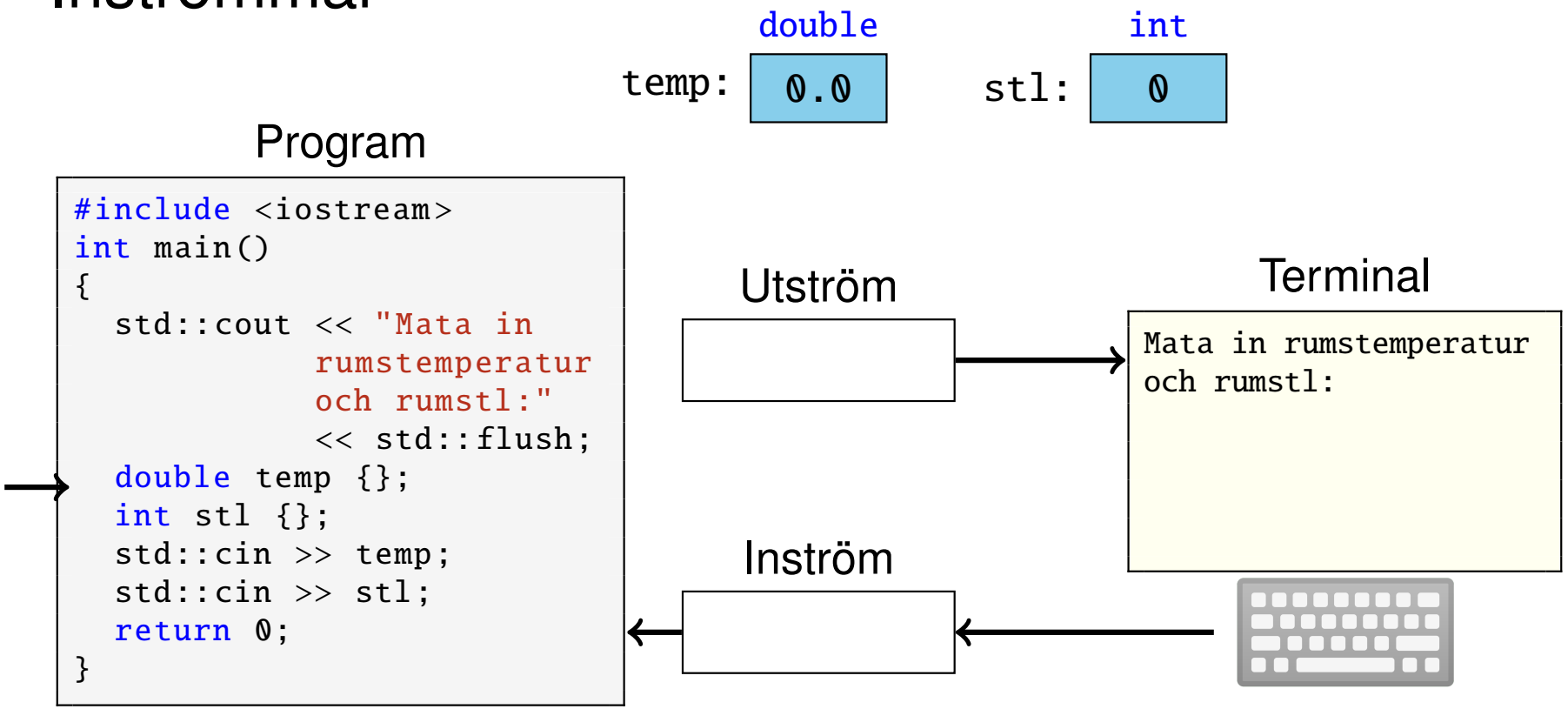
Inströmmar


Program
#include <iostream>
int main()
{
std::cout << "Mata in
rumstemperatur
och rumstl:"
<< std::flush;
double temp {};
int stl {};
std::cin >> temp;
std::cin >> stl;
return 0;
}
Utström
Mata in rumstemperatur
och rumstl:
Terminal
Inström

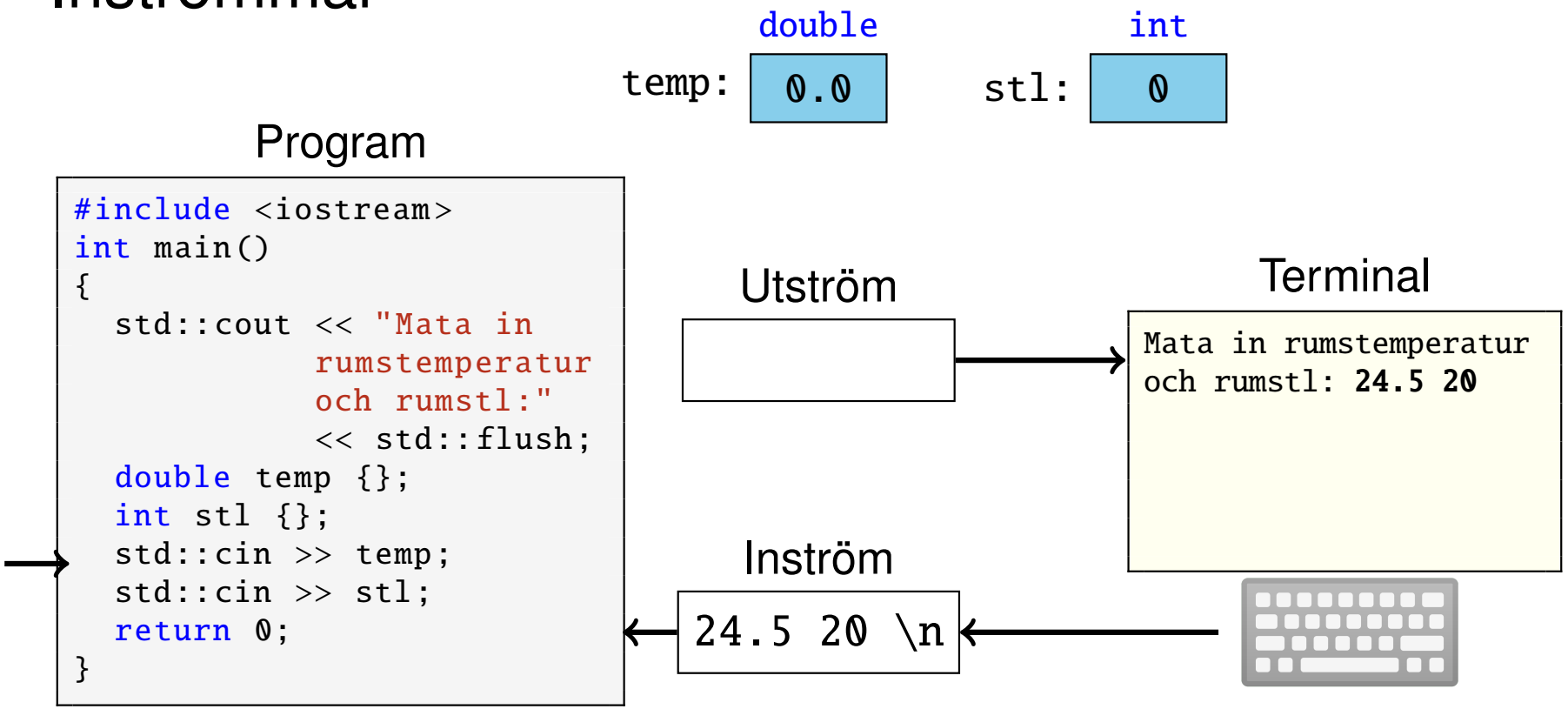
Inströmmar


Program
#include <iostream>
int main()
{
std::cout << "Mata in
rumstemperatur
och rumstl:"
<< std::flush;
double temp {};
int stl {};
std::cin >> temp;
std::cin >> stl;
return 0;
}
Utström
Terminal
Mata in rumstemperatur
och rumstl:
Inström

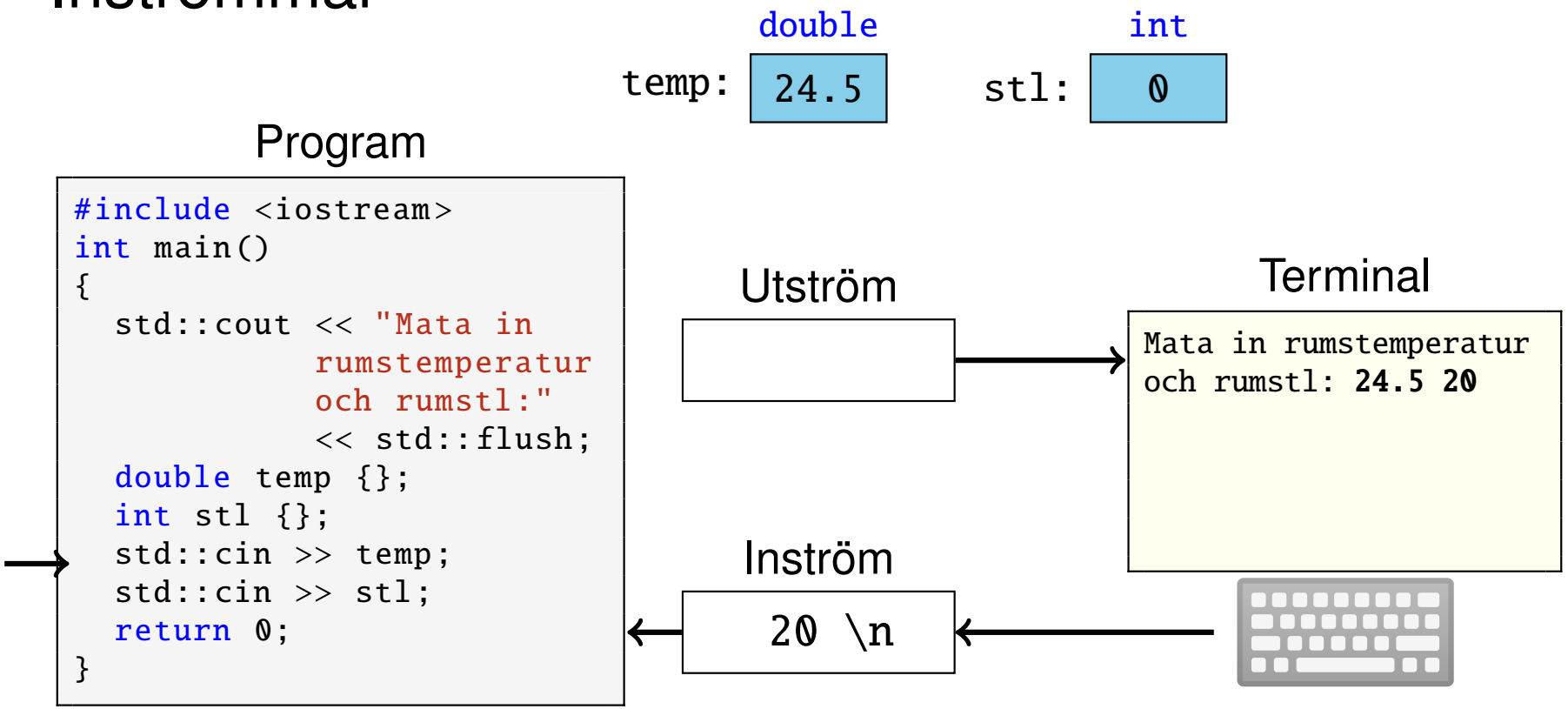
Inströmmar


double
int
temp:
0.0
stl:
0
Program
#include <iostream>
int main()
{
std::cout << "Mata in
rumstemperatur
och rumstl:"
<< std::flush;
double temp {};
int stl {};
std::cin >> temp;
std::cin >> stl;
return 0;
}
Utström
Terminal
Mata in rumstemperatur
och rumstl:
Inström

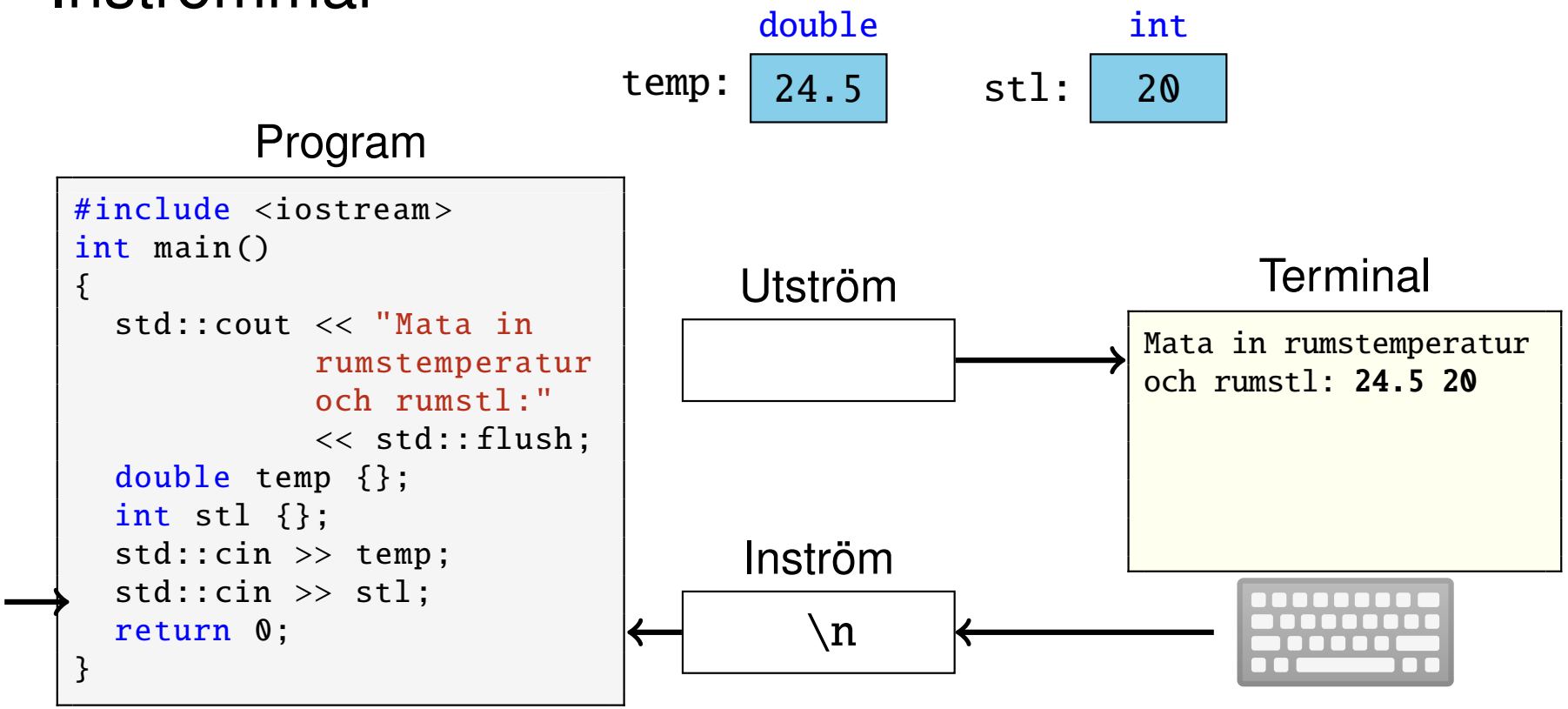
Inströmmar


double
int
temp:
0.0
stl:
0
Program
#include <iostream>
int main()
{
std::cout << "Mata in
rumstemperatur
och rumstl:"
<< std::flush;
double temp {};
int stl {};
std::cin >> temp;
std::cin >> stl;
return 0;
}
Utström
Terminal
Mata in rumstemperatur
och rumstl: 24.5 20
Inström
24.5 20 \n

Inströmmar


double
int
temp:
24.5
stl:
0
Program
#include <iostream>
int main()
{
std::cout << "Mata in
rumstemperatur
och rumstl:"
<< std::flush;
double temp {};
int stl {};
std::cin >> temp;
std::cin >> stl;
return 0;
}
Utström
Terminal
Mata in rumstemperatur
och rumstl: 24.5 20
Inström
20 \n

Inströmmar


double
int
temp:
24.5
stl:
20
Program
#include <iostream>
int main()
{
std::cout << "Mata in
rumstemperatur
och rumstl:"
<< std::flush;
double temp {};
int stl {};
std::cin >> temp;
std::cin >> stl;
return 0;
}
Utström
Terminal
Mata in rumstemperatur
och rumstl: 24.5 20
Inström
\n

Villkor

```
int main()
{
  std::cout << "Vad är temperaturen? "; // Ingen flush??
  int temp {};
  std::cin >> temp;
  if (temp >= 40)
  {
    std::cout << "Är du i en bastu?" << std::endl;
  }
  return 0;
}
```

Villkor

```
int main()
{
  std::cout << "Vad är temperaturen? "; // Ingen flush??
  int temp {};
  std::cin >> temp;
  if (temp >= 40)
  {
    std::cout << "Är du i en bastu?" << std::endl;
  }
  else
  {
    std::cout << "Låter alldeles lagom" << std::endl;
  }
  return 0;
}
```

Villkor

```
int main()
{
  std::cout << "Vad är temperaturen? "; // Ingen flush??
  int temp {};
  std::cin >> temp;
  if (temp >= 40)
  {
    std::cout << "Är du i en bastu?" << std::endl;
  }
  else if (temp < 15)
  {
    std::cout << "Maila studentbostäder" << std::endl;
  }
  else
  {
    std::cout << "Låter alldeles lagom" << std::endl;
  }
  return 0;
}
```

Misslyckade inläsningar

Kan vi kontrollera om en inläsning gått bra?

```
int temp {};
if ( std::cin >> temp )
{
  std::cout << "Du matade in temperaturen " << temp << std::endl;
}
else
{
  std::cout << "Tyvärr, du matade inte in ett giltigt heltal"
            << std::endl;
}
```

Vad gör vi nu...

Mer om inläsning

- `std::cin.clear();`
  Nollställer felflaggan så vi kan fortsätta läsa.
- `std::cin.ignore(5, '\n');`
  Rensar strömmen på max 5 tecken, eller stannar vid första nyradstecknet.
- `std::getline(std::cin, str, '\n');`
  Läser en hel rad tecken (fram till nyradstecknet) och sparar dessa i variabeln `str`.

LINKÖPING UNIVERSITY

Vad har vi gått igenom?

- Kompilering
- Inmatning
- Utmatning
- Variabler
- Villkor

Vad har vi kvar för att lösa första labben?

- Iteration
- Funktioner
- Operatorer

Resurser och Informationssökning

Hur hittar man information om C++?

- Givet material i kursen (se kurshemsidan)
- Rekommenderad bok om C++
- Fråga en lärare (i sal eller på mail)
- Sök på `cppreference.com`

Vanskliga alternativ (ditt mål är att *lära* dig C++, *inte* att producera lablösning)

- Fråga kompisar “varför...?” inte “hur...?”
- Googla? Om du ställer frågan “varför...?” inte “hur...?”
- chatGPT? Om du ställer frågan “varför...?” inte “hur...?”

Finn fem fel!

```
#include <string>
int main() {
  std::cout << "Skriv in två heltal: " << std::endl;
  int x; int y {};
  std::cin << x << y;
  if ( std::cin )
  {
    if x < y
      std::cout << x << " är mindre än" << y;
  }
  else
  {
    std::cout << "inläsningen misslyckades" << std::endl;
  }
}
```

Finn minst nio fel!

- string inkluderas istället för iostream
- x deklareras utan måsvingar
- pilarna i strömoperatorn för std::cin går åt fel håll
- if-satsen saknar parenteser och måsvingar
- det saknas en std::flush på andra utskriften
- två satser ligger på samma rad
- det saknas en return 0 i slutet av main
- koden är för kompakt skriven
- med mera...

Hur felsöker vi vår kod?

```
#include <iostream>

int main()
{
    float temp { -6 };
    unsigned ice_thickness { 0 };

    if (ice_thickness =! 0)
        if (temp < -5)
        {
          std::cout << "Safe to pimpla" << std::endl;
        }
    else
        std::cout << "Bring a boat" << std::endl;
    return 0;
}
```

Slut för idag

Registrera er på kursen och anmäl er i WebReg!

www.liu.se

LINKÖPING UNIVERSITY