

Velkommen til TDT4102

Prosedyre- og objektorientert programmering, våren 2017!

**Vi er mange: pakk godt sammen
på benkeradene ! ☺**

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

The C++ logo, consisting of a large blue 'C' followed by two blue plus signs, is superimposed over the C++ code snippet.

Lasse Natvig
www.ntnu.edu/employees/lasse

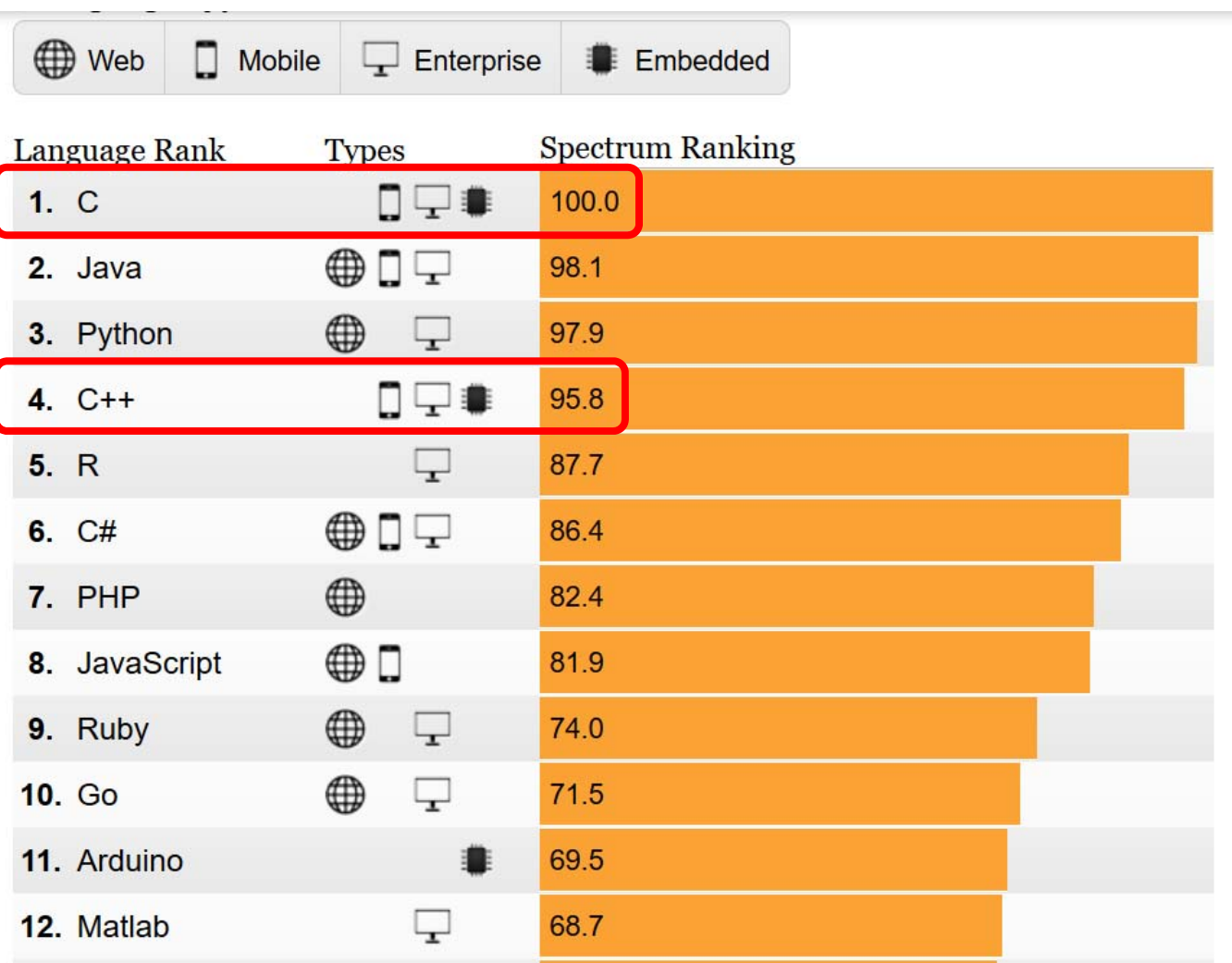
Dagens forelesning

- Om faget
- Praktisk informasjon
- Læringsmål for i dag
 - Bli kjent med faget
 - Litt om C++, programmering, og kompilering
 - Komme i gang med programmering
 - Bruk av verktøy

Programmering og programmeringsspråk

- I dette faget skal du lære å **programmere**
 - beskrive hva datamaskinen skal gjøre for deg
- Programmeringsspråket du skal lære er **C++**
 - Standardisert, mye benyttet
 - kan brukes på mange plattformer
 - representativt både for imperative språk og objekt-orienterte språk
 - (mye felles med f.eks. Java og C#)
 - **Det viktigste programmeringsspråk for de aller fleste av dere!! (se neste foil)**

Programmeringsspråk topp 12



Fra <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016>

Læringsutbytte

- Kunnskap
 - Syntaks og konstruksjoner i **programmeringsspråket**
 - Kjennskap til anvendelsesområder, begrensinger og underliggende teori
- Ferdigheter
 - **Beherske programmering som metode og ferdighet**
 - Kan løse enkle og mer komplekse programmeringsoppgaver
 - Kan skrive kode som er oversiktlig, ryddig og feilfri
 - Kan videreutvikle deg som programmerer og raskt lære seg andre programmeringsspråk
- I tillegg skal du ha kompetanse i
 - Bruk av **programmeringsverktøy**
 - **Å finne løsninger på problemer på en systematisk og iterativ måte.**

Forkunnskaper

- IT-GK
 - Informasjonsteknologi, grunnkurs (TDT4105 eller TDT4110)
 - eller tilsvarende
- Forutsetter
 - noe kunnskap om en datamaskins oppbygging
 - noe erfaring i programmering (tilsv. TDT4105/TDT4110)
- Men,
vi starter med det helt grunnleggende og gir
et ”komplett” innføringskurs i programmering
- Det er stor variasjon i den enkeltes utgangspunkt!
 - Undervisningen er tilpasset dette
 - Flere ulike tilbud

Prosedural- og objektorientert programmering

- I første omgang fokuserer vi på **prosedural programmering**
 - løse problemer ved å **tenke sekvensielt** og bruke **funksjoner**
- Senere i semesteret skal vi se på **objektorientert programmering**
 - organisere koden ved hjelp av **klasser**
 - samhandling mellom **objekter**
- Forskjellen ligger i hvordan større programmer bygges opp av mindre deler
- VIKTIG: «Mestre kompleksitet» → **abstraksjon**

Faglig opplegg

- Forelesinger
 - introduksjon til teorien
 - eksempler
 - Kahoot (litt, nytt i år)
- Øvingene og du selv
 - lære selv
 - innarbeide ferdigheter og forståelse
- Øvingsforelesninger
- «Hjelp i R1» – ett nytt ekstra-tilbud
- Ekstra tilbud for de ivrigste
 - **Kattis** og Climbing Mont Blanc (CMB)

Programmering krever både kunnskap og ferdigheter!

- Kunnskap og forståelse opparbeides over tid.
- Ferdigheter i programmering bygges med erfaring (øvingsopplegget el. annen programmering) og innsikt (forståelse/refleksjon).
- Den som har bred og solid erfaring gjør det bestandig bra på eksamen!

Studieteknikk

- Kombiner **teori** og **utprøving** i praksis
- Ikke forvent at du skal kunne lage og forstå komplisert kode uten videre
- Bygg opp koden din **del for del** og **undersøk** og **test ut** hvordan den oppfører seg
- **Vær nysgjerrig og utprøvende!**
- Bruk forelesingene, boka og forelesningsnotatene til å få svar

Lærebok

Walter Savitch, **Absolute C++, 6th ed.**

*Hele boken er pensum!
Forøvrig er alt forelesnings- og
øvingsmateriale (tekster, lysark og
løsningsforslag) pensum.*

Merk at det finnes mange andre bøker som
dekker samme stoff og som dere gjerne må
benytte i tillegg. (Men *ikke* ha med på eksamen)

Det finnes også mye nyttig stoff om C++ på
web!



Tillatt hjelpemiddel på eksamen

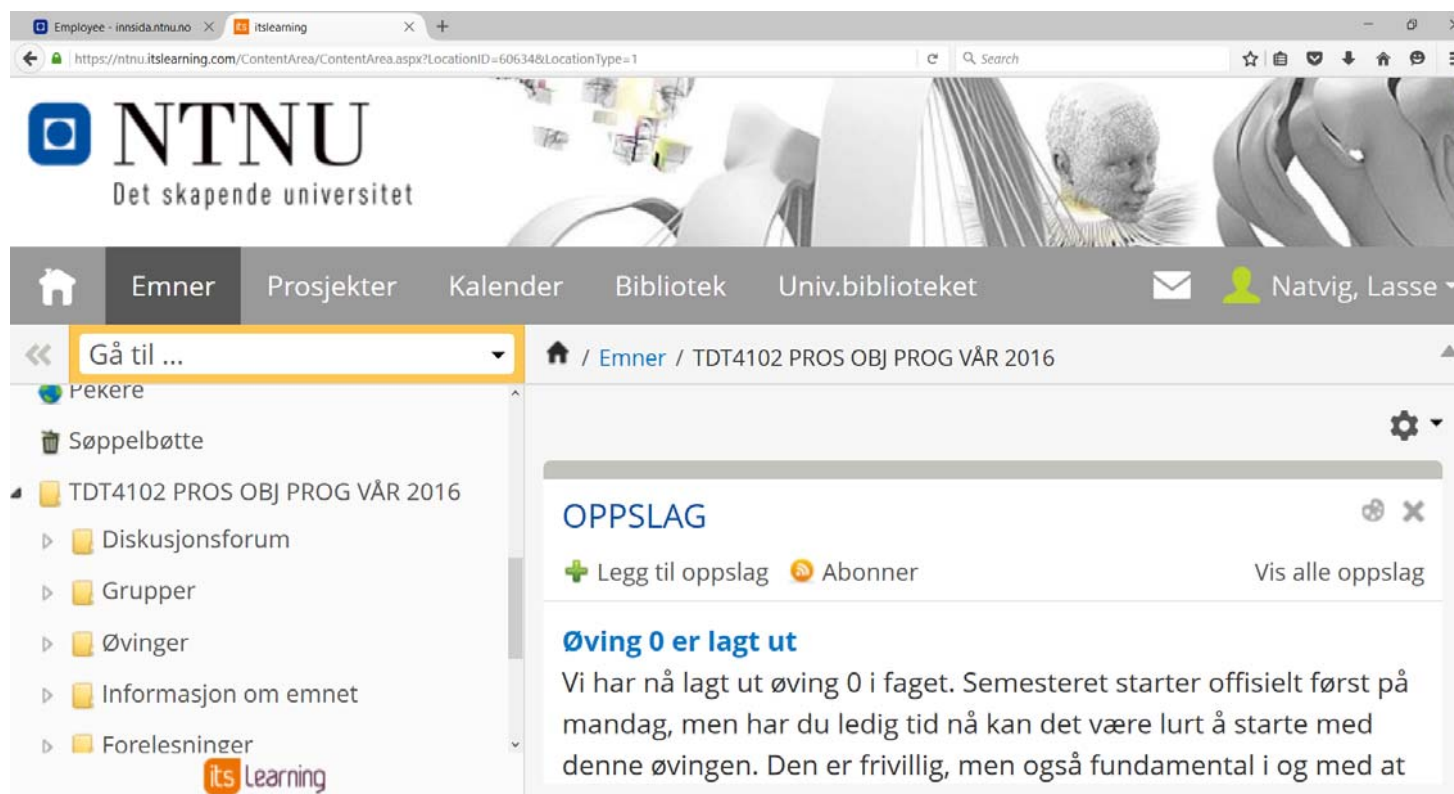
- Hjelpemiddel kode: C
- Som betyr: "Spesifiserte trykte og håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt."
- Vi tillater at du har med Absolute C++

Egne notater i boka?

- Streng kontroll på eksamen!
- Boken du har med på eksamen skal være **uten egne** notater/kommentarer
- Greit å bruke "utguling" eller plast-lapper uten tekst
- *Ikke lurt å kjøpe en kommentert bruktbok...*
- *Vanskelig å selge bok med notater*

Fagsider

- Vi bruker It's Learning!
- Spørsmål og svar
 - **piazza** diskusjonsforum!
 - Hjelp hverandre!
 - «Studentene selv er NTNUs største pedagogiske kraft!»



Fagstab

- Faglærer/foreleser

- Lasse Natvig

- Professor ved IDI (1997)
 - Siv.ing. Datateknikk IDB, NTH, 1982.
 - E-mail: Lasse@computer.org
 - Forskningsinteresse: Energy Efficient Computing Systems
<https://www.ntnu.edu/ime/eecs>
og spesielt CMB-prosjektet.



- Vit.ass Sivert Krøvel

- Stipendiater

- Johannes H Jensen og Antonin Klima

- 7 und.asser

- Anna Lovise Rekdal, Ole K Pedersen, Daniel Solberg Strømmen, Torje Digernes, Fredrik Pe Ingebrigtsen, Petter Taule , Sondre Baugstø

- 42 stud.asser

Vi oppfordrer til bruk av **diskusjonsforumet piazza** for alle faglige og praktiske spørsmål

Det du lurere på er veldig ofte også relevant for andre!

Du kan poste anonymt hvis du vil, men bruk gjerne navn også

Praktiske spørsmål om øvingene kan sendes til:

Epostlista: tdt4102-undass@idi.ntnu.no

Spørsmål eller kommentarer av mer "privat" karakter:

Faglærer: Lasse.Natvig@idi.ntnu.no

eller bruk e-postlista: tdt4102-fagans@idi.ntnu.no

(Denne når bare faglærer Lasse og vit.ass Sivert)

En del av henvendelse som faglærer får vil bli behandlet av vit.ass. eller i noen tilfeller und.ass.

Vennligst IKKE bruk den nye chat-funksjonen i Its'learning, bruk da heller vanlig e-post

Timeplan

- Forelesninger:
 - Mandager i F1, 14:15-16:00 (alle)
 - Torsdager i **R1**, 16:15-18:00 (noen)
- Øvingsforelesninger:
 - Onsdager i F1, 16:15-18:00
- Øvingstimer på sal:
 - Avhenger av hvilken gruppe du er med i
 - (0815-2000 man-tir-ons-tor-fredag)
- «Hjelp i R1»
 - Alle eller de fleste torsdager hvor det ikke er forelesning, i F1 16:15-18:00.
- Endringer;
 - følg med på «**Hva skjer, når og hvor?**»
 - F.eks. **nødvendig endring denne første uken**

Øvingsopplegg (i)

- **Alle** blir manuelt fordelt på grupper
 - Basert på studieprogram
 - Får tildelt stud.ass som finner tidspunkter
 - Følg med på It's Learning
 - Stud.ass ansvarlig for å finne egnede tidspunkter på sal – i denne uken!
- **Saler**
 - 4. etasje i «P15 bygget» (H3, Høgskoleringen 3), salene 424 Vembi og 414 Gorg
- **Undervisningsassistente**
 - **Finner du på en av salene i "kontortiden"**
 - Tar i mot alle typer forespørsler
 - Bruk dette tilbudet aktivt!

Øvingsopplegg (ii)

- 12 øvinger i programmering (i tillegg til øving 0) med veiledning og **godkjenning på sal**
- Må ha godkjent 8 av disse (minst 70% av hver)
- I tillegg:
 - Denne uka er det **"komme i gang"-øving nr. 0**
 - teller ikke, men er fundamental for alle øvinger
 - **Øving 1 har frist **fredag 20/1****
 - Tema: Overgang fra matlab/python
- Deretter 1 ny øving i uka
 - Øving legges ut den uka temaet foreleses, men noen ganger litt i forkant
 - Innlevering ca. 10 dager senere

Verktøy til programmering

- Ingen formelle krav til bruk av verktøy
 - men vi legger opp til bruk av IDE (Integrated Development Environment) som er programvare som kombinerer editor, kompilator, hjelpeverktøy med mer.
- Windows
 - Visual Studio Community 2015 (**for Windows Desktop**)
- Mac
 - XCode (Apple sitt utviklingsverktøy)
- Linux
 - Fritt valg...

Se Øving 0 for mer info om verktøy, Veiledning i installasjon og bruk i øvingsforelesingen **denne uke!**

Referansegruppe

Vi trenger frivillige til en referansegruppe,
1 (eller fler) fra hvert studieprogram

Liste blir sendt rundt til den er fylt ut!

HVEM HAR FAGET?

Kilde: <https://www.ntnu.no/studier/emner/TDT4102#tab=timeplan>

Kode	Program	Referanse- gruppemedlem, Navn + e-post
<u>BFY</u>	Bachelor 3-årig Fysikk	
<u>MTPETR</u>	Siv.ing./master 5 år Petroleumsfag	
<u>MLREAL</u>	Masterprogram realfag / lektor, 5-årig Lektorutdanning i realfag	
<u>MTELSYS</u>	Siv.ing./master 5 år Elektronisk sys.des og innov.	
<u>MTENERG</u>	Siv.ing./master 5 år Energi og Miljø	
<u>MTFYMA</u>	Siv.ing./master 5 år Fysikk og matematikk	
<u>MTIØT</u>	Siv.ing./master 5-år Industriell økonomi og tek.led.	
<u>MTKJ</u>	Siv.ing./master 5 år, Industriell kjemi og biotekn.	
<u>MTNANO</u>	Siv.ing./master 5 år, Nanoteknologi	
<u>MTTK</u>	Siv.ing./master 5 år, Kybernetikk og robotikk	

Spørsmål?

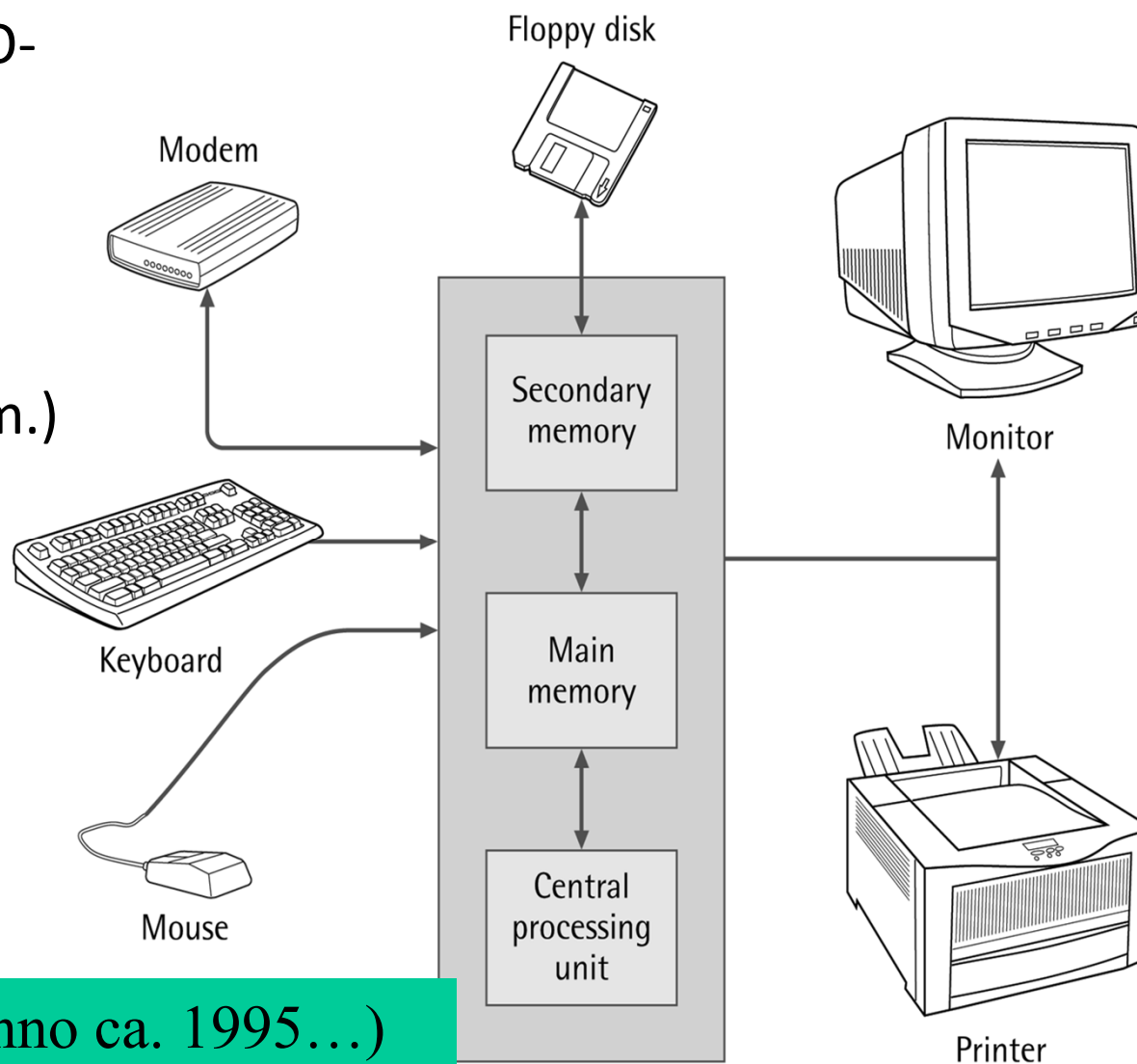
Introduksjon til programmering

- Maskinvare og programvare
- Maskinkode, assemblerkode, høynivå programmeringsspråk
- C++
- Fra kildekode til kjørbare fil

Maskinvare og programvare

- Datamaskiner består av:
 prosessor, minne, lagringsenheter, IO-enheter (tastatur, mus, skjerm), kommunikasjonsenheter (nettverkskort m.m.)

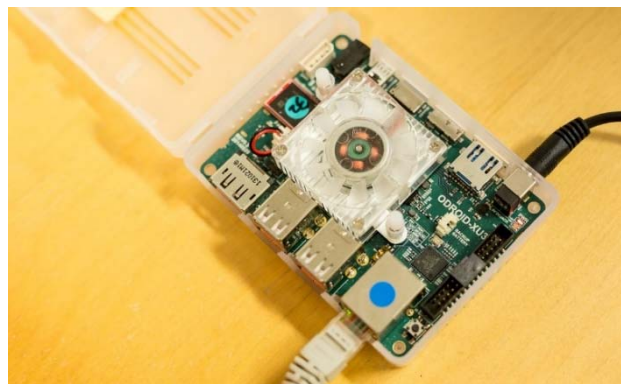
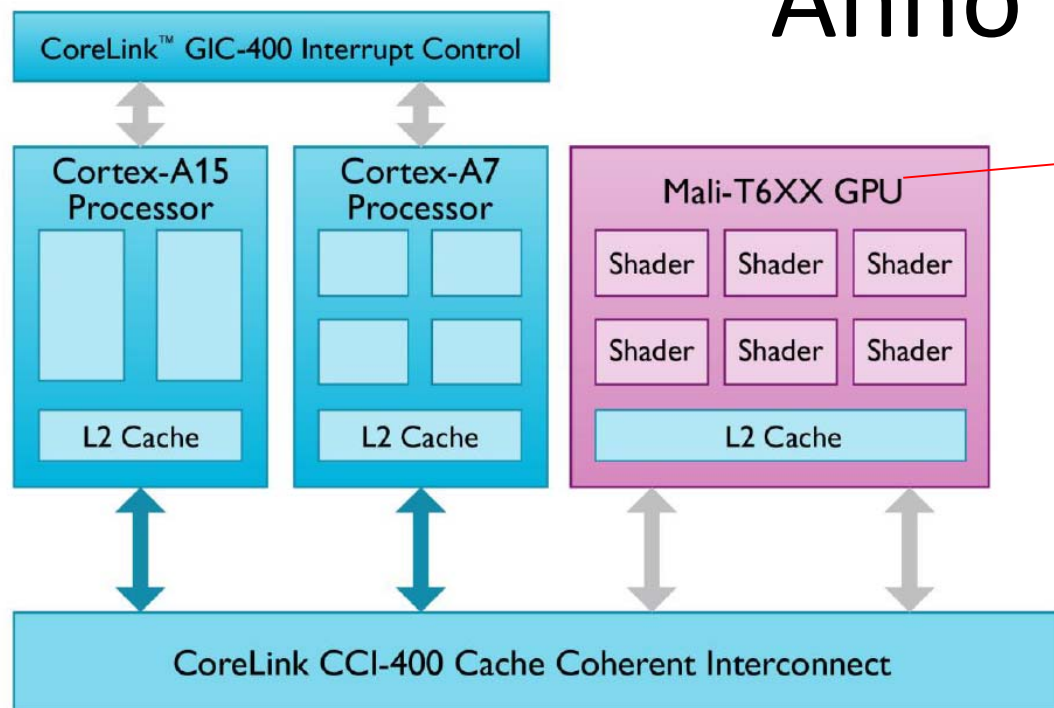
- Men er også i veldig stor grad programvare



(Anno ca. 1995...)

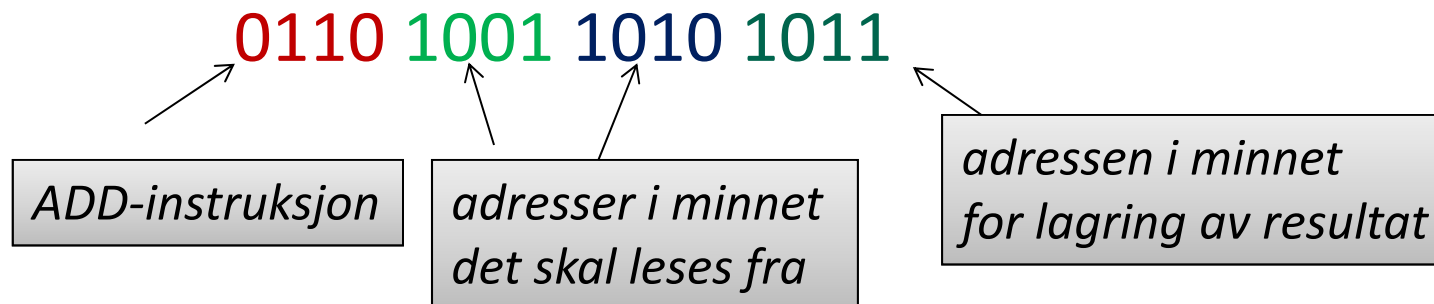
Anno 2016, f.eks.

Utviklet av
NTNU-studenter



Maskinkode

- Maskinkode er instruksjoner lagret i binært format som kan utføres direkte av en prosessor (CPU) uten oversetting
 - Ulike prosessorer har forskjellige sett av tilgjengelige instruksjoner
 - [http://no.wikipedia.org/wiki/Instruksjon_\(datamaskin\)](http://no.wikipedia.org/wiki/Instruksjon_(datamaskin))
- F.eks. kan vi ha en instruksjon som
 - legger sammen verdiene lagret på to forskjellige minnelokasjoner og lagrer resultatet i en tredje minnelokasjon
- På binær form kan dette være:



Assembler kode

- Assemblerkode brukes for å gjøre det enklere å skrive og lese maskinkode

ADD X Y Z

- Assembler-språk er kun en mnemonisk utgave av maskinspråket
 - kan oversettes direkte til maskinkode
 - er derfor også plattformavhengig
- Maskinkodeinstruksjonene er i tillegg temmelig primitive operasjoner
 - tidkrevende å skrive komplekse programmer, vanskelig å formulere ønsket funksjonalitet direkte vha. maskinkode

Høynivå programmeringsspråk

- For å kunne:
 - uttrykke ønsket funksjonalitet med et språk som er «lett» å lese/skrive
 - effektivt programmere og håndtere store og komplekse programmer
 - slippe å forholde seg til maskinvaren på laveste nivå, bruke mer komplekse/abstrakte måter å beskrive en løsning på
 - kunne lage kode som ikke er bundet til spesifikke sett av instruksjoner (plattformavhengig) ➔ dvs. oppnå portabilitet (flyttbarhet)

Programmeringsspråk

Les selv ☺



- Over hundre forskjellige høynivå språk
- Laget for spesifikke formål og mange er fortsatt i daglig bruk
- Felles for alle er at kode skrevet i disse språkene må oversettes til maskinkode
- Kjente historiske navn for de som er interessert:
 - Charles Babbage (1791-1871)
 - Augusta Ada King, Countess of Lovelace (1815-1852)
 - Og ikke minst Alan Turing

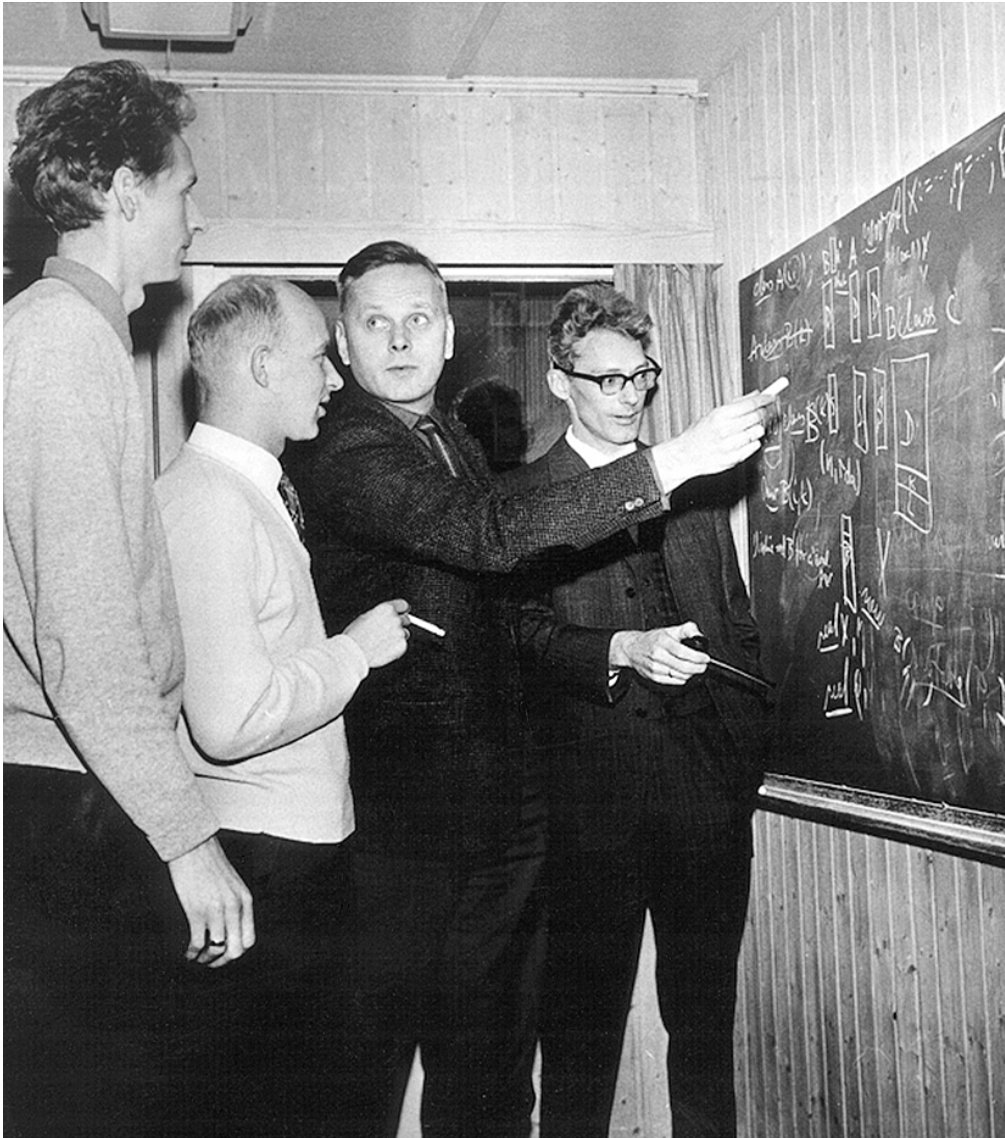
COBOL
FORTRAN
BASIC
Pascal
Ada
Visual Basic
Delphi
C
C++
Java
C#
...

C++



- **C++** er en videreutvikling av programmeringsspråket **C**
- **C++** ble utviklet av Bjarne Stroustrup (AT&T Bell Labs) på 1980-tallet
 - For å løse en del begrensninger ved **C**
 - Støtter objektorientert programmering
 - **C** er fortsatt en delmengde av **C++**
- **C** ble utviklet av Dennis Ritchie (AT&T Bell Labs) på 1970-tallet
 - Brukt for å skrive og vedlikeholde UNIX
 - Finnes fortsatt svært mye kommersiell programvare skrevet i **C**

C++ støtter objektorientert programmering

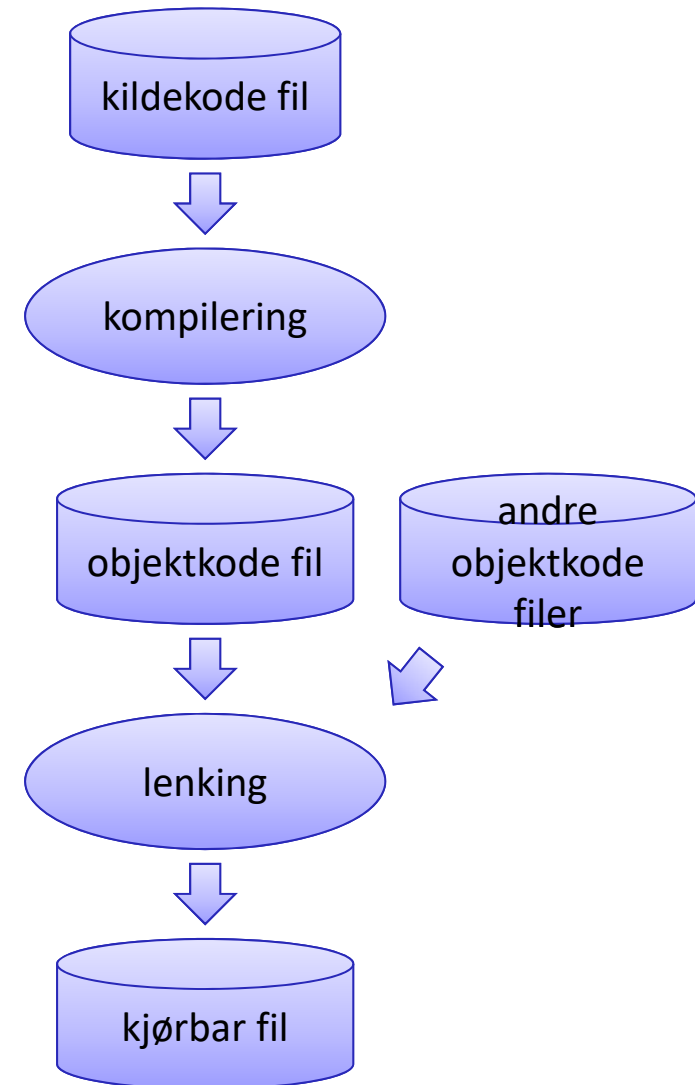


- Hvem oppfant objektorientert programmering?



Fra kildekode til kjørbart fil

- Koden skrevet i et programmeringsspråk kalles **kildekode** (Eng. source code)
- Å oversette fra kildekode til maskinkode kalles **kompilering**
 - gjøres ved hjelp av et program som kalles **kompilator**
 - resultatet kalles ofte **objektkode**
- Maskinkoden som vi selv lager må kombineres med/lenkes til annen eksisterende kode (f.eks. bibliotek)
 - dette kalles for **lenking** (linking)
- Sluttresultatet er en **kjørbart fil**



Areal-VS - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

Debug x86 Local Windows Debugger Auto

Search Solution Expl

Solution 'Areal-VS' (1)

- Areal-VS
 - References
 - External Depen
 - Header Files
 - Resource Files
 - Source Files
 - Areal.cpp

```
#include <iostream>
using namespace std;
int main() {
    int h = 5; // Høyde
    int b = 8; // Bredde
    int Areal = h * b;
    cout << "Areal av rektangel = " << Areal << endl;
}
```

161 %

Output

Show output from: Build

```
1>----- Build started: Project: Areal-VS, Configuration: Debug Win32 -----
1> Areal.cpp
1> Areal-VS.vcxproj -> c:\users\lasse\documents\visual studio 2015\Projects\oving_test\Areal-VS\Debug\Areal-VS.exe
1> Areal-VS.vcxproj -> c:\users\lasse\documents\visual studio 2015\Projects\oving_test\Areal-VS\Debug\Areal-VS.pdb (Partial PDB)
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

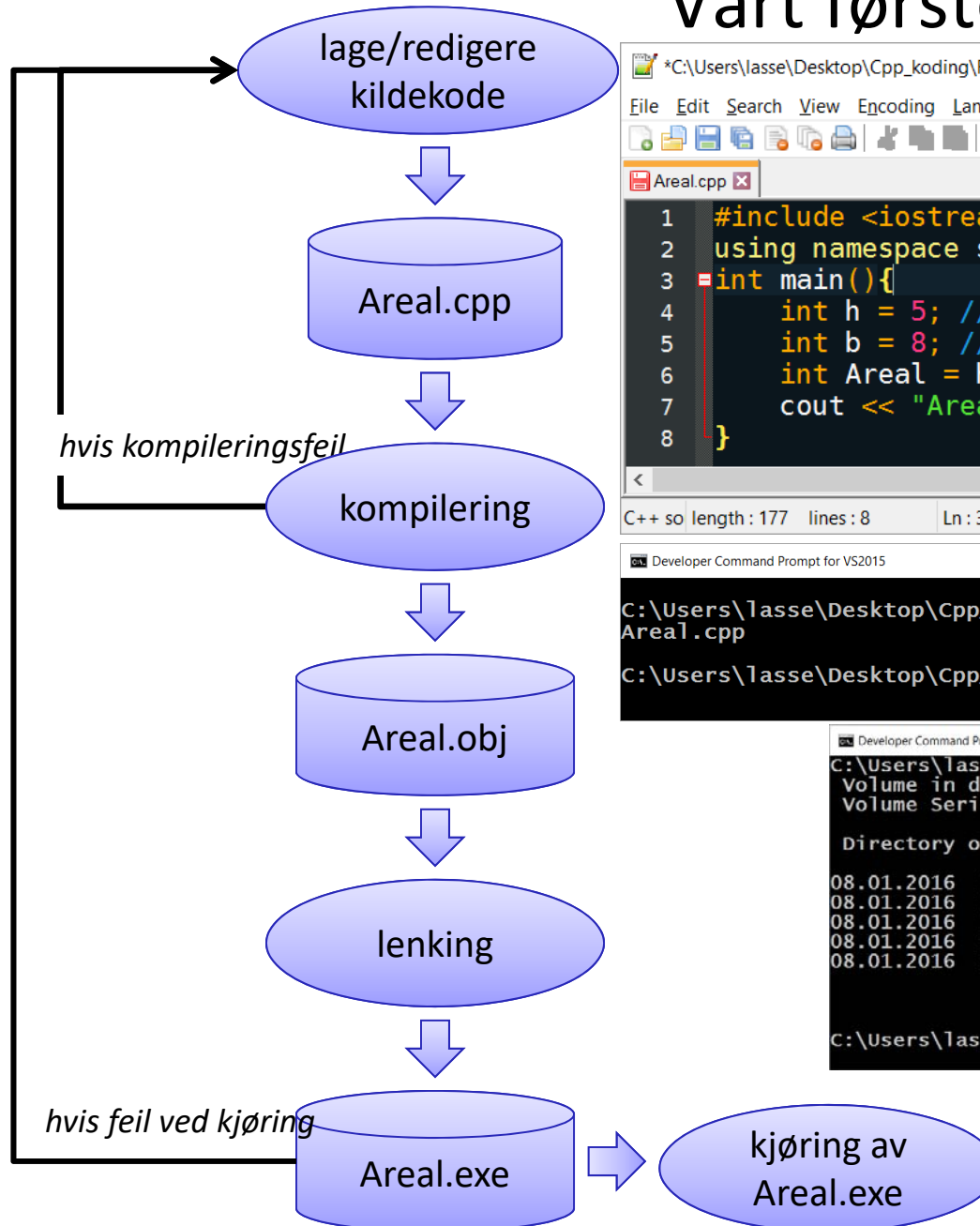
Output Error List

Build succeeded

Ln 4 Col 24 Ch 21 INS

34

Vårt første C++ program



```

C:\Users\lasse\Desktop\Cpp_koding\Fore1-1\Areal.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Areal.cpp
1 #include <iostream>
2 using namespace std;
3 int main(){
4     int h = 5; // Høyde
5     int b = 8; // Bredde
6     int Areal = h * b;
7     cout << "Areal av rektangel = " << Areal << endl;
8 }
C++ so length: 177 lines: 8 Ln: 3 Col: 12 Sel: 0|0 Dos\Windows UTF-8 INS
  
```

```

Developer Command Prompt for VS2015
C:\Users\lasse\Desktop\Cpp_koding\Fore1-1>c /nologo /EHsc Areal.cpp
Areal.cpp
C:\Users\lasse\Desktop\Cpp_koding\Fore1-1>_
  
```

```

Developer Command Prompt for VS2015
C:\Users\lasse\Desktop\Cpp_koding\Fore1-1>dir
Volume in drive C has no label.
Volume Serial Number is DAD2-CA50

Directory of C:\Users\lasse\Desktop\Cpp_koding\Fore1-1

08.01.2016 19:34 <DIR> .
08.01.2016 19:34 <DIR> ..
08.01.2016 19:19          93 Areal.cpp
08.01.2016 19:39       171 520 Areal.exe
08.01.2016 19:39        92 144 Areal.obj
                3 File(s)      263 757 bytes
                2 Dir(s)    286 116 417 536 bytes free

C:\Users\lasse\Desktop\Cpp_koding\Fore1-1>_
  
```

```

Developer Command Prompt for VS2015
C:\Users\lasse\Desktop\Cpp_koding\Fore1-1>Areal
Areal av rektangel = 40
C:\Users\lasse\Desktop\Cpp_koding\Fore1-1>_
  
```

Programmeringsverktøy

- Effektiv programmering krever mer enn det en vanlig teksteditor + kompilator kan gi
- Trenger hjelp til å
 - skrive og lese kode effektivt
 - følge språkets regler og unngå de verste feilene før kompilering
 - kompilere, kjøre, undersøke programmet vårt
 - teste og finne feil, debugging
- Vi bruker derfor utviklingsverktøy ofte kalt "IDE"
 - "integrated development environment": integrert utviklingsverktøy
 - program som støtter prosessen å konstruere programvare
- Microsoft Visual Studio er et eksempel for Windows, Xcode er tilsvarende for Mac

Feilfinning (debugging, avlusing)

Photo # NH 96566-KN (Color) First Computer "Bug", 1947

92

9/9


0800 Andam started
 1000 " stopped - andam ✓

1300 (032) MP-MC ~~1.582647000~~
 (033) PRO 2 2.130476415
 convd 2.130676415

Relays 6-2 in 033 failed special speed test
 in relay 11.00 test.

Relays changed

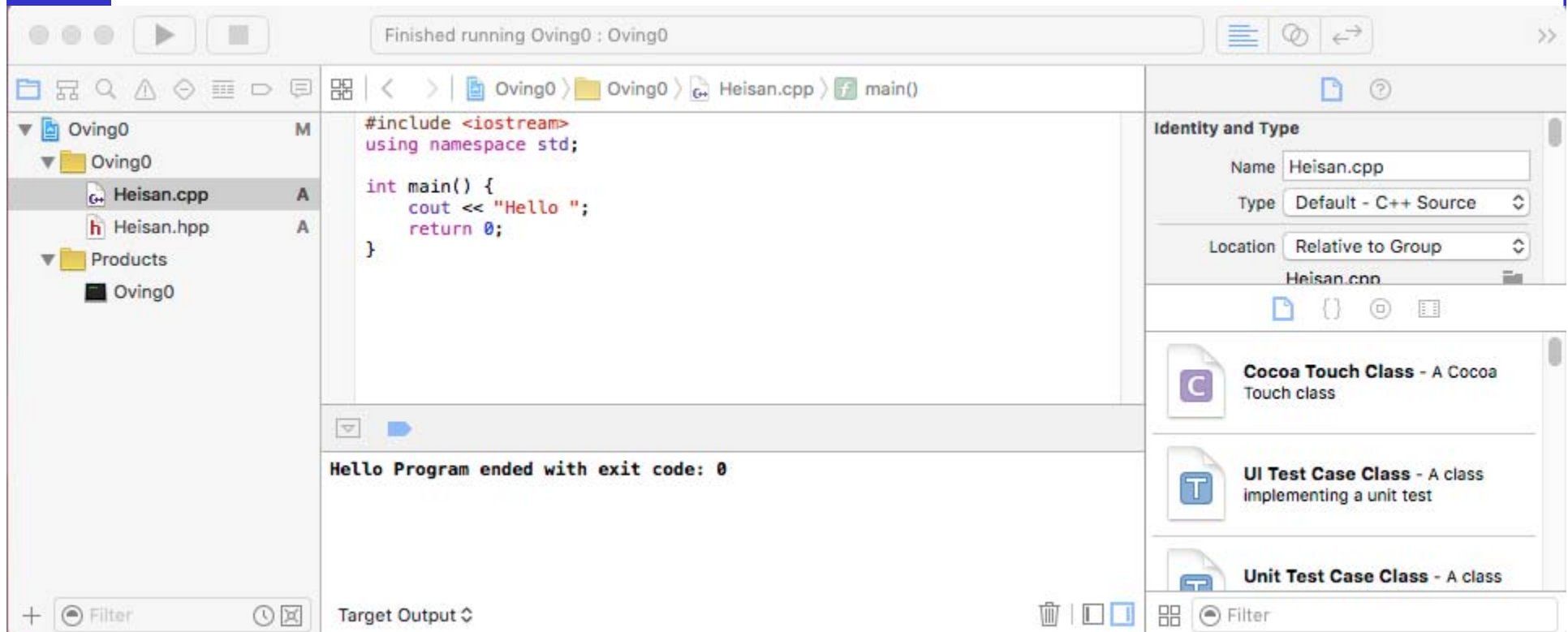
1100 Started Cosine Tape (Sine check)
 1525 Started Multy Adder Test.

1545  Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.

~~1630~~ 1630 Andam started.
 1700 closed down.

Relay 2145
 Relay 2370



Installering på egne maskiner

- Visual Studio 2015 finnes i en gratis utgave (Community)
- Apple Xcode er også gratis og tilgjengelig i App Store
- Se øving 0 for nærmere detaljer
 - Dette er STORE verktøy, prøv å last ned FØR øvingstime o.l., spør gjerne noen du kjenner

Andre versjoner?

- De fleste tidligere versjoner av Visual Studio og Xcode vil fungere greit
- GUI kan være litt forskjellig, men stort sett samme funksjonalitet
- Siste versjoner vil ha best støtte for C++ -14 (siste versjon av språket), men det har liten betydning i dette faget

Øvingene

- Øvingene er lagt ut
 - Øving 0 er en guide for å komme i gang
 - Øving 1 skal være temmelig enkel dersom du husker en del av det du lærte i IT-intro, frist er **fredag 20/1**.
- Grupper er satt opp
 - Nye studenter tilordnes etter hvert
 - Stud.ass-ene har ansvar for å finne aktuelle saltider
- Hjelp til å komme i gang med verktøy
 - Les og gjør øving 0 grundig
 - Gå på sal og få hjelp av und.ass/stud.ass
 - Bruk diskusjonsforumet (Piazza)
 - Spør fornuftig
 - Hjelp andre når du kan

Godkjent øvingsopplegg i TDT4102 fra tidligere år?

- Gir deg automatisk rett til å gå opp til ny eksamen
- Har du vært oppe til eksamen tidligere er godkjent øvingsopplegg registrert

Spørsmål?

TDT4102

Prosedyre- og objektorientert programmering

The image features a large, stylized 'C++' logo in blue. Behind the logo, a snippet of C++ code is visible, tilted at an angle. The code includes a preprocessor directive, a namespace declaration, a main function signature, and a cout statement.

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

Grunnleggende C++ syntaks

(«IT-GK i C++»)

Innhold

- Grunnleggende C++
 - basis for alle programmer
 - variabler, datatyper, uttrykk og tilordning, m.m.
 - inndata og utdata (lese inn og skrive ut verdier)
 - litt om løkker og forgreininger
 - Funksjoner, introduksjon

Kap. 1- (2-3)



Rektangel



```

C:\Users\lasse\Desktop\TDT4102 foreles\Uke 2\Kode\Project1\Areal.cpp - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Areal.cpp
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int h = 5; // Høyde
6     int b = 8; // Bredde
7     int Areal = h * b;
8     cout << "Areal av rektangel = " << Areal << endl;
9 }
C++ source file length : 179 lines : 9 Ln : 4 Col : 13 Sel : 0 | 0 Windows (CR LF) ANSI IN

```

Linje 2-9 utgjør en funksjon som heter **main** som gjør en enkel beregning og skriver ut resultatet



#include <iostream>

- Vi bruker ofte funksjoner som er “ferdiglaget”
 - Kompilert og tilgjengelig i bibliotek
 - Ikke en del av selve språket, men del av «programmerings-miljøet» som kommer med språket
- Vi må deklarere disse med **#include <bibliotek>**
- Vi trenger **<iostream>** for å kunne skrive ut til skjerm

`int main(){...}`

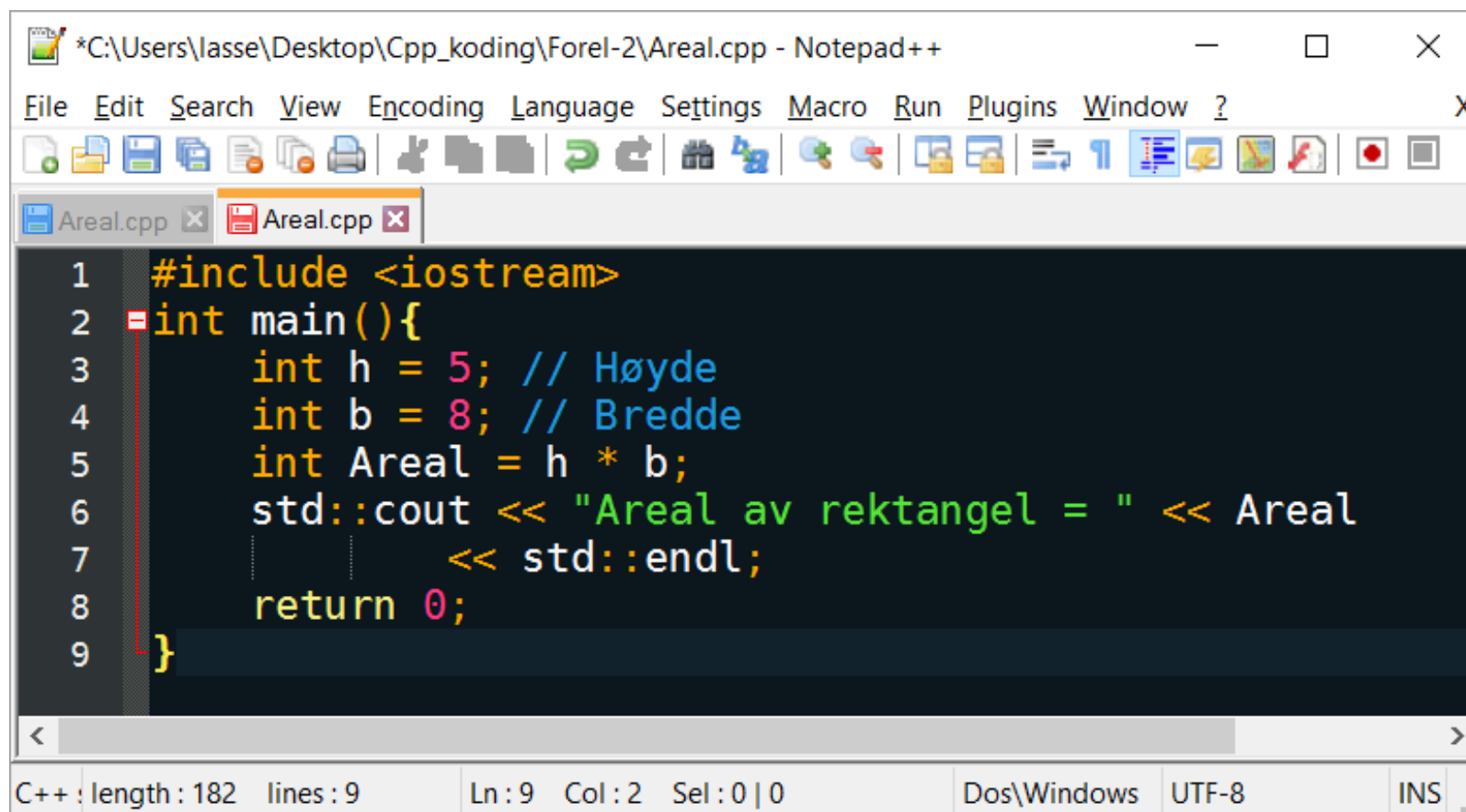


- Er en litt spesiell funksjon som **MÅ** være med i alle kjørbare programmer
- "Startpunktet" for programmet ditt
- **int** foran funksjonsnavnet deklarerer returtypen til funksjonen
 - Spesifiserer at funksjonen returnerer verdi av heltallstypen
- Hvis du ikke eksplisitt skriver return i main vil kompilatoren legge til return 0; før avsluttende krøllparantes.

Blokker, setninger, uttrykk

- På laveste nivå av instruksjoner i programmet har vi **uttrykk** `h * b`
- En **setning** er en helhetlig instruksjon som vi skriver i programmeringsspråket
 - Ett eller flere uttrykk `int Areal = h * b;`
 - Avsluttes alltid med semikolon
 - Setningene utføres i den rekkefølgen de er skrevet
- Setninger kan grupperes i **blokker** `{ ... }`
 - Starter med og avsluttes med krøllparantes

Variabler og operatorer



The screenshot shows a Notepad++ window titled '*C:\Users\lasse\Desktop\Cpp_koding\Forel-2\Areal.cpp - Notepad++'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations and editing. The code editor shows the following C++ code:

```
1  #include <iostream>
2  int main(){
3      int h = 5; // Høyde
4      int b = 8; // Bredde
5      int Areal = h * b;
6      std::cout << "Areal av rektangel = " << Areal
7          << std::endl;
8      return 0;
9  }
```

The status bar at the bottom indicates 'C++ : length : 182 lines : 9', 'Ln : 9 Col : 2 Sel : 0 | 0', 'Dos\Windows', 'UTF-8', and 'INS'.

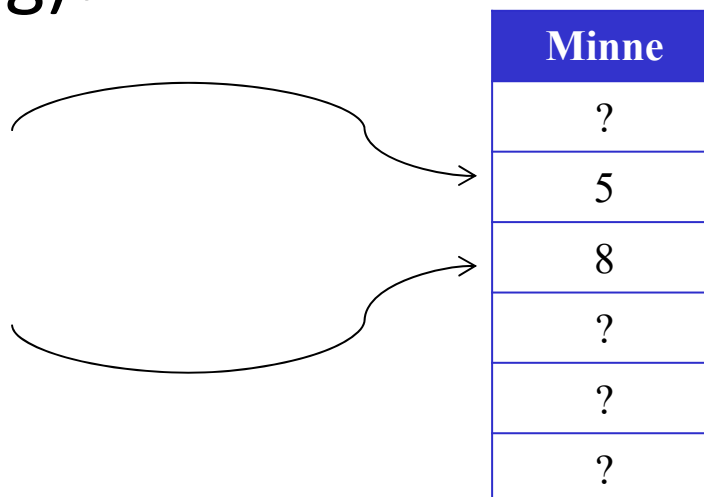
- ▶ Linje 3 **deklarerer variabelen** h med **datatype** int som betyr heltall, og setter den samtidig til verdien 5 som er en **konstant** (kalles også literal)
- ▶ * i linje 5 er en **operator** (multiplikasjon), og ; avslutter en **program-setning**

C++ variabler

```
#include <iostream>
using namespace std;
int main()
{
    cout << "hello World!\n";
    return 0;
}
```

- Er plassholdere for data i programmet
- En **navngitt minnelokasjon**
- Alle variabler i et C++ program må deklarereres før bruk (eller når de brukes første gang)!

```
int h = 5;
int b = 8;
```



Deklarering av variabler



- Deklarering av **variabler** består i å
 - Bestemme hvilken data-type som variabelen skal kunne lagre
 - Gi variabelen en **identifikator** (et navn)
- C++ har en del **basis-datatyper**
- Datatypen forteller kompilatoren
 - **hvor stor plass** verdien krever i minne
 - **hvilke operasjoner** som skal kunne utføres på variablene

C++ datatyper

Typenavn	Minnebruk	Verdier	Presisjon
short	2 bytes	-32768 t.o.m. 32767	na
int	4 bytes	-2.147.483.648 t.o.m. 2.147.483.647	na
long	4 bytes	-2.147.483.648 t.o.m. 2.147.483.647	na
float	4 bytes	ca. 10^{-38} til 10^{38}	7 siffer
double	8 bytes	ca. 10^{-308} til 10^{308}	15 siffer
long double	10 bytes	Ca. 10^{-4932} til 10^{4932}	19 siffer
char	1 byte	256 tegn (ascii)	na
bool	1 byte	true, false	na

char-typen kan også brukes til å representere et tall eller en byte
For alle heltallstypene finnes det også en **unsigned** variant



Initialisering av variabler



```
#include <iostream>
using namespace std;
int main()
{
    cout << "hello World!\n";
    return 0;
}
```

- Variabler som deklarerer uten å samtidig gis en spesifikk verdi har udefinert innhold!

```
int bredde;
```

*Bredde kan være et
hvilket som helst tall*

- Initialisering
 - Du kan deklarere og bruke variabler i en og samme linje!
 - Du bør alltid initialisere med en fornuftig “startverdi”
 - Ikke-initialiserte variable er farlig!

```
int y = 0;           // initialiserer med 0
int z(0);            // alternativ syntax -- anbefales IKKE
int x = y + z;       // initialiserer med y+z
```

Literaler



- Spesifikke verdier som vi skriver inn i koden vår kalles for **literaler (betyr bokstavelig verdi)**
- Datatypen disse får avhenger av syntaks
- Heltall skrives rett frem: **5**
- Flyttall (reelle tall) skrives med desimalnotasjon: **2.0**
- Teksttegn skrives med enkle anførselstegn: **'c'**
- Tekststrenger skrives med doble anførselstegn
"Dette er en tekststreng"

C++ identifikatorer



- Navn du gir til variabler (eller funksjoner eller annet) kalles for **identifikatorer**
- C++ har en del regler for hva som er **lovlige** identifikatorer
 - Må starte med bokstav eller underscore
 - De øvrige tegn kan være bokstaver, tall, underscore
 - Ikke lov å bruke nøkkelord/reserverte ord
 - NB! store og små bokstaver er forskjellige tegn
- I tillegg til reglene for lovlige identifikatorer har vi **konvensjoner** for hvordan du bør konstruere identifikatorer («best practice»)
 - Variabler bør begynne med liten bokstav
 - Hvis du kombinerer ord kan ordene synliggjøres ved å bruke stor forbokstav (men ikke for første ord)
 - Bruk meningsfulle identifikatorer
- **Kode med god lesbarhet er viktig!**

C++ nøkkelord



- Typenavnene som vi bruker når vi deklarerer variabler er eksempler på **C++ nøkkelord**
- Nøkkelord er **identifikatorer** som har en spesiell **predefinert betydning**.
- De kalles også **for reserverte ord**
- NB! kan derfor ikke brukes som navn på variabler eller andre enheter vi selv gir navn til
- En moderne IDE oppdager nøkkelord når du skriver dem og viser dem **i egen farge**

Konstanter



- Vi har ofte bruk for å benytte spesifikke verdier i programmene våre eks:
 - forhåndsdefinert maksimumsverdi eller minimumsverdi
 - PI (π)
- Vi kan definere slike variabler ved å lage konstanter:

```
const double PI = 3.14159;
```

- Kan ikke endre verdi ved kjøring!
- Vanlig konvensjon å bruke STORE_BOKSTAVER
- Kalles **deklarert konstant** eller navngitt konstant

Uttrykk og operatorer



$3 + 4$
$2 * 4$

- Programmeringsspråket lar deg skrive **uttrykk** basert på **operatorer** og **operander**
- De vanlige aritmetiske operatorene **+**, **-**, *****, **/**, **%**
- I programmeringsspråk brukes termen **operator** om mer enn de aritmetiske og boolske
 - Eks: tilordningsoperatoren **=**
- Ett uttrykk gir eksakt en verdi
- Alle uttrykk i C++ har returverdi

Aritmetiske operatører

+ - * / %



- Variabler, konstanter og literaler kan være operander
- **Datatypen** til en **operand** definerer **hvilke** operatører som kan benyttes
- Når aritmetiske operatører benyttes med talltyper vil resultatverdien avhenge av datatypene som benyttes!
 double = double * double //flyttallsmultiplikasjon
 int = int * int //heltallsmultiplikasjon
- I noen tilfeller kan du bruke operander av forskjellig type
- Hvilken datatype som produseres er avhengig av operanden som har høyest "prioritet"
 double = int * double //flyttallsmultiplikasjon