TDT4102 – C++ OOP
Lecture 1: Information Compilation

Bart Iver van Blokland
(Rune Sætre)

Vi ønsker dere hjertelig velkommen til TDT4102 Prosedyre- og Objektorientert Programmering!

Vi gjør foilene tilgjengelig for dere på BlackBoard, med notater slik at dere kan følge med på norsk på det som forklares.

Before we begin..

- My apologies for any language errors
- We have made fresh slides for you!
- Slides are in English, but have notes in Norwegian

Vi lager nye foiler i år. Send oss gjerne kommentarer på mail om det vi kan forbedre til neste år!

Slides (lysark) blir gitt på Engelsk, med forelesning og utdypende notater på Norsk. Programmeringsspråk er vanligvis på Engelsk, så dette gir bedre konsistens med alt annet som vises. I tillegg får dere på denne måten et følelse for relasjonen mellom Norsk og Engelsk dataterminologi.

Dessuten er Norsk ikke mitt morsmål, og jeg forventer at det kommer til å oppstå språklige feil.

Today..

1. Answer questions about the course:
   - Who are these people?
   - Where do we find information?
   - When are the lectures and deadlines?
   - What is the course about?
   - Why do we care?
   - How do we do the coursework?

2. First steps in C++

May? Yes of course you may! <3

Før vi begynner med noen smakbiter av programmering med C++ i andre time av dagens forelesning er det en del spørsmål om faget vi bør svare på.

Av disse er det aller viktigst å forstå hvorfor C++ brukes i dag, og hva dere kan gjøre med språket.

Today..

1. Answer questions about the course:
   - **Who are these people?**
   - Where do we find information?
   - When are the lectures and deadlines?
   - What is the course about?
   - Why do we care?
   - How do we do the coursework?

2. First steps in C++

May? Yes of course you may! <3

Først begynner vi med å introdusere fagstaben.

Bart Iver van Blokland

- Assistant Professor at the Department of Computer Science (IDI)
- Interests:
  - Computer Graphics
  - GPU Programming
  - 3D Object Recognition
  - High Performance Computing
- Free time:
  - Kayaking
  - Hiking
  - Video and boardgames

https://github.com/bartvbl

Hey look, it's me! :D

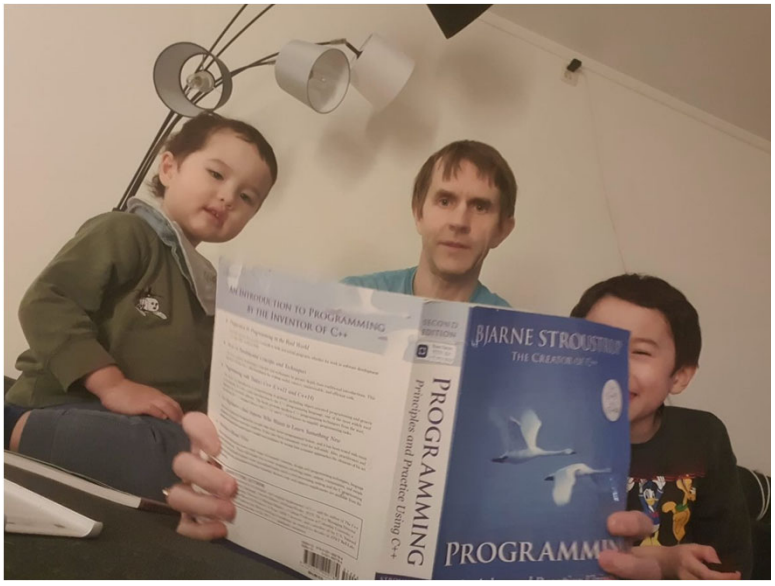
Jeg foreleser parallell 1

Rune Sætre

- Associate Professor at the Department of Computer Science (IDI)

- Interests:
  - Artificial Intelligence
  - Natural Language Processing
  - Software Engineering
  - Health IT

- Free time:
  - Climbing
  - Skydiving
  - Biking
  - Swimming



Rune foreleser parallell 2

Erling Syversveen Lie

- Vit.Ass. at the Department of Computer Science (IDI)
- Master Student in Cybernetics
- Black Belt in Blackboard Fu
- Elite operative in the Student Emergency Special Response Unit
- Guru of the C++ language

Asta Skirbekk

- Vit.Ass. at the Department of Computer Science (IDI)
- Master Student in Electrical Engineering
- Black Belt in Blackboard Fu
- Elite operative in the Student Emergency Special Response Unit
- Guru of the C++ language

Asta og Erling er ansvarlig for alt som skjer utenfor vanlige forelesninger og eksamen.

Today..

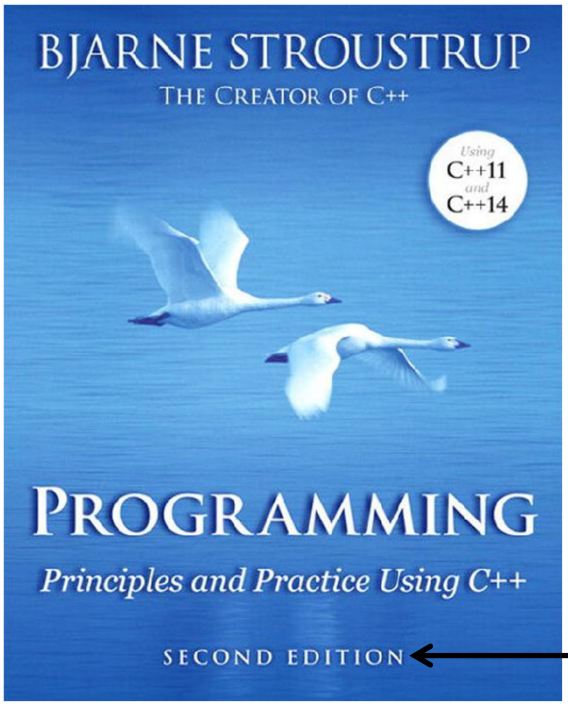
1. Answer questions about the course:
   - Who are these people?
   - **Where do we find information?**
   - When are the lectures and deadlines?
   - What is the course about?
   - Why do we care?
   - How do we do the coursework?

2. First steps in C++

May? Yes of course you may! <3

Det er forskjellige ressurser som utnyttes i faget.

Recommended reading



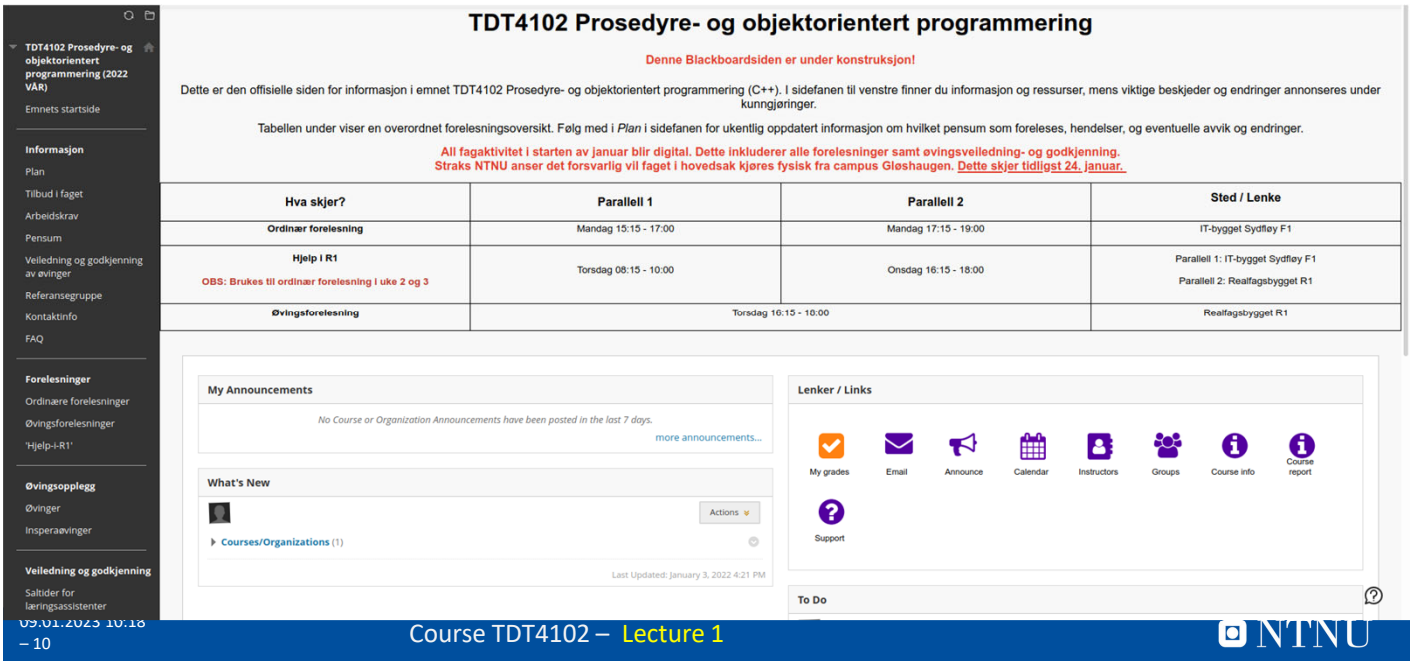
Note: second edition!

Vi kommer til å bruke boka som er skrevet av Bjarne, som er opprinnelige forfatteren av C++ språket.

Vi følger ikke boka veldig nøye i forelesningene, og spesielt i begynnelsen kan dere oppleve at vi hopper litt fram og tilbake.

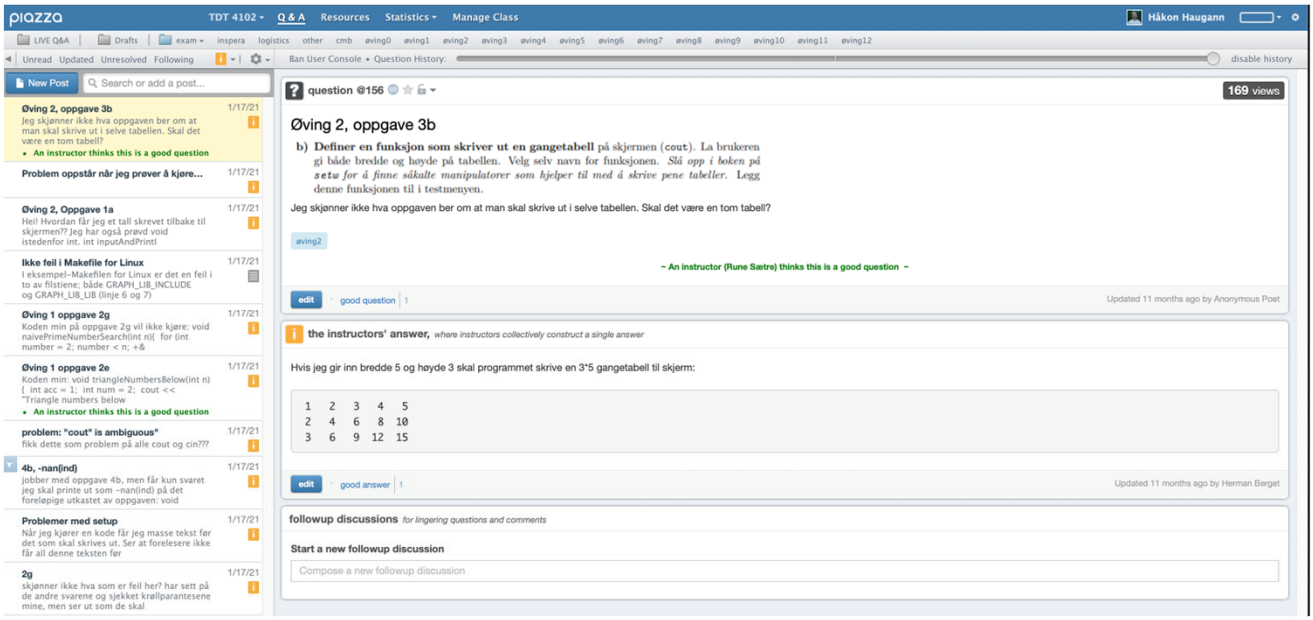
I tillegg er det flere avsnitt i boka som ikke behandles i de hele tatt.

Blackboard: Course Information and Assignments



Oppdatert informasjon om emnet, kunngjøringer, og innlevering av øvinger skjer på blackboard.

Piazza: Forum for getting help



Vi har i tillegg skaffet lisens på Piazza. Her kan dere spille spørsmål om øvingene, pensum, og alt annet relatert til faget.

Rune kan svare «live» på spørsmål med tag Forelesning-0x mens Bart underviser (og omvendt?)

Today..

1. Answer questions about the course:
   - Who are these people?
   - Where do we find information?
   - **When are the lectures and deadlines?**
   - What is the course about?
   - Why do we care?
   - How do we do the coursework?

2. First steps in C++

May? Yes of course you may! <3

Når er fristene og forelesningene?

Lecture Schedule (tentative)

Parallel 1 (Bart)

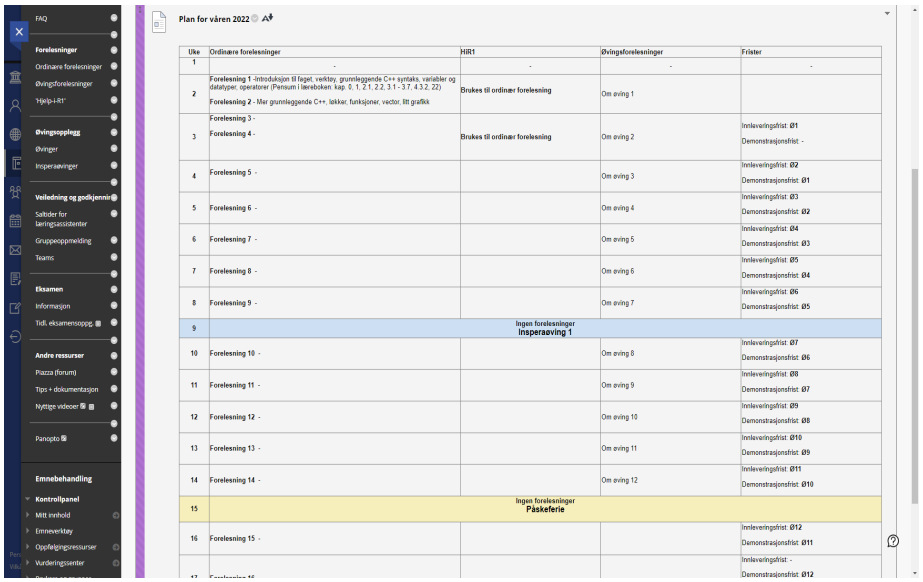
Monday 10:15 – 12:00	09.01	16.01	23.01	30.01	6.02	13.02	20.02	27.02	6.03	13.03	20.03	27.03	17.04	24.04
Tuesday 14:15 – 16:00	10.01	17.01	No lectures (instead “Hjelp i F1”)											

Parallel 2 (Rune)

Wednesday 08:15 – 10:00	11.01	18.01	25.01	1.02	8.02	15.02	22.02	1.03	8.03	15.03	22.03	29.03	19.04	26.04
Thursday 14:15 – 16:00	12.01	19.01	No lectures (instead “Hjelp i R1”)											

Vi kommer til å ha et ordinær forelesning per uke, men unntak av de første to ukene hvor det blir to. Etterhvert blir perioden på Tirsdag og Torsdag brukt til «Hjelp i R1/F1» forelesningene, hvor det behandles spørsmål relatert til øvingsopplegget.

Assignment Schedule (tentative)



Plan for våren 2022

Uke	Ordinære forelesninger	HIR1	Øvingsforelesninger	Frister
1	-	-	-	-
2	Forelesning 1 -Introduksjon til faget, verktøy, grunnleggende C++ syntaks, variabler og datatyper, operatorer (Pensum i læreboken: kap. 0, 1, 2.1, 2.2, 3.1 - 3.7, 4.3.2, 22) Forelesning 2 - Mer grunnleggende C++, løkker, funksjoner, vector, litt grafikk	Brukes til ordinær forelesning	Øv øving 1	
3	Forelesning 3 - Forelesning 4 -	Brukes til ordinær forelesning	Øv øving 2	Innleveringsfrist: Ø1 Demonstrasjonsfrist: -
4	Forelesning 5 -		Øv øving 3	Innleveringsfrist: Ø2 Demonstrasjonsfrist: Ø1
5	Forelesning 6 -		Øv øving 4	Innleveringsfrist: Ø3 Demonstrasjonsfrist: Ø2
6	Forelesning 7 -		Øv øving 5	Innleveringsfrist: Ø4 Demonstrasjonsfrist: Ø3
7	Forelesning 8 -		Øv øving 6	Innleveringsfrist: Ø5 Demonstrasjonsfrist: Ø4
8	Forelesning 9 -		Øv øving 7	Innleveringsfrist: Ø6 Demonstrasjonsfrist: Ø5
9	Ingen forelesninger Inspeøøving 1			
10	Forelesning 10 -		Øv øving 8	Innleveringsfrist: Ø7 Demonstrasjonsfrist: Ø6
11	Forelesning 11 -		Øv øving 9	Innleveringsfrist: Ø8 Demonstrasjonsfrist: Ø7
12	Forelesning 12 -		Øv øving 10	Innleveringsfrist: Ø9 Demonstrasjonsfrist: Ø8
13	Forelesning 13 -		Øv øving 11	Innleveringsfrist: Ø10 Demonstrasjonsfrist: Ø9
14	Forelesning 14 -		Øv øving 12	Innleveringsfrist: Ø11 Demonstrasjonsfrist: Ø10
15	Ingen forelesninger Påskeferie			
16	Forelesning 15 -			Innleveringsfrist: Ø12 Demonstrasjonsfrist: Ø11
17				Innleveringsfrist: - Demonstrasjonsfrist: Ø12

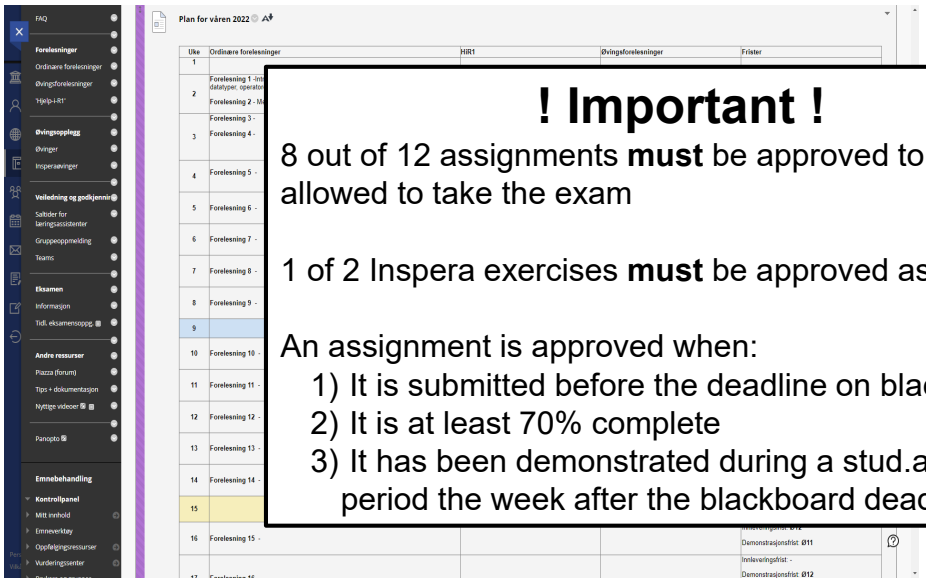
Assignment 1 is due on:

Friday, 20th of January

You can find the assignment plan on Blackboard!

Basert på det som vi behandler i forelesningene skal dere levere en øving hver uke, fra og med uke 3.

Assignment Schedule (tentative)



! Important !

8 out of 12 assignments **must** be approved to be allowed to take the exam

1 of 2 Inspera exercises **must** be approved as well

An assignment is approved when:

1) It is submitted before the deadline on blackboard
2) It is at least 70% complete
3) It has been demonstrated during a stud.ass. lab period the week after the blackboard deadline

Assignment 1 is due on:

0th of January

find the
ent plan
board!

For å få lov å gå på eksamen må minst 8 av 12 øvingene, og 1 av 2 inspera-øvinger være godkjent. Øvinger må demonstreres for en stud.ass før den godkjennes.

Today..

1. Answer questions about the course:
   - Who are these people?
   - Where do we find information?
   - When are the lectures and deadlines?
   - **What is the course about?**
   - Why do we care?
   - How do we do the coursework?

2. First steps in C++

May? Yes of course you may! <3

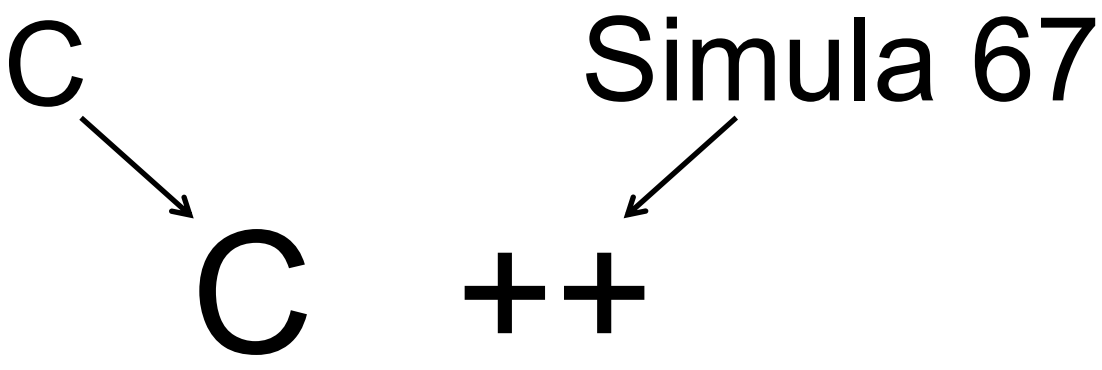
Da har vi kommet til det som faget handler om.

C++

Det som vi kommer til å bruke mest tid på er programmeringsspråket C++.

C ++

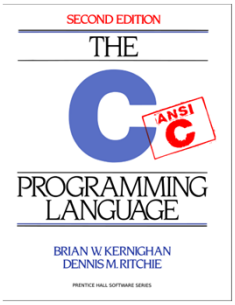
På en måte kan dette navnet deles opp i to deler; C, og ++.



Dette er fordi C++ er inspirert av to andre programmeringsspråk. I læreboka (kap. 22) skriver Bjarne Stroustrup at det var spesielt to språk som lå til grunn da han jobbet på C++; C, og Simula 67.

Dennis Ritchie

- Creator of C in 1972 at Bell labs
- One of the main developers of the UNIX operating system
- Won the Turing award in 1983
- Wrote *THE BOOK* on C



C (sammen med C++) er et av de mest brukte språk i verden. Store deler av operativsystemer (Linux, Windows, etc.) er vanligvis skrevet i C. I tillegg er programmeringsspråk som Python implementert i C.

The C Programming Language

- C is itself one of the most popular programming languages in use today
- Created to be simpler than assembly

Assembly

```
circleArea:
        sub     sp, sp, #16
        str     s0, [sp, #12]
        ldr     s0, [sp, #12]
        fcvt    d1, s0
        adrp    x8, .LCPI0_0
        ldr     d2, [x8, :lo12:.LCPI0_0]
        fmul    d1, d2, d1
        ldr     s0, [sp, #12]
        fcvt    d2, s0
        fmul    d1, d1, d2
        fcvt    s0, d1
        add     sp, sp, #16
        ret
```

C

```
float circleArea(float radius) {
    return M_PI * radius * radius;
}
```


Ritchie hadde behov for et språk som var lettere å bruke enn Assembly (se kodeeksempel) da han jobbet med utviklingen av operativsystemet UNIX. Språket var så godt at UNIX ble senere gjenskrevet i C.

Har du lyst å leke litt med assembly? Eksemplet som er vist finnes her: https://godbolt.org/z/evzsr9d5d

Kristen Nygaard & Ole-Johan Dahl

- Created the Simula programming language in 1962
- Established modern object oriented concepts
- Made Commander of the Order of St. Olav in 2000
- Won the Turing award in 2002

Kristen Nygaard Ole-Johan Dahl

Historien om det andre språket som lå til grunn for C++ kom fra et sted mye nærmere hjem: Norsk Regnesentral.

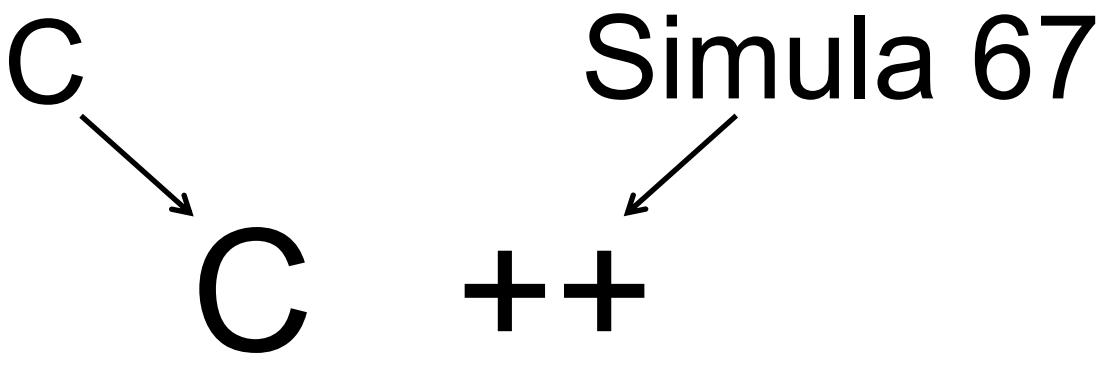
Kristen Nygaard hadde mange ideer om hvordan funksjonalitet i et program kan bli beskrevet i et programmeringsspråk. Disse ideene er hva som i dag er kjent som Objektorientert Programmering.

The Simula Programming Language

- First object-oriented programming language
- Influenced many of the most commonly used languages today (such as C++, Java, and C#)

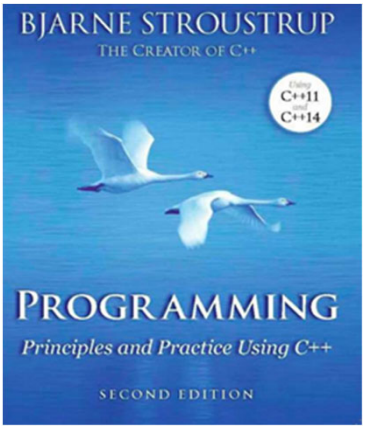
```
Class Rectangle (Width, Height); Real Width, Height;
Begin
    Real Area, Perimeter;

    Procedure Update;
    Begin
        Area := Width * Height;
        Perimeter := 2*(Width + Height)
    End of Update;

    Boolean Procedure IsSquare;
        IsSquare := Width=Height;
    Update;

    OutText("Rectangle created: "); OutFix(Width,2,6);
    OutFix(Height,2,6); OutImage
End of Rectangle;
```


Resultatet var programmeringsspråkene Simula I og Simula 67. Selv om det er nesten ingen som programmerer Simula nå, ligger konseptene brukt i språket for objektorientert programmering til grunn for noen av de mest populære språkene som er i bruk i dag.


C
Simula 67
C ++

Bjarne Stroustrup

- Creator of C++ in 1979
- Author of the course book



Bjarne er forfatteren av C++ språket. Da han utviklet det var objektorientert programmering kjent for å være for tregt og for ressurskrevende, og dermed ikke egnet til bruk i praktisk utvikling av dataprogram. Bjarne kombinerte språket C med ideene om implementering av objektorientert programmering fra Simula. Resultatet var en «bedre C»; C++.

What is C++?

- C++ is a compiled programming language

Så hva konkret er C++? Det er et kompilert programmeringsspråk.

What is C++?

- C++ is a compiled programming language
  - A C++ program is compiled (translated) into machine code
  - The compiler is the program that performs this translation

From C++ source code..

```
float circleArea(float radius) {
    return M_PI * radius * radius;
}
```

Compiler

.. To machine code

```
circleArea:
        sub     sp, sp, #16
        str     s0, [sp, #12]
        ldr     s0, [sp, #12]
        fcvt    d1, s0
        adrp    x8, .LCPI0_0
        ldr     d2, [x8, :lo12:.LCPI0_0]
        fmul    d1, d2, d1
        ldr     s0, [sp, #12]
        fcvt    d2, s0
        fmul    d1, d1, d2
        fcvt    s0, d1
        add     sp, sp, #16
        ret
```

program.exe

```
D10043FF
BD000FE0
BD400FE0
1E22C000
90000000
FD443801
1E610801
BD400FE0
1E22C000
1E600820
1E624000
910043FF
D65F03C0
```

Your program becomes a bunch of ones and zeroes that the processor can understand, also known as «a binary»

Et språk er kompilert dersom en «kompilator» oversetter all programkode til maskinkode på en gang.

Maskinkode er de 1ere og 0ere som prosessoren som kjører programmet forstår.

What is C++?

- C++ is a compiled programming language
  - A C++ program is compiled (translated) into machine code
  - The compiler is the program that performs this translation
- C++ is low-level: programs are executed on «bare metal»

program.exe

```
circleArea:
        sub     sp, sp, #16
        str     s0, [sp, #12]
        ldr     s0, [sp, #12]
        fcvt    d1, s0
        adrp    x8, .LCPI0_0
        ldr     d2, [x8, :lo12:.LCPI0_0]
        fmul    d1, d2, d1
        ldr     s0, [sp, #12]
        fcvt    d2, s0
        fmul    d1, d1, d2
        fcvt    s0, d1
        add     sp, sp, #16
        ret
```

```
D10043FF
BD000FE0
BD400FE0
1E22C000
90000000
FD443801
1E610801
BD400FE0
1E22C000
1E600820
1E624000
910043FF
D65F03C0
```


Fordi programmet oversettes til 1ere og 0ere, kjører den på prosessoren med prosessorens hastighet (som i dag er veldig raskt).

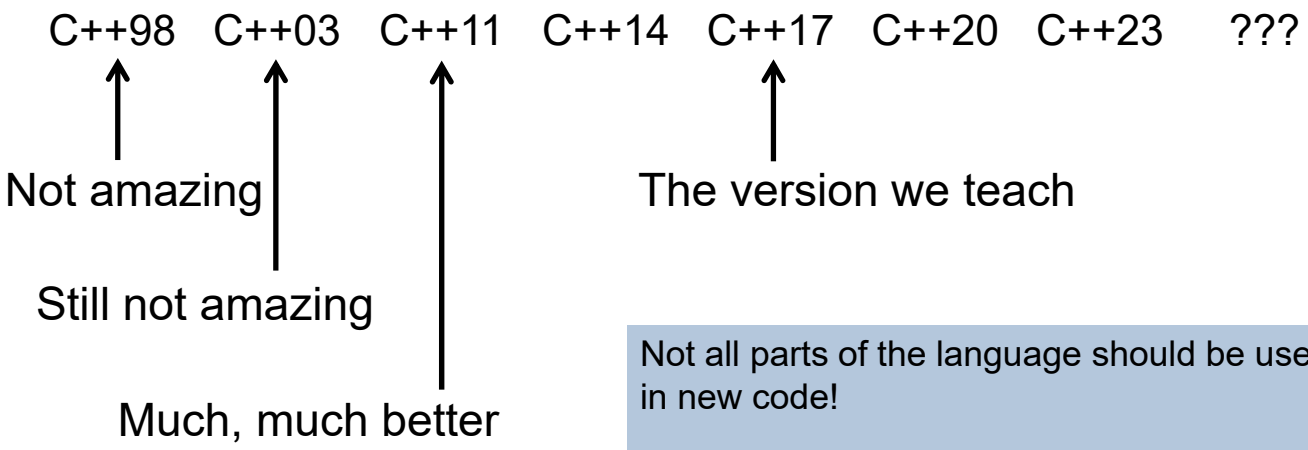
I tillegg gir C++ en stor grad med kontroll over hvordan programmet fungerer, som gir mulighet for bedre ytelse og mindre bruk av dataminne.

What is C++?

- C++ is a compiled programming language
  - A C++ program is compiled (translated) into machine code
  - The compiler is the program that performs this translation
- C++ is low-level: programs are executed on «bare metal»
- C++ is lean: you generally don't pay for what you don't use
- C++ is flexible: it allows many different programming styles

I tillegg er C++ lettvekt; språket er satt opp slik at du ikke «betaler» for det som ikke brukes, og det er støtter mange forskjellige programmeringsstiler.

Major language revisions



Not all parts of the language should be used in new code!

They only still exist to keep old code working.

Den opprinnelige versjonen C++98 mangler mange av de funksjoner som vi kommer til å bruke i emnet.
C++11 markerer et slags «Renessanse» i språkets utviklingen, etter det hadde stått stille i mange år.
Vi kommer til å bruke en av de nyere versjonene: C++17.

Today..

1. Answer questions about the course:
   - Who are these people?
   - Where do we find information?
   - When are the lectures and deadlines?
   - What is the course about?

•Why do we care?

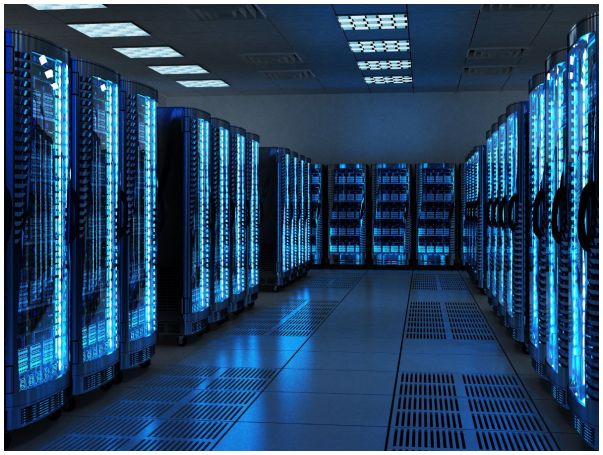
   - How do we do the coursework?

2. First steps in C++

May? Yes of course you may! <3

Da har vi kommet til det som kanskje er den viktigste spørsmålet: hvorfor er det relevant å lære seg C++?

Reason 1: C++ is used in all kinds of places



Første grunnen er at C++ brukes nesten overalt, fra dataspill til behandling av store datamengder i datasentre.

Reason 1: C++ is used in all kinds of places

Noen andre eksempler: nettlesere er vanligvis i stor grad skrevet i C++, samt et av de mest populære bibliotekene å lage brukergrensesnitt (laget av en Norsk bedrift!)

Reason 2:

Trends in Hardware

En annet grunn er litt mer langsiktig, og for å forstå den må vi først se tilbake i tid.

Early computers

IBM 704

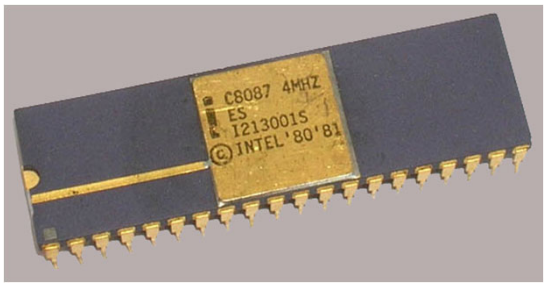
- Launched in 1954
- ~12 000 calculations per second (FLOPS)
- Likely consumed more than 100 kW (!!)

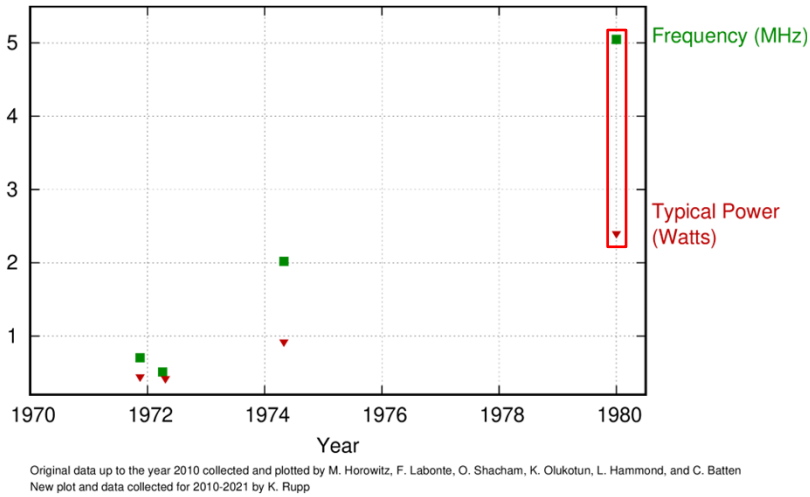
De første datamaskinene var store tunge maskiner som kunne gjøre noen tusen beregninger per sekund. Samtidig brukte de store mengder med energi (100kW).

Til sammenligning: en vannkoker bruker vanligvis omtrent 1-2kW, så maskinen som er vist bruker like mye energi som 50 vannkokere som er slått på samtidig!

1970 - 1980

Intel 8087





- Launched in 1979
- ~5 000 000 instructions per second
- Consumed 2.4W of power

Så ble integrerte kretser oppfunnet i 1958, og de var både langt raskere og brukte nesten ingen energi i forhold til de datamaskinene som fantes før.

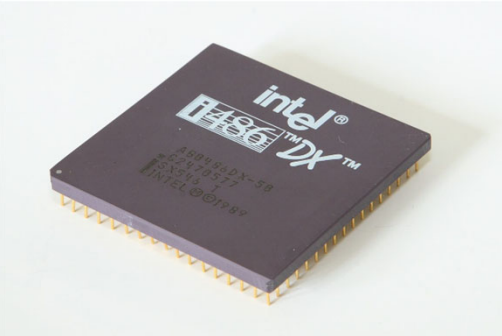
Prosessoren som er vist er:

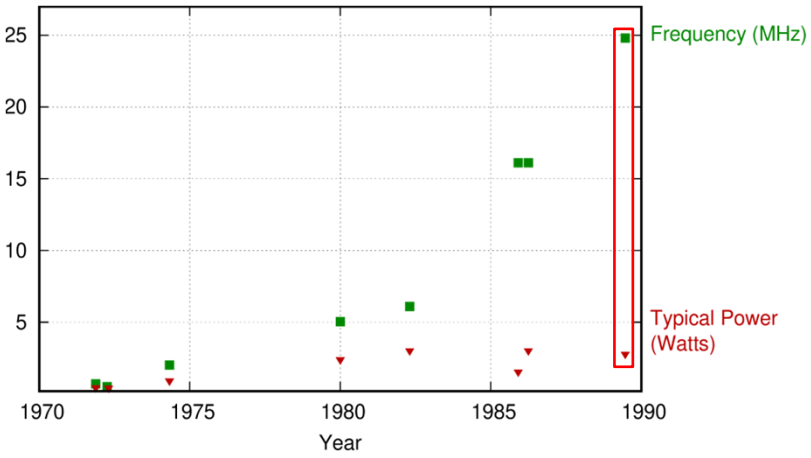
> 400x raskere enn IBM maskinen som ble vist på forrige bilde

> bruker omtrent 40 000x mindre energi (pr. beregning)

1980 - 1990

Intel i486

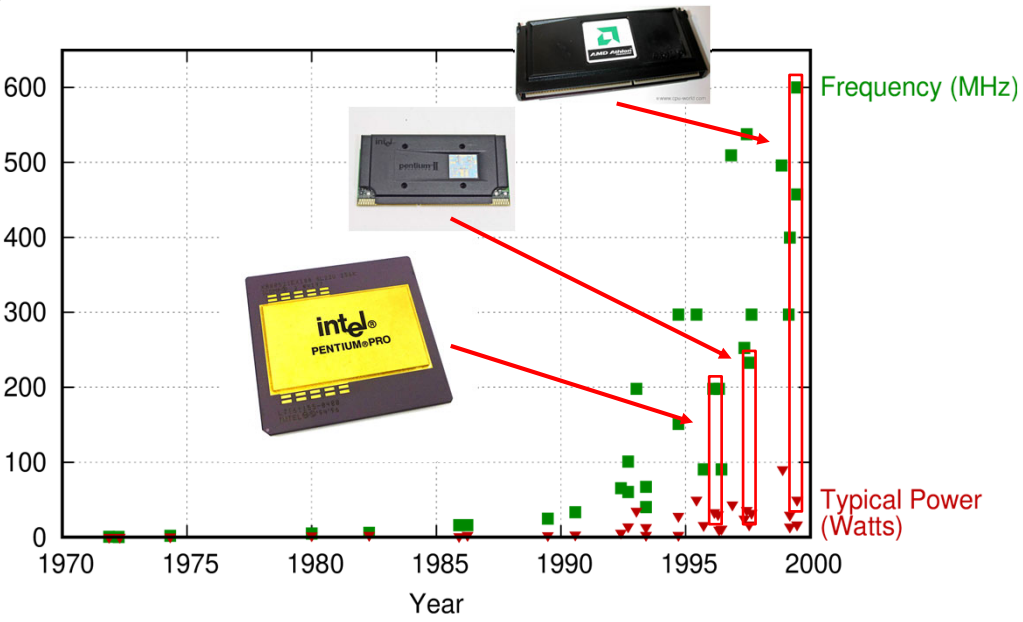




- Launched in 1989
- ~25 000 000 instructions per second
- Consumed 2.8W of power

Utviklingen av prosessorene fortsatt gjennom 80-tallene, som resulterte i en femdobling av ytelse over en periode på 10 år

1990 - 2000

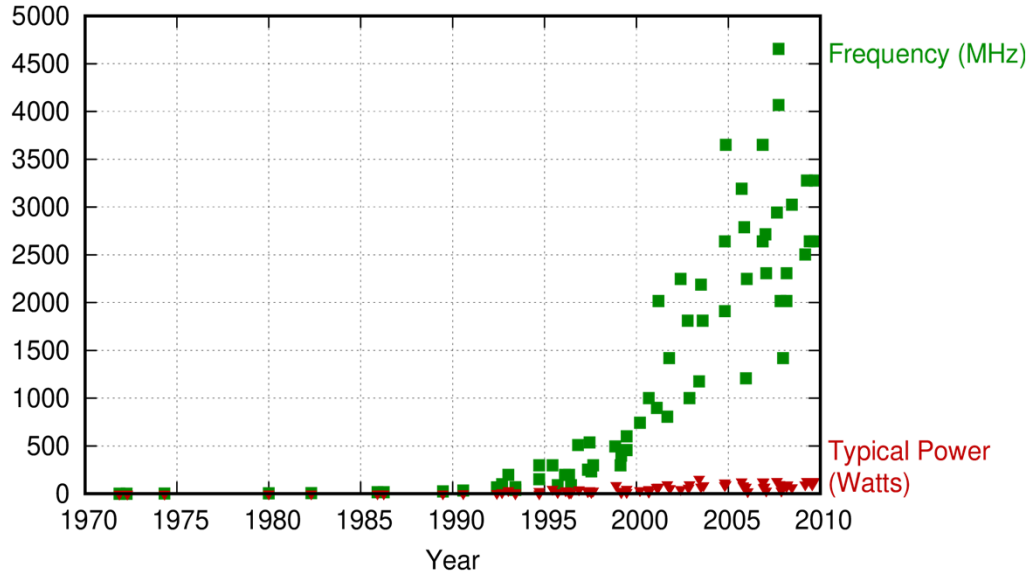


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

På 90-tallet ser vi igjen at den samme trenden fortsetter enda raskere. Det var en periode hvor ytelsen ble doblet hvert år, uten at strømforbruket doblet seg samtidig.

Intel Pentium pro: 35W, 200MHz
Intel Pentium II: 35W, 233 MHz
Athlon 600: 50W, 600MHz

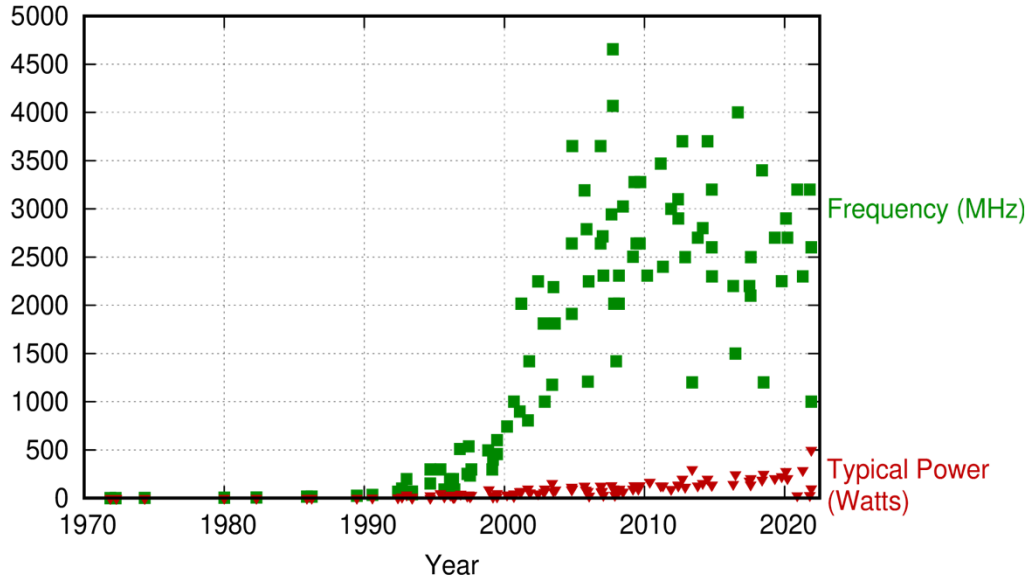
1990 - 2010



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

De 10 årene etter millenniumskiftet resulterte igjen i langt mer ytelse enn før, men endringen skjer ikke like raskt som før.

2010 - 2023



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

Og her ser vi en ny trend: klokkefrekvensen til prosessorene øker ganske lite. Grafikken viser hovedsakelig prosessorer laget til servere og ikke de som er laget for personlige maskiner (som pleier å ha lavere klokkefrekvens).

Likevel ser vi at forbedringen på prosessorene går stadig langsommere. Ikke bare på klokkefrekvens. I stedet har fabrikantene nå gått over til å bygge inn flere kjerner (i hver prosessor-brikke).

Reason 2:

Hardware improvements are stagnating.

Using the available processing power efficiently will thus become increasingly important.

Og da kommer vi til den andre grunnen for hvorfor det er viktig å lære seg C++: det er fortsatt en god del innovasjon innen prosessorer, men de forbedringene blir hvert år stadig mindre.

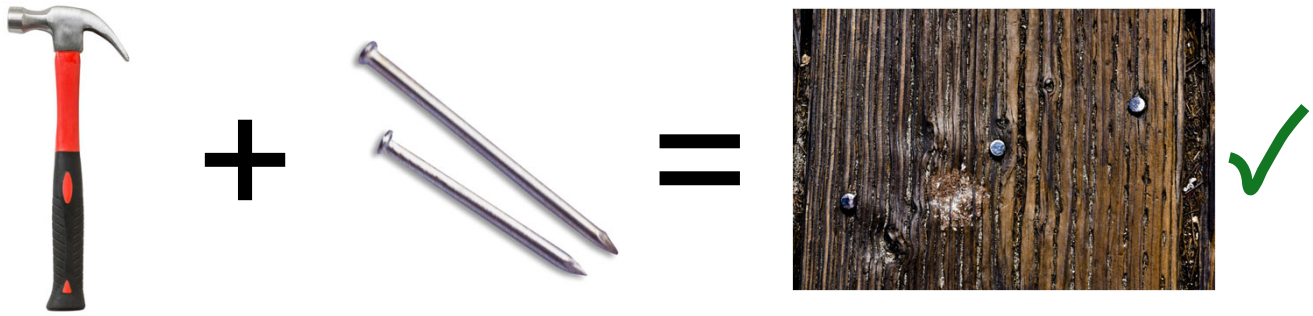
Reason 3:

Grunn 3

Meanwhile, C++ is a tool..

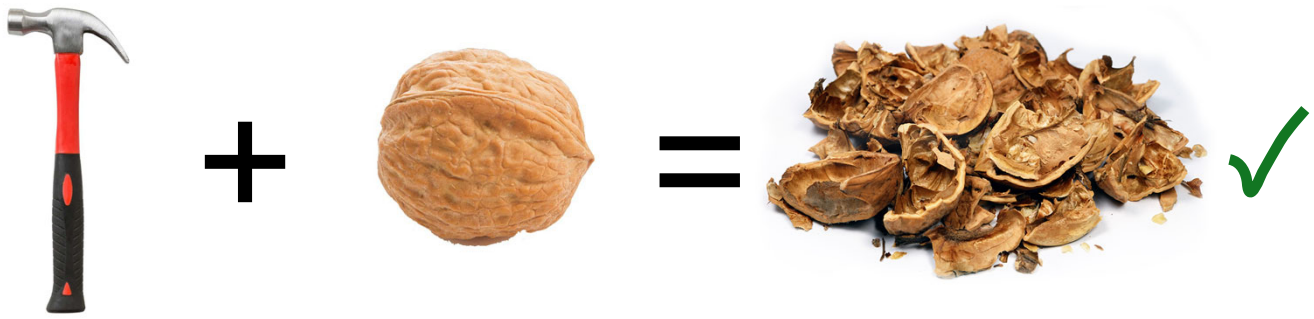
Samtidig er C++ et verktøy. Så når vi skal snakke om kjennskap til C++ er det viktig at vi ser både på fordelene og ulempene. Vi kan da gjøre et bedre valg for å bestemme hvilket verktøy som passer best for jobben vi prøver å gjøre.

C++ is a tool...



Dette er fordi et verktøy kan brukes for å løse problemer

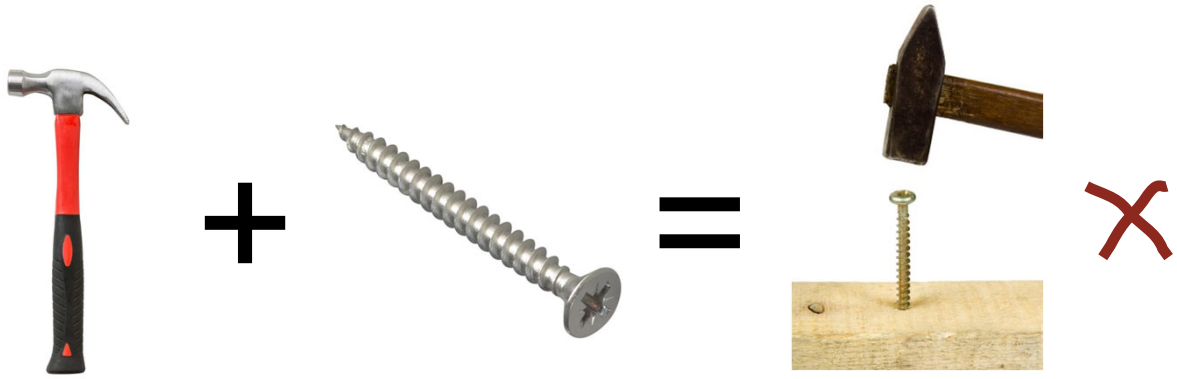
C++ is a tool...



.. Which works well for many applications

Og disse kan være mange ulike problemer

C++ is a tool...



.. Which works well for many applications,
but not for others.

Men samtidig går det ikke alltid.

Any tool has advantages and disadvantages

C++	Python
• ???	• ???

Så det vi skal gjøre nå er å sammenligne C++ med språket som ble benyttet i ITGK: Python.

Let’s do a speed test..

$$\sum_{i=1}^{100,000,000} i = 1 + 2 + 3 + (\dots) + 99,999,999 + 100,000,000$$

Python

C++

La oss begynne med å sammenligne språkenes hastiget. Her har vi en sum som skal beregnes for tallene fra 1 til hundre millioner.
De to programmene er ganske enkle, og er implementert omtrent helt likt i begge språkene.
Vi måler tiden det tar hvert språk å utføre programmet.

Let’s do a speed test..

$$\sum_{i=1}^{100,000,000} i = 1 + 2 + 3 + (\dots) + 99,999,999 + 100,000,000$$

Python: 6.001s

C++

Det tok Python litt over 6 sekunder.

Let's do a speed test..

$$\sum_{i=1}^{100,000,000} i = 1 + 2 + 3 + (\dots) + 99,999,999 + 100,000,000$$

Python: 6.001s

C++: 0.149s (40.3x faster!)

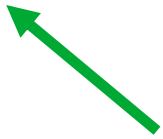
Og C++ omtrent 150 millisekunder.
Her ser vi forskjellen mellom et språk som er interpretert og et språk som er kompilert.
Fordi C++ kompileres til maskinkode klarer den å kjøre litt over 40x raskere i dette eksempelet.

Any tool has advantages and disadvantages

C++	Python
• Code runs at the speed the processor is capable of	• Code runs slower

Let's do a speed test..

$$\sum_{i=1}^{100,000,000} i = 1 + 2 + 3 + (\dots) + 99,999,999 + 100,000,000$$



The result of this sum is always the same!

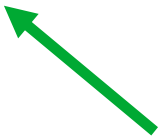
Python: 6.001s

C++: 0.149s (40.3x faster!)

La oss se igjen på sum-eksemplet, fordi her kan vi se et fordel til.
Moderne kompilatorer er nemlig veldig avansert, og kan gjenkjenne at summen aldri kommer til å endre seg.
Det gir oss en mulighet å slå på automatisk optimalisering av koden vi har skrevet.

Let’s do a speed test..

$$\sum_{i=1}^{100,000,000} i = 1 + 2 + 3 + (\ldots) + 99{,}999{,}999 + 100{,}000{,}000$$



The result of this sum is always the same!

Python: 6.001s

C++: 0.149s (40.3x faster!)

C++ (optimised): 0.002s (3000x faster!)

Etter vi har gjort det kjører programmet på 2 millisekunder; 3000x raskere enn Python programmet. Dette er vel et ekstremt eksempel, men det viser en vesentlig fordel med kompilerte språk.

Any tool has advantages and disadvantages

C++	Python
• Code runs at the speed the processor is capable of • Code can be automatically optimised • A number of common mistakes are caught by the compiler	• Code runs slower • Options for optimisating code are limited (or require using C++) • Some common errors may not be caught until the program is run

Kanskje har du skrevet et Python-program som kjørte i en time, og helt på slutten krasjet fordi du prøvde å skrive ut resultatene, men glemte å oversette en datatype. Det er en type feil som er vanskelig å oppdage i Python (men det finnes "type hints" i nye versjoner av Python som prøver å løse dette problemet). Det finnes flere slike typer feil. Med C++ vil programmet da ikke kompilere i det hele tatt.

Any tool has advantages and disadvantages

C++	Python
• Code runs at the speed the processor is capable of • Code can be automatically optimised • Several common mistakes are caught by the compiler • Tight control over program	• Code runs slower • Options for optimisating code are limited (or require using C++) • Some common errors may not be caught until the program is run • Limits to what you can control

Fordi C++ er relativt "lav-nivå" er det mulig å kontrollere tett hvordan programmet gjør jobben sin. Fordelen med det er at du kan unngå at programmet bruker for mye ressurser i datamaskinen, som da gjør at du kan spare på minnebruk og sørge for at programmet kjører raskere.

Any tool has advantages and disadvantages

C++	Python
• Code runs at the speed the processor is capable of	• Code runs slower
• Code can be automatically optimised	• Options for optimisating code are limited (or require using C++)
• A number of common mistakes are caught by the compiler	• Some common errors may not be caught until the program is run
• Tight control over program	• Limits to what you can control
• Massive number of libraries available	• Massive number of libraries available

Det finnes mange såkalte “biblioteker” til både Python og C++ som inneholder kode som løser mange forskjellige problemer. Dette betyr at når du skal skrive et program, og for eksempel ønsker å lage en PDF-fil, da finnes et bibliotek som kan gjøre det for deg, og du behøver ikke å finne ut hvordan man lager en PDF-fil selv.

Any tool has advantages and disadvantages

C++	Python
• Code runs at the speed the processor is capable of	• Code runs slower
• Code can be automatically optimised	• Options for optimisating code are limited (or require using C++)
• A number of common mistakes are caught by the compiler	• Some common errors may not be caught until the program is run
• Tight control over program	• Limits to what you can control
• Massive number of libraries available	• Massive number of libraries available
• No single go-to package manager	• Has a package manager (pip) for easily installing libraries

For å installere slike biblioteker finnes et verktøy som heter pip i Python. I C++ derimot finnes ikke et slikt standard-program for å installere biblioteker. Det har vært noen initiativer, og på Linux kan mange lett installeres gjennom pakkehåndteringssystemene, men ingenting er like populært og tilgjengelig som pip med Python.

Any tool has advantages and disadvantages

C++	Python
• Code runs at the speed the processor is capable of	• Code runs slower
• Code can be automatically optimised	• Options for optimisating code are limited (or require using C++)
• A number of common mistakes are caught by the compiler	• Some common errors may not be caught until the program is run
• Tight control over program	• Limits to what you can control
• Massive number of libraries available	• Massive number of libraries available
• No single go-to package manager	• Has a package manager (pip) for easily installing libraries
• Compilation is time consuming	• Program runs instantly

Det er også verdt å nevne at kompilering av C++ kode tar tid. Når man ønsker å kjøre et Python program, begynner programmet med en gang.

Today..

1. Answer questions about the course:
   - Who are these people?
   - Where do we find information?
   - When are the lectures and deadlines?
   - What is the course about?
   - Why do we care?
   - **How do we do the coursework?**

2. First steps in C++

May? Yes of course you may! <3

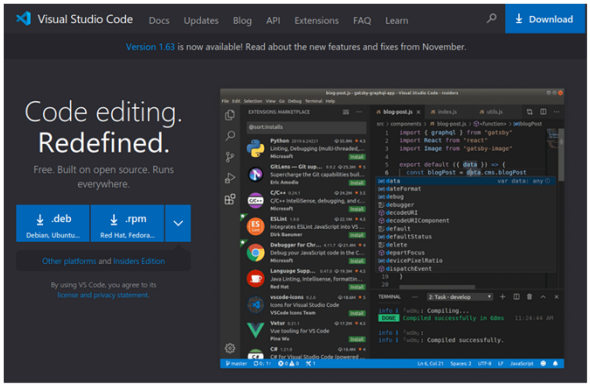
Vi har tidligere nevnt at det er obligatoriske øvinger i emnet. Nå skal vi se på verktøyene som vi bruker for å løse dem.

We will use the Visual Studio Code editor

Vi bruker hovedsakelig Visual Studio Code. Vi har laget en utvidelse («TDT4102-tools extension») som automatisk setter opp alt som trenges for å kompilere og kjøre et C++ program.

Assignment 0 (Abridged Edition)

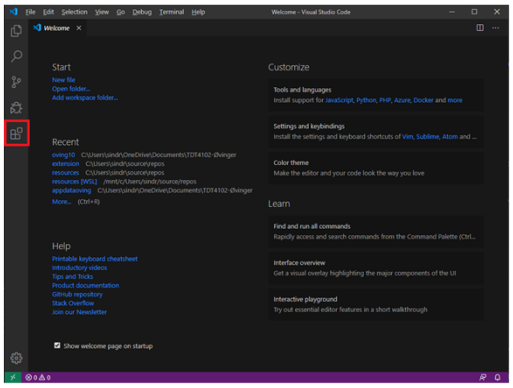
1. Go to https://code.visualstudio.com



2. Download the installer
3. Run the installer
   (refer to assignment 0 for which settings to use)

4. Open VS Code

5. Install the TDT4102 extension



6. Use Ctrl+Shift+P, then «install required tools»

Alt som dere må gjøre er å installere Visual Studio Code, og deretter utvidelsen som noen dyktige studenter har laget. Den laster ned alt man trenger for å komme i gang.

Vi anbefaler at dere gjør oppgave 0 snarest mulig, i tilfelle det oppstår tekniske problemer.
(Sjekk Piazza, siden flere sikkert har samme problem og løsning.)

Demonstration:

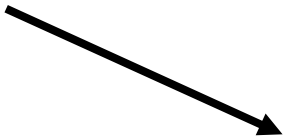
Installation on Windows and Mac

Vi skal nå demonstrere installasjonen av VS Code og utvidelsen.

How to run the code examples

Each example we discuss during the lectures is available in VS Code.

1. Press Ctrl+Shift+P (or Cmd+Shift+P on Mac)
2. Use the command «Create Project from TDT4102 Template»
3. Select «Lectures»
4. Select the lecture number
5. Select the example number shown on the slide



Example 1

Vi kommer til å gå gjennom en del kodeeksempler i forelesningene. Dere kan kjøre disse selv i etterkant gjennom VS Code.

Referansen til eksemplet blir alltid vist nederst til høyre.

Today..

1. Answer questions about the course:
   - Who are these people?
   - Where do we find information?
   - When are the lectures and deadlines?
   - What is the course about?
   - Why do we care?
   - How do we do the coursework?

2. **First steps in C++**

May? Yes of course you may! <3

Nå har vi sett hva emnet innebærer, og vi kan gå videre med å skrive noen enkle C++ program!

Assignment 1

Dette markerer hvor vi begynner med å dekke alt av pensum som brukes i øving 1.

Demonstration:

Basics of Compiling and Running C++ Programs

Example 1

Summary of everything shown:

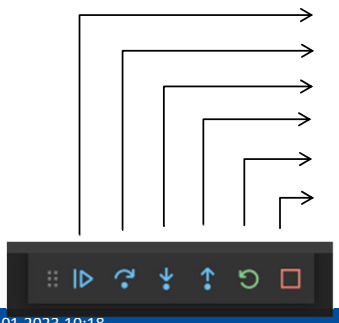
- Run a program by clicking on Run > Start Debugging
- Code must be placed inside main()
- Each “sentence” must be terminated using a ;
- The equivalent of Python’s `print()` function is `cout << "message" << endl;`
- Unlike Python, whitespace and indentation has no meaning in C++
- The equivalent of Python’s `import` is `#include ""`
- The debugger allows you to run your program one line at a time.

Example 1

```
3   int main() {
4       cout << "A very Happy New Year!" << endl;
```

Breakpoint: when using the “Start Debugging” option to run your program, you can click in the left hand margin in VS code to set a breakpoint. The program will pause execution and let you inspect what is going on when it encounters a line with one active.

When it does so, the following panel will show up:



“Continue”:	Stop at the next breakpoint
“Step over”:	Stop at the next line in this function
“Step into”:	Stop at the next line of code
“Step out”:	Stop at the next line of code after the function exits
“Restart”:	Restart the program
“Stop”:	Stop the program

Example 2

Mens jeg kjører programmet kommer jeg til å demonstrere debuggeren. Denne kan brukes til å gå gjennom programmet linje for linje mens det kjører.

- Images used:

https://www.bigw.com.au/product/dunn-claw-hammer/p/14630
http://diymaketech.blogspot.com/2012/10/copper-plated-nail.html
https://publicdomainpictures.net/en/view-image.php?image=132764&picture=three-nails-in-a-plank
https://texassandblasting.com/media-blasting/walnuts1/
https://www.hipstitchabq.com/shop/c/p/Crushed-Walnut-Shells-x37655395.htm
https://www.freeimageslive.co.uk/free_stock_image/envelope-jpg
https://pokemon.fandom.com/wiki/Accelgor
https://pokemon.fandom.com/wiki/Ekans
https://www.deviantart.com/jenske05/art/pokemon-X-Y-battle-scene-360546240
https://www.wellyoufound.me?hidinghere=34&show_img=3598765&format=jpg
https://www.pinclipart.com/picdir/middle/534-5345877_python-logo-clipart.png
http://noclipmode.com/wp-content/uploads/2011/01/bottle-smash.jpg
https://commons.wikimedia.org/wiki/File:Bjarne-stroustrup_(cropped).jpg
https://1.bp.blogspot.com/-4s412BNDQTs/XhJFclNPpxI/AAAAAAAAJ9Q/VhOM-yEADoMXUxoOoLsUt_RzsGiITD5rACLcBGAsYHQ/s1600/programming-principles-and-practice-using-c-2nd-edition-bjarne-stroustrup.jpg
By Denise Panyik-Dale - https://www.flickr.com/photos/dpanyikdale/5740011186/, CC BY 2.0,
https://commons.wikimedia.org/w/index.php?curid=20276654
http://cs-exhibitions.uni-klu.ac.at/fileadmin/template/pictures/NygaardDahl_leg.jpg
https://simplecore.intel.com/itpeernetwork/wp-content/uploads/sites/38/2018/09/hpc-ai-servers.jpeg
https://developer-tech.com/wp-content/uploads/sites/3/2020/05/unreal-engine-5-ps5-demo-xbox-series-x-gaming-tech-performance-game-developer-2048x1152.jpg
https://www.qt.io/hubfs/_website/QtV2/qt_product_overview_right.png#keepProtocol
By Mozilla Corporation - https://phabricator.services.mozilla.com/D41016, MPL 2, https://commons.wikimedia.org/w/index.php?curid=81490422
http://pngimg.com/uploads/chrome_logo/chrome_logo_PNG17.png

Og dette er lenker til bildene jeg har brukt i forelesningen

https://www.flickr.com/photos/dpanyikdale/5740011186/ gir «404: not found»