Objektorienteret Programmering i C++

6. februar 2025

Ole Dolriis - CV

1953	Født i København
1976-1977	Programmør Danfoss
1977-1979	Programmør Regnecentralen
1979-1983	Softwareudvikler og projektleder ELSAM (Ørsted)
1983-1989	Softwareudvikler og projektleder Mærsk Data
1989-1993	Projektchef Mærsk Data
1993-2001	Underviser TietgenSkolen (Datamatikeruddannelsen)
2001-2006	Lektor Ingeniørhøjskolen Odense Teknikum
2006-	Lektor/Ingeniørdocent SDU
2024-	Underviser Københavns Erhvervsakademi (KEA)

Uddannelse

High School Graduate	1972	Pennsylvania, USA
Mat./fys. Student	1974	Mulernes Legatskole
EDB-assistent	1976	Tietgenskolen
HD (regnskab)	1986	Odense Universitet
Bachelor i datalogi	1998	Aalborg Universitet
Cand. scient. i datalogi	2007	Aalborg Universitet
Cand. mag. i historie	2016	Syddansk Universitet

Row call

Studieordning

Målbeskrivelse - viden

Objektorienteret programmering i C++ (CPP):

- grundlæggende objektorienteret C++ programmering
- kontrolstruktur: betingede udtryk, løkkestrukturer
- input/output: tastatur, skærm
- standard biblioteksfunktioner
- klasser, objekter, konstruktør, destruktør
- pointere, dynamisk hukommelsesallokering, funktions- og operatoroverload
- containers, arrays
- arv

Studieordning

Målbeskrivelse - færdigheder

Efter gennemførelse af kurset kan den succesfulde studerende:

•Skrive og kompilere syntaktisk korrekt C++ kode inden for de områder der er angivet i fagbeskrivelsen under punktet “Målbeskrivelse - viden”

Målbeskrivelse - kompetencer

Efter gennemførelse af kurset kan den succesfulde studerende:

•analysere relativt simple problemstillinger og dertil udvikle et program til løsning af problemerne

skrive relativt simple, velfungerende programmer i et objektorienteret sprog på baggrund af en specifikation

Eksamen

3 timers skriftlig eksamen

Intern censur

(Meget) Foreløbig kursusplan OOP C++

6:	Introduktion; udviklingsmiljø; det første program
7:	Simple programelementer; sekvens, operatorer; intro. til klasser
8:	Selektion og iteration; opgaver i Dato-klassen; arrays
9:	Associering, aggregering og arv
10:	Associering, aggregering og arv (fortsat)
11:	Pointere; dynamisk hukommelsesallokering; operatoroverload
12:	Opsamling og konsolidering
14:	Opsamling og konsolidering
15:	Opgaveløsning
16:	Introduktion til datastrukturer
17:	Repetition og opgaveløsning
18:	Prøveeksamen

Udviklingsmiljø - IDE

Microsoft

Visual Studio

2022 eller tidligere

Undervisningsmaterialer?

- Egenproducerede/YouTube-videoer
- *”Engineering problem solving with C++”, 4. udg.*, Dolores M. Etter & Jeanine A. IngberGregorie *et al.*
- ”*Professional C++”, xth ed*., Gregorie *et al.*
  - www.cplusplus.com/doc/tutorial
  - www.cplusplus.com/reference

ChatGPT

- Har allerede medført store ændringer i softwareudviklingen i industrien.
- Hvad kan den i forhold til programmering?
- Hvordan skal I bruge den?
- Hvor meget har I allerede brugt den?

Citat

”Alt det der med syntaks

GLEM DET!

ChatGPT kan klare det.

De skal blive gode til at læse og finde fejl i kode, de ikke selv har skrevet.”

C++

er et nærmest astronomisk stort programmeringssprog med et klassebibliotek, så Guderne må sig forbarme, og intet levende menneske har detaljeret overblik over.

Ergo, I må ikke forvente, at jeg kan hjælpe jer med det hele ☺

C++

er en stor pædagogisk udfordring som begyndersprog.

Compileren er tit både besværlig og uforståelig – det samme kan fejlmeddelelserne være.

En lidt overfladisk tilgang her i starten kan derfor være nyttig i forhold til realistiske læringsmål.

Det er ikke meningen, at I skal forstå det hele til bunds her på 2. semester.

Underviserens opgave

er at sørge for, at hovedfeltet ikke falder for tidsgrænsen. Det er *ikke* at køre om kap med udbryderne hen over det næste bjerg.

Hiv fat i trøjen på ham, hvis I synes, det går for stærkt.

Differentieret undervisning tilbydes.

Den studerendes opgave

er at blive så god som mulig til at programmere i den begrænsede tid, der er til rådighed, primært gennem opgaveløsning og ikke så meget ved lærebogslæsning.

At bestå eksamen med en hæderlig karakter bør **ikke** være det eneste mål.

Hvordan skal vi gribe det an?

- Vil I helst have en lærebog?
- Gider I at læse?
- Eller vil I hellere se YouTubes eller andre videoer?
- Gider I at høre efter, hvad jeg siger i timerne, eller vil I hellere sidde og fedte med et eller andet på jeres computer eller telefon?

Hvad er et program?

Et **computerprogram**, eller blot **program**, er en samling instruktioner, som sætter **computeren** i stand til at løse en bestemt opgave. Software er en generel betegnelse for samlinger af **programmer**.

Typer af systemer

- *Informationssystemer* (IS) er forholdsvis store applikationer som håndterer store datamængder og interagerer med (mange) andre systemer.
- Udvikling af *indlejrede systemer* er som oftest en ganske anden historie, fordi de som regel er forholdsvis små og har en 'vandtæt' specifikation.
- *Kunstig intelligens* og/eller *maskinlæring* er så en helt tredje historie, som kort fortalt går ud på at løse særdeles vanskelige problemer.

Om programmer…

"… et program er kun fejlfrit, indtil man finder den næste fejl."

"At skrive et program har mere til fælles med at skrive en bog end med at konstruere en hvilken som helst fysisk genstand."

Om programmering

1. A få et lille program til at virke er meget nemt, og det kan bevises matematisk, at det er fejlfrit.
2. At skrive et stort program, som er fejlfrit, er umuligt, og det kan også bevises.
3. At få et stort program til at virke bare sådan nogenlunde er overordentlig vanskeligt (tæt på raketvidenskab).

Mange mennesker, som tror de ved noget om programmering, er kun bekendt med punkt 1.

Om programmering

Man kan ikke lære at programmere ved at læse en bog.

Det kræver mange timers øvelse og mindst to års fuldtidsbeskæftigelse at blive fuldbefaren.

Det er et håndværk.

Succesfuld programmerings forudsætninger

- Omtanke
- Præcision
- Tålmodighed
- Disciplin

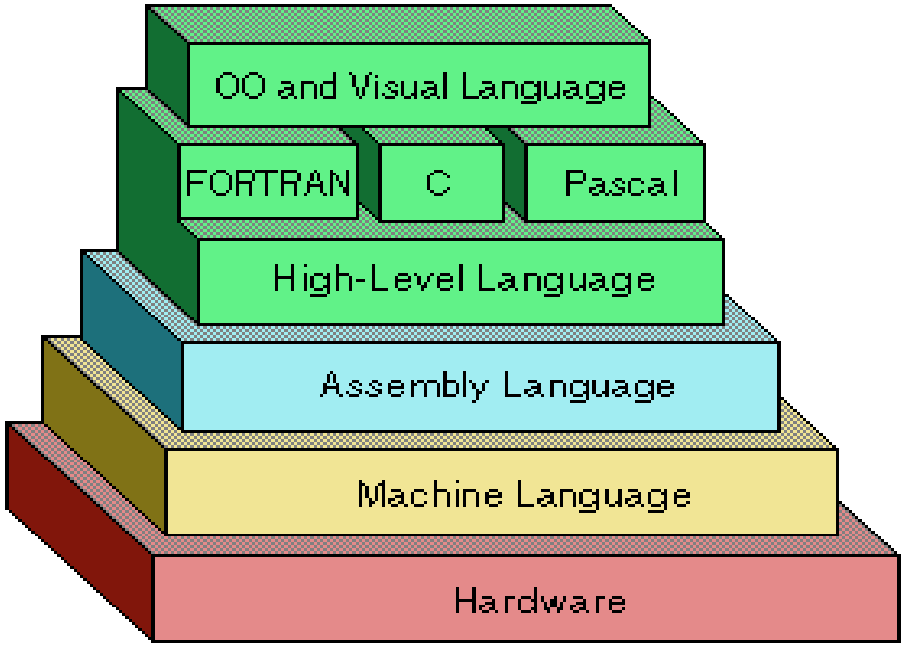
Man skal være forberedt på at komme til at bruge en masse tid på at finde helt trivielle fejl.

Man kan med fordel tillægge sig nogle gode vaner fra start af.

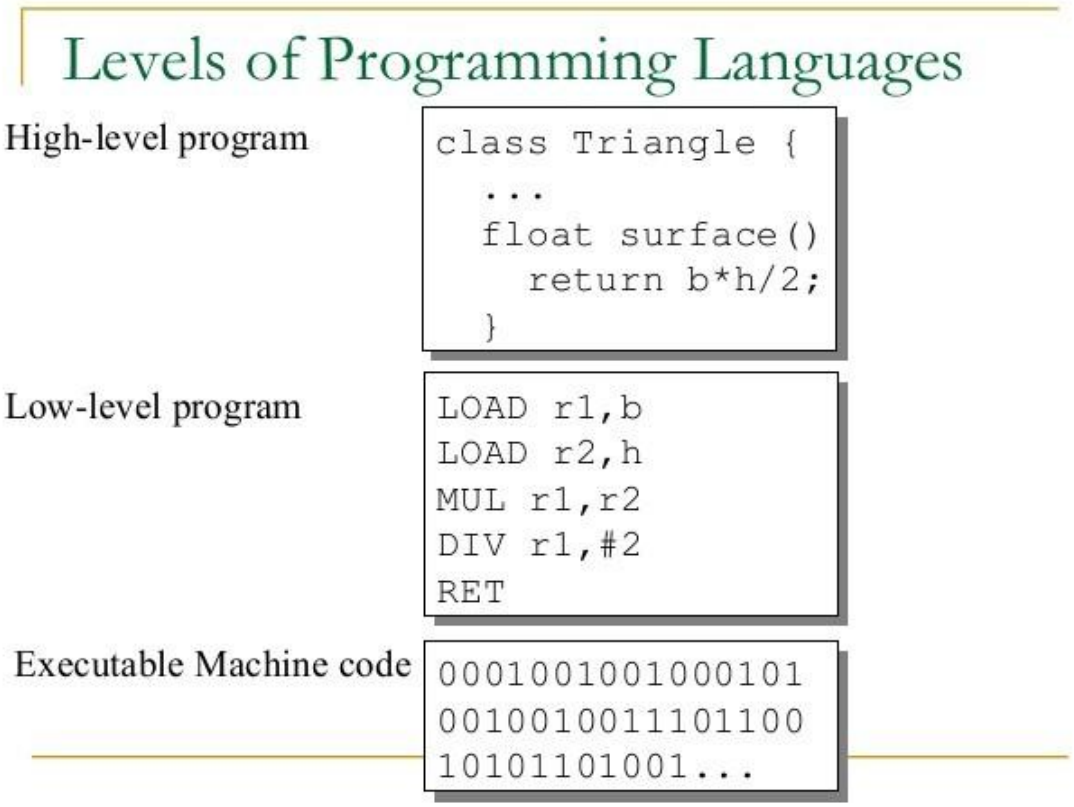
Programmeringssprog

Cobol	Fortran	PL1
Pascal	Python	LISP
Algol	C	C++
Assembler	Java	C#
HTML	JavaScript	XML

Niveauer


OO and Visual Language
FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware

High og low level programmer



Hvad kan vi?

- Sekvens
- Selektion
- Iteration

Tilgangen

Vi går i gang med at skrive de første programmer allerede nu.

Derfor vil der være en hel del ting, som I måske ikke forstår fra start af; men det kommer I til ”as we go”.

Det er min erfaring, at det er den mest effektive pædagogiske metode.

Men jeg er naturligvis helt med på, at folk lærer forskelligt.

Det første program

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello Class!" << endl;
    return 0;
}
// C++ er case-sensitiv (A != a)
```

Hvorfor OO og hvad går det ud på?

- OOP er det største fremskridt i programmeringens historie.
- Et program består af interagerende objekter.
- Objekterne er indkapslede og arbejdsdelingen er klar.
- Small is beautiful.
- Genbrugelige komponenter.

Klassebegrebet

En *klasse* i C++ (og andre objektorienterede sprog) kan beskrives som en skabelon for *objekter*. Objekter er karakteriseret ved at have:

- Identitet
- Tilstand
- Adfærd

Objekter er forekomster eller instanser af klasser.

En simpel klasse

```
#include <string>
using namespace std;
class Bil
{
	public:
		Bil();
		Bil(string, int);
		string getRegNr();
		int getAargang();
		~Bil();
	private:
		// Attributter er ALTID private
		string regNr;
		int aargang;
};
```

Data types

også kaldet

Primitive datyper

TYPE	DESCRIPTION	USAGE
`int`	Positive and negative integers, range depends on compiler	`int i = 7;`
`short (int)`	Short integer (usually 2 bytes)	`short s = 13;`
`long (int)`	Long integer (usually 4 bytes)	`long l = -7;`
`long long (int)`	Long long integer, range depends on compiler, but at least the same as long (usually 8 bytes)	`long long ll = 14;`

`unsigned int`
`unsigned short (int)`
`unsigned long (int)`
`unsigned long long (int)`
	Limits the preceding types to values >= 0	`unsigned int i = 2;`
		`unsigned short s = 23;`
		`unsigned long l = 5400;`
		`unsigned long long ll = 140;`

`float`	Floating-point numbers	`float f = 7.2f;`

(continued)

TYPE	DESCRIPTION	USAGE
`double`	Double precision numbers, precision is at least the same as for `float`	`double d = 7.2;`
`long double`	Long double precision numbers, precision is at least the same as for `double`	`long double d = 16.98L;`
`char`	A single character	`char ch = 'm';`
`char16_t`	A single 16-bit character	`char16_t c16 = u'm';`
`char32_t`	A single 32-bit character	`char32_t c32 = U'm';`
`wchar_t`	A single wide-character size depends on compiler	`wchar_t w = L'm';`
`bool`	true or false (same as non-0 or 0)	`bool b = true;`
`auto`	The compiler will decide the type automatically	`auto i = 7; // i will be an int`
`decltype(expr)`	The type will be the type of the expression `expr`	`int i = 7;` `decltype(i) j = 8; // j will also be an int`

Tekst

- Tekst gemmes i objekter af klassen `string`.
- Strings er arrays af datatypen `char`.
- For at bruge strings skal man skrive:
  `#include <string>`

Repræsentation

- 1 bit er mindste enhed: 0 eller 1
- 1 byte fylder 8 bits
- 1 `char` fylder 1 byte
- 1 `int` fylder 4 bytes
- 1 `double` fylder 8 bytes

ASCII alfabetet

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	Start of Header	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	Start of Text	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	End of Text	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	End of Transmission	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enquiry	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Acknowledgment	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Backspace	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	Horizontal Tab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	Line feed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	Vertical Tab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	Form feed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	Carriage return	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	Shift Out	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	Shift In	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	Data Link Escape	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	Device Control 1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	Device Control 2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	Device Control 3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	Device Control 4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	Negative Ack.	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Synchronous idle	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	End of Trans. Block	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Cancel	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	End of Medium	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Substitute	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Escape	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	File Separator	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	Group Separator	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	Record Separator	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	Unit Separator	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Del

asciicharstable.com

Forskellen på primitive datatyper og objekter (i runde tal)

Primitive datatyper har fast længde.

Det har objekter som udgangspunkt IKKE.

Operators

OPERATOR	DESCRIPTION	USAGE

`=`	Binary operator to assign the value on the right to the variable on the left.	`int i;`
		`i = 3;`
		`int j;`
		`j = i;`

`!`	Unary operator to complement the true/false (non-0/0) status of a variable.	`bool b = !true;` `bool b2 = !b;`

`+`	Binary operator for addition.	`int i = 3 + 2;`
		`int j = i + 5;`
		`int k = i + j;`

`-`
`*`
`/`
	Binary operators for subtraction, multiplication, and division.	`int i = 5-1;`
		`int j = 5*2;`
		`int k = j / i;`

`%`	Binary operator for remainder of a division operation. Also referred to as the *mod* operator.	`int remainder = 5 % 2;`
`++`	Unary operator to increment a variable by 1. If the operator occurs after the variable or *post-increment*, the result of the expression is the unincremented value. If the operator occurs before the variable or *pre-increment*, the result of the expression is the new value.	`i++;` `++i;`
`--`	Unary operator to decrement a variable by 1.	`i--;` `--i;`

OPERATOR	DESCRIPTION	USAGE
`+=`	Shorthand syntax for `i = i + j`	`i += j;`

`-=`
`*=`
`/=`
`%=`
	Shorthand syntax for
	`i = i - j;`
	`i = i * j;`
	`i = i / j;`
	`i = i % j;`
		`i -= j;`
		`i *= j;`
		`i /= j;`
		`i %= j;`

`&` `&=`	Takes the raw bits of one variable and performs a bitwise "AND" with the other variable.	`i = j & k;` `j &= k;`
`|` `|=`	Takes the raw bits of one variable and performs a bitwise "OR" with the other variable.	`i = j | k;` `j |= k;`

`<<`
`>>`
`<<=`
`>>=`
	Takes the raw bits of a variable and "shifts" each bit left (<<) or right (>>) the specified number of places.	`i = i << 1;`
		`i = i >> 4;`
		`i <<= 1;`
		`i >>= 4;`

`^` `^=`	Performs a bitwise "exclusive or" operation on the two arguments.	`i = i ^ j;` `i ^= j;`

Evalueringsrækkefølge for beregningsudtryk:

```
int i = 34 + 8 * 2 + 21 / 7 % 2;
```

Adding parentheses makes it clear which operations are happening first:

```
int i = 34 + (8 * 2) + ( (21 / 7) % 2 );
```

Breaking up the statement into separate lines makes it even clearer:

```
int i = 8 * 2;
int j = 21 / 7;
j %= 2;
i = 34 + i + j;
```

Har I set tutorials – om klasser?

https://www.youtube.com/watch?v=vz1O9nRyZaY

https://www.youtube.com/watch?v=b9wialxvcV

Mine kommentarer

- Problematisk navngivning
- Alt for mange kommentarer.
- Mutator-metoder er noget skrald, hvis attributterne er `private`.
- `const` er ikke særlig vigtig.
- 2'eren er ret forvirrende til sidst.
- Krøllede parenteser under hinanden.

Forslag modtages gerne

Hvis I finder nyttige tutorials, så gør dem venligst tilgængelige for alle.