

Objektorienteret Programmering i C++

20. februar 2025

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.

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. At 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.

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.

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;
};
```

Repræsentation

- 1 bit er mindste enhed: 0 eller 1
- 1 byte fylder 8 bits

Simple/primitive datatyper

int	heltal	4 bytes
Long	heltal	8 bytes
float	decimaltal	4 bytes
double	decimaltal	8 bytes
char	karakter	1 byte
bool	true/false	1 byte

Tekst

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

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

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.	<pre>int i; i = 3; int j; j = i;</pre>
!	Unary operator to complement the true/false (non-0/0) status of a variable.	<pre>bool b = !true; bool b2 = !b;</pre>
+	Binary operator for addition.	<pre>int i = 3 + 2; int j = i + 5; int k = i + j;</pre>
- * /	Binary operators for subtraction, multiplication, and division.	<pre>int i = 5-1; int j = 5*2; int k = j / i;</pre>
%	Binary operator for remainder of a division operation. Also referred to as the <i>mod</i> operator.	<pre>int remainder = 5 % 2;</pre>
++	Unary operator to increment a variable by 1. If the operator occurs after the variable or <i>post-increment</i> , the result of the expression is the unincremented value. If the operator occurs before the variable or <i>pre-increment</i> , the result of the expression is the new value.	<pre>i++; ++i;</pre>
--	Unary operator to decrement a variable by 1.	<pre>i--; --i;</pre>

OPERATOR	DESCRIPTION	USAGE
<code>+=</code>	Shorthand syntax for <code>i = i + j</code>	<code>i += j;</code>
<code>-=</code>	Shorthand syntax for	<code>i -= j;</code>
<code>*=</code>	<code>i = i - j;</code>	<code>i *= j;</code>
<code>/=</code>	<code>i = i * j;</code>	<code>i /= j;</code>
<code>%=</code>	<code>i = i / j;</code> <code>i = i % j;</code>	<code>i %= j;</code>
<code>&</code> <code>&=</code>	Takes the raw bits of one variable and performs a bitwise “AND” with the other variable.	<code>i = j & k;</code> <code>j &= k;</code>
<code> </code> <code> =</code>	Takes the raw bits of one variable and performs a bitwise “OR” with the other variable.	<code>i = j k;</code> <code>j = k;</code>
<code><<</code> <code>>></code> <code><<=</code> <code>>>=</code>	Takes the raw bits of a variable and “shifts” each bit left (<code><<</code>) or right (<code>>></code>) the specified number of places.	<code>i = i << 1;</code> <code>i = i >> 4;</code> <code>i <<= 1;</code> <code>i >>= 4;</code>
<code>^</code> <code>^=</code>	Performs a bitwise “exclusive or” operation on the two arguments.	<code>i = i ^ j;</code> <code>i ^= j;</code>

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;
```