

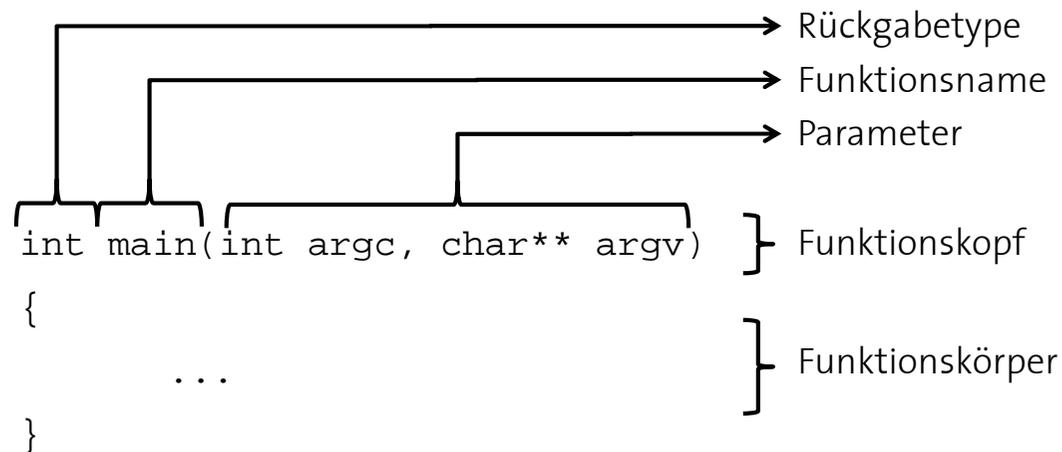
Computer Vision
and Geometry Lab

Informatik I for D-MAVT

Exercise Session 3

Übung 1

■ Kommentare...



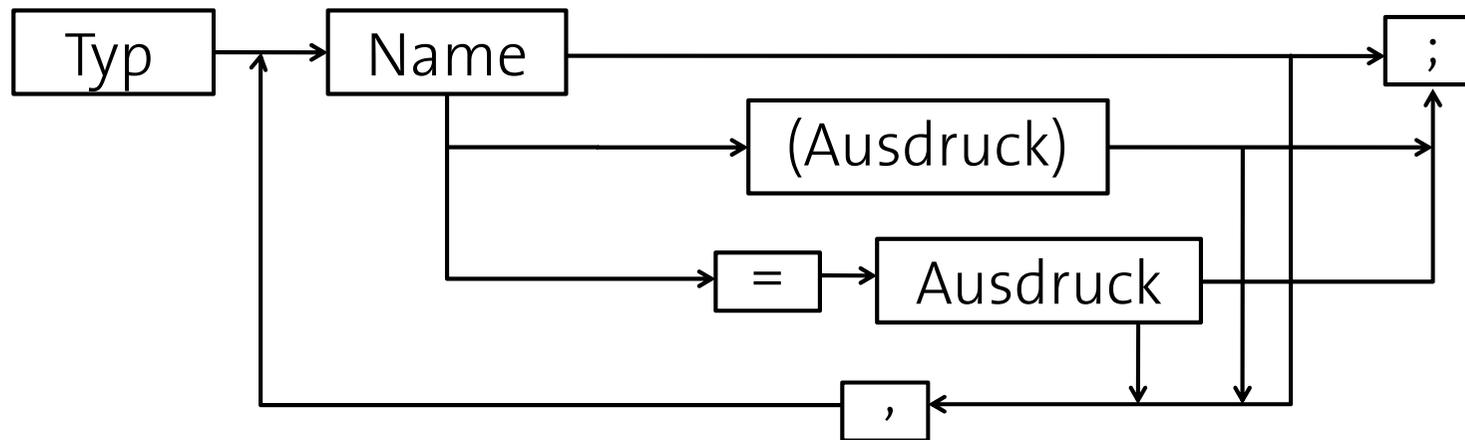
```
cout << "so " << "funktioniert " << "es" << "!" << endl;  
cout << "aber " << "es " << "funktioniert " << "auch " << "so" << "!" << endl;
```

■ Abgabe in gedruckter Form

Repetition

■ Definition von Variablen

```
int a;  
float b = 3.0f;  
double c = 3.0, d, e, f(9.0);
```



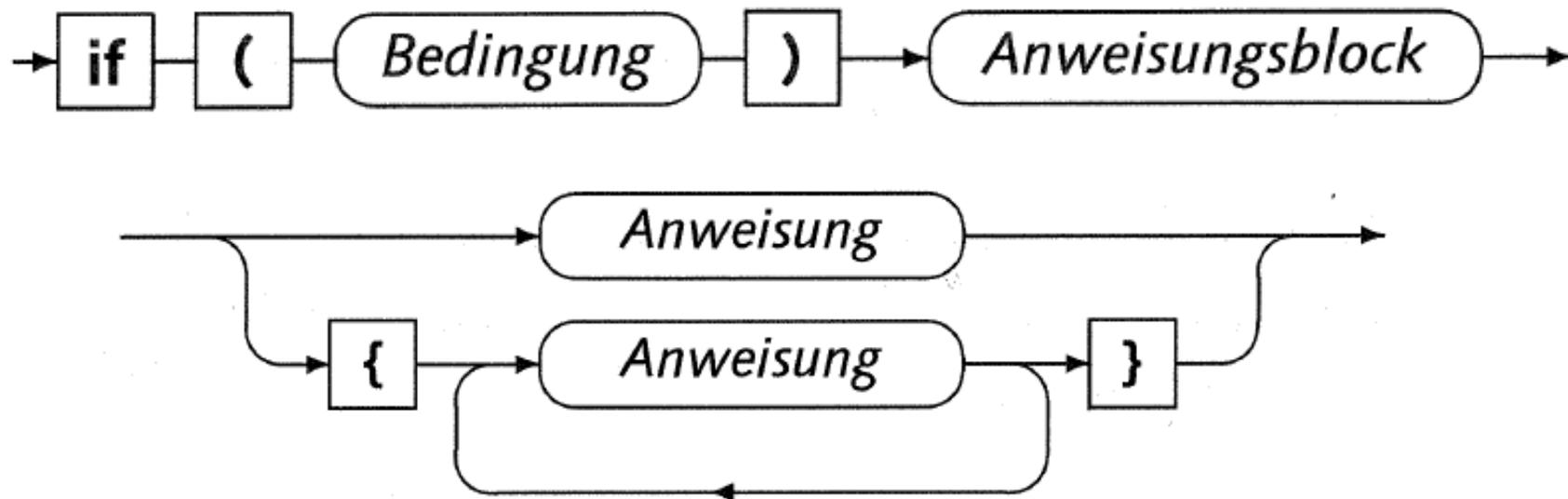
Repetition

■ Hierarchie für implizites Casting

- Long double
- Double
- Float
- Unsigned long int
- Long int
- Unsigned int
- Int

if-Anweisung

- Wird verwendet, um Anweisungen nur unter bestimmten Bedingungen auszuführen
- Syntax



if-Anweisung

- Bedingung muss Ausdruck vom Typ `bool` sein
- Anweisungsblock wird genau dann ausgeführt, wenn der Ausdruck den Wert `true` hat
- Enthält der Anweisungsblock mehrere Anweisungen, so sind geschweifte Klammern nötig

```
if (x < 0)
    x = -x;
```

```
if (x1 > x2)
{
    float tmp = x1;
    x1 = x2;
    x2 = tmp;
}
```

if-Anweisung

- Ausdruck von einem anderen Typ wird nach **bool** konvertiert
- Achtung

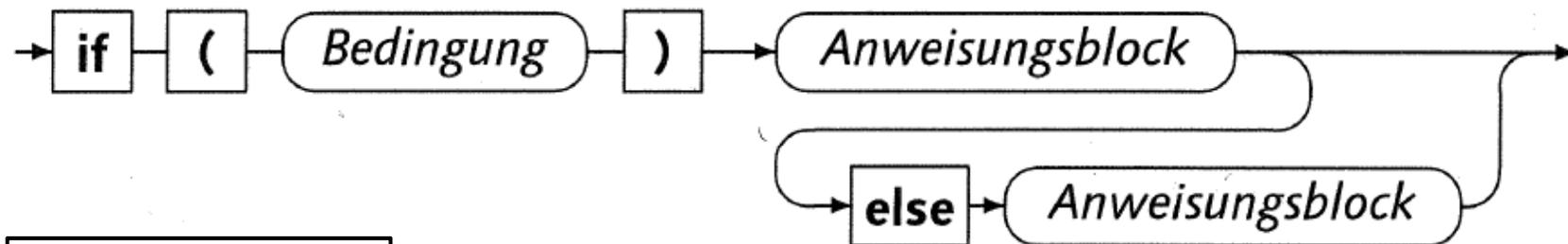
```
int a = 0;
if (a == 2) {cout << "a == 2" << endl;}
if (a = 2) {cout << "a = 2" << endl;}
cout << "a hat jetzt den Wert " << a << endl;
```

Ausgabe:

```
a = 2
a hat jetzt den Wert 2
```

if-else-Anweisung

- Bestimmt zusätzlich, welche Anweisungen ausgeführt werden, wenn die Bedingung nicht erfüllt ist
- Syntax



```
if (x2 >= x1)
    xmax = x2;
else
    xmax = x1;
```

if-else-Anweisung

■ Verschachtelung

```
bool a = true, b = false;
if (a)
    if (b)
        cout << "...";
    else
        cout << "zweites if";
```

```
bool a = false, b = true;
if (a)
{
    if (b)
        cout << "...";
}
else
    cout << "erstes if";
```

Ausgabe:

zweites if

erstes if

?: - Operator

■ Ternärer Operator

```
if (x2 >= x1)
    xmax = x2;
else
    xmax = x1;
```

```
xmax = x2 > x1 ? x2 : x1;
```

Logische Operatoren

- Logische Operatoren

```
a || b      // oder
a && b      // und
!a         // nicht
```

- Vergleiche

```
a == b
a != b
a > b
a < b
a >= b
a <= b
```

- Lazy evaluation

```
if (x > 0.0 && y/x > 1.0) {}

if (x > 0.0)
    if (y/x > 1.0)
    {
    }
```

switch

- Wird verwendet für Fallunterscheidungen

```
switch ( Ausdruck ) {  
    case Marke1 : Anweisungsfolge1 ; break ;  
    :  
    case Marken : Anweisungsfolgen ; break ;  
    default: Anweisungsfolgen+1 ; }
```

- Ausdruck muss ganze Zahl ergeben
- Marke muss konstanter Wert eines ganzzahligen Datentyps sein (z.B. char oder int)
- Beliebige Anzahl Marken

switch

- default nicht zwingend, beliebige Position
- Fehlt break, so werden alle Anweisungsfolgen bis zum nächsten break-Befehl ausgeführt

```
switch (x)
{
    case 1:
        cout << "x==1\n";
        break;
    case 2:
        cout << "x==2\n";
        break;
    default:
        cout << "x!=1 && x!=2\n";
}
```

```
switch (x)
{
    case 1:
    case 2:
    case 3:
        cout << "x==1 || x==2 || x==3\n";
        break;
    default:
        cout << "x!=1 && x!=2 && x!=3\n";
}
```

ASCII-Tabelle

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0	0x00	(sp)	32	40	0x20	@	64	100	0x40	`	96	140	0x60
(soh)	1	1	0x01	!	33	41	0x21	A	65	101	0x41	a	97	141	0x61
(stx)	2	2	0x02	"	34	42	0x22	B	66	102	0x42	b	98	142	0x62
(etx)	3	3	0x03	#	35	43	0x23	C	67	103	0x43	c	99	143	0x63
(eot)	4	4	0x04	\$	36	44	0x24	D	68	104	0x44	d	100	144	0x64
(enq)	5	5	0x05	%	37	45	0x25	E	69	105	0x45	e	101	145	0x65
(ack)	6	6	0x06	&	38	46	0x26	F	70	106	0x46	f	102	146	0x66
(bel)	7	7	0x07	'	39	47	0x27	G	71	107	0x47	g	103	147	0x67
(bs)	8	10	0x08	(40	50	0x28	H	72	110	0x48	h	104	150	0x68
(ht)	9	11	0x09)	41	51	0x29	I	73	111	0x49	i	105	151	0x69
(nl)	10	12	0x0a	*	42	52	0x2a	J	74	112	0x4a	j	106	152	0x6a
(vt)	11	13	0x0b	+	43	53	0x2b	K	75	113	0x4b	k	107	153	0x6b
(np)	12	14	0x0c	,	44	54	0x2c	L	76	114	0x4c	l	108	154	0x6c
(cr)	13	15	0x0d	-	45	55	0x2d	M	77	115	0x4d	m	109	155	0x6d
(so)	14	16	0x0e	.	46	56	0x2e	N	78	116	0x4e	n	110	156	0x6e
(si)	15	17	0x0f	/	47	57	0x2f	O	79	117	0x4f	o	111	157	0x6f
(dle)	16	20	0x10	0	48	60	0x30	P	80	120	0x50	p	112	160	0x70
(dc1)	17	21	0x11	1	49	61	0x31	Q	81	121	0x51	q	113	161	0x71
(dc2)	18	22	0x12	2	50	62	0x32	R	82	122	0x52	r	114	162	0x72
(dc3)	19	23	0x13	3	51	63	0x33	S	83	123	0x53	s	115	163	0x73
(dc4)	20	24	0x14	4	52	64	0x34	T	84	124	0x54	t	116	164	0x74
(nak)	21	25	0x15	5	53	65	0x35	U	85	125	0x55	u	117	165	0x75
(syn)	22	26	0x16	6	54	66	0x36	V	86	126	0x56	v	118	166	0x76
(etb)	23	27	0x17	7	55	67	0x37	W	87	127	0x57	w	119	167	0x77
(can)	24	30	0x18	8	56	70	0x38	X	88	130	0x58	x	120	170	0x78
(em)	25	31	0x19	9	57	71	0x39	Y	89	131	0x59	y	121	171	0x79
(sub)	26	32	0x1a	:	58	72	0x3a	Z	90	132	0x5a	z	122	172	0x7a
(esc)	27	33	0x1b	;	59	73	0x3b	[91	133	0x5b	{	123	173	0x7b
(fs)	28	34	0x1c	<	60	74	0x3c	\	92	134	0x5c		124	174	0x7c
(gs)	29	35	0x1d	=	61	75	0x3d]	93	135	0x5d	}	125	175	0x7d
(rs)	30	36	0x1e	>	62	76	0x3e	^	94	136	0x5e	~	126	176	0x7e
(us)	31	37	0x1f	?	63	77	0x3f	_	95	137	0x5f	(del)	127	177	0x7f

Operatoren

Precedence	Operator	Description	Example	Overloadable	Associativity
1	::	Scope resolution operator	Class::age = 2;	no	none
2	()	Function call	isdigit('1')	yes	left to right
	()	Member initialization	c_tor(int x, int y) : _x(x), _y(y*10){};	yes	
	[]	Array access	array[4] = 2;	yes	
	->	Member access from a pointer	ptr->age = 34;	yes	
	.	Member access from an object	obj.age = 34;	no	
	++	Post-increment	for(int i = 0; i < 10; i++) cout << i;	yes	
	--	Post-decrement	for(int i = 10; i > 0; i--) cout << i;	yes	
	const_cast	Special cast	const_cast<type_to>(type_from);	no	
	dynamic_cast	Special cast	dynamic_cast<type_to>(type_from);	no	
	static_cast	Special cast	static_cast<type_to>(type_from);	no	
	reinterpret_cast	Special cast	reinterpret_cast<type_to>(type_from);	no	
typeid	Runtime type information	cout << typeid(type).name();	no		
3	!	Logical negation	if(!done) ...	yes	right to left
	not	Alternate spelling for !			
	~	Bitwise complement	flags = ~flags;	yes	
	compl	Alternate spelling for ~			
	++	Pre-increment	for(i = 0; i < 10; ++i) cout << i;	yes	
	--	Pre-decrement	for(i = 10; i > 0; --i) cout << i;	yes	
	-	Unary minus	int i = -1;	yes	
	+	Unary plus	int i = +1;	yes	
	*	Dereference	int data = *intPtr;	yes	
	&	Address of	int *intPtr = &data;	yes	
	new	Dynamic memory allocation	long *pVar = new long;	yes	
	new []	Dynamic memory allocation of array	MyClass *ptr = new MyClass(args);	yes	
	delete	Deallocating the memory	delete pVar;	yes	
	delete []	Deallocating the memory of array	delete [] array;	yes	
	(type)	Cast to a given type	int i = (int) floatNum;	yes	
sizeof	Return size of an object or type	int size = sizeof(float);	no		
4	->*	Member pointer selector	ptr->*var = 24;	yes	left to right
	.*	Member object selector	obj.*var = 24;	no	
5	*	Multiplication	int i = 2 * 4;	yes	left to right
	/	Division	float f = 10.0 / 3.0;	yes	
	%	Modulus	int rem = 4 % 3;	yes	
6	+	Addition	int i = 2 + 3;	yes	left to right
	-	Subtraction	int i = 5 - 1;	yes	

Operatoren

Precedence	Operator	Description	Example	Overloadable	Associativity
7	<<	Bitwise shift left	int flags = 33 << 1;	yes	left to right
	>>	Bitwise shift right	int flags = 33 >> 1;	yes	
8	<	Comparison less-than	if(i < 42) ...	yes	left to right
	<=	Comparison less-than-or-equal-to	if(i <= 42) ...	yes	
	>	Comparison greater-than	if(i > 42) ...	yes	
	>=	Comparison greater-than-or-equal-to	if(i >= 42) ...	yes	
9	==	Comparison equal-to	if(i == 42) ...	yes	left to right
	eq	Alternate spelling for ==			
	!=	Comparison not-equal-to	if(i != 42) ...	yes	
	not_eq	Alternate spelling for !=			
10	&	Bitwise AND	flags = flags & 42;	yes	left to right
	bitand	Alternate spelling for &			
11	^	Bitwise exclusive OR (XOR)	flags = flags ^ 42;	yes	left to right
	xor	Alternate spelling for ^			
12		Bitwise inclusive (normal) OR	flags = flags 42;	yes	left to right
	bitor	Alternate spelling for			
13	&&	Logical AND	if(conditionA && conditionB) ...	yes	left to right
	and	Alternate spelling for &&			
14		Logical OR	if(conditionA conditionB) ...	yes	left to right
	or	Alternate spelling for			
15	? :	Ternary conditional (if-then-else)	int i = (a > b) ? a : b;	no	right to left
16	=	Assignment operator	int a = b;	yes	right to left
	+=	Increment and assign	a += 3;	yes	
	-=	Decrement and assign	b -= 4;	yes	
	*=	Multiply and assign	a *= 5;	yes	
	/=	Divide and assign	a /= 2;	yes	
	%=	Modulo and assign	a %= 3;	yes	
	&=	Bitwise AND and assign	flags &= new_flags;	yes	
	and_eq	Alternate spelling for &=			
	^=	Bitwise exclusive or (XOR) and assign	flags ^= new_flags;	yes	
	xor_eq	Alternate spelling for ^=			
	=	Bitwise normal OR and assign	flags = new_flags;	yes	
	or_eq	Alternate spelling for =			
	<<=	Bitwise shift left and assign	flags <<= 2;	yes	
	>>=	Bitwise shift right and assign	flags >>= 2;	yes	
17	throw	throw exception	throw EClass("Message");	no	
18	,	Sequential evaluation operator	for(i = 0, j = 0; i < 10; i++, j++) ...	yes	left to right

Übung 3

■ Aufgabe 1

```
'a' == 'b'  
'a' != 97  
'a' > 'B'
```

```
long int i = -5;  
unsigned int j = 2;  
bool b = (i+j < 0.0);
```

```
long int i = -5;  
double j = 2.0;  
bool b = (i+j < 0.0);
```

```
false  
false  
true
```

b hat den Wert false

b hat den Wert true

Übung 3

■ Aufgabe 2

```
!false || false  
!(15%4) && a!=b && c < d && a==e  
(277 / 100) % 10 == 2 && (277 / 10) % 10 == 7 && 277 % 10 == 7
```

```
true  
false  
true
```

■ Aufgabe 3

```
// Der Integer a ist kleiner als 100 und durch 7 teilbar  
a<100 && a%7==0  
// Der Double b ist nicht 0 und a/b ist grösser als c  
b!=0 && a/b > c  
// Das Zeichen c ist kein Grossbuchstabe  
c < 65 || c > 90
```

Übung 3

■ Aufgabe 4

```
#include <iostream>
using namespace std;
Int main() {
    double m,l,bmi;
    cout << "Berechnung des BMI (Body-Mass-Index)" << endl;
    cout << "\t Gewicht Grösse? ";
    cin >> m >> l;
    bmi = m/l^2;
    cout << "Der BMI beträgt " << bmi << "." << endl;
    if (bmi >= 18.5 && bmi <= 25.0)
        cout << "Normalgewicht" << endl;
    else
        if (bmi < 18.5)
            cout << "Untergewicht" << endl;
        else
            cout << "Uebergewicht" << endl;
    return 0;
}
```

Übung 3

- Aufgabe 5
 - Übersetzen von Zahlen in die natürliche Sprache
 - Interval [0,99]
 - Illegale Zahlen abfangen
 - Bsp: 4 → vier, 211 → keine gültige Zahl
 - Spezialfälle 0,1,11,12,16,17
 - Ziffern extrahieren und zu einem Wort kombinieren, ggf. „und“ einfügen
 - Verwende if-else- und switch-Anweisungen

Übung 3

- Aufgabe 5
 - Extrahieren von Ziffern: Verwende Integer-Division und %-Operator!
 - Zur Erinnerung: Bei der Integer-Division wird das Ergebnis abgerundet
 - $a \% b == a - b * (a / b)$