C++ - Programm

- Einführung LINUX
  - Progammierwerkzeuge: Editor, Shell Commands, Compiler, Linker
  - CIP Pool
- C++ - ein Beispielprogramm
  - Typen, Variablen, Operatoren
  - Input/Output
- C++
  - Bedingte Anweisungen und Schleifen
  - Arrays und Pointer
  - Funktionen und Klassen
- C++ - Informationen und Beispiele

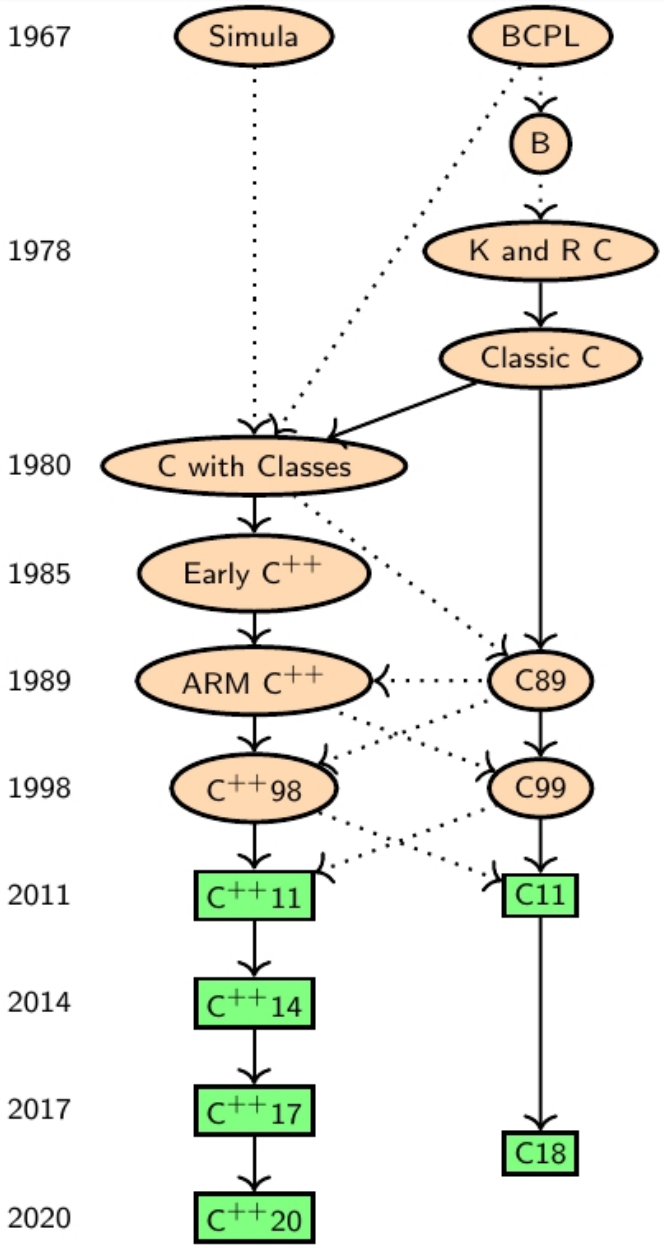
Wiki Book C++ Programmierung — Anfängerfreundlich
https://en.cppreference.com/w/ — Listet und vergleicht alle C++ Standards
https://www.tutorialspoint.com/cplusplus/index.htm
http://www.cplusplus.com/
https://stackoverflow.com/ — Fragen zur Progammierung
https://stackexchange.com → Unix & Linux — Fragen zu Unix / Linux

Einführung in die Programmierung mit C++ , Bjarne Stroustrup, ISBN 978-3-86894-005

C++ Einleitung

- Mitte der sechziger Jahr wurde bei AT&T an Mehrbenutzer-Betriebssytemen gearbeitet.
- Nach dem Rückzug von AT&T aus dem Großprojekt Multics entwickelten Thompsen und Ritchie Ende der sechziger Jahre eine erste in Assembler geschriebene Version, UNIX, auf einer PDP 7.
- Die Implementierung von UNIX in der von Ritchie entwickelten prozeduralen Programmiersprache C stellt einen Schlüsselbaustein unserer heutigen Computing Technologie dar.
- C++ wurde von Bjarne Stroustrup (Bell Laboratories) ab 1979 entwickelt, um objektorientiertes Programmieren zu erleichtern.
- C++ ist eine Erweiterung der Sprache C und abwärtskompatibel
- 1985 wurde aus „C with classes“ C++ mit dem release 2.0 im Jahr 1989
- GNU Compiler collection (gcc), implementiert die 6 ISO C++ standards der letzten Jahre (C++98, C++03, C++11, C++14, C++17, C++20, (C++23))
- Der letzte offizielle Standard wurde 2020 festgelegt, ISO/IEC 14882:2020(E) see http://www.open-std.org/jtc1/sc22/wg21/

C++ Einleitung



C inventor
Dennis M. Ritchie

C++ inventor
Bjarne Stroustrup

- Both C and C++ are born in Bell Labs
- C++ *almost* embeds C
- C and C++ are still under development
- We will discuss all C++ specs but C++20
- Each slide will be marked with first spec introducing the feature

Transparency: S.Ponce (CERN)

C++ Einleitung

- Computerprogramm
  Computer prozessieren Daten, führen Berechnungen durch, fällen Entscheidungen mit Hilfe eines Satzes von Anweisungen.
  Eine Aktion wird mit Hilfe von Schlüsselwörtern beschrieben, die durch eine Programmiersprache, z. B. C++, charakterisiert werden.
  Die Schlüsselwörter werden im Klartext in einer Datei gespeichert und von einem Übersetzungsprogramm (Compiler) in Maschinensprache umgewandelt
- Wir benutzen als C++ Compiler die öffentlich verfügbare GNU Compiler Collection (gcc), die seit version 14 den letzten C++ Standard unterstützt.
- Im Rahmen dieses Kurses wird keine Entwicklungsumgebung verwendet. Wir arbeiten auf der shell in der Linux Umgebung des CIP pools.
- Arbeiten Sie in Gruppen, dadurch werden Fehler beim Implementieren der Programme reduziert.
  Es ist ok Programme oder Programmteile zu kopieren, aber
  - Verwenden Sie nur code, den Sie auch verstehen und getestet haben
  - Urheberrechte beachten (GNU General Public License, Version)

C/C++ Schlüsselworte

- Schlüsselworte sind von der Programmiersprache verwendete Ausdrücke
- C Schlüsselworte sind in C++ verfügbar

auto break case char const continue default do double else enum extern	float for goto if inline (since C99) int long register restrict (since C99) return short	signed sizeof static struct switch typedef union unsigned void volatile while	_Alignas (since C11) _Alignof (since C11) _Atomic (since C11) _Bool (since C99) _Complex (since C99) _Generic (since C11) _Imaginary (since C99) _Noreturn (since C11) _Static_assert (since C11) _Thread_local (since C11)

C/C++ Schlüsselworte

- Schlüsselworte sind von der Programmiersprache verwendete Ausdrücke
- C Schlüsselworte sind in C++ verfügbar
- C++ Schlüsselworte

alignas (since C++11)	default(1)	register(2)
alignof (since C++11)	delete(1)	reinterpret_cast
and	do	requires (since C++20)
and_eq	double	return
asm	dynamic_cast	short
atomic_cancel (TM TS)	else	signed
atomic_commit (TM TS)	enum	sizeof(1)
atomic_noexcept (TM TS)	explicit	static
auto(1)	export(1)(3)	static_assert (since C++11)
bitand	extern(1)	static_cast
bitor	false	struct(1)
bool	float	switch
break	for	synchronized (TM TS)
case	friend	template
catch	goto	this
char	if	thread_local (since C++11)
char8_t (since C++20)	inline(1)	throw
char16_t (since C++11)	int	true
char32_t (since C++11)	long	try
class(1)	mutable(1)	typedef
compl	namespace	typeid
concept (since C++20)	new	typename
const	noexcept (since C++11)	union
consteval (since C++20)	not	unsigned
constexpr (since C++11)	not_eq	using(1)
constinit (since C++20)	nullptr (since C++11)	virtual
const_cast	operator	void
continue	or	volatile
co_await (since C++20)	or_eq	wchar_t
co_return (since C++20)	private	while
co_yield (since C++20)	protected	xor
decltype (since C++11)	public	xor_eq
	reflexpr (reflection TS)	

Ein Beispiel

myFirst.cc

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

Ein Beispiel

// oder /* */	Kommentare

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

#	Preprozessoranweisungen
include < >	File suchen und einfügen
<iostream >	Input/Output Definitionen

Ein Beispiel

// oder /* */	Kommentare

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{
```

#	Preprozessoranweisungen
include < >	File suchen und einfügen
<iostream >	Input/Output Definitionen

Präprozessor Schlüsselworte

if
elif
else
endif
	ifdef
	ifndef
	define
	undef
		include
		line
		error
		pragma
			defined
			__has_include (since C++17)
			__has_cpp_attribute (since C++20)
				export (C++20)
				import (C++20)
				module (C++20)

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

using namespace std ; Programm - namen kommen aus Standard C und C++ Bibliotheken

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

main() {C ++ Anweisungen} Haupt - programm Block, Anweisungen werden sequentiell ausgeführt

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

Definition von Variablen werden angegeben — Typ und Name

Variablen und Konstanten

- Syntax

```
Datentyp Name ;
```

- Typ Definitionen

Datentyp	Speicherbedarf
Ganze Zahlen	
`int`	16/32
`short int`	16
`long int`	32
`unsigned int`	16/32
`unsigned short int`	16
`unsigned long int`	32
`unsinged longlong int`	64

- Wertebereich ist durch Speichergrösse gegeben und betriebssystemabhängig.

short int ≤ int ≤ long int

Datentyp	Speicherbedarf
Reelle Zahlen	
`float`	32
`double`	64
`long double`	64
Buchstaben und Zeichen	
`char`	16/32
`unsigned char`	16/32
Logische Zeichen	
`bool`	

Variablen und Konstanten

Wertebereich einer durch n bit dargestellten Integer Variablen:

$-2^{n-1}, ..., 0, ..., 2^{n-1} - 1$

das Vorzeichen ist im Bit am linken Rand kodiert

Bei Operationen mit Überschreitung des Wertebereiches kommt es zu falschen Ergebnissen (undefiniertem Verhalten) ohne Fehlermeldung.

Zum Wertebereich der Variablen erhält man Zugang mit Hilfe von Funktionen, deren Definitionen wir hinzufügen müssen.

```
#include <limits>
```

https://www.cplusplus.com/reference/limits/numeric_limits/

```
…..
cout << numeric_limits <int>::min() << endl;
```

oder wie in C üblich

```
#include <climits>
```

https://www.cplusplus.com/reference/climits/

```
…..
cout << INT_MIN << endl;
```

Variablen und Konstanten

`double` und `float` werden als 64 und 32bit Gleitkommazahlen dargestellt.

Darstellung der Zahlen: $r = (-1)^V \cdot 2^{exp-127} \cdot 1.mant$

Zum Wertebereich, den Limits der Exponenten und weiteren Infos erhält man Zugang mit Hilfe von Funktionen, deren Definitionen wir hinzufügen müssen.

```
#include <limits>   https://www.cplusplus.com/reference/limits/numeric_limits
…..
cout << numeric_limits <double>::epsilon() << endl;
cout << numeric_limits <double>::round_error() << endl;
……..
```

Abweichung zwischen 1 und dem nächst größeren Wert

Maximaler Rundungsfehler

Variablen und Konstanten

Ermittlung des Speicherbedarfs von Variablen oder Typen

- Syntax

```
sizeof ( Variable ) ;
sizeof ( Datentyp ) ;
```

Mit dem Zeichen = werden Variablen Werte zugewiesen

- Syntax

```
Datentyp Name = Wert ;
```

Notwendig, sonst ist der Speicherinhalt zufällig !

Mit dem Schlüsselwort const werden konstante Variable definiert. Der Wert wird zur Compile Zeit zugewiesen und darf nicht mehr geändert werden.

- Syntax

```
const Datentyp Name = Wert ;
```

Für Zeichenketten werden weitere Definitionen gebraucht, string

- Syntax

```
#include <string> ;
const string ZeichenKette = "Strings bestehen aus char";
```

Variablen und Konstanten

Umwandlung von Typen während der Ausführung des Programms erfolgt automatisch oder mit Anweisungen.

Die automatische Konversion ist Compiler spezifisch und sollte vermieden werden !

- Syntax

```
int myInteger ;
float myFloat ;
myFloat = (float) myInteger ;

static_cast<type> expression ;
```

Wird später diskutiert

Automatische Konversion – Beispiele :

double / float zu int	Nachkommastellen werden abgeschnitten ohne zu runden
char zu int	Character werden mit einer Integer Zahl entsprechend des Ascii Zeichensatzes dargestellt. Bei einer Konversion werden diese Zeichen dargestellt

ASCII ZeichenTabelle

Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII	Dez	Hex	Okt	ASCII
0	00	000	NUL	32	20	040	SP	64	40	100	@	96	60	140	`
1	01	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	02	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	03	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	04	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	05	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	06	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	07	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	08	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	09	011	HT	41	29	051	)	73	49	111	I	105	69	151	i
10	0A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	0E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

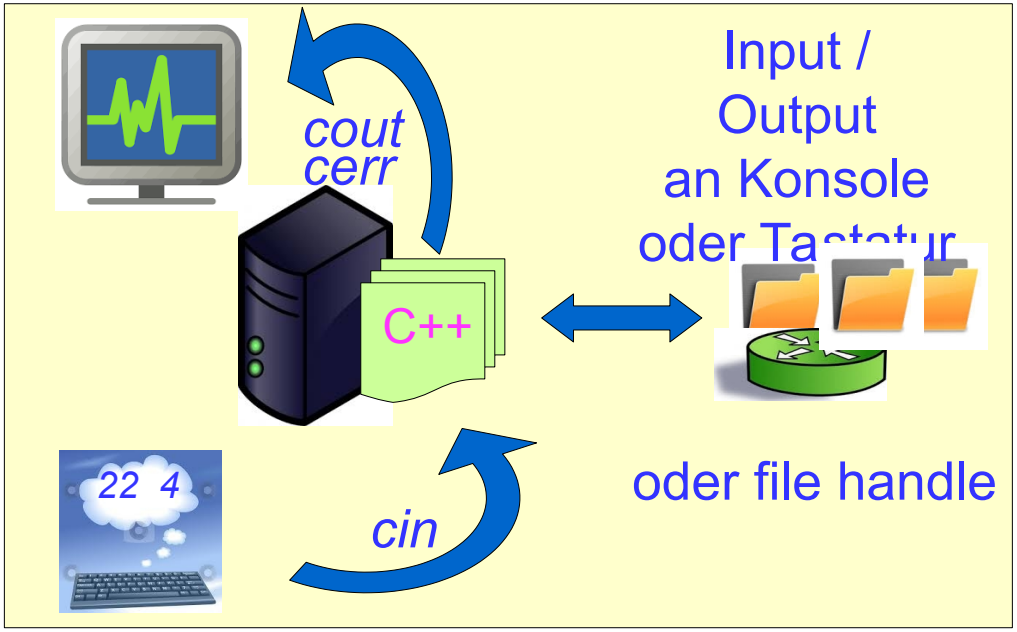
- ASCII : American Standard Code for Information Interchange
- Standard-Code für Schriftzeichen
- 7 Bit Zeichenkodierung
- 128 Zeichen, 33 nicht druckbar
- Erweiterung z.B. für asiatische Zeichen

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

cout / cin Ausgabe und Eingabe von Variablen oder strings

Input / Output



- Funktionsdefinitionen in

```
#include <iostream>;
```

- Daten werden in einen output stream geschrieben, gepuffert und an die Konsole gesendet.

- Output stream: cout

- Input stream: cin

- Output stream für Fehler: cerr

- Syntax

```
cout <<  output string << Variable << endl;
cin >> Variable1  >> Variable2 ;
```

- Beispiel:

```
const double Pi = 3.14, d = 12.0 ;
double myResult;   myResult = Pi * d ;
cout << "Umfang = " << myResult << endl;
```

Viele Optionen für Formatierungen und Steuerzeichen: #include <iomanip>

Input / Output

- Mit Hilfe von Manipulatoren lässt sich die Ausgabe formatieren. Die Manipulatoren werden in die Ausgabeströme zwischen die Umleitungsoperatoren eingefügt.

```
#include <iomanip>;
using namespace std ;
main(){
……
cout << setw(8) << i << endl;
cout << setfill('-');
cout << left << setw(8) << i << endl;
```

Beispiel: mit i = 123

```
   123
-----123
123-----
```

Details sind unter diesem link gut erklärt

http://www.willemer.de/informatik/cpp/iostream.htm

- Eine alternative und weitreichendere Formatierung lässt sich mit Hilfe von C print Befehlen erreichen.

https://de.wikibooks.org/wiki/C-Programmierung:_Einfache_Ein-_und_Ausgabe

```
printf(format_string,ausgabe_liste);
sprintf(zeichenkette,format_string,ausgabe_liste);
fprintf(dateizeiger,format_string,ausgabe_liste);
```

Steuerzeichen (Escape-Sequenzen)

- Nicht druckbare Zeichen des gewählten Zeichensatzes.
- Beginn mit einem umgekehrten Schrägstrich (Backslash).
- Datentyp char.

… in C++

Escape-Sequenz	Erläuterung
\b	Backspace. Einen Schritt zurück.
\f	Form Feed. Seitenvorschub
\n	New Line. Zeilenvorschub
\r	Carriage Return. Wagenrücklauf
\t	Horizontaler Tabulator
\v	Vertikaler Tabulator
\“	Anführungszeichen
\'	Apostroph
\\	Backslash. Umgekehrter Schrägstrich
\0	Ende-Zeichen eines C-Strings.

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

sum = a + b Zuweisung von Ausdrücken

return 0 Rückgabewert des Programms

Lebensdauer und Anwendungsbereich der Variablen

- Scope (Anwendungsbereich) von Variablen
  In welchem Bereich bleiben definierte Variablen erhalten?
  - In einem Code Block { ……. }
  - In einer Funktion, Klassen und einem namespace
  - Im globalen Bereich, solange sie vor den Klammern definiert werden

```
{ int a ;
    { int b ;
    } // end of b scope
} // end of a scope
```

Variablen sollten initialisiert werden, wenn sie gebraucht werden

- namespace erlaubt code zu segmentieren, um Kollisionen von Variablen-namen zu vermeiden → Variable können ausserhalb verwendet werden

```
int a;
namespace A{ int a; } // no overlap
namespace B{ int a;
   namespace inner { int a; } } // no overlap
A::a = 3;
B::a = 4;
```

Zugriff auf Variablen im namespace

Operatoren

Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren
  Berechnung von Werten
- Bit Operatoren
  Manipulation einzelner Bits
- Vergleichsoperatoren
  Überprüfung von Aussagen
- Logische Operatoren
  Verknüpfung von Aussagen

Operatoren	Berechnungen
*	Multiplikation
/	Division
%	Division mit Rest
+	Summe
-	Differenz

Operatoren	Berechnungen
*=	Multiplikation
/=	Division
%=	Division mit Rest
+=	Summe
-=	Differenz

Berechnung wird mit den linken Operatoren durchgeführt und verändern dann den links stehenden Wert.

```
P += 7; →  P=P+7
P += 7+2; → P=P+7+2
```

N++	N um 1 erhöhen
m- -	m um 1 verkleinern

- Benutzung von Klammern wie in der Mathematik

Operatoren

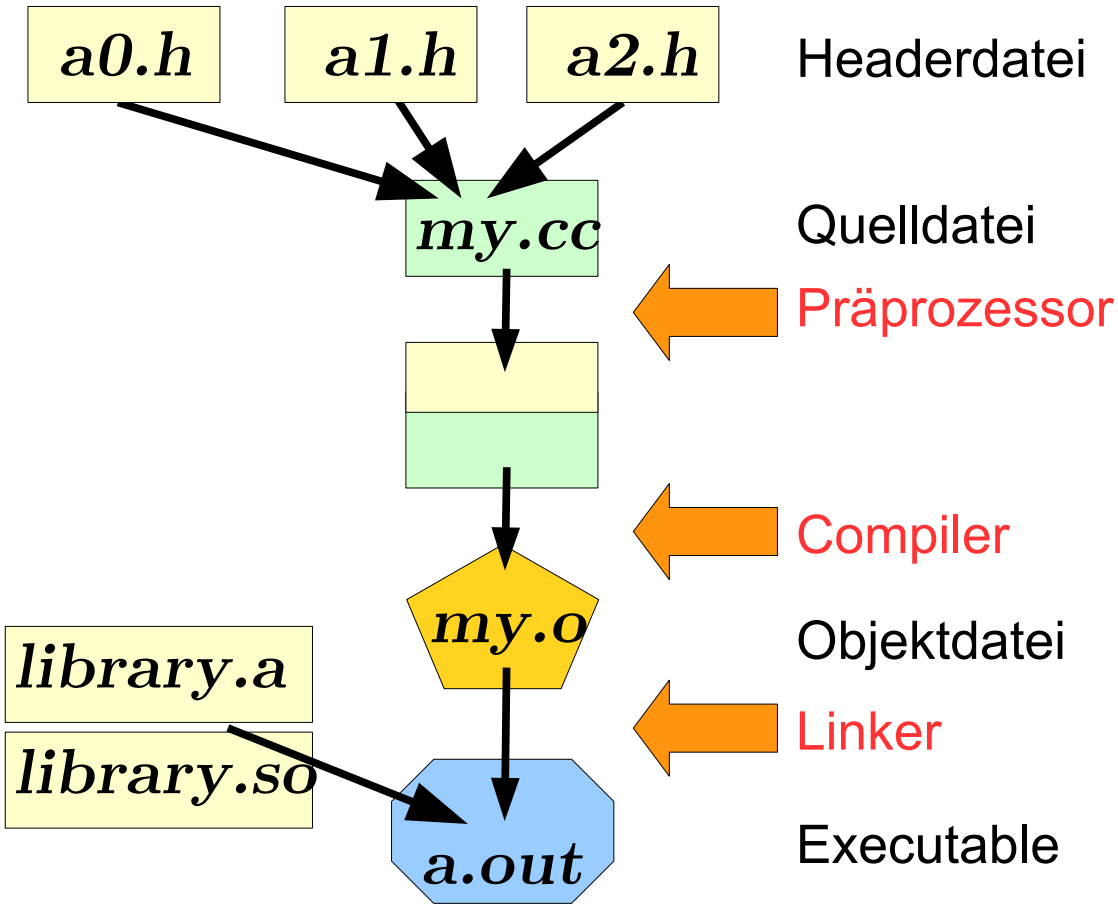
Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren
  Berechnung von Werten
- Bit Operatoren
  Manipulation einzelner Bits
- Vergleichsoperatoren
  Überprüfung von Aussagen
- Logische Operatoren
  Verknüpfung von Aussagen

Operatoren	Bit Operationen
~	nicht
&	und
|	oder
^	entweder oder
>>	nach rechts verschieben
<<	nach links verschieben

Ein ausführbares C++ Programm

Die Erzeugung eines ausführbaren C++ Programmes erfolgt in 3 Schritten:



Daten zur Struktur und Definitionen des Programms

Enthält Anweisungen zum Ablauf und Verhalten des Programms

Aus dem C++ Quellcode übersetzter Maschinencode

Alle vom Linker zusammen-gebundenen Objektdateien bilden das ausführbare Programm

Einleitung - GNU Compiler Collection

- Übernimmt generell folgende Aufgaben: preprocessing, compilation, assembly and linking
- Beinhaltet Compiler für die Programmiersprachen C, C++, Objective-C, Fortran, Ada, Go und Java
- Freie Software unter GNU General Public License
- Unterstützung vieler Hardware und Betriebssystemplattformen (die am häufigsten portierte Compiler Suite, > 60 Plattformen)
- Standard Compiler der linuxartigen Betriebssysteme
- Unterstützt auch mitgelieferte Softwaresammlungen (libraries, libstdc++,..)
- Dokumentation https://gcc.gnu.org/onlinedocs/
- Verhalten wird durch Optionen in Form von Flags und File-Namen gesteuert
  - Optionen haben „multi letter names“, daher nicht wie in Unix Befehlen: -dv $\neq$ -d -v
  - File-Name Endungen bestimmen den aktivierten compiler

C++
Files: .C, .cc, .cpp, .cxx, .c++, .h,
Extension: .hh, .hpp, .hxx, .h++

  - Details auf der shell mit dem Befehl `man gcc` oder ausührlicher `info gcc`
    (falls die Hilfen installiert sind, fehlen in CIP pool, https://www.gnu.org/software/gcc/)

GNU Compiler Collection gcc

- Sprachoptionen:
  - c c-header cpp-output
  - c++ c++-header c++-cpp-output → g++
  - objective-c objective-c-header objective-c-cpp-output
  - objective-c++ objective-c++-header objective-c++-cpp-output
  - assembler assembler-with-cpp
  - ada
  - f77 f77-cpp-input f95 f95-cpp-input
  - go
  - java

- Verwendung von g++ setzt Standard Library Pfade für den Linker

GNU Compiler Collection gcc

- Übersetzen eines C++ Programmes

```
g++ myFirst.cc        →    a.out
```

- Nützliche Optionen von g++
  - Programm Namen vergeben

```
g++ myFirst.cc -o MyProgramm        →    MyProgramm
```

  - Compiler Warnungen einschalten

```
g++ -Wall -Wextra myFirst.cc -o MyProgramm
```

  - Nur Compiler verwenden ohne Linker

```
g++ -c myFirst.cc -o mFirst.o        →    myFirst.o
```

  - Abschalten der nicht standard-conformen Compiler Optionen von g++

```
g++ -ansi myFirst.cc
```

  - Warnungen bei nicht standard-conformen Erweiterungen von g++

```
g++ -pedantic myFirst.cc
```

  - Automatische Optimierungen des Programms

```
g++ -Ox myFirst.cc     x [1,2,3]
```

  - Einschalten von speziellen Versionen

```
g++ -std=c++xx  mit xx=11,14,17,20
```

Ein Be

```
// Add 2 Integer typed in by the user via

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" <

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

Arbeitsanweisungen:

- loggen Sie sich mit Ihrem userid ein.

- erzeugen Sie ein Arbeits – Directory.

- starten Sie einen Editor, z.B. emacs

- schreiben Sie unserem Beispiel entsprechend ein C++ Programm und speichern sie es als myFirst.cc

- Übersetzen Sie Ihr File und erzeugen Sie ein ausführbares Programm mit g++ myFirst.cc -o myFirst

- Probieren Sie es mit ./myFirst

- Bauen Sie C++ Syntaxfehler in myFirst.cc ein, übersetzen Sie das File und suchen Sie nach Hinweisen auf den Fehler.

Vergleiche gcc und g++:

- Probieren Sie g++ myFirst.cc -o myFirst und gcc myFirst.cc -o myFirst

- Fügen Sie Pfade in gcc hinzu um die Fehlermeldungen zu entfernen
  gcc myFirst.cc -L/usr/lib/gcc/x86_64-linux-gnu/12/ -lstdc++ -o myFirst
  -L setze Pfadnamen
  -l setze Bibliotheksnamen (dabei wird das beginnende lib weggelassen)

  Pfad ist installationsabhängig!

- Die verwendete Sprache wird durch die Endungen festgelegt
  mv myFirst.cc myFirst.kk
  gcc myFirst.kk -L/usr/lib/gcc/x86_64-linux-gnu/12/ -lstdc++ -o myFirst
  → Compiler Fehler

  Hinzufügen von -x c++
  gcc -x c++ myFirst.kk -L/usr/lib/gcc/x86_64-linux-gnu/12/ -lstdc++
  -o myFirst
  aktiviert den c++ Teil von gcc
  → kein Compiler Fehler

- Weitere Optionen mit der Online Hilfe auf der shell oder im Web
  man gcc https://gcc.gnu.org/onlinedocs/gcc-12.4.0/gcc/