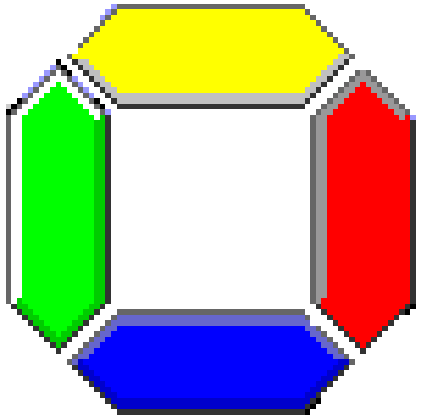
Vorlesung aus C++

Univ.-Prof. Dr. Rony G. Flatscher

Email: rony@wi-inf.uni-essen.de

WWW: http://nestroy.wi-inf.uni-essen.de/

Sprechstunde: Dienstag, $10^{00} - 11^{00}$ Uhr



Universität GH Essen

1. LV-Einheit C++

1. Zur Person des Vortragenden
2. Stellung der LV, Veranstaltungsform
3. Literatur, Online-Information
4. Abgrenzung C++ und andere Programmiersprachen
5. Vom Problem zum Programm
6. Elemente der Programmiersprache C++
7. Erste C++-Programme
8. Vom Quellencode zum ausführbaren Programm

Veranstaltungsform

- **Zweistündige Vorlesung:**
  Mo $12^{00} - 14^{00}$, Präsentation von neuem Stoff
- **Zweistündige Übung** (Fredj Dridi):
  - Unix Rechnerraum R09 R02 H02
    Mi. $8^{00} - 10^{00}$ Uhr
    Linux Systeme mit Gnu C++
  - Vergabe von Benutzerkennungen über Anmeldeformular
  - Unix Einführung (Hochschulrechenzentrum): UNIX
    Praktikum I und II, zweitägig
- **Übungszeit im Unix-Schulungsraum**:
  Öffnungszeiten $9^{00} - 17^{00}$ Uhr
  ausgenommen: Vorlesungszeiten

Primärliteratur

- Peter A. Darnell and Phillip E. Margolis: *"C: A Software Engineering Approach"*, Springer Verlag, 1996, ISBN: 0387946756

  ANSI C, didaktisch sehr gut, guter Programmierstil, Beispiele

- Ira Pohl: *"Object Oriented Programming Using C++"*, Addison-Wesley Object Technology Series, December 1996, ISBN: 0201895501

  Schwergewicht und schrittweise Einführung in die OO-Programmierung in C++, C-Grundkenntnisse sind vorteilhaft.

Häüfig eingesetzten C++–Einführungsbuch

- Stanley B. Lippman: *"C++ Primer"*, 2nd Edition, Addison-Wesley 1991, ISBN: 0201548488

  C++ für C-Programmierer von einem der Entwickler des AT&T C++-Compilers.

  Schwerpunkte: Function Templates and Virtual Functions, Kapitel über OOD.

  Lippman gibt es in Deutsch und Englisch wobei die deutsche Version durch die "Verdeutschung" vieler Fachausdrücke eher zum abgewöhnen ist.

Literatur – Objektorierntierte Programmiermethodik

- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *"Design Patterns, Elements of Reusable Object-Oriented Software"*
  In diesem Buch werden immer wiederkehrende Objektklassen und Abstraktionen über Zusammenhänge identifiziert und klassifiziert. Im ersten Teil wird die Grundidee der Design-Patterns und die verwendetete Notation vorgestellt, im zweiten Teil folgt ein Referenzhandbuch von klassischen Design-Patterns. Das Buch ist vorzüglich dazu geeignet, für die Klassenbildung in konkreten Anwendungsfällen Denkanstöße zu bieten.

Detail- und Ergänzungsliteratur

- Bjarne Stroustrup: *"The C++ Programming Language"*
  Der "Klassiker" vom Erfinder der Sprache, enthält die Sprachdefinition "Reference Manual", z.T. ausführlicher als Lippman.
- Brian W. Kernigham and Dennis M. Ritchie: *"The C Programming Language"*
  Der Klassiker für C, didaktischer Wert umstritten.
- Ellis and Stroustrup: *"Annotated C++ Reference Manual"* (ARM)
  Schwer zu lesen, aber *alle* Details.

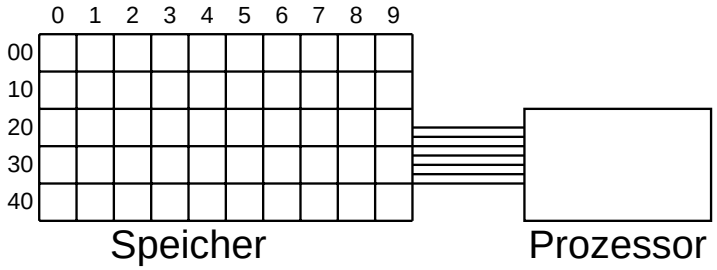
Zusatzinformationen

Über WWW (World Wide Web):

- Wirtschaftsinformatik und Software-Technik
- ...Lehre
- ...C++
- http://nestroy.wi-inf.uni-essen.de/Lv/cpp.html/

Entwicklung von prozeduralen Programmiersprachen

John v. Neumann: Programmierbare Rechenmaschine — 1940




```
X = 1 + 2

LDA A      | 0: 1 7
ADA B      | 2: 2 8
SPA X      | 4: 3 9
STP        | 6: 0
DATA A 1   | 7: 1
DATA B 2   | 8: 2
DATA X 0   | 9: 0
```

John Backus (IBM): Fortran — 1957
FORmular TRANslator

IBM: PL/1 — 1964
General Purpose PL

Dahl: Simula — 1967
Simulation

N. Wirth: Pascal — 1971
Strukturierte Programmierung

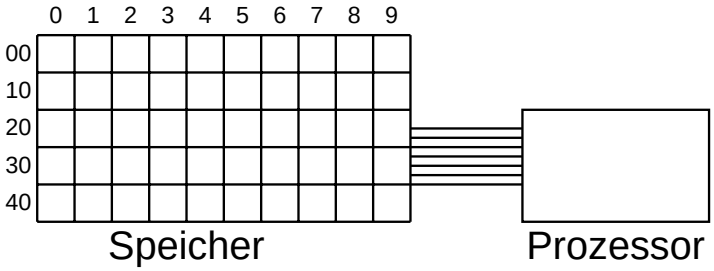
D. Ritchie (AT&T): C — 1974
Systemprogrammierung

N. Wirth: Modula-2 — 1980
Verbesserte Datenkaspelung

B. Stroustroup (AT&T): C++ — 1984
OO Erweiterung von C

Entwicklung von symbolischen Programmiersprachen

John v. Neumann: Programmierbare Rechenmaschine — 1940




```
X = 1 + 2

LDA A     | 0: 1 7
ADA B     | 2: 2 8
SPA X     | 4: 3 9
STP       | 6: 0
DATA A 1  | 7: 1
DATA B 2  | 8: 2
DATA X 0  | 9: 0
```

John Backus (IBM): Fortran — 1957
FORmular TRANslator

J. McCarthy (MIT): LISP — 1960
LISt Processing Language,
Funktionale Programmierung

K. Iverson (IBM): APL
Array-orientierte Programmierung

A. Colmerauer: Prolog — 1972
Logikorientierte Programmierung

A.Goldberg (Xerox): Smalltalk80 — 1980
Objektorientierte Programmierung

J. Jaffar et.al. (IBM): CLP
Logikorientierte Programmierung
mit Nebenbedingungen

Was ist C++?

- C++ ist ein "besseres C"
- C++ ist eine objekt-orientierte Programmiersprache (OOPL)

  OOPLs für große und komplexe Programmsysteme (100.000 LOC). Weiterentwicklung der strukt. Programmierung, Code Reuse.
- C++ ist eine Multi-Paradigm PL

  Prozedurale Programmierung: Fortran, C, Pascal

  Abstrakte Datentypen: Ada, Modula 2

  OOP: Simula, Smalltalk

Vorteile von C++

Wachstum: C++ ist die dominierende OOPL am Markt, C++ wird *eingesetzt*.

Encapsulation: Verstecken der Implementationsdetails, Schnittstelle. Modultechnik vs. Abstrakte Datentypen.

Mehrfache Objekte: durch Encapsulation in "Structs" ("Records")

Operator Overloading: Syntaktische Verfeinerung, Lesbarkeit

Vererbung: Code Reuse, Flexibilität

Häufige Fragen:

- Läuft C++ auf dem Rechner X unter dem Betriebssystem Y?
- Wer verwendet C++?
- Ist C++ rückwärtskompatibel zu ANSI-C?
- Gibt es einen C++-Standard?
  X3J16 - ANSI, WG21 - ISO (ARM)
- Wie lange braucht man, um C++ zu lernen?
  ca. 6 Monate (9 Monate für Programmiersprachenanfänger)
- Gibt es einen Übersetzter von C++ nach C?

Vom Problem zum Programm

1. Problembeschreibung und -abgrenzung
2. Entwurf der Problemlösung
   Objektentwurf, Funktionsentwurf, Ereignisentwurf
3. Kodierungsphase
   - Erstellen des Quellprogramms mit einem Editor
   - Compilation des/der Programmteile
   - Laden und Binden
4. Testphase, üblicherweise gefolgt von Iterationen in 3 oder 2

Beschreiben von Modellen

"Man nehme 2 dag Schmalz und erhitze es in einer Pfanne. Inzwischen versprudelt man 3 Eier mit einer Brise Salz. Wenn das Fett heiß ist, gieße man die Eier in die Pfanne. Mit einer Gabel umrühren, bis die Eier stocken, sofort heiß servieren..."

- Kochrezepte
- Gebrauchsanweisungen
- Algorithmen
- Programme

Sprache ist ein Werkzeug zur Beschreibung von Modellen

- Natürliche Sprache

  Deutsch, Englisch, Latein,...

- Formale Sprache

  APL, Basic, C++, Cobol, Fortran, Lisp, Pascal, Prolog,...

Programmieren ist das *detaillierte Beschreiben* von Modellen.

Natürliche versus formale Sprache

- Natürliche Sprache ist vielfach syntaktisch u./o. semantisch mehrdeutig
- Natürliche Sprache ist fehlertolerant
- Vorgegebener Wortumfang in natürlicher Sprache sehr hoch: 80000 Worte Langenscheidt vs. ca. 50 C++
- Programmiersprache ist primitiv genug, um in Maschinencode übersetzt werden zu können

Übersetzung von Sprachen

Sprache A $\overset{\text{Übersetzung}}{\longrightarrow}$ Sprache B

Sprache A $\overset{\text{Syntax}}{\longrightarrow}$ Struktur $\overset{\text{Semantik}}{\longrightarrow}$ Sprache B

Compiler, Interpreter

Definition einer Sprache

Wie lernt man eine Sprache?

- Die Menge der Symbole, die in Worten verwendet werden kann.
- Die Menge der gültigen Sätze, die aus Worten gebildet werden können:

  “Grammatikregeln”. Konjugationen, Deklinationen, Satzstruktur.
- Die Bedeutung der gültigen Sätze:

  Vokabeln, einfache-zusammengesetzte Vokabeln.

Vordefinierte Worte in C++

Schlüsselworte, "reservierte" Worte:

asm	auto	break	case	catch
char	class	const	continue	default
delete	do	double	else	enum
extern	float	for	friend	goto
if	inline	int	long	new
operator	private	protected	public	register
return	short	signed	sizeof	static
struct	switch	template	this	throw
try	typedef	union	unsigned	virtual
void	volatile	while		

Zeichen mit vordefinierter Bedeutung:

```
. , : ; ' " { } [ ] ( ) + = ! & * / % |
```

Vom Programmierer wählbare Worte in C++

Bezeichner:

Beginnen mit Buchstaben oder "_", gefolgt von Buchstaben, Ziffern und "_".

gültig: `v1, Vector, semaphore_pointer, __X`

ungültig: `5j, $name, int, name%#*@`

- C++ ist "case-sensitive"!
- Die Bedeutung von Operatoren kann in C++ vom Programmierer verändert werden.

Sätze in C++

```
j += 1;

cout << "brave new world\n";      /* this is a test */

class Vector {
public:
  Vector(int, char *);
  Vector(int, vec_type, char *);
  ~Vector();
  vec_type sum();                 // summation operator
  vec_type &operator[](int);      // index operator
  friend ostream& operator<<(ostream &, Vector &);
private:
  unsigned int _length;
};
```

Wie sieht C++ aus?

Programm: p0.cc

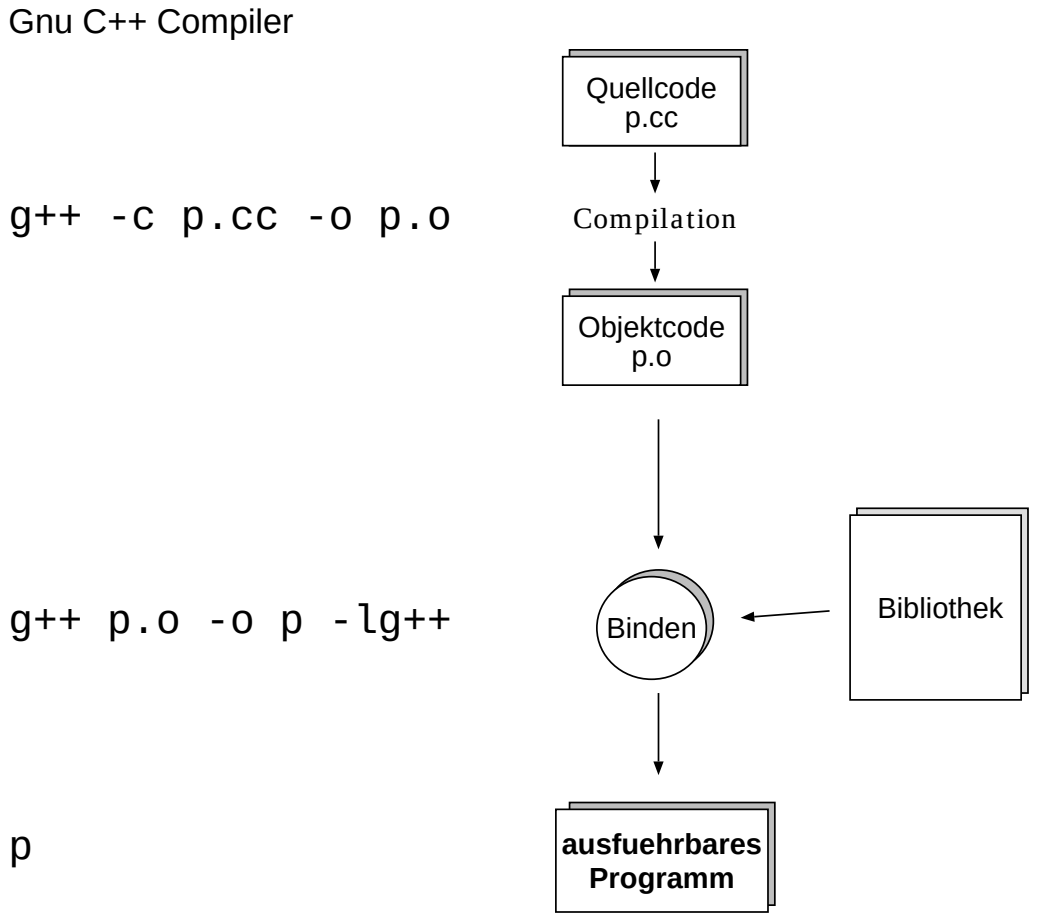
```
#include <iostream.h>

void
main(void) {
  cout << "hello world" << endl;
}
```

Wie sieht C++ aus?

Programm: p1.cc

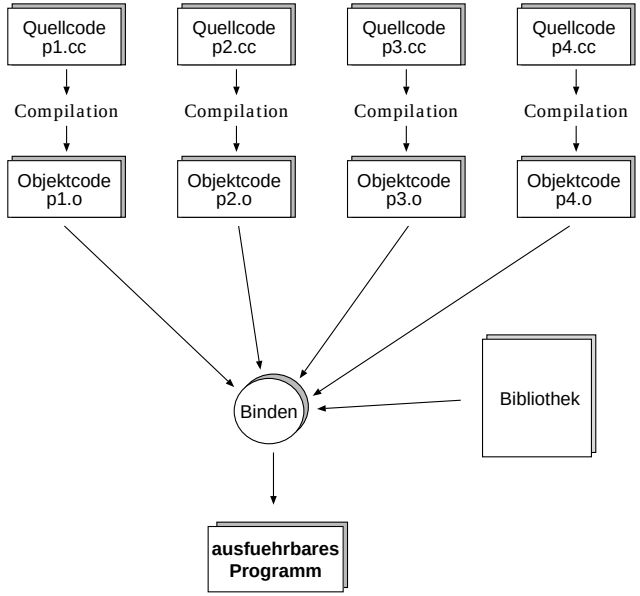
```
#include <iostream.h>

int
main(int argc, char **argv) {
  int a=1;
  int b=2;
  cout << a << " + " << b << " = " << a+b;
  return 0;
}
```

Vom Quellencode zum ausführbaren Programm



Alternative: `make p`

Vom Quellencode zum ausführbaren Programm



Alternative: `make p`

Makefile:

```
OBJS = p1.o p2.o p3.o p4.o
p: $(OBJS)
   g++ $(OBJS) -o p
```

Kompilieren eines C++-Programms

C++ Programme werden in Dateien abgespeichert, die das Suffix `.cc` besitzen.

`g++ <filename>` erzeugt `a.out`

`g++ <filename.cc> -o <name of executable>`

Syntaxfehler vs. semantische Fehler, Fehlermeldungen.