

Programmiersprachen

Programmentwicklung

IDE (Integrated Development Environment)

https://de.wikipedia.org/wiki/Liste_von_integrierten_Entwicklungsumgebungen



Code::Blocks (z.Z. Version 20.03 - Juni 2020)

<http://www.codeblocks.org/>

<https://de.wikipedia.org/wiki/Code::Blocks>

- freie, quelloffene Entwicklungsumgebung für C, C++, D und Fortran
- basiert auf wxWidgets
- läuft unter GNU/Linux, Windows und den meisten Unix-Derivaten
- Buildsystem ohne Makefiles
- unterstützt Syntax-Highlighting, Code-Faltung, Klassen-Browser
- Importfunktion für Visual Studio Projekte
- verschiedene Plugins

Orwell Dev-C++ (früher Dev-C++) (z.Z. Version 5.11 - Juni 2020)

<https://sourceforge.net/projects/orwelldevcpp/>

https://de.wikipedia.org/wiki/Orwell_Dev-C%2B%2B

- freie, quelloffene Entwicklungsumgebung für C, C++
- ursprünglich von Bloodshed Software
- nutzt MinGW als Compiler (eine Windows-Portierung der GNU Compiler Collection)

The screenshot displays the Code::Blocks IDE interface. The main editor window shows a C program named `main.c` with the following code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     printf("Hello world!\n");
7     return 0;
8 }
9
```

An "About..." dialog box is open in the foreground, featuring the Code::Blocks logo (four colored cubes) and the text:

Code::Blocks

The open source, cross-platform IDE

20.03

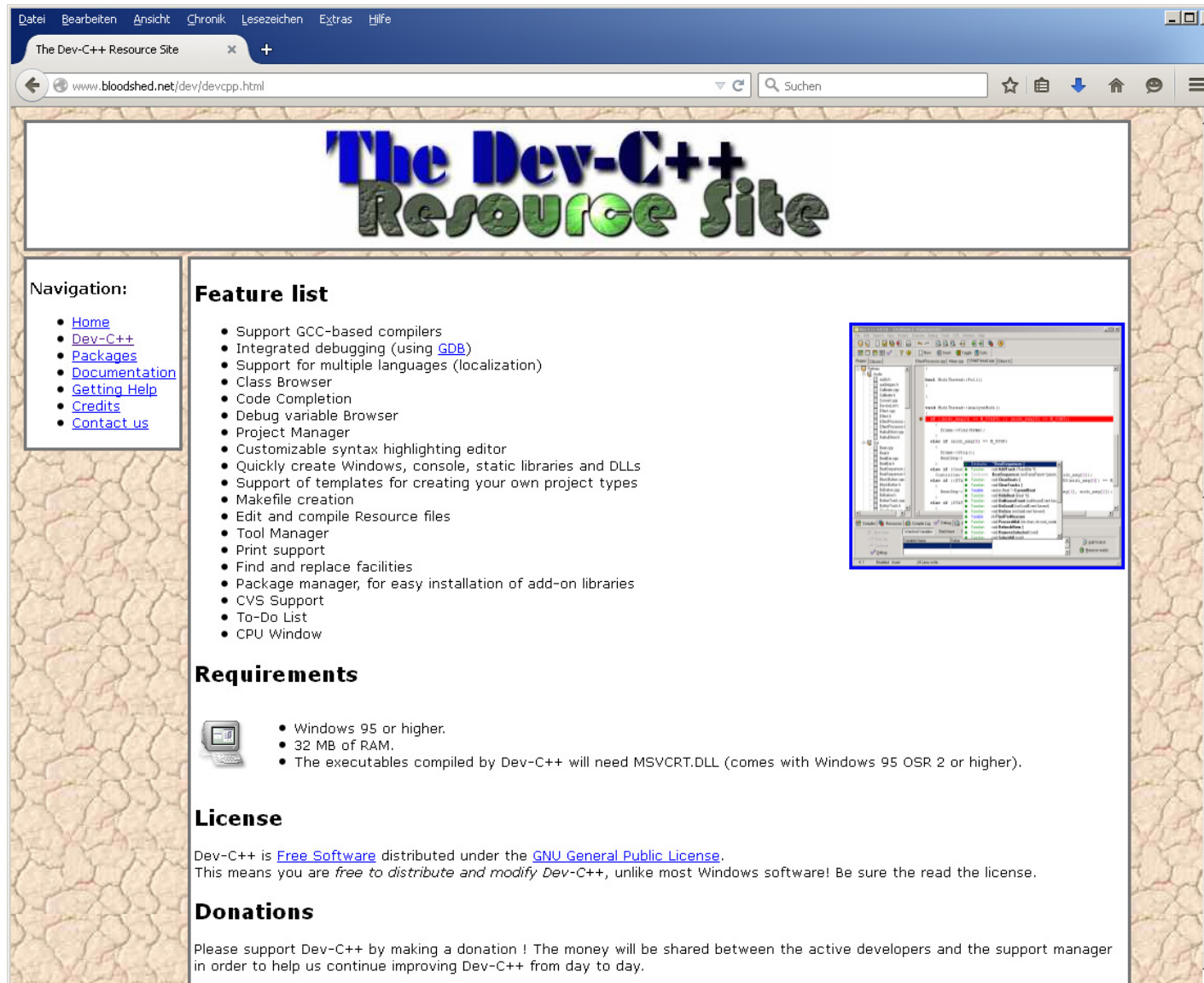
Build: Mar 14 2020, 18:11:34 - wx3.1.3 - gcc 8.1.0 (Windows, unicode) - 64 bit

The dialog box includes tabs for "Description", "Information", "Thanks to...", and "License". The "Description" tab is active, showing the following text:

Welcome to Code::Blocks 20.03!
Code::Blocks is a full-featured IDE (Integrated Development Environment) aiming to make the individual developer (and the development team) work in a nice programming environment offering everything he/she would ever need from a program of that kind.
Its pluggable architecture allows you, the developer, to add any kind of functionality to the core program, through the use of plugins...

A "Close" button is located at the bottom right of the dialog box.

The IDE's status bar at the bottom shows the current file path, language (C/C++), window title (WINDOWS-1252), and cursor position (Line 1, Col 1, Pos 0).



The screenshot shows a web browser window with the address bar displaying "www.bloodshed.net/dev/devcpp.html". The page title is "The Dev-C++ Resource Site". The main content area has a textured background and contains the following sections:

The Dev-C++ Resource Site


Navigation:

- [Home](#)
- [Dev-C++](#)
- [Packages](#)
- [Documentation](#)
- [Getting Help](#)
- [Credits](#)
- [Contact us](#)

Feature list

- Support GCC-based compilers
- Integrated debugging (using [GDB](#))
- Support for multiple languages (localization)
- Class Browser
- Code Completion
- Debug variable Browser
- Project Manager
- Customizable syntax highlighting editor
- Quickly create Windows, console, static libraries and DLLs
- Support of templates for creating your own project types
- Makefile creation
- Edit and compile Resource files
- Tool Manager
- Print support
- Find and replace facilities
- Package manager, for easy installation of add-on libraries
- CVS Support
- To-Do List
- CPU Window

Requirements

-  Windows 95 or higher.
- 32 MB of RAM.
- The executables compiled by Dev-C++ will need MSVCRT.DLL (comes with Windows 95 OSR 2 or higher).

License

Dev-C++ is [Free Software](#) distributed under the [GNU General Public License](#). This means you are *free to distribute and modify Dev-C++*, unlike most Windows software! Be sure the read the license.

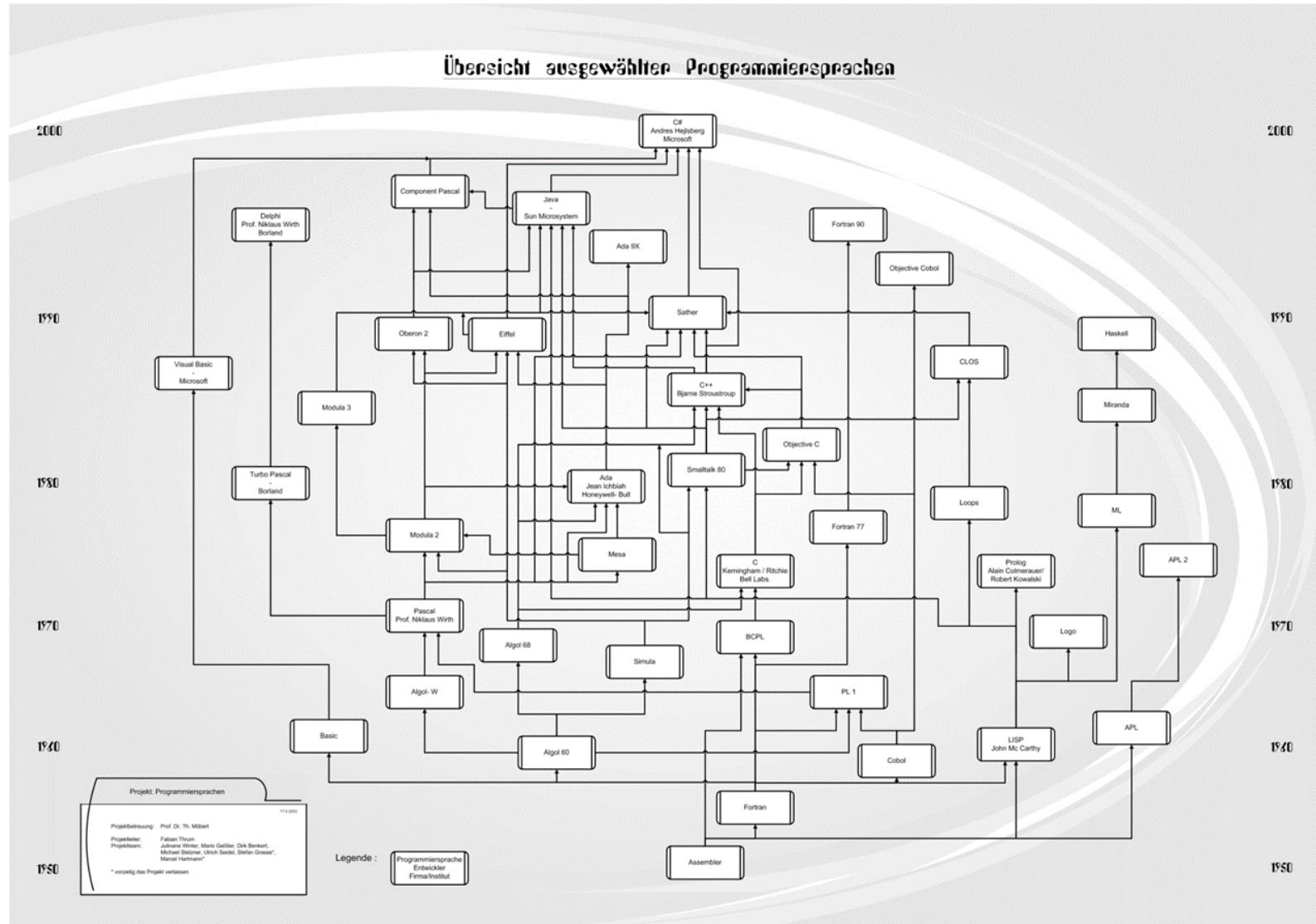
Donations

Please support Dev-C++ by making a donation ! The money will be shared between the active developers and the support manager in order to help us continue improving Dev-C++ from day to day.

An inset image shows the Dev-C++ IDE interface, highlighting the Project Manager window.

Systematik der Programmiersprachen

- 1) imperative Sprachen (z. B. Basic o. ä.)
 - 2) prozedurale Sprachen (z. B. ANSI-C oder Pascal)
 - 3) objektorientierte Sprache (z.B. Smalltalk, CommonLisp, EIFFEL)
 - 4) funktionale Sprachen (z. B. Haskell, ML, CommonLisp, Scheme, Dylan)
 - 5) deklarativ-logische Sprachen (z. B. Prolog)
 - 6) symbol-verarbeitende Sprachen (z. B. Scheme, CommonLisp)
 - 7) Assembler-Sprachen (z. B. MIPS, x86, m68k)
 - 8) Shell-Sprachen (z.B. bash, csh, ksh, scsh...)
 - 9) Markup-Sprachen (SGML, HTML...)
 - 10) RDBMS-Query Sprachen (gibts eigentlich zur Zeit nur noch SQL)
 - 11) Graphik/Seitenbeschreibungssprachen (PS=Postscript)
 - 12) 3D-Szenenbeschreibungssprachen (VRML, OpenGL, PovRay, DXF)
 - 13) Multimedia-Scripting (SMIL, LINGO, OpenScript)
 - 14) Scriptsprachen (VBScript, JScript, Javascript, REXX, SAP ABAP)
 - 15) Fachsprachen ...
 - 16) Telekommunikationsskripte (Telix, Telemate)
 - 17) Esoterische Programmiersprachen (Brainfuck, Whitespace, LOLCode, Piet, Ook!, HQ9+, ZOMBIE,BIT, Haifu, Whenever)
- CLI, Scripting, MS
PowerShell
netsh



Programmiersprachengenerationen

1. Generation (Maschinensprachen)

- prozessorabhängig (z.B. x86)
- im Bootblock/ROM-BIOS
- kurze effiziente Programme
- verwenden absolute Adressen

2. Generation (Assemblersprachen)

- seit Anfang der 50er Jahre
- strukturäquivalent zu den Befehlen der Maschinensprache
- Zugriff auf die Prozessorbefehle (bzw. Peripherie)
- mnemotechnische Abkürzungen als Bezeichner für Variablen und Befehle
- Mechanismen zur Vereinfachung de Codierungsaufwandes
- Makros
- Übersetzung in Maschinenprogramme durch (Makro-)Assembler

3. Generation (prozedurale / problemorientierte Programmiersprachen)

- rechnerunabhängige Codierung prozeduraler Algorithmen
- Abstraktion von der Hard- und Softwarestruktur
- algorithmischer Aspekt hinsichtlich der Von-Neumann-Rechnerarchitektur

4. Generation (4GL / seit Ende der 60er Jahre)

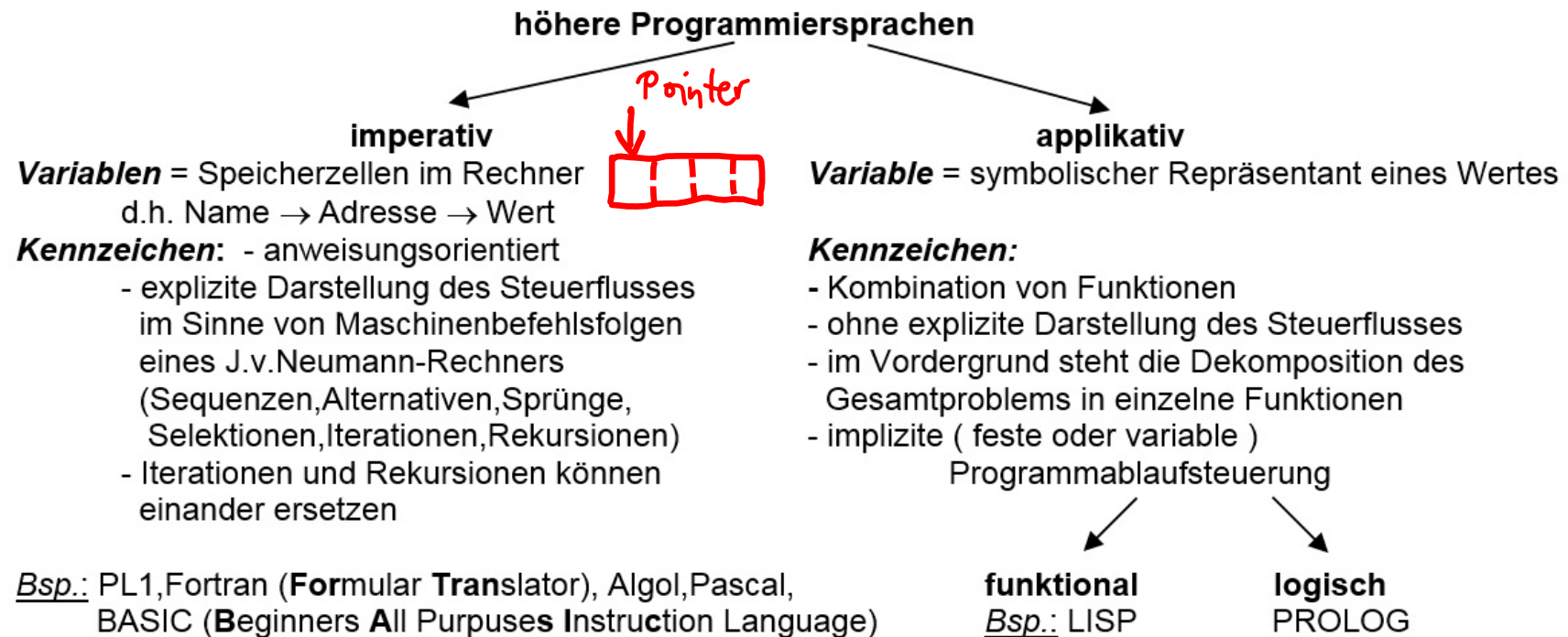
- Definition und Operation auf Datenbank (SQL/NATURAL)
- Werkzeuge zur Gestaltung von Oberflächen (Masken)

5. Generation (deklarative Programmiersprachen)

- Problembeschreibung
- System
- logische Sprachen (PROLOG)
- funktionale Sprachen (ML, MIRANDA)

Kombination sinnvoll: logisch-funktionale Sprachen

Programmiersprachen - Konzepte



objektorientierte Programmierung

Programmierung von Objekten mit speziellen Eigenschaften und Verhalten.

Die Objekte werden als Instanz einer Klasse (=gleichartige Objekte) erzeugt und existieren dann eine gewisse Zeitdauer. Die einmal für eine Klasse beschriebenen Eigenschaften und Verhaltensweisen (Methoden) sind jeder Instanz einer Klasse zueigen und vererben sich in der aufgebauten Objekthierarchie von einer Superklasse auf die Subklassen. Die Programmabarbeitung entspricht einem Nachrichtenaustausch zwischen den Objekten und einer entsprechenden Reaktion und einer evtl. Zustandsänderung des Objektes auf die empfangene 'message'. Datenabstraktionen führen zur Möglichkeit der Verwendung verallgemeinerter Typen und Operatoren sowie dynamischer Typprüfung. (RTTI = runtime type information)

objektorientierte Programmierung

Kennzeichen: Objektorientierung / ereignisgesteuerte Programme

Abstraktion: Merkmalsreduktion; fundamentale menschliche Fähigkeit, welche es uns ermöglicht, die Komplexität der Realität zu erfassen und zu beschreiben

Objekt Modell: beschreibt die statische Struktur (Objekte, Beziehungen / objects, relationships)

Dynamisches Modell: beschreibt die Steuer- (control) Struktur (Ereignisse, Zustände)

Funktions-Modell: beschreibt die Struktur der Werte (computational structure) –

(Werte, Funktionen, Einschränkungen/values, functions, constraints)

Class-Diagramm: Schema, Pattern (Muster), Template (Struktur-Vorlage) zur Beschreibung endlich vieler Instanzen einer Klasse

Instanz-Diagramm: beschreibt eine Menge von Objekten, die in Beziehung zueinander stehen

Bsp: Klasse= Person

Instanzen der Klasse (Objekte): Olaf Müller, Peter Otto

Attribute: Name, Alter, Gewicht (Attribute haben nur Werte, aber keine existenzielle Identität)

Klassenbox

Person
Name: string
Alter: integer
Gewicht: real

Objektboxen = *Instanzen*

Olaf Müller
24
75,8

Peter Otto
36
65,2

objektorientierte Programmierung

Datenkapselung:

Zusammenschweißen von Programmteilen und Daten in einem Objekt

Polymorphie (griech. viele Formen, gestalten):

Einführung verschiedener Prozeduren (Methoden) unter gleichem Namen und gleicher Parameterliste

Vererbung:

Attribute und Methoden (=Prozeduren zur Manipulation von Objekten einer Klasse) können auf Objekte der von der Oberklasse abgeleiteten Klasse vererbt (übernommen) werden.

Vorteile:

- einfache Erweiterbarkeit objektorientierter Entwürfe ohne bestehende Programmteile ändern zu müssen durch Hinzufügen des zusätzlichen Codes
- hochmoderne und angenehme Nutzerinterfaces
- späte Bindung = Linken erst während der Laufzeit (erfordert dynamische Link-Bibliothek)

Prolog

Prolog-Programm = Menge von Klauseln = Menge von Listen (eine Klausel wird als Liste notiert)

Liste = n-Tupel von Elementen (Objekten, Termen)

rekursive Def einer Liste:

() ist die leere Liste

(x|Z) ist eine Liste, falls Z eine Liste und x ein Element ist

Bsp:

((el Paul Horst))	((w Ida))	((alt Paul 1850 1901))
((el Paul Heinz))	((w Ilse))	((alt Ilse 1856 1918))
((el Paul Olga))	((w Susi))	((alt Horst 1876 1940))
((el Ilse Horst))	((w Anna))	((alt Heinz 1878 1941))
((el Ilse Heinz))	((w Emma))	((altOlga 1886 1964))

Zielklausel:

```

?((m Max) → liefert ?, weil ((m Max )) nicht als Fakt vorhanden
? ((el Paul z)) → ohne Anzeige
? ((el Paul z) (PP z)) → Horst
? ((el Paul z) (PP z) FAIL) → liefert Horst
                                Heinz
                                Olga

WHICH (z (el Paul z)) → liefert Horst
                                Heinz
                                Olga
  
```

Programmiersprachen

"Hello-World" in Postscript

```

%!PS-Adobe-2.0 EPSF-2.0
%%DocumentFonts: Courier-Bold Helvetica-Bold
%%BoundingBox: 5 -5 540 120
%%Title: test.eps
%%EndComments

0 200 moveto
(Hallo Welt) show % schwarz
gsave
  0 100 moveto
  1 0 0 setrgbcolor
  (hallo Welt) show % rot
grestore

showpage

```

HelloWorld in MSIL:

Microsoft Intermediate Language

```

.assembly MyAssembly {}
.class MyApp {
.method static void Main()
{ entrypoint
  ldstr "Hello World!"
  call void
  System.Console::WriteLine
  (class System.Object)
  ret
} }

```

Hello World Programm in C#

```

using System;
public class MyHelloWorld{
public MyHelloWorld() {
    console.WriteLine('Hello World!');
}
}

```

HelloWorld in Turbo Pascal:

```

program test;
uses Overlay,CRT,DOS,TOOLS,ReadChr;
const b:byte=13;
var i:integer;r:real;

procedure ausschrift;
begin
    writeln('Hello world!');
end;

begin (* Beginn des Hauptprogrammes *)
    ausschrift;
    i:=5;r:=12,3456;
    writeln('i=', i:2,' r=',r:6:2,chr(b));
end.

```

Syntaxdiagramme

Beschreibung von Sprachen mittels graphischer Elemente (Knoten und gerichtete Bögen zwischen diesen Knoten). Jeder Weg in einem solchen Syntaxdiagramm beschreibt ein syntaktisch zulässiges Sprachkonstrukt der zu definierenden Sprache.

Sprachelemente:

Allgemeine Symbole: Programmablaufrihtung, Kommentar

Sequenz

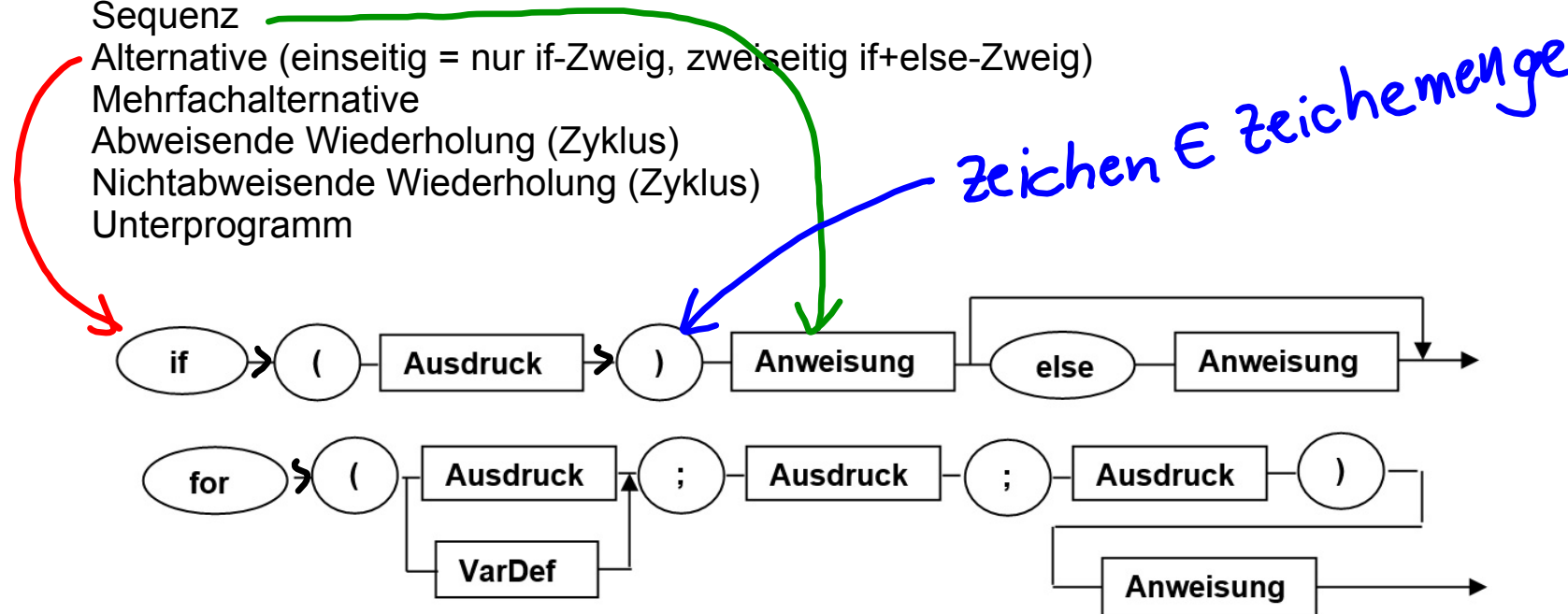
Alternative (einseitig = nur if-Zweig, zweiseitig if+else-Zweig)

Mehrfachalternative

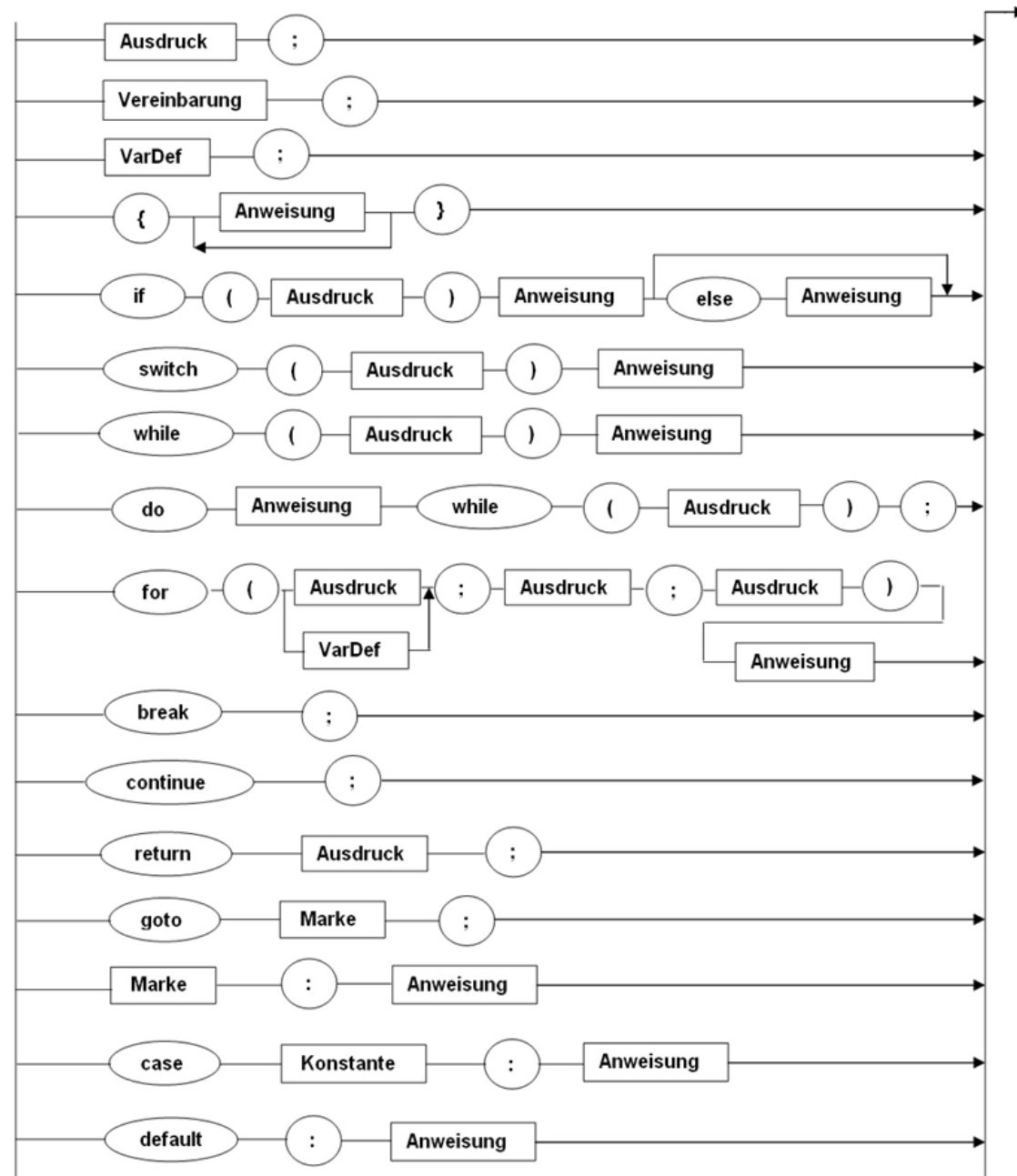
Abweisende Wiederholung (Zyklus)

Nichtabweisende Wiederholung (Zyklus)

Unterprogramm



Syntaxdiagramm der Programmier- sprache C



Formale Sprachen / Metasprachen

Def: künstliche Sprache = Formale Sprache (Lexik, Syntax) + Interpretation/Bedeutung (Semantik)

Def: Grammatik ist eine Beschreibung einer Sprache (Syntaxregeln), d.h. eine Metasprache, also eine Sprache zur Beschreibung einer Sprache.

Def: Ein Alphabet ist eine beliebige, endliche, nichtleere Menge.
Die Elemente eines Alphabetes heißen Zeichen oder Symbole.

Def: Ein Wort (über dem Alphabet A) ist eine Verkettung (Konkatenation/Hintereinanderschreibung) von endlich vielen Zeichen aus ein und demselben Alphabet A. → vereinbarte Schreibweise: "abcdef"

Def: BNF (Backus Naur Form, vorher Backus-Normal-Form) ist eine formale Beschreibung einer Grammatik definiert durch Jim Backus/Pete Naur zur Definition Algol60

Weiterentwicklungen:

- EBNF (Extending BNF) durch Nikolaus Wirth für Pascal
- ISO/IEC 14977 (EBNF)
- BS 6154
- ABNF (Augmented BNF nach RFC 2234)
- XML, YACC/BISON-Grammers, XBNF

Die erweiterte Backus-Naur-Notation (EBNF)

Diese Metasprache dient der Beschreibung Syntax einer Programmiersprache, Kommandosprache, Fachsprache oder sonstigen formalen Sprachen.

Die erweiterte Backus-Naur-Notation besteht aus den folgenden Sprachelementen:

a) Notations- oder Metavariablen = Begriff, der nicht selbst Element der zu definierenden Sprache ist
(wird in Kleinbuchstaben kursiv/rot notiert oder in Klammern <>)

Bsp.: (alle folgenden Beispiele beziehen sich auf die Beschreibung von MSDOS-Betriebssystemkommandos bzw. cmd-CLI-Kommandos)

datei1 pfad dateiname laufwerk ..oder .. <datei1> <pfad> <dateiname> <laufwerk>

b) Notationskonstante = Zeichen oder Zeichenfolge (Schlüsselwörter) der zu definierenden Sprache

Bsp.: DIR TYPE TREE FORMAT ERASE CD MD RD

c) Definitionssymbol "::=" zur Definition von Metavariablen

Syntax: zu definierende Metavariablen ::= Ausdruck in Backus-Form

Bsp.: kommando1 ::= DIR dateiname

d) Alternativsymbol "|" zur Trennung alternativer Notationskonstanten/Notationsvariablen

(als 'oder' zu lesen)

Bsp.: internes_Kommando ::= DIR|TYPE|REN|DEL|TIME|DATE|CD|MD|RD

e) Möglichkeitssymbol "[]" ; die in den Klammern stehenden Metavariablen bzw. Notationskonstanten können an dieser Stelle stehen, müssen es aber nicht

Bsp.: DIR [/w /p] [dateispezifikation]

f) Auswahlsymbol "{ }" ; eine der vom Auswahlsymbol umschlossenen Alternativen muss stehen

Bsp.: COPY { datei | datei1 datei2 | datei pfadangabe }

g) Wiederholungssymbol "*" ; die vor dem Wiederholungssymbol

stehende Metavariablen/Konstante bzw. Metaausdruck kann mehrmals hintereinander stehen

Bsp.: vollständige_pfadangabe ::= {\knotenname}*
also z.B.: pfad ::= \AAA\BBB\CCC\DDD

h) Listensymbol "'°°°'" ; analog wie das Wiederholungssymbol, aber vor jeder Wiederholung muss ein

Komma (bzw. ein speziell definierter Separator) stehen. Unter einer Liste versteht man eine durch Komma getrennte Aneinanderreihung von Sprachelementen

Definition eines deutschen Satzes:

<Satz> = <Subjekt> <Prädikat>[<Objekt>] .
 <Subjekt> = <Eigename> | <Artikel><Substantiv>.
 <Prädikat> = <Verb> .
 <Objekt> = <Artikel><Substantiv>

Formeldefinition für die Grundrechenarten:

Start: <formel> ::= <ausdruck>
 <ausdruck> ::= <term> { + <term> | - <term> }
 <term> ::= <faktor> { * <faktor> | / <faktor> }
 <faktor> ::= <zahl> | (<ausdruck>)
 <zahl> ::= [+|-] <ziffer> {<ziffer>}* [. {<ziffer>}*]
 <ziffer> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Definition einer "Paradiessprache":

<Eigename> = "Adam" | "Eva" .
 <Verb> = "ißt" | "beißt" | "liebt" .
 <Artikel> = "der" | "die" .
 <Substantiv> = "Apfel" | "Schlange" .

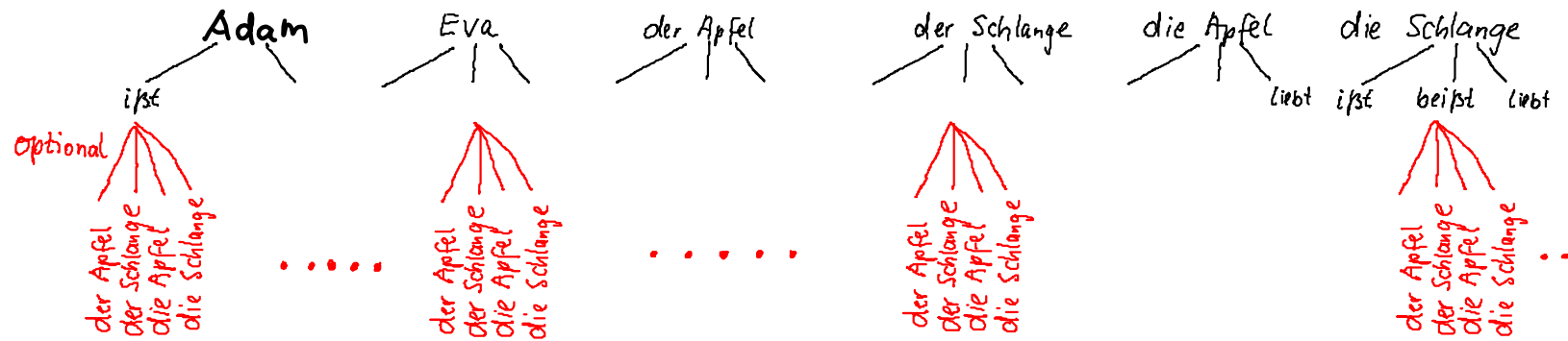
Bsp: Startsymbol: $\langle \text{paradiessprache} \rangle ::= (\langle \text{satz} \rangle)^*$

- $\langle \text{satz} \rangle ::= \langle \text{subjekt} \rangle \langle \text{prädikat} \rangle [\langle \text{objekt} \rangle].$
- $\langle \text{subjekt} \rangle ::= \langle \text{eigennamen} \rangle \mid \langle \text{artikel} \rangle \langle \text{substantiv} \rangle$
- $\langle \text{prädikat} \rangle ::= \langle \text{verb} \rangle$
- $\langle \text{objekt} \rangle ::= \langle \text{artikel} \rangle \langle \text{substantiv} \rangle$
- $\langle \text{eigennamen} \rangle ::= \text{"Adam"} \mid \text{"Eva"}$
- $\langle \text{verb} \rangle ::= \text{"ißt"} \mid \text{"beißt"} \mid \text{"liebt"}$
- $\langle \text{artikel} \rangle ::= \text{"der"} \mid \text{"die"}$
- $\langle \text{substantiv} \rangle ::= \text{"Apfel"} \mid \text{"Schlange"}$

(1. Regel, rule, Produktion)

zulässige Sätze (auch manchmal Wortegenannt) der definierten Sprache

Bsp: Adam liebt Eva. ~~der Apfel~~ ~~beißt~~ die Schlange.
Eva ißt.
⋮



Problem: falls ein Metasymbol selbst Bestandteil der zu definierenden Sprache ist, so muß die Meta-Bedeutung für dieses Zeichen aufgehoben werden (in der reduzierten Notation ohne " "). Dabei benutzt man ein Quotingsymbol
Bsp: $\langle \text{sondertext} \rangle ::= \text{Mengen werden in } [\] \text{ eingeschlossen und } 2 \setminus 1.$ z.B. " \ "

1 < 2

ABNF – Augmented Backus Naur Form – RFC 2234 → RFC 4234 → RFC 5234

Crocker & Overell
RFC 5234

Standards Track
ABNF

ietf.org
[Page 10]
January 2008

```

rulelist      :: = 1*( rule / (*c-wsp c-nl) )

rule          = rulename defined-as elements c-nl
              ; continues if next line starts
              ; with white space

rulename      = ALPHA *(ALPHA / DIGIT / "-")

defined-as    = *c-wsp ("=" / "=/") *c-wsp
              ; basic rules definition and
              ; incremental alternatives

elements      = alternation *c-wsp

c-wsp         = WSP / (c-nl WSP)

c-nl          = comment / CRLF
              ; comment or newline

comment       = ";" *(WSP / VCHAR) CRLF

alternation   = concatenation
              *( *c-wsp "/" *c-wsp concatenation)

```

```

concatenation = repetition *(1*c-wsp repetition)
repetition   = [repeat] element
repeat       = 1*DIGIT / (*DIGIT "*" *DIGIT)
element      = rulename / group / option /
              char-val / num-val / prose-val
group        = "(" *c-wsp alternation *c-wsp ")"
option       = "[" *c-wsp alternation *c-wsp "]"
char-val     = DQUOTE *(%x20-21 / %x23-7E) DQUOTE
              ; quoted string of SP and VCHAR
              ; without DQUOTE
num-val      = "%" (bin-val / dec-val / hex-val)
bin-val      = "b" 1*BIT
              [ 1*("." 1*BIT) / ("-" 1*BIT) ]
              ; series of concatenated bit values
              ; or single ONEOF range
dec-val      = "d" 1*DIGIT
              [ 1*("." 1*DIGIT) / ("-" 1*DIGIT) ]
hex-val      = "x" 1*HEXDIG
              [ 1*("." 1*HEXDIG) / ("-" 1*HEXDIG) ]
prose-val    = "<" *(%x20-3D / %x3F-7E) ">"
              ; bracketed string of SP and VCHAR
              ; without angles
              ; prose description, to be used as
              ; last resort

```

≙ "|" Alternativsymbol

Syntaxbeschreibung mittels modifizierter BNF (RFC 822 / E-Mail-Format)

- a) RULE1 / RULE2: ALTERNATIVES (Alternative)
- b) (RULE1 RULE2): LOCAL ALTERNATIVES (lokale Alternative)
- c) *RULE: REPETITION (Wiederholung)
 - <l>*<m>element
 - bedeutet: mindestens 1 element, höchstens m elemente
 - 0 und unendlich sind Standard
 - Bsp: 1*DIGIT bedeutet mindestens eine Ziffer, *3DIGIT bedeutet: höchstens 3 Digits
- d) [RULE]: OPTIONAL (Möglichkeitssymbol)
- e) NRULE: SPECIFIC REPETITION (Wiederholung)
 - "<n>(element)" is equivalent to "<n>*<n>(element)"
 - bedeutet: genau n-faches Auftreten des Elementes
- f) #RULE: LISTS
 - <l>#<m>element
 - bedeutet: mindestens 1 element, höchstens m elemente durch Komma getrennt
- g) ; COMMENTS (Kommentar bis zum Zeilenende)
- h) "\" (Quotingsymbol)
 - Aufheben der Sonderbedeutung eines Zeichens
- i) "zeichenkette"

```

j) LEXICAL TOKENS
CHAR          = <any ASCII character>           ; ( Octal, Decimal.)
              = <any ASCII character>           ; ( 0-177, 0.-127.)
ALPHA         = <any ASCII alphabetic character>
              = <any ASCII alphabetic character> ; (101-132, 65.- 90.)
              = <any ASCII alphabetic character> ; (141-172, 97.-122.)
DIGIT         = <any ASCII decimal digit>        ; ( 60- 71, 48.- 57.)
CTL           = <any ASCII control
              character and DEL>                 ; ( 0- 37, 0.- 31.)
              = <any ASCII control
              character and DEL>                 ; ( 177, 127.)
CR            = <ASCII CR, carriage return>      ; ( 15, 13.)
LF            = <ASCII LF, linefeed>            ; ( 12, 10.)
SPACE        = <ASCII SP, space>                ; ( 40, 32.)
HTAB         = <ASCII HT, horizontal-tab>       ; ( 11, 9.)
<">         = <ASCII quote mark>               ; ( 42, 34.)
CRLF         = CR LF
LWSP-char    = SPACE / HTAB                     ; semantics = SPACE
linear-white-space = 1*([CRLF] LWSP-char)      ; semantics = SPACE
              = 1*([CRLF] LWSP-char)            ; CRLF => folding
specials     = "(" / ")" / "<" / ">" / "@"
              / "," / ";" / ":" / "\" / "<"
              / "." / "[" / "]"                 ; Must be in quoted-
              / "." / "[" / "]"                 ; string, to use
              / "." / "[" / "]"                 ; within a word.
delimiters   = specials / linear-white-space / comment
text         = <any CHAR, including bare
              CR & bare LF, but NOT
              including CRLF>                   ; => atoms, specials,
              = <any CHAR, including bare
              CR & bare LF, but NOT
              including CRLF>                   ; comments and
              = <any CHAR, including bare
              CR & bare LF, but NOT
              including CRLF>                   ; quoted-strings are
              = <any CHAR, including bare
              CR & bare LF, but NOT
              including CRLF>                   ; NOT recognized.
atom         = 1*<any CHAR except specials, SPACE and CTLs>
quoted-string = <"> *(qtext/quoted-pair) <">; Regular qtext or
              = <"> *(qtext/quoted-pair) <">; quoted chars.
              = <"> *(qtext/quoted-pair) <">; => may be folded
qtext        = <any CHAR excepting <">,
              "\" & CR, and including
              linear-white-space>
domain-literal = "[" *(dtext / quoted-pair) "]"
dtext        = <any CHAR excluding "[",
              "]" , "\" & CR, & including
              linear-white-space>
              = <any CHAR excluding "[",
              "]" , "\" & CR, & including
              linear-white-space>
comment      = "(" *(ctext / quoted-pair / comment) ")"
ctext        = <any CHAR excluding "(" ,
              ")" , "\" & CR, & including
              linear-white-space>
              = <any CHAR excluding "(" ,
              ")" , "\" & CR, & including
              linear-white-space>
quoted-pair  = "\" CHAR                         ; may quote any char
phrase       = 1*word                            ; Sequence of words
word         = atom / quoted-string

```

Programmentwicklung

Programme

Lebenszyklus von Programmen:

Programmentwurf: PAP, Datenflussdiagramme

Programmierung: mittels Programmiersprachen und Fachsprachen

Programmtestung: Testen des Programms auf Korrektheit

Programmnutzung: Tools, Programmentwicklungsumgebungen,
Anwenderprogramme, integrierte Systeme

Programmpflege: Weiterentwicklung, Fehlerbeseitigung, Neuanpassungen
(Bug-Reports, Updates, Code-Patches, Servicepacks)
(Patch =Flicken, Fleck, Stück)

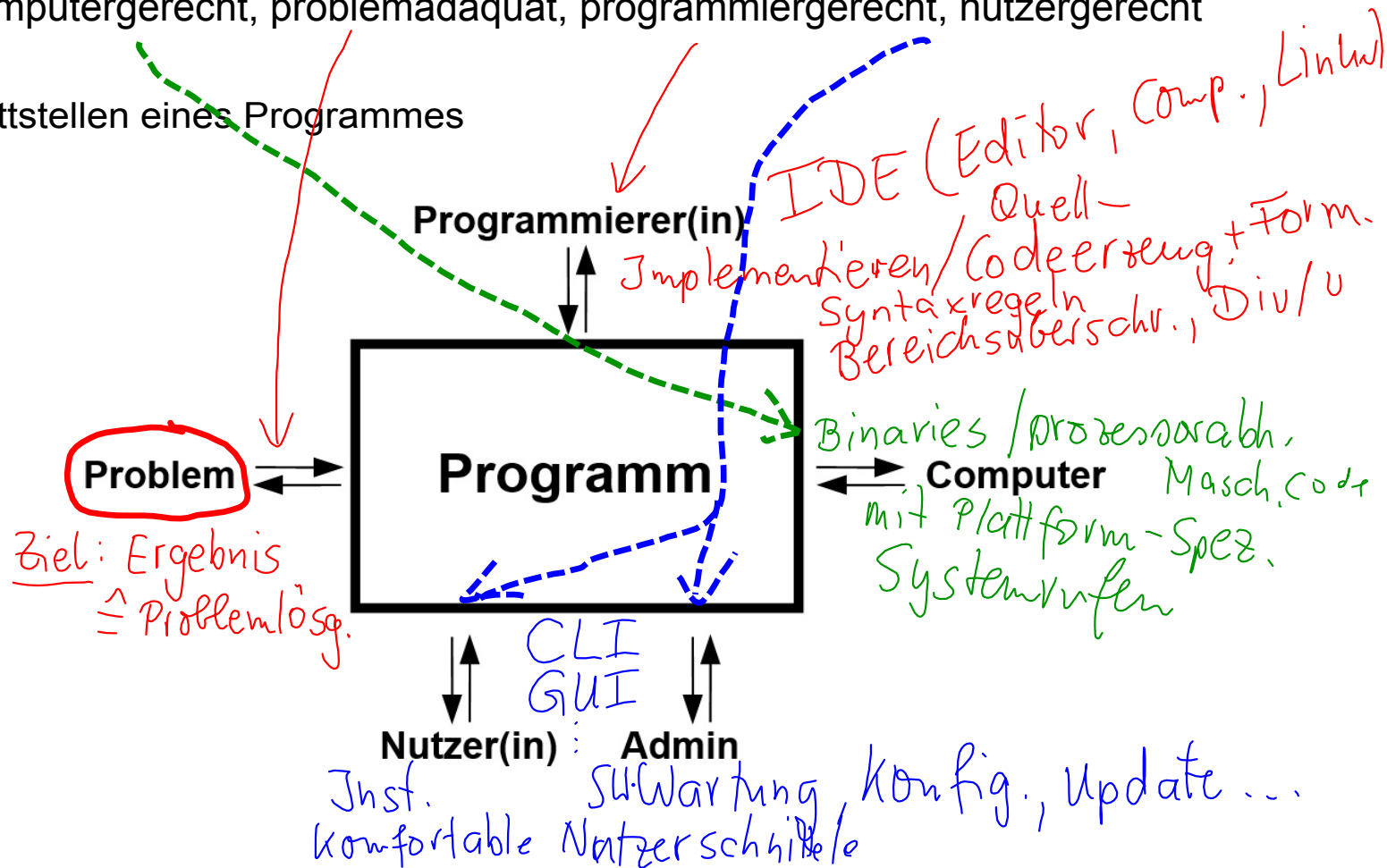
Migration: Programmumstellung (mit Übernahme der Daten)

Programme

Anforderungen

→ computergerecht, problemadäquat, programmiergerecht, nutzergerecht

Schnittstellen eines Programmes



Programmentwicklungszyklus

(compilierende Arbeitsweise)

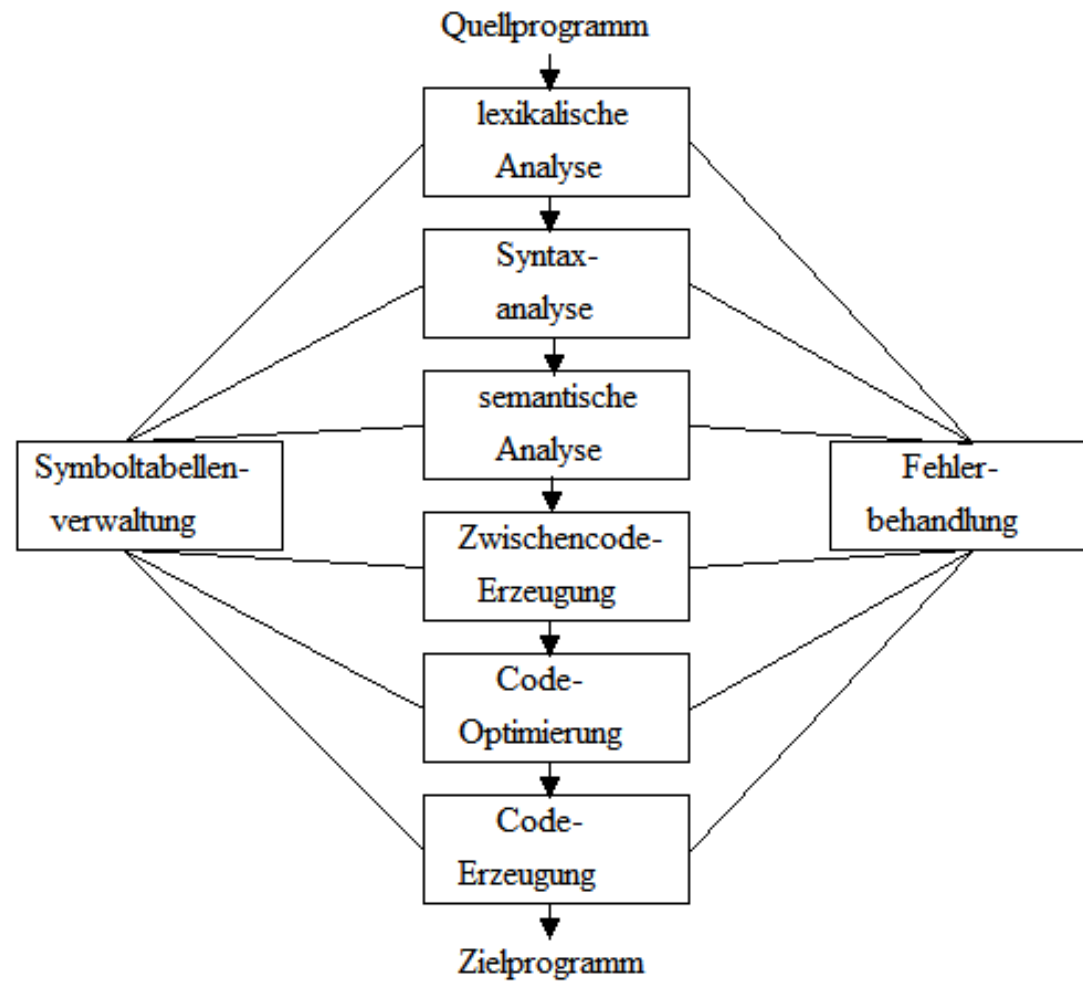
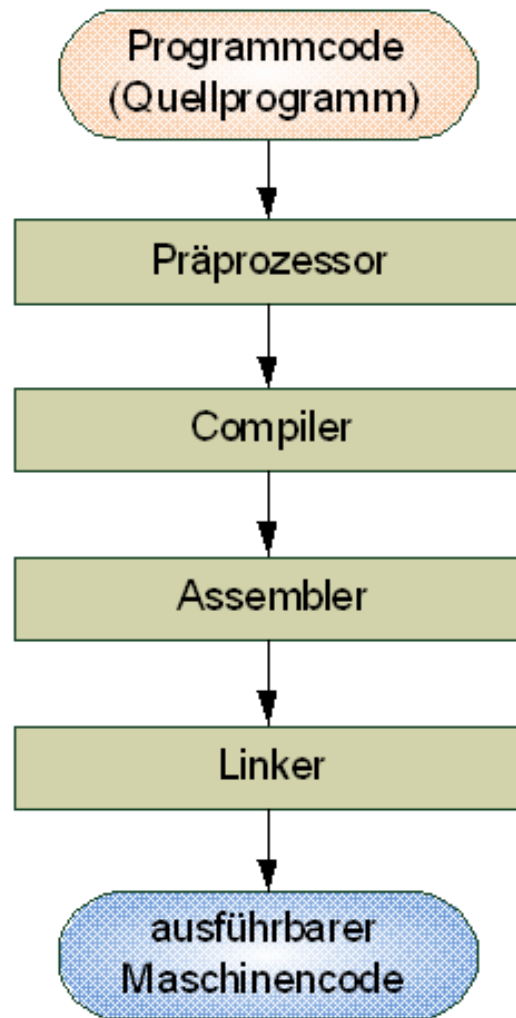
algor. Problemmodell.

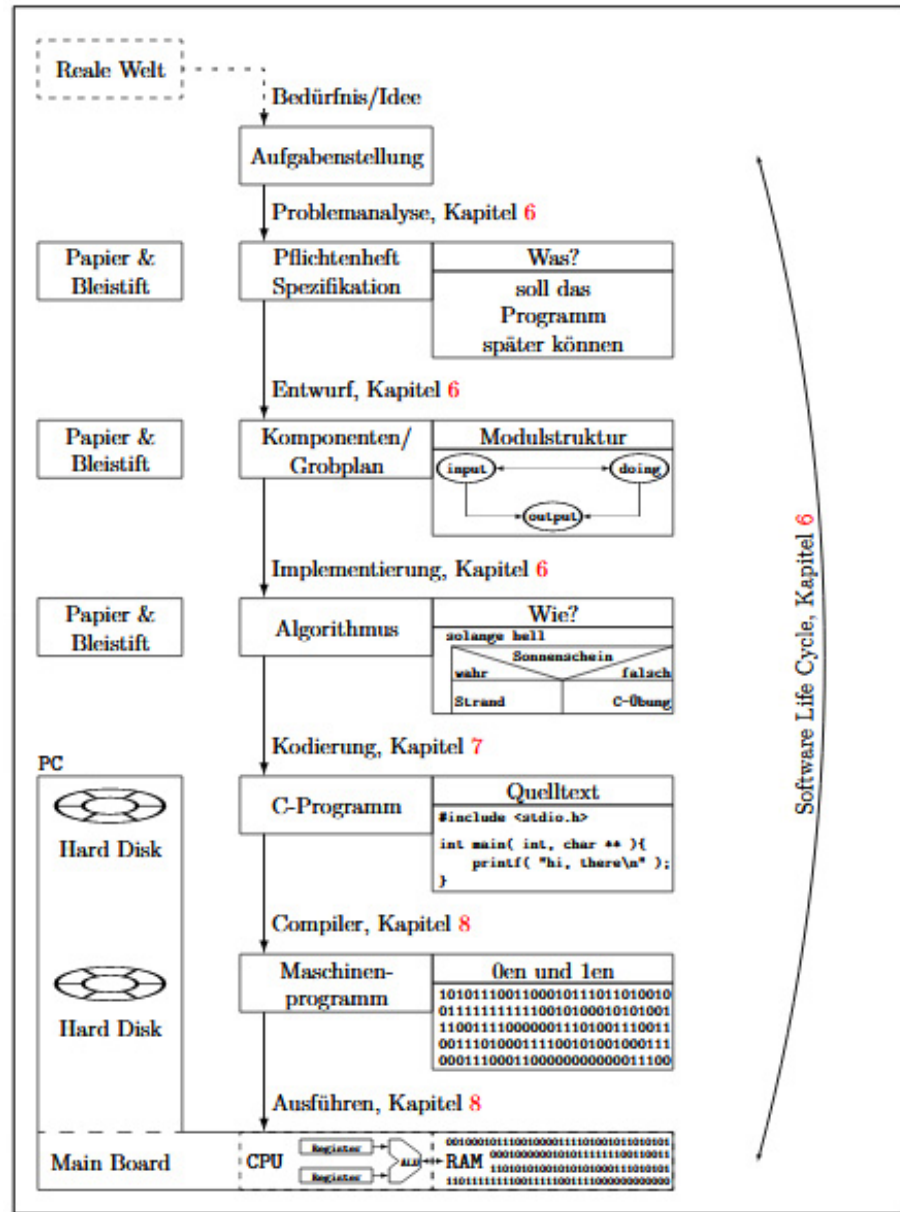
- 1) algorithmischer und struktureller Entwurf (PAP, Struktogramm, UML-Diagramm)
- 2) Erarbeitung/Editieren des Quelltextes: Implementation/Codierung des Programmes in einer Programmiersprache mit Hilfe eines Texteditors, evtl. auch mit Hilfe eines Quelltextgenerators, d.h. mittels CASE-Tool
- 3) Übersetzen des Quelltextes mittels eines Compilers:
Einpaß-bzw. Mehrpaßcompiler; evtl. mit Codeoptimierung;
Erzeugung des Maschinencodes [Objektcodedatei= *.OBJ]
mit provisorischen Einsprung- und Rücksprungadressen
- 4) Verbinden ("linken") des Programmes, d.h. Zusammensetzen der verschiedenen compilierten Module und der Laufzeitumgebung zu einem ausführbaren Lademodul ("binaries") mit Hilfe des Linkers. Dabei findet eine Klärung der gegenseitigen Referenzen (Berechnung der Variablen- und Moduladressen) einschließlich der Anbindung der verwendeten Bibliotheksmodule statt. Außerdem wird ein für das jeweilige Betriebssystem genormter Programmheader erzeugt. Im Gegensatz zu diesem 'statischen Bindevorgang' werden beim 'dynamischen Binden' (= späte Bindung) die einzelnen Moduln erst zur Laufzeit aus den dynamischen Linkbibliotheken bzw. aus dem Hauptspeicher geladen und gebunden. Damit wird u.a. ein Code-Sharing durch mehrere Applikationen ermöglicht. DLL

Programmentwicklungszyklus

(compilierende Arbeitsweise)

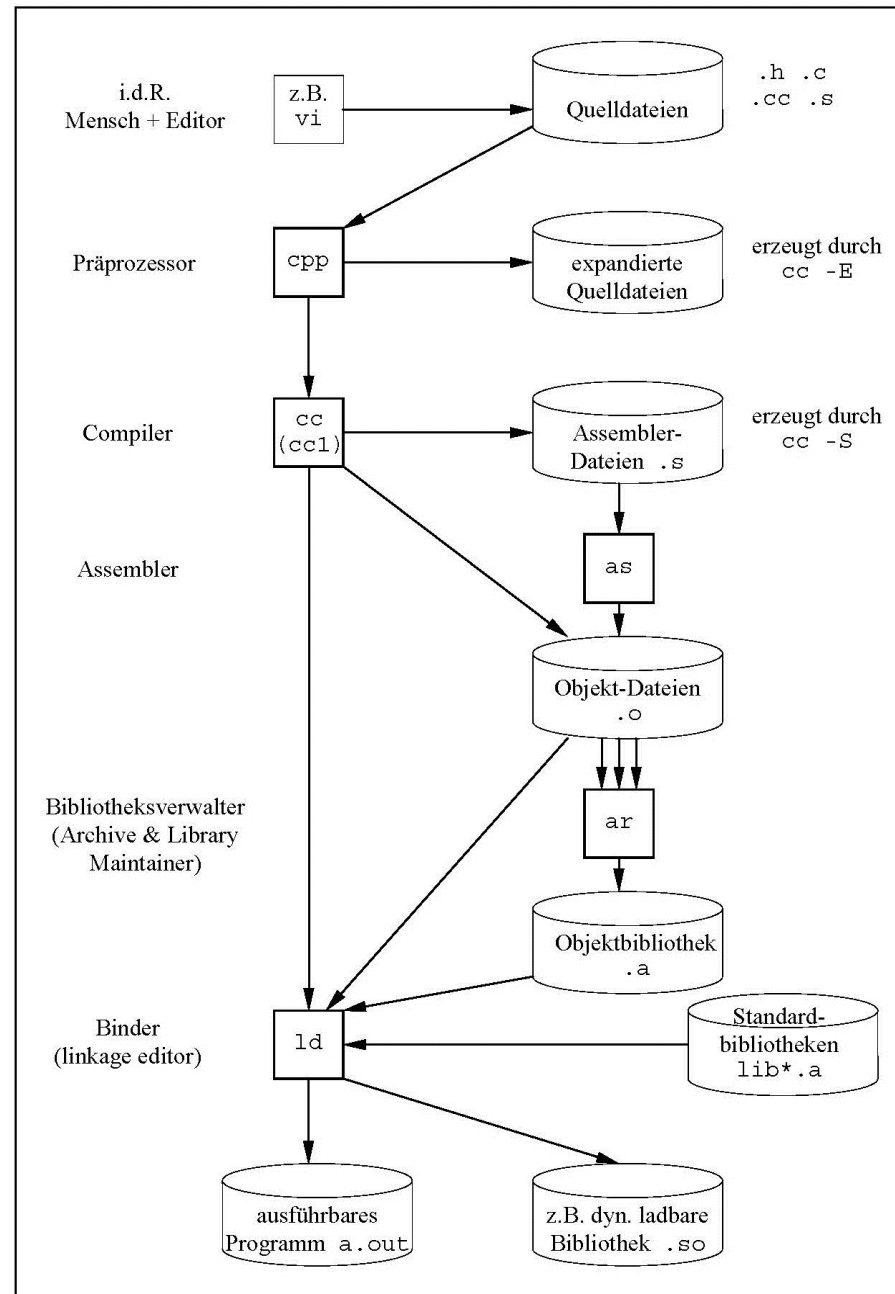
- 5) Laden des Lademoduls in den Hauptspeicher und Start des Programmes
an dem "Entry Point" (Startadresse) → Laufzeit, Prozess
 - 6) Testen des Programmes z.B. mit Hilfe eines Debuggers
Hierbei wird der Lademodul schrittweise (z.B. zeilenweise "Tracing") abgearbeitet.
Es können Programmunterbrechungspunkte ("Breakpoints") gesetzt, momentane
Variablenbelegungen und Modulaufrufverschachtelungen angezeigt sowie
Prozessorregister und Variablenwerte verändert werden.
In "Watch-Points" kann man aktuellen Werte ausgewählter Variablen anzeigen.
Bei Quelltextdebuggern besteht die Möglichkeit des Verfolgens des
Programmablaufes am Quelltext.
- bei einer interpretativen Arbeitsweise wird der Quelltext "zeilenweise"
mittels eines Interpreters übersetzt und sofort abgearbeitet





<https://www.amd.e-technik.uni-rostock.de/ma/rs/lv/hopi/script.pdf>





GNU-Projekt

GNU-Projekt: Projekt zur Entwicklung "Freier Software" in weltweiter Teamarbeit, welches von Richard Stallman am 27.Sept.1983 initiiert wurde

https://en.wikipedia.org/wiki/GNU_Project <http://www.gnu.org/>

GNU (recursives Akronym mit der Bedeutung: "GNU's not Unix!") - "DJI-N-JUH"

siehe auch: GNU General Public License (GPL)

https://de.wikipedia.org/wiki/GNU_General_Public_License

Free Software Foundation (FSF)

https://de.wikipedia.org/wiki/Free_Software_Foundation, <https://www.fsf.org/>



Richard Stallman

stallman.org



GNU Maskottchen

By Ruben Rodriguez - <https://media.libreplanet.org/u/libreplanet/m/richard-stallman-at-libreplanet-2019-2113/>,

CC BY 4.0, <https://commons.wikimedia.org/w/index.php?curid=79484097>

GNU Software

3dlat	8sync	azps	acot	acom	adns	alive
aruobis	api	archimedes	aris	artanis	aspell	audtex
autoconf	autoconf-archive	autogen	automake	avi	ballandpaddle	barcode
basn	bayonne	bazaar	bc	btb	binutils	bison
bool	bpef2owin	c-graph	ccaudio	cod2cue	ccode	corp
coscript	cfow	cgicc	chess	clm	classpath	classpathx
cilep	combine	commoncpp	complexity	conffg	consensus	coreutils
cpio	cppl	csc	csynth	dap	datamash	dc
ddd	ddrescue	dejagnu	denemo	dia	dico	dictionary
diffutils	dionysus	direvent	djgpp	dominion	dr-geo	easejs
ed	edna	electric	emacs	emms	enscript	enscript
eprints	epsilon	folix	ferret	findutils	fiscalab	foliot
fontopia	fontutils	freedink	freefont	freelmi	freetak	fridiol
g-golf	gama	garpo	gawk	gbehlstun	goal	goc
golce	gcl	goompris	gdb	gdm	genen	gengetopt
gettext	gforth	ggradebook	ghostscript	gilt	gimp	glean
gleem	glib	global	glik	glue	gmediaserver	gmp
gnaah	gnat	gnats	gnatsweb	gneuralnetwork	gnome	gnowsys
gnu-c-manual	gnu-crypto	gnu-pw-mgr	gnucae	gnuastro	gnubatch	gnubg
gnubiff	gnubik	gnucap	gnucash	gnucobol	gnucomm	gnudocs
gnue	gnufm	gnugo	gnutt	gnujdoc	gnujump	gnukart
gnutls	gnunach	gnumed	gnumeric	gnun3d	gnun	gnunet
gnupg	gnupod	gnuprogjava	gnuradio	gnurobots	gnuschool	gnushogi
gnusound	gnuspeech	gnuspool	gnustandards	gnustep	gnutls	gnutrition
gnuzilla	goptical	gorm	gpaint	gperf	gprolog	grabcomics
greg	grep	greti	grotf	grub	gsasl	gsesgrafix
gsf	gsfp	gsfc	gsa	gtick	gtk+	gtypist
gulle	gulle-ov	gulle-dbi	gulle-gnome	gulle-nourses	gulle-opengl	gulle-rtc
gulle-sdl	gutx	gurgle	gv	gvpe	gwl	gmessage
gzlp	halifax	health	hello	help2man	hp2xx	html-info
httptunnel	hard	hyperbole	loecat	loutils	lgnutil	indent
inetutils	inkingreader	intfonts	jacal	jami	java-getopt	jel
iw	jwhols	kawa	kopi	leg	less	libc
libool	libobn	liberty-efifel	libextractor	libfcall	libgcrjpt	libiconv
libon	libjt	libmathval	libmicronrtpd	libredwg	librejs	liblsgsegv
libtasn1	libtool	libunistring	libxml	lightning	lilypond	lms
linux-libre	liquidwar6	lispintro	lrzsz	lsh	m4	macchanger
mailman	mailutils	make	marst	maverik	mc	mcron
mosim	mdk	mediagoblin	melting	mempool	mes	metaexchange
metanmi	metalogic-inference	mlfluz	mlg	miscfiles	mlt-scheme	moe
motl	mpc	mpr	mpria	mtools	nana	nano
nano-archimedes	ncurses	nettle	network	ocrad	octave	oleo
oo-browser	orgadoc	oslp	panorama	parallel	parted	pascal
patch	paxutils	pcb	pdf	pem	pevec	phantom_home
ples	plp	plotutils	poke	polymass	powerguru	proxknife
pspp	psychosynth	ptn	pyconfigure	pythonwebkit	qexo	quickthreads
r	radius	rcs	readline	recutils	reflex	remotectl
rotlog	rpge	rush	sather	son	screen	sed
serveez	sharutils	shepherd	shishi	shmm	shool	spwltch
silo	smalltalk	social	softee	spacechart	speex	spell
sqlutor	sro-highlight	ssw	stalkerfs	stow	stump	superopt
swlib	sysutils	taler	talkutils	tar	termcap	termutils
teseq	teximpatient	texinfo	texmacs	thales	time	tramp
trans-coord	trueprint	unifont	units	unifrt	userv	uucp
vc-dwim	vodimgner	vera	vmgen	wb	wdiff	websocket4j
webstump	wget	which	womb	xaos	xboard	xlogmaster
xmiat	zne	xorriso	zile			

gcc
gnuplot
...

<https://www.gnu.org/>



gcc - GNU Compiler Collection

gcc main options:

option	description
<code>gcc -c</code>	compile source files to object files without linking
<code>gcc -Dname[=value]</code>	define a preprocessor macro
<code>gcc -fPIC</code>	generate position independent code for shared libraries
<code>gcc -glevel</code>	generate debug information to be used by GDB
<code>gcc -Idir</code>	add include directory of header files
<code>gcc -llib</code>	link with library file
<code>gcc -Ldir</code>	look in directory for library files
<code>gcc -o output file</code>	write build output to output file
<code>gcc -Olevel</code>	optimize for code size and execution time
<code>gcc -shared</code>	generate shared object file for shared library
<code>gcc -Uname</code>	undefine a preprocessor macro
<code>gcc -w</code>	disable all warning messages
<code>gcc -Wall</code>	enable all warning messages
<code>gcc -Wextra</code>	enable extra warning messages

<https://www.rapidtables.com/code/linux/gcc.html#syntax>

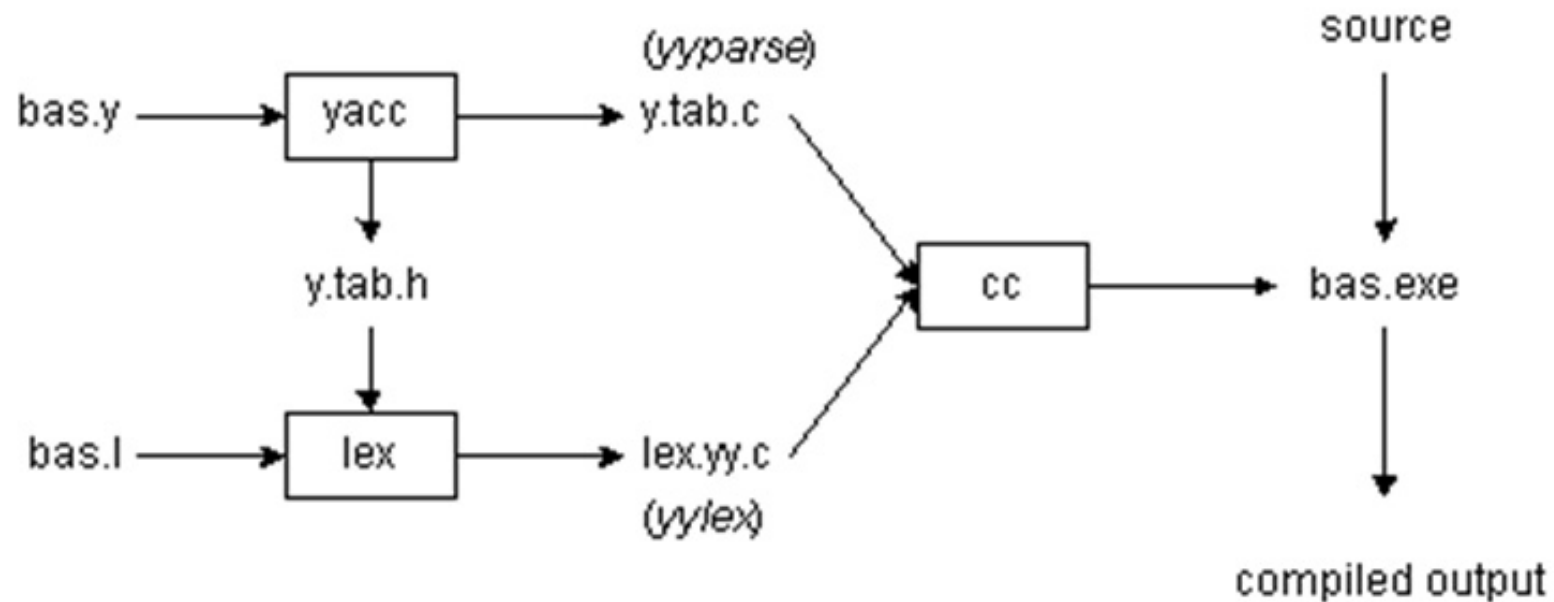
<https://gcc.gnu.org/onlinedocs/gcc/>

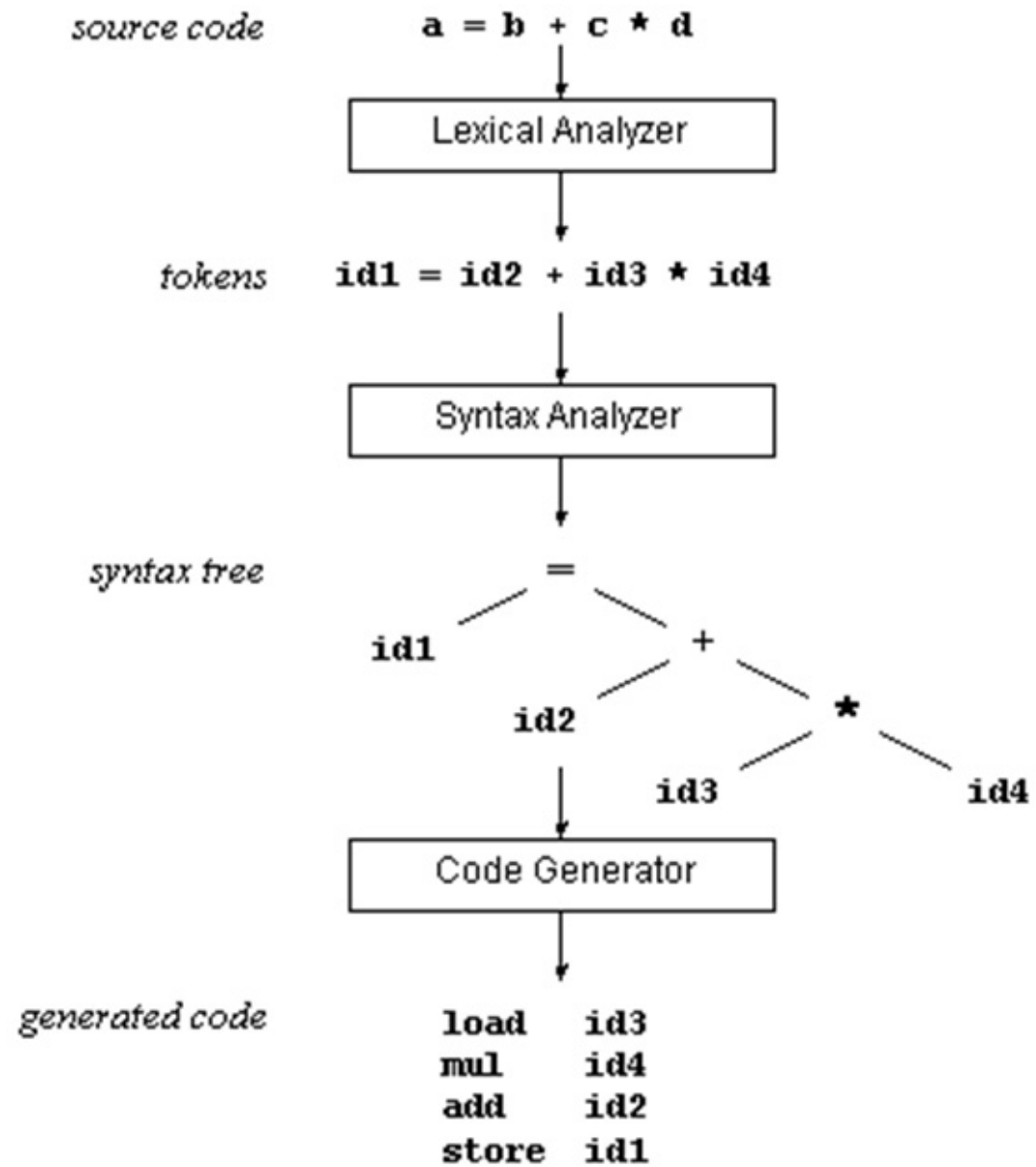
Parser/Compiler-Tools

Lex & YACC (siehe z.B. <http://www.epaperpress.com/lexandyacc/index.html>)

LEX - A Lexical Analyzer Generator (Lesk 1975)

YACC - Yet Another Compiler-Compiler (Johnson 1975)





Programmentwicklungszyklus

(compilierende Arbeitsweise)

Laufzeitbibliothek:

Module zur Realisierung komplexer Sprachkonstrukte (math. Funktionen)
Systemfunktionen, die nicht vom Betriebssystem bereitgestellt werden
Fehlerbehandlung (Fehlermeldung)

Programmierfehler:

Syntaxfehler: unkorrekte Syntax → leicht zu finden
logische Fehler: z.B. fehlende oder falsche Variableninitialisierung,
falsche Zyklen → schwer zu finden
Laufzeitfehler: Fehler zur Laufzeit, typisch z.B. Bereichsüberschreitung nach
mathematischen Operationen, Eingabe einer unzulässigen
Zeichenkombination, Division durch Null

Debuggen (entwanzen):

schrittweise Abarbeitung
oder Abarbeitung bis zu Haltepunkten (breakpoints)
Verfolgung von Variablenwerten (Watch-List)

Programmentwicklungszyklus

(compilierende Arbeitsweise)

„Unter Softwaretechnik (engl. Software Engineering) versteht man allgemein die (Ingenieur-) Wissenschaft, die die kosteneffiziente Entwicklung von qualitativ hochwertiger Software behandelt“.

GI (Gesellschaft für Informatik) / Fachgruppe für Softwaretechnik

Phasen des SW-Entwicklungsprozesses (Software Life Cycles)

[Herold, Helmut, Lurz, Bruno und Wohlrab, Jürgen. Grundlagen der Informatik. München : Pearson Studium, 2007.](#)

- 1) Anforderungsanalyse
- 2) Systementwurf
- 3) Programmentwurf
- 4) Implementierung
- 5) Test (Validierung und Verifikation)
- 6) Betrieb und Wartung