

Definition: Algorithmus

Ein **Algorithmus** ist eine allgemeine Rechenvorschrift, die aus mehreren elementaren Instruktionen (Anweisungen bei Programmiersprachen, Befehlen bei Maschinensprachen) besteht, die in einer bestimmten Reihenfolge ausgeführt werden müssen und nach einer endlichen Anzahl von Schritten zu einem Ergebnis führt.

Charakteristische Eigenschaften

1. Fintheit:
Die Rechenvorschrift muß in einem endlichen Text eindeutig beschreibbar sein. Desweiteren darf zu jedem Ausführungszeitpunkt des Algorithmus der Speicherbedarf nur endlich groß sein.
2. Ausführbarkeit:
Jede Instruktion des Algorithmus muß tatsächlich ausführbar sein.
3. Terminierung:
Der Algorithmus muß für jede Art der Eingabe ein Ergebnis liefern.
4. Determiniertheit:
Der Algorithmus muß unter denselben Eingabeparametern dasselbe Ergebnis liefern.
5. Determinismus:
Die nächste anzuwendende Instruktion ist zu jedem Ausführungszeitpunkt definiert.

Beschreibungsformen

- *Verbale Beschreibung* (abstrakte implementationsunabhängige Rechenvorschrift)
- *Pseudocode* (sprachliche Beschreibung, die näher an Programmiersprache gelehnt ist)
- Ablaufdiagramme:
 - *Programmablaufplan* (Flussdiagramm)
 - *Struktogramm* (Nassi-Shneiderman-Diagramm)
- *Programmiersprache* (C++, C, Pascal, ...)

Weitere Informationen

- Dictionary of Algorithms and Data Structures:
<http://www.nist.gov/dads/>
- Donald Knuth: *The Art of Computer Programming*, Vol 1–3, Addison Wesley 1998
- Thomas H. Cormen, Charles Leiserson, Ronald L. Rivest, Clifford Stein:
Algorithmen - Eine Einführung, Oldenbourg Wissenschaftsverlag, 2004

Beispiel: Größter gemeinsamer Teiler zweier gegebener natürlicher Zahlen a und b. (ggT)

Euklidischer Algorithmus (verbale Beschreibung):

1. wenn $a=b$ ist, dann ist a der gesuchte ggT und das Verfahren bricht ab;
sonst
2. wenn $a < b$ ist, dann werden die Werte von a und b vertauscht;
3. wenn $a > b$ ist, dann wird die Differenz $a-b=c$ gebildet;
a bekommt den Wert von b und b bekommt den Wert von c;
Fortsetzung bei 1

Beispielrechnung: gesucht sei $ggT(8,6)$

- $ggT(8,6)$:
 - 1. Schritt: 8 ist nicht gleich 6, also gehe zu Schritt 2
 - 2. Schritt: 8 ist auch nicht kleiner als 6, also gehe weiter
 - 3. Schritt: 8 ist größer 6, also ist $c = 8-6 = 2$, $a = 6$, $b = 2$
- $ggT(6,2)$: -> führt zu $a=2$, $b=4$
- $ggT(2,4)$: -> führt zu $a=4$, $b=2$
- $ggT(4,2)$: -> führt zu $a=2$, $b=2$
- $ggT(2,2)$: -> Verfahren bricht im 1. Schritt ab, Ergebnis ist 2

Definition: Programm

Ein **Programm** ist ein für den Computer in einer formalen Sprache aufbereiteter Algorithmus. Die verwendete Sprache wird "Programmiersprache" genannt, welche durch ihre Syntax und Semantik spezifiziert ist. Der Begriff "Programm" wird sowohl für Quelltexte in einer speziellen Programmiersprache verwendet, als auch für Maschinencode.

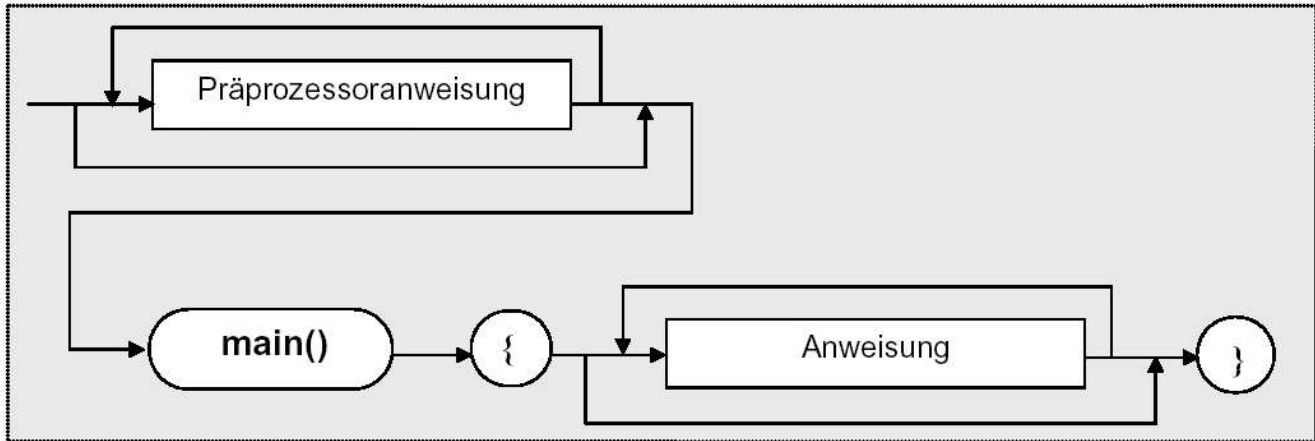
Als weltweit erstes Computerprogramm gilt eine Vorschrift für die Berechnung von Bernoulli-Zahlen, die Ada Lovelace in den Jahren 1842/1843 für die mechanische Analytical Engine von Charles Babbage erstellte. Dieses Programm konnte zu ihrer Zeit nur von Hand ausgeführt werden, denn wegen Fertigungsproblemen gab es im 19. Jahrhundert keine funktionsfähige Maschine.

C++ Compiler

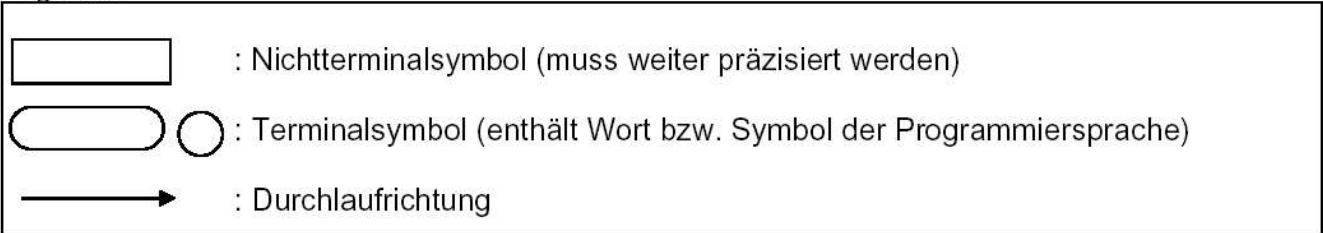
- **g++**: Kommandozeilencompiler (Linux, Windows, ...): <http://gcc.gnu.org/>
- **Microsoft Visual C++**: Entwicklungsumgebung für Windows, Nachfolger Visual C++ 2005 Express zur Zeit (2004) frei verfügbar: <http://lab.msdn.microsoft.com/express/visualc/>
- **DJGPP**: Freier Kommandozeilencompiler für Windows: <http://www.delorie.com/djgpp/>

Aufbau eines C++ Programms

Vereinfachtes Syntaxdiagramm



Legende:



Vereinfachtes C++ Programmgerüst

```
#include <iostream>
using namespace std;

main()
{
    // Programmanweisungen getrennt durch Semikoleon
}
```

Funktion main() = initiale Programmfunktion

Präprozessor

Ein **Präprozessor** ist ein Programm, welches einen Eingabetext konvertiert und das Ergebnis ausgibt. Die Konvertierung kann durch Präprozessoranweisungen beeinflusst werden.

Der C++ Präprozessor läuft vor dem eigentlichen Kompilier-Vorgang und realisiert verschiedene nützliche Aufgaben, z.B.:

- Kopieren (`#include <iostream>`) von Dateien in den Quelltext
- Definition von Textersetzungen, sogenannten "Makros" (`#define PI 3.14159`)
- bedingtes Kompilieren mittels "Compiler-Direktiven" (`#if`, `#elif`, `#else`, `#endif`)

Ein- und Ausgabe mit Streams

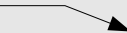
Der Begriff **“Stream”** bezeichnet einen “Strom” von Zeichen, welcher eingelesen oder ausgegeben werden kann.

Standardstreams

- Eingabestream:
 - `cin`: Stream-Objekt für die Standardeingabe (meist Kommandozeile)
- Ausgabestreams:
 - `cout`: Stream-Objekt für die Standardausgabe (meist Kommandozeile)
 - `cerr`: Stream-Objekt für ungepufferte Fehlerausgabe)
 - `clog`: Stream-Objekt für gepufferte Fehlerausgabe)

Fähigkeiten von Streams

- Operatoren `>>` und `<<` für formatierte Ein- und Ausgabe
- Manipulatoren, die in den Ein-/Ausgabe-Strom eingefügt werden (siehe 2. Praktikum)
- Methoden zu unformatierten Ein-/Ausgabe, Statusabfrage, Flag-Veränderung etc.

Beispiel: Manipulator für Zeilenumbruch 

```
cin << "Bitte eine Zahl eingeben:" << endl;  
cout >> my_variable;
```

Escape-Sequenzen (Steuerzeichen)

Escapesequenz	Bedeutung
<code>\n</code>	Cursor auf den Anfang der nächsten Zeile setzen (wie endl-Manipulator)
<code>\t</code>	Cursor macht horizontalen Tabulator-Sprung (normalerweise 8 Zeichen)
<code>\r</code>	Cursor springt an den Anfang der Zeile
<code>\b</code>	Cursor geht eine Stelle nach links

Vollständige Liste: http://www.cppreference.com/escape_sequences.html

Variablen und Konstanten

Zulässige Namen

In einem Programm werden z.B. Variablen und Funktionen über *Namen* angesprochen.

- Name besteht aus Folge von Buchstaben (A-Z, a-z), Ziffern (0-9) oder Unterstrichen (_), wobei zwischen Groß- und Kleinschreibung unterschieden wird
- erstes Zeichen muß ein Buchstabe oder Unterstrich sein
- Name kann beliebig lang sein, alle Zeichen sind signifikant
- C++-Schlüsselworte sind reserviert und dürfen nicht als Name verwendet werden

Deklaration von Variablen

Eine **Variable** ist ein Bezeichner für einen Platzhalter im Hauptspeicher, welcher typabhängig eine bestimmte maximale Größe hat und stets verändert werden kann.

In C++ muß jede Variable vor ihrer Verwendung vereinbart werden:

Syntax: `datentyp name1 [, name2 ...]`

Beispiel:

```
int i, zaehler;  
char c;
```

Initialisierung

Eine Variable kann gleich während ihrer Deklaration initialisiert und somit definiert werden.

Syntax: `datentyp name1=wert1 [, name2=wert2 ...]`

Beispiel:

```
int zaehler=0,  
    i=10, counter=100;
```

Jede verwendete Variable sollte initialisiert werden, da sie unter bestimmten Umständen sonst einen undefinierten Wert annimmt.

Konstanten

Eine **Konstante** ist ein Bezeichner für einen Platzhalter im Hauptspeicher, welcher typabhängig eine bestimmte maximale Größe hat und nicht mehr verändert werden kann.

2 Arten der Konstantendefinition:

- Präprozessoranweisung :
- mit const im Programm:

```
#define GRUSS "Hallo Welt"
```

```
const char* = "Hallo Welt"
```

Elementare Datentypen

Zahlen

Ganze Zahlen		Gleitkommazahlen	
char	1 Byte	float	4 Byte
short int	2 Byte	double	8 Byte
int	2, 4 Byte	long double	8, 10, 16 Byte
long int	4, 8 Byte		

Länge der internen Darstellung variiert von Rechner zu Rechner und Compiler zu Compiler.

- Ermitteln der internen Datenstrukturlänge: `sizeof(...)`

```
#include <iostream>
using namespace std;

main()
{
    char Ch;
    int In;
    short Sh;
    long LoI;
    float Fl;
    double Do;
    long double LoD;

    cout << "char:          " << sizeof(Ch) << " Byte\n";
    cout << "int :            " << sizeof(In) << " Byte\n";
    cout << "short:         " << sizeof(Sh) << " Byte\n";
    cout << "long int:      " << sizeof(LoI) << " Byte\n";
    cout << "float:         " << sizeof(Fl) << " Byte\n";
    cout << "double:        " << sizeof(Do) << " Byte\n";
    cout << "long double:" << sizeof(long double) << " Byte\n";
}
```

GNU C++ Compiler Ausgabe:

```
char:          1 Byte
int :          4 Byte
short:         2 Byte
long int:      4 Byte
float:         4 Byte
double:        8 Byte
long double: 12 Byte
```

MS Visual C++ Compiler Ausgabe:

```
char:          1 Byte
int :          4 Byte
short:         2 Byte
long int:      4 Byte
float:         4 Byte
double:        8 Byte
long double: 8 Byte
```

Wahrheitswerte

- werden durch den Datentyp `bool` repräsentiert, welcher Werte "true" oder "false" annehmen kann (intern wird `true` mit 1 und `false` mit 0 repräsentiert)

```
bool repeat=true;
char eingabe;
do{
    cout << "Beenden? (j/n) : ";
    cin >> eingabe;
    if (eingabe == 'j')
        repeat=false;
} while(repeat);
```

Repräsentation von Zeichen und Zeichenketten

- einzelne Zeichen: `char` (stets in einfachen Anführungszeichen)
- Zeichenketten:
 - `char kette[anzahl]` : ein Feld namens *kette* der Länge *anzahl*, welches Elemente vom Typ `char` enthält (Vorsicht: Überlauf kann auftreten)
 - `string` : Objektdatentyp zur Zeichenkettenrepräsentation (überlaufsicher) => kann benutzt werden, wenn Liste verfügbarer Klassen, Funktionen etc. eingebunden wird: `#include <string>`

```
#include <iostream>
#include <string>
using namespace std;

main()
{
    char newline = '\n';
    char name[20];
    string ort;

    cout << "Geben Sie ihren Vornamen ein (max. 20 Zeichen): ";
    cin >> name;
    cout << "Geben Sie ihren Wohnort ein: ";
    cin >> ort;
    cout << "\nHallo " << name << newline;
    cout << "Sie wohnen in " << ort << endl;
}
```

Darstellungsformen von Konstanten

Ganzzahlige Konstanten

- Darstellung als Dezimal-, Oktal- oder Hexadezimalzahl (siehe 1. Übung)
 - Dezimalzahl: beginnt mit einer von 0 verschiedenen Zahl: 12, 33
 - Oktalzahl: beginnt mit einer führenden 0: 012, 033 (Dezimalwert 27)
 - Hexadezimalzahl: beginnt mit 0x oder 0X: 0x12, 0X33 (Dezimalwert 51)

Gleitpunktkonstanten

- werden stets dezimal dargestellt, wobei gebrochener Anteil durch Punkt getrennt wird
- zulässige Darstellungsformen:

5.19	12.	0.75	0.00004
0.519E1	12.0	.75	0.4e-4
0.0519e2	.12E+2	7.5e-1	.4E-4
519.0E-2	12e0	75E-2	4E-5

Zeichenkonstanten

- in einfachen Anführungszeichen eingeschlossene Zeichen: 'A', '2', '\n'
- jedes Zeichen ist intern durch einen Zeichencode repräsentiert (ASCII-Code), die Konstante 'A' hat den Wert 65

String-Konstanten

- wie oben erwähnt werden diese durch doppelte Anführungszeichen begrenzt: "Hans"
- intern als Feld von Zeichen repräsentiert, welches durch String-Endzeichen begrenzt wird:

"Hans" - - ->

'H'	'a'	'n'	's'	'\0'
-----	-----	-----	-----	------

Vergleichsoperatoren

- jeder Vergleich ein Ausdruck vom Typ `bool`, z.B. : `hoehe == laenge` (true oder false)

<i>Operator</i>	<i>Bedeutung</i>
<	kleiner
<=	kleiner gleich
>	größer
>=	größer gleich
==	gleich
!=	ungleich

<i>Priorität</i>	<i>Operator</i>
hoch	arithm. Operatoren
↑	< <= > >=
↓	== !=
niedrig	Zuweisungsoperatoren

```
5 >= 6           ... false
4+2 == 5        ... false
2*4 != 7        ... true
test = 2*4 == 8 ... true
```

Logische Operatoren

Logische Operatoren sind boolesche Operatoren, mit denen zusammengesetzte Bedingungen (logische Ausdrücke) formuliert werden können. Die Operanden sind stets vom Typ `bool`.

Operator	Bedeutung	A	B	A B	A && B	!A	!B
<code>&&</code>	UND	true	true	true	true	false	false
<code> </code>	ODER	true	false	true	false	false	true
<code>!</code>	NICHT	false	true	true	false	true	false
		false	false	true	false	true	true

Beispiele für logische Ausdrücke

x	y	Logischer Ausdruck	Ergebnis
1	-1	<code>x <= y y >= 0</code>	false
0	0	<code>x > -2 && y == 0</code>	true
-1	0	<code>x && !y</code>	true
0	1	<code>!(x+1) y-1 > 0</code>	false