Algorithmen und Programme

Protokolliert von Rouven Czerwinski

Version vom 2. November 2011

Inhaltsverzeichnis

Tabellenverzeichnis

1 Einführung

- Kleinstcomputer (eingebettete Systeme) mit Alg. in allen Bereichen des täglichen Lebens: Taschenrechner, Handy, DvD-Player, MP3-Player, Waschmaschine, TV, Autos, Funkuhren...
- Programmierkenntnisse werden erwartet:
  - Programmierung und Steuerung komplexer Geräte und Maschinen
  - Erstellung interaktiver Medien (Internet, Videospiele, DVD, BluRay, E-Books...)
  - Verwaltung und Auswertung von Datenbanken

1.1 Algorithmusbegriff

Intuitiv: Alg. = Verarbeitungsvorschrift
Im Alltag z.B. Kochrezept, Spielregeln, Noten, Waschmaschinenprogramme, ...
Man spricht von einem Alg., wenn die Vorschrift präzise, eindeutig, vollständig und ausführbar ist.

Definition:

> Ein Alg. ist eine präzise formulierte Verarbeitungsvorschrift, die unter Verwendung elementarer Operationen einen Eingangszustand bzw. Einganswerte in einen Ausgangszustand bzw. Ausgangswerte überführt

Formal: Abbildung f: Eingabe $\rightarrow$ Ausgabe

Beispiele:

- Mathematische Formeln: $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ z.B. Addition zweier Zahlen $f(q, p) = q + p$
- Primzahlentest: $f : \mathbb{N} \rightarrow \{ja, nein\}$

$$f(n) = \begin{cases} \text{ja, falls } n \text{ Primzahl} \\ \text{nein, sonst} \end{cases}$$

- Euklidischer Alg. ggT(x,y)

Alg. dienen zur Lösung von Problemen, sie werden als Programme so abgefasst, dass sie von einem Rechner ausgeführt werden können:

$$\underbrace{\text{Problem} \rightarrow \text{Algorithmus} \rightarrow \text{Progr}}\text{amme} \rightarrow \text{Maschine}$$

Gegenstand der Vorlesung

2 Algorithmische Grundkonzepte

2.1 Eigenschaften von Algorithmen

- Terminiertheit
  Ein Alg. terminiert, wenn er für jede Wahl von gültigen Eingabewerten nach endlich vielen Schritten anhält

- Determiniertheit
  Ein Alg. ist determiniert, wenn er bei gleicher Eingabe stets auf das gleiche Ergebnis führt.

- Determinismus
  Ein Alg. ist deterministisch wenn er bei gleicher Eingabe stets über die gleichen Zwischenergebnisse zum gleichen Ergebnis führt.

- Beispiel: Berechnung eines Terms
  - hält immer an $\Rightarrow$ terminiert
  - gleiches Ergebnis $\Rightarrow$ determiniert
    $\Rightarrow$ nicht deterministisch

2.2 Daten, Operanden und Operationen

Daten:

- Darstellung von Informationen im Rechner zur Eingabe, Verarbeitung und Ausgabe

- zB. Zahlen, Zeichen, Texte, Tabellen, Graphen, Bilder, ...

- Rechnerinterne Darstellung zB. (komprimiert vs. unkomprimiert)

Datentyp:

- Zusammenfassung von Wertebereich und darauf def. Operationen zu einer Einheit

- Beispiel: Standarddatentypen: int, float, char, ...

- Ein Alg. lässt sich auffassen als Anwenden von Operationenauf Objekte bestimmten Datentyps (=Operanden).

- Operand können Konstanten, Variablen oder Ausdrücke sein.

- Ausdrücke (Terme) entstehen indem Operanden mit Operationen verknüpft werden
- Datentypen legen die Wertemenge fest, aus der die Operanden Werte annehmen können

2.3 Standard-Datentypen

:
(In den meisten Programmiersprachen vorgegeben)

2.3.1 Integer:

(in der Programmiersprache C/C++: int)
C/C++ Beispiel: (Deklaration einer Variabln a vom Typ Integer)

```
int a; // Kommentar
```

Datentyp, Variablenname, Befehlsende
Wertemenge: Z = {. . . ,-1,0,1,2,...} (im realen Rechner nach oben und unten begrenzt)
Rechenoperationen: +, -, *, /, % (modulo), ++ (Inkrement), - -(Dekrement)
Zuweisungsoperator: =
Vergleichsoperatorn: <, >, == (gleich), != (ungleich), <= (kleiner gleich), >=, . . .
C/C++ Beispiel:

```
int a; // Deklaration der Variablen a
int b; // Deklaration der Variablen b
a = 5; // Wertezuweisung: a wird auf 5 gesetzt
b = a +8;
```

Variablen bestehen aus einem Namen (Referenz auf einen Speicherplatz) und einem Wert (Inhalt des Speicherplatzes)

```
int c;
c = b / a; // Division: c wird auf b/a also 2 gesetzt
```

Problem bei der Integer-Division:
Ergebnis wird ganzzahlig abgerundet: 13/5 ergibt 2,
Allg. zur Berechnung des Restes der ganzzahligen Division b/a:

```
int rest;
rest = b - (b/a)*a;
```

Der Rest kann in C/C++ auch durch den Modulo -Operator % beschrieben werden

```
rest = b % a;
```

2.3.2 Real:

(in C/C++: float und double)
Wertemenge: Q (I'm realen Rechner nur eine Teilmenge von Q)
Operatoren wie oben
C/C++ Beispiel

```
double c,d; // Deklaration der Var. c und d
c = 0,3; // c wird auf 0,3 gesetzt
d = (4,5 - c)*1,5; // d <- 6,3
```

2.3.3 Character:

(in C/C++: char)
Wertemenge zB. {'a', 'b',...,'A','B', ...,'1','2',...,'#', ...}

```
char e;
e = 'z';
char f = '#'
```

2.3.4 String:

(in C kein Standarddatentyp, statt dessen Array vom Typ char, in C++ std::string)
Wertemenge: Zeichenketten, zB. "Hallo", "Guten Tag", ...
C++ Beispiel:

```
char wort[6] = "Hallo"; //Deklaration als char-array
std::string s = "Guten Tag"; // Deklaration als std::string
```

2.3.5 Boolean:

(in C++ bool; in C kein Standarddatentyp, stattdessen int)
Wertemenge: {true, false} btw. {0,1}
Einstelliger: ! (not) Zweistellige Verknüpfungensoperatoren: && (and), || (or), == (gleich), != (ungleich), ...
C++ Beispiel

```
bool ergebnis,op1,op2;
op1 = true;
op2 = !op1; // op2 wird false
ergebnis = op1 && op2; // ergebnis wird false
Ergebnis = op1 || op2; // ergebnis wird true
```

Verknüpfungstabellen:

Tabelle 1: And Tabelle

op1	&& op2	Ergebnis
true	true	true
true	false	false
false	true	false
false	false	false

Tabelle 2: Or Tabelle

op1	|| op2	Ergebnis
true	true	true
true	false	true
false	true	true
false	false	false

Tabelle 3: Negation Tabelle

!op1	Ergebnis
true	false
false	true

Die Reihenfolge der Auswertung von Ausdrücken ergibt sich durch Klammerung und Vorrangregeln (Punkt vor Strich usw z.B. $(a * b + c * d/e)$)
Beispiel:

```
bool Ergebnis,op1,op2,op3;
Ergebnis = op1 || op2 && op3;
Ergebnis = op1 || (op2 && op3);
```

3 Imperative Algorithmen

- Basis für imperative Programmiersprachen wie C, Pascal, Modula, Basic, PHP, ...
- bekannteste/häufigste Art Alg. zu formulieren
- Alternative: Deklarative Programmierung z.B.
  funktionale Programmiersprachen wie z.B
  Lisp, Scheme, Haskell, ...

3.1 Ein- und Ausgabe

C++ Beispiel

```
int a;
  cin >> a; //Eingabe des Wertes von a ueber Konsole
  cout << "Sie haben" << a << "eingegeben" << ende; // Ausgabe
```

3.2 Verzweigungen (bedingte Anweisung)

Mit Verzweigungen können in Abhängigkeit einer Bedingung unterschiedliche Anweisungen ausgeführt werden.
Syntax:

```
if (BEDINGUNG) ANWEISUNG;
else ANWEISUNG2;
```

Interpretation:
Führe ANWEISUNG1 aus, falls die boolsche BEDINGUNG wahr (true) ist, ansonsten führe ANWEISUNG2 aus. (Der Else-Teil ist optional)
C++ Beispiel:

```
if (a<0) cout << "a ist negativ" << endl;
else cout << "a ist positiv" << endl;

bool b = ((a % 2) == 0); // b wird true, falls a modulo 2 null ist
if(b) cout << "a ist gerade" << endl;
```

Eine ANWEISUNG darf auch ein mit Klammern {} zusammengefasster Block von mehreren Anweisungen sein.

Syntax:

```
if (BEDINGUNG)
{
  Anweisung 1.1
  Anweisung 1.2
}
```

Tabelle 4: Modulo Tabelle

a	$a/2$	a%2	$a/3$	a%3
0	0	0	0	0
1	0	1	0	1
2	1	0	0	2