

Algorithmen und Datenstrukturen in C/C++

Prof. Dr. rer. nat. Peter Gerwinski

16. April 2018

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 ...

Rasterung

$O(n \log n)$

array

struct

Wegfindung

CORDIC

FFT

B-Baum

Verschlüsseln

pointer

verkettete Liste

Datenbanken

Datenkompression

digitale Signatur

Hash-Tabelle

Prüfsumme

kryptographische Hash-Funktion

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 Einführung in C++

3 Datenorganisation

- Listen, Bäume, Hash-Tabellen, ...

4 Datenkodierung

- Fehlererkennung und -korrektur
- Kompression
- Kryptographie

5 Hardwarenahe Algorithmen

- FFT, CORDIC, ...

6 Optimierung

- Wegfindung, ...

7 Numerik



Änderungen
vorbehalten

1 Einführung

Algorithmen

- Beispiele: https://de.wikipedia.org/wiki/Liste_von_Algorithmen
- Häufig: „Wie würde ein Mensch es machen?“
- Beispiel: Insertionsort
- Ausnahme: Überblick behalten
- Beispiel: Rekursion
- Beispiel: Stundenplanprogramm

Datenstrukturen

- Modellierung (UML)
- Beispiel: Optimierungspotential der Datenstrukturen in einem Computerspiel (592 Zeilen Python)

Übungsaufgabe: Verkettete Listen

Schreiben Sie eine Funktion, die eine verkettete Liste in umgekehrter Reihenfolge ausgibt.

- (a) Die Funktion soll möglichst schnell arbeiten.
Auf den Speicherverbrauch kommt es nicht an.
- (b) Die Funktion soll möglichst wenig Speicher verbrauchen.
Auf die Rechenzeit kommt es nicht an.
- (c) Finden Sie einen Kompromiß zwischen (a) und (b).

Geben Sie für alle Funktionen das Landau-Symbol für die Laufzeit und für den Speicherverbrauch an.

Hinweis zu (a): Man kann es rekursiv machen.

2 Einführung in C++

2.0 Was ist C?

Etabliertes Profi-Werkzeug

- kleinster gemeinsamer Nenner für viele Plattformen



Hardware und/oder Betriebssystem

- Hardware direkt ansprechen und effizient einsetzen
- ... bis hin zu komplexen Software-Projekten

→ Man kann Computer vollständig beherrschen.

2 Einführung in C++

2.0 Was ist C?

Etabliertes Profi-Werkzeug

- kleinster gemeinsamer Nenner für viele Plattformen
- Hardware direkt ansprechen und effizient einsetzen
- ... bis hin zu komplexen Software-Projekten
- leistungsfähig, aber gefährlich

„High-Level-Assembler“

- kein „Fallschirm“
- kompakte Schreibweise

C makes it easy to shoot yourself in the foot.

Bjarne Stroustrup, ca. 1986

http://www.stroustrup.com/bs_faq.html
[#really-say-that](#)

Unix-Hintergrund

- Baukastenprinzip
- konsequente Regeln
- kein „Fallschirm“

2 Einführung in C++

2.1 Was ist C++?

Etabliertes Profi-Werkzeug

- kompatibel zu C

C++ is a better C.

C++ unterstützt

- *objektorientierte Programmierung*
- *Datenabstraktion*
- *generische Programmierung*

Bjarne Stroustrup, Autor von C++
<http://www.stroustrup.com/C++.html>

*C makes it easy to shoot yourself in the foot;
C++ makes it harder, but when you do
it blows your whole leg off.*

Bjarne Stroustrup, Autor von C++, ca. 1986
http://www.stroustrup.com/bs_faq.html
[#really-say-that](#)

Motivation:

Vermeidung unsicherer Techniken,
insbesondere von Präprozessor-Konstruktionen und Zeigern,
unter Beibehaltung der Effizienz

2.2 Elementare Neuerungen gegenüber C

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //
- Konstante:
const int answer = 42;

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //
- Konstante:
const int n = 5;
int prime[n] = { 2, 3, 5, 7, 11 };

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //
- Konstante:
`const int n = 5;`
`int prime[n] = { 2, 3, 5, 7, 11 };`
- Ab C++11: `constexpr`-Funktionen
... *anscheinend auch ohne „constexpr“* ...
darf nur aus einem einzigen `return`-Statement bestehen
→ keine Schleife – *oder vielleicht doch außer ab C++14* –, aber
Rekursion erlaubt

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //
- Konstante:
const int n = 5;
int prime[n] = { 2, 3, 5, 7, 11 };
- Ab C++11: **constexpr**-Funktionen
... **anscheinend auch ohne „constexpr“** ...
darf nur aus einem einzigen **return**-Statement bestehen
→ keine Schleife – **oder vielleicht doch außer ab C++14** –, aber
Rekursion erlaubt
- leere Parameterliste: **void** optional

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //
- Konstante:
const int n = 5;
int prime[n] = { 2, 3, 5, 7, 11 };
- Ab C++11: **constexpr**-Funktionen
... **anscheinend auch ohne „constexpr“** ...
darf nur aus einem einzigen **return**-Statement bestehen
→ keine Schleife – **oder vielleicht doch außer ab C++14** –, aber
Rekursion erlaubt
- leere Parameterliste: **void** optional
in C: ohne **void** = Parameterliste wird nicht geprüft

2.2 Elementare Neuerungen gegenüber C

- Kommentare mit //
- Konstante:
const int n = 5;
int prime[n] = { 2, 3, 5, 7, 11 };
- Ab C++11: **constexpr**-Funktionen
... **anscheinend auch ohne „constexpr“** ...
darf nur aus einem einzigen **return**-Statement bestehen
→ keine Schleife – **oder vielleicht doch außer ab C++14** –, aber
Rekursion erlaubt
- leere Parameterliste: **void** optional
in C: ohne **void** = Parameterliste wird nicht geprüft
- Operatoren **new** und **delete**
als Alternative zu den Funktionen **malloc()** und **free()**

2.3 Referenz-Typen

```
void calc_answer (int &answer)
{
    answer = 42;
}
```

... als Alternative zu ...

```
void calc_answer (int *answer)
{
    *answer = 42;
}
```

- Zeiger „verborgen“, übersichtlicher und sicherer
- Es gibt keinen **NULL**-Wert.
→ Für verkettete Listen u. ä.: Tricks erforderlich

2.4 Überladbare Operatoren und Funktionen

```
#include <iostream>
```

```
int main ()  
{  
    std::cout << "Hello, world!" << std::endl;  
    return 0;  
}
```

Bemerkungen:

- Compilieren mit `g++` statt `gcc`:
C++-Bibliotheken mit einbinden
- Der Operator `<<` hat normalerweise keinen Seiteneffekt, hier schon.

2.4 Überladbare Operatoren und Funktionen

```
#include <iostream>
```

```
struct vector  
{  
    double x, y, z;  
};
```

```
vector operator + (vector u, vector v)  
{  
    vector w = { u.x + v.x, u.y + v.y, u.z + v.z };  
    return w;  
}
```

- ++ wird zum Präfix-Operator.
- ++ mit zusätzlichem (ungenutzten) **int**-Parameter wird zum Postfix-Operator.

2.4 Überladbare Operatoren und Funktionen

```
void print (const char *s)
```

```
{  
    printf ("%s", s);  
}
```

```
void print (int i)
```

```
{  
    printf ("%d", i);  
}
```

Für den Linker:
veränderte, eindeutige Namen

Wenn man das nicht will:
extern "C" { ... }

Optionale Parameter:

```
void print (const char *s = "\n")
```

```
{  
    printf ("%s", s);  
}
```

wird vom Compiler erledigt

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 Einführung in C++

2.0 Einführung in C

2.1 Einführung in C++

2.2 Elementare Neuerungen gegenüber C

2.3 Referenz-Typen

2.4 Überladbare Operatoren und Funktionen

2.5 Namensräume

2.6 Objekte

3 Datenorganisation

4 Datenkodierung

5 Hardwarenahe Algorithmen

6 Optimierung

7 Numerik



Änderungen
vorbehalten