

Übung zur Vorlesung "Grundlagen Betriebssysteme und Systemsoftware"

(Prof. Dr. J. Schlichter, WS 2008 / 2009)

Übungsleitung: Dr. Georg Groh (grohg@in.tum.de)

Tutoren: Dipl. Inform. Vivian Prinz (prinzv@in.tum.de), Dr. Nils Kammenhuber (kammenhuber@net.in.tum.de), Dipl. Inform. Robert Schmohl (schmohl@in.tum.de), Dipl. Inform. Dipl. Geogr. Jan Herrmann (hermanj@in.tum.de), Dipl. Inform. Robert Eigner, David Brodski (brodski@in.tum.de), Yang Guo (yang.guo@gmx.de), Jan Finis (finis@in.tum.de), Dr. Georg Groh (grohg@in.tum.de)

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809>

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809/uebung>

Blatt 1

- Abgabe: bis 27.10.2008 12:00 Uhr per E-Mail an den Tutor der eigenen Gruppe. Die Mail soll einen Zip-Ordner als attachment haben, der für jede Hausaufgabe einen Unterordner enthält, in dem die Lösung als .txt-File(s), als .c-File(s) o.ä. enthalten ist.
- Beachten Sie bitte das Format fuer den Betreff der E-Mail! (GruppenNr, Name, BlattNr, Tutor, ggf. Zusammenarbeits-Gruppenmitglieder) (Bsp: **Gr 1, Müller, Blatt 3, Tutor: Prinz, mit Schmidt und Meier zusammengearbeitet**)
- Musterlösungen Hausaufgaben: ab 27.10.2008 12:00 Uhr auf der Übungswebseite zum Download.
- Musterlösungen Tutoraufgaben: ab 3.11.2008 12:00 Uhr auf der Übungswebseite zum Download.

Stoff

Es wird empfohlen folgende Literatur durchzuarbeiten:

- Das Gnu C-Programming Tutorial (Burgess, Hale-Evans) (als pdf zum Download von der Seite <http://www.crasseux.com/books/ctut.pdf>). Hierbei sind wichtig: Kapitel 9, 14, 17, 20, 21. Völlig weglassen kann man Kapitel 16, 22, 23. Das Tutorial ist sehr einfach geschrieben und kann mit guten JAVA-Vorkenntnissen in einem Tag ohne Probleme durchgearbeitet werden. Wichtig sind vor allem die Themen/Aspekte Pointer, Pointerarithmetik, Speicherallokation.

Wegen der Einarbeitung in C gibt es auf diesem Blatt nur eine Hausaufgabe. Diese ist allerdings auch etwas umfangreicher als zukünftige Hausaufgaben.

Warum C "lernen"?

- C und Betriebssysteme sind sehr eng verbunden. Sehr viele wichtige Betriebssystem-Elemente sind in C geschrieben.
- Die Vorlesung verwendet Java und C zur Formulierung von Beispielen und für manche Übungsaufgaben
- Gewisse Erfahrungen mit C sind auch für viele andere Bereiche der Informatik von Nutzen (z.B. Computergrafik (OpenGL)) und sind ein guter Startpunkt für die Beschäftigung mit C++.
- Der Umgang mit einer Sprache, die Zeiger und unmittelbares Memory-Management unterstützt, trägt zum Background-Verständnis moderner Sprachen wie Java bei.

1 Hausaufgabe (C-Programmierung)

Lernziele

Es soll der Umgang mit C am Beispiel exemplarisch ausprobiert werden, insbesondere der Umgang mit Zeigern, Referenzen und Dynamic Memory-Allocation.

Aufgabe

Es soll ein binärer Baum in C implementiert werden. Verwenden Sie am besten ein Unix / Linux System und gcc als C Compiler. Falls Sie auf Ihrem Rechner nur Windows verwenden, empfiehlt sich z.B. Knoppix als Linux Variante, die komplett von CD / DVD lauffähig ist.

Abgabe

Als Abgabe ist ein (getestetes, compilierbares) Gesamt-.c-File, das alle Teilaufgaben umfasst, ausreichend.

1.1 Teilaufgabe

Implementieren Sie mit Hilfe von `struct` eine Datenstruktur `Node` die je einen Zeiger auf den Vaterknoten, auf den linken und rechten Sohn enthält. Außerdem soll die Datenstruktur eine `double` Variable `nodeValue` enthalten.

1.2 Teilaufgabe

Implementieren Sie eine Funktion `makeRandomTree`, die als Eingabe-Parameter eine Integer-Zahl `numberOfNodes` nimmt und als return-Parameter einen Zeiger auf `Node` liefert. Die Funktion soll mit Hilfe von dynamischer Speicher-Allokation einen binären Suchbaum mit `numberOfNodes` Knoten aufbauen, deren `nodeValue` eine (Pseudo-)Zufallszahl in $[0, 1]$ ist.

Hierbei sind folgende Aspekte interessant:

- Wie kann man das zur Instantiierung von Objekten aus Klassen verwendete `new` in Java in gewissem Sinn analoge dynamische Neu-Erzeugen von `struct` Instanzen mit Hilfe von `malloc` realisieren?
- Welchen Vorteil hat dynamic memory allocation gegenüber static memory allocation (`static` Variablen) und automatic memory allocation ((Block-)lokale Variablen)?
- Wie ist der dynamisch allozierte Speicher organisiert im Vergleich zu den anderen Varianten?
- Wie kann man Pseudo-Zufallszahlen in C erzeugen? Welche Präprozessor-Includes muss man in den Header schreiben? Muss man die zugehörige Bibliothek beim Kompilieren extra hinzu linken?

1.3 Teilaufgabe

Implementieren Sie eine Funktion `inorderTraversal`, die einen binären Baum inorder traversiert. "Bearbeiten" des Knotens soll in der Ausgabe von `nodeValue` bestehen.

1.4 Teilaufgabe

Implementieren Sie eine `main` Funktion, die einen binären Suchbaum mit 100 Knoten erzeugt, diesen inorder traversiert und dabei den Sinus der enthaltenen `nodeValue`s (mal 2π) berechnet, den Wert in `nodeValue` schreibt und untereinander ausgibt (Zuerst die Nummer des Knotens, dann den Wert). Implementieren Sie hierzu eine zweite Variante `inorderTraversalTwo` der Inorder-Traversal Funktion.

Hierbei sind folgende Aspekte interessant:

- Welche Präprozessor-Includes muss man jetzt in den Header schreiben? Muss man jetzt die zugehörige Bibliothek beim Compilen extra hinzu linken?

2 Tutoraufgabe (Pointer und Arrays in C, Pointerarithmetik)

Lernziele

Das Konzept der Pointer in C soll weiter geübt und die Kenntnisse an Beispielen vertieft werden. Insbesondere soll das Zusammenspiel von Arrays und Pointern verdeutlicht werden. Für das Verständnis und die Implementierung von Betriebssystemen ist das "systemnahe" Pointer-Konzept, das in modernen höheren Sprachen oft nicht mehr vorkommt bzw. nur implizit verwendet wird, recht hilfreich.

Aufgabe

Es sollen an einer Reihe vorgegebener C Programmstücke die beteiligten Entitäten untersucht und interpretiert werden.

Abgabe

Die Aufgabe wird in den Tutorübungen gemeinsam erarbeitet. Die Aufgabe soll NICHT abgegeben werden.

2.1 Teilaufgabe

Folgende Deklarationen und Initialisierungen seien gegeben:

```
int arrayXYZ[10]; //allocates memory for 10 int variables
int i;
int *pi;
int intVar;
for(i=0;i<10;i++)
    arrayXYZ[i]=i;
```

Wie sind die folgenden Anweisungen zu interpretieren?

```
pi = &arrayXYZ[7];
pi = arrayXYZ;
pi = &arrayXYZ[0];
```

Sind die zweite und die dritte Zuweisung äquivalent?

2.2 Teilaufgabe

Die Deklarationen aus 2.1 und folgendes Code-Stück sei gegeben:

```
intVar = arrayXYZ[8];
intVar = *(arrayXYZ+8);
pi = arrayXYZ;
*(pi + 12) = 178;
pi[12] = 178;
```

- Sind die erste und die zweite Anweisung äquivalent?
- Sind die vierte und die fünfte Anweisung äquivalent?
- Was ist das Problem bei den Anweisungen vier und fünf?
- Wenn die Deklarationen aus 2.1 mit Hilfe von `double` Variablen, Arrays und Pointern formuliert worden wären: Hätte man dann in der dritten Anweisung `*(pi + 4*12)` schreiben müssen?

2.3 Teilaufgabe

Welche der folgenden Anweisungen sind mit den Deklarationen aus 2.1 korrekt?

```
pi = arrayXYZ;
pi++;
pi = arrayXYZ;
arrayXYZ++;
arrayXYZ = pi;
```

2.4 Teilaufgabe

Interpretieren Sie die folgenden Deklarationen! Wo sind die Unterschiede?

```
int (*arrayXYZ)[10];
int *(arrayXYZ[10]);
int *arrayXYZ[10];
```

2.5 Teilaufgabe

In welcher Hinsicht ist

```
int *arrayXYZ[10];
for(i=0;i<10;i++)
    arrayXYZ[i]=(int *)malloc(10*sizeof(int));
```

flexibler als

```
int arrayXYZ[10][10];
```

?

2.6 Total optionale Zusatzaufgabe für Nerds und Kenner :-)

(Keine Besprechung, keine ML)

Welche Funktion berechnet folgender Code und wie?

```
float xhalf = 0.5*x;
int i = 0x5f3759df - ((*int*)&x)>>1;
x = *(float*)&i;
x = x*(1.5-xhalf*x*x);
```

Idee und Hintergründe siehe folgendes Paper:

<http://www.math.purdue.edu/~clomont/Math/Papers/2003/InvSqrt.pdf>

3 Tutoraufgabe

Lernziele

Vertiefung von C

Aufgabe

Folgendes Code-Stück in C sei gegeben (mit Zeilennummern):

```
....
1  int arrayXYZ[10];
2  int i;
3  int *pi;
4  int horst;
5  int intVar;
6  for(i=0;i<10;i++)
7      arrayXYZ[i]=i;
8  horst = arrayXYZ[4];
9  //eins
10 printf("%i\n", horst);
11 pi = &arrayXYZ[7];
12 horst = *pi;
13 //zwei
14 printf("%i\n", horst);
15 pi = arrayXYZ;
16 horst = *pi;
17 //drei
18 printf("%i\n", horst);
19 pi = &horst;
20 *pi = *(arrayXYZ + 3);
21 //vier
22 printf("%i\n", horst);
....
```

Welche Werte hat die Variable `horst` an den durch die Kommentare `eins`, `zwei`, `drei` und `vier` bezeichneten Punkten im Programmablauf (bzw. was ist die Ausgabe des Programm-Stücks)? (Nur die vier Werte nennen!)

Abgabe

Die Aufgabe wird in den Tutorübungen gemeinsam erarbeitet. Die Aufgabe soll NICHT abgegeben werden.