Beleg

Benjamin Herzog,
Felix Krautschuk,
Markus Noack

Modul: Internettechnologien II

Studiengang: Allgemeine Informatik

- Wintersemester 2014/15 -

Inhaltsverzeichnis

1 Vorwort

Diese Dokumentation beschreibt unsere Vorgehensweise beim Erstellen des Beleges in dem Modul Internettechnologien II. Als Aufgabe hatten wir uns die Implementierung eines vorgegebenen Algorithmus innerhalb eines Webservices ausgesucht. Dieser war in Form einer C-Funktion gegeben. Es handelt sich um den Gauss-Algorithmus zum Lösen von einer bestimmten Anzahl Gleichungen mit einer bestimmten Anzahl Unbekannter. Es stand uns dabei frei, den Algorithmus in eine andere Programmiersprache zu übersetzen und auf einem Webserver auszuführen oder den fertigen C-Algorithmus direkt auszuführen.
Aus Lernzwecken haben wir uns für die letztere Variante entschieden. Es stand dabei vor allem im Vordergrund, dass wir im vorherigen Semester mit Javascript gearbeitet hatten und auch während eines anderen Projektes schon viel damit programmiert haben. Um während der Bearbeitung des Beleges möglichst viel zu lernen, haben wir abgestimmt, dass wir die bestehende C-Datei soweit anpassen, dass wir eine kompilierte cgi-Variante auf dem Server laufen lassen können und dabei dennoch den bestehenden Algorithmus beibehalten.

2 Aufbau

2.1 Grundlegender Aufbau

Das Projekt wurde realisiert durch eine HTML-Datei, welche die Eingabe von 2 Variablen zulässt. Zum einen die Anzahl der Gleichungen und zum anderen die Anzahl der Unbekannten. Nach Betätigung des Buttons "Generiere Matrix"wird anhand der Eingaben eine entsprechende Matrix erstellt. Das Script zur Generierung dieser wurde zur Übersichtlichkeit in eine index.js-Datei ausgelagert.
Der Javascript-Code erstellt die Matrix innerhalb einer Form, bei dessen Betätigung die cgi-Datei ausgeführt wird. Damit diese reibungslos auf dem Server funktioniert, wird sie auch auf diesem kompiliert. Dafür haben wir ein kleines Shell-Script geschrieben, was vom Browser aus ausgeführt werden kann. (Eine entsprechende Einstellung auf dem Webserver vorausgesetzt.) Dieses Script kann unter **www.benchr.de/IT2/compile.sh** ausgeführt werden.
Die eigentliche Berechnung befindet sich dabei in dem gauss.c-File, welches den Algorithmus und unsere Ergänzung zum Aufruf enthält.

2.2 index.html

Die HTML-Seite wird als erstes aufgerufen, wenn man das Projekt startet. Sie trägt den Titel "IT2-Beleg" und beinhaltet die grobe Struktur, auf der das Javascript basiert.
Teil dessen sind 2 Eingabgefelder, einmal für die Anzahl der Gleichungen und einmal für die Anzahl der Unbekannten. Unter diesen befindet sich der Button mit der Aufschrift "Generiere Matrix", der bei Betätigung die Funktion start() im Script aufruft. Unter diesem wurde von uns ein Absatz eingefügt, der allerdings keinen Inhalt hat. Dieser dient als Platzhalter, um dann später im Javascript die Matrix an die richtige Stelle zu setzen.
Da wir das Script zur Übersichtlichkeit ausgegliedert haben, muss es noch in der HTML-Datei eingebunden werden. Dafür setzten wir einen script-Tag mit dem Attribut src, was auf die Javascript-Datei verweist. Der Name ist dabei gleich wie die HTML-Datei, nur die Endung unterscheidet sich.

2.3 index.js

In dieser Datei befindet sich der Code, der die Matrix zum Eingeben erstellt und diesen an die Datei weitergibt, die für die eigentliche Berechnung zuständig ist.
Um den Code möglichst skalierbar zu halten, haben wir wichtige und eventuell umzuändernde Werte an den Anfang geschrieben. In diesem Fall handelt es sich dabei lediglich um die Angabe des Maximums der Werte, welche in die Textfelder eingetragen werden können. Bei Abgabe handelt es sich dabei um die Zahl 14.
Ansonsten besteht die Datei aus zwei weiteren Funktionen. Die erste dabei ist die Funktion, die aufgerufen wird, wenn der "Generiere Matrix"-Button geklickt wurde - die Funktion **start()**.
Da es sich lediglich um einen Event Handler für den Button handelt, bekommt diese Funktion keine Parameter übergeben. Am Anfang der Funktion werden zwei lokale Variablen angelegt, die jeweils mit den Inhalten der beiden Textfelder gefüllt werden. Da als nächstes überprüft werden soll, ob die Angaben valide sind, also auch nicht größer als das oben definierte Maximum, wird hier nicht der Inhalt als Text genommen, sondern direkt die sich darin befindende Zahl geparst.
Das Auslesen geschieht durch die Methode **getElementById** von der Instanz **document**, welche das gesamte Dokument, in dem gearbeitet wird, darstellt. Das Auslesen der geschriebenen Zahl geschieht durch das Aufrufen der Property value. Wir gehen dabei davon aus, dass der Nutzer dort keine Buchstaben eingibt, sondern lediglich Zahlen. Eine gesonderte Fehlerbehandlung sahen wir deswegen nicht als nötig an.
Worauf allerdings eine Fehlerbehandlung abzielt, ist das Überprüfen auf das Überschreiten des definierten Maximums. Wird dieses überschritten, erhält der Nutzer eine Fehlermeldung und kann die Eingaben korrigieren.
Da es genau eine Spalte mehr geben muss in der Matrix, als die Anzahl Unbekannter, welche vom Nutzer eingegeben wurde, muss die Variable, welche diese Anzahl beinhaltet, um genau eins erhöht werden.
Da in den nächsten Schritten eine Tabelle angelegt wird, in der die Matrix eingegeben werden soll, wird an dieser Stelle überprüft, ob diese Tabelle schon

exisitiert. Das wäre der Fall, wenn der Nutzer ein weiteres Mal eine Matrix erstellen wöllte. Falls dieser Fall gegeben sein sollte, würde diese Matrix zu erst entfernt werden.
Die Matrix soll über dem oben erwähnten Platzhalter eingefügt werden. Für die Erstellung der Matrix haben wir eine extra Funktion implementiert, auf diese wir jetzt näher eingehen.
Die Funktion haben wir **createTable()** genannt. Wir übergeben ihr zwei Parameter, die Anzahl der Zeilen (**row**) und die Anzahl der Spalten (**col**). Ein Rückgabetyp wird bei Javascript zwar nicht in der Deklaration mit angegeben, spielt aber beim Aufruf in der Funktion **start()** eine Rolle. Der Rückgabetyp der Funktion **createTable()** ist ein HTML-Konstrukt, das eine Form enthält.
Diese Form wird in der restlichen Funktion generiert. Zuerst wird diese erstellt und in eine lokale Variable gespeichert. Dafür verwenden wir die Methode **createElement()** von der Instanz **document**.
Um die Form später wieder überschreiben zu können, müssen ihr ein paar Attribute zugeteilt werden. Attribute teilt man über die Methode **createAttribute()** und **setAttributeNode()** zu. Der Form werden also die **id** und die **action** bei Betätigung zugewiesen. Die **id** ist dabei "reusableForm" und die **action** eine Verlinkung auf das cgi-Script, auf welches wir später eingehen.
Als nächstes wird der Eventhandler für die Betätigung der Form überschrieben. Die Funktion überprüft dabei alle Textfelder in der Matrix auf Leereingaben. Falscheingaben, wie beispielsweise Buchstaben werden hier wieder nicht beachtet, weil dies die Berechnung nicht beeinflusst.
In den restlichen Zeilen der Funktion **createTable()** werden die einzelnen Parameter des Aufrufes des cgi-Scriptes zusammengesetzt. Als erstes spielt dabei die Spalten- und Zeilenanzahl eine große Rolle. Diese werden als unsichtbare input-Felder der Form hinzugefügt. Input-Felder deswegen, weil der Inhalt dessen automatisch dem Script als Parameter übergeben wírd bei Betätigung der Form.
Danach wird eine Variable für die Tabelle angelegt. Zum Aufbau der Tabelle setzen wir auf zwei verschachtelte for-Schleifen. Diese gehen durch die Anzahl der Zeilen beziehungsweise Spalten hindurch. Für jede einzelne Zelle

wird dabei ein input-Feld erstellt, welches einen Namen zugeteilt bekommt, welcher jeweils aus der Zeile und Spalte besteht. Dies ist wichtig, um später aus dem Query-String, welcher die Parameter des cgi-Scriptes darstellt, die richtigen Werte rauszulesen. Außerdem bekommt jedes input-Feld eine id, welche ebenfalls aus Zeile und Spalte besteht. Diese ist wichtig, um Leeringaben abzufangen.
Handelt es sich um input-Felder, die vor dem "=" stehen, wird ein x mit der jeweiligen Spaltenzahl dahinter geschrieben, um die Unbekannten darzustellen.
Danach wird die Tabelle der Form als Bestandteil hinzugefügt mit der Methode **appendChild()**. Zum Schluss wird noch der Bestätigen-Button nach der Tabelle hinzugefügt, damit die Form abgeschickt werden kann. Beim Abschicken der Form wird automatisch die URL, welche als **action** hinterlegt ist, aufgerufen, wobei die Parameter in der Form
Parameter1=Wert1&Parameter2=Wert2&.. angehängt werden.

2.4 gauss.c

In der Datei **gauss.c** gehen wir nicht auf den Algorithmus von Gauss ein, da dieser gegeben war und von uns in keinster Weise angepasst wurde. Lediglich die Deklaration wurde dahingehend geändert, dass eine beliebige Anzahl von Gleichungen gelöst werden kann. Diese Anzahl entspricht dem Maximum aus der Datei **index.js** und muss manuell angepasst werden. Die Variable ist durch eine Preprozessordirektive am Anfang der Datei deklariert.
Die Datei muss während des Ablaufes eine HTML-Datei erzeugen, die der Browser direkt anzeigen kann. Dies wird dadurch erreicht, dass alle HTML-Ausgaben dementsprechend formatiert sind und je nach Ablauf über stdout ausgegeben werden.
Um die mitgelieferten Parameter auszulesen wird die Environment-Variable *QUERY_STRING* ausgelesen. Dies wird mit der Funktion **getenv()** durchgeführt.
Mittels der Funktion **strtok()** wird der Query-String soweit zerlegt, dass die Anzahl der Zeilen (**row**) und die Anzahl der Spalten (**col**) ausgelesen werden

können. Diese sind notwendig, um danach über eine for-Schleife die entsprechenden Werte der eingegeben Zahlen in ein Array zu speichern, welches der gegebenen Funktion übergeben werden kann.
Der restliche Teil wurde der mitgelieferten **main()**-Funktion entnommen und ruft die mitgelieferte Funktion für die Berechnung des Gauss-Algorithmus' auf. Die Ausgabe nach stdout wurde dabei so umformatiert, dass ein "\n" durch ein "
" ersetzt wurde. Zum Schluss wird die erzeugte HTML-Datei mit den entsprechenden Tags abgeschlossen.
Als Hilfsfunktionen haben wir zwei Funktionen geschrieben, die jeweils aus einem gegebenen String den Wert nach dem "=" als Integer beziehungsweise Float zurückgeben.

2.5 compile.sh

Die Datei **compile.sh** dient lediglich dazu, die bestehende C-Datei auf dem Server zu kompilieren. Dazu muss lediglich die URL (siehe oben) aufgerufen werden und es wird eine HTML-Seite generiert, welche einen Link auf die **index.html** enthält. Die Datei wurde in der Shell geschrieben.

3 Ausführung

Das Programm kann unter **www.benchr.de/IT2/compile.sh** kompiliert werden und unter **www.benchr.de/IT2/** ausgeführt werden. Für die langfristige Erreichbarkeit unter diesen Links wird keine Haftung übernommen. Das Hosting befindet sich momentan auf einem privaten Webserver von Benjamin Herzog.

4 Selbstständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet haben. Insbesondere versichern wir, dass wir alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht haben.