HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Institut für Informatik

Prof. Dr. Martin Mauve

Rechnernetze und Kommunikationssysteme

Grundlagen der Technischen Informatik SoSe 2015
C-Programmieraufgabe

1 Organisatorisches

Schauen Sie regelmäßig auf die Website zum C-Projekt [1].Dort werden eventuell kleine Änderungen der Aufgabenstellung oder zusätzliche Hilfestellungen veröffentlicht.

Implementieren Sie in **Einzelarbeit** ein Programm in der Programmiersprache C, das die folgende Problemstellung löst. Bitte wenden Sie sich bei Fragen zur Aufgabenstellung an den *Tutor Ihrer Gruppe* oder an die Mitarbeiter: *Andreas Disterhöft*[2] oder *Tobias Krauthoff*[3].

Plagiate führen zum Ausschluss von der Veranstaltung. Bedenken Sie, dass wir bei zwei identischen Abgaben nicht unterscheiden können, welche Abgabe das Original und welche das Plagiat ist. In diesem Fall führen beide Abgaben zum Ausschluss von der Veranstaltung.

Das Lösen der Programmieraufgabe ist entscheidend für die Zulassung zur Klausur. Falls Sie die theoretischen Übungen (voraussichtlich) nicht bestehen werden, so können Sie trotzdem beim C-Projekt teilnehmen. Die erbrachte Leistung beim C-Projekt wird auf Folgejahre übertragen und anerkannt. Jede abgegebene Lösung wird in einem kurzen Kolloquium besprochen.

Es gibt drei Abgabetermine, die darüber entscheiden, wann sie frühestens an einem Kolloquiumstermin teilnehmen können (siehe Tabelle). **Schicken Sie Ihre Abgabe früh genug ein, um sicher zu gehen, dass Sie auch eine Antwort erhalten und in allen Systemen ordentlich registriert sind.** An welcher Klausur Sie teilnehmen können hängt wiederum von Ihrem Kolloquiumstermin ab. Sie werden erst nach bestehen des Kolloquiums zur Klausur zugelassen, daher können Sie an einer Klausur nur teilnehmen, wenn Sie vorher Ihr Kolloquium bestanden haben.

Spätester Abgabetermin	Frühester Zeitraum für das Kolloquium	Möglicher Klausurtermin
03.07.2015	13.07.2015 bis 17.07.2015	Hauptklausur, Nachklausur
06.09.2015	14.09.2015 bis 23.09.2015	Hauptklausur, Nachklausur
09.10.2015	19.10.2015 bis 23.10.2015	Nachklausur

Abgaben nach dem **09.10.2015** können nicht mehr berücksichtigt werden (mit Ausnahme von krankheitsbedingten Gründen). Wenn Sie an der Hauptklausur teilnehmen wollen, muss Ihre Abgabe bis zum **06.09.2015** erfolgen und Ihr Kolloquium muss vor der Klausur erfolgreich abgeschlossen sein. Bitte beachten Sie für die Anmeldung zur Klausur unbedingt die Anmeldefristen des Prüfungsamtes. Sie können sich dort auch anmelden wenn Ihr Kolloquiumstermin noch nicht stattgefunden hat und

[1] http://www.cn.hhu.de/lehre-und-abschlussarbeiten/aktuelle-lehrveranstaltungen/vorlesungen/pflichtmodul-informatik-ii/c-projekt-informatik-ii.html

[2] Sprechstunde Freitags 10:00 - 11:00 in Raum 25.12.01.23, `info2@cs.uni-duesseldorf.de`

[3] Sprechstunde Dienstags 14:00 - 15:00 in Raum 25.12.02.45, `info2@cs.uni-duesseldorf.de`

sie daher die Zulassungsvoraussetzungen noch nicht komplett erfüllen. Sofern Sie Informatik studieren werden sie automatisch von der Klausur abgemeldet wenn Sie die Zulassung nicht erreichen. Ansonsten gelten die Regelungen Ihres Hauptfachs.

Wir empfehlen Ihnen eine erste Abgabe mindestens zwei Wochen vor dem endgültigen Termin, so dass wir Ihnen frühzeitig Rückmeldung geben können und Sie bis zum endgültigen Abgabetermin Ihr Programm gegebenenfalls noch einmal nachbessern können. Dies gilt insbesondere für den dritten Abgabetermin (09.10.2015). Geben Sie daher **spätestens bis zum 24.08.2015 eine erste Fassung** Ihres Programms ab, sofern Sie an der Hauptklausur teilnehmen möchten! Bei der Nachklausur verlängert sich die Frist auf den **28.09.2015**. Bei beiden Terminen ist es unbedingt notwendig, dass Sie den ersten Test bestehen (siehe Abschnitt 3)! Sie dürfen gerne auch jederzeit früher eine Version abgeben und sich Feedback einholen. Weiterhin sind auch weitere Abgaben möglich nachdem alle Tests bestanden wurden. Im Kolloquium wird die jeweils letzte Abgabe vor Abgabeschluss besprochen. Dies erlaubt auch nachträgliches Kommentieren oder Bugfixing.

Rechnen Sie damit, dass ein ungeübter C-Programmierer vier bis fünf Wochen Vollzeit zur Lösung der Aufgabenstellung benötigt. Fangen Sie rechtzeitig an! Lesen Sie sich dazu zunächst die komplette Aufgabenstellung durch und skizzieren Sie sich einen groben Plan zur Erstellung der Lösung, bevor Sie mit der Implementierung beginnen. Sie dürfen für Ihre Lösung ausschließlich die ANSI-C-Standardbibliothek C89/C90 (siehe Abschnitt 3) verwenden.

Für das erfolgreiche Bestehen der C-Programmieraufgabe müssen Sie zwei Kriterien erfüllen:

(a) Ihr Programm muss die gestellten Anforderungen erfüllen (insbesondere Abschnitt 3) und korrekt funktionieren. Wir überprüfen es anhand einiger Testdatensätze und automatisierter Tests z. B. auf Speicherlecks.

(b) Sie müssen uns im Kolloquium Ihr Programm und Ihre Algorithmen erläutern.

Das Bestehen der Onlinetests bedeutet nicht, dass Ihr Programm alle Kriterien erfüllt. Es wird anschließend noch von einem Mitarbeiter geprüft.

Zur Überprüfung Ihrer Lösung stellen wir voraussichtlich ab dem 18.05.2015 einige Tests bereit, die Sie offline durchführen können. Ihre Abgabe wird zusätzlich durch ein Online-Abgabesystem überprüft, welches ebenfalls ab dem 01.06.2015 auf der Website zum C-Projekt[4] zur Verfügung steht. Das System sendet Ihnen eine automatisch generierte Zusammenfassung aller online durchgeführten Tests zu. Eine Erklärung der Datei finden sie im Abschnitt 4.

Damit Sie Ihre Programm online überprüfen lassen können, senden Sie bitte Ihre ungepackte C-Quellcode-Datei mit dem Namen **boxpack.c** und einem Token an:

Adresse: **info2-c@cs.uni-duesseldorf.de**
Betreff: "Abgabe,*Mein-Vorname*,*Mein-Nachname*,*Mein-Token*".

Das Token senden wir Ihnen rechtzeitig an der von Ihnen im Abgabesystem hinterlegten E-Mail-Adresse zu. Zum Beispiel wäre folgender Betreff gültig: *Abgabe,Max,Mustermann,M9LPt94Mzp* (keine Leerzeichen!), wobei `M9LPt94Mzp` das Token ist.

Abgaben mit anderem Betreff oder von einer anderen Absender E-Mail-Adresse als der im Abgabesystem am Anfang des Semesters von Ihnen verwendeten werden nicht akzeptiert! Insbesondere ist "@hhu.de" nicht das gleiche wie "@uni-duesseldorf.de". Achten Sie auf die Einstellungen in Ihrem (Web-)Mailprogramm und Ihren Logindaten im Abgabesystem.

[4] http://www.cn.hhu.de/lehre-und-abschlussarbeiten/aktuelle-lehrveranstaltungen/vorlesungen/pflichtmodul-informatik-ii/c-projekt-informatik-ii.html

2 Aufgabenstellung

Das optimale Verstauen von Gegenständen in mehreren Taschen ist oft im Alltag erforderlich, aber auch in der Informatik sehr beliebt. Das Problem wird in den verschiedensten Disziplinen der Informatik und Mathematik erforscht und soll in in dieser C-Programmieraufgabe in einer vereinfachten Form implementiert und gelöst werden.

Stellen Sie sich Wall-E[5] vor. Nach Jahrzehnten mühseliger Aufräumarbeit ist Wall-E immer noch damit beschäftigt die Erde aufzuräumen. Dies liegt vor allem daran, dass er ineffizient arbeitet, da er den zur Verfügung stehenden Raum mangels Überblick nicht optimal nutzen kann. Eine Gruppe von erfahrenen Aufräumspezialisten und Sie als Informatiker/-in erhalten die ehrenhafte Möglichkeit Wall-E zu helfen, sodass die Erde, der einzige Planet auf dem es Schokolade gibt, nach Jahrzehnten wieder bewohnbar ist.

Dank der hervorragenden Vorarbeit der Aufräumspezialisten, die fleißig Satellitenbilder auswerteten, steht Ihnen eine Liste bereit, die die Reihenfolge und *Fit*-Strategien beinhalten, um den Müll in unterschiedlich große Müllcontainer zu werfen. Hierbei ist Vorsicht geboten, denn die Müllcontainer dürfen nicht überfüllt werden, da dies wiederum zu einer Verschmutzung der Erde führt. Da die Satellitenbilder teilweise von schlechter Qualität sind, wurden Sie als Informatiker/-in beauftragt diese Listen zu validieren und werden im Gegenzug mit der Klausurzulassung belohnt[6]. Ihre Aufgabe ist es zu überprüfen, ob die Verteilung der Müllboxen auf die zur Verfügung stehende Müllcontainer möglich ist ohne diese zu überfüllen. Die zu validierende Listen enthalten die Kapazitäten der Müllcontainer, die Größe der Müllboxen und deren Einsortierungsstrategie. Diese Strategien geben an in welche Container die Boxen abgelegt werden sollen. So ist “Best Fit” eine mögliche Strategie, die Boxen in Container einsortiert, in denen nach der Einsortierung am wenigsten Platz übrig bleibt. Ist die Validierungsarbeit von Ihnen getan, so kann Wall-E mithilfe des validen Plans die Müllboxen in die dazugehörigen Müllcontainer einsortieren, sodass eine baldige Heimkehr auf die Erde möglich ist.

2.1 Aufruf

Ihr Programm soll sich mit dem Aufruf `boxpack input output` starten lassen, wobei `input` die Eingabe- und `output` die Ausgabedatei beschreibt. Der exakte Aufbau der input- und output Datei, welcher einzuhalten gilt, ist in Abschnitt 2.2 beschrieben.

2.2 Ein- und Ausgabeformate

2.2.1 Eingabe

Die Eingabedatei besteht aus genau zwei Zeilen, welche mit einem newline-Character (“
n”) getrennt werden. Wichtig: Am Ende der letzten Zeile jeder Datei ist daher kein newline-Character vorzufinden. Die erste Zeile beinhaltet die maximale Kapazität der verwendeten (Müll-)Container. Die zweite Zeile enthält die verschiedenen Fit-Verfahren (= Einsortierungsstrategien) und die zu verstauenden (Müll-)Boxen unter Angabe der Größe. Die Fit-Verfahren geben an nach welcher Strategie nachfolgend gelistete Boxen in die Container einsortiert werden müssen. Somit ist das Aufkommen eines Fit-Verfahrens als Präfix zu verstehen und gilt für alle darauf folgenden Boxen bis ein anderes Fit-Verfahren dieses ablöst. Sollte zu Beginn der zweiten Zeile kein Fit-Verfahren angegeben werden, so ist das First-Fit-Verfahren zu wählen. Jegliche Information innerhalb einer Zeile wird durch ein Leerzeichen separiert.

[5] Weitere Informationen finden Sie unter: `http://filme.disney.de/wall-e`.

[6] Vorausgesetzt alle weiteren Anforderungen und formalen Kriterien zur Zulassung sind erfüllt.

Die Fit-Verfahren sind wie folgt kodiert:

- bf $\Leftrightarrow$ best-fit
- nf $\Leftrightarrow$ next-fit
- ff $\Leftrightarrow$ first-fit
- awf $\Leftrightarrow$ almost-worst-fit

Eine genaue Erklärung der Verfahren finden Sie in Abschnitt 2.4. Ein Beispiel-Input könnte wie folgt aussehen:

```
10 21 42 913 481 11 25 103
11 87 20 bf 90 nf 18 89 2 bf 1 nf 91 34 awf 134 1
```

Die Eingabe ist wie folgt zu verstehen: Der erste Container hat die Kapazität von 10, der Zweite 21, der Dritte 42 usw. Insbesondere ist die Reihenfolge zu beachten. Die erste Box aus Zeile 11 wird per first-fit in den entsprechenden Container aus der ersten Zeile eingeordnet. Box zwei und drei werden ebenso per first-fit einsortiert. Die vierte Box mit der Größe 90 wird per best-fit in den entsprechenden Container einsortiert, da zuvor auf das best-fit Verfahren (Schlüsselwort: bf) gewechselt wurde. Die nächsten drei Boxen (Größen: 18, 89 und 2) werden per next-fit einsortiert usw.

Achtung: Beachten Sie die Groß- und Kleinschreibung.

2.2.2 Ausgabe

Die Ausgabedatei, welche im zweiten Argument des Programmaufrufs spezifiziert ist, ist wie folgt zu formatieren. Zeilenweise sind die Indizes der (Müll-)Container, nummeriert beginnend mit der 0 in der selben Reihenfolge wie in der Eingabedatei, mit deren Inhalt aufzuführen. Der Index eines Containers wird mit einem Doppelpunkt (:) durch seinen Inhalt getrennt. Der Inhalt eines Containers ist durch eine durch Leerzeichen getrennte Aufzählung der (Müll-)Boxengröße aufzuzeigen. Zusammenfassend folgen die Zeilen der Ausgabedatei das folgende Muster:

```
<Containerindex>: <Boxgröße_1> <Boxgröße_2> ... <Boxgröße_n>\n.
```

Dabei steht die Escape-Sequenz `\n` in C für einen Zeilenumbruch. Achten Sie auf die Leerzeichen nach dem Doppelpunkt und zwischen zwei Boxgrößen. Am Zeilenende darf kein Leerzeichen vorhanden sein. Die Ausgabe der in Abschnitt 2.2.1 erläuterten Eingabe sähe wie folgt aus:

```
0: 0
1: 11
2: 20 18 1
3: 87 89 2 91 34
4: 134 1
5: 0
6: 0
7: 90
```

Achtung: Die Ausgabedatei soll am Ende(!) des Programms beschrieben werden, sodass für benötigte Ressourcen dynamischer Speicher mittels `malloc/free` alloziert bzw. freigegeben werden soll. Mehr hierzu erfahren Sie in Abschnitt 2.3. Außerdem soll diese Datei nur geschrieben werden, sofern die Korrektheit der Eingabedatei gegeben ist!

2.3 Programmablauf

Zur strukturieren Lösung der Aufgabe, wird im folgenden der Programmablauf exemplarisch aufgezeichnet.

1. Lesen Sie zunächst die Eingabedatei ein, deren Struktur in Abschnitt 2.2.2 erläutert ist. Achten Sie darauf, dass der Input passend gespeichert wird.
2. Anschließend können Sie den Input verarbeiten. Dabei ist es wichtig, Formatierungsfehler zu finden und passend damit umzugehen. Die Fehlerbehandlung wird in Abschnitt 2.5 thematisiert.
3. Während Sie die Eingaben verarbeiten, können Sie schrittweise Operationen durchführen. Ausgewählte Fit-Strategien, die in einer gültigen Eingabe vorkommen können, sind in Abschnitt 2.4 aufgezählt und definiert. Machen Sie sich vorher unbedingt klar welche Datenstrukturen Sie verwenden werden und wie der Aufbau der Strukturen aussehen soll. Achten Sie hier ausdrücklich auf eine dynamische Speicherverwaltung mittels `malloc/free`.
4. Es wird von Ihnen verlangt, dass Sie die Addition und Subtraktion zweier Integer als Methode auslagern und die Operationen als Inline-Assembler lösen. Verwenden Sie dafür ausschließlich eine der beiden folgenden Definition der Methodenköpfe:

   - Variante A:
     uint32_t inlineAddition(uint32_t, uint32_t);
     uint32_t inlineSubtraction(uint32_t, uint32_t);
   - Variante B:
     int inlineAddition(int, int);
     int inlineSubtraction(int, int);

   Die Methoden sollen zwei Werte entgegennehmen, diese per Inline-Assembler addieren/subtrahieren und dann das Ergebnis als return-Wert zurückliefern.
5. Nachdem die Eingabe verarbeitet wurde, sollen Sie nun Ihre Datenstruktur, wie in Abschnitt 2.2.2 gefordert, ausgeben.
6. Vergessen Sie nicht den verwendeten Speicher wieder freizugeben. Zur Überprüfung ist die Nutzung von *valgrind* überaus hilfreich.

2.4 Fit-Verfahren

Alle von Ihnen zu implementierenden Fit-Verfahren werden nachfolgend aufgezeigt.

Tie-Breaker Regel: Kann eine Box nach dem gewählten Fit-Verfahren in mehr als einen Container einsortiert werden, so ist immer der Container mit dem kleinsten Index zu wählen.

First-Fit Im First-Fit Verfahren wird die Box in den ersten Container, beginnend bei Index 0, eingeordnet, der genügend Platz bietet.

Best-Fit Boxen werden mit der Best-Fit Strategie in den Container einsortiert, der nach dem Einfügen am wenigsten Platz hat.

Next-Fit Das Next-Fit Verfahren ordnet Boxen ausgehend von dem zuletzt verwendeten Container (hier: aktiver Container) ein. Falls solch ein Container nicht vorhanden ist, so ist dies der Container mit dem Index `0`. Es werden so lange Boxen in den aktiven Container eingeordnet bis die einzufügende Box nicht mehr reinpasst. In diesem Fall wird der Container mit dem Index `(Index + 1) modulo n`, wobei n für die Anzahl an Containern steht, als aktiv markiert und nach dem gleichen Schema befüllt.

Almost-Worst-Fit Mit aktiviertem Almost-Worst-Fit Verfahren werden Boxen in den Container einsortiert, der am zweitmeisten Platz zur Verfügung hat. Ist dieser Container zu klein für die einzufügende Box, so ist stattdessen der Container zu wählen, der am meisten Platz zur Verfügung hat.

2.5 Fehlerbehandlung

Ihre Implementierung soll mit Fehlern während der Laufzeit geeignet umgehen. Ist die Einsortierung einer Box in den vorhergesehenen Container mangels Platz nicht möglich so soll `validation failed` mit anschließendem Zeilenumbruch nach `stdout` geschrieben werden. Bei einem Formatierungsfehler in der Eingabedatei ist dieser im Format `Error: <Beschreibung>` mit einer angemessenen Beschreibung und anschließendem Zeilenumbruch nach `stderr` zu schreiben. Achten Sie hier auf die genaue Formatierung.

2.6 Beispiel

Im Folgenden finden Sie zwei Beispiele, die die gewünschte Funktionsweise des Programms erläutern. Hierbei ist jeweils die Eingabe und deren Ausgabe abgebildet. Zum besseren Verständnis der Aufgabe ist das Nachvollziehen dieser Beispiele ratsam.

```
Eingabe:
10 10 10
bf 6 ff 8 awf 1 nf 5

Ausgabe:
0: 6 1
1: 8
2: 5
```

```
Eingabe:
10 21 42 230 140 11 103
11 87 20 bf 90 nf 18 89 2 bf 1 nf 91 34 awf 50 1 nf 4 9 awf 5 bf 7 nf 2 awf 5

Ausgabe:
0: 9
1: 11 2 5
2: 20 18 1
3: 87 89 2 50
4: 91 34
5: 5
6: 90 1 4 7
```

3 Wichtige Anforderungen

Beachten Sie bei der Erstellung Ihres Programms **unbedingt** folgende Punkte:

1. Schreiben Sie Ihr Programm ausschließlich in der Programmiersprache C (und nicht in C++).
2. Ihr Programm muss unter Linux x86 lauffähig und compilierbar sein.
3. Ihr Programm muss auf der Kommandozeile ohne grafische Oberfläche lauffähig sein.
4. Überprüfen Sie die Funktionalität Ihres Programms (alle Operationen müssen fehlerfrei unterstützt werden).
5. Kommentieren Sie Ihren Quelltext in angemessenem Umfang.
6. Die erste Zeile Ihres Quelltextes muss ein Kommentar sein, der Ihren Vor- und Nachnamen in der gleichen Schreibweise wie im Betreff Ihrer E-Mail enthält.
7. Halten Sie sich strikt an das oben definierte Ein- und Ausgabeformat und die Aufrufkonventionen. Wir überprüfen Ihr Programm halbautomatisch auf die Korrektheit der ausgegebenen Lösungen. *Eine falsch formatierte Ausgabe wird nicht als korrekt anerkannt!* Im Zweifelsfall fragen Sie *rechtzeitig vor Abgabe* nach!
8. Das Programm soll sich nach Abarbeitung der Operation sofort beenden.
9. Ihr Programm soll auf Fehler während der Laufzeit angemessen reagieren. Mehr Informationen hierzu finden Sie in Abschnitt 2.5.
10. Sie dürfen folgende ANSI-C-Standardbibliotheken C89/C90 nutzen: `assert.h`, `ctype.h`, `errno.h`, `float.h`, `locale.h`, `math.h`, `setjmp.h`, `signal.h`, `stdarg.h`, `stddef.h`, `stdio.h`, `stdlib.h`, `string.h`.
    Achtung: Weitere Bibliotheken dürfen nicht verwendet werden.
11. Achten Sie darauf, dass Ihr Programm auf x86 Linux-Systemen lauffähig ist.
12. Benutzen Sie für Inline-Assembler ausschließlich den in der Vorlesung vorgestellten x86/IA-32-Assembler-Befehlssatz. Es ist vorgesehen, dass Sie die Addition und die Subtraktion zweier Zahlen als Methode auslagern und diese per Inline-Assembler lösen. Das Ergebnis soll als Rückgabewert zurückgegeben werden. Achten Sie auf Ihre Definition der Methodenköpfe.
13. **Verwenden Sie eine einzige C-Quelltextdatei mit dem Namen boxpack.c**. Ihr Programm muss sich mit einem einzelnen GCC-Compileraufruf fehlerfrei übersetzen, linken und compilieren lassen. Benutzen Sie dazu auch die Parameter `-Wall -Werror -Wshadow`, hiermit führen alle Compilerwarnungen automatisch zum Abbruch des Compilevorgangs. Die Verwendung einer grafischen Entwicklungsumgebung wie z. B. Visual Studio, Qt, Netbeans oder Eclipse kann Ihnen möglicherweise die Programmierung und vor allem das Debugging erleichtern. Sie müssen aber verstanden haben, wie die Kompilierung durch direktes Aufrufen des Compilers auf der Konsole erfolgt.
14. Legen Sie Ihr Programm so aus, dass es mit beliebig großen Eingabedaten umgehen kann. Setzen Sie hierfür unbedingt dynamische Speicherverwaltung mit `malloc/free` ein. Die Verwendung von `realloc` ist nicht vorgesehen. Wir werden Ihr Programm mit *großen* Datensätzen testen! Das bedeutet, dass (beliebig) lange Ein- und Ausgabedateien auftreten werden.

15. Ihr abgegebenes Programm wird automatisch auf Speicherlecks überprüft. Stellen Sie sicher, dass es keine Speicherlecks aufweist. Geben Sie jeglichen dynamisch angeforderten Speicher vor der Terminierung des Programms korrekt wieder frei. Für Ihre Suche nach Speicherlecks eignet sich das Programm *valgrind*[2]. Nützliche Parameter sind `--leak-check=full`, `--track-origins=yes` und `--show-reachable=yes`. Dies schließt nicht aus, das weitere Paramter sinnvoll sind.

4 Erklärungen zur Rückgabe des Onlinetesters

Nachdem Sie Ihr Programm per E-Mail abgesendet haben, kann es einige Zeit dauern, bis Sie eine Zusammenfassung der Online-Tests erhalten. Der Online-Checker kontrolliert nur zu bestimmten Zeiten eingegangene E-Mails. Der Zeitraum zwischen der Absendung der E-Mail Ihrerseits und dem Empfang der Testresultate kann je nach Anzahl eingehender Abgaben variieren, sodass Sie vor allem einige Tage vor Abgabeschluss mit größeren Wartezeiten rechnen müssen.
Sofern Sie dann Ihre Antwort erhalten haben, finden Sie im Anhang der E-Mail eine HTML-Datei. Im Text finden Sie die Information, ob Ihr Programm fehlerfrei getestet wurde und im Anhang befindet sich eine ausführliche Version als HTML-Datei mit dem Namen `results.html`. Die generierte HTML-Seite beinhaltet JavaScript und kann daher in einigen Mailprogrammen und/oder Webmail-Diensten nicht korrekt angezeigt werden. Laden Sie bitte dazu die Datei herunter und öffnen diese lokal auf Ihrem Rechner.

Im ersten Abschnitt `Summary` erhalten Sie eine kurze Zusammenfassung mit Informationen zu Ihrer Abgabe, dem Ergebnis (“failed” oder “succeed”) und einer Aufzählung von Errorcodes, die Ihnen den Grund für fehlgeschlagene Tests liefern.
Der zweite Abschnitt `Test Results` enthält die Ergebnisse zur Prüfung Ihres Programms. Das Programm durchläuft einige grundlegende Tests (Compiliervorgang, Test des Quellcode-Kopfs etc.), von uns generierte Testdatensätze, valgrind Tests und schließlich die Validierung der Nutzung von Inline-Assembler. Jedes der Testergebnisse lässt sich durch Klicken auf `details` ausklappen, wodurch die Test-Eingabe, die geforderte Ausgabe, die Ausgabe Ihres Programms und ggf. der Errorcode zum Vorschein kommt.
Im dritten und letzten Abschnitt `Code` ist der Quellcode Ihres Programms gelistet, welches als Grundlage für die generierten Testergebnisse verwendet wurde.

[2] http://www.valgrind.org