Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WiSe 2023/2024
Fakultät für Angewandte Informatik
Institut für Informatik
Prof. Dr. J. Hähner, P. Schalk, V. Schmid, S. Lutz, J. Linne 06.12.2023

Semesteraufgabe Informatik 1

Abgabe spätestens bis: 31.01.2024

Allgemeine Aufgabenstellung

Sie können als freiwillige Zusatzaufgabe eine der unten stehenden Systembeschreibungen implementieren. Die Aufgabe kann durch die in Vips hinterlegten Teams bearbeitet werden, es dürfen für die Semesteraufgabe aber auch neue Teams aus maximal 3 Personen gebildet werden. Sie erhalten für die vollständige und korrekte Bearbeitung bis zu 5 Übungspunkte. Die Bewertung erfolgt durch eine Abnahme. Die Teams mit dem besten Programm zu einer Aufgabe erhalten eine Urkunde. Eine Teilnahme mit gutem Erfolg ist empfohlen, wenn man Tutor für Informatik 1 werden möchte.

Abgabe

Die Abgabe muss in folgender Form erfolgen:

- Upload der Materialien, verpackt in einem .zip-Archiv, in den Unterordner „Semesteraufgabe“ → „Abgabe“ in der Digicampus-Veranstaltung „Informatik 1“.
- Die Upload-Materialien müssen enthalten:
  - den vollständigen Quellcode
  - Hinweise zur Ausführung und Benutzung des Programms
  - vollständige Auflistung der Personen, die an der Aufgabe gearbeitet haben
  - Angabe von zeitlichen Überschneidungen am Tag der Abnahme am **12.02.2024**
  - Bitte **keine** ausführbare .exe-Datei, da die Abgabe per E-Mail weitergeleitet wird und E-Mail-Filter darauf empfindlich reagieren. Schreiben Sie stattdessen in eine Text-Datei, wie das Programm richtig kompiliert und ausgeführt wird.

Allgemeine Hinweise

- Jedes Team bewertet die Lösung eines anderen Teams zu einer anderen Aufgabe:
  - Die Lösung des anderen Teams wird vom Lehrstuhl per Mail am **01.02.2024** zugeschickt.
  - Die Bewertung muss bis **07.02.2024** in den Unterordner „Semesteraufgabe“ → „Bewertung“ der Digicampus-Veranstaltung „Informatik 1“ hochgeladen werden (als PDF-Datei).
- Abnahme am **12.02.2024**:
  - Bei der Abnahme müssen **alle** Teammitglieder anwesend sein.
  - Die Teammitglieder führen ihr Programm vor und beantworten Fragen zur Funktionalität und zur Programmierung.
  - Die Abnahme erfolgt in Präsenz, der Raum wird noch bekannt gegeben.
- In der Vorlesung wurden noch nicht alle notwendigen Anforderungen besprochen. Sie sollten bereits jetzt mit der Programmierung der bereits behandelten oder aus dem Vorkurs bekannten Konzepte beginnen, und diese dann nach und nach um die weiteren Anforderungen ergänzen und verbessern, sobald diese in der Vorlesung besprochen wurden.

Technische Mindestanforderungen

- Das Programm muss in der Programmiersprache C erstellt worden sein, muss sich ohne Fehler und Warnungen mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` compilieren lassen, und muss fehlerfrei ausführbar sein.
- Es dürfen bei der Implementierung keine anderen Bibliotheken als die C89-Standardbibliothek benutzt werden.
- Der Code muss sich im Wesentlichen an den Vorgaben der Vorlesung orientieren, zum Beispiel:
  - Programmier-Konventionen
  - Ausführliche Fehlerbehandlung von Funktionsaufrufen
  - Sinnvolle Aufteilung des Quellcodes auf verschiedene Übersetzungseinheiten
  - Sinnvolle Verwendung von lokalen, statischen und globalen Variablen
  - Sinnvolle Verwendung von symbolischen Konstanten und Makros
  - Vermeidung von Speicherlecks
- Vorgaben für die textuelle Benutzerschnittstelle:
  - Ausführliche Benutzerführung (Informationen über die möglichen Eingaben, verständliche Status- und Fehlermeldungen)
  - Übersichtlich formatierte Ausgaben
- Sie müssen alle Implementierungsentscheidungen begründen können.
- Die Anforderungen der spezifischen Aufgabenstellung müssen erfüllt sein.
- Der Code muss eigenständig erstellt worden sein.

Bewertung

- Die Bewertung erfolgt durch den Lehrstuhl auf Grundlage der Abnahme.
- Die Abnahme besteht aus einer 10-minütigen Präsentation des Programms (und zwar der Anwendung, nicht des Quellcodes) durch alle Teammitglieder, mit anschließender 5-minütiger Diskussion. Jedes Teammitglied muss einen Teil der Präsentation übernehmen.
- Erfüllt das Programm alle Mindestanforderungen, so erhält man 2 Übungspunkte.
- Für besondere Aspekte der Umsetzung, die über die Mindestanforderungen hinaus gehen, kann man bis zu 2 weitere Übungspunkte erhalten, z.B. für
  - besonders schöne und durchdachte textuelle Benutzerschnittstelle mit Benutzerführung.
  - zusätzliche sinnvolle Funktionalitäten, die deutlich über die Anforderungen der spezifischen Aufgabenstellung hinaus gehen.
  - Berücksichtigung zusätzlicher sinnvoller Daten, die nicht in der spezifischen Aufgabenstellung erwähnt werden und einen erkennbaren Mehrwert für die Anwendung bedeuten.
  - besonders gelungene Strukturierung des Quellcodes für eine bessere Wartung und Erweiterbarkeit des Programms.

  Solche besonderen Aspekte müssen in den Hinweisen zur Ausführung und Benutzung und in der Abnahme deutlich gemacht werden.
- Die Bearbeitung einer der freiwilligen Zusatzaufgaben bringt zusätzlich maximal 1 Übungspunkt.

Systembeschreibung A: Minesweeper

Es soll das Spiel Minesweeper[1] implementiert werden, das ausschließlich auf der Kommandozeile gespielt wird. Dabei sollen die folgenden Aspekte berücksichtigt werden:

- Die Größe des Spielfeldes soll mit Hilfe von Kommandozeilenparametern bestimmt werden. Gibt der Benutzer keine Kommandozeilenparameter für die Größe des Spielfeldes an, soll eine sinnvolle Standardgröße für das Feld genutzt werden.
- Das Feld soll übersichtlich auf Kommandozeile ausgegeben werden. Dabei soll deutlich erkennbar sein, welche Positionen im Feld bereits aufgedeckt sind und welche nicht.
- Mithilfe von Benutzereingaben soll es dem Benutzer möglich sein, ein Feld aufzudecken, mit einer Flagge zu markieren oder eine solche Markierung von einem Feld zu entfernen.
- Mit Flaggen markierte Felder sollen nicht aufgedeckt werden können.
- Am Ende des Spiels soll dem Benutzer angezeigt werden, wie viele Sekunden verstrichen sind und wie viele Befehle er gebraucht hat.

Freiwillige Zusatzaufgaben
Die Zeit und die Anzahl der Befehle soll in einer Textdatei gespeichert werden. Der Benutzer soll die Möglichkeit erhalten, sich anhand dieser Daten eine Bestenliste anzeigen zu lassen.

Systembeschreibung B: Gleitkommacodierung

Es soll eine Anwendung für die Codierung und Decodierung von reellen Zahlen bzgl. der Gleitkomma-Codierung mit und ohne Darstellung der zugehörigen Rechnung erstellt werden. Dafür sollen folgende Codierungen verwendet werden können:
$c_{GK,5,8}$, $c_{GK,11,16}$, $c_{GK,24,32}$
Dabei soll die Anwendung mit der Eingabe von reellen Zahlen sowohl als Bitmuster als auch als Dezimalzahl in Fest- und Gleitkomma-Schreibweise umgehen können. Außerdem soll die Addition und Subtraktion von zwei reellen Zahlen dargestellt werden können.

Freiwillige Zusatzaufgaben

- Interaktive Schritt-für-Schritt-Ausführung von Rechnungen und (De-)Codierungen.
- Darstellung von Multiplikation von zwei reellen Zahlen in Bitmuster-Schreibweise

Systembeschreibung C: Kryptographie

Es soll ein Programm erstellt werden, das in der Lage ist, Texte zu verschlüsseln und diese wieder zu entschlüsseln. Dafür sollen die folgenden Arten der Verschlüsselung zur Verfügung stehen:

- Gartenzaun-Methode: https://de.wikipedia.org/wiki/Transposition_(Kryptographie)
- Cäsar-Chiffre: https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung
- Vigenère-Chiffre: https://de.wikipedia.org/wiki/Vigen%C3%A8re-Chiffre

Außerdem soll es mit Hilfe des Programms auch möglich sein, verschlüsselte Texte zu entschlüsseln, auch wenn man nicht über den benötigten Schlüssel verfügt. Hierzu können Brute-Force-Verfahren (wobei z.B. nach jedem Versuch der Benutzer gefragt werden kann, ob der Anfang einer entschlüsselten Nachricht Sinn ergibt) oder Häufigkeitsanalysen[2] verwendet werden.

Freiwillige Zusatzaufgabe
Implementierung weiterer Verschlüsselungs-Verfahren.

[1] https://de.wikipedia.org/wiki/Minesweeper
[2] https://de.wikipedia.org/wiki/H%C3%A4ufigkeitsanalyse