Dezember 2021

Praktikum zur Lehrveranstaltung Mikrorechentechnik I
Versuch 3, C-Programmierung

Organisation des Praktikumsversuchs: M. Herhold

Inhaltsverzeichnis

1 Ziele dieses Praktikumsversuchs

- Festigung des Vorlesungsstoffes zur C-Programmierung
  - durch Benutzung von Arrays und Speicherbereichen in C
  - durch den Austausch von Daten innerhalb eines C-Programms mittels Zeiger und Strukturen
  - durch aufteilen von Funktionalität eines C-Programms auf mehrere C-Module
- Eigenständiges Entwerfen und Implementieren eines Programms in der Programmiersprache C.
- Schulung der Teamfähigkeit.

2 Versuchsaufgabe

Programmieren Sie unter Zuhilfenahme der Programmiersprache C ein Computerprogramm, welches eine grafische Animation, nach vorgegebenen Regeln generiert und mit Hilfe der Bibliothek SDL2 auf dem Bildschirm ausgibt. Benutzen Sie dazu die Funktionen des vorgegebenen Quelltextes und ergänzen Sie diesen Quelltext um die notwendige Funktionalität. Arbeiten Sie dazu im Team und verteilen Sie selbstständig Teilaufgaben unter den einzelnen Gruppenmitgliedern.

2.1 Animationsbeschreibung

Bei der Animationsfläche handelt es sich um ein Gitter mit `x` Pixeln horizontal und `y` Pixeln vertikal. Die Pixel-Koordinate (`x=0`, `y=0`) befindet sich links oben. Jedes Pixel stellt eine Einheit dar und kann die Werte `frei` oder `belegt` annehmen.

Der Folgezustand eines Pixels (zum Zeitpunkt t_{n+1}) ergibt sich aus den derzeitigen Zuständen (Zeitpunkt t_n) seiner 8 Nachbarpixel.

? ? ?
? ! ?
? ? ?

Zur Berechnung eines Animationsschrittes sind eine Reihe von Regeln – beschrieben in Abschnitt 3.3 – für jeden einzelnen Pixel anzuwenden.

3 Teilaufgaben

Folgende Teilaufgaben sind zu bearbeiten um die Praktikumsaufgabe zu lösen.

3.1 Datenstruktur- und Funktionsprototypen definieren

Dies ist eine Teilaufgabe, welche zuallererst in Angriff genommen werden und an der alle Teammitglieder beteiligt werden sollten.

Diese Teilaufgabe soll als Ergebnis eine geeignete Datenstruktur zum Austausch wichtiger Laufzeitdaten zwischen den einzelnen C-Modulen, sowie erste Schnittstellen-Prototypen der wichtigsten Funktionen der einzelnen C-Module, haben.

Folgende Daten sollen in der Datenstruktur `Laufzeit-Daten` auf jeden Fall enthalten sein.:

- die Anzahl der Pixel der Animationsfläche in Richtung der X-Achse (`Spalten`)
- die Anzahl der Pixel der Animationsfläche in Richtung der Y-Achse (`Zeilen`)
- ein Zähler, welcher den derzeitigen Animationsschritt speichert (`Schritt`)
- die maximale Anzahl an Animationsschritten die berechnet und angezeigt werden sollen (`Schritte`)
- ein Wert, welcher die Pause zwischen zwei Animationsschritten in Millisekunden angibt (Delay)
- ein ein- oder zwei-dimensionales C-Array, im Folgenden als `Animations-Puffer` bezeichnet, welches den letzten berechneten Animationsschritt speichert. Dieses Array benötigt Platz für $(Zeilen + 2) * (Spalten + 2)$ Elemente – siehe 3.3 Pkt 5.

Möglicherweise müssen Sie weitere Daten in der Struktur ablegen.

Hinweis: Im Quellcode des Programms und den Schnittstellen der einzelnen Funktionen, ist es sinnvoll mit einem Zeiger auf eine Instanz dieser Datenstruktur zu arbeiten um unnötige Speicherkopieroperationen zu vermeiden.

3.2 Einlesen einer Konfigurationsdatei

Diese Teilaufgabe benötigt mittleren programmiertechnischen Aufwand, fordert aber Wissen zu Speicherallokation und C-Strings, und kann von einem einzelnen Teammitglied:in[1] bearbeitet werden.

Die Funktionalität der Teilaufgabe soll im C-Modul `config.c` implementiert werden.

Zweck der Teilaufgabe ist das Befüllen der zuvor ausgearbeiteten Datenstruktur `Laufzeit-Daten` mit, aus einer Datei eingelesenen, Werten.

Eine Beispielkonfigurationsdatei – `settings-1.txt` – befindet sich im Projektverzeichnis. Implementieren Sie die Funktionalität anhand dieser Datei. Während des Praktikumsversuchs, soll Ihr Programm in der Lage sein, eine andere Datei einzulesen.

Zeigen Sie Fehler beim Einlesen der Datei über den Standardfehlerausgabe-Stream (`stderr`) dem Nutzer an. Nutzen Sie perror und fprintf, wenn die Datei nicht geöffnet werden konnte, oder ein Parameter nicht im geforderten Datentyp vorliegt.

[1] scnr

Inhalt der Datei settings-1.txt:

```
Zeilen: 100
Spalten: 100
Schritt: 0
Schritte: 15
Pause: 0.1 s

Animations-Puffer:
..x....x..
xx.xxxx.xx
..x....x..
```

Folgendes ist beim Auslesen der Datei zu beachten:

- Die Reihenfolge der Zeilen kann variieren. Sie müssen also eine Zeile identifizieren, bevor Sie wissen, welche Einstellung gerade eingelesen wird.
- Die Animations-Puffer Daten kommen immer am Ende der Datei.
- Wenn eine Zeile mehrfach, aber mit unterschiedlichen Parameterwert vorkommt, dann ist nur das letzte Vorkommen der Zeile in der Datei relevant (also Werte weiter unten in der Datei überschreiben vorherige Werte)
- Die Datei kann leere Zeilen enthalten.
- Die Animations-Puffer Daten sind folgendermaßen zu interpretieren:
  - . (Punkt) beschreibt einen freien Pixel
  - x beschreibt einen belegten Pixel
- Wenn die Animations-Puffer Daten der Datei den Animations-Puffer nicht ausfüllen, dann sind die ausgelesenen Daten mittig im Animations-Puffer der Laufzeit-Daten zu platzieren. Nicht in der Datei beschriebene Pixel sind mit dem Wert frei zu initialisieren.

Folgende Funktionen aus der C-Standard-Bibliothek sind für die Lösung der Teilaufgabe hilfreich: fopen(), fclose(), getline(), fscanf(), sscanf(), strncmp(), atoi(), atof(), malloc(), calloc(), memset() und perror().

3.2.1 freiwillige Zusatzaufgabe

Erstellen Sie eine settings-Datei und Quellcode, welche Ihr Programm veranlasst den Animations-Puffer mit zufälligen Werten zu füllen.

3.3 Berechnen des nächsten Animationsschrittes

Diese Teilaufgabe ist die anspruchsvollste Teilaufgabe und kann von einem einzelnen Teammitglied bearbeitet werden.

Die Funktionalität dieser Aufgabe soll im C-Modul engine.c implementiert werden.

Ziel der Teilaufgabe ist die pixelweise Berechnung und Speicherung des nachfolgenden Animationsschrittes (t_{n+1}).

Zur korrekten Berechnung des jeweils nächsten Animationsschrittes (t_{n+1}) muss der aktuelle Zustand (t_n) betrachtet und folgende Regeln eingehalten werden:

1. Ein freier Pixel mit exakt 3 belegten Nachbarpixeln wird zu einem belegten Pixel.

   Beispiele:
   ```
   . . .                 . . .                 x . x                 x . x
   . O . t_n ⇒ t_n+1     . X .   oder          . O . t_n ⇒ t_n+1     . X .
   x x x                 x x x                 x . .                 x . .
   ```

2. ein belegter Pixel mit weniger als zwei belegten Nachbarpixeln wird frei

   Beispiele:
   ```
   . . x                 . . x                 . . .                 . . .
   . X . t_n ⇒ t_n+1     . O .   oder          . X . t_n ⇒ t_n+1     . O .
   . . .                 . . .                 . . .                 . . .
   ```

3. ein belegter Pixel mit zwei oder drei belegten Nachbarpixeln bleibt belegt

Beispiele:	x	.	x		x	.	x		.	.	.		.	.	.
	.	X	.	$t_n \Rightarrow t_{n+1}$	.	X	.	oder	x	X	.	$t_n \Rightarrow t_{n+1}$	x	X	.
	x	.	.		x	.	.		x	.	.		x	.	.

4. ein belegter Pixel mit mehr als drei belegten Nachbarpixeln wird frei

Beispiele:	x	.	x		x	.	x		x	x	x		x	x	x
	x	X	.	$t_n \Rightarrow t_{n+1}$	x	O	.	oder		X		$t_n \Rightarrow t_{n+1}$		O	
	x	.	.		x	.	.		x	x	x		x	x	x

5. Randpixel, welche an den Rändern (rechts, links, oben, unten) außerhalb der dargestellten Animationsfläche liegen, werden zwar nicht in der Grafikausgabe ausgegeben, sollen aber dennoch berechnet werden.
   Dies erklärt $(Zeilen + 2)$ und $(Spalten + 2)$ bei der Angabe der Mindest-Element-Anzahl des Animations-Puffers – siehe Teilaufgabe 3.1. Alle Pixel, welche noch weiter außerhalb der Animationsfläche liegen, gelten als unveränderbare freie Pixel.

In den obigen Beispielen:

- . und O bedeutet frei, bei O wird die Regel auf den Pixel angewendet.
- x und X bedeutet belegt, bei X wird die Regel auf den Pixel angewendet.

Zur eigenständigen Kontrolle auf Korrektheit des Algorithmus:

- Die Berechnung ist korrekt, wenn bei der vorgegebenen Konfiguration, nach 15 Animationsschritten, das Muster von Animationsschritt 0, wieder im Animations-Puffer vorliegt.
- Testen Sie auch das Verhalten mit größeren x-y-Dimensionen und einer, mit Zufallswerten initialisierten Animationsfläche (wiederkehrende Muster ergeben sich nur selten).

3.4 Grafische Darstellung eines Animationsbildes

Diese Teilaufgabe ist einfach und kann von einem Teammitglied bearbeitet werden.

Die Funktionalität soll im C-Modul gfx.c implementiert werden.

Bei diesem Teil soll der aktuelle Animations-Puffer mit Hilfe der zur Verfügung gestellten Grafik-Funktionen aus dem C-Modul graphic.c in das Grafikausgabefenster gezeichnet werden.

Dabei ist zu beachten, dass der Animations-Puffer Bereiche (den Rand der Animationsfläche) enthält, welche nicht gezeichnet werden sollen (Details finden Sie unter dem Stichwort Randpixel im Punkt 5 in der Teilaufgabe 3.3).

Außerdem muss die Leinwand vor dem ersten Zeichnen in der korrekten Größe initialisiert werden.

Die aus dem Modul graphic.c anzuwendenden Funktionen heißen:
grafik_create_paint_area(), grafik_paint_point(), grafik_lock_for_painting() und grafik_unlock_and_show().

3.5 Behandlung von Nutzereingaben

Diese Teilaufgabe ist einfach und kann von einem Teammitglied bearbeitet werden.

Die Funktionalität dieser Teilaufgabe soll im C-Modul ui.c implementiert werden.

Auf Tastatureingaben vom Nutzer soll das Programm folgendermaßen reagieren:

- Taste q : Programm beenden
- Leertaste : Animation pausieren, oder falls pausiert, weiterführen
- Taste . (Punkt) : den nächsten Animationsschritt anzeigen und dann die Animation pausieren
- Weitere Tastatureingaben dürfen ignoriert werden.

Bei dieser Teilaufgabe bietet es sich an, die benötigte Pause zwischen den einzelnen Animationszyklen einzubauen. Die Funktion grafik_user_input nimmt eine Wartezeit in Millisekunden als Parameter entgegen.

Hinweis: falls dieser Programmteil über einen längeren Zeitraum verhindert, dass die Grafikausgabe-Funktionen aufgerufen werden, könnte die Grafik im Ausgabefenster zerstört werden. Dies ist sehr gut während der farb_demonstration zu erkennen, wenn Sie das zusätzliche Informations-Dialogfenster mit der Maus verschieben.

3.6 Programm-Hauptschleife

Diese Teilaufgabe ist einfach und kann von einem Teammitglied bearbeitet werden.

Die Programm-Hauptschleife (manchmal auch als `main event loop` bezeichnet) soll im C-Modul `main.c` implementiert werden.

Die Hauptschleife führt die oben beschriebenen Teilaufgaben zu einem funktionierenden Programm zusammen und ruft dabei die Funktionen der einzelnen C-Module in geeigneter Reihenfolge und Anzahl auf. Dies soll so lange geschehen, bis der Nutzer das Programm mit der Taste `q` beendet.

4 Hinweise zur Entwicklungsumgebung (Toolchain)

Die Lösung der Praktikumsaufgabe und der vorgegebene Quelltext wurden auf der MRT Virtual Appliance 2019 (kurz: `MRT-VM`) entwickelt und getestet.

Die Entwicklung und Präsentation des Programms, während des Praktikumstermins, erfolgt in der `MRT-VM`.

Es ist denkbar, die Aufgabe mit einer eingerichteten Toolchain unter Windows, Mac, Linux oder dem RaspberryPi zu programmieren. Stellen Sie jedoch sicher, dass Ihr Programm für die Präsentation und Abgabe in der `MRT-VM` compiliert und funktioniert.

Hilfestellung kann seitens der Praktikumsbetreuung nur für die `MRT-VM` und Debian-basierte Linux Systeme gegeben werden.

Es ist ihnen freigestellt, welche IDE oder Editoren Sie verwenden möchten. Sie können `Eclipse`, welches in der Vorlesung genutzt wurde verwenden und den vorgegebenen Quelltext in ein Projekt importieren. Sie können allerdings auch ohne Eclipse und mit dem vorgegebenen `Makefile` arbeiten.

Führen Sie folgende Schritte durch, um die Entwicklungsumgebung in der `MRT-VM` für diesen Praktikumsversuch zu vervollständigen:

1. Die `MRT-VM` benötigt eine Internetverbindung zum Herunterladen der benötigten Daten.
2. Kopieren Sie das entpackte Archiv mit dem Quellcode zur Aufgabe in einen Order auf der `MRT-VM`.
   z.B.: direkt auf den `Schreibtisch`
3. Öffnen Sie eine Konsole (`XTerm`) in der `MRT-VM` und wechseln Sie in das Verzeichnis mit dem Quellcode:

   ```
   cd ~/Schreibtisch/MRT1__V3__Animation/
   ```

4. Führen Sie folgendes Kommando in der Konsole aus:

   ```
   make setup
   ```

   Dies installiert die Entwicklungsdateien der Grafikbibliothek `SDL2` und alle benötigten Software-Abhängigkeiten.
   Das geforderte Passwort lautet `mrt`, bestätigen Sie die nächste Abfrage mit der Taste `Return`.
5. Der Quellcode kann anschließend in Eclipse importiert werden, mittels:

   ```
   File -> New -> Makefile Project with Existing Code
   ```

   Wählen Sie `Linux GCC` bei `Toolchain for Indexer Settings` aus.

   Der vorgegebenen Quelltext kann auch ohne Eclipse, mittels folgenden Kommandos kompiliert werden:

   ```
   make
   ```

6. Das so entstandene Programm können Sie mittels des Kommandos

   ```
   ./animation
   ```

   starten.
7. Zum Aufräumen des Verzeichnisses führen Sie folgendes Kommando aus:

   ```
   make clean
   ```

   Alle generierten Dateien `*.o`, `*.gch` und die ausführbare Datei werden dann gelöscht.

5 Hinweise zur Praktikumsdurchführung

- Der Praktikumsversuch wird im Wintersemester 2021/2022 in Präsenz durchgeführt.
- Durchführungsort ist im WS 2021/2022 der Raum BAR E33/34.
- Um am Praktikumsversuch teilnehmen zu können ist ein tagesaktueller Corona-Test für alle Teilnehmenden erforderlich (3G+) Impf- und Testnachweise sind vor dem Praktikumstermin dem Betreuer vorzulegen.
- Am Anfang des Praktikums wird ein kurzer (10-20 Minuten) schriftlicher Eingangstest durchgeführt. Abgefragt wird für den Versuch benötigtes Grundlagenwissen. Wenn Sie aktiv an der Lösung der Praktikumsaufgabe programmiert haben, sollten Sie den Test bestehen können.
- Idealerweise bringen Sie zum Praktikumstermin ein funktionierendes Programm mit, welches die Praktikumsaufgabe löst und stellen es dem Betreuer vor. Das Programm muss unter Linux in der, in der Vorlesung genutzten, MRT Virtual Appliance laufen. Nach erfolgreicher Präsentation dürfen Sie den Praktikumstermin vorzeitig beenden. Es können zum Praktikumstermin Probleme zusammen mit dem Betreuer gelöst und Fragen an den Betreuer gestellt werden. Es ist jedoch aufgrund der knappen Zeit und der begrenzten IT-Ressourcen nicht realistisch die Aufgabe ohne ausreichend Vorarbeit seitens der Gruppe vollständig zu lösen.
- Eine Woche nach dem Praktikumstermin ist eine Programmdokumentation als PDF per E-Mail beim Betreuer abzugeben – Details dazu im Praktikumsversuch.

6 Hinweise zur Aufgabe:

Schauen Sie sich die Funktion `farb_demonstration` genau an, darin sind einige Lösungshinweise enthalten.

Funktionen eines Moduls können leicht getestet werden indem Ergebnisse aus anderen Modulen, welche als Vorarbeit benötigt werden, einmal händisch berechnet und in den Quelltext geschrieben werden. Die finalen Tests können Sie dann später durchführen, vergessen Sie nicht die vorberechneten Sachen auszukommentieren.

Geben Sie zum Überprüfen Ihres Quellcodes, Zwischenergebnisse auf der Konsole aus oder schauen Sie sich die entsprechenden Speicherbereiche im Debugger an.

Wenn Sie unsicher sind, wie sich ein Teil ihres Programms verhält, dann schreiben Sie ein kleines separates Testprogramm in C, welches nur diese eine Sache ausführt und die Ergebnisse auf die Konsole ausgibt. So können Sie ihren Quellcode testen und verbessern. Der so entstandene Quellcode kann anschließend ins „große“ Programm übernommen werden.

Schreiben Sie kleine, kurze Funktionen mit aussagekräftigen Funktions- und Variablennamen. Dies senkt die Komplexität und hilft bei der Fehlersuche. (Stichwort: Divide and Conquer)

Die Praktikumsaufgabe ist weniger komplex als es diese Aufgabenstellung vermuten lässt.

Sämtliche Personenbezeichnungen in diesem Dokument wurden im geschlechterneutralen Sinn verwendet.

Bei Fragen – organisatorischer oder inhaltlicher Natur, und auch bei Problemen mit der Aufgabe oder Fehlern in der Aufgabenstellung – können Sie sich per E-Mail an den Betreuer des Praktikumsversuchs Mario Herhold wenden. Die E-Mailadresse finden sie auf den Webseiten der TU Dresden.

7 Arbeits- und Brandschutzhinweise

7.1 Vorbeugende Maßnahmen:

- Die Praktikumsteilnehmer haben sich so zu verhalten, dass Gefahrensituationen und Unfälle vermieden werden.
- Die Befugnis zum Bedienen und Nutzen von Geräten ist auf den zugewiesenen Praktikumsplatz beschränkt.
- Eingriffe in die zum Praktikumsaufbau gehörenden Geräte sind nicht erlaubt.
- Der Anschluss und der Betrieb privater Geräte in den Praktikumsräumen ist verboten.
- Defekte an Geräten oder Gebäudeeinrichtungen sind unverzüglich dem Betreuer mitzuteilen. Betroffene Geräte sind außer Betrieb zu nehmen. Andere Personen sind vor Gefahren zu warnen.
- Den Anweisungen der Praktikumsbetreuer bzw. anderer aufsichtsführender Personen ist unbedingt Folge zu leisten.
- Betriebsfremde dürfen sich nur mit Erlaubnis des Praktikumsbetreuers in den Praktikumsräumen aufhalten.
- Rauchen und Umgang mit offenem Feuer ist nicht gestattet.
- Nach Ende des Praktikums ist der Arbeitsplatz in sauberem und aufgeräumtem Zustand zu hinterlassen.
- Außergewöhnliche Ereignisse bzw. besondere Vorkommnisse sind umgehend dem Betreuer oder dem diensthabenden Assistenten zu melden.
- Die aktuell gültigen Verhaltensregeln in Bezug auf die COVID 19 Erkrankung sind einzuhalten.

7.2 Verhalten im Falle eines Brandes:

- Beachten der richtigen Reihenfolge: **MELDEN - RETTEN - LÖSCHEN**

7.2.1 Feuer melden:

- Telefonische Brandmeldung:
  - Notruf 112 der Feuerwehr (von jedem Telefon aus möglich)
  - Notruf HA 34515 der Technischen Leitzentrale der TUD
- Deutliche, genaue und vollständige Angaben:
- Wo brennt es?
- Was brennt?
- Angaben zu verletzten oder gefährdeten Personen
- Wer meldet?

7.2.2 Personen retten:

- Erste Hilfe leisten
- Weitere Hilfe organisieren, medizinische Hilfe anfordern
- Gefahrenbereich räumen; Fluchtwege benutzen, keine Aufzüge
- Andere Personen warnen, Sammelplatz (Platz vor Turmeingang zum Barkhausenbau) aufsuchen
- Behinderten und älteren Personen helfen

7.2.3 Löschversuch unternehmen

- Feuerlöscher verwenden (Standorte: Gänge des Barkhausenbaues), dabei sich nicht selbst gefährden
- Fenster und Türen schließen, aber nicht abschließen
- Möglichst elektrische Verbraucher abschalten

7.3 Rufnummern für Notfälle:

Helfer	Telefonnummer
Rettungsdienst	112
Polizei	110
TUD-Notruf	34515
Betriebsärztlicher Dienst	36199
Klinikum Friedrichstadt Notaufnahme	0351-480 1938