Jan. 2020

Praktikum zur Lehrveranstaltung Mikrorechentechnik I
Versuch 3, C–Programmierung

Organisation des Praktikumversuchs: M. Herhold

Inhaltsverzeichnis

1 Versuchsziel

Versuchsziel ist die Festigung der Vorlesungskenntnisse zur Sprache C durch Entwurf und Implementierung eines einfachen[1] Anwendungsprogramms.

Schwerpunktmäßig werden betrachtet:

- methodische Vorgehensweise bei der Problemanalyse und Strukturierung von C-Programmen
- Anwendung wichtigster Statements der Sprache C
- Modularisierung von Programmen und Schnittstellengestaltung
- Anwendung von Bibliotheksfunktionen
- Umgang mit Werkzeugen zur Programmerzeugung (Editor, Compiler, Linker)
- Systematik der Fehlersuche mittels Debugger.

Als Werkzeug wird die Entwicklungsumgebung Eclipse CDT eingesetzt.

[1] https://en.wikipedia.org/wiki/Greenspun%27s_tenth_rule

2 Allgemeine Aufgabenstellung

Folgendes Beispiel aus der Welt der fraktalen Grafiken soll numerisch untersucht werden:
Für die Iterationsgleichung

$$z_{i+1} = c + z_i^2 \tag{1}$$

in der z_i, Z_{i+1} und c komplexe Zahlen mit $z_i = x_i + jy_i$ und $c = a + jb$ bedeuten, soll geprüft werden, nach wie vielen Iterationen i die komplexe Zahl z_i ein vorgegebenes Gebiet G (z.B. einen Kreis) verlässt. Jeder Zahl i, für die z_i das vorgegebene Gebiet verlässt, soll eine Farbe zugeordnet werden. Wenn sich z_i für $i > imax$ immer noch in diesem Gebiet befindet, wird die Iteration abgebrochen und diesem Punkt eine bestimmte Farbe (z.B. schwarz) zugeordnet. Die auf diese Weise entstehenden geometrischen Figuren werden

- bei Variation von c als Mandelbrot-Menge[2] und
- bei Variation von z_i als Juliamenge[3] (nach dem Mathematiker Gaston Julia)

bezeichnet.

Zur numerischen Untersuchung und grafischen Darstellung des Phänomens soll ein Anwendungsprogramm entworfen und implementiert werden. Das Anwendungsprogramm soll die Vorgabe von Untersuchungsparametern mittels Dialog ermöglichen.

3 Detaillierte Aufgabenstellung

Sicher gibt es viele Möglichkeiten, das unter Abschnitt 2 beschriebene Problem programmiertechnisch umzusetzen. Hier, im Versuch, geht es weniger um eine besonders schnelle Ausgabe fraktaler Grafiken, sondern vielmehr um den systematischen Entwurf und Realisierung von Programmen.
Mit den folgenden Aufgaben sollen vor allem verschiedene C-Sprachkonstrukte und deren Anwendungsmöglichkeiten zu einer übersichtlichen, modularen Implementierung angesprochen werden.

Ein vorbereiteter Programmrahmen ist als Download über das Internet möglich. (siehe Abschnitt Downloads):

3.1 Entwurf Grobstruktur

Entwerfen Sie die Grobstruktur für Ihr Anwendungsprogramm.
Eine Hilfestellung zur Strukturierung, wird durch die nachfolgenden Teilaufgaben gegeben.
Das Programm sollte zweckmäßig aus folgenden Teilen bestehen:

- Hauptprogramm
- Mathematischer Teil mit Funktionen zur fraktalen Analyse (Header und Implementierung)
- Grafischer Teil zur pixelweisen Ergebnisdarstellung (Header und Implementierung)
- Ein-/Ausgabe-Teil zur Benutzerkommunikation (Header und Implementierung).

3.2 Mathematischer Teil, Entwurf und Implementierung

3.2.1 Teilaufgabe Struktur `param_t`

Entwerfen Sie eine Struktur `param_t` mit folgenden Datenelementen:

- `radius` R zur Beschreibung des kreisförmigen Gebietes G für komplexe Zahlen
- `imax` zur Beschreibung der maximalen Iterationsanzahl
- `fraktal_t` als Enumerationstyp für die Fraktalvarianten Mandelbrot- und Juliamenge
- `xmin`, `xmax`, `ymin`, `ymax` zur Kennzeichnung des komplexwertigen Analysegebietes
- `xpoints`, `ypoints` als Anzahlen für die Linien im Analysegebiet.

[2] https://de.wikipedia.org/wiki/Mandelbrot-Menge
[3] https://de.wikipedia.org/wiki/Julia-Menge

Optional: Verzichten Sie, zu Übungszwecken, bei der Definition und Deklaration der Struktur `param_t` auf die Verwendung des Schlüsselwortes `typedef`.

3.2.2 Teilaufgabe Struktur `complex_t`

Entwerfen Sie eine Struktur `complex_t` zur Beschreibung komplexer Zahlen durch kartesische Koordinaten!

3.2.3 Teilaufgabe Funktion: `get_itera`

Konzipieren und implementieren Sie eine Funktion `get_itera` zur Bestimmung der Iterationsanzahl nach Gleichung 1 mit folgenden Schnittstelleneigenschaften:

- *Eingabeparameter*: komplexe Zahl c, komplexe Zahl z, (Datenstruktur `complex_t`) zur Analyse
- *Rückgabewert*: Iterationsanzahl.

3.2.4 Teilaufgabe Funktion `get_color_value`

Konzipieren und implementieren Sie eine Funktion `get_color_value` zur Zuordnung von Farbwerten zu einer Iterationszahl mit folgenden Schnittstelleneigenschaften:

- *Eingabeparameter*: aktuelle Iterationszahl `i`, maximale Iterationszahl `imax`
- *Rückgabewert*: Farbwert (als Enumerationstyp `color_name_t`, vgl. Datei `graphic.h`)

Hinweis: Falls mehr Farben, als verfügbar, benutzt werden, sollte die aktuelle Iterationszahl `i` vor ihrer Farbzuordnung auf die mögliche Anzahl der Farbwerte reduziert werden. Setzen Sie hierzu zweckmäßig die Modulo-Funktion ein!

3.2.5 Teilaufgabe Funktion `fraktal`

Konzipieren und implementieren Sie eine Funktion `fraktal` mit folgenden Schnittstelleneigenschaften:

- *Eingabeparameter*: komplexe Zahl c, komplexe Zahl z, (Datenstruktur `complex_t`) zur Analyse
- *Rückgabewert*: keiner.

Die Funktion soll die Mandelbrot- oder Juliamenge für das vorgegebene Analysegebiet berechnen und grafisch darstellen. Nutzen Sie hierzu die Funktionen `get_itera` und `get_color_value` der Aufgaben 3.2.3 und 3.2.4, sowie die vorgegebene Funktion `grafik_paint_point` aus dem grafischen Teil.

Anmerkung: Es gibt eine Besonderheit bei der vorbereiteten `libSDL2`[4] Grafikausgabefunktionalität. Sie müssen vor dem Benutzen von `grafik_paint_point` die Funktion `grafik_lock_for_painting` und danach die Funktion `grafik_unlock_and_show` aufrufen. Diese Funktionen sind im Grafikteil des Programms (Dateien graphic.*) definiert.

3.3 Benutzerschnittstelle, Entwurf und Implementierung

3.3.1 Teilaufgabe Funktion `param_dialog`

Konzipieren und implementieren Sie eine Funktion `param_dialog` mit folgenden Schnittstelleneigenschaften:

- *Eingangs- / Ausgangsparameter*: Parameterstruktur nach Aufgabe 3.2.1, komplexe Zahl c
- *Rückgabewert*: Ganzzahl-Datentyp; falsch für Programmabbruch; andere Werte, sonst.

Die Dialogfunktion soll dem Nutzer alle benötigten Werte anzeigen und deren Änderung ermöglichen. Die Sinnhaftigkeit des Parametersatzes ist zu testen, entsprechende Fehlerausschriften sind zu realisieren.

Hinweis: Hier können Sie kreativ werden und die Möglichkeiten der SDL2[4] Bibliothek erkunden, oder eine Minimallösung mit schrittweiser Abfrage in der Terminalemulation[5] implementieren.

[4] https://www.libsdl.org/
[5] https://de.wikipedia.org/wiki/Terminalemulation

3.4 Hauptprogramm, Entwurf und Implementierung

3.4.1 Abarbeitungsreihenfolge

Das Hauptprogramm soll folgende Codesequenz beinhalten:

1. globale Variablendeklaration für die komplexen Zahlen `c1` und `z1` sowie die Parameterstruktur vom Typ `param_t` und deren Anfangswertinitialisierung
2. `while`-Schleife mit
   a) Abbruchmöglichkeit für das gesamte Anwendungsprogramm
   b) Parameterdialog nach 3.3.1
   c) Grafikinitialisierung
   d) Fraktalberechnung und -darstellung
   e) Aufruf einer Tastaturfunktion zum Beenden der Grafikansicht
   f) Close-Funktion für die Grafik.

3.4.2 Variablen-Initialisierung

Das Hauptprogramm soll folgende Initialisierungen als Voreinstellungen vornehmen:

- Fraktaltyp = Mandelbrot-Menge
- Quadrat des Konvergenzradius = 4
- Analysegebiet $-2 \leq x \leq +2$; $-2 \leq +2$
- maximale Iterationsanzahl = 75
- Linienanzahl für x-Intervall = 400
- Linienanzahl für y-Intervall =300.

Als Initialwerte für die komplexen Zahlen werden $c1 = 0.4 + j0.4$ und $z1 = 0.0 + j0.0$ empfohlen.

3.4.3 Hinweise

Das vorbereitete Hauptprogramm beinhaltet einige auskommentierte Funktionsaufrufe. Aufrufe von `grafik_create_paint_area`, `grafik_lock_for_painting` und `grafik_unlock_and_show`, dienen der Erzeugung und Aktualisierung der Grafikausgabe. Je nach Art und Weise ihrer Aktualisierungsstrategie zur Grafikausgabe, müssen diese Funktionen im Hauptprogramm oder einer Unterfunktion aufgerufen werden. Ein Beispiel gibt die Funktion `farb_demonstration`.

3.5 Programmtest

Bereiten Sie eine Teststrategie zur schrittweisen Inbetriebnahme und zum Funktionsnachweis Ihrer Programme vor!

3.6 Hinweise zur Inbetriebnahme des vorbereiteten Programmrahmens

Bei dem vorbereiteten Programmrahmen handelt es sich um einen kompletten `Eclipse-Workspace`, welcher, nach dem Entpacken, über folgenden Menüpunkt in Eclipse geladen werden kann:

```
"File" → "Switch Workspace" → "Other" ... (dann Auswahl des Workspace-Verzeichnisses und Bestätigen mit "OK")
```

Der vorbereitete Programmrahmen nutzt zur graphischen Ausgabe die Bibliothek Simple DirectMedia Layer[4] (SDL) in Version 2.0. Zur Installation in der Virtual Appliance der Vorlesung MRT1 nutzen Sie bitte folgenden Kommandozeilenbefehle[6]:

```
sudo apt-get update; sudo apt-get install libsdl2-dev
```

Nach Erfolgreichem Kompilieren und Ausführen des Programmrahmens, in Eclipse oder per `Makefile`, sehen Sie ein Fenster mit den 32 vorgegebenen Farben und einen Informationsdialog.

3.6.1 FAQ:

- *Wie bekomme ich die Daten in die Virtuelle Maschine?*: Über die Einstellungen der MRT-Appliance, können Sie vor dem Start der VM einen `Gemeinsamen Ordner` festlegen, welcher dann auf Ihrem PC und in der VM sichtbar ist. Sie können auch einen physischen USB-Massenspeicher über das `Geräte`-Menü in die VM einbinden. Sie können mit `git` arbeiten und ein selbst angelegtes Repository clonen. Sie können die Daten mittels `wget` von einem Web-Server herunterladen. Fast alle Methoden Daten von einem Rechner auf einen anderen zu transportieren sind möglich.
- *Warum wird der Bildinhalt des Grafikausgabefensters zerstört, wenn ich mit Eclipse weiterarbeite?*: Das liegt daran, dass ihr Programm solche Ereignisse nicht abfängt und das Grafikfenster nicht aktualisiert. Ihr C-Programm ist für das Neuzeichnen selbst verantwortlich. *Optionale Aufgabe:* versuchen Sie das Problem programmiertechnisch zu beheben.
- *Die Grafikausgabe ist extrem langsam, ist mein Rechner nicht schnell genug?*: Heutige Prozessoren rechnen schnell genug, höchstwahrscheinlich liegt es an der Strukturierung ihres Programms, versuchen Sie den Fehler selbständig zu finden.
- *Der Debugger funktioniert nicht.*: installieren Sie `gdb` mit folgendem Kommandozeilenbefehl:

  ```
  sudo apt-get update; sudo apt-get install gdb
  ```

3.7 Kontrollfragen:

- Welche Bedeutung hat die `main()` Funktion?
- Erklären Sie die Bedeutung der Dateien `*.h` und `*.c`!
- Wie wird die getrennte Übersetzbarkeit von Quellcodeabschnitten während der Programmentwicklung realisiert?
- Welche Bedeutung haben `*.so` und `*.a` Dateien?
- Was bedeuten die Begriffe *Präprozessor*, *Compiler*, *Linker*, *Debugger*?
- Erklären Sie Syntax und typische Anwendungen der Konstrukte `if`, `switch ... case`!
- Erklären Sie Syntax und typische Anwendungen der Konstrukte `for`, `while`, `do ... while`!
- Was ist unter den Begriffen *Iteration* und *Rekursion* in Hinsicht auf die Programmabarbeitung zu verstehen?
- Erklären Sie verschiedene Arten der Parameterübergabe bei Funktionen!
- Erklären Sie Bedeutung und Anwendung von Increment- /Decrementoperatoren!
- Welche wichtigsten vordeklarierten Datentypen existieren in C?
- Wie kann man benutzerdeklarierte Datentypen in C vereinbaren? Nennen Sie einige typische Anwendungsbeispiele!
- Was ist ein Pointer und wie benutzt man diese in C?
- Wie arbeitet man unter C mit Arrays?
- Wie arbeitet man unter C mit Zeichenketten (`strings`) und wie ändert man diese?
- Welche Lebensdauer und welchen Gültigkeitsbereich (`scope`) haben Variablen?
- Wie werden komplexe Datenstrukturen als Parameter oder Funktionsrückgabe gehandhabt?

[6] selbstverständlich kann auch ein grafisches Nutzerinterface zum Paketmanager genutzt werden.

4 Hinweise zum Praktikumstermin und dessen Durchführung

4.1 Vorbereitung

- Arbeiten Sie die Aufgabenstellung und die Arbeitssicherheitshinweise in diesem Dokument[7] durch.
- Laden Sie den vorbereiteten Programmrahmen (Eclipse Workspace) herunter. (siehe Abschnitt Downloads)
- Installieren Sie die Entwicklerdateien zur libSDL v2.0 in Ihrer Programmierumgebung (siehe 3.6)
- Sie können ihr Anwendungsprogramm zu Hause, mit Hilfe der VirtualBox Appliance (siehe Downloads), soweit wie möglich vorbereiten und fertig stellen.
- Bringen Sie den von Ihnen so vorbereiteten Eclipse Workspace zur Versuchsdurchführung mit. (*Hinweis:* Archivierungsdateiformat nutzen, z.B.: `zip` oder `tar.gz` um die Datenintegrität zu gewährleisten)
- Fragen zur Versuchsaufgabe und Programmierung können Sie per E-Mail an M.Herhold (Lehrstuhl Automatisierungstechnik) stellen (eine Bearbeitung erfolgt nur während der regulären Büroarbeitszeiten).

4.2 Kolloquium

Vor dem Beginn der praktischen Durchführung steht ein schriftlicher Eingangstest mit Fragen zu Themen der Vorlesung, die für die Durchführung dieses Versuchs relevant sein könnten.

Die beste Strategie zum Bestehen dieses Tests ist die selbständige Bearbeitung der zur Vorlesung gehörenden praktischen Übungsaufgaben und die selbständige Bearbeitung der Versuchsaufgabe vor der Teilnahme am Praktikum.

4.3 Aufbau und Nutzung des Versuchsplatzes

Der Versuchsplatz ist ein PC mit Linux und vorhandener (leicht eingeschränkter) Internetanbindung. Im System ist die Virtualisierungssoftware VirtualBox, der Firma Oracle, installiert.

Die Programmierung und Ausführung Ihres Programms erfolgt innerhalb der, von der Vorlesung zur Verfügung gestellten und genutzten, VirtualBox Appliance (siehe Downloads). Nutzername und Passwort innerhalb der Appliance lauten: `mrt`.

Die Übertragung ihres Eclipse Workspaces ist per USB-Stick oder Download aus dem Internet möglich. Die Nutzung eines Archivierungsdateiformats (z.B.: `zip`) stellt sicher, dass die Daten korrekt übertragen werden und Dateiberechtigungen erhalten bleiben. Diese Vorgehensweise eliminiert einige mögliche Fehlerursachen.

Versehen Sie das Verzeichnis Ihres genutzten Eclipse Workspaces mit ihrer Gruppennummer `workspace_GrNr-xx`

Nutzen Sie die IDE Eclipse CDT in der VirtualBox Appliance zur Bearbeitung und Vorführung Ihres Anwendungsprogramms. Anleitung zur Nutzung der IDE, siehe Vorlesung und Vorlesungsunterlagen zu Mikrorechentechnik 1 & 2.

4.4 praktische Versuchsdurchführung

Erproben Sie systematisch Ihr Anwendungsprogramm. Führen Sie einen Nachweis zur korrekten Arbeitsweise der einzelnen Komponenten aus! Testen Sie insbesondere auch die Programmsequenzen zur Parameteranalyse (Welche Kombinationen von Eingangsparametern könnten zu Programmabstürzen führen?)

Führen Sie mit Ihrem Programm Experimente zur Veranschaulichung fraktaler Grafiken aus.

Präsentieren Sie Ihr Anwendungsprogramm dem anwesenden Betreuer und beantworten Sie Fragen zu Ihrer Lösung. Nach erfolgreicher Präsentation Ihrer Anwendung, kann der Versuchstermin vorzeitig beendet werden.

4.4.1 Kontrollfragen zur Präsentation

- Begründen Sie den Entwurf Ihrer programmiertechnischen Lösung!
- Begründen und erklären Sie Implementierungsdetails!
- Erklären Sie Ihre Vorgehensweise zum Test der korrekten Implementierung!

[7] erstellt mit Hilfe von org-mode

4.5 Versuchsauswertung

Zum Bestehen des Praktikums ist von jeder Praktikumsgruppe ein gemeinsames Protokoll, spätestens 1 Woche nach dem Versuchstermin, abzugeben.

Das Protokoll soll eine Dokumentation des Projektes darstellen.

Folgende Abschnitte sollen enthalten sein:

- Protokolldeckblatt gemäß der Vorlage! (Siehe Abschnitt Downloads)
- Programmentwurfs (Erklärung der Entwurfsentscheidungen)
- ein UML-Aktivitätsdiagramm
- die Dokumentation der Inbetriebnahme (Strategie, Vorgehensweise Fehlerkorrektur),
- eine kritische Einschätzung der eigenen Leistungen,
- mögliche Verbesserungen zur eigenen Vorgehensweise
- (ausschließlich) die Quelltextabschnitte, welche sich vom bereitgestellten Programmrahmen unterscheiden.

Das Protokoll, zusammen mit dem zu verwendenden Deckblatt, ist in Papierform abzugeben.

Ort der Abgabe: im Sekretariat des Lehrstuhls Automatisierungstechnik (Frau Möge, Raum BAR E04), oder einzuwerfen in den blauen Briefkasten vor dem Raum BAR E03.

5 Downloads

VirtualBox Appliance siehe Opal-Ressourcen zur Vorlesung Mikrorechentechnik 1 & 2

Programmrahmen (Eclipse Workspace) siehe Opal-Ressourcen zum Praktikum Mikrorechentechnik 1, ART3 – C-Programmierung, oder den Webseiten des Lehrstuhls Automatierungstechnik: `http://www.et.tu-dresden.de/ifa/` - Studium - Hinweise Praktikum Mikrorechentechnik

Protokolldeckblatt siehe Opal-Ressourcen zum Praktikum Mikrorechentechnik 1, ART3 – C-Programmierung, oder den Webseiten des Lehrstuhls Automatierungstechnik: `http://www.et.tu-dresden.de/ifa/` - Studium - Hinweise Praktikum Mikrorechentechnik

Simple DirectMedia Library Version 2.0 Entwicklerdateien `https://www.libsdl.org/download-2.0.php`

6 Literatur

- siehe Buch-Empfehlungen der Vorlesung.

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.

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