PR3 - Übungsblatt A.2

WS2021/22 (Stand 2021-09-29 16:39)

Prof. Dr. Holger Peine
Hochschule Hannover
Fakultät IV – Abteilung Informatik
Raum 1H.2.60, Tel. 0511-9296-1830
Holger.Peine@hs-hannover.de

Thema

Quelldateien und Makefiles

Termin

Ihre Arbeitsergebnisse zu diesem Übungsblatt führen Sie bitte in den Übungen am 13.-14.10.2021 vor.

1 Eigene Headerdatei für zwei Quellcode-Module (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 2c

a) Schreiben Sie ein Programm `math` mit folgendem beobachtbaren Verhalten:

```
$ ./math
Ihre Wahl:
<S>umme oder <D>ifferenz? S
Erster Summand: 4
Zweiter Summand: 5
Ergebnis: 9
$ ./math
Ihre Wahl:
<S>umme oder <D>ifferenz? D
Minuend: 7
Subtrahend: 2
Ergebnis: 5
```

Das `$`-Zeichen steht hier für das Unix-Eingabeprompt. Benutzereingaben sind **unterstrichen** dargestellt.

Das Programm soll in folgende Module aufgeteilt werden:

- `math.c`: Funktion `void berechne(void)` mit der Logik für die Benutzerauswahl `<S>umme` oder `<D>ifferenz` und dem Aufruf der ausgewählten Funktion und der Ausgabe von deren Ergebnis. Weiterhin soll in `math.c` folgende `main`-Funktion enthalten sein:

  ```
  int main(void) {
      berechne();
      return 0;
  }
  ```

- `summe.c`: Hier wird eine Funktion `int summe(void)` zur Berechnung der Summe implementiert. Die Entgegennahme der Eingaben sollen an das Modul `input.c` delegiert werden (s. u.). Die Funktion gibt die Summe der eingegebenen Zahlen zurück.
- `differenz.c`: Hier wird eine Funktion `int differenz(void)` zur Berechnung der Differenz implementiert. Die Entgegennahme der Eingaben soll an das Modul `input.c` delegiert werden. Die Funktion gibt die Differenz der eingegebenen Zahlen zurück.

- `input.c`: Hier wird eine Funktion mit einem Prompt-Parameter implementiert:
  `int get_input(char prompt[]);`
  Diese Funktion soll den Prompt ausgeben und die Benutzereingabe zurück liefern. Z.B. könnte der erste Aufruf im obigen Beispiel durch `get_input("Erster Summand")` realisiert werden, was die Zahl 4 zurückgeben würde.

Die Funktion `get_input` wird von den Modulen `summe.c` und `differenz.c` aufgerufen. Die Funktionen `summe` und `differenz` werden vom Modul `math.c` aufgerufen. Damit das auch übersetzbar ist, implementieren Sie bitte drei Headerdateien `summe.h`, `differenz.h` und `input.h` und includieren diese geeignet.

Hinweise:

- Für die Ein- und Ausgaben sollen Sie die Funktionen `printf` und `scanf` benutzen. Zur Verwendung von `printf` und `scanf` lesen Sie bitte die „Hinweise vor dem Start“ auf Aufgabenblatt A.1.
- Die Eingabe eines einzelnen Zeichens kann man mit
  `char zeichen; scanf("%c", &zeichen);`
  realisieren, analog die Eingabe einer Ganzzahl in Dezimalnotation mit
  `int i; scanf("%d", &i);`
- Der Parameter `prompt` der Funktion `get_input` vom Typ `char[]` ist eine Zeichenkette, die Sie einfach mit `printf("%s", prompt);` ausgeben können.
- Beim Aufruf von `get_input` geben Sie den gewünschten Prompt als Zeichenkettenkonstante in Anführungszeichen an (ganz wie in Java).

b) Versuchen Sie einmal, die Funktion `get_input` direkt in der Header-Datei `input.h` zu **definieren**, d. h. sie vollständig mit dem Funktionsrumpf dort anzugeben (damit die Datei `input.c` danach nicht leer ist, was der Compiler nicht akzeptieren würde, ersetzen Sie ihren Inhalt durch `int dummy;`) und das Programm zum Laufen zu bringen. Beobachten Sie die Fehlermeldung: Welcher der drei Übersetzungsphasen ist die Meldung zuzuordnen? Erläutern Sie die Ursache der Fehlermeldung.

c) Angenommen, aus irgendeinem Grund müsste in allen Header-Dateien die Präprozessor-Deklaration `#define PROJECT_NAME "Tolles Projekt"` bekannt sein. Wie kann man dies so erreichen, dass dabei diese Deklaration nur an einer einzigen Stelle in einer einzigen Datei steht (damit man später den Projektnamen leicht an dieser einen Stelle ändern kann)?

2 Inhalt von Objektdateien (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 2c

Betrachten Sie mit `nm` den Inhalt der Objektdateien aus Aufgabe 1 (`math.o, summe.o, differenz.o, input.o`) und erklären Sie, welche Funktionen in welchen Dateien benutzt oder definiert werden.

3 Makefile schreiben (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 2d

Schreiben Sie ein Makefile unter Einsatz von Pattern rules und Variablen für Ihr Programm aus Aufgabe 1.

4 Fehler in Makefile finden (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 2d

Welche Fehler enthält das folgende Makefile für das folgende Programm? Der Einfachheit halber wurden `gcc`-Optionen wie `-Wall` hier weggelassen; das ist also nicht gemeint.

g.h:
```
#include <stdio.h>
extern void g(void);
```

g.c:
```
void g(void) { /* … */ }
```

f.h:
```
extern void f(void);
```

f.c:
```
#include <g.h>
void f(void) { g(); }
```

m.c:
```
#include <f.h>
#include <g.h>
void m(void) { f(); g(); }
```

Makefile:
```
program: m.o f.o g.o
     gcc -o prog m.o f.o g.o

f.o: g.h
     gcc f.c

g.o: stdio.h
     gcc g.c

m.o: f.h g.h
     gcc m.c

m.c: f.h g.h
   gcc -o m.c f.h g.h
```

5 Portierung von Java nach C (0 Punkte)

basiert auf Vorlesung bis einschl. Abschnitt 3

Gegeben sei das folgende Java-Programm, das Sie nach C99 portieren sollen.

```
import java.io.IOException;
import java.util.Scanner;
class Vielfache {
    public static void main(String args[]) throws IOException {
        int a;
```

```
        do {
            System.out.print("Bitte positive ganze Zahl eingeben: ");
            Scanner console = new Scanner(System.in);
            a = console.nextInt();
        } while (a<=0);
        for (int i=0;i<10;i++) {
            for (int j=0;j<10;j++) {
                if (i==0) {
                    System.out.print(" ");
                }
                System.out.print(i*10+j);
                if ((i*10+j)%a==0) {
                    System.out.print("# ");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}
```

Zur Verwendung von `printf` und `scanf` lesen Sie bitte die „Hinweise vor dem Start“ auf Aufgabenblatt A.1.

6 Eine IDE ausprobieren (1 Punkt)

basiert auf Vorlesung bis einschl. Abschnitt 2.e

Diese Aufgabe soll Ihnen zeigen, dass man auch in C/C++ statt auf der Kommandozeile auch in einer IDE arbeiten kann. Falls Sie schon Erfahrung mit einer IDE für die C-Programmierung haben, dürfen Sie die weiter benutzen – Ihre Übungsaufgabe besteht dann darin, die Funktionen dieser IDE anhand eines C-Programms vorzuführen.

Falls Sie noch keine Erfahrung mit einer IDE für die C-Programmierung haben, sollen Sie in dieser Übungsaufgabe Eclipse für C/C++ kennen lernen. Im Rechnerpool ist eclipse mit dem Plugin CDT (C/C++ Development Toolkit) bereits installiert. Wenn Sie auf Ihrem eigenen Rechner arbeiten, installieren Sie eclipse und das Plugin CDT. Sie erhalten weiter unten dazu einige Installationstipps.

Erzeugen Sie danach ein neues C-Projekt:

- Menüpunkt File - New – C Project – Executable, Hello World ANSI C Project. Rechts unter Windows "Toolchain Cygwin GCC" oder unter Linux: "Toolchain Linux GCC" auswählen.
- Namen für das Projekt eingeben, z. B. „Hello“.
- Next, Next, Finish.

Das soeben erzeugte C-Projekt sollten Sie nun konfigurieren:

- Kontextmenü auf dem Projekt im Projektexplorer. Menüpunkt Properties
- Links C/C++ Build wählen und dann den Unterpunkt Settings, Reiter "Tool Settings"
  Cygwin bzw. GCC C Compiler (diese Einstellung ist nur für ein C-Projekt relevant)
  Dialect
  - ISO C99

  Warnings
  - Häkchen bei –pedantic-errors und –Wall

Miscellaneous
- Other flags: -Wstrict-prototypes ergänzen

*Für **C++**-Projekte, die Sie im Verlaufe dieser Veranstaltung später erstellen werden, hier bereits jetzt zur Information:*
Cygwin bzw. GCC C++ Compiler, falls vorhanden (diese Einstellung ist nur für ein C++-Projekt relevant)
Warnings
- *Häkchen bei –pedantic-errors und –Wall*

Miscellaneous
- *Other flags: -std=c++14 ergänzen*

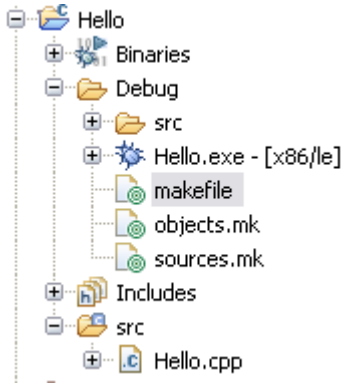
- Wenn Sie Windows und Cygwin verwenden, ist es u. U. erforderlich, folgende Einstellungen vorzunehmen (versuchen Sie dies erst dann, wenn irgendetwas nicht wie gewünscht funktioniert): Kontextmenü auf dem Projekt im Projektexplorer. Menüpunkt Properties. Links C/C++ Build wählen und dann die folgenden Unterpunkte:
  - Settings
    Cygwin C Linker
    Miscellaneous
    - Linker flags: -enable-auto-import
  - Discovery Options:
    Ggf. Discovery profiles scope auf Configuration-wide setzen
    In beiden Konfigurationen (Debug und Release) unten das Compiler invocation command ändern (Browsen zu gcc.exe im cygwin-Pfad …./cygwin64/bin bzw. g++.exe für ein C++-Projekt)

Übersetzen Sie das Projekt (Menüpunkt Project – Build Project oder Build All).
Achten Sie auf eventuelle Fehlermeldungen oder Warnungen im Problems-Fenster. Wenn Sie wollen, dass das Projekt komplett neu übersetzt wird, wählen Sie vorher Project – Clean… . Dies entfernt bei einem früheren Buildvorgang erzeugte Objektcodedateien und alte ausführbare Dateien.

Möglicherweise muss noch eine Lauch Configuration erstellt werden:
- Kontextmenü auf dem Projekt im Projektexplorer. Menüpunkt Properties.
- Run/Debug Settings:
  - Neue Konfiguration anlegen: New C/C++ Application auswählen
  - Ggf. Einstellungen anpassen

Debuggen Sie das Projekt, indem Sie von der Programmier-Perspektive umschalten auf die Debug-Perspektive: Window – Perspective – Open perspective – Debug. Setzen Sie dann einen Haltepunkt auf die Ausgabeanweisung des Hauptprogramms setzen (Rechtsklick links neben entsprechende die Quelltextzeile auf die Zeilennummer – Toggle Breakpoint, dann Menüpunkt Run - Debug). Mit dem Menüpunkt Run – Step Over können Sie nun schrittweise durch das Programm marschieren. Beobachten Sie dabei die (unter Windows evtl. als separates Fenster geöffnete) Konsolenausgabe: Vor dem Step Over sehen Sie noch nichts, aber durch das Step Over wird die Programmzeile mit der Ausgabe ausgeführt, und Sie sehen die Ausgabe im Konsolenfenster. (Wenn Sie immer noch keine keine Konsolenausgabe sehen, ergänzen Sie folgendes Statement am Anfang von main, um die Pufferung der Standardausgabe auszuschalten: `setbuf(stdout, NULL);`)

Wechseln Sie nun zurück zur Perspektive "C/C++" und schauen Sie sich das Makefile an, das Eclipse automatisch für Sie erstellt hat. Es liegt im Unterverzeichnis Debug oder Release Ihres Projektverzeichnisses:



Öffnen Sie dieses mit der rechten Maustaste – Open With – Makefile-Editor. Das automatisch erstellte Makefile nutzt einen Include-Mechanismus von gnumake, um ein Unter-Makefile im Verzeichnis src/subdir.mk einzubinden. Schauen Sie sich auch hier ein wenig um.

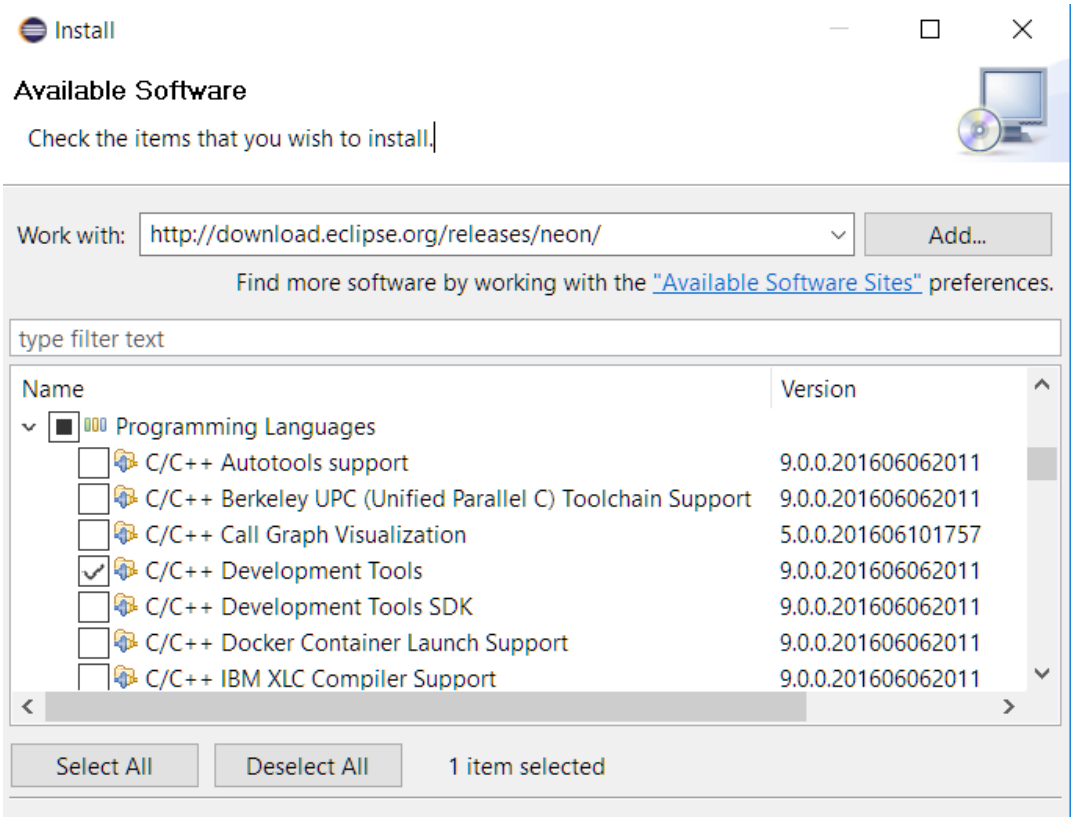
Probieren Sie schließlich aus, was mit den Makefiles geschieht, wenn Sie eine neue C-Datei zu Ihrem Projekt hinzufügen (Kontextmenü auf dem Projekt im Projektexplorer – New – Source file, Next, Dateinamen eingeben, Finish) die nur aus der Zeile `int dummy;` besteht (denn leere Dateien lässt der Compiler nicht zu) und das Projekt neu bauen (Project – Build All).

Nachdem Sie nun einen ersten Blick auf Eclipse CDT werfen konnten, ist Ihnen freigestellt, ob Sie im Verlaufe der weiteren Übungen Eclipse oder eine andere DIE verwenden, oder eigene Makefiles mit einem Texteditor erstellen wollen.

Installationstipps auf dem eigenen Rechner

Auf den Pool-PCs ist Eclipse CDT bereits installiert.
Es gibt mehrere Möglichkeiten zur Installation auf einem eigenen Rechner:

- Laden Sie Eclipse inklusive CDT herunter. Auf der Seite https://www.eclipse.org/downloads/packages/release/2020-09/r/eclipse-ide-cc-developers finden Sie Download-Links für die verschiedenen Betriebssysteme.
- Falls Sie schon ein Eclipse installiert haben, können Sie CDT als Plugin dazu installieren. Am einfachsten erledigen Sie dies über die Softwareupdates in Eclipse:
    1. In Eclipse den Menüpunkt "Help | Install New Software" auswählen
    2. In das Eingabfeld "Work with" folgende URL eingeben: http://download.eclipse.org/tools/cdt/releases/10.0
    3. Aus der Liste "Programming Languages | C/C++ Development Tools" auswählen, "Next" klicken und den Anweisungen folgen
    - So etwa sollte das Auswahlfenster aussehen:



Da die Installation auf den Pool-PCs wahrscheinlich älter ist als eine frische Installation, die Sie selbst erstellen, befinden sich die oben beschriebenen Einstellungen hier möglicherweise an anderen Stellen oder sind etwas anders benannt.

Besondere Installationshinweise für Cygwin unter Windows (habe ich nicht selbst getestet!)

- Unter Windows installieren Sie bitte, falls noch nicht geschehen, cygwin mit dem gnu-Compiler/Debugger (Pakete gcc, gcc-core, gcc-g++, make, gdb).
- Konfigurieren Sie in Eclipse die source lookup paths wie folgt: Menüpunkt Windows – Preferences - C/C++ - Debug - Common Source Lookup Paths - Add: Path Mapping - OK – Name: beliebig – Add - Compilation Path: /cygdrive/c , Local file system path: C:\ - OK.