Hochschule für Angewandte Wissenschaften Hamburg

Hamburg University of Applied Sciences

Praktikumsbericht

PRP2-1

Fabian Liebold, Malte Müller

20. Oktober 2018

Aufgabe 1: Pointer

1.a

Stellen Sie dar, welche Ausgabe das nachfolgend dargestellte Programm erzeugt.

```
 1  H
 2  e
 3  l
 4  l
 5  o
 6  Erster Wert von i war 5! Korrekt ?
 7  H
 8  I
 9  J
10  K
11  L
12  Erster Wert von i war 0! Korrekt ?
```

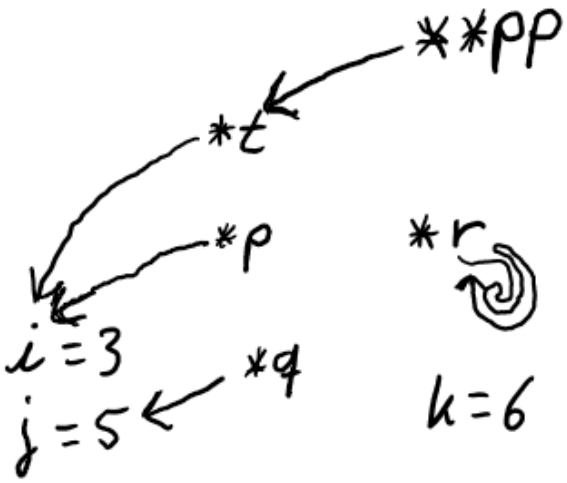
1.b

Geben Sie weiterhin die Hexadezimaldarstellung der Adressen sowie bei Zeigern den Inhalt, auf den er zeigt, an. Erweitern Sie hierfür das Programm durch printf-Anweisungen oder verwenden Sie den Debugger.

```
 1  Adresse i: 010FFCE4
 2  Wert i: 5
 3
 4  Adresse first_i: 010FFCCC
 5  Wert first_i: 0
 6
 7  Adresse first_i_ptr: 010FFCD8
 8  Wert first_i_ptr: 010FFCE4
 9  Wert von Adresse 010FFCE4: 5
10
11  Adresse onechar: 010FFC4B
12  Wert onechar: M
13
14  Adresse strptr: 010FFC54
15  Wert strptr: 010FFC65
16  Wert von Adresse 010FFC65:  World!
```

2.a

Zeichnen Sie graphisch die Zeigerabhängigkeiten auf.



2.b

```
1  wert = (p == i);
```

Wert ist 0. Da bei p nicht der Wert, sondern die Adresse betrachtet wird sind nicht beide echt gleich.

```
1  wert = *p / *q;
```

Hier wird 3/5 gerechnet und nur in ganzen Zahlen dargestellt, was hier 0 ist. Falls kein Leerzeichen zwischen / und *p steht, denkt das Programm, es wird auskommentiert

```
1  wert = *p / *q + 3;
```

Hier wird zu dem Ergebnis 0 von der letzten Berechnung eine 3 hinzuaddiert, deshalb ist das Resultat 3.

```
1  wert = **&p;
```

Wert ist 3. Da der Wert hinter der Adresse, auf die **p zeigt, dargestellt wird.

```
1  wert = *(r = &k);
```

Wert ist 6. r wird auf die Adresse von k gesetzt und als Pointer zeigt r auf die Adresse von k mit dem Wert 6.

```
1  wert = *(r = &k) = *p**q;
```

Wert ist 15. Zuerst wird *r auf die Adresse von k gerichtet, dann wird die Adresse von k auf den Wert 15 (von 3(p)*5(q)) gesetzt und am Ende wir „wert“ diesem Wert gleichgesetzt.

```
1  wert = *p***pp;
```

Wert ist 9. Der Wert auf den p zeigt, wird mit dem Wert auf den q zeigt multipliziert und auf „wert“ gespeichert.

Aufgabe 2: Schach

(1) Aufbau eines Arrays, dass das Schachbrett und die Schachfiguren darstellen kann
Es wurde ein zweidimensionales char-Array (playground) initialisiert. Die Größe des Arrays ist über ein Makro (SIZE) definiert. In unserem Fall entspricht die Größe den Dimensionen eines Schachfeldes. Bei dem Feld A1 fängt das Array bei [0][0] an.
Weiße Felder werden durch ein Leerzeichen dargestellt, für schwarze wird der Ascii-Code 177 verwendet.

(2) String der Schachnotation in einzelne Züge aufschlüsseln
Wurde mit **strtok** (string token) realisiert, durch Festlegen eines Trennzeichens (hier: „/“). Rückgabe ist ein Pointer, der auf die einzelnen gespaltenen Teile zeigt. In einer Schleife wird dieser Prozess wiederholt, bis alle Kommandos abgearbeitet wurden.

(3) Automatisches Nachspielen der gegebenen Züge
Nachdem ein Kommando aus dem Partie-String extrahiert wurde, wird die Funktion **play-str** aufgerufen und die Länge des Kommandos abgefragt. Bei einer Länge von 5 ist keine Spielfigur angegeben und es muss sich um einen Bauern handeln. Bei einer Länge von 6 ist die Spielfigur festgelegt. Als nächstes wird überprüft, ob es sich um einen Setzzug oder Schlagzug handelt. Die Stelle des ausschlaggebenden Characters ist durch die bereits abgeschlossene Abfrage zur Länge des Kommandos eindeutig.
Durch die counter-Variable, die in jedem Schleifendurchlauf erhöht wird, kann überprüft werden, welcher Spieler gerade am Zug ist. Mit der Modulo-Operation wird überprüft, ob die Spielfigur als Groß- oder Kleinbuchstabe in das playground-Array gespeichert werden muss. Anschließend wird die Funktion **update-playgroud** aufgerufen und das Spielfigur wird im Array auf die Zielposition gesetzt. Das Startfeld wird entweder durch ein weißes oder schwarzes Feld ersetzt.

(4) Darstellung des veränderten Schachbrettes auf dem Bildschirm
Ausgegeben wird das Spielfeld mit einer einfachen for-Schleife in der Funktion **print-playground**.

(5,6) Benutzer soll das Spiel weiterspielen
Sobald der Partie-String vollständig gespielt wurde, wird in der main-Funktion gefragt ob der Spieler weiterspielen möchte oder nicht. Dies kann mit einer einfachen Eingabe von ”j”bestätigt und mit n”verneint werden. Falls die Frage mit einem Nein”beantwortet wurde wird das Programm beendet. Falls weitergespielt werden möchte, wird die Funktion **play-input** aufgerufen. Hier wird zunächst die Startposition und die Zielposition abgefragt. Bei der Eingabe von ”00äls Startpostion wird das Spiel beendet. Als nächstes werden die x- und y-Koordinaten der Startposition in Integer umgewandelt und die Figur, welche sich auf der Startposition befindet herausgearbeitet. Anschließend wird die Funktion **update-playground** aufgerufen und das Spielfeld wie bereits in (3) nachgespielt und anschließend auf der Konsole mit der Funktion **print-playground** ausgegeben.