Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 - WS 2022/23

Prof. Dr. Michael Rohs, Tim Dünte, M.Sc., Jan Feuchter, M.Sc.

Präsenzübung 6

Diese Aufgaben sind zur Lösung während der einstündigen Präsenzübung gedacht. Sie können die Aufgaben auf einem mitgebrachten Laptop oder auf Papier lösen.

Aufgabe 1: Operatoren

In C existieren Infix-, Präfix- und Postfix-Operatoren verschiedener Stelligkeit (unäre, binäre und ternäre Operatoren). Die Operatoren haben unterschiedlichen Rang und Assoziativität.

a) Geben Sie je ein Beispiel für einen Infix-, einen Präfix- und einen Postfix-Operator in C.

b) Geben Sie je ein Beispiel für einen unären, einen binären und einen ternären Operator in C.

c) Was ist der Unterschied zwischen `a & b` und `a && b`? (a und b seien vom Typ `int`)

d) Wie ist es möglich, nur mit Hilfe des exklusiv-oder-Operators (^) und ohne Verwendung von Hilfsvariablen, die Inhalte zweier Variablen zu vertauschen? Warum funktioniert das?

e) Was ist der Unterschied zwischen Rang und Assoziativität eines Operators? Welche Rolle spielen Klammern dabei?

f) Warum ist in C der Wert des Ausdrucks `5 / 3 * 2` nicht gleich dem Wert des Ausdrucks `5 * 2 / 3`?

g) Warum ist in C der Wert des Ausdrucks `5 * 2 / 3` nicht gleich dem Wert des Ausdrucks `5 * 2 / 3.0`?

Aufgabe 2: Ausdrücke

Werten Sie folgende Ausdrücke „auf dem Papier“ in einem Auswertungsdiagramm (evaluation diagram) Schritt für Schritt aus. Verwenden Sie hierfür die in der Vorlesung vorgestellte Schreibweise. Beachten Sie den Rang (Präzedenz) der Operatoren.

a) `x = 1 + 2 * 3 + 4`

b) `x = (1 * (y = (1 + 2) * (z = 3)))`

c) `11 > 2 * 3 && (x = 4 * 5 == 6 / 3.0) || 1`

Aufgabe 3: C-Datentypen

C-Datentypen legen eine Interpretation der in Variablen gespeicherten Bitmuster fest und bestimmen, welche Operatoren anwendbar sind.

a) Die Ausgabe des folgenden Programmstücks lautet: 127 + 1 = -128, 255 + 1 = 0. Erklären Sie, warum.

```
signed char a = 127;
signed char b = a + 1;
printi(a); prints(" + 1 = "); printiln(b);
unsigned char u = 255;
unsigned char v = u + 1;
printi(u); prints(" + 1 = "); printiln(v);
```

b) Die Ausgabe des folgenden Programmstücks lautet: 1, 2, 4, 8, 16, 32, 64, -128. Erklären Sie, warum. Erklären Sie insbesondere die `while`-Bedingung und den `<<`-Operator.

```
signed char a = 1;
while (a) { printiln(a); a = a << 1; }
```

c) Erklären Sie den Unterschied zwischen folgenden Deklarationen. Wie können Sie herausfinden, wie viele Bytes werden jeweils von i1 bis i4 auf Ihrem System belegt werden?

```
long i1;
long int i2;
signed long int i3;
signed long i4;
```

d) Wieso genügt es, das niederwertigste Bit einer int-Variablen zu prüfen, um zu entscheiden, ob die Zahl gerade oder ungerade ist? Wie kann diese Prüfung implementiert werden?

Aufgabe 4 (optional): Enumerationen und Mehrfachauswahl

Beim Kartenspiel „Skat“ haben die Farben Karo, Herz, Pik und Kreuz die Grundwerte 9, 10, 11 und 12. Schreiben Sie eine Funktion, die für eine gegebene Farbe den Grundwert zurückgibt. Die Farben sollen als C-Enumeration spezifiziert werden.

Das Programm soll mit geeigneten Testfällen (`test_equal(actual, expected)`) überprüft werden. Es soll ein Dokumentationsstring (Purpose Statement) formuliert werden und als Kommentar (`/* Purpose… @param… @return… */`) vor die Funktion geschrieben werden. Verwenden Sie die Vorgehensweise aus der Vorlesung:

1. **Problem Statement:** Durch den Aufgabentext gegeben. Diskutieren Sie, ob die Problembeschreibung eindeutig ist oder nicht.
2. **Data Definition:** Diskutieren Sie, welche Daten im Programm repräsentiert werden müssen.
3. **Function Name and Parameter List:** Finden Sie einen geeigneten und aussagekräftigen Namen für die Funktion und eine geeignete Parameterliste.

4. **Function Stub and Purpose Statement:** Schreiben Sie einen Funktionsrumpf, der zunächst nur aus einem beliebigen Wert aus dem Wertebereich der Funktion besteht.
5. **Examples with Expected Results:** Überlegen Sie sich einige Beispiele für Werte, die der Funktion übergeben werden könnten und was Sie als Ergebnis erwarten.
6. **Implementation:** Implementieren Sie die Funktion. Überlegen Sie, ob verschiedene Fälle unterschieden werden müssen.
7. **Test and Revision:** Prüfen Sie Ihr Programm an Hand der zuvor aufgeschriebenen Beispiele.