Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WS 2020/2021
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Dr. Martin Frieb, Marius Brendle, Johannes Metzger 11.11.2020

Freiwilliges Programmieren 1 Lösungsvorschlag

In diesem Freiwilligen Programmieren werden Wiederholungsaufgaben zu den Themen 2 und 3 des Vorkurses behandelt. Für jede Teilaufgabe soll ein C-Programm erstellt und dazu die in Worten beschriebenen Anweisungen in C-Anweisungen umformuliert werden. Dabei ist jeder Satz in genau eine C-Anweisung zu überführen.

Erstellen Sie für jede Teilaufgabe jeweils eine C-Datei mit einer eigenen `main`-Funktion. Kompilieren Sie Ihre Programme mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` und führen Sie sie aus (jeweils über ein Kommandozeilen-Programm).

Aufgabe 1.1 *(Ganze Zahlen)*

a)

- Deklarieren Sie eine `int`-Variable `n`.
- Weisen Sie `n` den Wert `4711` zu.
- Geben Sie den Wert von `n` aus.

Lösung:

```c
#include <stdio.h>

int main(void)
{
        int n;
        n = 4711;
        printf("%i\n", n);
        return 0;
}
```

b)

- Deklarieren Sie eine `int`-Variable `n`.
- Weisen Sie `n` den Wert `-INT_MAX` zu.
- Geben Sie den Wert von `n` aus.

Lösung:

```c
#include <stdio.h>
#include <limits.h>

int main(void)
{
        int n;
        n = -INT_MAX;
        printf("%i\n", n);
        return 0;
}
```

c)

- Deklarieren Sie eine `int`-Variable `n`.
- Erzeugen Sie mit `rand` eine ganze Zufallszahl zwischen 10 und 20 (jeweils eingeschlossen) und weisen Sie `n` den Wert der Zufallszahl zu
- Geben Sie den Wert von `n` aus.

Lösung:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
        int n;
        srand(time(NULL));
        n = (rand() % 11) + 10;
        printf("%i\n", n);
        return 0;
}
```

Aufgabe 1.2 *(Arithmetische Rechenausdrücke)*

a)

- Deklarieren Sie drei `int`-Variablen `a`, `b` und `c`.
- Weisen Sie `a` den Wert `12` zu.
- Weisen Sie `b` den Wert `21` zu.
- Weisen Sie `c` den Wert der Addition von `a` und `b` zu.
- Geben Sie den Wert von `c` aus.

Lösung:

```c
#include <stdio.h>

int main(void)
{
        int a, b, c;
        a = 12;
        b = 21;
        c = a + b;
        printf("%i\n", c);
        return 0;
}
```

b)

- Deklarieren Sie drei `int`-Variablen `a`, `b` und `c`.
- Weisen Sie `a` den Wert `35` zu.

- Weisen Sie `b` den Wert `9` zu.
- Weisen Sie `c` den Wert des Rests der (ganzzahligen) Division von `a` und `b` zu.
- Geben Sie den Wert von `c` aus.

Lösung:

```c
#include <stdio.h>

int main(void)
{
        int a, b, c;
        a = 35;
        b = 9;
        c = a % b;
        printf("%i\n", c);
        return 0;
}
```

c)

- Deklarieren Sie eine `int`-Variable `d`.
- Weisen Sie `d` den Wert `22` zu.
- Verdoppeln Sie dann den Wert von `d`.
- Geben Sie den Wert von `d` aus.

Lösung:

```c
#include <stdio.h>

int main(void)
{
        int d;
        d = 22;
        d = d * 2;
        printf("%i\n", d);
        return 0;
}
```

Aufgabe 1.3 *(Auswertungsreihenfolge und Overflow)*

a)

- Deklarieren Sie drei `int`-Variablen `a`, `b` und `c` und weisen Sie `a` den Wert `8`, `b` den Wert `7` und `c` den Wert `6` zu.
- Geben Sie den Wert der folgenden Rechnung aus: Addieren Sie zuerst `a` und `b`, und multiplizieren Sie dann das Ergebnis mit `c`.

Lösung:

```c
#include <stdio.h>

int main(void)
{
        int a = 8, b = 7, c = 6;
        printf("%i\n", (a + b) * c);
        return 0;
}
```

b)

- Deklarieren Sie eine `int`-Variable `d`.

- Weisen Sie ihr den Wert `INT_MIN` zu.
- Erniedrigen Sie dann den Wert von `d` um `1`.
- Geben Sie den Wert von `d` aus.

Lösung:

```c
#include <stdio.h>
#include <limits.h>

int main(void)
{
        int d;
        d = INT_MIN;
        d = d - 1;
        printf("%i\n", d);
        return 0;
}
```

c)

- Deklarieren Sie eine `int`-Variable `d`.
- Weisen Sie ihr den Wert `INT_MAX` zu.
- Addieren Sie dann `INT_MAX` auf den Wert von `d` und weisen Sie das Ergebnis wieder `d` zu.
- Geben Sie den Wert von `d` aus.

Lösung:

```c
#include <stdio.h>
#include <limits.h>

int main(void)
{
        int d, e;
        d = INT_MAX;
        d = d + INT_MAX;
        printf("%i\n", d);
        return 0;
}
```

Aufgabe 1.4 *(ASCII-Zeichen)*

a)

- Deklarieren Sie eine `char`-Variable `e`.
- Weisen Sie ihr den Wert `'\"'` zu.
- Geben Sie den Wert von `e` als Zeichen aus.
- Geben Sie den Wert von `e` als ganze Zahl aus.

Lösung:

```c
#include <stdio.h>

int main(void)
{
        char e;
        e = '\"';
        printf("%i\n", e);
        printf("%c\n", e);
        return 0;
}
```

b)

- Geben Sie das ASCII-Zeichen mit dem ASCII-Code `64` aus (ohne in der ASCII-Tabelle nachzusehen).

Lösung:

```c
#include <stdio.h>

int main(void)
{
        printf("%c\n", 64);
        return 0;
}
```

c)

- Erzeugen Sie mit einem `printf`-Aufruf die folgende Ausgabe:

```
*   *
 * *
*   *
 * *
*   *
```

Lösung:

```c
#include <stdio.h>

int main(void)
{
        printf("*   *\n * *\n*   *\n * *\n*   *\n");
        return 0;
}
```

Aufgabe 1.5 *(Zeichenfunktionen)*

a) Geben Sie jeweils in einer eigenen Zeile den Rückgabewert des Funktionsaufrufs `isupper(c)` für verschiedene Zeichen `c` aus:

- Für das Zeichen `'v'`
- Für das Zeichen `'W'`
- Für das Zeichen `'?'`
- Für das Zeichen `'1'`
- Für das Zeichen `'\v'`

Lösung:

```c
#include <stdio.h>
#include <ctype.h>

int main(void)
{
        printf("%i\n", isupper('v'));
        printf("%i\n", isupper('W'));
        printf("%i\n", isupper('?'));
        printf("%i\n", isupper('1'));
        printf("%i\n", isupper('\v'));
        return 0;
}
```

b) Geben Sie jeweils in einer eigenen Zeile den Rückgabewert des Funktionsaufrufs `toupper(c)` für verschiedene Zeichen `c` als Zeichen zwischen zwei Hochkommas (z. B. `'*'`) aus:

- Für das Zeichen `'v'`
- Für das Zeichen `'W'`
- Für das Zeichen `'?'`
- Für das Zeichen `'1'`
- Für das Zeichen `'\v'`

Lösung:

```c
#include <stdio.h>
#include <ctype.h>

int main(void)
{
        printf("'%c'\n", toupper('v'));
        printf("'%c'\n", toupper('W'));
        printf("'%c'\n", toupper('?'));
        printf("'%c'\n", toupper('1'));
        printf("'%c'\n", toupper('\v'));
        return 0;
}
```

c) Geben Sie jeweils in einer eigenen Zeile den Rückgabewert des Funktionsaufrufs `isalnum(c)` für verschiedene Zeichen `c` aus:

- Für das Zeichen `'v'`
- Für das Zeichen `'W'`
- Für das Zeichen `'?'`
- Für das Zeichen `'1'`
- Für das Zeichen `'\v'`

Lösung:

```c
#include <stdio.h>
#include <ctype.h>

int main(void)
{
        printf("%i\n", isalnum('v'));
        printf("%i\n", isalnum('W'));
        printf("%i\n", isalnum('?'));
        printf("%i\n", isalnum('1'));
        printf("%i\n", isalnum('\v'));
        return 0;
}
```