Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TU Kaiserslautern
Fachbereich Informatik

Übungsblatt 5: Programmieren in C (WS 2019/20)

Abgabe: Montag, 09.12.18, 12:00

Aufgabe 1 *Präsenzaufgabe: Zeiger (keine Abgabe)*

1. Ergänzen Sie die folgenden Lücken:
   (a) Ein Zeiger repräsentiert die Adresse und den _____ eines Objekts.
   (b) Angenommen, die Zeiger `p1` und `p2` sind vom selben Typ. Wie lautet die Zuweisung, um `p2` auf dasselbe Objekt wie `p1` zeigen zu lassen?
   (c) Wenn `k` eine int-Variable ist, hat der Ausdruck `&k` den Typ _____.
   (d) Die Adresse einer Zeigervariablen `ptr` liefert der Ausdruck _____. Skizzieren Sie die Situation zum Verständnis!
   (e) Gegeben sei eine double-Variable `y`. Die Definition eines Zeigers `dptr`, der so initialisiert wird, dass es auf `y` zeigt, lautet _____.
   (f) Gegeben sei der Zeiger `iptr`, der auf die int-Variable `i` mit Wert 5 zeigt. Was bewirkt der Ausdruck `2**iptr` ?
   (g) Der float-Zeiger `fptr` enthalte die Adresse 20000. Nach der Zuweisung `fptr = fptr + 1;` enthält `fptr` die Adresse _____.
   (h) Wieso ist folgende Anweisung fehlerhaft?

```
int a = 2, int b = 4, *p1 = &a, *p2 = &b;
printf("%d\n", *(p1+p2));
```

2. Berechnen Sie die Werte in folgendem Programmfragment der Reihe nach (auf Papier):

```
int a = 8, b = 4, c = 2, d = 3;
a = a + 1;          // a=
a = a + b * 4;      // a=
a = a / 2;          // a=
a = a % 2;          // a=

b = 5;
a = b / 2;          // a=
c = b % 2;          // c=
b = b * -3;         // b=
d = d % 3;          // d=
c = c + (b * d + 4);  // c=
a = --b + d++;     // b=    , a=    , d=

b = 3; c = 2;  d = 6;
a = (b + 2) * 2 * c + 1;     // a=
a = ++b * d++ * ++c * (-1); // a=
b = 1 - --b;                 // b=

int *e, *f;
e = &b, f = &a;

a = *e * *f; // a =   , b =
*e -= 3;     // a =   , b =
b = a * b;   // a =   , b =
e = f;       // a =   , b =
b = *f - 13; // a =   , b =
a = *e % *f; // a =   , b =
```

3. Laden Sie das Programm `operatoren.c` herunter. Starten Sie den Debugger und steppen Sie durch das Programm hindurch. Verwenden Sie den Debugger um die Werte der Variablen `a`,`b` und `c` an den einzelnen Programmstellen zu ermitteln und so Ihre Lösung zu kontrollieren.

Aufgabe 2 *Präprozessor I (4 Punkte)*

Abgabe: `praeprocessor.c` Gegeben seien folgende beide Dateien:

`praeprocessor.c`

```c
#include <stdio.h>

#define def

int main(void)
{
    int b = 5;
    int c = 2;

#define b c
#include "andere_datei.c"

    printf("\nZeile %d", __LINE__);

#ifdef def
    printf("\ndef defined\n");
#else
    printf("\ndef not defined\n");
#endif

    return 0;
}
```

`andere_datei.c`

```c
printf("b = %d\n", b);

#undef def
```

Wie sieht das Programm aus nach dem Präprozessor-Schritt?

Aufgabe 3 *Präprozessor II (4 Punkte)*

1. Gegeben sei folgendes Programm:

   ```c
   #include <stdio.h>
   #define SQUARE(y) y*y

   int main()
   {
       int a = 5;
       printf("Square a: %d", SQUARE(a+1));
       return 0;
   }
   ```

   Welche Ausgabe wird beim Ausführen dieses Programms erzeugt? Erläutern Sie Ihre Antwort!

2. Gegeben sei folgendes Programm:

   ```c
   #include <stdio.h>

   int main()
   {
       int y = 5;
       int x = 3;
       #ifdef x
       printf("x: %d", x);
       #else
       printf("y: %d", y);
       #endif
   ```

```
    return 0;
}
```

Welche Ausgabe wird beim Ausführen dieses Programms erzeugt? Erläutern Sie Ihre Antwort!

Aufgabe 4 *Rechnen mit Zahlen (4 Punkte)*

Abgabe: `unsigned.c`

Gegeben sei das folgende Programm:

```
#include <stdio.h>
int main (void)  {
   signed char a = 119;
   signed char b = 19;
   signed char c;

   c = a + b;

   printf("%i\n", c);
   return 0;
}
```

1. Überlegen Sie zuerst, was das Programm ausgibt!
2. Implementieren Sie das Programm! Erklären Sie das Ergebnis.
3. Ändern Sie den Datentyp der Variablen `a`, `b` und `c` in `unsigned char`. Implementieren Sie das Programm erneut als `unsigned.c`! Erklären Sie das neue Ergebnis.

Aufgabe 5 *Speicherrepräsentierung von Daten (6 Punkte)*

Abgabe: `speicher.c`

In der Vorlesung wurde erläutert, dass der Speicherbedarf für Integer-Zahlen (`int`, `short`, `long`, `long long`) vom Compiler abhängt. Diese Wahl erfolgt dabei insbesondere in Abhängigkeit von Prozessor und Betriebssystem.

- Schreiben Sie ein C-Programm, welches die tatsächlichen Speichergrößen für die einzelnen Integervarianten Ihres Compilers anzeigt. Man übergibt dafür dem `sizeof()` Befehl den Datentyp und erhält die Speichergröße in Byte. Wie groß ist der jeweilige Wert in Bit für char, short, int, long, float und double? Das Programm soll auch die Wertebereiche, d.h. die kleinste und größte Zahl, des jeweiligen ganzzahligen Datentyps (z.B. `INT_MAX`, `INT_MIN` und `UINT_MAX`) ausgeben. Die Namen der Konstanten finden Sie in `limits.h`.
- Ergänzen Sie das Programm, so dass es Ihnen die Werte und die Adressen folgender Variablen `a`,`b`,`c` und `d` ausgibt, wie sie im Speicher des Rechners abgelegt sind. Wählen Sie passende Formatelemente für die Ausgabe. Ermitteln Sie zusätzlich die Größe der Adressen in Bit (d.h. den Speicherbedarf für einen Pointer).

  ```
  int a = 46, b = 89;
  double c = 5.3, d=2017.411;
  ```

  Deklarieren Sie geeignete Pointer für die Variablentypen. Weisen Sie Ihnen die Werte der Variablen zu und geben Sie den Wert, die Speicheradresse und die Größe des Pointertyps aus.

  Hinweis: Die Größe der Datentypen und damit der Wertebereich können in Ihrem System abweichen. Außerdem sind die Werte der Pointer in jedem Programmlauf verschieden.

  Die Ausgabe soll folgendermaßen aussehen:

  ```
  sizeof(char)      = 1 Byte; ([-128 ... 127])
  sizeof(short)     = 2 Byte; ([-32768 ... 32767])
  sizeof(int)       = 4 Byte; ([-2147483648 ... 2147483647])
  sizeof(long)      = 8 Byte; ([-9223372036854775808 ... 9223372036854775807])
  sizeof(long long) = 8 Byte; ([-9223372036854775808 ... 9223372036854775807])
  ```

```
sizeof(float)     = 4 Byte;
sizeof(double)    = 8 Byte;
Wert von a: 46, Speicheradresse von a:0x7ffee8c94718
Wert von b: 89, Speicheradresse von b:0x7ffee8c94714
Wert von c: 5.300000, Speicheradresse von c:0x7ffee8c94708
Wert von d: 2017.411000, Speicheradresse von d:0x7ffee8c94700
```