# RA FS 18 Repetitionsserie: Teil C

Qiyang Hu, Givi Meishvili, Adrian Wälchli, Mauro Kiener

Die Repetitionsserie dient der individuellen Prüfungsvorbereitung und muss nicht abgegeben werden. Für Fragen steht im ILIAS jederzeit ein Forum zur Verfügung.
Viel Spass!

## 1 C-Programmierung

### 1.1 Pointer (1 Punkt)

Angenommen `x` werde im untenstehenden Programm an der Adresse `0xbffff97c` initialisiert (`0x` bedeutet, dass die Adresse `bffff97c` hexadezimal ist).

Auf welche Adresse im Speicher zeigt `px` nach Ausführung des Programms? Gehen Sie von einem 32-Bit Prozessor aus.

```
1 main() {
2   short x, *px;
3   px = &x;
4   px++;
5 }
```

### 1.2 StringCopy (1 Punkt)

Der folgende Code-Ausschnitt zeigt eine typische Implementation der `strcpy`-Funktion. Welche Probleme können auftreten?

```
1 char *strcpy(char *s1, const char *s2) {
2   char *dst = s1;
3   const char *src = s2;
4   while( (*dst++=*src++)!='\0');
5   return s1;
6 }
```

### 1.3 Sizeof

Welche Ausgabe erzeugt das folgende Codestück?

```
1 char a[10];
2 char* b = a;
3 printf("%i %i\n",sizeof(a), sizeof(b));
```

### 1.4 Arrays initialisieren

Wieso kann in C eine Array-initialisierende Funktion nicht folgendes machen?

```
1 int* init() {
2   int i[32];
3   return i;
4 }
```

### 1.5 Pointerarithmetik

Wieso gibt `int *a = address; a++;` und `char *a = address; a++;` nicht das selbe Resultat?

## 1.6 Variablentypen

Welchen Typ haben die Variablen aus `int* a, b`?

## 1.7 Operatorenpräzedenz

In welcher Reihenfolge werden die Ausdrücke in `*&a++` evaluiert?

## 1.8 Best coding practices

Warum ist `if (3 == a) {}` besser als `if (a == 3) {}`?

## 1.9 Preprocessor Fun

Welche Ausgabe erzeugt das folgende Codestück?

```
1  #define MIN(a,b) ((a) < (b) ? (a) : (b))
2
3  int a=3;
4  int b=4;
5  int m=MIN(a++,b++);
6  printf("min(%i, %i) = %i\n", a, b, m);
```

## 1.10 Datenstrukturen in C

Um was für eine Struktur handelt es sich bei `tr`? Stellen Sie `tr* a` graphisch dar und beschreiben Sie Output und Ablauf des folgenden Programms

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  typedef struct tr_ {
5    char n;
6    struct tr_ *l;
7    struct tr_ *r;
8  } tr;
9
10 tr* ct(char n, tr *l, tr *r) {
11    tr *t = malloc(sizeof(t));
12    t->n = n; t->l = l; t->r = r;
13    return t;
14 }
15
16 void rec(tr *t, void (*o)(tr *)) {
17    if (t) {
18       tr *l = t->l;
19       tr *r = t->r;
20       o(t);
21       rec(l, o);
22       rec(r, o);
23    }
24 }
25
26 void p(tr *t) {
27   printf("%c", t->n);
28 }
29
30 void s(tr *t) {
31   tr* tmp = t->l;
32   t->l = t->r;
33   t->r = tmp;
34 }
35
```

```
36 void fr(tr *t) {
37   free(t);
38 }
39
40 int main() {
41   tr* g = ct('g',NULL,NULL);
42   tr* f = ct('f',NULL,NULL);
43   tr* e = ct('e',NULL,NULL);
44   tr* d = ct('d',NULL,NULL);
45   tr* c = ct('c',f,g);
46   tr* b = ct('b',d,e);
47   tr* a = ct('a',b,c);
48   rec(a, &p);
49   printf("\n");
50   rec(a, &s);
51   rec(a, &p);
52   printf("\n");
53   rec(a, &fr);
54   return EXIT_SUCCESS;
55 }
```

### 1.11 Mehr Fehlersuche

Das folgende Programm gibt den Output `(null) (null), 0`. Wo liegt der Fehler? Korrigieren Sie die Fehler, so dass das Programm die intendierte Funktion erbringt.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #define MAX_STRING_LENGTH 40
5
6  typedef struct {
7    char *firstName;
8    char *lastName;
9    unsigned int age;
10 } Person;
11
12 void assignValues(Person p, const char *firstName, const char *lastName, unsigned int age) {
13   p.firstName = firstName;
14   p.lastName = lastName;
15   p.age = age;
16 }
17
18 int main(int argc, char** argv) {
19   Person p = *(Person*)malloc(sizeof(Person));
20   assignValues(p, "Sandra", "Muster", 33);
21   printf("%s %s, %i\n", p.firstName, p.lastName, p.age);
22   return EXIT_SUCCESS;
23 }
```