Klausur 2 – 25.03.2015

(90 Min., keine Hilfsmittel
(außer einfachem, nicht-programmierbarem Taschenrechner))

Name, Vorname: [redacted] Matrikelnr.: [redacted]

[letzte Prüfungsmöglichkeit wg. (zutreffenden falls ankreuzen): ❑ 3. Versuch]

1. Aufgabe
Erläutern Sie kurz, was man beim Programmentwurf unter *Modularisierung* versteht und warum diese angestrebt wird. 4 /4

2. Aufgabe
Welches ist die kleinste und die größte (Dezimal-)Zahl, die mit einem 16-bittigen integer-Datentyp üblicherweise darstellbar ist? Begründen Sie Ihre Angabe. 1,5 /2

3. Aufgabe
   a) Erläutern Sie den Unterschied der Parameterübergabe zwischen den beiden Funktionsaufrufen 4 /4

```
double e, arg = 2.;
unsigned int r, par = 3;
e = func1(arg);
r = func2(&par);
```

   und wozu dies nützlich ist.
   b) Notieren Sie die jeweils passenden Prototypen beider Funktionen. 2 /2

4. Aufgabe
Begründen Sie kurz, warum der folgende – syntaktisch korrekte – Programmausschnitt fehlerhaft ist: 3,5 /4

```
typedef struct { float x, y; } vector_t;
vector_t* add_vectors(vector_t *v1, vector_t *v2) {
      vector_t sumvec = { v1->x + v2->x, v1->y + v2->y };
      return &sumvec;
}
```

5. Aufgabe
Geben Sie die formatierte Ausgabe des folgenden Programms an: 0 /7

```
#include <stdio.h>
int main() {
      int i = 0, n = 33;
      while ( n > 1 ) {
            if ( n % 2 == 0 )
                  n /= 2 + 1;
            else
                  n = n * 2 - 2;
            i++;
            if (n % 10 == 0) continue;
            printf("%2d: %2d\n",i,n);
      }
      return 0;
}
```

Übertrag: 15 /23

Übertrag: 15 /23

6. Aufgabe — 7 /8

Schreiben Sie eine Funktion, die als Parameter ein eindimensionales Feld von positiven Ganzzahlen und seine (belegte) Länge erhält. Ihre Funktion soll die *Summe* der Elemente im Feld zurückgeben, die jeweils das Dreifache ihres Vorgänger-Elementes betragen.

7. Aufgabe — 9 /12

Gegeben sei folgende Variablendefinition mit Initialisierungsliste:

```
computer_t meinRechner = {
        3387651234,                     /* Modellnr. */
        Notebook,                       /* Geräteklasse */
        { "Drintel P7001",              /* CPU-Bez., -Daten: */
                {     8,                /* Anzahl Kerne */
                      4.7,              /* max. Takt [GHz] */
                      45.9              /* TDP [Watt] */,
                      "GAL 555",        /* Bez. Sockel */
                },
        },
        450.5,                          /* Festplattengröße [GB] */
        1199.90                         /* Preis [€] */
};
```

Schreiben Sie dazu passende Struktur-Definitionen, so dass diese Initialisierungsliste mit ihren unterschiedlichen Datentypen gültig wird.

8. Aufgabe — 13,5 /15

Gegeben seien folgende Definitionen:

```
enum { MAXBTL = 1000 };
typedef enum { Stahl, Messing, Aluminium, Kupfer } werkstoff_t;
typedef struct {
        unsigned int ID;
        char *Bezeichnung;
        werkstoff_t Material;
        float Stueckpreis;
} bauteil_t;
bauteil_t Bauteile[MAXBTL];
```

Schreiben Sie eine Funktion, die das `Bauteile`-array, seine (belegte) Länge und einen `ReferenzPreis` erhält (alle Werte werden als sinnvoll vorbelegt vorausgesetzt). Die Funktion soll die *Anzahl* aller `Messing-Bauteile` zurückliefern, deren `Stueckpreis` unterhalb des `ReferenzPreises` liegt und den *Mittelwert* ihrer `Stueckpreise`.

2,3

06.04.2015, ka

44,5 /58

[redacted]

(1)

Unter Modularisierung versteht man das Auslagern von Funktionalitäten in sogenannte Module. Dies wird angestrebt, da Module die Übersichtlichkeit des Programmes verbessern, die Wiederholung von Codesegmenten verhindern (DRY = don't repeat yourself) und die Möglichkeit der einzelnen Compilierung dieser es erlaubt in Teams an unterschiedlichen Teilen des Programmes zu arbeiten. + verbesserte Wartbarkeit

(2) $2^{16} = 65536$

$65536 - 1 = 65535$?

Die kleinste Zahl ist -32768 und die größte 32767. warum ?!

(3) a) Call by Value (e = func1(arg);):

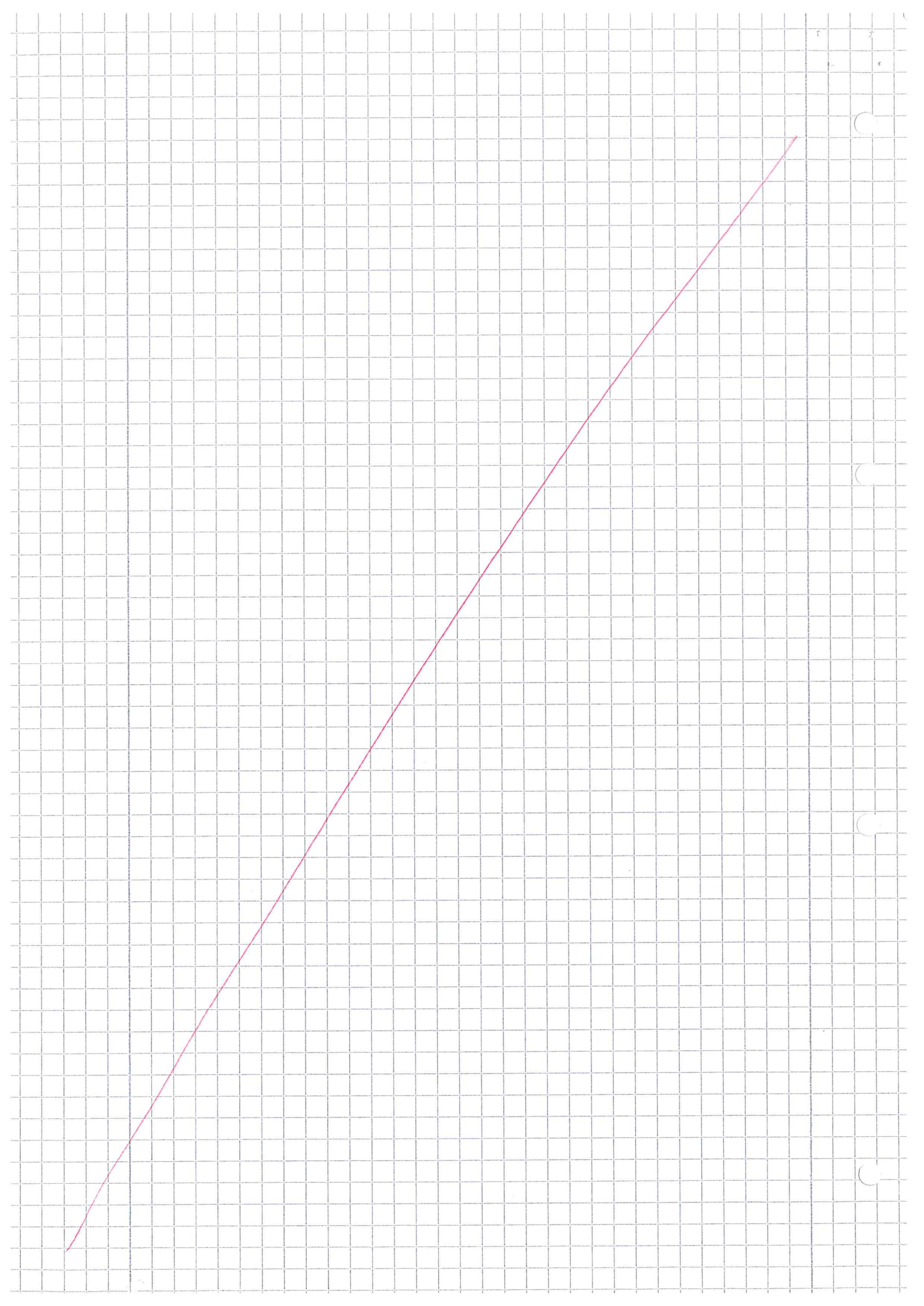
Bei dieser Parameterübergabe wird die Kopie einer Variable an die Funktion übergeben. Das hat zur Folge, dass eine Veränderung dieser den Aktualparameter die Originvariable nicht verändert. Des weiteren existiert die Kopie nur lokal innerhalb der Funktion. Dies kann z. B. zum Schutze der Originalvariable vor Veränderungen genutzt werden. Um die Natur der Kopie zu verdeutlichen, wird sie mit "const" im Prototyp initialisiert.

Fachsprache!

Call by reference (r = func2(&par);):

Bei dieser Parameterübergabe wird ein Pointer auf eine Variable übergeben, was es ermöglicht diese durch Dereferenzierung d. z. aus der Funktion heraus zu verändern. Da nur eine Adresse übergeben wird, sind Übergaben per Call-by-reference grundsätzlich schneller und das Programm verbraucht insgesamt auch weniger Speicherplatz.

1/



3.)

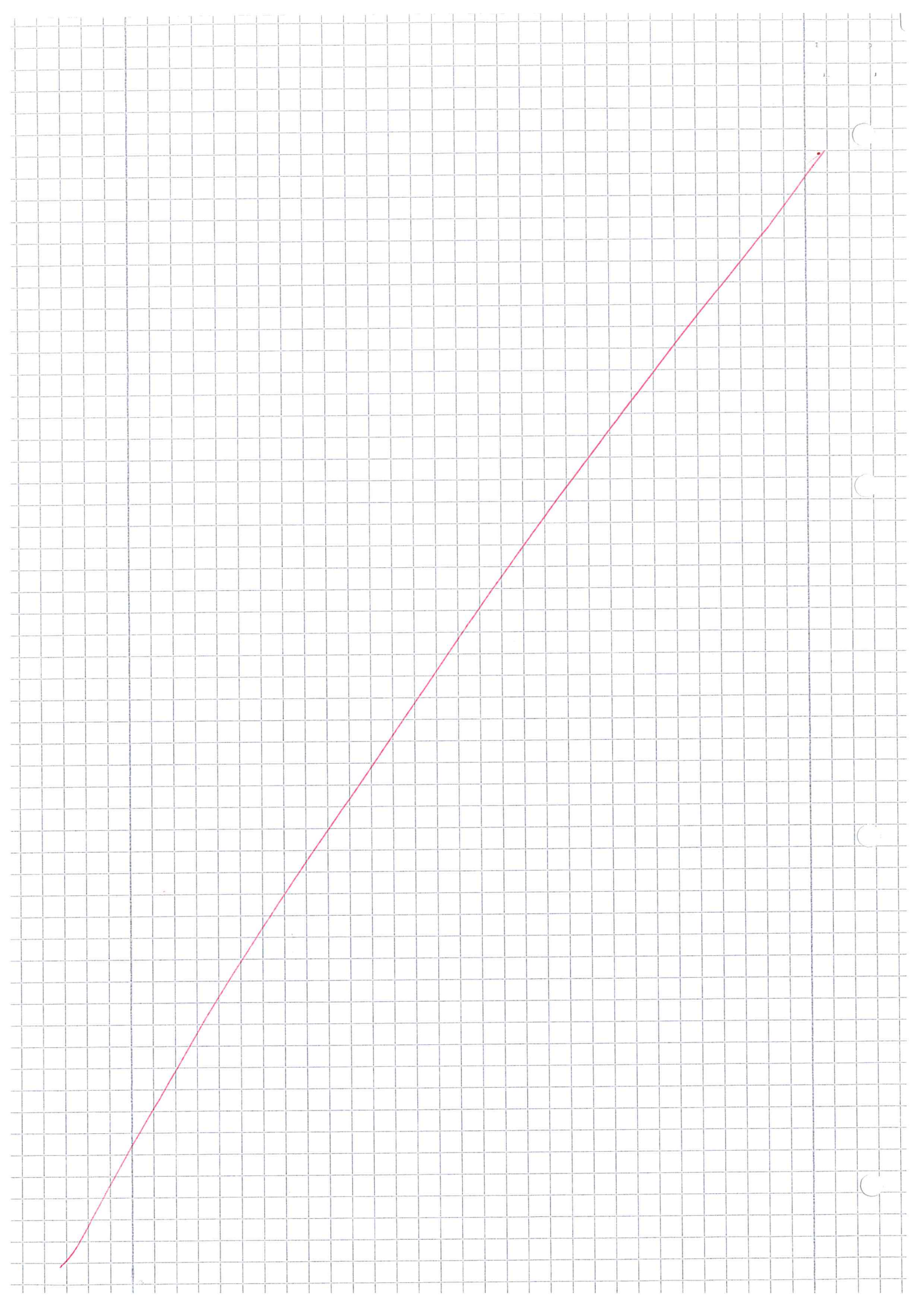
double func1(const double arg);
unsigned int func2(unsigned int* par);

4.)

Dieser Programmausschnitt ist Fehlerhaft, da von der Funktion die Adresse für eine, innerhalb der Funktion deklariertes, struct, zurück gegeben wird. Wenn die Funktion beendet ist existiert dieses lokale struct nicht mehr, da der stackfra der Fkt. freigegeben wird und ein Zugriff auf die Adresse würde ins "leere" gehen und im schlimmsten Fall ein Zugriff auf anderweitig genutzten Speicher bedeuten und kein Absturz evtl. führen.

5.)

i	n	n%2	n%10	printf
0	33	1		
1	64			␣1:␣64
2	32			2:␣32
1	62			␣1:␣62
2	32			
1	62			␣1:␣62
2	20			␣2␣20
3	6			␣3:␣␣6
4	2			␣4:␣␣2
5	0			␣5:␣␣0



[redacted]

6) ~~(int sum_f(~~

```
unsigned int sum_f(unsigned int feld[], const int LEN){
    int i;
    unsigned int sum = 0;
    for(i=0; i<LEN-1; i++){
        if( feld[i] == feld[i+1] * 3){ sum += feld[i+1];
    }
    return sum;
}
```

+= ?!

f

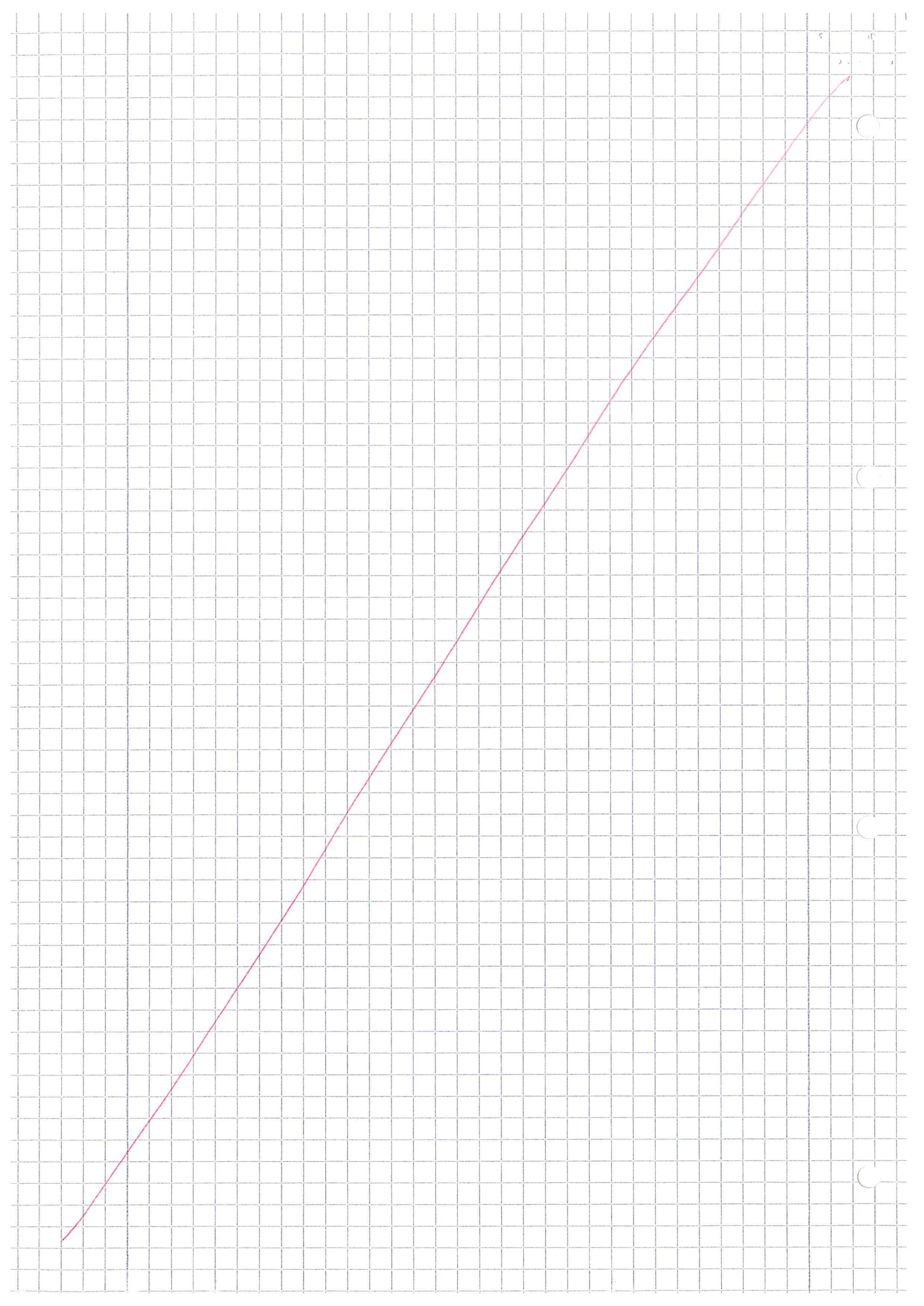
gerade verkehrt herum!

7)

```
typedef struct {
    short Kern;
    float Takt;
    float TDP;
    char Sockel[20];
} cpudat_t;
typedef struct {
    char cpu_typ[20];
    cpudat_t cpu1;
} cpu_t;
typedef struct {
    int mod;
    char geraet;
    cpu_t cpu;
    float festpl;
    float preis;
} computer_t;
```

(für char xy[x] kann auch geschrieben werden char *xy)

Die Geräteklasse kann kein char sein → enum-Defin. erforderlich!



[redacted]

(8)

```
typedef struct {
    int count;
    float mittel;
} rueck;
```

~~(rueck messing_f (Bauteile feld bauteile_[], int LEN) {~~

~~rueck messing_f (Baute~~

```
rueck messing_f ( bauteile_t Bauteile [ ],
                  const int LEN,
                  const float (*x, stueckp)) {
```

x = refp

```
    (int count = 0;)
    rueck Erg = {0.0};
    int i;
    for (i = 0; i < LEN; i++) {
        if (Bauteile[i].Material == Messing &&
            Bauteile[i].Stueckpreis < refp) {
                Erg.mittel += Bauteile[i].Stueckpreis;
                Erg.count += i;
```

f: } if (Erg.count > 0)

```
    Erg.mittel /= Erg.count;
    return Erg;
}
```