CPP Labor 1 – Ü3 – Patrick Wels (922434) und Philipp Meißner (922432)

Aufgabe 1)

```
#include <stdlib.h> // Includes random-functions
#include <time.h>   // Includes current time for srand-seed
#include <iostream> // Input and outputs
using namespace std;

int main () {
  srand(time(NULL));
  float sum, mw, mini, maxi, counter, curr_num;

  // We need 5 runs
  for ( int i = 1; i <= 5; i++ ) {

    // Reset all variables per run.
    sum       = 0;
    mw        = 0;
    mini      = 1;
    maxi      = 0;
    counter   = 0;

    while ((curr_num = (static_cast<float> (rand())) / RAND_MAX) <= 0.85) {

      counter++;
      sum += curr_num;

      // We have a new maximum
      if ( curr_num > maxi) {
        maxi = curr_num;
      }

      // We have a new minimum
      if ( curr_num < mini ) {
        mini = curr_num;
      }

      // Print our current number
      cout << "[" << i << "] ";
      cout << curr_num << endl;

    }
    // Make sure there is numbers to calculate something
    if (counter) {
      mw = sum / counter;
      cout << "Zufallswerte insgesamt:\t" << counter << endl;
      cout << "Minimum:\t\t" << mini << endl;
      cout << "Maximum:\t\t" << maxi << endl;
      cout << "Mittelwert:\t\t" << mw << endl << endl;
    } else {
      cout << "Die erste Zufallszahl war bereits groß genug, um die Schleife zu unterbrechen." <<
```

```
endl << endl;

    }
 }

  return 0;
}
```

Ausgabe A1:
[1] 0.553195
[1] 0.255778
[1] 0.251361
[1] 0.0121495
[1] 0.307854
[1] 0.422661
[1] 0.568959
Zufallswerte insgesamt: 7
Minimum: 0.0121495
Maximum: 0.568959
Mittelwert: 0.338851

[2] 0.523872
[2] 0.518638
[2] 0.827108
[2] 0.774036
[2] 0.578233
[2] 0.287582
[2] 0.372912
[2] 0.797159
[2] 0.447041
[2] 0.159202
[2] 0.158232
[2] 0.81269
Zufallswerte insgesamt: 12
Minimum: 0.158232
Maximum: 0.827108
Mittelwert: 0.521392

[3] 0.175756
[3] 0.773048
[3] 0.0508516
[3] 0.241166
[3] 0.256338
Zufallswerte insgesamt: 5
Minimum: 0.0508516
Maximum: 0.773048
Mittelwert: 0.299432

Die erste Zufallszahl war bereits klein genug, um die Schleife zu unterbrechen.

[5] 0.11288

```
[5] 0.2238
[5] 0.827959
[5] 0.666075
[5] 0.479578
[5] 0.0793197
[5] 0.678224
[5] 0.787432
[5] 0.501981
[5] 0.247183
[5] 0.73583
[5] 0.0258527
[5] 0.765821
[5] 0.562938
[5] 0.799889
[5] 0.344054
Zufallswerte insgesamt:        16
Minimum:                0.0258527
Maximum:                0.827959
Mittelwert:             0.489926
```

Aufgabe 2)

```
#include <iostream> // In- and Output
#include <string>   // String for our structure
using namespace std;

// Prototype declaration
void showSizes();

// Students struct
struct student {
  int id;
  string name, studiengang;
  float note;
  bool bestanden;

  // Standardconstructor
  student(){};
  // Constructor
  student (int i, string na, string sg, float no, bool b) : id(i), name(na), studiengang(sg), note(no),
bestanden(b) {}

  // Print the students information
  void showStudent() {
    cout << "ID:\t\t"        << id          << endl;
    cout << "Name:\t\t"      << name        << endl;
    cout << "Studiengang:\t" << studiengang << endl;
    cout << "Note:\t\t"      << note        << endl;
    cout << "Bestanden:\t"   << bestanden   << endl   << endl;
  }
};
```

```
int main () {
  // Print information about datatypes
  showSizes();

  // initialized students
  struct student studentarr[3] = {
    student(1, "Friedrich", "BWL", 2.0, true),
    student(2, "Christian", "VWL", 3.4, true),
    student(3, "Carina", "KoDe", 1.0, true)
  };
  // Print first three students from array
  for (int i = 0; i < 3; i++) {
    studentarr[i].showStudent();
  }

  // Manually created students
  //struct student peter(5, "Peter", "TMM", 2.3, true);
  struct student karl, peter;
  karl.id = 4;
  karl.name = "Karl";
  karl.studiengang = "INI";
  karl.note = 4.7;
  karl.bestanden = false;

  peter.id = 5;
  peter.name = "Peter";
  peter.studiengang = "TMM";
  peter.note = 2.3;
  peter.bestanden = true;


  // Print information about students
  karl.showStudent();
  peter.showStudent();

  return 0;
}

// Print information about elementary datatypes
void showSizes () {
  cout << "INT:\t"    << sizeof(int)     << "Bytes" << endl;
  cout << "FLOAT:\t"  << sizeof(float)   << "Bytes" << endl;
  cout << "DOUBLE:\t" << sizeof(double)  << "Bytes" << endl;
  cout << "CHAR:\t"   << sizeof(char)    << "Bytes" << endl;
  cout << "BOOL:\t"   << sizeof(bool)    << "Bytes" << endl;
  cout << "Student:"  << sizeof(student) << "Bytes" << endl   << endl;
}
```

Ausgabe A2:
INT: 4Bytes
FLOAT: 4Bytes
DOUBLE: 8Bytes
CHAR: 1Bytes
BOOL: 1Bytes
Student:32Bytes

ID: 1
Name: Friedrich
Studiengang: BWL
Note: 2
Bestanden: 1

ID: 2
Name: Christian
Studiengang: VWL
Note: 3.4
Bestanden: 1

ID: 3
Name: Carina
Studiengang: KoDe
Note: 1
Bestanden: 1

ID: 4
Name: Karl
Studiengang: INI
Note: 4.7
Bestanden: 0

ID: 5
Name: Peter
Studiengang: TMM
Note: 2.3
Bestanden: 1

Aufgabe 3a)

```
#include <iostream>
using namespace std;

int main ( int argc, char* argv[] ) {
  int x = 5;
  int& y = x;

  y = 9;
  int a[100];
  int &b = a[55];

  b = y;
```

```
  int *p = &y;
  int& z = *p;

  z = 33;
  // For validation
  cout << "X: " << x << endl;
  cout << "Y: " << y << endl;
  cout << "Z: " << z << endl;
  cout << "B: " << b << endl;
}
```

Ausgabe A3a:
X: 33
Y: 33
Z: 33
B: 9

Aufgabe 3b)

```
#include <iostream> // In- and Output
#include <string>   // String for our structure
using namespace std;

// Prototype declaration
void showSizes();

// Students struct
struct student {
  int id;
  string name, studiengang;
  float note;
  bool bestanden;

  // Constructor
  student (int i, string na, string sg, float no, bool b) : id(i), name(na), studiengang(sg), note(no),
bestanden(b) {}

  // Print the students information
  void showStudent() {
    cout << "ID:\t\t"       << id          << endl;
    cout << "Name:\t\t"     << name        << endl;
    cout << "Studiengang:\t" << studiengang << endl;
    cout << "Note:\t\t"     << note        << endl;
    cout << "Bestanden:\t"  << bestanden   << endl   << endl;
  }
};

int main () {
  // initialized students
```

```
  struct student studentarr[3] = {
    student(1, "Friedrich", "BWL", 2.0, true),
    student(2, "Christian", "VWL", 3.4, true),
    student(3, "Carina", "KoDe", 1.0, true)
  };

  // References
  float& ref1 = studentarr[0].note;
  struct student& ref2 = studentarr[1];
  bool& ref3 = ref2.bestanden;

  cout << "Ref1: Expected Output [" << studentarr[0].note << "] and is [" << ref1 << "]." << endl;
  cout << "Ref2: Expected Output [" << studentarr[1].bestanden << "] and is [" << ref3 << "]." <<
endl;
}
```

Ausgabe A3b:

```
Ref1: Expected Output [2] and is [2].
Ref2: Expected Output [1] and is [1].
```

Aufgabe 4)

```
#include <iostream>
#include <time.h>
#include <stdlib.h>
using namespace std;

int addiere(int, int *, int&);

int main () {
  // Initialize srand-seed
  srand(time(NULL));

  // Initialize variables with random-integers. Range 1..100
  int var1 = rand() % 100 + 1, var2 = rand() % 100 + 1, var3 = rand() % 100 + 1;
  int& ref = var3;

  // Print the numbers
  cout << "Variablen nach Initialisierung:" << endl;
  cout << "Variable 1:\t" << var1 << endl;
  cout << "Variable 2:\t" << var2 << endl;
  cout << "Variable 3:\t" << var3 << endl << endl;

  // Addiere ( value var1, address var2, reference of var3)
  cout << "Summe von 'Addiere (return)': " << addiere(var1, &var2, ref) << endl << endl;

  // Print information after addiere
  cout << "Variablen nach 'Addiere':" << endl;
  cout << "Variable 1:\t" << var1 << endl;
  cout << "Variable 2:\t" << var2 << endl;
  cout << "Variable 3:\t" << var3 << endl << endl;
```

```
  return 0;
}

int addiere(int par1, int *par2, int& par3) {
  int sum = 0;
  sum = par1 + *par2 + par3;

  // Supersize me!
  par1 *= 2;
  *par2 *= 2;
  par3 *= 2;

  // Print doubled variables
  cout << "Variablen in 'Addiere':" << endl;
  cout << par1 << endl;
  cout << *par2 << endl;
  cout << par3 << endl << endl;

  return sum;
}
```

Ausgabe A4:
Variablen nach Initialisierung:
Variable 1:	53
Variable 2:	71
Variable 3:	16

Variablen in 'Addiere':
106
142
32

Summe von 'Addiere (return)': 140

Variablen nach 'Addiere':
Variable 1:	53
Variable 2:	142
Variable 3:	32

Diskussion A4:
Man erkennt, dass durch das Übergeben der Adresse und Referenz an die Addiere-Funktion die lokalen Variablen aus der Main-Funktion überschrieben werden.

Lediglich die lokale Variable 'var1' wird nicht überschrieben, da keine Referenz, sondern nur der Wert an die Addiere-Funktion übergeben wird.

Wichtig war in der gesamten Aufgabe, die notwendigen Variablen zu dereferenzieren, um nicht nur eine Adresse, sondern auch einen Wert zu bekommen beziehungsweise die richtigen Variablen zu überschreiben/übergeben.