All Times were measured with 24MHz

Aufgabe 3.3

Difference between If- and Switch-Statement:

```
Help for >BasicUart_1 (FreeSoC2)<:
        h,H help
        L: LED on, l: LED off
        I: test empty loop int, i: test empty loop volatile int
        J: test loop global int assignment, j: test loop local int assignment
Delay 0, 1, 2, 5, 10ms: 24, 24103, 72168, 192235, 432301
Time for 'printf' 116275
Time for branch if/else: 26500023
Time for branch switch: 31000024
Time for branch if/else: 26500023
Time for branch switch: 31000024
Time for branch if/else: 26500023
Time for branch switch: 31000022
```

```
if ( i % 2 == 0 || (i == 100 && i < 100)){…}

switch (i % 2){
      case 0: {…} break;
      case 1: {…} break;
```

Result: The If-statement ist faster then the Switch-Statemen. Even if you switch the statement in the if it is faster than the switch. So it seems overall to be faster for few branches.

```
if ( (i == 100 && i < 100) || i % 2 == 0){…}
```

```
Time for 'printf' 116248
Time for branch switch: 31000024
Time for branch if/else: 30000022
Time for branch if/else: 30000022
Time for branch switch: 31000024
Time for branch if/else: 30000022
Time for branch if/else: 30000022
```

Difference between float and double

Code:

```
float aa = 1.31;
float b = 3.31;
float sum = 0;
for ( i = 0; i < 10000; i++ ) {
      sum = aa+b;
}


double a1 = 1.31;
double b2 = 3.31;
```

```
double sum1 = 0;
for ( i = 0; i < 10000; i++ ) {
      sum1 = a1+b2;
}
```

```
Time for floats: 750092
Time for double: 1120158
Time for floats: 750092
Time for double: 1120158
Time for floats: 750090
Time for double: 1120160
```

Result: Calculating with floats ist way faster then double!

Amount of function parameter:

```
Time for function with 1 param: 110062
Time for function with 2 param: 110070
Time for function with 3 param: 110070
Time for function with 4 param: 110075
Time for function with 5 param: 110081
Time for function with 6 param: 110086
Time for function with 7 param: 110090
```

Result: The result of this was quite confusing. In the Lecture was said, that the amount of time is way bigger when using a function with four or more parameter. But our result is approximately linear.

Aufgabe 3.4

```
int getHightestPrim(int range){
    int x, i,n;
    for (x = 2; x <= range; x++)
    {
        int rnd  = rand()/1000000;
        int sum = 0;
        if ( rnd %2 == 0){
            for(int j = 0; j < rnd;j++){
                sum += j;
                sum /= 100;
            }
        }
        for (i = 2; i <= x; i++)
        {
            if (x%i == 0 && x != i)
                break;
            if (i == x){
                n = x;
                sum++;
            }
        }
    }
    return n;
```

```
}
```

We used this function to measure a worst and best execution time. The *rand()* is used to get a variety of cylices. Without it, it take almost the same amount of time. The function *getHighestPrim(int)* was called 1000 times and calculated the highest prim up to 100.

The Result can be seen in the Screenshot below:

```
STARTE !!!!
Run getHightestPrim() for 1000 :
Best Execution Time for Prim: 1110394
Worst Execution Time for Prim: 2365896
```