Lösung der Klausur 1

Datum der Klausur: 11. März 2020, 10:15 bis 11:45

Aufgabe 1 (4 Pkt.)

Gegeben ist das folgende `C` Programm.

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <stdlib.h>
4
5  void MisteryFunction(unsigned int inputNumber){
6      unsigned int divisor=2;
7      if(inputNumber==1)
8          return;
9      while(true){
10         if(inputNumber%divisor==0){
11             printf("%d ", divisor);
12             inputNumber/=divisor;
13             if(inputNumber==1)
14                 break;
15         } else {
16             divisor++;
17         }
18     }
19     printf("\n");
20 }
21
22 int main(int argc, char *argv[]){
23     if(argc!=2)
24         return 0;
25     else
26         MisteryFunction(atoi(argv[1]));
27 }
```

(i) Welches aus der Schulmathematik bekannte Verfahren ist in der Funktion `MysteryFunction` implementiert?

(ii) Welche Bildschirmausgabe erhält man, wenn man das kompilierte Programm mit `./prog` im Terminal startet?

(iii) Welche Bildschirmausgabe erhält man, wenn man das kompilierte Programm mit `./prog 100` im Terminal startet?

Lösung zu Aufgabe 1

(2 pts.) (i) The function `MisteryFunction` decomposes the given `inputNumber` in prime factors and prints them to the screen (Primfaktorzerlegung).

(1 pts.) (ii) If the program is executed without any command line parameter, line 24 in the `main` function is executed and nothing is done.

(1 pts.) (iii) Passing 100 as command line parameter to the programs executes line 26 and the number is given to the `MisteryFunction` which prints `2 2 5 5` to the output.

Aufgabe 2 (5 Pkt.)

Gegeben ist das folgende fehlerhafte C Programm.

```
1 int main(void){
2   int sum;
3   for(u=0; u<50; u++)
4     sum += 2u + 1;
5   printf("The sum of the first %d odd numbers is %d^2 = %d\n", u, u);
6 }
```

Das Programm soll eigentlich Qudratzahlen über die Summe ungerader Zahlen berechnen und die folgende Bildschirmausgabe liefern:

```
The sum of the first 50 odd numbers is 50^2 = 2500
```

Finde und nenne die fünf im Programm befindlichen Fehler. Gib eine entsprechende korrigierte Version des Programms an.

Lösung zu Aufgabe 2

(0.5 pts.) 1. The header `<stdio.h>` is missing.
(0.5 pts.) 2. Add `<stdio.h>` before line 1.
(0.5 pts.) 3. The variable `sum` is not initialized.
(0.5 pts.) 4. Change line 2 to `int sum=0;`.
(0.5 pts.) 5. The variable `u` is not defined.
(0.5 pts.) 6. Add `int u;` before the `for` loop.
(0.5 pts.) 7. The multiplication `*` sign is missing on line 4.
(0.5 pts.) 8. Change line 4 to `sum += 2*u + 1;`.
(0.5 pts.) 9. The last argument of the `printf` statement is missing.
(0.5 pts.) 10. Change line 5 to `printf(..., u, u, sum);`.

Aufgabe 3 (7 Pkt.)

Gegeben ist die folgende C++ Implementation einer Klasse, die komplexe Zahlen beschreibt.

```
1  class C_number
2  {
3    public:
4      C_number(double re_, double im_)
5      {
6          re = re_;
7          im = im_;
8      }
9
10   private:
11     double re;
12     double im;
13 };
```

(i) Ist die folgende Verwendung der Klasse `C_number` in der `main` Funktion korrekt, d.h. lässt sich der Code kompilieren?

```
14   int main()
15   {
16       C_number c1(0.0, 0.0);
17       c1.re = 5.0;
18   }
```

Falls nicht, begründe, warum es zu einem Problem kommt und wie dieses durch Verändern der Klassendefinition behoben werden kann.

(ii) Erweitere die Klassendefinition um eine Member-Function `print`, die die Attribute `re` und `im` am Bildschirm in der Form

```
re = 5.000 , im = 0.000
```

ausgibt, d.h. in Dezimaldarstellung mit drei Nachkommastellen. Gib dabei auch alle `#include` Statements an, die zum fehlerfreien Kompilieren des Programs erforderlich sind.

(iii) Erweitere die Klassendefinition um eine Member-Function `mult`, die die durch die Klasse repräsentierte komplexe Zahl mit einer reellen `double` Zahl (übergeben als Funktionsparameter) multipliziert. Das Ergebnis soll nicht zurückgegeben werden, sondern `re` und `im` soll im Objekt überschrieben werden.

(iv) Schreibe eine `main` Funktion, in der Du die erweitere Klassendefinition verwendest,

- um zunächst eine komplexe Zahl $3 + 4\,i$ zu definieren,
- diese dann mit 2.5 zu multiplizieren und
- das Ergebnis schließlich am Bildschirm auszugeben.

Lösung zu Aufgabe 3

(1 pts.) (i) The code does not compile, because of line 17 where the *private* member `re` is attempted to be accessed outside the class.

(1 pts.) Making the private members public, i.e. removing line 10 would fix the problem. Alternatively a setter method could be added to change `re`. The best solution is to remove line 17 and change line 16 to `C_number c1(5.0,0.0);`.

(1 pts.) (ii) In the interface of the class, something like the following should be added (only `<iostream>` or `<cstdio>` needed[1]).

```
void print(){ printf("re = %.3f , im = %.3f\n", re, im); }
```

Of course, using `std::cout` is also accepted

```
void print(){
    std::cout << std::fixed  << std::setprecision(3)
              << "re = " << re << " , im = " << im << "\n";
} // <iostream> and <iomanip> needed
```

(0.5 pts.) The needed header files have to be included.

(0.5 pts.) This function has to be public.

(1 pts.) (iii) The `mult` function defined inside the class could look like the following.

```
void mult(double factor){
    re *= factor;
    im *= factor;
}
```

(0.5 pts.) This function has to be public.

(1.5 pts.) (iv) The `main` function could look like this.

```
int main(){
    C_number c1(3.0 , 4.0);
    c1.mult(2.5);
    c1.print();
}
```

[1] Including `<stdio.h>` in C++ code is working and it was accepted as corrected, but it should be avoided.

Aufgabe 4 (7 Pkt.)

Schreibe ein `C` Programm, das die Lösung der Gleichung

$$\sin(x) = \frac{x}{2}$$

mit der Eigenschaft $x > 0$ auf mindestens sechs Dezimalstellen genau berechnet und ausgibt. Verwende dazu die in der Vorlesung besprochene Bisektion.

Lösung zu Aufgabe 4

The solution could look like the following.

```
(0.5 pts.)  #include<math.h>
(0.5 pts.)  #include<stdio.h>

  (1 pts.)  double f(double x){  return sin(x)-x/2;  }

            int main(void){
(0.5 pts.)      double eps = 1.e-6;
(0.5 pts.)      double x_min =0.1; double x_max =3.0;
(0.5 pts.)      double f_min = f(x_min), f_max = f(x_max);

(0.5 pts.)      while(x_max-x_min > eps)
                {
(0.5 pts.)          double x_tmp = 0.5*(x_min+x_max);
                    double f_tmp = f(x_tmp);

  (1 pts.)          if(f_min * f_tmp < 0.0){
(0.5 pts.)              x_max = x_tmp;  f_max = f_tmp;
                    } else {
(0.5 pts.)              x_min = x_tmp;  f_min = f_tmp;
                    }
                }

                double x_0 = 0.5*(x_min+x_max);
(0.5 pts.)      printf("Found root: f(%+f) = %+e\n", x_0, f(x_0));
            }
```

Aufgabe 5 (4 Pkt.)

Schreibe eine `C` Funktion, die als Parameter eine positive Integer-Zahl erhält und deren Quersumme berechnet (die Quersumme ist die Summe der Ziffern der Zahl) und diese zurückgibt.

Lösung zu Aufgabe 5

The solution could look like the following.

```
  (1 pts.)  unsigned int SumOfDigits(unsigned int inputNumber){
(0.5 pts.)      unsigned int sum=0;
  (1 pts.)      while(inputNumber!=0){
(0.5 pts.)          sum += inputNumber%10;
(0.5 pts.)          inputNumber /= 10;
                }
(0.5 pts.)      return sum;
            }
```

Aufgabe 6 (6+7=13 Pkt.)

Hinweis: (i) und (ii) können im Wesentlichen unabhängig bearbeiten werden. Denke daran, alle `#include` Statements, die zum fehlerfreien Kompilieren des Codes erforderlich sind, anzugeben.

(i) Implementiere die `C` Funktion

```
void mu_sigma_min_max(int N, double *A,
                      double *p_mu, double *p_sigma,
                      double *p_min, double *p_max)
```

Diese Funktion soll ein Array `A` von `N` `double` Zahlen erhalten und den Mittelwert

$$\mu = \frac{1}{N} \sum_{n=0}^{N-1} A_n$$

und die Standardabweichung

$$\sigma = \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} (A_n - \mu)^2)}$$

berechnen, sowie die kleinste und die größte Zahl in `A` bestimmen. Die Rückgabe dieser vier Ergebnisse soll über die Zeiger `p_mu`, `p_sigma`, `p_min` und `p_max` realisiert werden.

(ii) Gegeben ist eine Datei `measurements.dat` die als ersten Eintrag eine positive Integer-Zahl enthält, die die Anzahl der folgenden Gleitkommazahlen angibt, also z.B. so aussehen könnte:

```
3
7.4
-1.2
88.5
```

Schreibe eine `main` Funktion, in der Du diese Datei öffnest und die darin gespeicherten Zahlen einliest. Die double Zahlen sollen dabei in einem dynamisch angeforderten Array passender Größe gespeichert werden. Rufe dann Deine in (i) implementierte Funktion mit diesem Array auf und gib die vier erhaltenen Ergebnisse (Mittelwert, Standardabweichung, kleinste und größte Zahl) in Exponentialdarstellung mit Vorzeichen auf sechs Stellen genau in eine Datei `analysis.dat` aus. Verwende zum Einlesen und Ausgeben `fopen`, `fclose`, usw. und nicht das Umleiten von stdin und stdout via `<` und `>`. Prüfe beim dynamischen Anfordern vom Speicher und beim Öffnen von Dateien auf mögliche Fehler und gib in solchen Fällen geeignete Fehlermeldungen aus. Achte außerdem darauf, angeforderten Speicher spätestens zum Programmende wieder freizugeben sowie geöffnete Dateien zu schließen.

Lösung zu Aufgabe 6

The solution could look like the following.

(1.5 pts.)

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void mu_sigma_min_max(int N, double* A,
                      double* p_mu, double* p_sigma,
                      double* p_min, double* p_max)
{
    double min=A[0], max=A[0], sum=0.0;

    for(int i=0; i<N; i++){
        sum += A[i];
        if(A[i]>max) max=A[i];
        if(A[i]<min) min=A[i];
    }
    *p_mu  = sum/N;
    *p_min = min;
    *p_max = max;

    sum = 0.0;
    for(int i=0; i<N; i++)
        sum += (A[i] - *p_mu) * (A[i] - *p_mu);
    *p_sigma = sqrt(sum/N);
}

int main(){
    double min, max, mu, sigma;
    FILE *file;   int N;   double* array;

    if((file = fopen("measurements.dat", "r")) == NULL){
        printf("Error opening \"measurements.dat\" file.\n");
        exit(1);
    }

    fscanf(file, "%d", &N);

    if((array = malloc(sizeof(double) * N)) == NULL){
        printf("Error allocating memory.\n"); exit(1);
    }

    for(int i=0; i<N; i++)
        fscanf(file, "%lf", &array[i]);
    fclose(file);
    mu_sigma_min_max(N, array, &mu, &sigma, &min, &max);

    if((file = fopen("analysis.dat", "w")) == NULL){
        printf("Error opening \"analysis.dat\" file.\n"); exit(1);
    }
    fprintf(file, "%+.6e %+.6e %+.6e %+.6e\n", mu,sigma, min,max);

    fclose(file);  free(array);
}
```

(1 pt.) (0.5 pts.) (0.5 pts.) (0.5 pts.) (0.5 pts.) (0.5 pts.) (0.5 pts.) (0.5 pts.) (1 pt.) (0.5 pts.) (1 pt.) (0.5 pts.) (1 pt.) (1 pt.)