

Abschlußtest Programmieren

29. Juni 2007

Name:

Punkte $\left(\begin{array}{l} 28 \text{ Pkt.} \cong 100\% \\ 32 \text{ Pkt.} \text{ max.} \end{array} \right)$:

1. (6 P) Finden Sie die in nachfolgendem Programm enthaltenen 6 Fehler und korrigieren Sie diese!

Der GNU-Compiler (`g++ -W -Wall k1.cpp`) findet zwei Fehler in den Zeilen 21 und 35. Alle weiteren Fehler sind programmlogischer Natur.

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  void swap(double &a, double &b)
6  {                                // Vertauschen zweier Zahlen
7      double tmp;
8      a = b;
9      tmp = a;
10     b = tmp;
11     return;
12 }
13
14 int main()
15 {
16     int n=1;
17     while ( n<1 && 20<n )          // Lese n vom Interval [1..20] ein
18     {
19         cout << "n = "; cin >> n;
20     }
21     double d = new double [n];
22
23     for (int i = 0; i<=n; ++i)     // Initialisierung der Daten
24     {
25         d[i] = exp(1/(i-4.4)/(i-4.4));
26     }
27
28     int    imax, imin;
29     for (int i = 1; i<n; ++i)     //  suche max/min
30     {
31         if (d[imax] < d[i]) imax = i;
32         if (d[imin] > d[i]) imin = i;
33     }
34
35     swap(d[imax],d+imin);         // Vertausche min/max
36
37     delete [] d;
38     return 0;
39 }
```

Name:

2. (4 P) Welche Ausgaben liefert das folgende Programm in den drei cout Anweisungen? Begründen Sie Ihre Aussage!

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  void Read(double &a, int i)
6  {
7      a = 1e-1;
8      i = 5;
9      return;
10 }
11
12 double Calc(const double p, const int m)
13 {
14     double s = 3.0;
15
16     for (int i=1; i<=m+1; ++i)
17     {
18         s += i*pow(p,i-1);
19     }
20     return s;
21 }
22
23 int main()
24 {
25     double eps = 1.0e+2;
26     int    n;
27
28     n = 11/4;
29     cout << eps << " " << n << endl;
30
31     Read(eps,n);
32     cout << eps << " " << n << endl;
33
34     cout << Calc(eps, n) << endl;
35
36     return 0;
37 }
```

Name:

3. (6 P) Die harmonische Reihe $\sum_{k=1}^{\infty} \frac{1}{k}$ ist divergent. Schreiben Sie eine Funktion mit dem Übergabeparameter C welche diejenige kleinste ganze Zahl M bestimmt, für die

$$\sum_{k=1}^M \frac{1}{k} > C$$

gilt. Diese Zahl M soll von der Funktion wieder an das aufrufende Programm zurückgegeben werden.

Wir nehmen an, daß der Parameter C aus dem Intervall $[1, 20]$ gewählt wird, sodaß keine Tests bzgl. C nötig sind und die Zahldarstellung im Computer noch genau genug ist (also alles möglichst einfach ist).

Hierbei sind keine `break`, `continue`, `goto` erlaubt.

4. (6 P) Programmieren Sie die Funktion

```
bool is_palindrom(const string &s);
```

welche an das aufrufende Hauptprogramm zurückgibt, ob der String `s` ein Palindrom ist, d.h., vorwärts und rückwärts gelesen das gleiche ergibt.

Wir nehmen an, daß nur Großbuchstaben verwendet werden (nicht darauf testen!). Ein Aufruf `is_palindrom("RADAR")` soll `true` zurückgeben, während der Aufruf `is_palindrom("ABCD")` in einem `false` resultiert.

Die Länge des Strings ist durch `s.size()` abfragbar.

Hierbei sind keine `break`, `continue`, `goto` erlaubt.

Name:

Name:

Mathematiker

5. (12 P) Die allg. Gleichung einer Ebene im \mathbb{R}^3 ist

$$ax + by + cz + d = 0 ,$$

sodaß jede Ebene durch diese vier Parameter charakterisiert ist. Der Abstandes eines Punktes $P_0 = (x_0, y_0, z_0)$ zu dieser Ebene berechnet sich über die Hessesche Normalform

$$\frac{ax_0 + by_0 + cz_0 + d}{\pm\sqrt{a^2 + b^2 + c^2}} .$$

Entwerfen Sie eine Klasse für die Ebene, eine Klasse für einen Polygonzug und geben Sie die Definitionen (also Implementierungen) aller im nachfolgenden Programmfragment notwendigen Methoden der Klassen an.

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  // gegebene Klasse Point, keine Methoden notwendig
6  class Point
7  {
8      public:
9          double x,y,z;
10 };
11
12 #include "ihre_deklarationen.hpp"
13
14 //----- Hauptprogramm -----
15 int main()
16 {
17     Ebene plain(3.2, 4.1, -0.2, 1.9);
18     Polygon pp("datafile.txt"); // dieser Konstruktor wird als
19                                 // gegeben angenommen!!
20
21     double d0 = plain.dist(pp[0]); // Abstand zum Anfang des Polynoms
22
23     int k_min; // Index des Polynompunktes mit
24               // minimalem Abstand
25     double d_min; // min.Abstand Polynom--Ebene
26
27               // Berechne diesen min. Abstand und
28     d_min = plain.minDist(pp,k_min); // den Index des Punktes
29
30     return 0;
31 } // !!
```

Name:

Lehramt

5. (12 P) Das folgende Programmfragment definiert die zwei 3D-Vektoren \underline{a} und \underline{b} , berechnet deren Skalarprodukt $\underline{a} \cdot \underline{b}$ und deren Kreuzproduktvektor $\underline{c} := \underline{a} \times \underline{b}$. Anschließend wird das Spatprodukt $(\underline{a} \times \underline{c}) \cdot \underline{b}$ berechnet.

```
1  #include <iostream>
2  using namespace std;
3  //-----  Datenstrukturen  -----
4  struct Vector3D
5  {
6      double x,y,z;          // Die drei Vektorkomponenten
7  };
8  //-----  Funktionen  -----
9
10 //-----  Hauptprogramm  -----
11 int main()
12 {
13     Vector3D a = {0,1,2}, b = {3,2,0}, c;
14     double  scal, spat;
15
16     scal = ScalarProduct(a,b);    // a . b
17     CrossProduct(a,b,c);         // a x b
18     spat = SpatProduct(a,c,b);   // (a x c) . b
19
20     cout << "Scalar : " << scal << endl;
21     cout << "Spat   : " << spat << endl;
22     return 0;
23 }
```

Schreiben Sie die drei Funktionsdefinitionen für `ScalarProduct`, `CrossProduct` und `SpatProduct` so auf, daß die Rufzeilen im Hauptprogramm stimmen und daß die Funktion `SpatProduct` **die beiden anderen Funktionen** zur Berechnung **benutzt**.

$$\text{Kreuzprodukt: } \underline{s} \times \underline{t} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} \times \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} := \begin{pmatrix} s_2 t_3 - s_3 t_2 \\ s_3 t_1 - s_1 t_3 \\ s_1 t_2 - s_2 t_1 \end{pmatrix}$$

$$\text{Inneres Produkt: } \underline{s} \cdot \underline{t} := s_1 t_1 + s_2 t_2 + s_3 t_3$$

Spatprodukt $(\underline{s} \times \underline{t}) \cdot \underline{r}$: Hintereinanderausführung von Spat- und Skalarprodukt.

Name:

Name: