Ingenieurinformatik – Diplom-FA
(C-Programmierung)

Name	Vorname	Matrikelnummer	Sem.-Gr.:	Hörsaal	Platz

Zulassung geprüft:

_____

(Grundlagenteil)	Aufgabe 1	Aufgabe 2	Aufgabe 3	Summe

Die Prüfung ist nur dann gültig, wenn Sie die erforderliche Zulassungsvoraussetzung erworben haben (erfolgreiche Teilnahme am Praktikum). Dies wird vom Aufgabensteller überprüft.

FA-Diplom

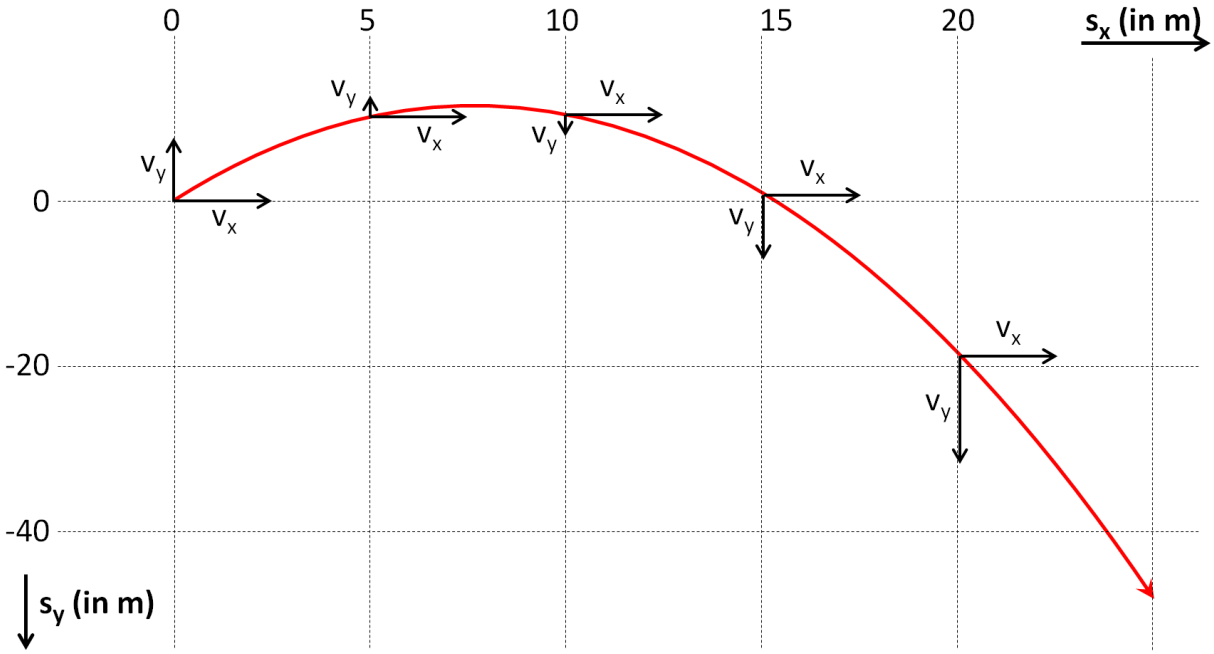
Aufgabensteller: Dr. Reichl, Dr. Küpper und Kollegen

Bearbeitungszeit: 60 Minuten

Hilfsmittel:
- Taschenrechner nicht zugelassen
- PC/Notebook nicht zugelassen
- Sonstige eigene Hilfsmittel sind erlaubt
- Bearbeitung mit Bleistift ist erlaubt

Aufgabe 1: (ca. 21 Punkte)

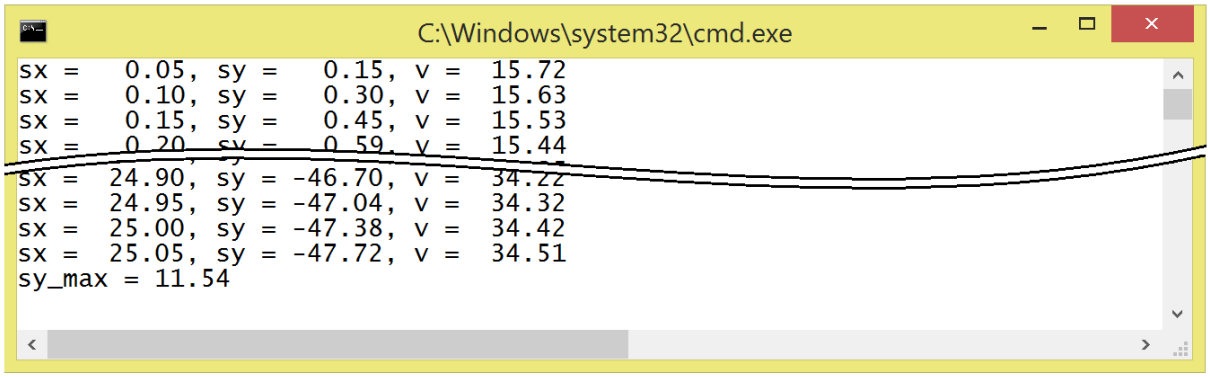
Eine Kugel wird zum Zeitpunkt t = 0s schräg nach oben geworfen. Erstellen Sie ein C-Programm, welches die x- und y-Koordinaten der Flugbahn (s_x und s_y) in Zeitschritten von Δt = 0,01s berechnet und tabellarisch auf dem Bildschirm ausgibt (siehe Bildschirmfoto unten auf dieser Seite).



1. Zu Beginn (t = 0) beträgt die x-Komponente der Geschwindigkeit v_x = 5m/s und die y-Komponente der Geschwindigkeit v_y = 15m/s. Die Kugel befindet sich zu Beginn an der Position $s_x = s_y = 0$.
2. Wenn die Koordinaten $s_x(t)$, $s_y(t)$ und die Geschwindigkeiten $v_x(t)$, $v_y(t)$ zu einem beliebigen Zeitpunkt t bekannt sind, können $s_x(t + \Delta t)$, $s_y(t + \Delta t)$ und $v_y(t + \Delta t)$ wie folgt berechnet werden:

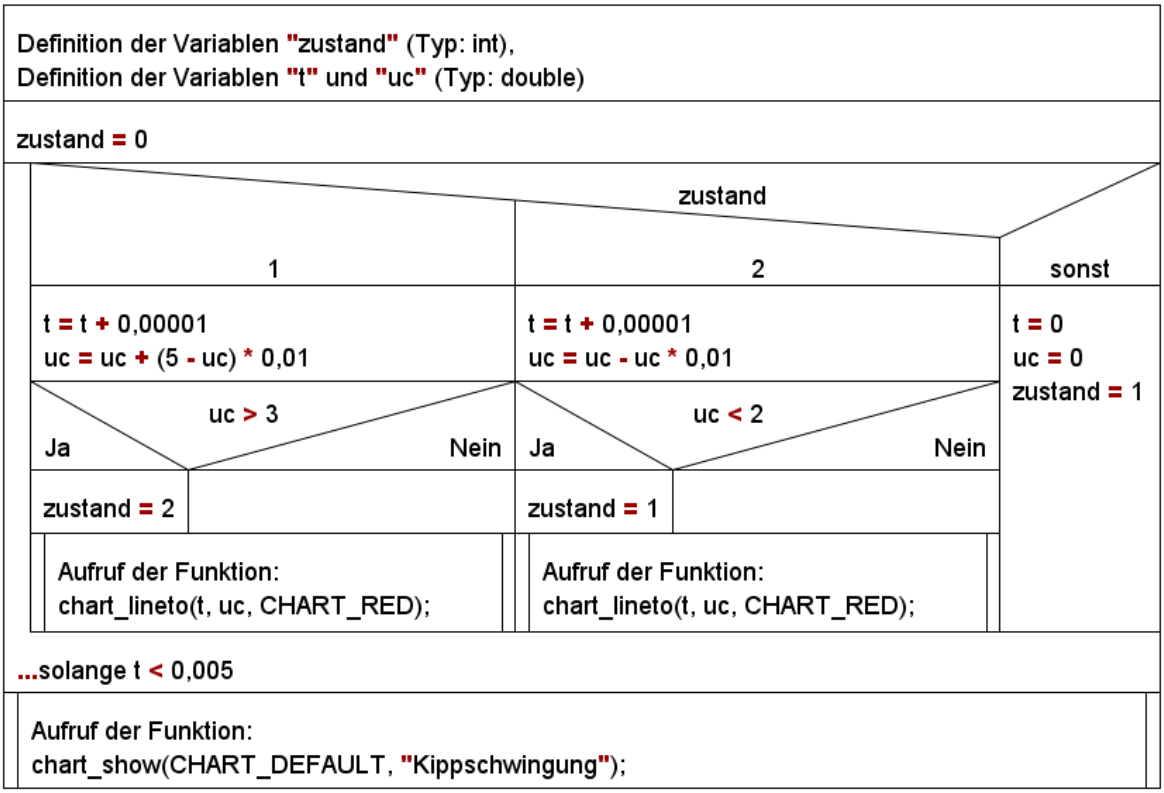
   $s_x(t + \Delta t) = s_x + \Delta t \cdot v_x$
   $s_y(t + \Delta t) = s_y + \Delta t \cdot v_y$
   $v_y(t + \Delta t) = v_y - \Delta t \cdot g$ (mit g = 9,81 m/s²)

   Die Geschwindigkeitskomponente v_x ändert sich nicht.
3. Nach jedem Zeitschritt Δt = 0,01s werden die aktuellen Werte von s_x, s_y und der Betrag der Kugelgeschwindigkeit v mit zwei Nachkommastellen ausgegeben.
4. Die Berechnungs-Schleife wird solange ausgeführt, bis der Zeitpunkt t = 5s erreicht ist.
5. Vor dem Beenden des Programms wird der maximale Wert von s_y ausgegeben, der während des Programmablaufs aufgetreten ist.



Aufgabe 2: (ca. 22 Punkte)

Das folgende Struktogramm zeigt den Ablauf eines Programms zur Simulation von elektronischen Kippschwingungen. (Hinweis: In der Datei **chart.h** befinden sich alle für den Aufruf der Funktionen **chart_lineto** und **chart_show** notwendigen Prototypen und symbolischen Konstanten.)



2.1. Vervollständigen Sie den C-Quelltext auf der folgenden Seite so, dass er genau dem abgebildeten Struktogramm entspricht.

2.2. Handelt es sich bei der Schleife im Struktogramm um eine abweisende oder um eine nicht-abweisende Schleife?

2.3. Wie groß (in Bytes!) ist die Variable **zustand**, wie groß ist die Variable **t**, wenn das C-Programm auf einem Windows-Rechner im Praktikumsraum ausgeführt wird?

2.4. Wie groß ist die größte Dezimalzahl, die in einer Variablen des Typs **unsigned short** abgespeichert werden kann? Die Angabe eines Potenz-Ausdrucks genügt. (Hinweis: Gehen Sie davon aus, dass Variablen des Typs **short** eine Größe von 2 Bytes haben.)

Unterpunkt 2.1., C-Quelltext:

```
/* Simulation von Kippschwingungen */
#include "chart.h"
int main(void)
{
        int zustand;
        double t, uc;









        return 0;
}
```

Aufgabe 3: (ca. 24 Punkte)

3.1. Korrigieren Sie die fünf Fehler, die sich in das folgende C-Programm zur Berechnung aller Primzahlen bis 100.000 eingeschlichen haben.

```
#include <stdio.h>
#define ENDE 100000;

int main(void);
{
    int n, t, prim, count = 1;
    printf("       2");
    for(n = 3; n <= ENDE; n += 2)
    {
        prim == 1;
        for(t = 2; t * t <= n && prim == 1; t++)
            if(n % t = 0) prim = 0;

        if(prim == 1)
        {
            printf("%8d", n);
            ++count;
        }
    }
    printf("\n%d Primzahlen gefunden.\n");
    return 0;
}
```

3.2. Wie sieht die Bildschirmausgabe des folgenden C-Programms aus?

```
#include <stdio.h>

void berechne(int wert, int anz, int *s, int *p);

int main(void)
{
    int x, result1, result2;
    for(x = 1; x < 4; x++)
    {
        berechne(x, x, &result1, &result2);
        printf("%d, %d\n", result1, result2);
    }
    return 0;
}

void berechne(int wert, int anz, int *s, int *p)
{
    int i;
    *s = 0;
    *p = 1;
    for(i = 0; i < anz; i++)
    {
        *s += wert;
        *p *= wert;
    }
}
```

Ausgabe:

3.3. Das folgende Programm soll das Kreuzprodukt (auch "Vektorprodukt" genannt) der Vektoren **v1** und **v2** berechnen und im Vektor **erg** speichern. Ergänzen Sie den abgebildeten C-Quelltext!

```
#include <stdio.h>
int main(void)
{
    double  v1[3], v2[3], erg[3];
    printf("Vektor 1:\n");
    scanf("%lf", &v1[0]); scanf("%lf", &v1[1]); scanf("%lf", &v1[2]);
    printf("Vektor 2:\n");
    scanf("%lf", &v2[0]); scanf("%lf", &v2[1]); scanf("%lf", &v2[2]);

    /* Kreuzprodukt berechnen und in Vektor "erg" speichern */









    printf("Ergebnis: (%f, %f, %f)\n", erg[0], erg[1], erg[2]);
    return 0;
}
```

3.4. Das folgende Programm soll alle 100 Elemente der Matrix **m** mit Zufallszahlen belegen. Die Zufallszahlen sollen im Intervall von 0,25 bis 0,75 liegen. Ergänzen Sie den C-Quelltext!

```
#include <stdlib.h>
#include <stdio.h>
int main(void)
{
    double m[10][10];


    /* Matrix "m" mit Zufallszahlen von 0.25 bis 0.75 belegen */









    printf("Zufallszahl in Element m[3][3] = %f\n", m[3][3]);
    printf("Zufallszahl in Element m[5][5] = %f\n", m[5][5]);
    printf("Zufallszahl in Element m[7][7] = %f\n", m[7][7]);
    return 0;
}
```

(Platz für Notizen und Nebenrechnungen)