Seminar 2

1. Dateien

Für die Dateioperationen werden Objekte der Klasse `fstream` verwendet. Wird in die Datei nur geschrieben, kann stattdessen die Klasse `ofstream` verwendet werden. Für reine Eingabedateien bietet sich die Klasse `ifstream` an. Auf die Objekte dieser Klassen können Ein- und Ausgabeoperatoren `(>>` und `<<)` angewandt werden. Vor einem Zugriff muss die Datei mit der Elementfunktion open() geöffnet werden.

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

int main()
{
    fstream f;
    f.open("test.dat", ios::out);
    f << "Dieser Text geht in die Datei"
<< endl;
    f.close();
}
```

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

int main(int argc, char *argv[])
{
    fstream f;
    char cstring[256];
    f.open(argv[1], ios::in);
    while (!f.eof())
    {
        f.getline(cstring, sizeof(cstring));
        cout << cstring << endl;
    }
    f.close();
}
```

2. Random

`int rand (void)` = gibt eine zufällige Ganzzahl zwischen 0 und RAND_MAX zurück
`RAND_MAX` = eine Konstante definiert in `<cstdlib>`

```
int v1 = rand() % 100;          // v1 in the range 0 to 99
int v3 = rand() % 30 + 1985;    // v3 in the range 1985-2014
double v4 = static_cast<double>(rand());
```

`void srand (unsigned int seed)` = stellt den Zustand (seed) des Pseudozufallszahlengenerators

3. Casting

In C++ kann eine explizite Typwandlung mit einem Typwandlungs-Operator geschrieben werden. Ein Umwandlungs-Ausdruck beginnt mit dem Schlüsselwort `static_cast`, dem der Namen eines Datentyps in spitzen Klammern und dann in runden Klammern der Ausdruck folgt, dessen Wert in den angegebenen Datentyp zu wandeln ist. So ist der Ausdruck `static_cast<double>(1)` ein Ausdruck vom Datentyp `double`, dessen Wert sich aus der Wandlung des Wertes des Numerales 1 (vom Datentyp `int`) in den Datentyp `double` ergibt, das sollte also der Wert 1.0 sein.

```
#include <iostream>
int main() {
```

```
  std::cout << static_cast<int>(1.5) << std::endl;
  std::cout << static_cast<int>('A');
}
```

4. Klassen

Objektorientierte Programmierung erfordert eine neue Denkweise:
1. Welche Objekte/Klassen sind schon im Problemraum vorhanden?
2. Welche Dienste muss jede Klasse erbringen?

Vergleichen Sie dies mit der prozeduralen Programmierung: anstatt das Problem in Daten + Algorithmen zu zerlegen, zerlegen wir es zuerst in Objekte (Daten + Verhalten).

Objekte liefern Abstraktionen. Ein Objekt kann ohne Wissen über die konkrete Implementierung verwendet werden. Dadurch kann Software aus Komponenten erstellt werden - genau wie andere Formen von Engineering-Systemen.

```
class Date {
public: // access functions
  int get_day();
  int get_month();
  int get_year();

// function to set the date
  void set_date(int d, int m, int y);

private:
  int day;
  int month;
  int year;
};
```

```
int Date::get_day(){ return day; }
int Date::get_month(){ return month; }
void Date::set_date(int d, int m, int y){
 // check month is 1..12
 if(m < 1 || m > 12) Raise_Error();

 // check day is 1..31
 if(d < 1 || d > 31) Raise_Error();

 // if April, May, September, November
 if(m==4 || m==6 || m==9 || m==11)
   if(d > 30)
     Raise_Error();

 // if February use leap year rules
 if(m==2){
   if((y%4==0)^(y%100==0)^(y%400==0)){
     if(d > 29) Raise_Error();
   }
   else if(d > 28) Raise_Error();
 }

 // if we got here d,m,y are OK
 day = d; month = m; year = y;
}
```

5. Übungen

1.A) Schreiben Sie eine Funktion mit folgender Deklaration von

```
char* find (const char* s, const char* f);
```

Die Funktion soll einen Zeiger auf das erste Auftreten von `s` in `f` (und sonst `NULL`) zurückgeben.

1.B). Implementieren und testen Sie die folgenden Funktionen mithilfe von Pointers:

`strcpy()`, die eine C-Style Zeichenkette in eine andere einfügt.

Überlegen Sie zuerst wie die Argumenttypen und der Rückgabetyp aussehen muss.

2. Implementieren Sie eine einfache Account-Klasse. Die Klasse soll Methoden für folgende bereitstellen: Ein- und Auszahlungen vorzunehmen, infos über den Saldo - eine private `double` Variable - zurückzugeben.

```
void Konto::Bezahlen(const double & Betrag); // Aus dem Konto nehmen
void Konto::Einzahlung(const double & Betrag);
double Konto::Kontostand(void); // Den Restbetrag zurückgeben
```

Stellen Sie sicher, dass der Konstruktor den Saldo mit Null initialisiert.

3. Erweitern Sie die Account-Klasse um eine Methode, die den Typ und Betrag jeder Transaktion in eine Datei speichert. Format: `IN/OUT Betrag`

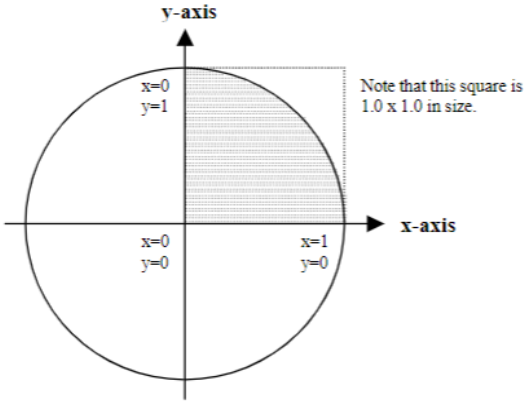
Beispiel:

```
IN 10 110
OUT 5 105
IN 1 106
```

4. Erweitern Sie die Account-Klasse um eine Methode, die den Typ und Betrag jeder Transaktion aus einer Datei einliest. Format: `IN/OUT Betrag`

Zusatz

5. Man kann die Kreiszahl Pi näherungsweise bestimmen:
(a) Stellen Sie sich einen Kreis vor, dessen Fläche ist: Fläche = PI * R^2
(b) Stellen Sie sich nun vor, Sie teilen den Kreis genau in 4 Quadranten
(c) Die Fläche eines Quadranten ist daher: Fläche = 0,25 * PI * R^2
(d) Setzen wir nun den Radius auf R = 1;
(e) Die Gleichung lautet daher: Fläche = 0,25 * PI
(f) Wir können also einfach sagen: PI = Fläche * 4



Die Frage ist nun, wie man die grau schattierte Fläche berechnet.

Idee:
Man stelle sich einen Kreis mit Mittelpunkt (0,0) und Radius 1 vor. Es werden zufällig Punkte erzeugt, bei denen sowohl x als auch y im Intervall [0,1[liegen. Dann wird die Entfernung dieser Punkte zum Ursprung ermittelt. Ist diese Entfernung kleiner als 1, so liegt der Punkt innerhalb des Kreises. Setzt man bei einer ausreichenden Zahl von Zufallspunkten die Zahl der Treffer in das richtigen Verhältnis zur Gesamtzahl der Punkte, so erhält man einen Näherungswert für Pi (Pi = 4 * AnzahlTreffer / AnzahlGesamt).

Schreiben Sie eine Funktion, das auf oben beschriebene Weise Pi berechnet.