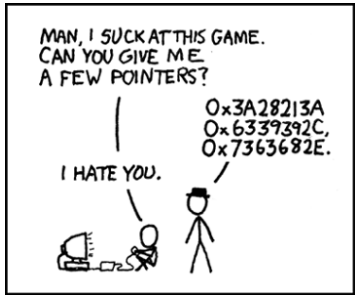
Einführung in die Programmiertechnik II

Übung 2



Johannes Wolf

Universität Potsdam – Digital Engineering Fakultät – Hasso-Plattner-Institut Sommersemester 2017 09.05.2017

Übung 2 | **Value, Reference, Pointer**

Funktionsdefinition

```
1  void f(int a)
2  {
3      a = 1;
4      std::cout << "a = " << a << "\n";
5  }
6
7  void g(int& a) // = int &a
8  {
9      a = 2;
10     std::cout << "a = " << a << "\n";
11 }
12
13 void h(int* a)
14 {
15     // a = 3; // error
16     *a = 3;
17     std::cout << "a = " << a << "\n";
18     std::cout << "*a= " << *a << "\n";
19 }
```

Main-Methode

```
1  int main(int argc, char * argv[])
2  {
3      int i = 0;
4      std::cout << "i = " << i << "\n";
5      std::cout << "\n";
6
7      f(i);
8      std::cout << "i = " << i << "\n";
9      std::cout << "\n";
10
11     g(i);
12     std::cout << "i = " << i << "\n";
13     std::cout << "\n";
14
15     int* p = &i;
16     h(p);
17     std::cout << "i = " << i << "\n";
18 }
```

Übung 2 | **Value, Reference, Pointer**

Funktionsdefinition

```
1  void f(int a)
2  {
3      a = 1;
4      std::cout << "a = " << a << "\n";
5  }
6
7  void g(int& a) // = int &a
8  {
9      a = 2;
10     std::cout << "a = " << a << "\n";
11 }
12
13 void h(int* a)
14 {
15     // a = 3; // error
16     *a = 3;
17     std::cout << "a = " << a << "\n";
18     std::cout << "*a= " << *a << "\n";
19 }
```

Main-Methode

```
1  int main(int argc, char * argv[])
2  {
3      int i = 0;
4      std::cout << "i = " << i << "\n";
5      std::cout << "\n";
6
7      f(i);
8      std::cout << "i = " << i << "\n";
9      std::cout << "\n";
10
11     g(i);
12     std::cout << "i = " << i << "\n";
13     std::cout << "\n";
14
15     int* p = &i;
16     h(p);
17     std::cout << "i = " << i << "\n";
18 }
```

Programm-Ausgabe

```
1  i = 0
2
3  a = 1
4  i = 0
5
6  a = 2
7  i = 2
8
9  a = 0000004DE91CFD34
10 *a= 3
11 i = 3
```

Übung 2 | **Zufall**

`https://en.wikipedia.org/wiki/Randomness`

- **Randomness** is the lack of pattern or predictability in events.

`https://de.wikipedia.org/wiki/Zufall`

- Von **Zufall** spricht man dann, wenn für ein einzelnes Ereignis oder das Zusammentreffen mehrerer Ereignisse keine kausale Erklärung gegeben werden kann. Als kausale Erklärungen für Ereignisse kommen in erster Linie allgemeine Gesetzmäßigkeiten oder Absichten handelnder Personen in Frage. Die Erklärung für Zufall ist also gerade der Verzicht auf eine (kausale) Erklärung.

`https://de.wikipedia.org/wiki/Zufallszahl`

- Als **Zufallszahl** wird das Ergebnis von speziellen Zufallsexperimenten bezeichnet.
- Zur Erzeugung von Zufallszahlen gibt es verschiedene Verfahren. Diese werden als **Zufallszahlengeneratoren** bezeichnet. Ein entscheidendes Kriterium für Zufallszahlen ist, ob das Ergebnis der Generierung als unabhängig von früheren Ergebnissen angesehen werden kann oder nicht.
- **Echte Zufallszahlen** werden mithilfe physikalischer Phänomene erzeugt: Münzwurf, Würfel, Roulette, Rauschen elektronischer Bauelemente, radioaktive Zerfallsprozesse oder quantenphysikalische Effekte. Diese Verfahren nennen sich physikalische Zufallszahlengeneratoren, sind jedoch zeitlich oder technisch recht aufwendig.
- In der realen Anwendung genügt häufig eine Folge von **Pseudozufallszahlen**, das sind scheinbar zufällige Zahlen, die nach einem festen, reproduzierbaren Verfahren erzeugt werden. Sie sind also nicht zufällig, da sie sich vorhersagen lassen, haben aber ähnliche statistische Eigenschaften (gleichmäßige Häufigkeitsverteilung, geringe Korrelation) wie echte Zufallszahlenfolgen. Solche Verfahren nennt man Pseudozufallszahlengeneratoren.

`https://de.wikipedia.org/wiki/Pseudozufall`

- Als **Pseudozufall** wird bezeichnet, was zufällig erscheint, in Wirklichkeit jedoch berechenbar ist. In diesem Sinn generieren Pseudozufallszahlengeneratoren, wie auch kryptographisch sichere Zufallszahlengeneratoren pseudozufällige Zahlen.
- In der Berechenbarkeitstheorie wird alles das als pseudozufällig bezeichnet, was aus der Perspektive des Betrachters *nicht von wirklicher Zufälligkeit unterschieden* werden kann.
- Beispiel Münzwurf: Befindet sich die Münze bereits in der Luft, ist es theoretisch möglich, anhand ihrer Rotation, Geschwindigkeit usw. das Ergebnis vorherzusagen. Jemandem, dem entsprechende Messgeräte (und Rechenkapazität) nicht zur Verfügung stehen, erscheint der Wurf aber immer noch zufällig; der Wurf mit der Münze in der Luft ist für ihn pseudozufällig.
- Generell definiert man in der Berechenbarkeitstheorie als pseudozufällig, was *durch effiziente Algorithmen nicht vorhergesagt werden kann*.
- Pseudozufälligkeit ist aber immer noch berechenbar (man kann sie effizient erzeugen), nur nicht vorhersagbar.

`https://de.wikipedia.org/wiki/Pseudozufall` (Forts.)

- Die wichtigste Anwendung von Pseudozufallszahlen sind die so genannten Zufallsgeneratoren, die in praktisch allen Programmiersprachen verfügbar sind.
- Es handelt sich dabei meist lediglich um **Pseudozufallszahlengeneratoren** (engl. PRNG, pseudo random number generator)
- Sie erzeugen eine Zahlenfolge, die zwar zufällig aussieht, es aber nicht ist, da sie durch einen *deterministischen Algorithmus* berechnet wird. Bei jedem Start der Zufallszahlenberechnung mit gleichem Startwert, der so genannten Saat (engl. seed), wird die gleiche pseudozufällige Zahlenfolge erzeugt.
- Sie verletzen damit bestimmte Eigenschaften echter Zufallszahlen, sind jedoch von Computern wesentlich einfacher herzustellen. Oft werden periodische Zahlenfolgen erzeugt, die gleichen Zahlen wiederholen sich also nach einer bestimmten Länge. Der Vorteil von PRNGs ist die hohe Geschwindigkeit. Durch geschickte Wahl der Parameter kann man die Periode genügend groß machen.

- The `numeric_limits` class template provides a standardized way to query various properties of arithmetic types.
- `http://en.cppreference.com/w/cpp/types/numeric_limits`

```cpp
int i = std::numeric_limits<int>::max(); // C: INT_MAX

unsigned long long l = -1ull;
```

Übung 2 | **Auswertung Übung 1**

1. Dreieckszahlen
   - overflow
   - Zahlenanalyse/Stringaufbau
2. Fibonacci
   - rekursiv vs. iterativ
   - Datentypen in C++
3. Münzrückgabe
   - ungültige Eingaben abfangen
   - Dateiausgabe

1. Dreieckszahlen
   - overflow
   - Zahlenanalyse/Stringaufbau
2. Fibonacci
   - rekursiv vs. iterativ
   - Datentypen in C++
3. Münzrückgabe
   - ungültige Eingaben abfangen
   - Dateiausgabe

Variablenname in einer Übungsabgabe:
`whythehelldidyouhavetoaddthisincrediblyannoyingfilestufftothisexercise`

1. Dreieckszahlen
   - overflow
   - Zahlenanalyse/Stringaufbau
2. Fibonacci
   - rekursiv vs. iterativ
   - Datentypen in C++
3. Münzrückgabe
   - ungültige Eingaben abfangen
   - Dateiausgabe

Variablenname in einer Übungsabgabe:

~~whythehelldidyouhavetoaddthisincrediblyannoyingfilestufftothisexercise~~

```
whyTheHellDidYouHaveToAddThisIncrediblyAnnoyingFileStuffToThisExercise
```