Zufallszahlen mit dem C-Generator

Ohne gute Kenntnisse in objektorientierter Programmierung sind die C++-Generatoren für Zufallszahlen kaum zu verstehen. Deshalb verwenden wir den Generator der Vorgänger-Sprache C. Er funktioniert auch in C++.

Beispiel für einen Zufallszahlen-Generator

Der folgende Algorithmus ist ein Beispiel für einen sehr einfachen Generator:

1. Starte mit einer beliebigen positiven Zahl.
2. Ziehe von der Zahl die dritte Wurzel.
3. Ermittle eine Pseudo-Zufallszahl, indem du die ersten drei Nachkommastellen des Ergebnisses als ganze Zahl betrachtest.
4. Wenn eine weitere Zahl benötigt wird, nimm die letzte Zufallszahl als neuen Startwert und springe zu 2.

Mit dem Startwert 13 würde dieser Generator die folgenden Zahlen berechnen:

$$\sqrt[3]{13} = 2{,}351\ldots \Rightarrow \mathbf{351}$$
$$\sqrt[3]{351} = 7{,}054\ldots \Rightarrow \mathbf{54}$$
$$\sqrt[3]{54} = 3{,}779\ldots \Rightarrow \mathbf{779}$$
$$\ldots$$

In Wirklichkeit werden kompliziertere Formeln verwendet. Es existieren auch Generatoren, die keine Pseudo-Zufallszahlen, sondern wirklich zufällige Werte erzeugen. Dafür kann man zum Beispiel das Rauschen eines elektrischen Bauteils digitalisieren.

Initialisieren des Zufallszahlen-Generators

Der Generator wird mit der Systemzeit initialisiert:

```
srand( (unsigned) time(nullptr));
```

Dieser Funktionsaufruf setzt die Systemzeit in Sekunden als Startwert. Man schreibt ihn nur einmal an den Programmanfang. Damit das funktioniert, müssen Sie die Headerdatei `<ctime>` einbinden.

Erzeugen von Zufallszahlen

Der Prototyp der Generator-Funktion sieht so aus: `int rand();`

Sie liefert Zahlen aus dem Intervall [0; RAND_MAX]. Bei RAND_MAX handelt es sich um eine Konstante, deren Wert in den meisten Umgebungen 32767 ist.

Beispiel für das Erzeugen einer Zahl:

```
int i;
i = rand();
```

Verändern des Zahlenbereichs

Einfach ist die Verwendung einer Obergrenze mit Hilfe des Modulo-Operators (Rest einer Ganzzahldivision):

```
i = rand() % 100; // liefert ganze Zahlen aus [0; 99]
```

Überlegen Sie, wie man andere Intervalle erhält!

a) [1; 100]

```
i = rand() % 100 +1
```

b) [0; 100]

```
i = rand() % 101
```

c) [50; 100]

```
i = rand() % 51 + 50
```

d) [-99; 99]

```
i = rand() % 199 - 99
```

e) {0; 0,5; 1; 1,5; …; 9,5; 10}

```
i = rand() % 21 (*0,5) / 2.0
```

Wenn man den Zahlenbereich mit Hilfe des Modulo-Operators einschränkt, kommen kleine Zahlen häufiger vor als große. Man sagt, die Zufallszahlen sind nicht gleichverteilt. Zum Ausprobieren von Programmen ist die Modulo-Methode trotzdem gut genug. Für professionelle Tests und für ernsthafte Anwendungen müssten Sie aber andere Formeln oder die neuen C++-Generatoren verwenden.