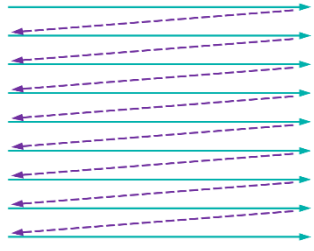
Praktikumstermin Nr. 05, INF: Säulendiagramm, Sudoku einlesen

(Pflicht-) Aufgabe INF-05.01: Säulendiagramm (Schleifen, `if-else`, Arrays)

Schreiben Sie ein C++ Programm, welches fünf ganzzahlige Zahlwerte zwischen `1` und `9` (beides inklusive) in ein Array einliest und die eingegebenen Werte dann als „stehendes Säulendiagramm aus Pluszeichen und Sternen" ausgibt.
„Leere Positionen" oberhalb der Säulen sollen durch Punkte repräsentiert werden.

Die Eingabewerte sollen geprüft werden, bei falscher Eingabe soll der Benutzer erneut zur Eingabe aufgefordert werden (siehe Testlauf). Sie können davon ausgehen, dass der Benutzer nur Zahlen eingibt.

Hinweis:
Mit den uns zur Verfügung stehenden Mechanismen kann die Ausgabe auf `cout` nur *von links nach rechts* und dabei gleichzeitig auch *von oben nach unten* erfolgen.



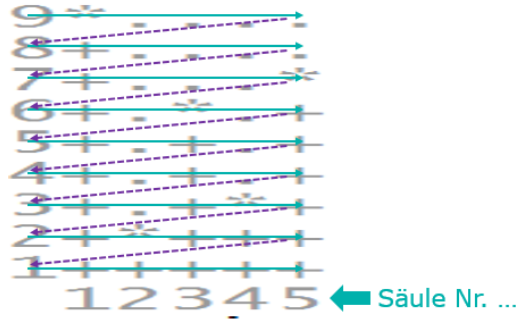
Legt man dieses Ausgabeverhalten über einen Beispiel-Testlauf …

Höhe der 1. Säule: 9
Höhe der 2. Säule: 2
Höhe der 3. Säule: 6
Höhe der 4. Säule: 3
Höhe der 5. Säule: 7

```
9*....
8+....
7+...*
6+.*.+
5+.+.+
4+.+.+
3+.+*+
2+*+++
1+++++
 12345  <- Säule Nr. ...
```

… so wird klar, dass das Programm alle „Höhenlinien-Werte“ von `9` bis `1` in dieser Reihenfolge durchlaufen muss, um dabei dann jeweils die fünf Eingabewerte nacheinander zu prüfen, ob sie "oberhalb" (*dann Punkt ausgeben*), "auf" (*dann Sternchen ausgeben*) oder "unterhalb" (*dann Pluszeichen ausgeben*) dieser „Höhenlinie“ liegen:

Höhe der 1. Säule: 9
Höhe der 2. Säule: 2
Höhe der 3. Säule: 6
Höhe der 4. Säule: 3
Höhe der 5. Säule: 7



Testläufe: *(Benutzereingaben sind zur Verdeutlichung unterstrichen)*

```
Bitte geben Sie die 1. Zahl ein: ? 99
Bitte geben Sie die 1. Zahl ein: ? -5
Bitte geben Sie die 1. Zahl ein: ? 9
Bitte geben Sie die 2. Zahl ein: ? 2
Bitte geben Sie die 3. Zahl ein: ? 6
Bitte geben Sie die 4. Zahl ein: ? 3
Bitte geben Sie die 5. Zahl ein: ? 7
9*....
8+....
7+...*
6+.*.+
5+.+.+
4+.+.+
3+.+*+
2+*+++
1+++++
 12345
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die 1. Zahl ein: ? 1
Bitte geben Sie die 2. Zahl ein: ? 2
Bitte geben Sie die 3. Zahl ein: ? 3
Bitte geben Sie die 4. Zahl ein: ? 4
Bitte geben Sie die 5. Zahl ein: ? 5
9.....
8.....
7.....
6.....
5....*
4...*+
3..*++
2.*+++
1*++++
 12345
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die 1. Zahl ein: ? 3
Bitte geben Sie die 2. Zahl ein: ? 3
Bitte geben Sie die 3. Zahl ein: ? 3
Bitte geben Sie die 4. Zahl ein: ? 3
Bitte geben Sie die 5. Zahl ein: ? 3
9.....
8.....
7.....
6.....
5.....
4.....
3*****
2+++++
1+++++
 12345
Drücken Sie eine beliebige Taste . . .
```

(Pflicht-) Aufgabe INF-05.02: Sudoku einlesen (zweidimensionale Arrays)

Sudoku ist ein Logikrätsel. In der üblichen Version ist es das Ziel, ein 9×9-Gitter mit den Ziffern 1 bis 9 so zu füllen, dass jede Ziffer in jeder Spalte, in jeder Zeile und in jedem Block (3×3-Unterquadrat) genau einmal vorkommt.

Schreiben Sie ein C++ Programm, welches ein „gelöstes" Sudoku einliest und in einem Array …

```
int sudoku[9][9] = {0};
```

… abspeichert. D.h. in dem Array sollen nur die 9x9 Zahlwerte abgespeichert werden, die „optischen Blocktrennzeichen" der Eingabe sollen in diesem Array nicht mehr vorkommen. Außerdem sollen die realen Zahlwerte abgespeichert werden, nicht die ASCII Codes der Zahlziffern (also eine „3" soll auch als Wert 3 abgespeichert werden).

Lesen Sie dazu die 11 Eingabezeilen zuerst einmal in ein Array …

```
string eingabe[11] = { "" };
```

… ein (es müssen zwei Einträge mehr sein wegen der beiden Zeilen mit den horizontalen Trennstrichen). Bitte beachten Sie, dass in der Eingabe keine Leerzeichen vorkommen, sondern anstelle dieser ein Punkt. Die Zahlen im Eingabetext sind immer einstellig.
Ihr Programm kann davon ausgehen, dass der Benutzer nur richtig formatierte Eingaben macht. *Die Gültigkeit der Zahlen (ob einstellig und gültig gemäß den Sudoku Regeln) muss daher nicht geprüft werden. Der Benutzer muss das Eingabe-Sudoku komplett selbst eingeben, inklusive der Punkte und Striche (senkrechte Striche und Minuszeichen).*

Anschließend soll das Sudoku wieder ausgegeben werden, und zwar mit anderen „Trennzeichen" als bei der Eingabe, siehe Testläufe.
Die Erzeugung der Ausgabe darf nur unter Verwendung der Werte in `sudoku[9][9]` geschehen, auf den Eingabetext `eingabe[]` darf dann nicht mehr zurückgegriffen werden.

Hinweis:
Kopieren Sie sich die Beispiel-Sudokus aus den Testläufen in einen separaten Editor oder in einen Kommentarblock in Ihrem C++ Quelltext. Dann können Sie von dort aus per *copy-paste* die Eingaben für Ihre Testläufe machen, ohne die Sudokus jedes Mal neu eintippen zu müssen.

Testläufe: *(Benutzereingaben diesmal **nicht** unterstrichen, da sonst zu unübersichtlich)*

```
Bitte geben Sie das Sudoku ein:
.5.1.4.|.8.6.9.|.7.2.3
.8.7.2.|.3.4.5.|.6.1.9
.9.6.3.|.2.1.7.|.5.4.8
-------|-------|-------
.6.2.8.|.1.3.4.|.9.5.7
.1.9.7.|.6.5.2.|.8.3.4
.4.3.5.|.7.9.8.|.1.6.2
```

```
-------|-------|-------
.2.4.6.|.9.7.1.|.3.8.5
.7.5.1.|.4.8.3.|.2.9.6
.3.8.9.|.5.2.6.|.4.7.1

Das Sudoku lautet:
;5;1;4;//;8;6;9;//;7;2;3
;8;7;2;//;3;4;5;//;6;1;9
;9;6;3;//;2;1;7;//;5;4;8
=======//=======//=======
;6;2;8;//;1;3;4;//;9;5;7
;1;9;7;//;6;5;2;//;8;3;4
;4;3;5;//;7;9;8;//;1;6;2
=======//=======//=======
;2;4;6;//;9;7;1;//;3;8;5
;7;5;1;//;4;8;3;//;2;9;6
;3;8;9;//;5;2;6;//;4;7;1
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie das Sudoku ein:
.9.4.6.|.3.1.8.|.2.7.5
.1.2.3.|.7.5.6.|.9.4.8
.5.8.7.|.2.4.9.|.6.3.1
-------|-------|-------
.8.1.4.|.9.2.5.|.7.6.3
.2.7.5.|.1.6.3.|.8.9.4
.6.3.9.|.8.7.4.|.1.5.2
-------|-------|-------
.3.6.8.|.5.9.2.|.4.1.7
.4.5.1.|.6.8.7.|.3.2.9
.7.9.2.|.4.3.1.|.5.8.6

Das Sudoku lautet:
;9;4;6;//;3;1;8;//;2;7;5
;1;2;3;//;7;5;6;//;9;4;8
;5;8;7;//;2;4;9;//;6;3;1
=======//=======//=======
;8;1;4;//;9;2;5;//;7;6;3
;2;7;5;//;1;6;3;//;8;9;4
;6;3;9;//;8;7;4;//;1;5;2
=======//=======//=======
;3;6;8;//;5;9;2;//;4;1;7
;4;5;1;//;6;8;7;//;3;2;9
;7;9;2;//;4;3;1;//;5;8;6
Drücken Sie eine beliebige Taste . . .
```

Für das Aufgaben-Tutorium am Freitag 4.12.2020, als freiwillige Aufgabe:

(Freiwillige) Aufgabe INF-05.03: Würfeln (Array, Schleife)

Schreiben Sie ein C++ Programm, welches `6000`-mal eine Zufallszahl zwischen `1` und `6` berechnet (Spielwürfel) und am Ende ausgibt, wie oft jede Zahl gewürfelt wurde.
Hinweis: Zu Erzeugung der Zufallszahlen binden Sie den folgenden Header ein:

```
#include <time.h>
```

Anschließend können Sie mit …

```
srand(time(NULL));
```

… den Zufallsgenerator initialisieren. Mit …

```
int zufallszahl = rand();
```

… wird dann eine Zufallszahl im Bereich von `0` bis höchstens `32767` erzeugt.

Häufig wird mit einer Modulo-Berechnung der Bereich der Zufallszahlen eingeschränkt; so auch in unserem Fall, wo wir die Zahlen auf den Wertebereich `1` bis `6` einschränken wollen.
Da der Generator Zufallszahlen ab `0` generiert, "korrigieren" wir mittels modulo die generierte Zufallszahl (im Bereich `0` bis `32767`) auf den Wertebereich `0` bis `5` *(wie kann man dies umsetzen? dies herauszufinden ist Teil der Aufgabe…)* und addieren dann `1` dazu, um den Wert in den Bereich `1` bis `6` zu "versetzen" …

Weitere Informationen zur `rand()`-Funktion finden Sie unter:

http://www.cplusplus.com/reference/cstdlib/rand/

Testläufe: *(keine Benutzereingaben; in ihren Testläufen sollten die ausgegebenen Werte anders sein, da die Zahlen ja zufallsgeneriert sein sollen …)*

```
Anzahl 1-er Wuerfe: 1077
Anzahl 2-er Wuerfe: 1018
Anzahl 3-er Wuerfe: 992
Anzahl 4-er Wuerfe: 988
Anzahl 5-er Wuerfe: 892
Anzahl 6-er Wuerfe: 1033
Drücken Sie eine beliebige Taste . . .
```

---

*Als **weitere freiwillige Aufgabe**, aber **nicht für das Aufgaben-Tutorium**! Sollten Sie Fragen zu dieser Aufgabe haben oder ihre Lösung "korrektur-lesen" lassen wollen, so nutzen Sie bitte **nur das Fragen-Tutorium** dazu!*

(Freiwillige) Aufgabe INF-05.04: Ausführungsmodell, Stack

Zeichnen Sie jeweils den Stack-Zustand an den drei mit Pfeilen markierten Stellen des unten angegebenen Programms. Verwenden Sie dazu das in der Vorlesung vorgestellte Ausführungsmodell.
Sie sollen also insgesamt *drei* Stack-Bilder zeichnen.

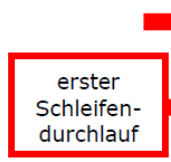
Geben Sie außerdem die Ausgabe des *gesamten* Programms an (d.h. eine komplette Ausgabe, wenn das Programm komplett durchgelaufen ist; dies ist unabhängig von den drei Stackbildern!).
Versuchen Sie (als Vorbereitung auf die GIP Klausur), die Ausgabe des Programms nur aus ihrer "Papierbearbeitung des Codes" abzuleiten, d.h. (idealerweise) ohne das Programm am Rechner auszuführen.

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

int a1;
double d1 = 0.5;

int main()
{
        int a = 1, b = 2;

        for (int i = 3; i > 0; i--)
        {
          int b = 2;
            a = a + i;
            b = b + 3;
        }
        cout << "a: " << a << endl;
        cout << "a1: " << a1 << endl;
        a = a + 7;
        cout << "d1: " << d1 << endl;
        int c = 5;
        for (a = 5; a < 6; a++)
        {
            double d1 = 3.14;
            cout << "a: " << a << endl;
            cout << "d1: " << d1 << endl;
        }
        system("PAUSE");
        return 0;
}
```



Hinweis: Advent of Code

Wenn Sie gerne weiteres, interessantes "Programmierfutter" hätten, dann möchte ich Sie auf den "Adventskalender" *Advent of Code* hinweisen, der in den letzten Jahren und auch dieses Jahr zur Adventszeit veröffentlicht wurde/wird: https://adventofcode.com/. Dort wird wie beim Adventskalender üblich jeden Tag eine neue Programmieraufgabe veröffentlicht, die man in einer beliebigen Programmiersprache (also warum nicht auch in C++) lösen soll. Die Aufgaben bleiben auch nach der Adventszeit verfügbar und auch die Aufgaben der letzten Jahre sind immer noch verfügbar, z.B. https://adventofcode.com/2015 …

Man muss sich auf dieser Webseite allerdings anmelden, da jeder Teilnehmer eine spezifisch für ihn variierte Eingabedaten bekommt mit entsprechend spezifischen Ergebnisdaten, die dann zur Prüfung hochgeladen werden können. Der Code des eigenen Lösungsprogramms kann und soll nicht hochgeladen werden (reine "Black Box" Prüfung, wie

das auch der Jenkins macht, der schaut auch nicht in ihren hochgeladenen Code, sondern prüft nur die Eingaben/Ausgaben ihres Programms …)

Der *Advent of Code* ist "eigentlich" ein Wettbewerb, bei dem die ersten 100 hochgeladenen korrekten Lösungen Punkte bekommen und es entsprechend ein "Leaderboard" gibt. Aber wenn man "nur" das Ziel hat, eine Lösung zu finden und durch Hochladen der Ergebnisdaten die Korrektheit prüfen zu lassen, dann ist das ein Vorgehen in gewisser Weise ähnlich zu unseren Offline-Aufgaben (auch wenn Sie "bei uns" ihren Programmcode hochladen, bei *Advent of Code* nicht…).

Die Aufgaben beim *Advent of Code* sind allerdings deutlich komplexer als das, was wir im Rahmen der GIP Veranstaltung an Aufgaben behandeln und die Aufgaben setzen eine Programmiererfahrung voraus, die weit über das hinausgeht, was im Rahmen des GIP Moduls als Lernziel angesetzt ist. Auch wird man für eine "elegante" Lösung einiger Aufgaben ggfs. dynamische Datenstrukturen verwenden, die wir erst später (bald, noch im Dezember) in diesem Semester behandeln werden.

Wenn Sie also "Programmier-Herausforderungen suchen", dann wäre das eine Möglichkeit …

Aber: Die Inhalte des Studiums sollten aus meiner Sicht Priorität bei Ihnen haben, denn das ist ihr aktueller Beruf. Beim Advent of Code mitzumachen ist dann zwar für GIP "berufsunterstützend", aber wenn es sinnvoller wäre, Zeit in andere Studienfächer zu investieren, dann tun Sie dies bitte zu ihrem eigenen Besten… Die Advent of Code Aufgaben können Sie dann ja später immer noch bearbeiten (diese bleiben auch jahrelang nach der Adventszeit verfügbar) …