Crashkurs

Aufgaben:	`http://icpc.iti.kit.edu/public/wettbewerbe/getconnected17`
Einsendung:	`http://domjudge.iti.kit.edu/main/team`
Referenzen:	`http://cppreference.com`
	`http://www.cplusplus.com/reference`
	`http://docs.oracle.com/javase/7/docs/api`

1 Kompilieren auf der Konsole

Zuerst legt man in einem Editor seiner Wahl die zu kompilierende Quelldatei an. Üblicherweise hängt man für C++-Dateien `.cpp` und für Java-Dateien `.java` als Dateierweiterung an, damit der Editor das richtige Syntaxhighlighting wählt. Man beachte auch, dass in Linux Groß- und Kleinschreibung unterschieden wird. Außerdem bereiten Leerzeichen einige Problem auf der Konsole. Es empfiehlt sich also, alle Dateinamen kleinzuschreiben und keine Leerzeichen zu verwenden.

Bei Java sollte weiterhin der Dateiname der öffentlichen Klasse in der Datei entsprechen. Eine Datei `Foo.java` sollte also nur eine öffentliche Klasse `Foo` beinhalten.

Zum eigentlichen Kompilieren öffne man dann eine Konsole. Man wird mit einem Prompt der Form

```
name@rechner(~)$
```

begrüßt. Vor dem Dollar sieht man den aktuellen Pfad in runden Klammer. Dabei steht `~` für das Home-Verzeichnis. Mit `cd` wechselt man das aktuelle Verzeichnis und mit `ls` listet man den Inhalt des aktuellen Verzeichnisses auf.

Zum Beispiel wechselt man mit

```
cd foo
```

in das Unterverzeichnis `foo` des aktuellen Verzeichnisses. Zum Kompilieren gehe man nun in das Verzeichnis, in dem die Quelldateien liegen.

Zum Übersetzen von C++-Programmen gibt man dann folgendes ein:

```
g++ -std=c++14 -Wall foo.cpp
```

Es wird eine ausführbare Datei namens `a.out` erzeugt. Der Parameter `-Wall` sorgt dafür, dass Warnungen über fragwürdigen Code ausgeben werden. Mit `-std=c++14` stellt man den zu verwendeten Dialekt von C++ ein.

Zum Ausführen verwendet man folgenden Befehl:

```
./a.out
```

Man kann die Ein- und Ausgabe auch umlenken:

```
./a.out < in > out
```

Hier liest das Programm die Eingabe aus der Datei `in` ein und gibt die Ausgabe in die Datei `out` aus. Man kann auch nur die Eingabe oder nur die Ausgabe umlenken. Falls nicht umgelenkt wird, ist die Eingabe oder die Ausgabe an die Konsole gebunden.

Java-Programme werden ähnlich kompiliert:

```
javac Foo.java
```

Zum Ausführen benutzt man im Gegensatz aber folgendes:

```
java Foo
```

Auch bei Java kann man die Ein- und Ausgabe umlenken:

```
java Foo < in
java Foo > out
java Foo < in > out
```

2 Weitere Hinweise

2.1 Vergleich mit der erwarteten Ausgabe

Um sicherzustellen, dass ein Programm die erwartete Ausgabe hat, sollte man `diff` verwenden, da ein manueller Vergleich korrekte Whitespaces und kleine Zahlendreher nicht mit Sicherheit erkennen kann.

Um zum Beispiel das Programm `foo` mit der Eingabe `sample.in` aufzurufen und die Ausgabe mit der erwarteten Ausgabe in `sample.out` zu vergleichen, benutzt man folgenden Befehl:

```
./a.out < sample.in > my.out
diff my.out sample.out
```

Das Programm `diff` vergleicht die beiden angegeben Dateien. Es gibt nichts (!) aus, falls die Dateien übereinstimmen. Andernfalls werden die Unterschiede angezeigt.

2.2 Mehrere Kommandos ausführen

Mit `&&` kann man 2 Befehle nacheinander ausführen, wobei der zweite Befehl nur dann ausgeführt wird, falls der erste erfolgreich war. Mit der oben erläuterten Verwendung von `diff` ergibt sich damit die folgende Kommandozeile zum Testen eines Programms:

```
g++ -std=c++14 -Wall foo.cpp && ./a.out < sample.in > my.out && diff my.out sample.out
```

Um Tipparbeit zu sparen kann man auf der Konsole die Pfeiltasten verwenden um Befehle zu wiederholen.

3 Beispielprogramme

Hier folgen nun noch einige Beispielprogramme zur Lösung des `SimpleSum`-Problems, die insbesondere die Ein- und Ausgabe in der jeweiligen Sprache illustrieren.

3.1 C++

In C++nimmt man `cout` und `cin` aus `iostream` oder die C-Funktionen `printf` und `scanf` aus `cstdio`:

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

int main() {
        int n, sum = 0;
        cin >> n;
        for (int i = 0; i < n; ++i) {
                int tmp;
                cin >> tmp;
                sum += tmp;
        }
        cout << sum << "\n";

        return 0;
}
```

3.2 Java

Die einfachste Möglichkeit zum Parsen von Eingaben in Java ist die `Scanner`-Klasse:

```
import java.util.Scanner;

class Main {
        public static void main(String[] args) {
                Scanner s = new Scanner(System.in);

                int n = s.nextInt();
                int sum = 0;
                for (int i = 0; i < n; ++i) {
                        sum += s.nextInt();
                }

                System.out.println(sum);
        }
}
```