


# Stream Input und Output

 **Aufgabe 9.1** Überlegen Sie sich bei den folgenden fünf Code-Fragmenten, was jeweils die erwartete Ausgabe in der mit einem Kommentar markierten letzten Zeile ist.


```
std::istringstream iss("Halli Hallo");
std::string s; iss >> s;
std::cout << iss.good() << iss.eof() << iss.fail() << std::endl; // 1

std::istringstream iss("Halli Hallo");
std::string s; iss >> s >> s;
std::cout << iss.good() << iss.eof() << iss.fail() << std::endl; // 2

std::istringstream iss("Halli Hallo");
std::string s; iss >> s >> s >> s;
std::cout << iss.good() << iss.eof() << iss.fail() << std::endl; // 3


std::istringstream iss("Halli Hallo");
double d; std::string s; iss >> d >> s >> s;
std::cout << iss.good() << iss.eof() << iss.fail() << std::endl; // 4

std::istringstream iss("Halli Hallo");
std::string s; std::getline(iss, s, 'o');
std::cout << iss.good() << iss.eof() << iss.fail() << std::endl; // 5
```

 **Aufgabe 9.2** Eine Programmiererin wünscht sich einen Input-Stream-Manipulator, mit dem alle gleichen aufeinanderfolgenden Zeichen ignoriert werden, so dass aber gleichzeitig die Anzahl ignoriierter Zeichen automatisch berechnet wird. Konkret stellt sie sich die Syntax `std::cin >> Count('x', i)` vor mit der Bedeutung, dass alle direkt folgenden Zeichen 'x' aus dem Stream entfernt werden und die Variable `i` vom Typ `int` um deren Anzahl vergrößert wird. Folgender Code scheint auf den ersten Blick den Wunsch der Programmiererin zu erfüllen. Sehen Sie vielleicht auf den zweiten Blick ein Problem?

```
struct Count { char m_ignore; int& m_counter; };

std::istream& operator>>(std::istream& is, const Count& c) {
    while(is.get() == c.m_ignore)
        c.m_counter++;
    return is;
}
```

 **Aufgabe 9.3** Implementieren Sie den Eingabeoperator und einen dazu passenden Ausgabeoperator für die folgende Klasse `Student`. Mit "passend" ist gemeint, dass ein ausgegebenes und dann von den ausgegebenen Daten wieder eingelesenes Objekt die gleichen Attribut-Werte hat wie vorher. Abgesehen von dieser Eigenschaft haben Sie freie Wahl, wie das Objekt in Textform repräsentiert wird. Achten Sie jedoch darauf, dass unsinnige oder nicht der Spezifikation entsprechende Eingaben beim Einlesen erkannt werden und der Input-Stream in einen der Situation entsprechenden Zustand versetzt wird.

```
class Student {
    bool m_bachelor; // Boolean mit Wahr oder Falsch
    std::string m_name; // Zeichenkette mit Buchstaben und Leerzeichen
    std::vector<double> m_grades; // Vektor mit beliebig vielen Zahlen

    friend std::ostream& operator<<(std::ostream& os, const Student& s);
    friend std::istream& operator>>(std::istream& is, Student& s);
};
```