Magische Zahlen

>> Beim Programmieren schreibt man mitunter Zahlen in den Code, deren Bedeutung für andere auf den ersten Blick nicht klar ersichtlich ist. Warum sind diese problematisch und wie können wir besser damit umgehen?

>> Solche festen Werte nennt man auch **Magic Numbers** oder „Magische Zahlen“.

Einige der Programme, die wir uns angeschaut haben, enthalten möglicherweise auch solche magischen Zahlen.

>> Beispiel: **Block-Pyramide** in einem Super-Mario-Spiel

Pyramidenhöhe: maximal 23

Es fehlt Kontext, z.B.: Höhe des Terminalfensters ist standardmäßig 24 Zeilen.

Dadurch schlechte Lesbarkeit und erschwerte Wartung des Codes.

```
card deal_cards(deck name)
{
    for (int i = 0; i < 52; i++)
    {
        // Karte austeilen
    }
}
```

Sehen Sie die Magic Number?

Sehen Sie, dass hier ein Problem entstehen könnte? Etwa, wenn diese Funktion eine von mehreren ist, die Kartenspiele manipulieren.

Erster Lösungsversuch

```
card deal_cards(deck name)
{
    int deck_size = 52;
    for (int i = 0; i < deck_size; i++)
    {
        // Karte austeilen
    }
}
```

Das behebt ein Problem, führt aber zu einem anderen.

Wenn *deck_size* global deklariert würde, könnte es passieren, dass eine andere Funktion in unserem Programm *deck_size* versehentlich manipuliert.

Präprozessor-Direktive #define

C bietet die Möglichkeit, mit einer Präprozessor-Direktive (auch Makro genannt) symbolische Konstanten zu erstellen.

```
#define NAME ERSATZ
```

Beim Kompilieren Ihres Programms durchsucht der Kompiler Ihren Code und ersetzt NAME durch ERSATZ.

#include ist also wie Kopieren/Einfügen, #define wie Suchen/Ersetzen.

Präprozessor-Direktive #define

C bietet die Möglichkeit, mit einer Präprozessor-Direktive (auch Makro genannt) symbolische Konstanten zu erstellen.

```
#define PI 3.14159265
```

Beim Kompilieren Ihres Programms durchsucht der Kompiler Ihren Code und ersetzt `PI` durch `3.14159265`.

#include ist also wie Kopieren/Einfügen, #define wie Suchen/Ersetzen.

Präprozessor-Direktive #define

C bietet die Möglichkeit, mit einer Präprozessor-Direktive (auch Makro genannt) symbolische Konstanten zu erstellen.

```
#define KURS "Inf-Einf-B"
```

Beim Kompilieren Ihres Programms durchsucht der Kompiler Ihren Code und ersetzt `KURS` durch `"Inf-Einf-B"`.

#include ist also wie Kopieren/Einfügen, #define wie Suchen/Ersetzen.

```
#define DECKSIZE 52


card deal_cards(deck name)
{
    for (int i = 0; i < DECKSIZE; i++)
    {
        // Karte austeilen
    }
}
```

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

Magic Numbers:
Zahlen im Code
ohne klare Bedeutung

Beste Lösung:
Präprozessor-Direktive
#define für symbolische
Konstanten

Probleme: Beeinträchtigen
Lesbarkeit und Wartbarkeit
des Codes

Vorteile von #define:
Klarheit, Unveränderlichkeit
und Flexibilität

Erste Verbesserung:
Variablen mit
aussagekräftigen
Namen

Alternative:
const für typsichere,
unveränderbare Variablen

„Könnte man nicht auch eine globale Variable verwenden, die mit dem Modifier *const* als unveränderbar gekennzeichnet wird?“

Ja, das geht.

Hier ein Beispiel:

```
const int max_students = 30;
```

Es gibt Unterschiede im Detail (für uns nicht relevant).

Ob #define oder const verwendet wird, hängt von spezifischen Anforderungen und Präferenzen ab.