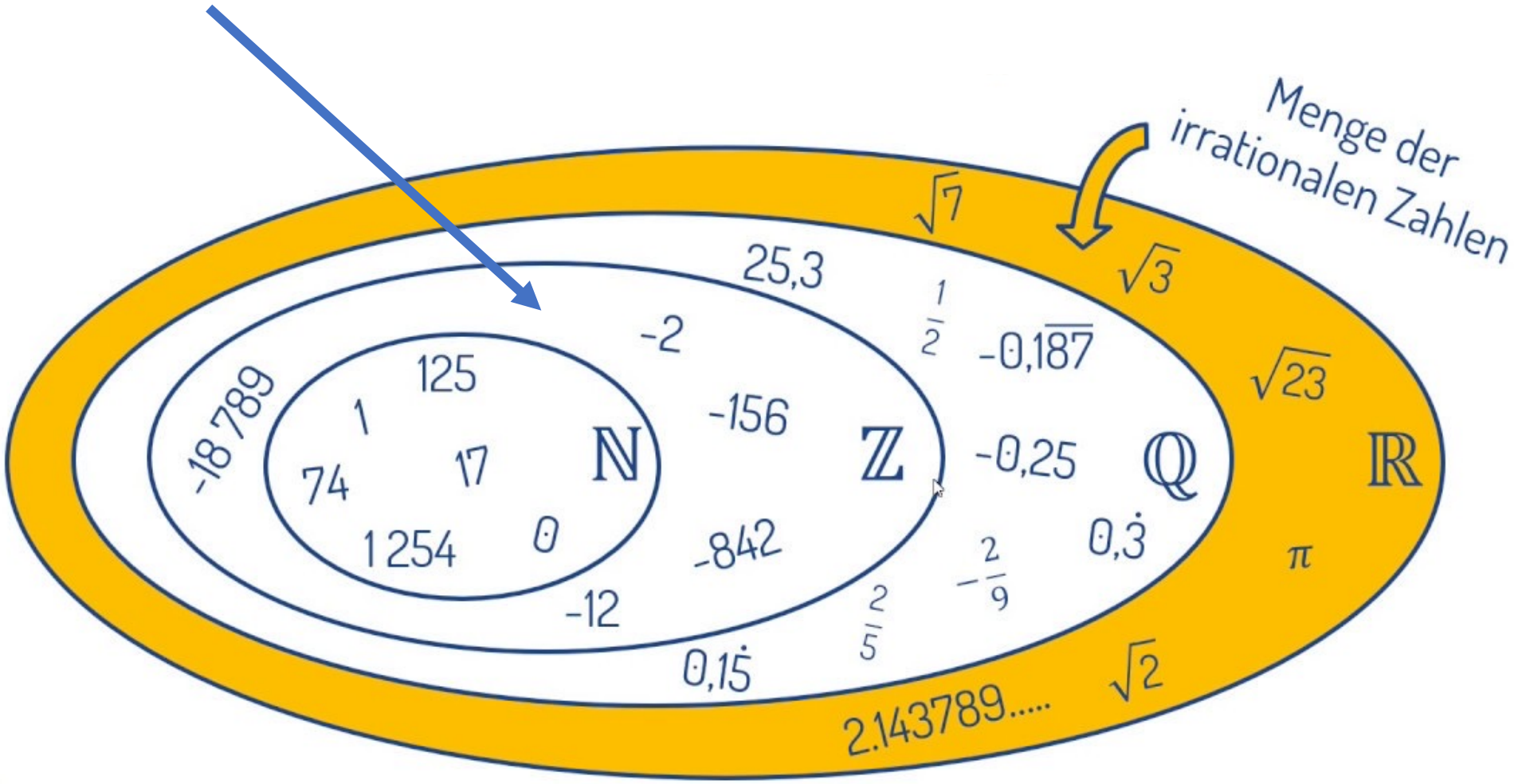
Ganze Zahlen & Debugging in C#

EF IF

Ganze Zahlen in C#

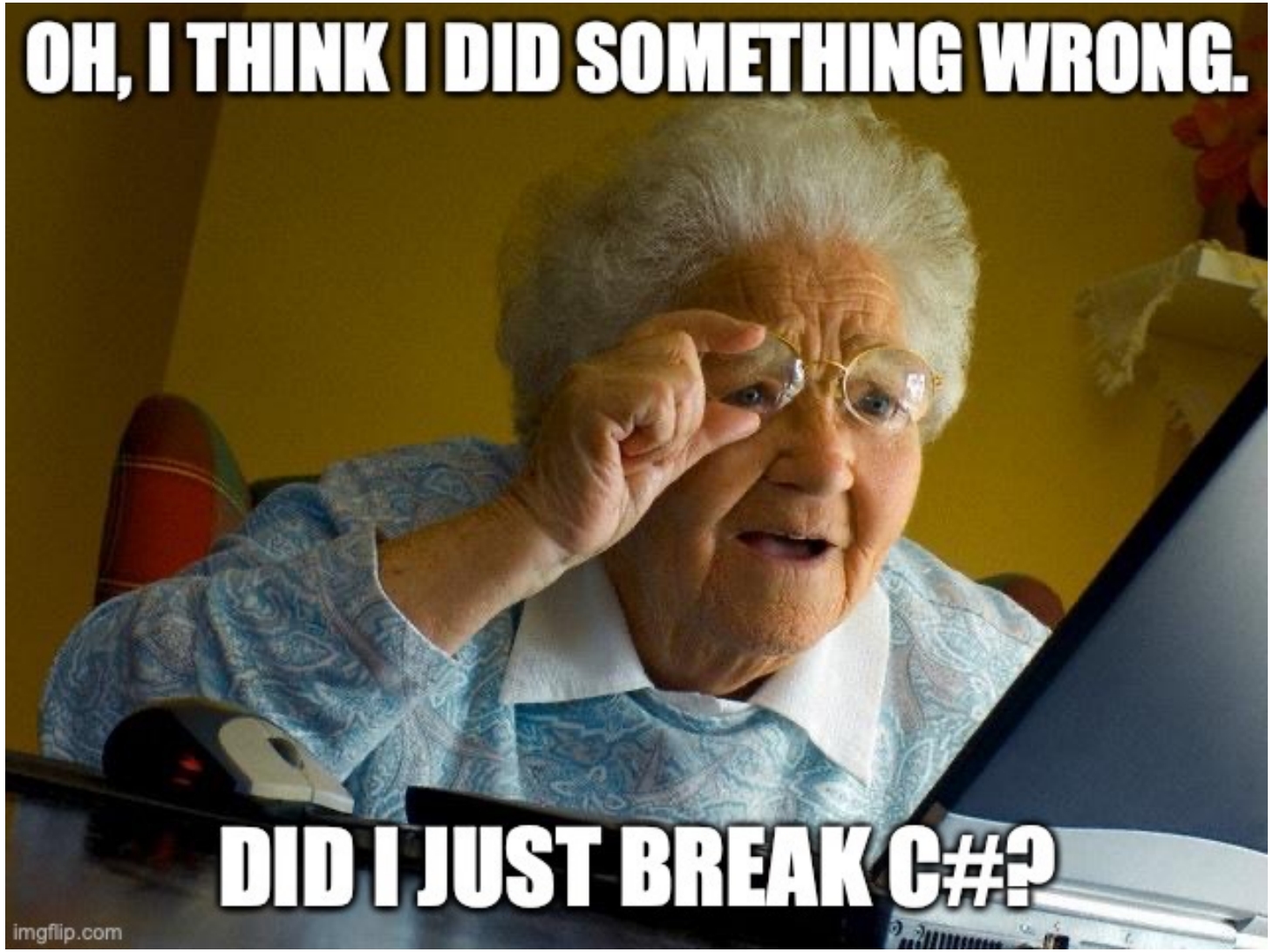
Ganze Zahlen in der Mathematik



Ganze Zahlen in C#

- Haben gesehen in Aufgabe: merkwürdiges Verhalten von C#

```
int x = 1000000000;
int inc = 1000000000;
Console.WriteLine(x);           // -> 1000000000
Console.WriteLine(x + inc);     // -> 2000000000
Console.WriteLine(x + 2 * inc); // -> - 1294967296
Console.WriteLine(x + 3 * inc); // -> - 294967296
Console.WriteLine(x + 4 * inc); // -> 705032704
```


OH, I THINK I DID SOMETHING WRONG.
DID I JUST BREAK C#?
imgflip.com

Ganze Zahlen in C#

- Grund für dieses Verhalten?

```
int x = 1000000000;
int inc = 1000000000;
Console.WriteLine(x);           // -> 1000000000
Console.WriteLine(x + inc);     // -> 2000000000
Console.WriteLine(x + 2 * inc); // -> - 1294967296
Console.WriteLine(x + 3 * inc); // -> - 294967296
Console.WriteLine(x + 4 * inc); // -> 705032704
```

- Ein int (Integer) hat **fest vorgegebene Speichergrösse**

Ganze Zahlen in C#

- Einem int in C# stehen genau 4 Bytes zur Verfügung.
- Aufgabe:
  1. Es stehen n Bits zur Verfügung, um ein int zu speichern. Bestimme grössten und kleinsten int.
  2. Grösster und kleinster int in C#?
- Lösungen
  1. Für n-Bits: $2^{n-1} - 1 \ \& \ -2^n$
  2. C#:
     - $4 \cdot 8$-Bits
     - $2^{31} - 1 = 2147483647 \quad \& \ -2^{31} = -2147483648$

Ganze Zahlen in C#

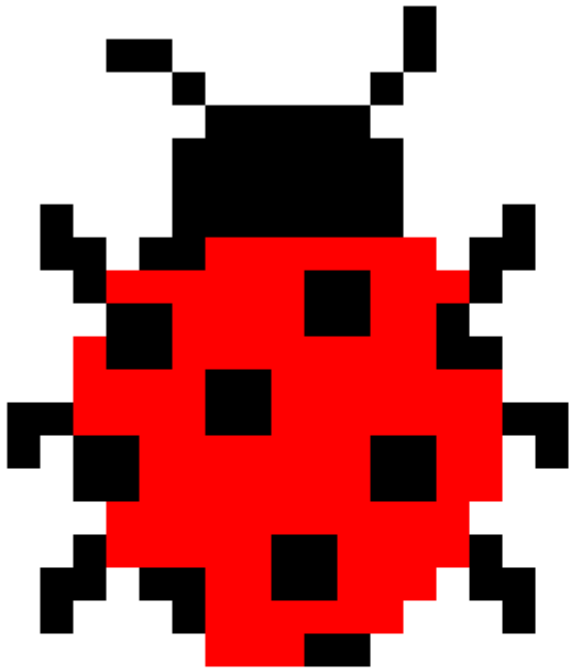
- Max. int in C#: $2'147'483'647$
- Vermögen Elon Musk (2023): $218'600'000'000\$$
- Int in C# kann also nicht einmal Vermögen von E. Musk speichern!
- Das reicht nicht!
- Ausweg? Ideen?
- Andere Datentypen
  - Unsigned integer **uint**: nach wie vor 8-Bytes aber nur positiv
  - **long**: wie int aber mit 16-Bytes
  - Unsigned long **ulong**

Ganze Zahlen

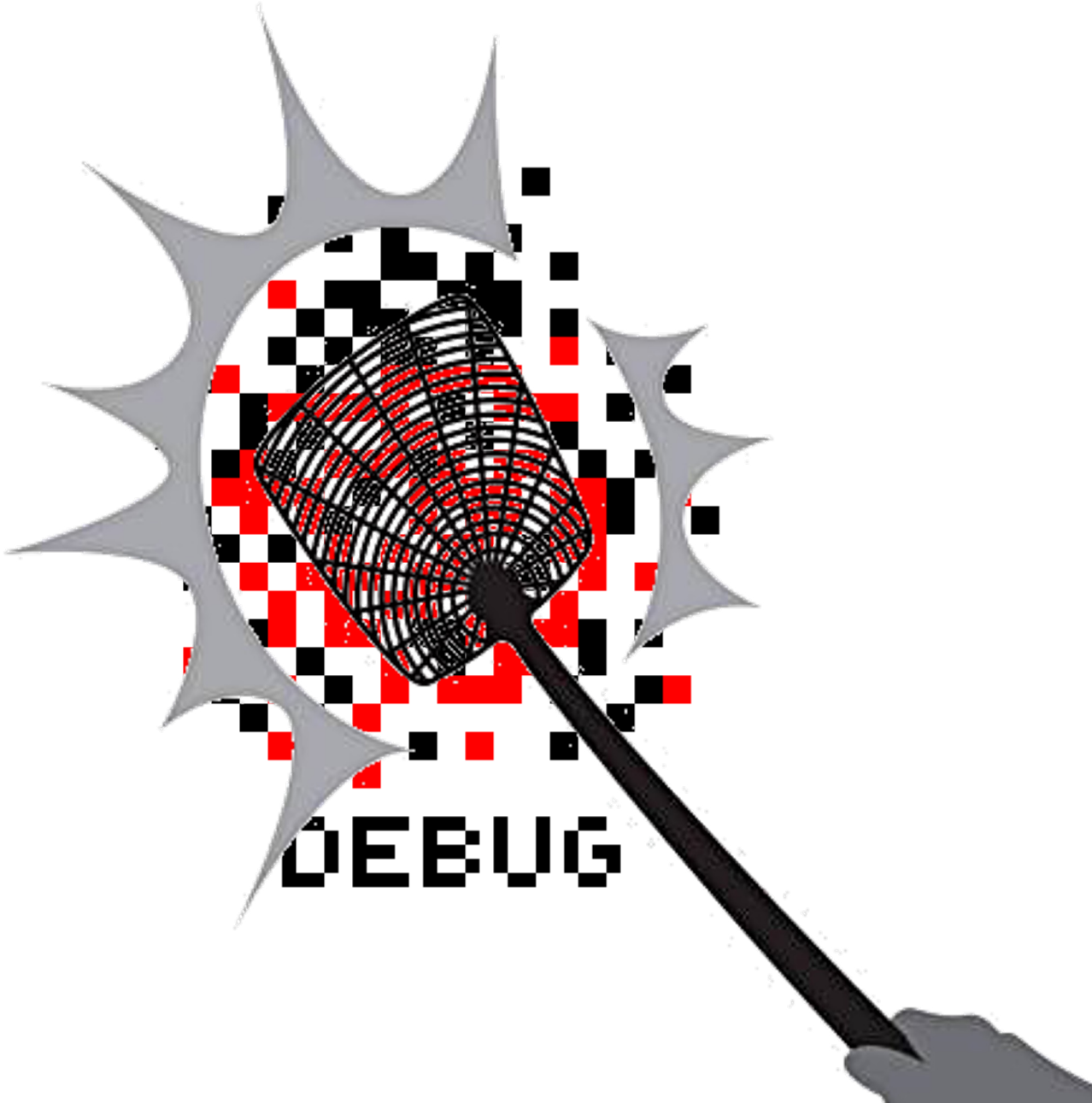
- Je nach Verwendung richtigen Datentyp verwenden:
  - 2-Byte: short & ushort
  - 4-Byte: int & uint
  - 8-Byte: long & ulong
- Typisches Vorgehen:
  - (u)int verwenden wenn reicht
  - Wenn nicht: (u)long

Debugging

Debugging



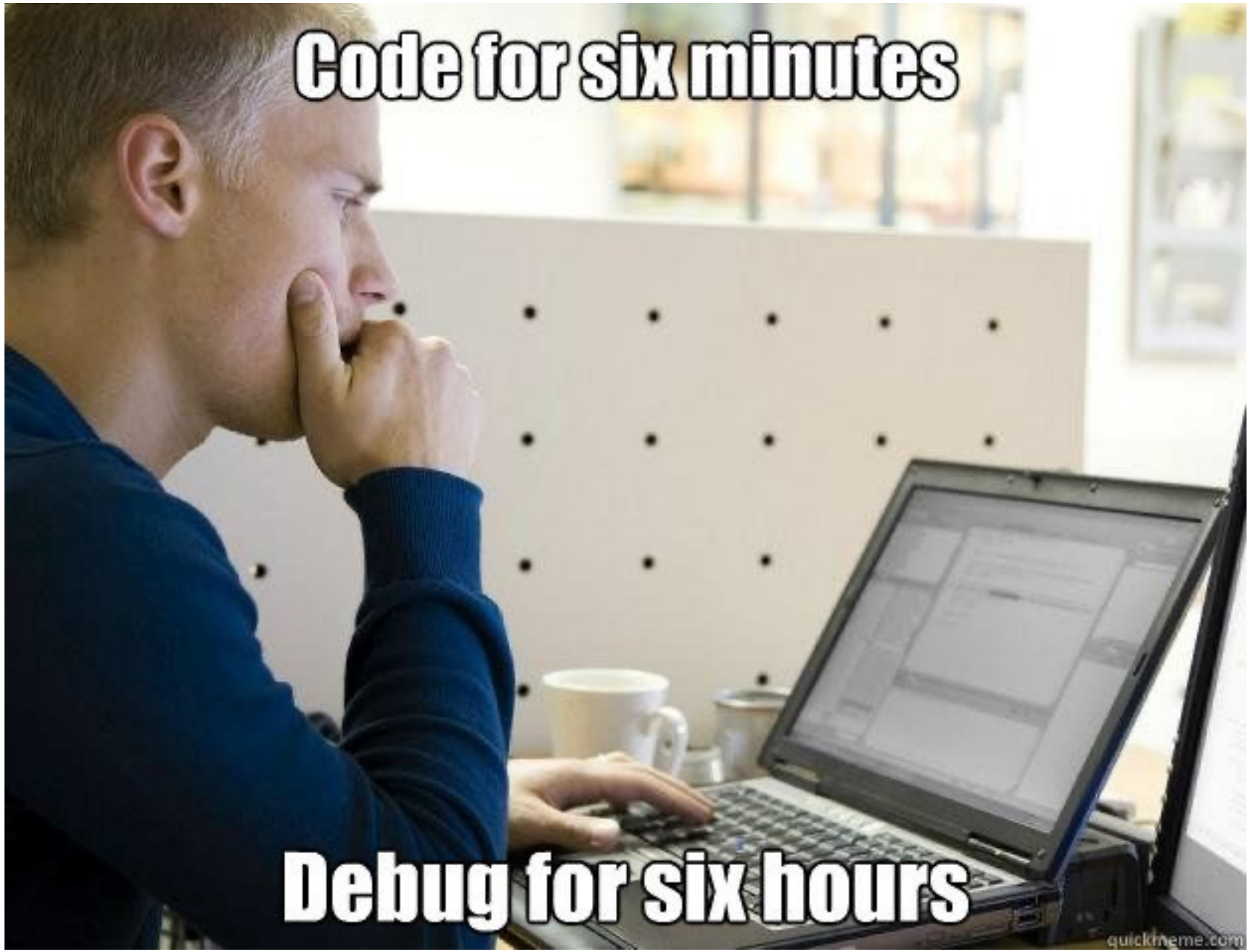
BUG

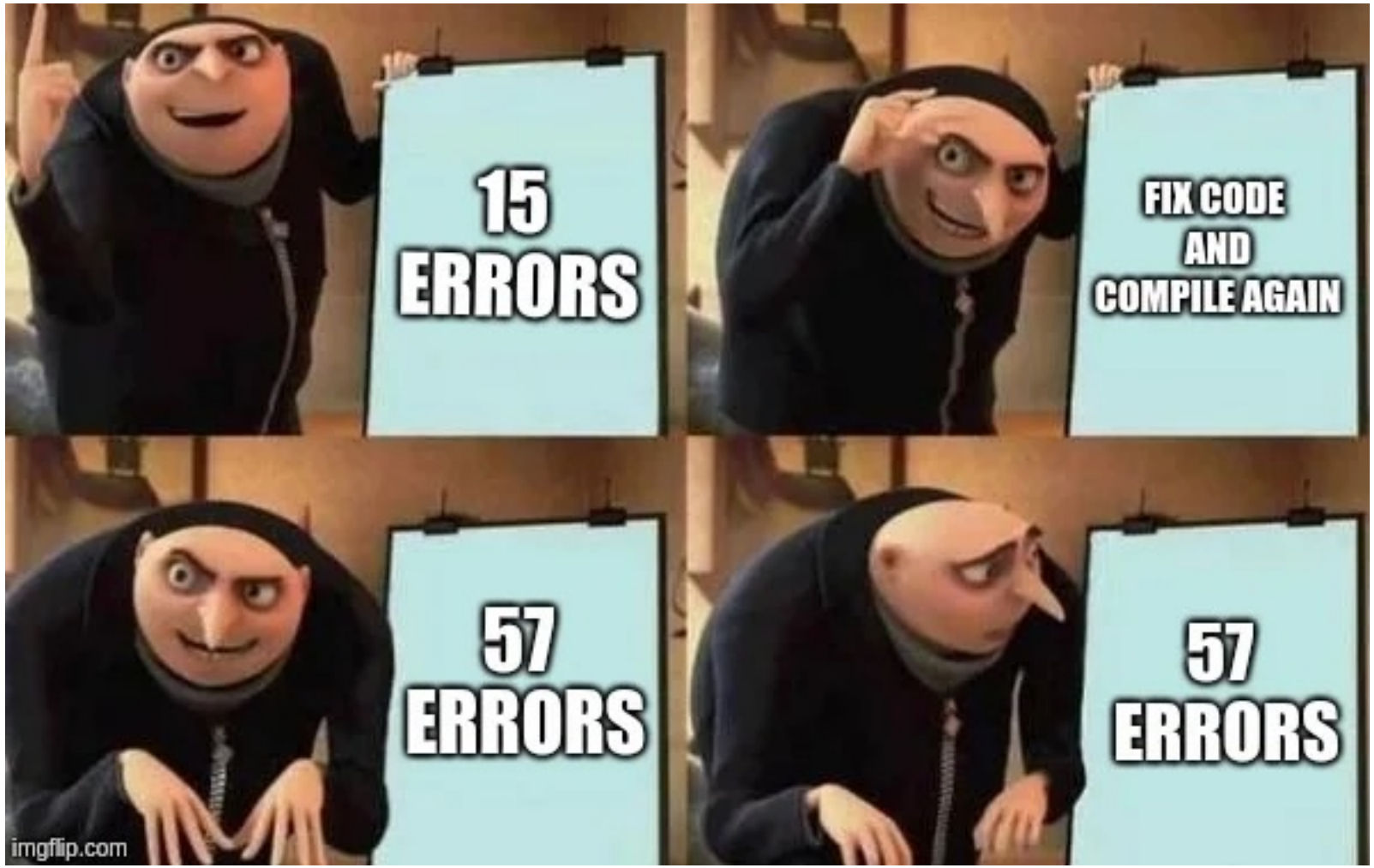


DEBUG

Debugging

- **Bug**: Fehler im Code
- **Debugging**: Bugs finden und eliminieren


Code for six minutes
Debug for six hours
quickmeme.com


15
ERRORS
FIX CODE
AND
COMPILE AGAIN
57
ERRORS
57
ERRORS
imgflip.com

SPEND 24HRS DEBUGGING
FINALLY! A DIFFERENT ERROR MESSAGE
imgflip.com

Fehler beim Programmieren

- Als Programmierer:in verbringt viel Zeit mit Debugging
- Kann viel Frust verursachen
- Wie kann Debugging-Zeit minimieren?
- Gleich einige Tipps zum besseren Programmieren
- Vorher aber …

Fehler beim Programmieren

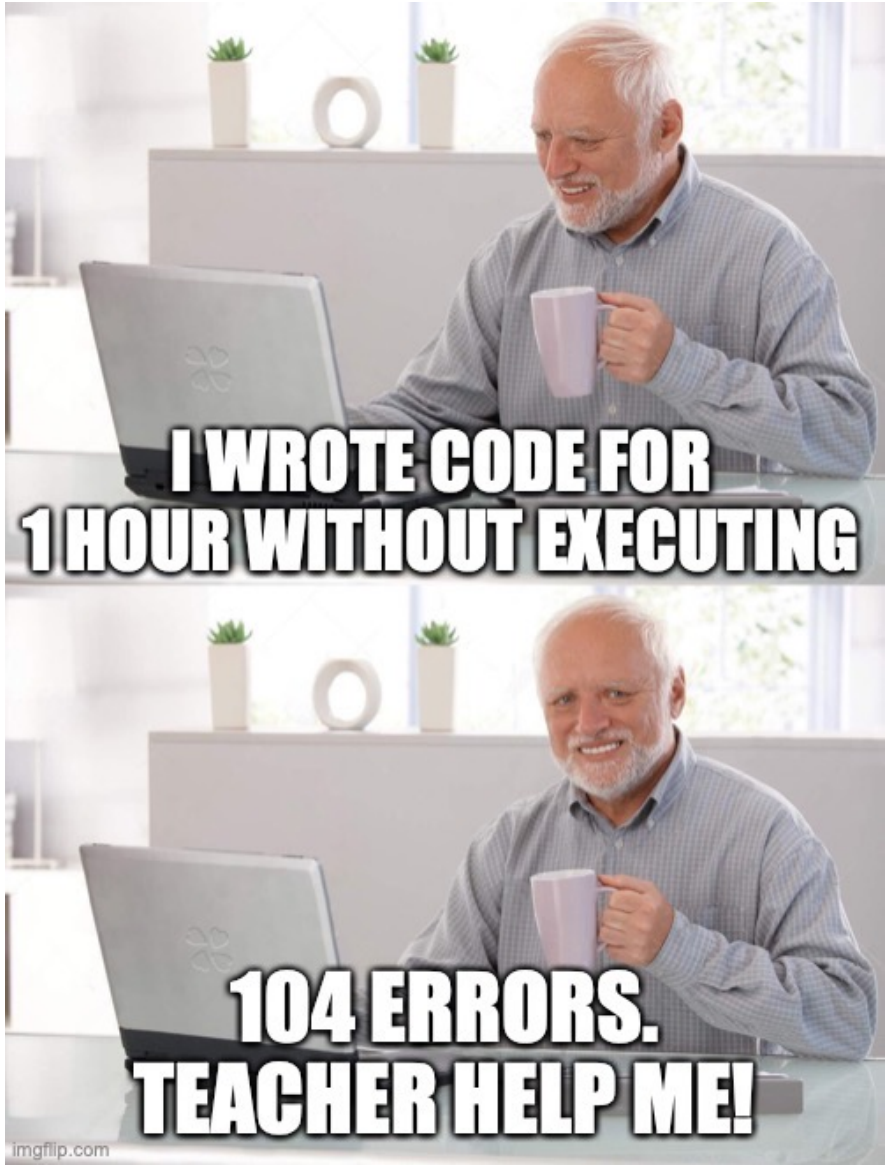
- Unterschiedliche Arten von Fehlern:
- **Syntaxfehler**:
  - Verstoss gegen Regeln der Programmiersprache
  - ‘Rechtschreibfehler’, ‘Grammatikfehler’
- **Laufzeitfehler**:
  - Werden erst bemerkt, wenn Code ausgeführt wird
  - Z.B. Division durch Null
- **Logikfehler**:
  - Code führt fehlerfrei aus, Resultat aber falsch
  - Fehlüberlegung der Programmierer:in
  - z.B. falsche Implementierung von Algorithmus

Tipp 1: Problem verstehen & Plan machen

- Beispiel: Implementiere den Bubble-Sort Algorithmus
- Fehler Nr. 1 bei SuS: Beginnen sofort mit Programmieren
- Wichtig: Du musst **Algorithmus zuerst selbst komplett verstehen**, ansonsten kannst du auch nicht programmieren (also ‘dem Computer beibringen’).
- Besseres Vorgehen:
  - **Algorithmus verstehen**: Auf Blatt Papier Algorithmus skizzieren, Beispiele machen, mit Kolleg:in diskutieren
  - **Plan für Code aufstellen:** Welche Variablen, Funktionen usw. brauche ich?
    - Kleinere Aufgaben: im Kopf
    - Grössere Projekte: aufschreiben!
  - **Programmieren!**
- Vermeide Logikfehler

Tipp 2: In kleinen Schritten programmieren.

- Typischer Fehler: Viel Code schreiben ohne auszuführen
- Besseres Vorgehen am Anfang:
  - Zeile Code schreiben
  - -> Ausführen
  - -> Bugs beheben
  - -> nächste Zeile schreiben
  - -> …
- Mehr Erfahrung -> mehr Code auf 1x
- Vermeidet zu viele Fehler auf 1x



Tipp 3: Fehlermeldungen *lesen*

- Gute IDE gibt (meist) nützliche Fehlermeldungen aus
- In Visual Studio:
  - Rot unterstrichen
  - Gibt Tipps
- Lese diese genau!
- Beispiel:

- Syntaxfehler eliminieren

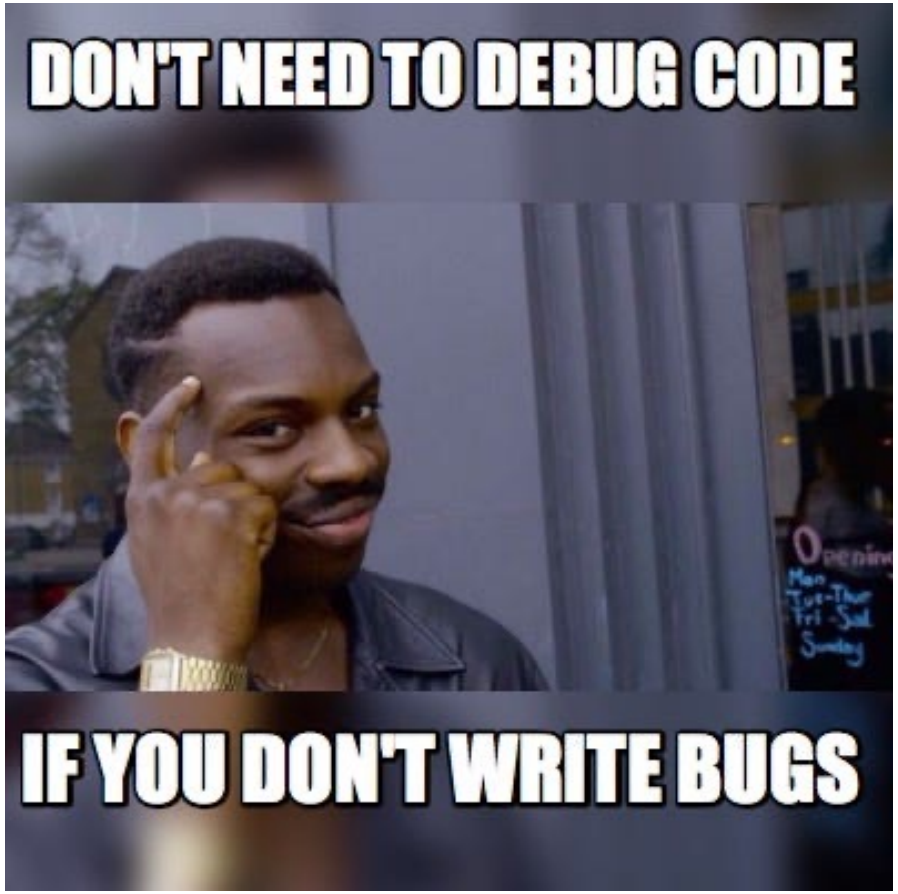
Tipp 4: Git verwenden

- Software zur Versionsverwaltung verwenden
- Standard: Git
- Cloud-Backup: GitHub
- Ein anderes Mal mehr dazu

Tipp 5: Debugger verwenden

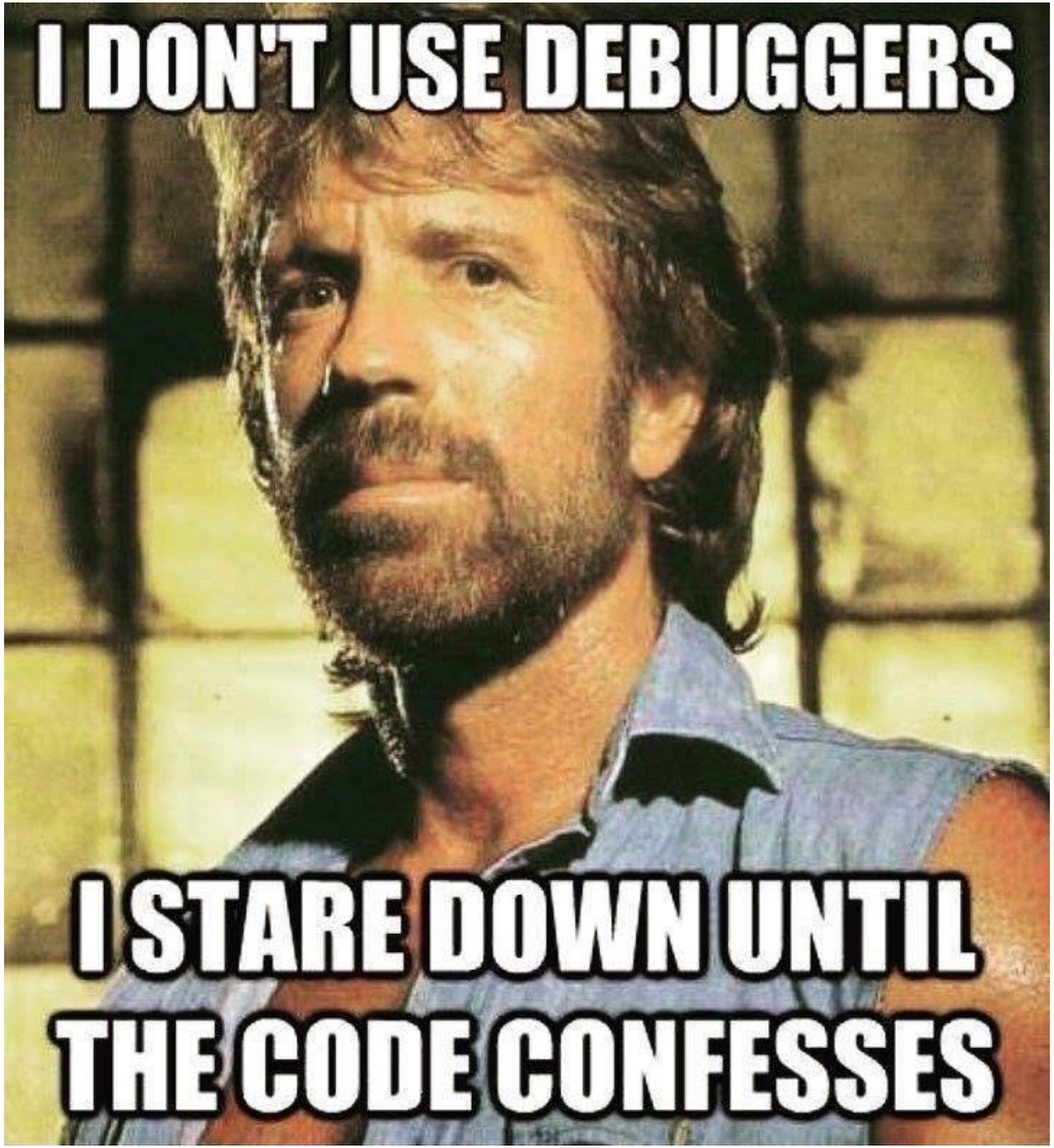
- Gute IDE hat integrierten **Debugger**
- Tool, um Bugs schnell(er) zu identifizieren
- Laufzeitfehler & Logikfehler identifizieren

Tipp 6



Debugger

- Tool in IDE, um Laufzeit- & Logikfehler zu identifizieren
- Kann damit Code Zeile für Zeile durchgehen, …
- … und beobachten, wie sich Variablen ändern
- Studiere Video-Tutorial dazu (Link auf Wiki)


I DON'T USE DEBUGGERS
I STARE DOWN UNTIL
THE CODE CONFESSES