Scope

>> Der Geltungsbereich (**Scope**) ist eine Eigenschaft einer Variable, die definiert, von welchen Funktionen aus auf die Variable zugegriffen werden kann.

Lokale Variablen: Nur innerhalb der Funktion, in der sie erstellt wurden, zugänglich.

Globale Variablen: Von jeder Funktion im Programm aus zugänglich.

Lokale Variablen

```
int triple(int x); // Prototyp

int main(void)
{
   int result = triple(5);
}

int triple(int x)
{
   return x * 3;
}
```

x ist lokale Variable in triple();
keine andere Funktion kann auf diese Variable zugreifen.

result ist lokale Variable in main().

Globale Variablen

```c
#include <stdio.h>

float global = 0.5050;

void triple(void)
{
   global *= 3;
}

int main(void)
{
   triple();
   printf("%f\n", global);
}
```

Globale Variablen werden außerhalb aller Funktionen deklariert.

Jede Funktion kann auf globale Variablen zugreifen.

Übergabe „als Wert“

Lokale Variablen werden in C meist **„als Wert“** übergeben. (Zumindest die, die wir bisher kennengelernt haben.)

Dabei erhält die aufgerufene Funktion (**„callee“**) eine **Kopie** der übergebenen Variable.

Die Variable im aufrufenden Code (**„caller“**) bleibt unverändert (es sei denn, sie wird mit dem Rückgabewert der Funktion überschrieben).

Übergabe als Wert

```
int triple(int x)
{
   return x *= 3;
}

int main(void)
{
   int foo = 4;
   triple(foo);
}
```

Kein Effekt auf foo in main().

x in triple() ist eine Kopie von foo.

Übergabe als Wert

```
int triple(int x)
{
   return x *= 3;
}

int main(void)
{
   int foo = 4;
   foo = triple(foo);
}
```

foo in main() wird überschrieben.

Der Rückgabewert von triple()
wird der Variable foo zugewiesen.

Gleiche Namen

Jede Funktion hat ihren eigenen Geltungsbereich für lokale Variablen.

In verschiedenen Funktionen können Variablen mit den gleichen Namen verwendet werden.

Dies kann zu Verwirrung führen.

Gleiche Namen

```
int increment(int x)
{
   x++;
   return x;
}

int main(void)
{
   int x = 1;
   int y;
   y = increment(x);
   printf("x is %i, y is %i\n", x, y);
}
```

x in main() und x in increment() sind unterschiedliche Variablen.

y erhält den Rückgabewert von increment().

Gleiche Namen

```
int increment(int xi)
{
   xi++;
   return xi;
}

int main(void)
{
   int xm = 1;
   int ym;
   ym = increment(xm);
   printf("x is %i, y is %i\n", xm, ym);
}
```

x_m in main() und x_i in increment() sind unterschiedliche Variablen.

y_m erhält den Rückgabewert von increment().

Gleiche Namen

```
int increment(int xi)
{
   xi++;
   return xi;
}

int main(void)
{
   int xm = 1;
   int ym;
   ym = increment(xm);
   printf("x is %i, y is %i\n", xm, ym);
}
```

Die Ausgabe ist daher: x is 1, y is 2

F1: Kann es innerhalb eines Programms mehrere Variablen mit dem gleichen Namen geben?

F2: Kann es innerhalb eines Programms mehrere globale Variablen mit dem gleichen Namen geben?

F3: Kann es lokale und globale Variablen mit dem gleichen Namen geben?

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

F4: Ist der Geltungsbereich einer Variablen immer die gesamte Funktion oder beginnt er erst ab dem Punkt, an dem die Variable in der Funktion deklariert wird?

```c
void example_function()
{
    int x = 5;  // x in ganzer Funktions gültig

    for (int i = 0; i < 3; i++)
    {
        // i ist nur in dieser Schleife gültig.
        printf("Erste Schleife: i = %d, x = %d\n", i, x);
    }
    // Hier ist i nicht mehr gültig.

    for (int i = 0; i < 3; i++)
    {
        // Neue Variable i, nur in dieser Schleife gültig.
        printf("Zweite Schleife: i = %d, x = %d\n", i, x);
    }
    // Hier ist i wieder nicht mehr gültig

    int i = 10;  // Neue Variable i, gültig bis Funktionsende
    printf("Außerhalb: i = %d, x = %d\n", i, x);
}
```

```c
#include <stdio.h>

int g = 5;

void changeGlobalOne(int l)
{
   l = 10;
}

void changeGlobalTwo(int g)
{
   g = 10;
}

void changeGlobalThree()
{
   g = 10;
}
```

```c
int main()
{
   printf("%i\n", g);
   changeGlobalOne(g);
   printf("%i\n", g);
   changeGlobalTwo(g);
   printf("%i\n", g);
   changeGlobalThree();
   printf("%i\n", g);
   return 0;
}
```

Lokale Variablen: nur innerhalb der Funktion zugänglich, in der sie deklariert wurden

Der Geltungsbereich einer Variablen reicht vom Punkt ihrer Deklaration bis zum Ende des Blocks {...}, in dem sie deklariert wurde

Globale Variablen: werden außerhalb aller Funktionen deklariert und sind im gesamten Programm zugänglich

Gleiche Variablennamen können in verschiedenen Funktionen verwendet werden – es handelt sich dann um unterschiedliche Variablen

Bei der Übergabe als Wert erhält eine Funktion nur eine Kopie der Variable – die Original-Variable bleibt unverändert.

Eine lokale Variable mit gleichem Namen verdeckt eine globale Variable innerhalb ihres Geltungsbereichs.