Schleifen

>> **Schleifen** (engl**.** ***loops***) ermöglichen es Programmen, Codezeilen wiederholt auszuführen. Sie ersparen uns das Kopieren und Einfügen oder das Wiederholen von Code.

>> C bietet einige verschiedene Möglichkeiten, Schleifen zu implementieren, von denen Ihnen einige wahrscheinlich aus Scratch bekannt vorkommen.

while-Schleifen

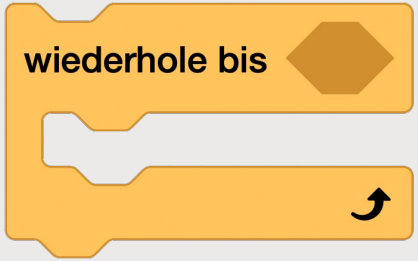
```
while (true)
{
    // einige Anweisungen
}
```

Dies ist eine **Endlosschleife**.

Die Codezeilen zwischen den geschweiften Klammern werden wiederholt von oben nach unten ausgeführt, bis wir aus der Schleife ausbrechen (mit der *break*-Anweisung, vgl. *switch*) oder unser Programm anderweitig beenden (`Strg-C`).

while-Schleifen

```
while (boolean-expr)
{
    // einige Anweisungen
}
```



Allgemeine Form der while-Schleife; wird ausgeführt, solange *boolean-expr* wahr ist. Code in geschweiften Klammern wird wiederholt bis der Ausdruck unwahr wird.

Ist *boolean-expr* schon zu Beginn unwahr, wird der Code übersprungen.

Ähnlich wie „wiederhole bis“ in Scratch, aber mit umgekehrter Logik.

Nützlich, wenn man nicht weiß, wie oft die Schleife ausgeführt werden soll.

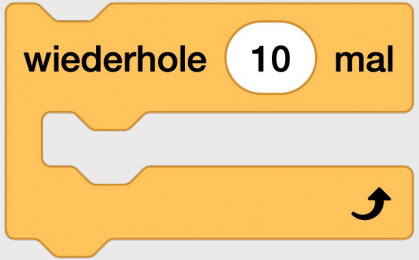
do-while-Schleifen

```
do
{
    // einige Anweisungen
}
while (boolean-expr);
```

Diese Schleife führt alle Codezeilen zwischen den geschweiften Klammern einmal aus und kehrt dann, wenn *boolean-expr* wahr ist, zurück und wiederholt diesen Vorgang, bis der Ausdruck *false* wird.

for-Schleifen

```
for (int i = 0; i < 10; i++)
{
    // einige Anweisungen
}
```



for-Schleifen führen eine feste Anzahl von Wiederholungen aus (hier: 10).

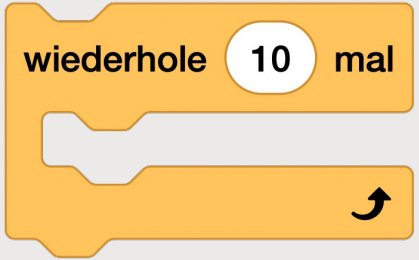
Zuerst wird/werden (eine) Zählvariable(n) (hier: `i`) gesetzt.

Dann wird der boolesche Ausdruck überprüft. Ist er *true*, wird der Code in der Schleife ausgeführt, bei *false* nicht.

Die Zählvariable wird um 1 erhöht und der boolesche Ausdruck wird erneut überprüft usw.

for-Schleifen

```
for (start; expr; increment)
{
    anweisungen
}
```



for-Schleifen führen eine feste Anzahl von Wiederholungen aus.

Zuerst werden die Anweisungen in *start* ausgeführt (ggf. mit Komma getrennt).

Dann wird der boolesche Ausdruck *expr* überprüft. Ist er *true*, wird der Code in der Schleife ausgeführt, bei *false* nicht.

Der *anweisungen* werden ausgeführt. Dann wird *increment* ausgeführt und *expr* wieder überprüft usw.

while Unbekannte Anzahl von Wiederholungen, möglicherweise keine

Beispiel: Spiel-Kontrollfluss

Do-while Unbekannte Anzahl an Wiederholungen, aber mindestens einmal

Beispiel: Validierung von Benutzereingaben

for Bekannte Anzahl von Wiederholungen

Beispiel: Iteration über Array-Elemente

break und continue

```c
#include <stdio.h>
#include <cs50.h>

int main(void)
{
    int number;

    while (true)
    {
        number = get_int("Zahl: ");
        if (number < 0)
        {
            continue; // Skip negative numbers
        }
        if (number == 0)
        {
            break; // Exit loop when user enters zero
        }
        printf("%d\n", number);
    }

    return 0;
}
```

ohne break
und continue

```c
#include <stdio.h>
#include <cs50.h>

int main(void)
{
    bool running = true;
    int number;

    while (running)
    {
        number = get_int("Zahl: ");

        if (number < 0)
        {
            // Skip negative numbers by doing nothing
        }
        else if (number == 0)
        {
            running = false; // Exit loop (user entered zero)
        }
        else
        {
            printf("%d\n", number);
        }
    }

    return 0;
}
```

Verschachtelte Schleifen

```c
#include <stdio.h>

int main()
{

    for (char letter = 'A'; letter <= 'B'; letter++)
    {
        printf("Paare mit %c\n", letter);

        for (int number = 0; number < 6; number++)
        {
            if (number % 2)
            {
                continue; // skip iteration in inner loop
            }
            printf("%c%d\n", letter, number);
        }

        printf("\n");
    }

    return 0;
}
```

Off-by-One-Fehler

```c
#include <stdio.h>

int main()
{
    int totalItems = 5, count = 0;

    for (int i = 0; i <= totalItems; i++) // Off-by-one error
    {
        printf("Processing item %d.\n", i);
        count++;
    }
    printf("Processed %d items\n", count); // output: 6
    return 0;
}
```

Off-by-One-Fehler: Anzahl der Ausführungen ist um 1 zu viel oder zu wenig; häufig, wenn von Null aus gezählt wird.

Versehentliche Endlosschleife

```c
#include <stdio.h>

int main() {

    int target = 5, current = 0;
    printf("Counting to %d:\n", target);

    while (current < target)
    {
        printf("%d ", current);
    }

    printf("\nFinished!");
    return 0;
}
```

Wird in while-Schleifen vergessen, die Schleifenvariable zu aktualisieren, kommt es zu Endlosschleifen. Immer sicherstellen, dass Bedingung irgendwann *false* wird!

Schleifen-Invarianten

```c
#include <stdio.h>

int main() {
    double balance = 1000.0;  // Initial investment
    double target = 1500.0;   // Target balance
    double rate = 0.08;       // 8% annual interest rate
    double fee = 10.0;        // Yearly account fee
    int years = 0;

    printf("Year 0: Balance = %.2f\n", balance);

    while (balance < target) {
        years++;

        // Deduct yearly fee
        balance -= fee;
        printf("Year %d (after fee): Balance = %.2f\n", years, balance);

        // Add interest
        balance += balance * rate;
        printf("Year %d (after interest): Balance = %.2f\n", years, balance);
    }

    printf("Target reached after %d years. Balance: %.2f\n", years, balance);
    return 0;
}
```

Guter Stil

Für einfache Iterationen Iterationsvariablen i, j, k nennen, in komplexen Iterationen beschreibende Namen verwenden.

Verschachtelung auf 3 Ebenen beschränken; werden mehr benötigt: Code-Entwurf noch einmal überdenken.

Konsistente Einrückung innerhalb von Schleifenkörpern.

Geschweifte Klammern auch bei einzeiligem Schleifenkörper.

Schleifenbedingungen einfach und lesbar halten.

Komplexe Schleifenkörper in eigene Funktionen extrahieren.

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

Was mache ich, wenn ich will, dass das „continue“ für die äußere Schleife gilt?

```c
#include <stdio.h>

int main()
{
    for (char letter = 'A'; letter <= 'D'; letter++)
    {
        bool skip_outer = 0;
        for (int number = 0; number < 3; number++)
        {
            if (number + letter >= 'E') // second letter only up to E
            {
                skip_outer = 1;
                break;  // Exit inner loop
            }
            printf("%c %c\n", letter, letter + number);
        }

        if (skip_outer)
        {
            printf("\n\n");
            continue;  // next iteration in outer loop
        }

        printf("***\n\n");
    }
    return 0;
}
```

```
A A
A B
A C
***

B B
B C
B D
***

C C
C D

D D
```

```c
#include <stdio.h>

int main() {
    int count = 0;
    for (char first = 'A'; first <= 'Z'; first++)
    {
        for (char second = 'A'; second <= 'Z'; second++)
        {
            for (char third = 'A'; third <= 'Z'; third++)
            {
                if (first == third)
                {
                    printf("%c%c%c\n", first, second, third);
                    count++;
                    if (count >= 30)
                    {
                        break;
                    }
                }
            }
        }
    }

    printf("Total palindromes printed: %d\n", count);
    return 0;
}
```

Dieses Programm gibt dreibuchstabige Palindrome aus, also Buchstabenfolgen, die von vorne und von hinten gelesen gleich sind, zum Beispiel A A A oder A B A oder M E M.

1. Wie viele dreistellige Palindrome gibt es eigentlich höchstens?

2. Offenbar will die Person, die das Programm geschrieben hat, offenbar nur die ersten 30 Palindrome ausgeben. Komischerweise gibt das Programm aber alle Palindrome aus, die es gibt. Warum? Und was ist zu tun, damit das Programm richtig arbeitet?

```c
for (int user_id = 10; user_id <= 99; user_id++)
{
    int attempts;
    for (attempts = 0; attempts < 10; attempts++) {
        // Compute a hopefully unique PIN for the user
        int pin = 1;
        for (int temp = user_id * 10 + attempts; temp > 0; temp--) {
            pin += temp * pin;
            while (pin % 10 == 0 || pin > 1000000)
                pin /= 10;
        }

        // Do not assign PINs that contain an 8 or a 9
        int has_bad_digit = 0, temp_pin = pin;
        while (temp_pin > 0) {
            int digit = temp_pin % 10;
            if (digit == 8 || digit == 9) {
                has_bad_digit = 1;
                break;
            }
            temp_pin /= 10;
        }
        if (has_bad_digit) continue; // Skip to the next attempt

        pin = (pin < 100000) ? pin + 100000 : pin; // Ensure PIN has 6 digits

        printf("User: %d, PIN: %d\n", user_id, pin);
        break; // Move to the next user_id
    }

    if (attempts >= 10)
        printf("Could not generate PIN for User %d.\n", user_id);
}
```

Das Programm soll für eine Reihe von Personen mit den User IDs 10–99 unterschiedliche sechsstellige PINs erzeugen. Die PINs sollen nicht die Ziffern 8 oder 9 enthalten, weil sie sich beim Ablesen leicht verwechseln lassen.

1. Tut das Programm, was es soll? Also erzeugt es tatsächlich sechsstellige PINs, die den Anforderungen genügen? Gibt es einen Fehler aus, wenn es keine PIN erzeugen kann?

2. Was passiert, wenn man in der drittletzten Zeile attempts >= 10 durch attempts >= 15 ersetzt, etwa weil es dem Programm bei vielen User IDs nicht gelingt, mit 10 Versuchen eine PIN zu finden? Wieso sind es ursprünglich eigentlich max. 10 Versuche?

3. Erzeugt das ursprüngliche Programm mit max. 10 Versuchen pro User ID tatsächlich unterschiedliche PINs bei allen Nutzern? Überlegen Sie sich, wie Sie das überprüfen können, ohne von Hand alle erzeugten PINs durchgehen zu müssen.

Drei Arten von Schleifen:
while, do-while und for

Kontrolle mittels
break und continue

Off-by-one-Fehler in
Schleifenbedingungen

Endlosschleifen und
verschachtelte Schleifen

Vergessen der Aktualisierung
von Schleifenvariablen

Schleifeninvarianten

Stil-Richtlinien und
Best Practices beachten