Calliope zu C/C++

Grundstruktur eines Arduino-Programms

```
void setup() {
  // Wird einmal beim Start ausgeführt
  // Hier initialisieren wir Pins, Serial, etc.
}

void loop() {
  // Wird endlos wiederholt
  // Hier kommt der Hauptcode hin
}
```

Variablen und Datentypen

Calliope	**C/C++**	**Erklärung**
Zahl	`int zahl = 42;`	Ganze Zahlen, typischerweise -32768 bis 32767
Kommazahl	`float komma = 3.14;`	Zahlen mit Nachkommastellen
Wahr/Falsch	`bool wahr = true;`	Logische Werte (true oder false)
Text	`String text = "Hallo";`	Zeichenketten

Zusätzliche wichtige Datentypen:

- `long`: Für größere ganze Zahlen
- `double`: Für präzisere Kommazahlen
- `char`: Für einzelne Zeichen

Ein- und Ausgabe

Calliope	**C/C++ (Arduino)**	**Erklärung**
Knopf A gedrückt	`digitalRead(BUTTON_A) == LOW`	Prüft, ob ein Knopf gedrückt ist
LED einschalten	`digitalWrite(LED_BUILTIN, LOW);`	Schaltet die eingebaute LED ein (invertiert)
LED ausschalten	`digitalWrite(LED_BUILTIN, HIGH);`	Schaltet die eingebaute LED aus (invertiert)
Zeige Zahl	`Serial.println(zahl);`	Gibt eine Zahl über Serial aus
Zeige Text	`Serial.println("Text");`	Gibt Text über Serial aus
Pin 0 als Eingang	`pinMode(0, INPUT);`	Konfiguriert Pin 0 als Eingang
Pin 1 als Ausgang	`pinMode(1, OUTPUT);`	Konfiguriert Pin 1 als Ausgang
Lies analogen Wert an A0	`int wert = analogRead(A0);`	Liest einen analogen Wert (0-1023)

Wichtig für Serial:

```
void setup() {
  Serial.begin(9600);  // Startet die serielle Kommunikation
}
```

Kontrollstrukturen

Wenn-Dann (If-Else)

Calliope

C/C++

```
if (Bedingung) {
  // Anweisungen wenn Bedingung wahr
} else {
  // Anweisungen wenn Bedingung falsch
}
```

Beispiel

```
if (temperatur > 25) {
  digitalWrite(VENTILATOR_PIN, HIGH);
} else {
  digitalWrite(VENTILATOR_PIN, LOW);
}
```

Schleifen

Calliope “x-Mal wiederholen”

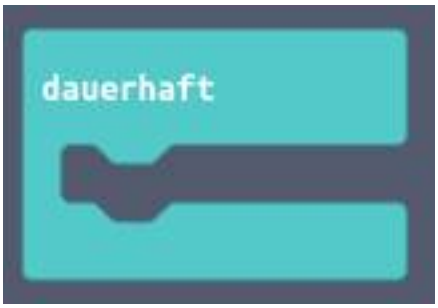
C/C++ “x-Mal wiederholen”

```
for (int i = 0; i < 5; i++) {
  // Anweisungen
}
```

Beispiel

```
for (int i = 0; i < 5; i++) {
  digitalWrite(LED_PIN, HIGH);
  delay(100);
  digitalWrite(LED_PIN, LOW);
  delay(100);
}
```

Calliope “dauerhaft”



C/C++ “dauerhaft”

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

Beispiel

```
while (digitalRead(BUTTON_PIN) == HIGH) {
```

```
  // Führe Aktionen aus, solange der Knopf gedrückt ist
}
```

Funktionen

Calliope

C/C++

```
void meineFunktion() {
  // Anweisungen
}
```

Beispiel

Funktion mit Rückgabewert und Parametern:

```
int addiere(int a, int b) {
  return a + b;
}
```

Operatoren

Operation	C/C++	Beispiel
Gleich	==	`if (a == b) { ... }`
Ungleich	!=	`if (a != b) { ... }`
Größer	>	`if (a > b) { ... }`
Größer gleich	>=	`if (a >= b) { ... }`
Kleiner	<	`if (a < b) { ... }`
Kleiner gleich	<=	`if (a <= b) { ... }`
Und	&&	`if (a && b) { ... }`
Oder	||	`if (a || b) { ... }`
Nicht	!	`if (!a) { ... }`

Wichtige Arduino-Funktionen

- `delay(ms)`: Pausiert das Programm für die angegebene Zeit in Millisekunden
  ```
  delay(1000);  // Wartet 1 Sekunde
  ```

- `random(max)` oder `random(min, max)`: Erzeugt eine Zufallszahl

```
int zufallszahl = random(1, 101);  // Zufallszahl zwischen 1 und 100
```

Arrays und Strings

Arrays

```
int zahlen[5] = {1, 2, 3, 4, 5};
int ersteZahl = zahlen[0];  // Zugriff auf das erste Element (Index 0)
```

Strings

```
String name = "Alice";
int laenge = name.length();  // Länge des Strings
```

Tipps für C/C++ Programmierung

1. Jede Anweisung endet mit einem Semikolon `;`
2. Blöcke werden mit geschweiften Klammern `{ }` umschlossen
3. Kommentare:
   - Einzeilig: `// Kommentar`
   - Mehrzeilig: `/* Kommentar */`
4. Variablen müssen vor der Verwendung deklariert werden
5. Groß- und Kleinschreibung ist wichtig (case-sensitive)
6. Verwende aussagekräftige Variablen- und Funktionsnamen
7. Einrückung verbessert die Lesbarkeit (wird vom Compiler ignoriert)
8. Nutze Konstanten für unveränderliche Werte:

```
const int LED_PIN = 13;
```

9. Debuggen mit Serial.print():

```
Serial.print("Debug: ");
Serial.println(variable);
```

Beispielprogramm: Bodenfeuchtesensor mit LED#

```
// als erstes definieren wir die Pins der LED, des Sensors sowie den
Schwellwert für den Vergleich
const int SENSOR_PIN = A0;
const int LED_PIN = 13;
const int SCHWELLWERT = 500;

// in der setup()-Funktion wird die serielle Schnittstelle aktiviert und der
LED-Pin als Ausgang definiert
void setup() {
```

```
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
}

// in der loop() Funktion wird..
void loop() {
  // .. die Feuchtigkeit mit Hilfe der Funktion leseFeuchtigkeit()
ausgelesen:
  int feuchtigkeit = leseFeuchtigkeit();
  // .. mit Hilfe der Funktion zeigeFeuchtigkeitAn() angezeigt:
  zeigeFeuchtigkeitAn(feuchtigkeit);
  // .. 1000ms gewartet
  delay(1000);
}

// die Funktion leseFeuchtigkeit() gibt das ausgelesene Signal des
Bodenfeuchtesensors zurück
int leseFeuchtigkeit() {
  return analogRead(SENSOR_PIN);
}

// die Funktion zeigeFeuchtigkeitAn bekommt den Bodenfeuchtemessert und ..
void zeigeFeuchtigkeitAn(int feuchtigkeit) {
    // .. gibt ihn auf der seriellen Schnittstelle aus
  Serial.print("Feuchtigkeit: ");
  Serial.println(feuchtigkeit);

    // .. vergleicht ihn mit dem Schwellwert:
  if (feuchtigkeit > SCHWELLWERT) {
    digitalWrite(LED_PIN, HIGH); // LED aus
    Serial.println("Pflanze benötigt kein Wasser");
  } else {
    digitalWrite(LED_PIN, LOW); // LED an
    Serial.println("Pflanze benötigt Wasser");
  }
}
```