Handout – Schleifen und Debuggen

Schleifen

While-Schleife

Führt den Schleifenrumpf wiederholt aus, solange die Schleifenbedingung gültig bleibt.

```
int i = 0;
while (i < 10) {
    // gibt die Nachricht 10 mal aus
    cout << "Hallo von Schleife Nummer " << i << '\n';
    i++;
}
```

Manchmal wollen wir die Schleife vorzeitig verlassen, bevor die Schleifenbedingung false wird. In diesem Fall benützen wir das Schlüsselwort **break**. Damit springen wir unverzüglich zur Linie gleich nach der Schleife.

```
int i = 0;
while (i < 10) {
    // gibt die Nachricht nur 3 mal aus
    cout << "Hallo von Schleife Nummer " << i << '\n';
    if (i == 2)
        break;
    i++;
}
```

Falls wir den Rest vom Schleifenrumpf auslassen wollen, aber die Schleife nicht verlassen wollen benützen wir das Schlüsselwort **continue**.

```
int i = 0;
while (i < 10) {
    if (i % 2 == 1)
        continue;
    // gibt die Nachricht nur aus wenn i gerade ist
    cout << "hello from loop number " << i << '\n';
    i++;
}
```

For-Schleife

Dieses Muster von Initialisierung einer Variablen, Variable in Schleifenbedingung benützen und die Variable am Ende der Schleife verändern tritt sehr häufig auf. C++ erlaubt es uns das etwas kürzer zu schreiben mit einer For-Schleife.

```
for (int i = 0; i < 10; i++) {
    // gibt die Nachricht 10 mal aus
    cout << "hello from loop number " << i << '\n';
}
```

Wir müssen nicht alle 3 Teile der For-Schleife verwenden, wir können auch einige leer lassen. Obwohl, wenn wenn wir 2 oder mehr frei lassen dan können wir auch gleich eine While-Schleife verwenden.

```
int i = 0;
for (; i < 10;) { // Macht das nicht!
    // gibt die Nachricht 10 mal aus
    cout << "hello from loop number " << i << '\n';
    i++;
}
```

For-ever-Schleife:

```
int i = 0;
for (;;) { // läuft für immer ...
    i = next(i);
    if (my_special_condition(i)) {
        break; // ... oder zumindest bis hier
    }
    cout << "current i: " << i << '\n';
}
```

Beispiel: Primzahltest

Aufgabe: Finde heraus ob eine gegebene Zahl n eine Primzahl ist oder nicht.

```
bool prime = true;
for (int i = 2; i < n; i++) {
    if (n % i == 0) {
        prime = false;
        break;
    }
}
```

Debugging

Was machen wenn das Programm nicht funktioniert?

- Compiliere dein Programm und lass den Compiler überprüfen das es keinen Syntax oder Typenfehler gibt.
- Wenn dein Programm abstürzt überprüfe deine Annahmen mit `assert(conditions)`
- Benutze `cerr << var` um Informationen auszugeben die dir helfen könnte das Problem zu finden.
- Benutze einen Debugger