Exercise Week 03

GianAndrea Müller mailto:muellegi@student.ethz

March 12, 2018

Time Schedule

- 5' Nachbesprechung
- 15' while, do while mit Übung
- 3' break und continue
- 7' Gültigkeitsbereich
- 5' float und double mit Übung
- 10' Typenumwandlungen mit Übung
- 20' Fehlersuche mit assert

Learning Objectives

- Kenntnis aller Schleifen und erweiterte Flusskontrolle
- Verständnis aller grundlegenden Variablentypen und deren Umwandlung
- Nutzung erweiterter Debugging-Methoden

Nachbesprechung

```
8>4>2>1
   true > 2 > 1
  1>2>1
   false>1
   0>1
   false
7
   2<a<4
               2 < a < 4
   true <4
             false<4
                0<4
   1 < 4
10
   true
                true
11
12
   2<a && a<4
13
14
   //Comment your code
15
```

Schleifen

```
while(condition)
statement

do{
statement1
statement2
}
while(condition);
```

Exercise $03_{-}1 \sim 5$ ' All positive numbers up to n

```
//loop 1
  for(int i = 1; i <= n; ++i)
    cout << i << "\n";
  //loop 2
  int i = 0;
  while(i < n)
    cout << ++i << "\n";
  //loop 3
10
  i = 1:
11
12
    cout <<i ++<< "\n";
13
  while (i \le n)
14
```

```
//if n == INT_MAX -> infinite loop
  for(int i = 1; i <= n; ++i)
    cout << i << "\n":
4
  //loop 2
  int i = 0;
  while(i < n)
    cout << ++i << "\n":
8
  //n<= 0 -> still outputs 1, if n=INT_MAX
10
     -> infinite loop
  i = 1;
11
12
    cout <<i ++<< "\n";
13
  while (i \le n)
14
```

Welche Schleife in welchen Fall?

Motivation

- So wenig code wie möglich.
- Einfach lesbarer code.

Möglichkeiten

 for: Es wird ein Zähler benötigt, Zähler wird nach der Schleife nicht mehr benötigt.

Wiederhole ein statement n mal.

 while: Die Bedingung hängt von einer Variable ab, die bereits vor der Schleife existiert.

Dekrementiere x bis es ein Vielfaches von 5 ist.

• do: Die Bedingung hängt von einer Variable ab, die erst in der Schleife erhalten wird.

Führe cin >> x aus bis x > 3

break und continue

```
while(true){
     statement1;
     if (condition) {
       break;
6
7
  for (int i = 0; i < 10; i++){
     statement1;
     if (condition) {
10
       continue;
11
     }
12
     statement2;
13
14
```

```
if (x < 7) {
int a=8;
}

std::cout << a; // Compiler error, a does
    not exist.</pre>
```

```
int a = 2;
if (x < 7) {
  a=8;
}
std::cout << a; // Outputs 2 or 8,
  depending on the if-statement.</pre>
```

```
for (int i = 0; i < 5; ++i) {
  std::cout << i << "\n"; // Outputs i
}

std::cout << i << "\n"; // Compiler error,
  i does not exist</pre>
```

```
unsigned int x = 2;
int i = 5;
if (x > 1) {
  int i = 3;
  std::cout << i; // outputs 3
}
std::cout << i; // outputs 5</pre>
```

float und double

```
float a = 1.0/6.0;
double b = 2/5;

cout << a; //outputs 0.166666672

float c = (a - 0.1)*10;

cout << c; //outputs 0.666666746</pre>
```

Datentypen

Typenumwandlung

char, bool < int < unsigned int < float < double

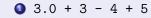
Typenumwandlung

```
#include <iostream>
  int main()
      int a = -4;
      unsigned int b = 2u;
6
7
      std::cout << a + b; //output:
8
         4294967294
      std::cout << int(a+b); //output: -2</pre>
9
10
      unsigned int c = a;
11
12
      return 0;
13
14
```

Exercise 03 2 \sim 5'

Evaluate by hand. Assume x = 1 is of type *int*

- \bullet 3.0 + 3 4 + 5
- 2 5 % 4 * 3.0 + true * x++
- 3 3 4u + 8.0



- 0 3.0 + 3 4 + 5
- ((3.0 + 3) 4) + 5

$$0 3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

- 0 3.0 + 3 4 + 5
- ((3.0 + 3) 4) + 5
- ((3.0 + 3.0) 4) + 5
- (6.0 4) + 5

$$0 3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

- 0 3.0 + 3 4 + 5
- **2** ((3.0 + 3) 4) + 5
- ((3.0 + 3.0) 4) + 5
- (6.0 4) + 5
- (6.0 4.0) + 5
- 02.0 + 5

$$0 3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

$$0 2.0 + 5$$

$$02.0 + 5.0$$

Solution (1)

$$\bullet$$
 3.0 + 3 - 4 + 5

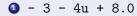
$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

$$02.0 + 5.0$$

3 7.0



Solution (2)

2 (4294967283u - 4u) + 8.0

- 0 3 4u + 8.0
- 2 (4294967283u 4u) + 8.0
- **4294967289u** + 8.0

- **1** 3 4u + 8.0
- 2 (4294967283u 4u) + 8.0
- **3** 4294967289u + 8.0
- 4294967289.0 + 8.0

- **1** 3 4u + 8.0
- ② (4294967283u 4u) + 8.0
- **4294967289u + 8.0**
- **4294967289.0 + 8.0**

Solution (3)

1 5 % 4 * 3.0 + true * x++

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- **③** (1 * 3.0) + (true * (x++))

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- **④** (1.0 * 3.0) + (true * (x++))

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- **④** (1.0 * 3.0) + (true * (x++))
- **3** 3.0 + (true * (x++))

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- **●** (1.0 * 3.0) + (true * (x++))
- 3.0 + (true * (x++))
- **o** 3.0 + (true * 1)

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- **●** (1.0 * 3.0) + (true * (x++))
- 0 3.0 + (true * (x++))
- **o** 3.0 + (true * 1)
- **0** 3.0 + (1 * 1)

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- **●** (1.0 * 3.0) + (true * (x++))
- **3** 3.0 + (true * (x++))
- **o** 3.0 + (true * 1)
- **0** 3.0 + (1 * 1)
- 0 3.0 + 1

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- \bullet (1.0 * 3.0) + (true * (x++))
- 0 3.0 + (true * (x++))
- **o** 3.0 + (true * 1)
- **0** 3.0 + (1 * 1)
- **3.0 + 1**
- 9 3.0 + 1.0

- **1** 5 % 4 * 3.0 + true * x++
- ② ((5 % 4) * 3.0) + (true * (x++))
- 3 (1 * 3.0) + (true * (x++))
- **●** (1.0 * 3.0) + (true * (x++))
- **3** 3.0 + (true * (x++))
- **o** 3.0 + (true * 1)
- 03.0 + (1 * 1)
- **3.0 + 1**
- 9 3.0 + 1.0
- **4.0**