Informatik I - Übung 2

Pascal Schärli

pascscha@student.ethz.ch

01.09.2019

ETH zürich

Was gibts heute?

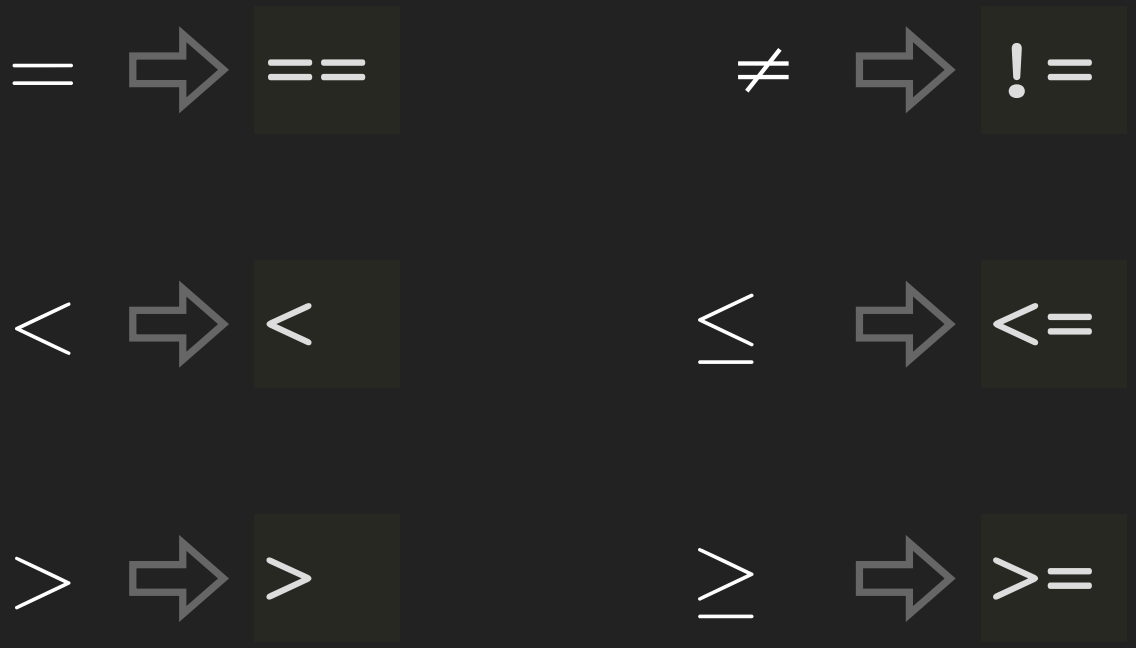
- Best-Of Vorlesung
  - Vergleiche
  - Präzedenz
  - Loops

- Vorbesprechung Serie 2

Best of Vorlesung

Vergleiche

In C++ können Variablen wie in der Mathematik miteinander verglichen werden.



If-Statement

- Falls die Condition in den Runden () Klammern erfüllt ist, wird der Code in den geschweiften {} Klammern ausgeführt.

- Jedes If-Statement besteht aus:
  1. Genau einem
  2. Beliebig vielen
  3. Höchstens ein

`if`

`else if`

`else`

```
int n = 50;

if(n > 128){
    std::cout << "n is larger than 128";
}
else if(n > 64){
    std::cout << "n is larger than 64";
}
else if(n > 32){
    std::cout << "n is larger than 32";
}
else if(n > 16){
    std::cout << "n is larger than 16";
}
else {
    std::cout << "n is smaller than 17";
}
```

```
n is larger than 32
```

Scopes

```
int a = 0;

if(a == 0){
    int b = 3;
}
else{
    int b = 4;
}

std::cout << b << std::endl;
```

error: ‘b’ was not declared in this scope

- Variablen sind immer nur in einem bestimmten Bereich (=scope) gültig
- Wenn dieser Bereich verlassen wird, wird die Variable wieder gelöscht

Boolean Expressions

AND `&&`

a	b	a && b
false	false	false
false	true	false
true	false	false
true	true	true

OR `||`

a	b	a || b
false	false	false
false	true	true
true	false	true
true	true	true

XOR `!=`

a	b	a != b
false	false	false
false	true	true
true	false	true
true	true	false

NOT `!`

a	!a
false	true
true	false

Kurzschluss Evaluation

```
2 > 3 && 17u - 55 <= ++x % y
```

```
(2 > 3) && (17u - 55 <= ++x % y)
```

```
false && (17u - 55 <= ++x % y)
```

```
false
```

int -> bool

```
if( 7 && 8){
    std::cout << "Hello" << std::endl;
}
else{
    std::cout << "World!" << std::endl;
}
```

```
Hello
```

Bei Logischen Operationen werden Integer zu Booleans umgewandelt

```
0             -> false
alles andere -> true
```

bool -> int

```
if( 3 > false){
    std::cout << "Hello" << std::endl;
}
else{
    std::cout << "World!" << std::endl;
}
```

```
Hello
```

Beim Vergleich mit Integern werden Booleans zuerst in Integer umgewandelt

```
false -> 0
true  -> 1
```

Präzedenz

Präzedenz	Operator	Assoziativität
2	`a++` `a--`	
3	`++a` `--a` `!`	
5	`a*b` `a/b` `a%b`	
6	`a+b` `a-b`	
9	`<` `<=` `>` `>=`	
10	`==` `!=`	
14	`&&`	
15	`||`	
16	`=` `+=`	

Komplette Tabelle: https://en.cppreference.com/w/cpp/language/operator_precedence

Präzedenz

```
int x = 1;
```

```
 x == 1  ||   1 / (x - 1)  < 1
```

```
 x == 1  ||  (1 / (x - 1)) < 1
```

```
 x == 1  || ((1 / (x - 1)) < 1)
```

```
(x == 1) || ((1 / (x - 1)) < 1)
```

```
(1 == 1) || ((1 / (x - 1)) < 1)
```

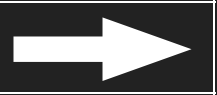
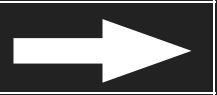
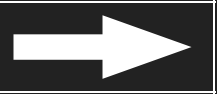
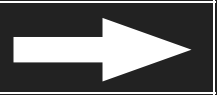
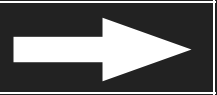
```
    true || ((1 / (x - 1)) < 1)
```

```
    true
```

P	Op	Assoz.
2	`a++` `a--`	→
3	`++a` `--a` `!`	←
5	`a*b` `a/b` `a%b`	→
6	`a+b` `a-b`	→
9	`<` `<=` `>` `>=`	→
10	`==` `!=`	→
14	`&&`	→
15	`||`	→
16	`=` `+=`	←

Präzedenz

```
int x = 1;

 !(1 && x)  + 1

(!(1 && x)) + 1

(!(1 && 1)) + 1

(!( true )) + 1

(  false  ) + 1

     0      + 1

            1
```

P	Op	Assoz.
2	`a++` `a--`	
3	`++a` `--a` `!`	
5	`a*b` `a/b` `a%b`	
6	`a+b` `a-b`	
9	`<` `<=` `>` `>=`	
10	`==` `!=`	
14	`&&`	
15	`||`	
16	`=` `+=`	

While-Loop

Solange die Condition in den Runden () Klammern erfüllt ist, wird der Code in den geschweiften {} Klammern wiederholt.

```
int i = 0;
while(i < 10){
    std::cout << i << " ";

    i++;
}
```

```
0 1 2 3 4 5 6 7 8 9
```

For-Loop

- While-Loops verfolgen häufig die selbe Struktur:

1. Initialization
2. Condition
3. Increment

```
int i = 0;
```

```
i < 10
```

```
i++
```

- Diese Struktur kann man daher auch schöner mit einem For-Loop schreiben.
- Der einzige unterschied ist, dass i nun im inneren Scope ist.

```
int i = 0;
while(i < 10){
    std::cout << i << " ";

    i++;
}
```

```
for(int i = 0; i < 10; i++){
    std::cout << i << " ";
}
```

```
0 1 2 3 4 5 6 7 8 9
```

Strange Sum

Schreibe ein Programm, welches die Summe aller positiven Zahlen, die ungerade aber nicht durch 5 teilbar sind, bis zur Obergrenze n zurückgibt.

```cpp
// Program: strangesum.cpp

#include <iostream>

int main () {

  // input
  unsigned int strangesum = 0;
  unsigned int n;
  std::cin >> n;

  // computation
  for (unsigned int i = 1; i <= n; i+=2) {
    if (i%5 != 0) {

        strangesum += i;

    }
  }

  // output
  std::cout << strangesum << "\n";

  return 0;
}
```

Largest Power

Schreibe ein Programm, welches für eine Zahl n die grösste Zweierpotenz p zurück gibt, welche kleiner als n ist.

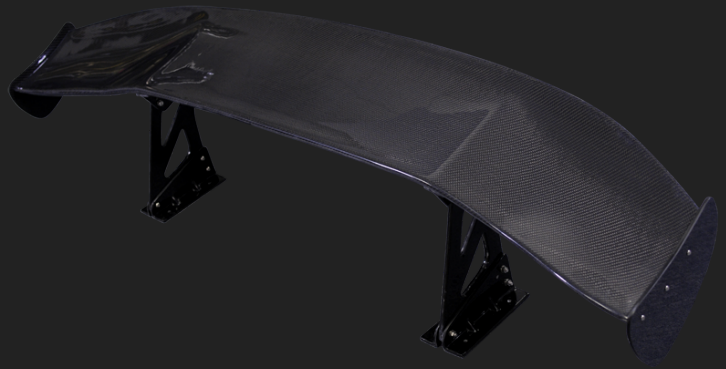
```cpp
#include <iostream>
#include <cassert>

int main () {
  unsigned int n;
  std::cin >> n;
  assert(n >= 1);

  unsigned int power = 1;
  for (; power <= n / 2; power *= 2);

  std::cout << power << std::endl;

  return 0;
}
```

Debugging

Output von std::cerr wird vom auto-grader ignoriert.

```cpp
#include <iostream>
#include <cassert>

int main () {
    const int n = 6;

    // Compute n^12
    unsigned int prod = 1;
    std::cerr << "Reached 1" << std::endl;
    for (int i = 1; 1 <= i && i < 13; ++i) {
        prod *= n;
    std::cerr << prod << std::endl;
    }
    std::cerr << "Reached 2" << std::endl;
    // Output stars
    for (int i = 1; i < prod; ++i) {
        std::cout << "*";
    }
    std::cerr << "Reached 3" << std::endl;
    std::cout << "\n";
    return 0;
}
```

Vorbesprechung



Pascal Schärli *01.01.2019*

ETH zürich

1 - Expression Evaluation

```
3 >= 3
 true || false  && false
(true || false) && false
3 > (1 < true)
8 > 4 > 2 > 1
2 < a < 4
```

Welche dieser Expressions sind true, welche sind false?

Tipps:

- Wenn ihr unsere Beispiele zur Präzedenz versteht, kann hier nicht mehr viel schiefgehen.

3 - Decimal to Binary

Schreibt ein Programm, welches die Binärrepräsentation einer positiven, ganzen Zahl bestimmt.

Tipps:

- Letzte Woche haben wir dazu einen Algorithmus angeschaut: Binäre Repräsentation

4a - Fibonacci Primes

Schreibt ein Programm, welches bis zu einer Zahl m alle Zahlen ausgibt, welche sowohl Fibonacci als auch Primzahlen sind.

Fibonacci Zahlen:

1 1 2 3 5 8 13 21

Primzahlen:

n ist eine Primzahl, falls für alle d zwischen 2 und n-1 gilt, dass n%d != 0

Tipps:

- Fibonacci: Benutzt schleife mit zwei Variablen, welche den letzten und vorletzten Wert speichern
- Prim: Implementiert eine Schleife, welche die Definition von oben prüft

4b - Fibonacci Overflow

Fibonnacci Zahlen wachsen schnell. Schreibe ein Programm, welches Fibonacci Zahlen berechnet aber aufhört bevor es einen Overflow gibt.

Tipps:

- Bei einem 32 Bit unsigned int passiert der overflow wenn die Zahl grösser als 2^32-1 wird

Pascal Schärli *01.01.2019*

Program of the Week

Viel Spass!