Informatik I - Übung 3

Pascal Schärli

pascscha@student.ethz.ch

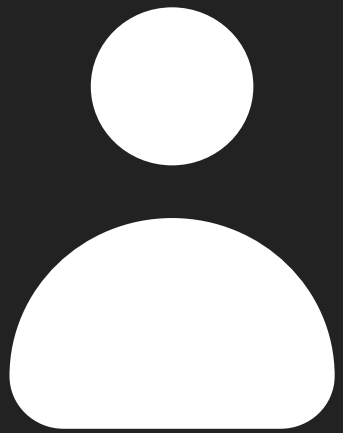
08.03.2019

ETH zürich

Was gibts heute?

- Self-Assesment
- Nachbesprechung Serie 1
- Best-Of Vorlesung
  - Fliesskomma Zahlen
  - Expressions #2
  - Scopes #2
  - Loops #2
- Vorbesprechung Serie 2

Self-Assesment



Pascal Schärli *08.03.2019*

ETH zürich

1 Werkzeuge

Wozu dient ein Compiler?

```
Übersetzen des Quelltexts in
ausführbaren Maschinen-Code
```

2 Anweisungen

Sind folgende C++ Anweisungen gültig?

```cpp
std::cout << "C++ is better than Java.";
```

```cpp
int a = 100
```

```cpp
if (1<2)
    std::cout << "I knew it!" << std::endl;
```

3 Ausdrücke

a) Was repräsentiert ein Ausdruck?

```
Eine Berechnung
```

b) Geben Sie einen primären Ausdruck an!

```
0
```

c) Geben Sie einen zusammengesetzten Ausdruck an!

```
1 + 1
```

d) Was bedeutet es, einen Ausdruck auszuwerten?

```
Seinen Wert zu bestimmen
```

e) Welchen Typ haben die folgenden beiden Ausdrücke?

1/2 ⇨ `int` 1*2 ⇨ `int`

4 Ausdrücke II

Gebt den Wert der folgenden Ausdrücke an

`3 + 4 * 5`	⇨	`23`
`5 / 2`	⇨	`2`
`0.9 * 10.0`	⇨	`9.0`
`17 < 4`	⇨	`false`

5 Variablen

Wozu dient eine Variable?

```
Zur Speicherung eines (veränderbaren) Wertes unter einem Namen
```

Wahr oder Falsch?

```
Ein Variablenname ist ein Ausdruck.
```

```
Jede Variable hat einen Typ.
```

6 Variablen II

Was ist die Ausgabe dieser Programme?

```
int x = 10;
x = 2 * x;
std::cout << x << std::endl;
```

```
20
```

```
int y = 5;
int z = 3 * y;
std::cout << y + z << std::endl;
```

```
20
```

7 If-Statement

Welche Frage beantwortet der folgende Programmabschnitt?

```
int a;
std::cin >> a;
if (a % 2 == 0) {
    std::cout << "Ja" << std::endl;
}
else {
    std::cout << "Nein" << std::endl;
}
```

Ist die Eingabezahl a gerade?

8 While-Statement

Was ist die Ausgabe des folgenden Programmabschnitts?

```
int b = 1;
while (b < 100) {
    std::cout << b << " ";
    b = 3 * b;
}
```

```
1 3 9 27 81
```

Nachbesprechung

Pascal Schärli *08.03.2019*

Bild: look again by corpus delicti from the Noun Project

ETH zürich

Task 1: Expressions

Ist diese Expression ein lvalue oder rvalue?

- **Nicht** ob die einzelnen Elemente lvalues sind, sondern die **ganze** Expression
- Kann man die ganze Expression umklammern und links von einem Gleichheitszeichen setzten?
- Ja!

    1. b = 5
    2. a = 5
    3. a = 42

```
a =  b = 5
```

```
a = (b = 5)
```

```
(a = (b = 5)) = 42
```

Task 1: Expressions #2

```
int a = 1;
```

Ist diese Expression valid? Ja

```
a + a++
```

```
((a) + (a++))
```

Was ist der Wert dieser Expression?

a++ hat höhere Präzedenz als die Addition, aber das bedeutet nicht, dass (a++) vor (a) ausgewertet wird!

Abhängig nach Auswertungsrichtung ist der Wert 3 oder 4

Best of Vorlesung

Fliesskomma Zahlen

Fliesskommazahlen können entweder als float oder double geschrieben werden.

`float` -> *32* Bit Floating point number

`double` -> *64* Bit, doppelte präzision

```
double x = 1.5 / 4;
std::cout << x << std::endl;
```

```
0.375
```

Expressions

Beim Rechnen mit verschiedenen Dateitypen wird immer in den "allgemeineren" Typ umgewandelt

Expressions

- Welche Expressions sind gültig?
- Welche gültigen Expressions sind lvalues?
- Was ist der Wert der gültigen Expressions?
  (x = 1, y = -1)

```
(y++ && y) + 2.0
```

```
(y++ * y) + 2.0
```

```
y = (x++ = 3)
```

```
3.0 + 3 - 4 + 5
```

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

Expressions

Welche Expressions sind gültig?

```
(y++ && y) + 2.0
```

```
(y++ * y) + 2.0
```

```
y = (x++ = 3)
```

```
3.0 + 3 - 4 + 5
```

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

Expressions

Welche Expressions sind lvalues?

```
(y++ && y) + 2.0
```
⇨ rvalue

```
(y++ * y) + 2.0
```
⇨ rvalue

```
y = (x++ = 3)
```

```
3.0 + 3 - 4 + 5
```
⇨ rvalue

```
5 % 4 * 3.0 + true * x++
```
⇨ rvalue

Expressions

Was ist der Wert dieser Expression?

```
(y++ && y) + 2.0
```

⇨ 2.0

```
(y++ * y) + 2.0
```

⇨ undefined

```
y = (x++ = 3)
```

```
3.0 + 3 - 4 + 5
```

⇨ 7.0

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

⇨ 4.0

Scopes

```
int a = 2;

if (x < 7) {
    int a = 8;
    std::cout << a;
}

std::cout << a;
```

```
82
```

```
int a = 2;

if (x < 7) {
    a = 8;
    std::cout << a;
}

std::cout << a;
```

```
88
```

Scopes

```
int sum = 0;
for (int i = 0; i < 5; ++i) {
    int a;
    std::cin >> a;
    sum += a;
}
```

Was ist der Scope dieser Variablen?

`sum` ⇨ Mindestens das ganze Code Snippet

`i` ⇨ Innerhalb der for-Schleife

`a` ⇨ Innerhalb einer Iteration der for-Schleife

Do-While

- Alternative zum while-Loop
- Kondition wird erst am Ende geprüft
- Schleifenkörper wird mindestens ein Mal ausgeführt

```
int i = 1;
do{
    std::cout << i << " ";
    i*=2;
} while(i < 10);
```

```
1 2 4 8
```

Loops

- Gibt es unterschiede zwischen diesen drei Loops?
- Falls ja, warum entstehen diese Unterschiede?

Lösung:

- Der letzte Loop hat den Output 1 falls $n \leq 0$
- Loops 1 und 3 terminieren nie wenn n der grösstmögliche Integer ist

```cpp
#include <iostream>

int main () {
    std::cout << "Enter a number: ";
    int n;
    std::cin >> n;

    // loop 1
    for (int i = 1; i <= n; ++i)
        std::cout << i << "\n";

    // loop 2
    int i = 0;
        while (i < n)
    std::cout << ++i << "\n";

    // loop 3
    i = 1;
    do
        std::cout << i++ << "\n";
    while (i <= n);

return 0;
}
```

Loops

Wie müsste man die folgende for-Schleife in eine while-Schleife umwandeln?

```
for (int i = 0; i < n; ++i)
    BODY
```

```
{
    int i = 0;
    while (i < n) {
        BODY
        ++i;
    }
}
```

Loops

Wie müsste man die folgende while-Schleife in eine for-Schleife umwandeln?

```
while (condition)
    BODY
```

```
for ( ;condition; )
    BODY
```

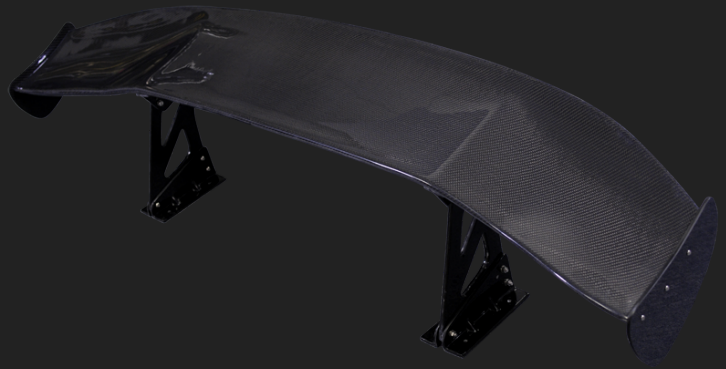
Loops

Wie müsste man die folgende do-while-Schleife in eine for-Schleife umwandeln?

```
do
    BODY;
while (condition);
```

```
BODY;
for ( ;condition; )
    BODY;
```

Vorbesprechung



Pascal Schärli *08.03.2019*

ETH zürich

Loop Snippets

- Was berechnen diese Loops?
- Welcher Loop würde jeweils besser passen?
- Schreibt die Loops in eine besser passende Form um

Tipp:

- Es gibt zu jedem Task eine eigene "Sandbox" um den Code auszuprobieren

```
unsigned int n;
std::cin >> n;
unsigned int f = 1;
if(n != 0) {
  do {
    f = f * n;
    --n;
  } while(n > 0);
}
std::cout << f << std::endl;
```

```
while(true) {
  int i1, i2;
  std::cin >> i1 >> i2;
  std::cout << i1+i2 << "\n";
  int again;
  std::cout << "Again?(0/1)\n";
  std::cin >> again;
  if(!again)
    break;
}
```

```
unsigned int z;
unsigned int d;

for(std::cin >> z >> d ; z >= d ; z = z-d);
std::cout << z << std::endl;
```

Loop Analysis

- Was berechnen dieser Loop?
- Für welche n ist der Output korrekt?
- Beweise, dass das Programm für alle n mit korrektem Output teriminiert (Induktion)
- Findet eine elegantere Implementation.
  -> Andere Schleifenart

```
unsigned int n;
std::cin >> n;

unsigned int x = 1;
if (n > 0) {
  unsigned int k = 0;
  bool e = true;
  do {
    if (++k == n) {
      e = false;
    }
    x *= 2;
  } while(e);
}
std::cout << x << std::endl;
```

Approximation of Pi

$$\frac{\pi}{4} = \sum_{n \geq 0} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

- Die Zahl Pi kann durch verschiedene Summen approximiert werden.
- Schreibt ein Programm welches diese Summe bis auf eine angegebene Anzahl Summanden approximiert
- Tipp: benutzt `double`

Approximation of Pi 2

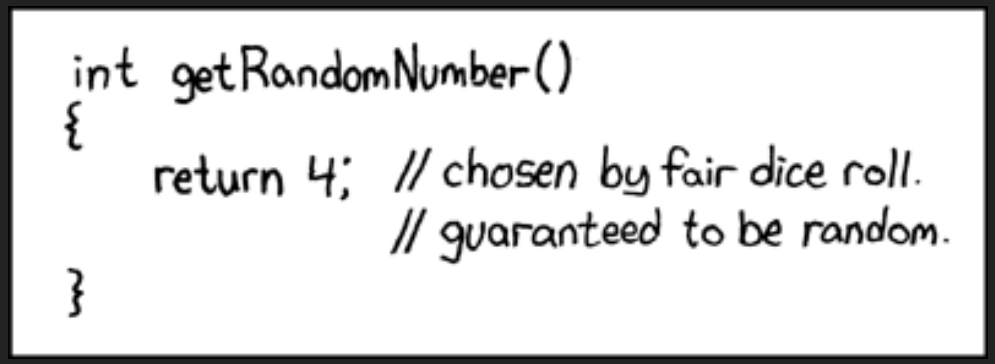
$$\frac{\pi}{2} = 1 + \sum_{1 \leq n} \frac{\prod_{0<1\leq n} i}{\prod_{0<i\leq n} (2i+1)}$$

$$= 1 + \frac{1}{3} + \frac{1\times2}{3\times5} + \frac{1\times2\times3}{3\times5\times7} + \frac{1\times2\times3\times4}{3\times5\times7\times9} + \cdots$$

- Auch mit dieser Summe kann pi approximiert werden
- Schreibt ein Programm welches diese Summe bis auf eine angegebene Anzahl Summanden approximiert
- Tipp: benutzt `double`

Program of the Week

Viel Spass!


int getRandomNumber()
{
return 4; // chosen by fair dice roll.
// guaranteed to be random.
}