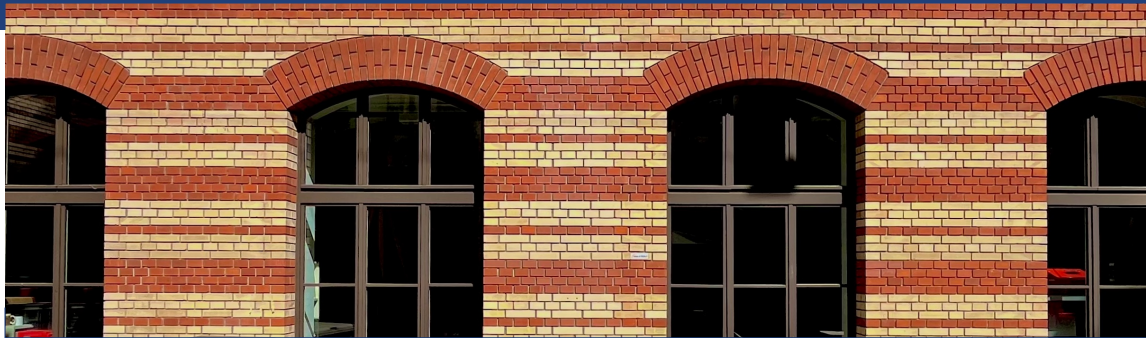




Informatik

Flieskommazahlen, Sichtbarkeitsbereiche (Scopes), Debugging, Ausdrücke

Adel Gavranović — ETH Zürich — 2025

Überblick

Lernziele

Sichtbarkeitsbereich

for-Schleifen

Debugging

Extra: Debugging in **code expert**

Repetition: Ausdrücke

Fliesskommazahlen in C++

Leitlinien für Fliesskommazahlen

Fliesskommazahlen vergleichen

Extra: Selektionsausdrücke



n.ethz.ch/~agavranovic

 Material

 Webpage

 Mail

1. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- Habe ich etwas vergessen?

2. Feedback zu **code** expert

Allgemeines bezüglich **code expert**

Fragen bezüglich **code expert** eurerseits?

□ punkte für text-based tabs

3. Lernziele

Ziele

Ziele

- ☐ In der Lage sein, Programme mit `for`-Schleifen nachzuvollziehen und *tracen* zu können
- ☐ In der Lage sein, einfache Programme zu schreiben, die `for`-Schleifen verwenden.
- ☐ Die Leitlinien zu Fließkommazahlen kennen und anwenden können
- ☐ Einfache Debugging-Methoden kennen und anwenden können

4. Zusammenfassung

Getting on the same page

Getting on the same page

- Was wurde diese Woche behandelt?

Kurzeinführung zu Scopes

Kurzeinführung zu Scopes

- Praktisch immer, wenn ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}

Kurzeinführung zu Scopes

- Praktisch immer, wenn ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}
- Stellt euch einen Scope als eine "Welt in einer Welt" vor

```
main {  
    x = ...;  
    {  
        a = 5;  
        print ... a;  
    }  
}
```

The image shows a handwritten code snippet in red and blue ink. The code is: `main {`, `x = ...;`, `{`, `a = 5;`, `print ... a;`, `}`, `}`. Annotations include: a blue bracket on the right side of the `main {` block; a red bracket on the left side of the inner `{` block; a red bracket on the left side of the `print ... a;` line; a blue bracket on the left side of the `print ... a;` line; and a grey arrow pointing from the `a` in `print ... a;` to the `a = 5;` line.

Kurzeinführung zu Scopes

- Praktisch immer, wenn ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}
- Stellt euch einen Scope als eine "Welt in einer Welt" vor
- Informationen/Variablen können nicht aus einem Scope rausfliessen, aber Informationen/Variablen von aussen sind im inneren Scope verfügbar

Kurzeinführung zu Scopes

- Praktisch immer, wenn ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}
- Stellt euch einen Scope als eine "Welt in einer Welt" vor
- Informationen/Variablen können nicht aus einem Scope rausfliessen, aber Informationen/Variablen von aussen sind im inneren Scope verfügbar
- Wenn das Programm bei der rechten geschweiften Klammer des Scopes ankommt, stirbt jegliche Information/Variable innerhalb dieses Scopes

Sichtbarkeitsbereich (Scopes)

Sichtbarkeitsbereich (Scopes)

Wenn ihr mehr über Sichtbarkeitsbereiche (Scopes) erfahren möchten, lest den C++-Leitfaden auf der Kurs-Website. Es enthält auch einen netten C++ Style Guide, der eine grosse Hilfe beim Erreichen von 3/3 TA-Punkten sein wird.  [Link](#)

Fragen zu allen Themen im Leitfaden könnt ihr natürlich auch direkt stellen.

Fragen/Unklarheiten?

6. for-Schleifen

Struktur einer `for`-Schleife

Struktur einer for-Schleife

```
for (init; condition; expression) {  
    someStatement;  
    someOtherStatementButItsAFunction();  
    ansAnotherStatement;  
    // ...  
} exp ~ ;
```

$i++;$
 $++i;$
 i
 $i \leftarrow i+1$

Wie wird so eine for-Schleife ausgeführt?

Beispiel

```
int sum = 0;

for (int i = 1; i <= 3; ++i){
    sum += i;
}

std::cout << sum << "\n";
```

Wie wird dieses Programm ausgeführt?

Beispiel for-Schleife

```
int sum = 0;
for (int i = 1; i <= 3; ++i){
    sum += i;
}
std::cout << sum << "\n";
```

Beispiel for-Schleife

```
→ int sum = 0;
   for (int i = 1; i <= 3; ++i){
       sum += i;
   }
   std::cout << sum << "\n";
```

int
sum:
0

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

0

int

i:

1

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    ⚡ sum += i;  
}  
std::cout << sum << "\n";
```

$i \leq 3$

$1 \leq 3$

true

sum:

0

i:

1

$1 \leq 3 \rightarrow \text{true}$

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

//

sum = sum + i;

sum = 5;

sum:

1

i:
1

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

1

i:

2

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

1

i:

2

2 <= 3 → true

Beispiel for-Schleife

```
int sum = 0;
for (int i = 1; i <= 3; ++i){
    sum += i;
}
std::cout << sum << "\n";
```

sum:

3

i:

2

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

3

i:

3

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

} body

sum:

3

i:

3

$3 \leq 3 \rightarrow \text{true}$

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

6

i:

3

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

6

i:

4

Beispiel for-Schleife

```
int sum = 0;  
for (int i = 1; i <= 3; ++i){  
    sum += i;  
}  
std::cout << sum << "\n";
```

sum:

6

i:

4

$4 \leq 3 \rightarrow \text{false}$

Beispiel for-Schleife

```
int sum = 0;
for (int i = 1; i <= 3; ++i){
    sum += i;
}
std::cout << sum << "\n";
```

sum:

6

Beispiel for-Schleife

```
int sum = 0;
for (int i = 1; i <= 3; ++i){
    sum += i;
}
std::cout << sum << "\n";
```

Ausgabe

6

Fragen/Unklarheiten?

Aufgabe Strange Sum

Aufgabe

Öffnet *Strange Sum* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen.

Description

Write a program that reads a number $n > 0$ from standard input and outputs the sum of all positive numbers up to n that are odd but not divisible by 5.

Aufgabe Strange Sum

Aufgabe

Öffnet *Strange Sum* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen.

Description

Write a program that reads a number $n > 0$ from standard input and outputs the sum of all positive numbers up to n that are odd but not divisible by 5.

```
int a;  
std::cin >> a;
```

and including : $i \in (0, n]$

Mögliche Lösung zu Strange Sum

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

// computation
for(int i = 1; i <= n; i++){
    → if((i % 2) == 1){
        if(i % 5){
            strangesum += i;
        }
    }
}

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

$i = 5n \quad n \in \mathbb{Z}$

$i \neq 5$

0

if (false)
X

Kompaktere Lösung zu *Strange Sum*

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

// computation
for(int i = 1; i <= n; i++){
    if( ((i % 2) == 1) && (i % 5) ){
        strangesum += i;
    }
}

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

Noch kompaktere Lösung zu *Strange Sum*

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

// computation
for(int i = 1; i <= n; i+=2){
    if(i % 5){
        strangesum += i;
    }
}

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

Fragen/Unklarheiten?

Aufgabe *Largest Power*

Aufgabe

Öffnet *Largest Power* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen.

Description

Write a program that inputs a positive natural number n and outputs the largest number p that is a power of 2 and smaller or equal to n .

Aufgabe *Largest Power*

Aufgabe

Öffnet *Largest Power* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen.

Description

Write a program that inputs a positive natural number n and outputs the largest number p that is a power of 2 and smaller or equal to n .

Mögliche Lösung zu Largest Power

$n \in \mathbb{R} (27, 129, 2^k \pm 1)$

```
#include <iostream>
```

```
int main () {
```

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

```
    int power; 200;
```

```
    for (; power <= n / 2; power *= 2);
```

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

```
    return 0;
```

```
}
```

preview:

```
int var = 1 ;
```

$pow \leq \frac{15}{2}$

true $\Leftrightarrow pow \leq \frac{15}{2}$

Fragen/Unklarheiten?

7. Debugging

Debugging

Debugging

...ist der Prozess des Auffindens und der Behebung von Fehlern (Defekte oder Probleme, die den korrekten Betrieb verhindern) in Programmen, Software oder Systemen.

Debugging

```
int main () {  
    const int n = 6;  
  
    // Compute n^12  
    int prod = 1;  
    for (int i = 1; 1 <= i < 13; ++i) {  
        prod *= n;  
    }  
  
    // Output stars  
    for (int i = 1; i < prod; ++i) {  
        std::cout << "*";  
    }  
    std::cout << "\n";  
    return 0;  
}
```

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Frage

Wieso steckt das Programm in der ersten for-Schleife fest?

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Frage

Wieso steckt das Programm in der ersten for-Schleife fest?

Antwort

Die condition ist falsch geschrieben!

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Frage

Wieso steckt das Programm in der ersten for-Schleife fest?

Antwort

Die condition ist falsch geschrieben! Sollte sein: `(1 <= i) && (i < 13.)`

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Antwort

Einfach den Wert von `prod` nach der ersten Schleife ausgeben lassen

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Antwort

Einfach den Wert von `prod` nach der ersten Schleife ausgeben lassen

Frage

Wie finden wir raus, wieso `prod` negativ wurde?

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Antwort

Einfach den Wert von `prod` nach der ersten Schleife ausgeben lassen

Frage

Wie finden wir raus, wieso `prod` negativ wurde?

Antwort

Einfach den Wert von `prod` in *jeder* Iteration in der Schleife ausgeben lassen

Fragen/Unklarheiten?

7. Debugging

7.1. Extra: Debugging in **code** expert

Debugging in **code** expert

Debugging in **code expert**

```
int main() {  
  
    int start = 2;  
    // Finding the largest power of 2 representable by int  
    while (true) {  
        if (start * 2 > INT_MAX)  
            std::cout << "Stopping..." << std::endl;  
            break;  
  
        start *= 2;  
    }  
  
    std::cout << "Largest power of two representable by int: "  
              << start << std::endl;  
    // ...  
}
```

Debugging in **code expert**

```
// ...  
// Finding the smallest negative power of 2 representable by int  
start = -2;  
while (true) {  
    if (start * 2 < INT_MIN)  
        std::cout << "Stopping..." << std::endl;  
        break;  
  
    start *= 2;  
}  
  
std::cout << "Smallest negative power of two representable by int: "  
    << start << std::endl;  
  
return 0;  
}
```

Debugging in **code expert**

Aufgabe

Öffnet das Beispiel "Debugging in CodeExpert" auf **code expert**

Description

Find the largest (N) and smallest negative (n) power of two that is representable by **int**. Formally that is:

$$\begin{aligned} N &= \max(\{2^i : i \in \mathbb{N} \text{ and } 2^i \text{ is representable by } \mathbf{int}\}) \\ n &= \min(\{-2^i : i \in \mathbb{N} \text{ and } -2^i \text{ is representable by } \mathbf{int}\}) \end{aligned}$$

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Antwort

2er-Potenzen berechnen, bis man den maximalen Wert von **int** erreicht

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Antwort

2er-Potenzen berechnen, bis man den maximalen Wert von **int** erreicht

Frage

Es gibt einen Syntaxfehler im Code. Kann jemand erkennen, was falsch ist?

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Antwort

2er-Potenzen berechnen, bis man den maximalen Wert von **int** erreicht

Frage

Es gibt einen Syntaxfehler im Code. Kann jemand erkennen, was falsch ist?

Antwort

die {} fehlen nach dem **if**-Zweig.

Debugging in **code expert**

Frage

Was ist an diesem Programm noch falsch?

Debugging in **code expert**

Frage

Was ist an diesem Programm noch falsch?

Antwort

Der Ausdruck `*2` führt zu einem **overflow**, was ein undefiniertes Verhalten für `signed int` ist. Hier führt es zu einer Endlosschleife!

Debugging in **code expert**

Frage

Was ist an diesem Programm noch falsch?

Antwort

Der Ausdruck `*2` führt zu einem **overflow**, was ein undefiniertes Verhalten für `signed int` ist. Hier führt es zu einer Endlosschleife!

Frage

Wie können wir also den Code korrigieren?

Debugging in **code expert** – Lösung

```
int main() {  
    int start = 2;  
    // Finding the largest power of 2 representable by int  
    while (true) {  
        if (start > INT_MAX / 2) {  
            std::cout << "Stopping..." << std::endl;  
            break;  
        }  
        start *= 2;  
    }  
  
    std::cout << "Largest power of two representable by int: "  
              << start << std::endl;  
    // ...  
}
```

Debugging in **code expert** – Lösung

```
// ...  
// Finding the smallest negative power of 2 representable by int  
start = -2;  
while (true) {  
    if (start < INT_MIN / 2) {  
        std::cout << "Stopping..." << std::endl;  
        break;  
    }  
    start *= 2;  
}  
  
std::cout << "Smallest negative power of two representable by int: "  
          << start << std::endl;  
  
return 0;  
}
```

Fragen/Unklarheiten?

for (...) {

a

b

c

:

8. Repetition: Ausdrücke

⚡

Werte die folgenden Ausdrücke aus

1. `5 < 4 < 1`

2. `true > false`

$(5 < 4) < 1$
false

true, bool

Lösung

$$5 < 4 < 1$$

$$(5 < 4) < 1$$

Lösung

5 < 4 < 1

(5 < 4) < 1

false < 1

Lösung

5 < 4 < 1

(5 < 4) < 1

false < 1

0 < 1

Lösung

5 < 4 < 1

(5 < 4) < 1

false < 1

0 < 1

true, True)

Lösung

```
true > false
```

Lösung

```
true > false
```

```
1 > 0
```

Lösung

```
true > false
```

```
1 > 0
```

```
true
```

Fragen/Unklarheiten?

9. Fließkommazahlen in C++

Leitlinie 1

Leitlinie 1

→ variablen des typs float oder double
“Vermeide es, zwei Fließkommazahlen auf Gleichheit zu prüfen, nachdem unterschiedliche Berechnungen mit ihnen durchgeführt wurden.”

double ..., a; b;

a == b;

↑

↑

1.1 ≠ 1.0 + 0.1

Leitlinie 1

“Vermeide es, zwei Fließkommazahlen auf Gleichheit zu prüfen, nachdem unterschiedliche Berechnungen mit ihnen durchgeführt wurden.”

Beispiel

Leitlinie 1

“Vermeide es, zwei Fließkommazahlen auf Gleichheit zu prüfen, nachdem unterschiedliche Berechnungen mit ihnen durchgeführt wurden.”

Beispiel

```
float a = 2.0f;  
if (std::sqrt( $\sqrt{a}$ 2.0f) * std::sqrt( $\sqrt{a}$ 2.0f) == a){  
    std::cout << "no output\n";  
}
```

Leitlinie 1

“Vermeide es, zwei Fließkommazahlen auf Gleichheit zu prüfen, nachdem unterschiedliche Berechnungen mit ihnen durchgeführt wurden.”

Beispiel

```
float a = 2.0f;  
  
if (std::sqrt(2.0f) * std::sqrt(2.0f) == a){  
    std::cout << "no output\n";  
}
```

Leitlinie 1

“Vermeide es, zwei Fließkommazahlen auf Gleichheit zu prüfen, nachdem unterschiedliche Berechnungen mit ihnen durchgeführt wurden.”

Beispiel

```
float a = 2.0f;  
  
if (std::sqrt(2.0f) * std::sqrt(2.0f) == a){  
    std::cout << "no output\n";  
}
```

Das ist falsch! Das Problem: `std::sqrt(2.0f)` ist *nicht* darstellbar!

Leitlinie 2

Leitlinie 2

“Vermeide Addition von Zahlen mit extrem unterschiedlichen Grössen”

Leitlinie 2

“Vermeide Addition von Zahlen mit extrem unterschiedlichen Grössen”

Beispiel

Leitlinie 2

“Vermeide Addition von Zahlen mit extrem unterschiedlichen Grössen”

Beispiel

+ 6 ... 1 - 6 ...

```
float a = 671088640.0f + 1.0f;  
  
if (a > 671088640.0f){  
    std::cout << "This does not output ... \n";  
}
```

Leitlinie 2

“Vermeide Addition von Zahlen mit extrem unterschiedlichen Grössen”

Beispiel

```
float a = 671088640.0f + 1.0f;  
  
if (a > 671088640.0f){  
    std::cout << "This does not output ... \n";  
}
```

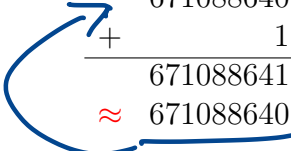
Das ist falsch! Das Problem: Signifikante zu kurz!

Leitlinie 2 – Rundung

Leitlinie 2 – Rundung

$$\begin{array}{r} 671088640 = 6.7108864 \cdot 10^8 \\ + \quad \quad \quad \underline{1} = 0.00000001 \cdot 10^8 \\ \hline 671088641 = 6.71088641 \cdot 10^8 \end{array} \quad \leftarrow$$

Leitlinie 2 – Rundung

$$\begin{array}{r} 671088640 = 6.7108864 * 10^8 \\ + \quad 1 = 0.00000001 * 10^8 \\ \hline 671088641 = 6.71088641 * 10^8 \\ \approx 671088640 = 6.7108864 * 10^8 \end{array}$$


92.0
7f → double
→ float

Leitlinie 2 – Rundung

$$\begin{array}{r} 671088640 = 6.7108864 * 10^8 \\ + \quad 1 = 0.00000001 * 10^8 \\ \hline 671088641 = 6.71088641 * 10^8 \\ \approx 671088640 = 6.7108864 * 10^8 \end{array} \quad (\text{Rundung!})$$

Fliesskommazahlen vergleichen

Fliesskommazahlen vergleichen

Kurzgesagt: nicht auf *Gleichheit* prüfen lieber auf "innerhalb der Toleranz" prüfen

// Example of "equality" check function for doubles

```
bool equal(double x, double y, double tol){
```

```
    double diff = x - y;
```

```
    if(diff < 0){
```

```
        diff *= -1;
```

```
    }
```

```
    return (diff < tol);
```

```
}
```

↑ "EPS"

ϵ

|diff|

(var eps)

Wichtig

Wichtig

Diese Folien sollen ein Gefühl dafür vermitteln, warum "Löcher" im Wertebereich auftreten. In einigen Wochen werden wir mehr darüber erfahren, wie Fließkommazahlen in C++ dargestellt werden.

10. Extra: Selektionsausdrücke

Even Numbers

Even Numbers

Aufgabe

Öffnet das Beispiel "Even Numbers" auf **code expert** und versucht es zu lösen

Description

Write a program that accepts an input value n (as an **int**). If the input value is non-negative, it outputs the biggest even number m that does not exceed the input value. If the input value is negative, then the program should output 0.

Input A whole number n

Output

- If $n \geq 0$, output the biggest even number $m \leq n$
- If $n < 0$, output 0

Examples

$2 \rightarrow 2,$ $13 \rightarrow 12,$ $43 \rightarrow 42$

Even Numbers – Lösung

Even Numbers – Lösung

```
int main () {  
  
    int n;  
    std::cin >> n;  
  
    if (n < 0) {  
        n = 0;  
    }  
  
    if (n % 2 == 1) {  
        n = n - 1;  
    }  
  
    std::cout << n << std::endl;  
  
    return 0;  
}
```

Fragen/Unklarheiten?

11. Outro

Allgemeine Fragen?

Bis nächstes Mal

Schöne Woche noch!