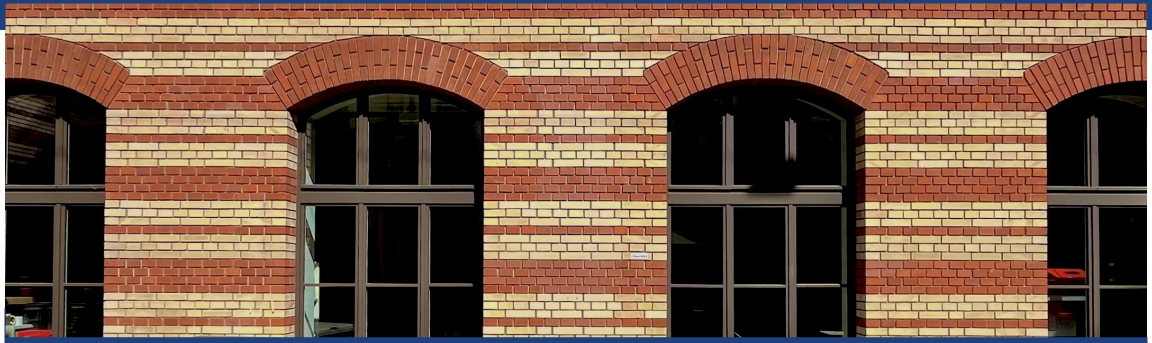


Exercise Session — Computer Science — 02

Expressions, `for`-loops, Debugging

Overview

Today's Plan

Feedback regarding **code** expert
Boolean Expressions
`for` Loops
Debugging
Debugging in **code** expert
Selection Statements
Outro

1. Feedback regarding **code** expert

General things regarding **code** expert

Any questions regarding **code** expert on your part?

2. Boolean Expressions

Booleans

`bool` is a data type that can only take two values: {`true`, `false`}

Conversions

- Bool to Number
  - if `true` is converted to a number, it will be the number 1
  - if `false` is converted to a number, it will be the number 0
- Number to Bool
  - if 0 is converted to a `bool`, it will be the value `false`
  - if a number $\neq 0$ is converted to a `bool`, it will be the value `true`

Precedence Ranking

Precedence Ranking[1]

1. `a++, a--`
2. `++a, --a, -a, !a, *a, &a`
3. `*, /, %`
4. `+, -`
5. `<, <=, >, >=`
6. `== !=`
7. `&&`
8. `||`
9. `=, +=, -=, *=, /=, %=`

[1] Only the currently most important ones are shown here. For the rest, see cppreference

(Use) (brackets)

- Parentheses work like in mathematics
- they are often used to make the correct evaluation obvious
- …or to change the evaluation order

Task

Make the evaluation obvious using brackets:

```
3 < 4 + 1 && 2 < 3
```

Hint: use the previous slide

Solution

```
(3 < (4 + 1)) && (2 < 3)
```

Multiple operators with the same precedence

Assoziativität

- Indicates the order in which operations are evaluated
- Either Right-to-left ($\leftarrow$) or Left-to-right ($\rightarrow$)
- 💡 Ideally, write a table on the summary with precedences and associativities

Mehrere Operatoren mit gleicher Präzedenz

Kurzaufgabe
How would you put the parentheses so that the evaluation of the expression below is obvious?
Hint: The associativity of `&&` *is left-to-right*

```
a && b && c
```

Solution

```
(a && b) && c
```

Short Circuits

The boolean operators `&&` and `||` first evaluate the left expression and then *only if necessary* the right expression.

In particular, this means that the right-hand expression *not* is evaluated if the result of the entire evaluation can already be derived.

Short Circuits in Code

```
if(3 > 2 && 10 > 11){
    std::cout << "Of course not!\n";
}    // not a short circuit evaluation
```

Short Circuits in Code

```
int a = 3;

if(false && ++a < 2){
    std::cout << "Of course not!\n";
} // a short circuit evaluation

std::cout << a << "\n"; // what will be the output?

if(++a < 2 && false){
    std::cout << "Of course not!\n";
} // another short circuit evaluation

std::cout << a << "\n"; // what will be the output?
```

Comprehension Check I

Task

Evaluate the following expression by hand and note each intermediate step.
Assume `int x = 1, int y = 1`:

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

Solution

```
(2>3) && 17 u - 55 <= ++x + y    start with left side
false && 17 u - 55 <= ++x + y    short circuit!
false
```

Remember (`false && ...`) always evaluates to (`false`)

Comprehension Check II

Task

Evaluate the following expression by hand and note each intermediate step. Assume `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Solution

```
!(1 < 2 && x == 1) + 1
(!((1 < 2) && (x == 1))) + 1
(!((true) && (true))) + 1
(!(true)) + 1
false + 1
0 + 1
1
```

Comprehension Check III

Task

Evaluate the following expression by hand and note each intermediate step.
Assume `int x = 1`:

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

Solution

First: Parentheses!

```
(x == 1) || ((1 / (x - 1)) < 1)    start with left side
(1 == 1) || ((1 / (x - 1)) < 1)
true || ((1 / (x - 1)) < 1)    short circuit!
true
```

Remember (`true || ...`) always evaluates to (`true`)

Questions?

Minimal Expression

Oftentimes, one can simplify expressions to be easier to understand or to be more elegant and take up less attention. For example:

```
(x > 3) == true || z - 3 >= 0
```

can be simplified to

```
(x > 3) || (z >= 3)
```

Comprehension Check I

Task

Minimize the following boolean expression. Assume `bool a,b; int n;`:

```
b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)
```

Solution

```
b && n > 0 && n < 6
```

or to make it even more obvious:

```
b && (n > 0) && (n < 6)
```

This is very wrong:

```
b && (0 < n < 6)
```

But why?

Comprehension Check II

Task

Minimize the following boolean expression. Assume `bool a,b; int n;`:

```
a && !(a == true) || (b == false)
```

Solution

```
!b
```

Comprehension Check III

Task

Minimize the following boolean expression. Assume `bool a,b; int n;`:

```
!(a != false && !(b == false) == false)
```

Solution

```
!a || b
```

Questions?

3. `for` Loops

`for` Loops

- What is a loop
- Brief introduction to *Scopes*
- Brief introduction to *Program Tracing*

Brief introduction to Scopes

- Basically every time you use {these curly braces}, you create a {scope}
- Think of a scope as a "world within a world"
- Information/variables cannot flow out of a scope, but information/variables from the outside are available in the inner scope
- When the program arrives at the right curly bracket of the scope, any information/variable within this scope dies

General structure of a `for` loop

```
for(init; condition; expression){
    statement 1;
    statement 2;
    // ....
}
```

Important Note

The `expression` part is executed *after* the `statements`.

Program Tracing

"*Program Tracing* is the process of executing a program manually with specific inputs."

Quite an important skill, especially at the beginning. At some point you'll be able to do it in your head. You can find a detailed guide here: Program Tracing

`for` loop example

`for` Loop

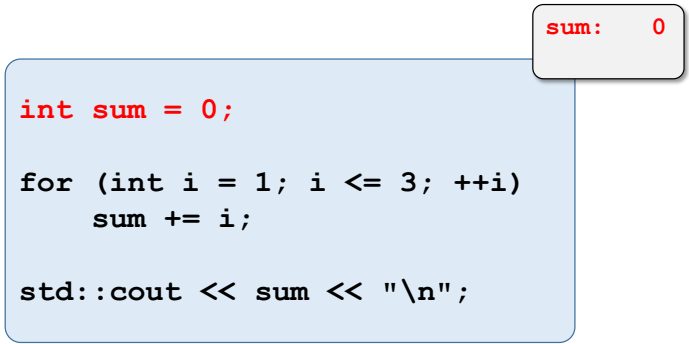
Example – `for` Loop

```
int sum = 0;

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

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

Example – `for` Loop



Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    0
i:      1
```

Example – `for` Loop

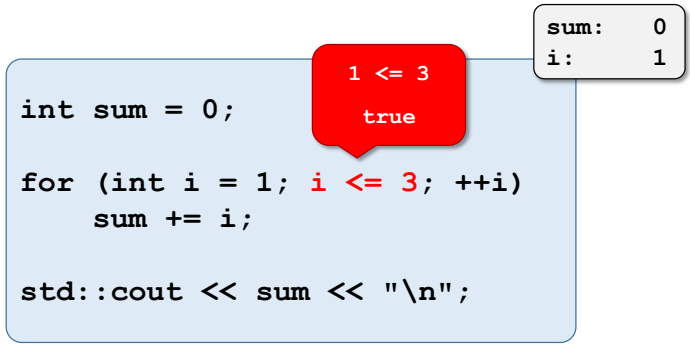
```
sum:    0
i:      1
```

```
int sum = 0;

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

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

Example – `for` Loop



Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    1
i:      1
```

Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    1
i:      2
```

Example – `for` Loop

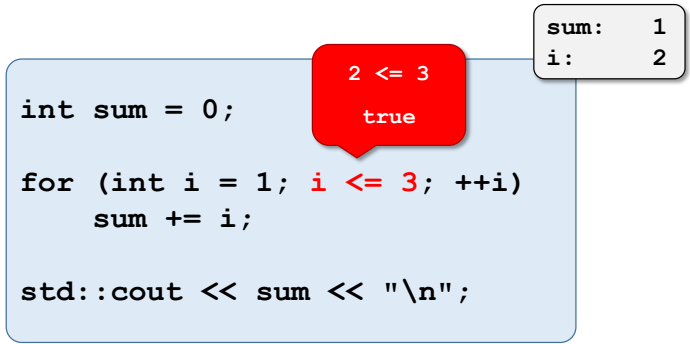
```
int sum = 0;

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

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

```
sum:    1
i:      2
```

Example – `for` Loop



Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    3
i:      2
```

Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    3
i:      3
```

Example – `for` Loop

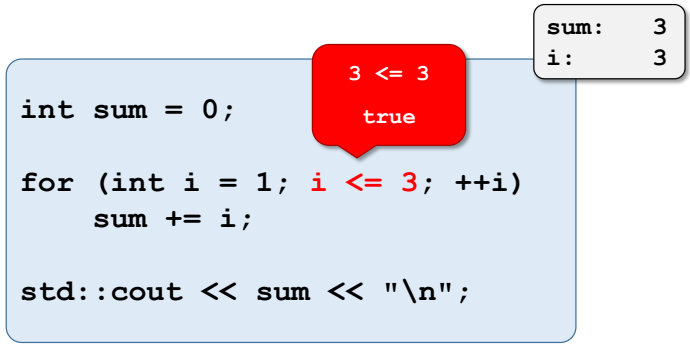
```
int sum = 0;

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

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

```
sum:    3
i:      3
```

Example – `for` Loop



Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    6
i:      3
```

Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    6
i:      4
```

Example – `for` Loop

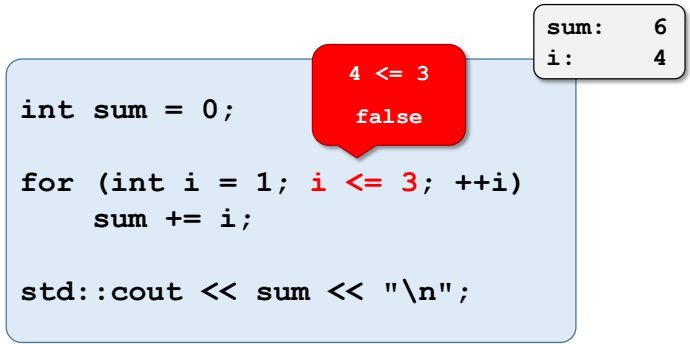
```
int sum = 0;

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

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

```
sum:    6
i:      4
```

Example – `for` Loop



Example – `for` Loop

```
int sum = 0;

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

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

```
sum:    6
```

Questions?

Task *Strange Sum*

Task
Open *Strange Sum* on **code** expert and try to solve it with pen and paper first.

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.

Task And now write the corresponding program.

Questions?

Possible solution to *Strange Sum*

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

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

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

More compact solution to *Strange Sum*

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

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

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

Even more compact solution to *Strange Sum*

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

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

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

Task *Largest Power*

Task
Open *Largest Power* on **code** expert and try to solve it with pen and paper first.

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.

Task
And now write the corresponding program.

Task
Discuss your approaches with the people next to you. Did you have the same approach? What can you learn from each other?

Possible solution to *Largest Power*

```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;
}
```

Questions?

4. Debugging

Debugging

Debugging
…is the process of finding and resolving bugs (defects or problems that prevent correct operation) within programs, software, or systems.

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

Question
How could we find out at which line the program is stuck?

Answer
Output things at interesting places in the code

Question
Why is the program stuck in the first for loop?

Answer
The condition is wrong!
Should be: `1 <= i && i < 13`.

Debugging - Live Demo

Question
How can we find out why nothing is being spent despite this?

Answer
Simply output the value of `prod` after the first loop

Question
How do we find out why `prod` became negative?

Answer
Simply output the value of `prod` in *each* iteration in the loop

Questions?

5. Debugging in **code** expert

Debugging in **code** expert

```
int main() {

    long start = 2;
    // Finding the largest power of 2 representable by long
    while (true) {
      if (start * 2 > LONG_MAX)
        std::cout << "Stopping..." << std::endl;
        break;

      start *= 2;
    }

    std::cout << "Largest power of two representable by long: "
              << start << std::endl;
    // ...
```

Debugging in **code** expert

```
    // ...
    // Finding the smallest negative power of 2 representable by long
    start = -2;
    while (true) {
      if (start * 2 < LONG_MIN)
        std::cout << "Stopping..." << std::endl;
        break;

      start *= 2;
    }

    std::cout << "Smallest negative power of two representable by long: "
              << start << std::endl;

    return 0;
}
```

Debugging in **code** expert

Task

Open the “Debugging in CodeExpert” example on **code** expert

Description

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

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

Debugging in **code** expert

Question
How is the program supposed to work?

Answer
Compute power of 2's until we reach the maximum `long` value.

Question
There is a syntax error in the code. Does anyone see what's wrong?

Answer
The `{}` are missing after the `if` branch.

Debugging in **code** expert

Question
What is still wrong with this program?

Answer
The expression start `*2` leads to an **overflow**, which is undefined behavior for `signed int`. Here, it leads to an infinite loop!

Question
So, how do we fix the code?

Debugging in **code** expert - Solution

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

    std::cout << "Largest power of two representable by long: "
              << start << std::endl;
    // ...
```

Debugging in **code** expert - Solution

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

    std::cout << "Smallest negative power of two representable by long: "
              << start << std::endl;

    return 0;
}
```

Questions?

6. Selection Statements

Even Numbers

Task
Open the "Even Numbers" example on **code** expert and try to solve it

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 $2 \to 2, \qquad 13 \to 12, \qquad 43 \to 42$

Even Numbers - Solution

```cpp
#include <iostream>
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;
}
```

Questions?

7. Outro

General Questions?

Bis zum nächsten Mal

Have a nice week!