Exercise 4 – Scope, Loops, Floating Types

Informatik I für Mathematiker und Physiker (HS 2015)

Yeara Kozlov

Slides courtesy of Kaan Yücer & Endri Dibra

Agenda

- HW #2 feedback
- **Scopes**
- Loops
- Floating point types
- Debugging

Scopes

- Scopes define the code lines of programs in which a variable (l-value) exists
- The scope of a variable
  - starts at the point of its definition
  - ends at the end of the block where it was defined.
- Code block: part of a code between { and }

```
int a = 2;
if (x < 7) {
      a=8;
}

                    // Outputs 2 or 8, depending on the if-statement.
```

Scopes

```
int sum = 0;

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

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

Scopes

```
int main()
{
    int x, a = 8;
    cin>> x;                    //Input = 5;
    {
        int a;
        cin>> a;                //Input = 3;
        a = a + a;
        cout<< a<<endl;
        x = a;
    }
    cout<< x*a<<endl;
    return 0;
}
```

Agenda

- HW #2 feedback
- Scopes
- **Loops**
- Floating point types
- Debugging

while loops

```
while (condition)
   statement
```

```
do
   statement
while (condition);
```

- Instruction inside the do - while loop are executed at least once before the condition is checked.
- Condition initial and changes need to be done outside the loop statement
- Allow for more complex execution logic

Example – `while` Loop

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

number:

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

number:

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

number:
6

number: 6

```
// Inpu
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  6
nbits:   0
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  6
nbits:   0
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  6
nbits:   0
```

```
// Input
unsigned int number;
std::cin >>
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

```
6 != 0

true
```

Example – `while` Loop

```
number:  3
nbits:   0
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  3
nbits:   1
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  3
nbits:   1
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  3
nbits:   1
```

```
// Input
unsigned int number;
std::cin >>
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

```
3 != 0

true
```

Example – `while` Loop

```
number:  1
nbits:   1
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  1
nbits:   2
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  1
nbits:   2
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  1
nbits:   2
```

```
// Input
unsigned int number;
std::cin >>
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

```
1 != 0

true
```

Example – `while` Loop

```
number:  0
nbits:   2
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  0
nbits:   3
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  0
nbits:   3
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  0
nbits:   3
```

```
// Input
unsigned int number;
std::cin >>
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

```
0 != 0
false
```

Example – `while` Loop

```
number:  0
nbits:   3
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbits;
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

Example – `while` Loop

```
number:  0
nbits:   3
```

```
// Input
unsigned int number;
std::cin >> number;
unsigned int nbits = 0;

// Determine number of bits
while (number != 0) {
    number /= 2;
    ++nbi
}

// Output
std::cout << "Number of bits: "
          << nbits << "\n";
```

```
Output:

Number of bits: 3
```

Example – `do` Loop

```
int input;

// Accept input only if >= 2
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Example – do Loop

input:

```
int input;

// Accept input only if >= 2
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Example – do Loop

input:

```
int input;

// Accept input only if >= 2
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Example – do Loop

input: 0

```
int input;

// Accept in                    if >= 2
do {
    std::cou                    put number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Input:
0

Example – do Loop

input: 0

```
int input;

// Accept input only if >= 2
do {
    std::cout                    ut number >= 2: ";
    std::cin                    ;
} while (input < 2);

... // do something with input
```

0 < 2

true

Example – do Loop

input: 0

```
int input;

// Accept input only if >= 2
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Example – do Loop

input: 5

```
int input;

// Accept in                    if >= 2
do {
    std::cou                    put number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Input:
5

Example – do Loop

input: 5

```
int input;

// Accept input only if >= 2
do {
    std::cout                    ut number >= 2: ";
    std::cin                    ;
} while (input < 2);

... // do something with input
```

5 < 2

false

Example – do Loop

```
input:   5
```

```
int input;

// Accept input only if >= 2
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);

... // do something with input
```

Why `do` and not `while`?

- Need `input` in condition.

```
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);
```

Why `do` and not `while`?

- Need `input` in condition.
- But: `input` receives value **in loop-body**
- Thus: condition must be evaluated **afterwards**

```
do {
    std::cout << "Input number >= 2: ";
    std::cin >> input;
} while (input < 2);
```

Example

What does this code snippet do?

```
int digits = 0;
int number;
std::cin >> number;

while (number != 0) {
  number /= 10;
  digits++;
}
```

Loop correctness

```
// Program: output_till_n.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;
}
```

n == 0?

nothing

nothing

1

Loop choice

- minimal code
- easy to understand

for	Some counting is done, but the counter is not needed after the loop. e.g. repeat something n times
while	The loop condition depends on variables which already exist before the loop. e.g. decrease x until it's a power of 5
do	The loop condition depends on variables which are obtained in the loop body. e.g. execute std::cin >> x until x > 3

loops - break execution

- break - stop loop execution
- continue - go immediately to the next loop iteration

Example – `break`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
a:        18
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

a:	18
n:	0

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

a:	18
n:	0
i:	1

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
a:        18
n:         0
i:         1
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

1 <= 5

true

a:	18
n:	0
i:	1

Example – `break`

```
a:        18
n:         0
i:         1
input:
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
a:        18
n:         0
i:         1
input:
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
const int a = 18;
int n = 0;

// How many                    t of 5) are divisors of a?
for (int i                     ; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Input

0

a:	18
n:	0
i:	1
input:	0

Example – `break`

```
const int a = 18;
int n = 0;

// How many                    t of 5) are divisors of a?
for (int i                     ; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Input

0

Note:

0 is

bad divisor

a:	18
n:	0
i:	1
input:	0

Example – `break`

```
a:        18
n:         0
i:         1
input:     0
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
const int a = 18;
int n = 0;

// How many                    t of 5) are divisors of a?
for (int i                     ; ++i) {
    int inp
    std::cin                   input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

0 == 0

true

a:	18
n:	0
i:	1
input:	0

Example – `break`

```
a:        18
n:         0
i:         1
input:     0
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
a:        18
n:         0
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `break`

```
a:        18
n:         0
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Output:

Number of divisors: 0

Example – `break`

```
const int a = 18;
int n = 0;

// How many inputs (out                    f a?
for (int i = 1; i <= 5
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

a:	18
n:	0

Note:

`i` and `input` **are gone**

Output:

Number of divisors: 0

Example – `continue`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input ==
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Note:

Same example,
using `continue`.

Example – `continue`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
a:        18
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

a:	18
n:	0

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

a:	18
n:	0
i:	1

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
a:        18
n:         0
i:         1
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

1 <= 5

true

a:	18
n:	0
i:	1

Example – `continue`

```
a:        18
n:         0
i:         1
input:
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
a:        18
n:         0
i:         1
input:
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
const int a = 18;
int n = 0;

// How many                    t of 5) are divisors of a?
for (int i                     ; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Input

0

a:	18
n:	0
i:	1
input:	0

Example – `continue`

```
const int a = 18;
int n = 0;

// How many                    t of 5) are divisors of a?
for (int i                     ; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Input

0

Note:

0 is

bad divisor

a:	18
n:	0
i:	1
input:	0

Example – `continue`

```
a:        18
n:         0
i:         1
input:     0
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
const int a = 18;
int n = 0;

// How many                    t of 5) are divisors of a?
for (int i                     ; ++i) {
    int inp
    std::cin                   input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

0 == 0

true

a:	18
n:	0
i:	1
input:	0

Example – `continue`

```
a:        18
n:         0
i:         1
input:     0
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
a:      18
n:       0
i:       2
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
a:      18
n:       0
i:       2
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Note:

++i is still executed

Example – `continue`

```
a:        18
n:         0
i:         2
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

2 <= 5

true

a:	18
n:	0
i:	2

Example – `continue`

```
a:        18
n:         0
i:         2
input:
```

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Example – `continue`

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <
    int input;
    std::cin >> inp
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

. . .

a:	18
n:	0
i:	2
input:	

break vs. continue

break:

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        break;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

continue:

```
const int a = 18;
int n = 0;

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> input;
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

// Output
std::cout << "Number of divisors: " << n << "\n";
```

Remark

- `continue` makes more sense here.

Remark

- `continue` makes more sense here.

- Reason:
  - `break`-version skips later inputs

Remark

- `continue` makes more sense here.

- Reason:
  - `break`-version skips later inputs
  - But output is still:

```
Number of divisors: ...
```

as if nothing went wrong.

Agenda

- HW #2 feedback
- Scopes
- Loops
- **Floating point types**
- Debugging

Variable types

- C++ has 5 built-in basic Datatypes

Data-type	Function	Memory
int	Saves integer numbers	2 Byte
float	Saves real numbers	4 Byte
double	Saves real numbers with higher precision than the float data-type	8 Byte
bool	Saves logical expressions (True/False or respectively 0/1)	1 Byte
char	Saves characters	1 Byte

Memory size may vary, because they depend on the implementation and Hardware.

float/double

- To express non-integer numbers, C++ has the built-in types float and double.
  - double number = 1.5
- double requires more space, and hence, has a higher precision

```
int j = 5/3;
std::cout << j;  // outputs 1

double j = 5.0/3.0;
std::cout << j;  // outputs 1.666667
```

- no modulos division for float and double.

Type Conversion

- Consider the following code:

```
float ft_var = 5.453;
int it_var = ft_var;
```

- The Variable „it_var“ cannot save a real number. So what happens when this code is executed?
- The value which will be assigned to „it_var“ will be converted to **int.** This process is called **type conversion**.

Type Conversion

- The assigned value will be converted to type of the variable to which it is assigned to
- At our Example:

```
float ft_var = 5.453;
int it_var = ft_var;
```

- This means that the value 5.453 will be converted to an integer.
- C++ does that by **truncating** the fractional part of the value
- Thus it_var has the value 5 after the assignment

Type Conversion - assignment

- Whenever a numerical value is assigned to a variable of another type, the value will be converted to the type of the receiving variable.

Type Conversion - assignment

- Before:

```
int j = 5/3;
std::cout << j;  // outputs 1

double j = 5.0/3.0;
std::cout << j;  // outputs 1.666667
```

- What happens here?

```
double j = 5/3;
```

Type Conversion - expression

- In arithmetic operations, variables are converted to the most general type:

```
5.453f * 3 * 9.86
```

float

int

double

Type Conversion

- C++ converts the expression into the more general type of the two variables.

```
bool < int < unsigned int < float < double
```

- For out example:

```
5.453f(float) * 3(int) * 9.86(double) =
   16.359(float)        * 9.86(double) =
                 161.29974(double)
```

Type Conversion

- Examples

```
13.0f * 4 =          \\52.0 (float)
7.0 * (3 / 7) =      \\0.0 (double)
20 / (10 / 6) =      \\20 (int)
20 / (10.0 / 6) =    \\12 (double)
```

Type Conversion

- 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
- 2.0 + 5
- 2.0 + 5.0
- 7.0

Type Conversion

- 5 % 4 * 3.0 + true * x++
- ((5 % 4) * 3.0) + (true * (x++))
- (1 * 3.0) + (true * (x++))
- (1.0 * 3.0) + (true * (x++))
- 3.0 + (true * (x++))
- 3.0 + (true * 1)
- 3.0 + (1 * 1)
- 3.0 + 1
- 3.0 + 1.0
- 4.0

Type Conversion

- -3 - 4u + 8.0
- (-3 - 4u) + 8.0
- (4294967293u - 4u) + 8.0
- 4294967289u + 8.0
- 4294967289.0 + 8.0
- 4294967297.0

Agenda

- HW #2 feedback
- Scopes
- Loops
- Float point numbers
- **Debugging**

Errors in Code…

- Problem:
  Code does not work…

- Question:
  What should I do?

General Hints

- Find infinite loops using test-outputs
  - e.g. `std::cout << "Hi";`

General Hints

- Find infinite loops using test-outputs
  - e.g. `std::cout << "Hi";`

- Output variables using `std::cout`

Example

Problem: This code **does not stop**…

```
#include <iostream>

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;
}
```

Example

Problem: This code **does not stop**…

Find **infinite loops** using **test-outputs**

```
    return 0;
}
```

Example

Problem: This code **does not stop**…

```
#include <iostream>

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;
}
```

Example

Problem: This code **does not stop**…

Output

```
#include <iostream>

int main () {
    const int n = 6;
    std::cout << "Hi!";
    // 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;
}
```

Example

Problem: This code **does not stop**…

NO Output

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i < 13; ++i)
        prod *= n;
    std::cout << "Hi!";
    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Example

Problem: This code **does not stop**…

Problem here

```
#include <iostream>

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;
}
```

Example

Problem: This code **does not stop**…

```
#include <iostream>

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;
}
```

Note:

Condition **only candidate**

Example

Prob

Problem: **Evaluation**

```
1 <= i < 13
```

```
(1 <= i) < 13
```

```
true < 13
   1 < 13
    true
```

```
false < 13
    0 < 13
     true
```

n only
date

}

Example

Prob

WRONG: `1 <= i < 13`

RIGHT: `1 <= i && i < 13`

n only
date

}

Example

This works!

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i)
        prod *= n;

    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Example

- Now program finishes.

Example

- Now program finishes.

- But **no output**.

Example

Problem: **no output**

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i)
        prod *= n;

    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Example

Problem: **no output**

Output variables using
`std::cout`

```
    return 0;
}
```

Example

Problem: **no output**

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i)
        prod *= n;

    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Example

Problem: **no output**

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i)
        prod *= n;
    std::cout << prod << "\n";
    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Output:

-2118184960

Example

Problem:

Why **negative?** …

```
int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i)
        prod *= n;
    std::cout << prod << "\n";
    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Output:

-2118184960

Example

Problem: **no output**

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i)
        prod *= n;

    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Problem here

Example

Problem: **no output**

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i){
        prod *= n;
        std::cout << prod << "\n";
    }
    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Output `prod` in every iteration

Example

Problem: **no output**

```
#include <iostream>

int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i && i < 13; ++i){
        prod *= n;
        std::cout << prod << "\n";
    }
    // Output stars
    for (int i = 1; i < prod; ++i)
        std::cout << "*";

    std::cout << "\n";

    return 0;
}
```

Output:

```
6
36
216
1296
7776
46656
279936
1679616
10077696
60466176
362797056
-2118184960
```

Example

Problem: **no output**

Problem: **Overflow**

6^{12} exceeds `int`.

```
    return 0;
}
```

```
ut:
6
76
362797056
-2118184960
```