break

`break` – Explanation

- Goal:
  - **Stop loop immediately**…
  - … and continue from after the loop.

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
for (int i
    int inp
    std::cin
    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`

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

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

// How many inputs (out                    of 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";
```

Note:

`i` and `input` are gone

Output:

```
Number of divisors: 0
```

continue

`continue` - Explanation

- Goal:
  - **Skip** to the **next iteration** right away.

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`

```
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";
```

a:	18

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
for (int i
    int inp
    std::cin
    if (input == 0)
        continue;
    else if (a % input == 0)
        ++n;
}

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

t of 5) are divisors of a?

; ++i) {

input;

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";
```

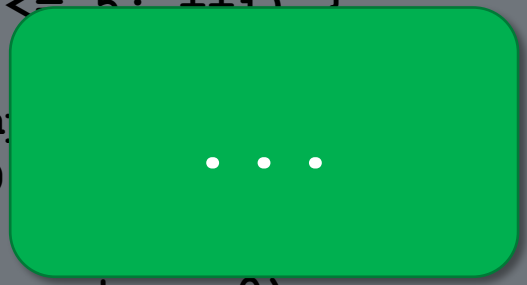
Example – `continue`

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

// How many inputs (out of 5) are divisors of a?
for (int i = 1; i <= 5; ++i) {
    int input;
    std::cin >> in
    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`

Contrast

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.