C++ Loops and Vectors

Fabian Lyck

30 October 2021

Swiss Olympiad in Informatics

General Printing

Instead of:

```
int a = 5;
string b = "blub"
print(a, b);
//prints "5 blub"
```

Use:

```
int a = 5;
string b = "blub"
cout << a << " " << b << "\n";
//prints "5 blub"
```

Loops

Repeated Check

`if` checks a condition once:

```
int n = 20;
if (n % 2 == 0) {
    n = n / 2;
}
cout << n << "\n"; // prints "10"
if (n % 2 == 0) {
    n = n / 2;
}
cout << n << "\n"; // prints "5"
if (n % 2 == 0) {
    n = n / 2;
}
cout << n << "\n"; // prints "5"
```

Repeated Check

`while` checks a condition until it fails:

```
int n = 20;
while (n % 2 == 0) {
    n = n / 2;
}
cout << n << "\n"; // prints "5"
```

Repeated Check

What will this code print?

```
int n = 20;
while (n % 2 == 0) {
    cout << n << "\n"; // prints ??
    n = n / 2;
}
```

Application: Repeating patterns

Repeat a piece of code multiple times:

```
cout << "The␣dates␣of␣October␣are:" << "\n";
cout << "October␣" << 1 << "\n";
cout << "October␣" << 2 << "\n";
cout << "October␣" << 3 << "\n";
// ...
cout << "October␣" << 29 << "\n";
cout << "October␣" << 30 << "\n";
cout << "October␣" << 31 << "\n";
```

Application: Repeating patterns

Repeat a piece of code multiple times:

```
cout << "The dates of October are:" << "\n";
int i = 1;
while (i <= 31) {
    cout << "October " << i << "\n";
    i = i + 1;
}
```

Application: Repeating patterns

Countdown from 10:

```
cout << "Counting down:" << "\n";
int x = 10;
while (x >= 0) {
    cout << x << "\n";
    x = x - 1;
}
```

What is the output?

What is the final value of `x`?

For Loops

`for` is a more compact way of counting:

```
cout << "The dates of October are:" << "\n";
int i = 1; // 1. Initialization
while (i <= 31) {  // 2. Check
    cout << "October " << i << "\n";
    i = i + 1; // 3. Increment
}
```

For Loops

`for` is a more compact way of counting:

```
cout << "The dates of October are:" << "\n";
int i = 1; // 1. Initialization
while (i <= 31) {  // 2. Check
    cout << "October " << i << "\n";
    i = i + 1; // 3. Increment
}
```

This does the same:

```
cout << "The dates of October are:" << "\n";
//  1.          2.        3.
for(int i = 1; i <= 31; i = i + 1) {
    cout << "October", i << "\n";
}
```

For Loops

Countdown from 10:

```
cout << "Counting down:" << "\n";
int x = 10; // 1. Initialization
while (x >= 0) { // 2. Check
    cout << x << "\n";
    x = x - 1; // 3. Increment
}
```

For Loops

Countdown from 10:

```
cout << "Counting down:" << "\n";
int x = 10; // 1. Initialization
while (x >= 0) { // 2. Check
    cout << x << "\n";
    x = x - 1; // 3. Increment
}
```

Using for loop:

```
cout << "Counting down:" << "\n";
//  1.          2.      3.
for(int x = 10; x >= 0; x = x - 1) {
    cout << x << "\n";
}
```

Two dimensional loops

Let's say we want to print multiples of each number, e.g.:

```
0  1  2  3  4  5
0  2  4  6  8  10
0  3  6  9  12  15
```

Two dimensional loops

Let's say we want to print multiples of each number, e.g.:

```
0 1 2 3 4 5
0 2 4 6 8 10
0 3 6 9 12 15
```

We can do this using for loops:

```
for(int i = 0; i <= 5; i = i + 1) {
    cout << i << " ";
}
cout << "\n";
for(int i = 0; i <= 5; i = i + 1) {
    cout << 2 * i << " ";
}
cout << "\n";
for(int i = 0; i <= 5; i = i + 1) {
    cout << 3 * i << " ";
}
cout << "\n";
```

Two dimensional loops

Can we write this more compact using a loop?

```
for(int i = 0; i <= 5; i = i + 1) {
    cout << i << " ";
}
cout << "\n";
for(int i = 0; i <= 5; i = i + 1) {
    cout << 2 * i << " ";
}
cout << "\n";
for(int i = 0; i <= 5; i = i + 1) {
    cout << 3 * i << " ";
}
cout << "\n";
```

Two dimensional loops

Can we write this more compact using a loop?

```
for(int i = 0; i <= 5; i = i + 1) {
    cout << i << " ";
}
cout << "\n";
for(int i = 0; i <= 5; i = i + 1) {
    cout << 2 * i << " ";
}
cout << "\n";
for(int i = 0; i <= 5; i = i + 1) {
    cout << 3 * i << " ";
}
cout << "\n";
```

Yes! Use a second loop:

```
for(int line = 0; line <= 2; line = line + 1) {
    for(int i = 0; i <= 5; i = i + 1) {
        cout << line * i << " ";
    }
    cout << "\n";
}
```

Containers

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```

0

```
fibonacci.push_back(0);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
fibonacci.push_back(1);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
fibonacci.push_back(1);
fibonacci.push_back(2);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
fibonacci.push_back(1);
fibonacci.push_back(2);
fibonacci.push_back(3);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
fibonacci.push_back(1);
fibonacci.push_back(2);
fibonacci.push_back(3);
fibonacci.push_back(5);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
fibonacci.push_back(1);
fibonacci.push_back(2);
fibonacci.push_back(3);
fibonacci.push_back(5);
fibonacci.push_back(8);
```

Vector

Define a vector to hold multiple values:

```
vector<int> fibonacci;
```


```
fibonacci.push_back(0);
fibonacci.push_back(1);
fibonacci.push_back(1);
fibonacci.push_back(2);
fibonacci.push_back(3);
fibonacci.push_back(5);
fibonacci.push_back(8);
fibonacci.pop_back();
```

Vector - Overview

A vector is defined using its type and optionally size:

```
vector<int> fibonacci; // An empty vector of integer numbers
vector<string> food(3); // A vector of size 3 that stores strings
```

Vector - Overview

A vector is defined using its type and optionally size:

```
vector<int> fibonacci; // An empty vector of integer numbers
vector<string> food(3); // A vector of size 3 that stores strings
```

Elements can be accessed at any position:

```
food[0] = "Pancakes"; // 0 indicates the first element
food[1] = "Muffins";
food[2] = "Cookies";
cout << food.size() << "\n"; // Prints "3", the size of food
```

Vector - Overview

A vector is defined using its type and optionally size:

```
vector<int> fibonacci; // An empty vector of integer numbers
vector<string> food(3); // A vector of size 3 that stores strings
```

Elements can be accessed at any position:

```
food[0] = "Pancakes"; // 0 indicates the first element
food[1] = "Muffins";
food[2] = "Cookies";
cout << food.size() << "\n"; // Prints "3", the size of food
```

They can be added or removed at the end:

```
food.push_back("Ruebli"); // Add "Ruebli" to the end of the food vector
cout << food[3] << "\n"; // Prints "Ruebli", the fourth element of food
cout << food[food.size() - 1] << "\n";
// Prints "Ruebli", the last element of food
food.pop_back(); // Remove "Ruebli"
```

Vector - Input/Output

Read N numbers:

```
int N = read_int();
vector<int> numbers;
for (int i = 0; i < N; i = i + 1) {
    int new_number = read_int();
    numbers.push_back(new_number);
}
```

Vector - Input/Output

Read N numbers:

```
int N = read_int();
vector<int> numbers;
for (int i = 0; i < N; i = i + 1) {
    int new_number = read_int();
    numbers.push_back(new_number);
}
```

Print a vector of numbers:

```
for (int i = 0; i < numbers.size(); i = i + 1) {
    cout << numbers[i] << " ";
}
cout << "\n";
```

Vector - Fibonacci

Calculate the first N Fibonacci numbers:

```
int N = read_int();
vector<int> fibonacci;
fibonacci.push_back(0);
fibonacci.push_back(1);
for (int i = 2; i < N; i = i + 1) {
    numbers.push_back(numbers[i - 2] + numbers[i - 1]);
}
```

Vector - Two Dimensions

Let's store the multiples of every number in a vector

```
0 1 2 3 4  5
0 2 4 6 8  10
0 3 6 9 12 15
```

We can do this using two dimensional vectors:

```
vector<vector<int>> table(3, vector<int>(6));
for(int line = 0; line < table.size(); line = line + 1) {
    for(int i = 0; i < table[line].size(); i = i + 1) {
        table[line][i] = line * i;
    }
}
```