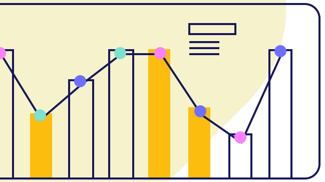
Data Structures & Algorithms

Lab 1



Installing C++ on Visual Studio

1. Go to the official Visual Studio website and download **Visual Studio Community**

(Choose the version Windows or Mac)

2. Run the Installer

Open the downloaded installer, you'll see a **workload selection screen**.

3. Select Workload for C++

•To program in C++, select: **"Desktop development with C++"**

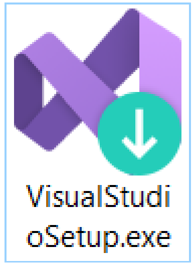
•Click **Install** and **Wait for Installation to Finish**

4. Open VS and Create a C++ Project

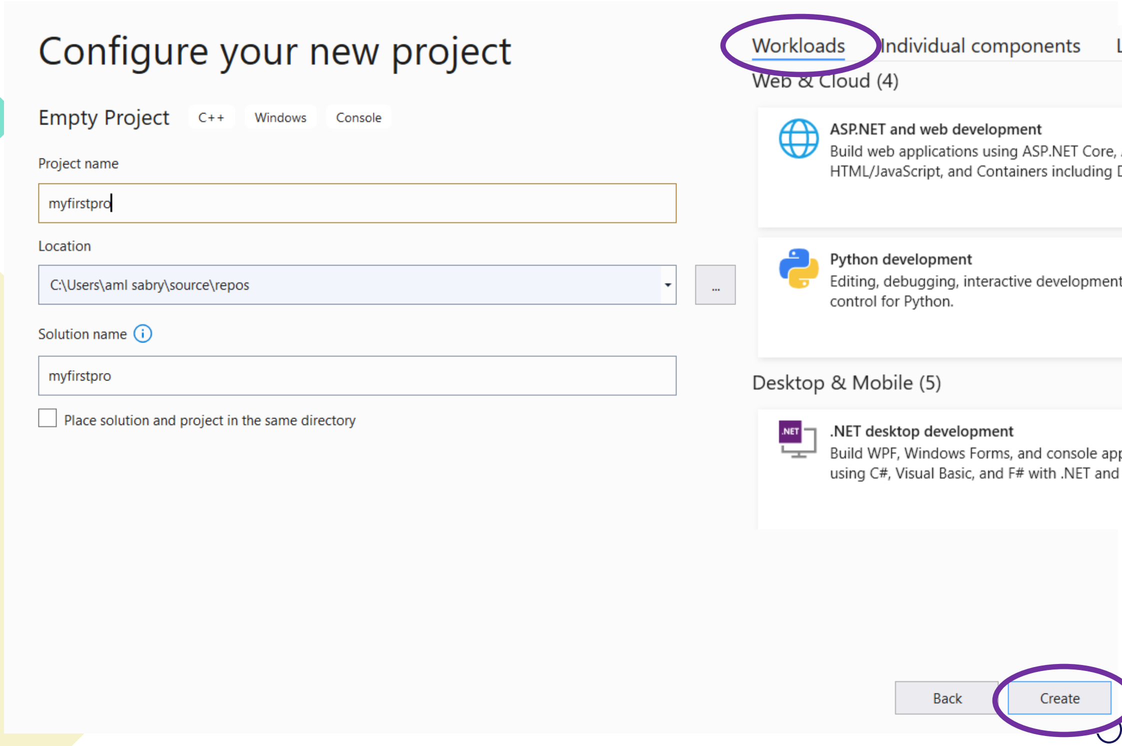
•Click **Create a new project**.

•Choose **Console App** → **C++**.

•Enter your project name and location → Click **Create**.



Installing C++ on Visual Studio



Configure your new project

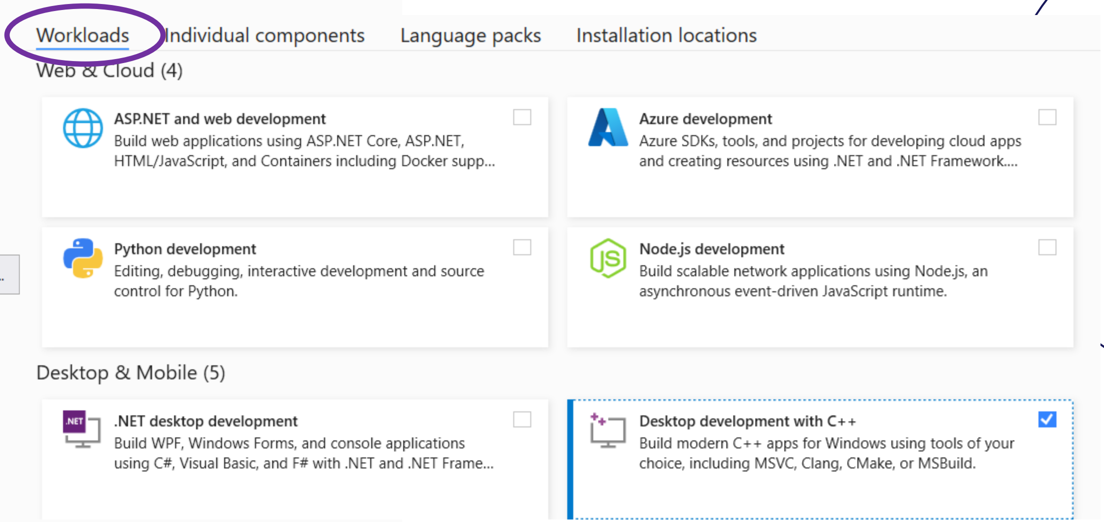
Empty Project C++ Windows Console

Project name: myfirstpro

Location: C:\Users\aml sabry\source\repos

Solution name: myfirstpro

☐ Place solution and project in the same directory



Web & Cloud (4)

- ☐ **ASP.NET and web development** Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
- ☐ **Azure development** Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework....
- ☐ **Python development** Editing, debugging, interactive development and source control for Python.
- ☐ **Node.js development** Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

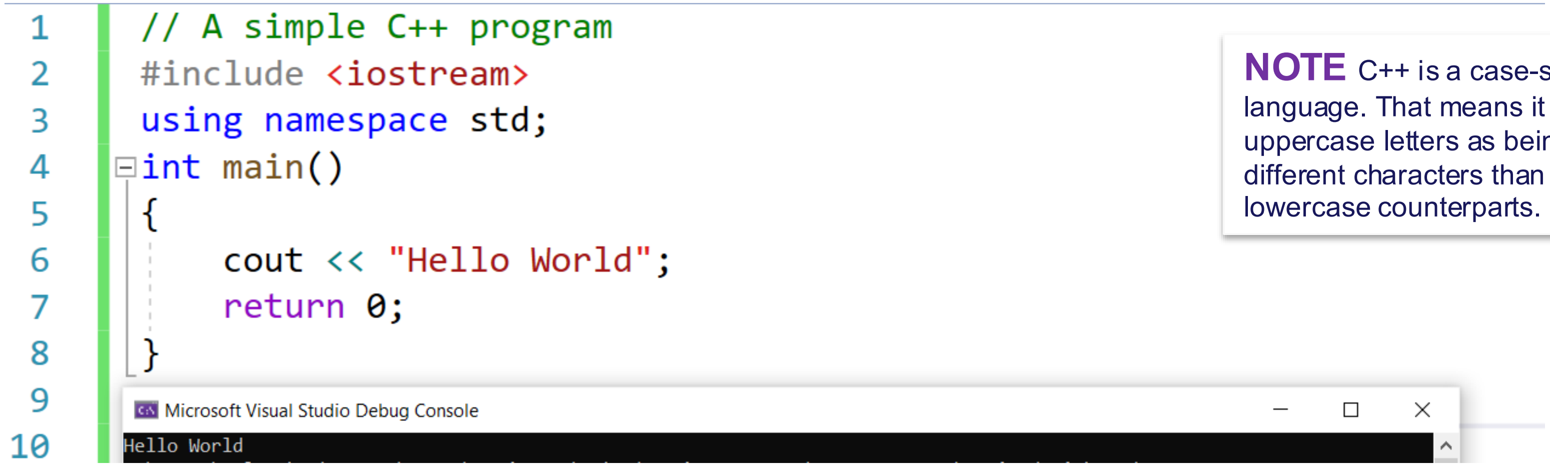
- ☐ **.NET desktop development** Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
- ✓ **Desktop development with C++** Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

Syntax in C++

Header File: #include <iostream> adds input/output objects (cin, cout, etc.)

Namespace Declaration: using namespace std; allows direct use of standard names like "cout" without std::

Comments: // for single line, /*....*/ for multi-line are ignored by the compiler.



```
// A simple C++ program
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World";
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
Hello World
```

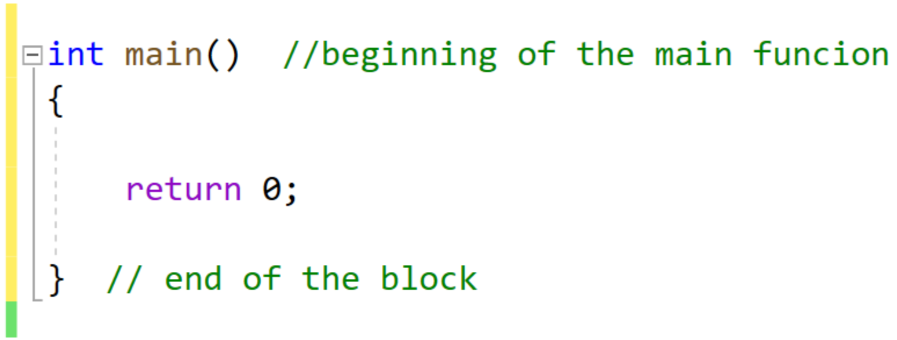
NOTE C++ is a case-sensitive language. That means it regards uppercase letters as being entirely different characters than their lowercase counterparts.

main() Function in C++

Main Function: every C++ program must have a function called main. It is the starting point of the program. If you are ever reading someone else's C++ program and want to find where it starts, just look for the function named main.

int stands for "integer." It indicates that the function sends an integer value back to the operating system when it is finished executing.

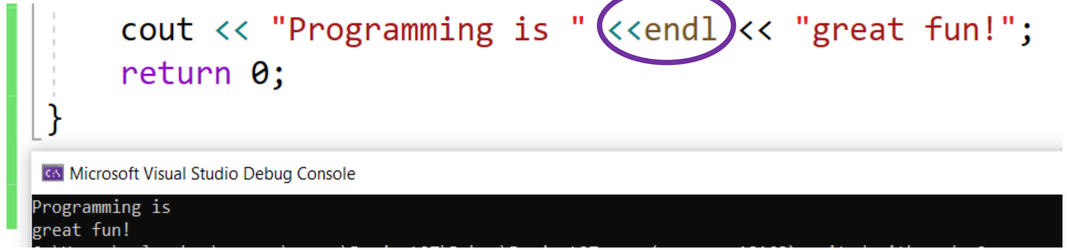
Return: The return 0; statement terminates the **main() function** and indicates that the program executed successfully.



```
int main()  //beginning of the main funcion
{

    return 0;

}  // end of the block
```

The cout << object

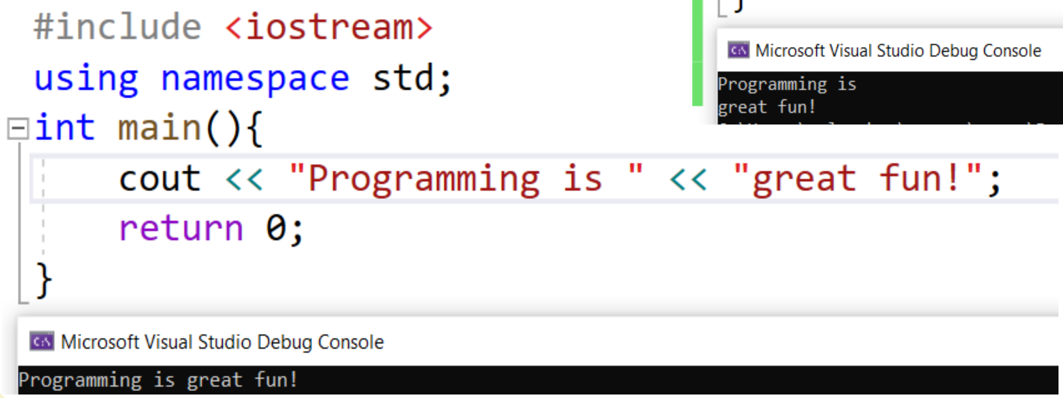
- The **cout** object with the **<<** operator is used to output values and print text.
- To instruct cout to start a new line, send cout a stream manipulator called endl (which is pronounced “end-line” or “end-L”).
- **NOTE:** The last character in endl is the lowercase letter L, not the number one.



```
    cout << "Programming is " <<endl << "great fun!";
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
Programming is
great fun!
```



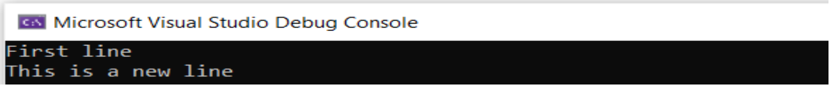
```
#include <iostream>
using namespace std;
int main(){
    cout << "Programming is " << "great fun!";
    return 0;
}
```

Microsoft Visual Studio Debug Console

```
Programming is great fun!
```

The cout << object

```
cout << "First line \n";
cout << "This is a new line";
```


Microsoft Visual Studio Debug Console

```
First line
This is a new line
```

Escape Sequence	Name	Description
`\n`	Newline	Causes the cursor to go to the next line for subsequent printing.
`\t`	Horizontal tab	Causes the cursor to skip over to the next tab stop.
`\a`	Alarm	Causes the computer to beep.
`\b`	Backspace	Causes the cursor to back up, or move left one position.
`\r`	Return	Causes the cursor to go to the beginning of the current line, not the next line.
`\\`	Backslash	Causes a backslash to be printed.
`\'`	Single quote	Causes a single quotation mark to be printed.
`\"`	Double quote	Causes a double quotation mark to be printed.

The cin >> object

- **cin** is the standard input object, It reads input from the console.
- The **>>** symbol is the stream extraction operator.

```
int main()
{
     int length, width, area;

     cout << "This program calculates the area of a ";
     cout << "rectangle.\n";
     cout << "What is the length of the rectangle? ";
     cin >> length;
     cout << "What is the width of the rectangle? ";
     cin >> width;
     area = length * width;
     cout << "The area of the rectangle is " << area << ".\n";
     return 0;
}
```

Think of the << and >> operators as arrows that point in the direction that data is flowing.

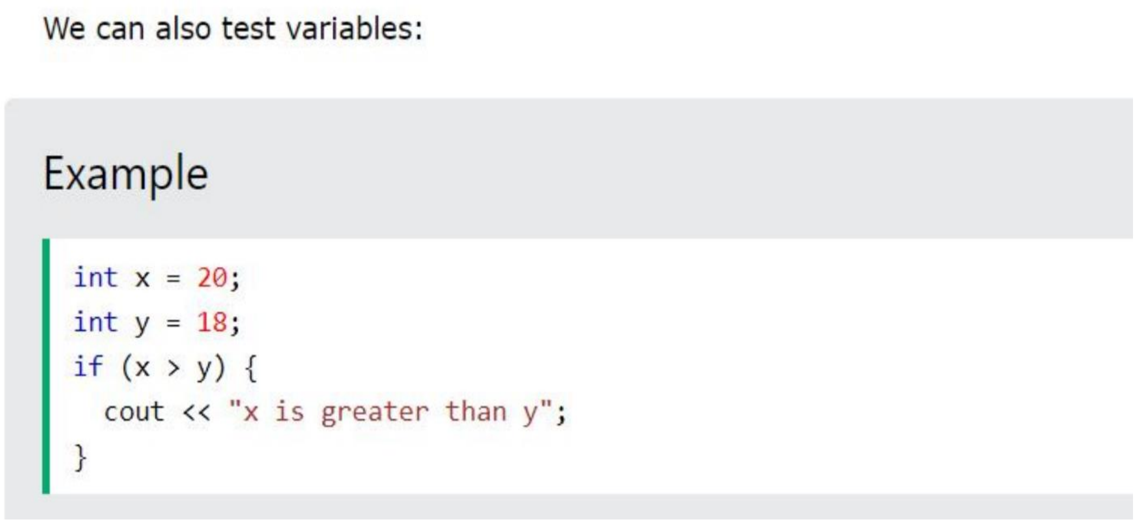
```
cout ← "What is the length of the rectangle? ";
cin → length;
```

Control Statements

- A C control statement redirects the flow of a program in order to execute additional code.
- These statements come in the form of:
- **conditionals (if-else, switch)**
- **loops (for, while, do-while)**.
- Each of them relies on a logical condition that evaluates to a boolean value in order to run one piece of code over another.

The if Statement

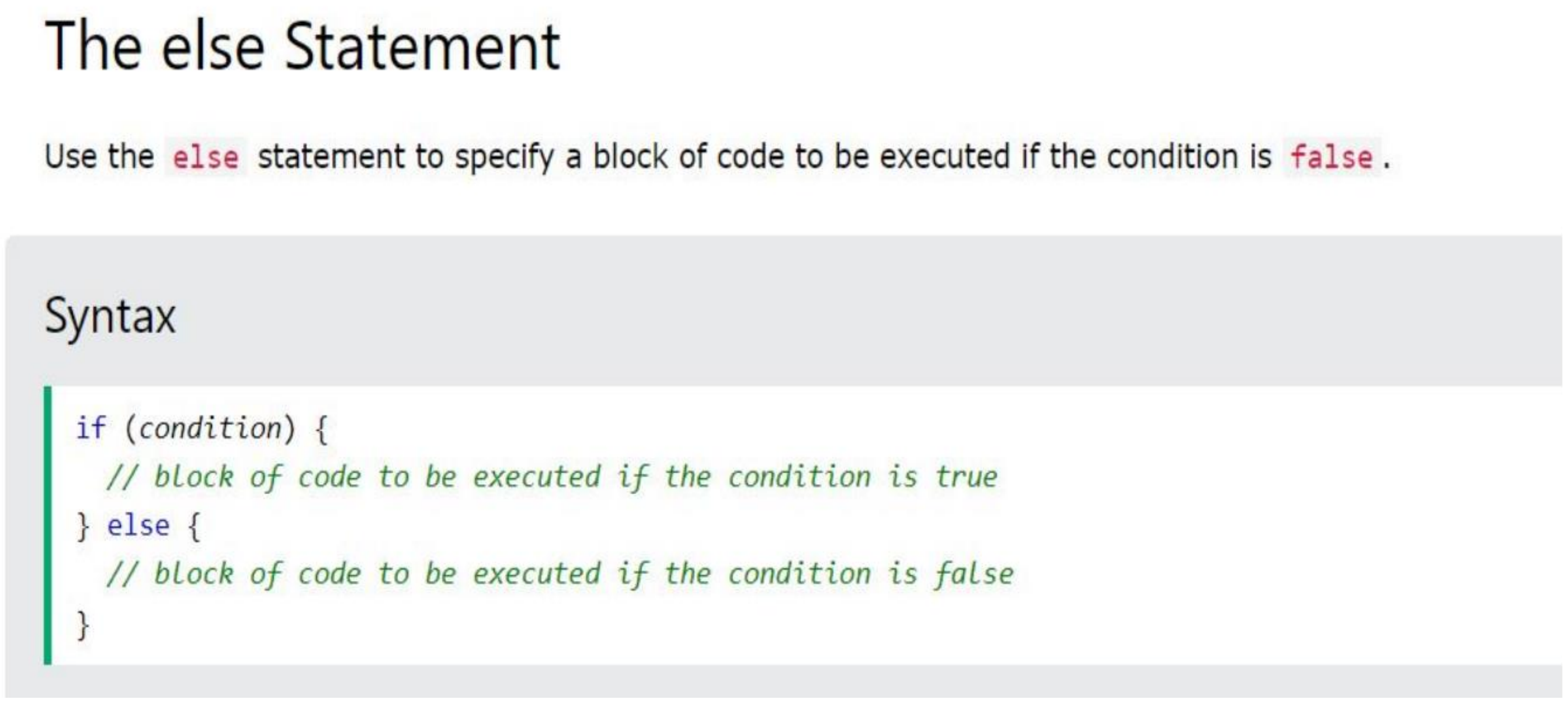
- Use the if statement to specify a block of code to be executed if a condition is true.
- Syntax
- if (condition) { // block of code to be executed if the condition is true}



We can also test variables:

Example

```
int x = 20;
int y = 18;
if (x > y) {
  cout << "x is greater than y";
}
```

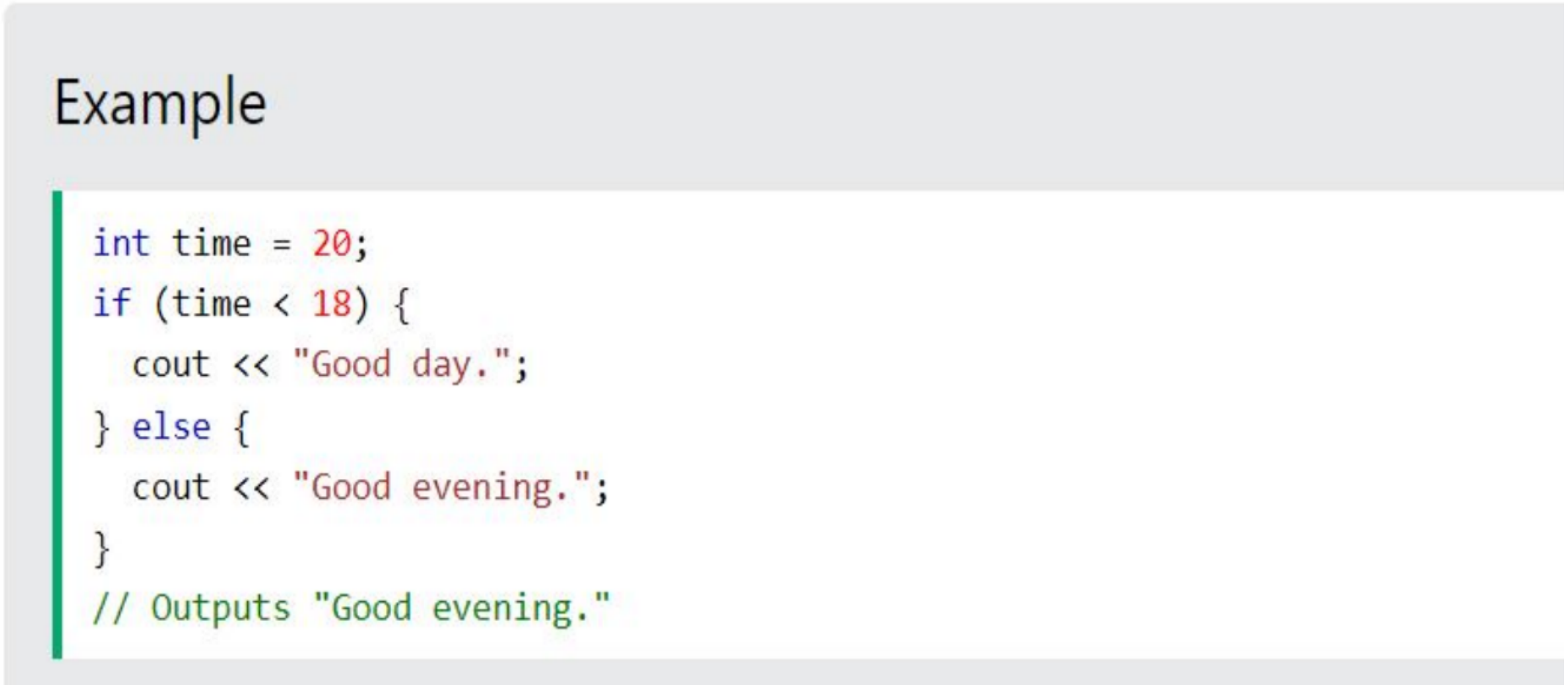


The else Statement

Use the `else` statement to specify a block of code to be executed if the condition is `false`.

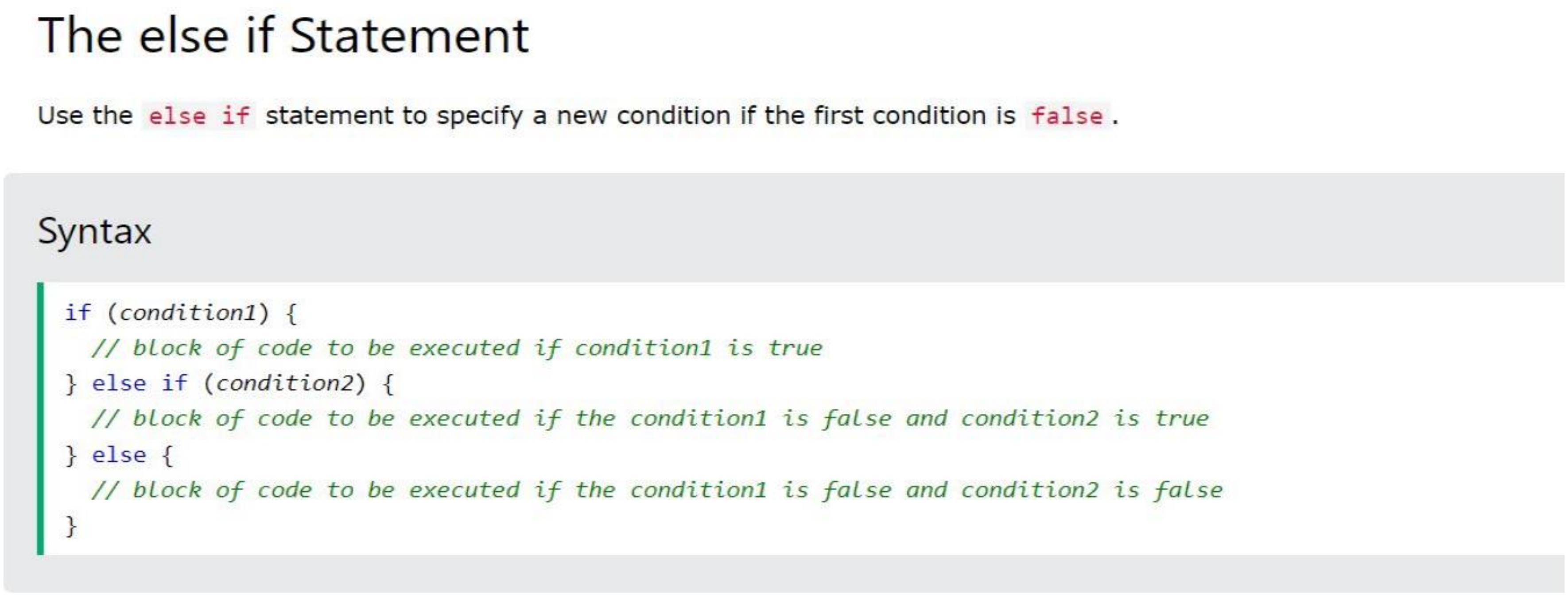
Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```



Example

```
int time = 20;
if (time < 18) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

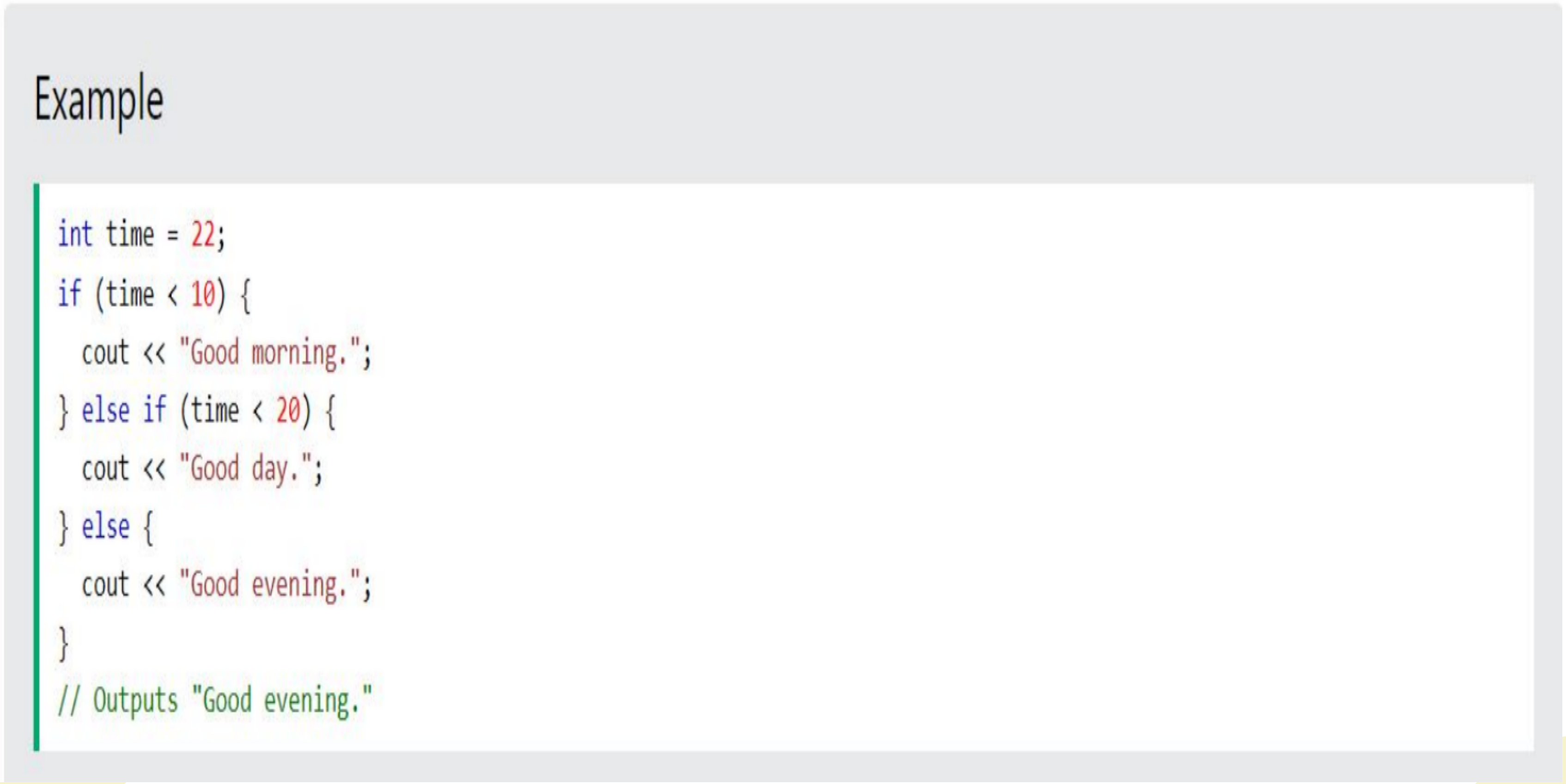


The else if Statement

Use the `else if` statement to specify a new condition if the first condition is `false`.

Syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```



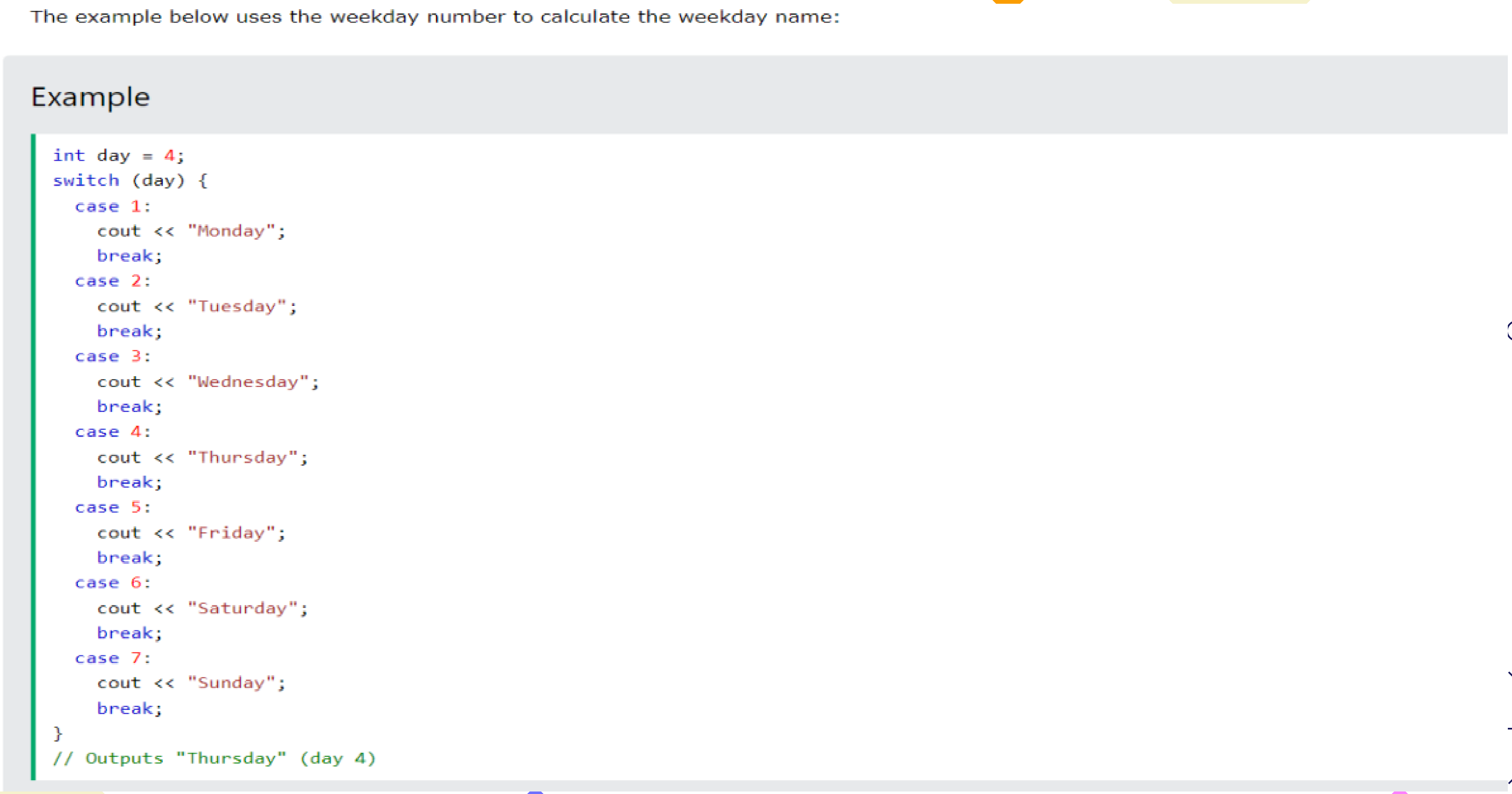
Example

```cpp
int time = 22;
if (time < 10) {
  cout << "Good morning.";
} else if (time < 20) {
  cout << "Good day.";
} else {
  cout << "Good evening.";
}
// Outputs "Good evening."
```

Switch Statement

- Instead of writing many if..else statements, you can use the switch statement.
- The switch statement selects one of many code blocks to be executed:
- **Syntax**
- switch(expression) {

```
  case x:
  // code block
  break;
   case y:
// code block
  break;
 default:
 // code block

}
```



The example below uses the weekday number to calculate the weekday name:

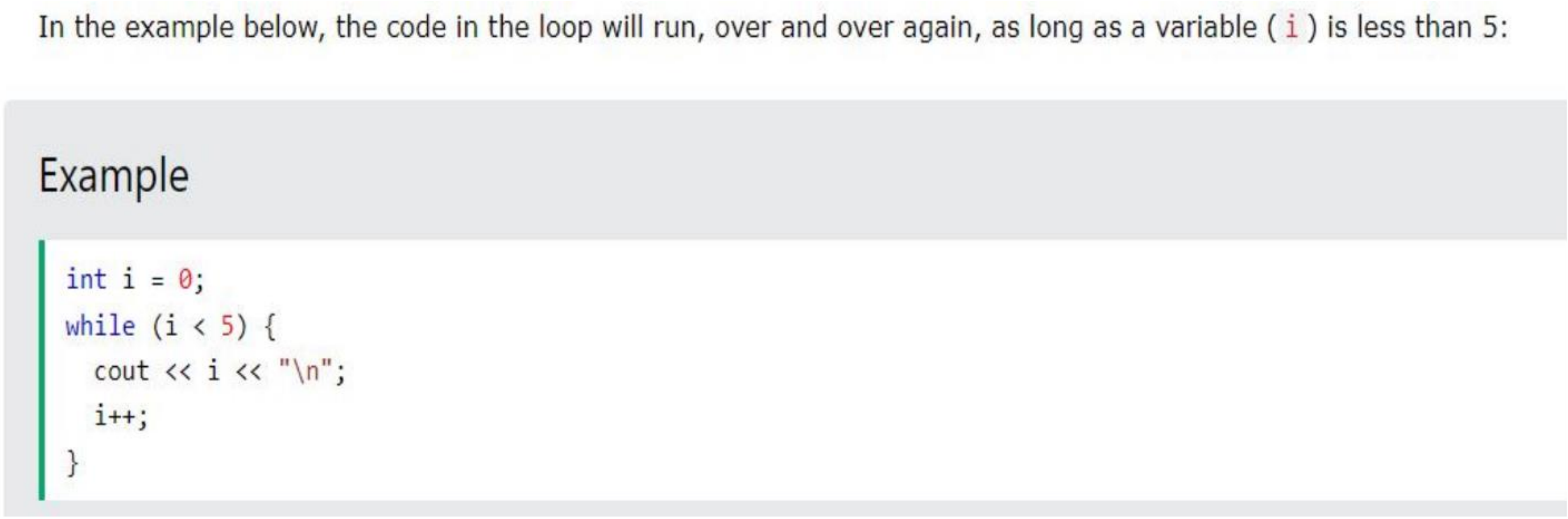
Example

```
int day = 4;
switch (day) {
  case 1:
    cout << "Monday";
    break;
  case 2:
    cout << "Tuesday";
    break;
  case 3:
    cout << "Wednesday";
    break;
  case 4:
    cout << "Thursday";
    break;
  case 5:
    cout << "Friday";
    break;
  case 6:
    cout << "Saturday";
    break;
  case 7:
    cout << "Sunday";
    break;
}
// Outputs "Thursday" (day 4)
```

Loop

•While Loop
•The while loop loops through a block of code as long as a specified condition is true:
•Syntax
•while (condition) {

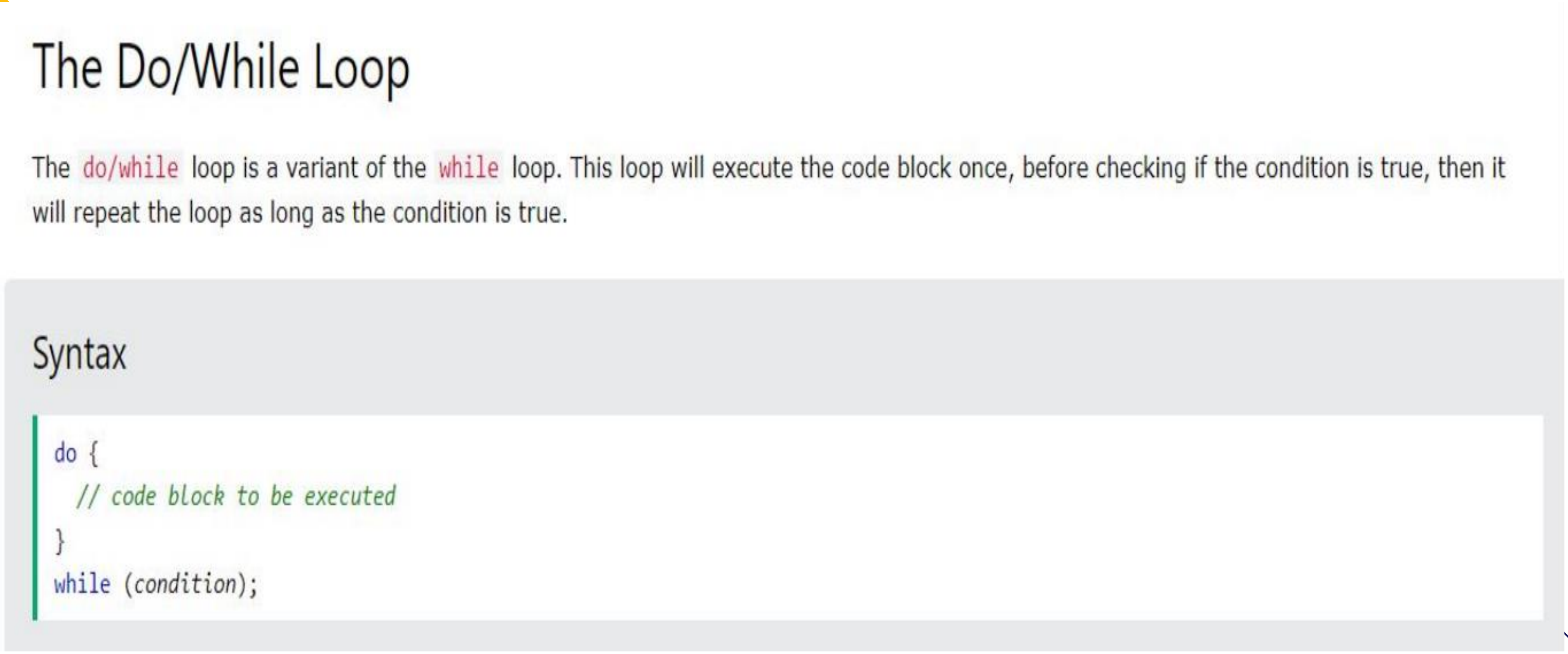
```
  // code block to be executed

}
```



In the example below, the code in the loop will run, over and over again, as long as a variable (i) is less than 5:

Example

```cpp
int i = 0;
while (i < 5) {
  cout << i << "\n";
  i++;
}
```

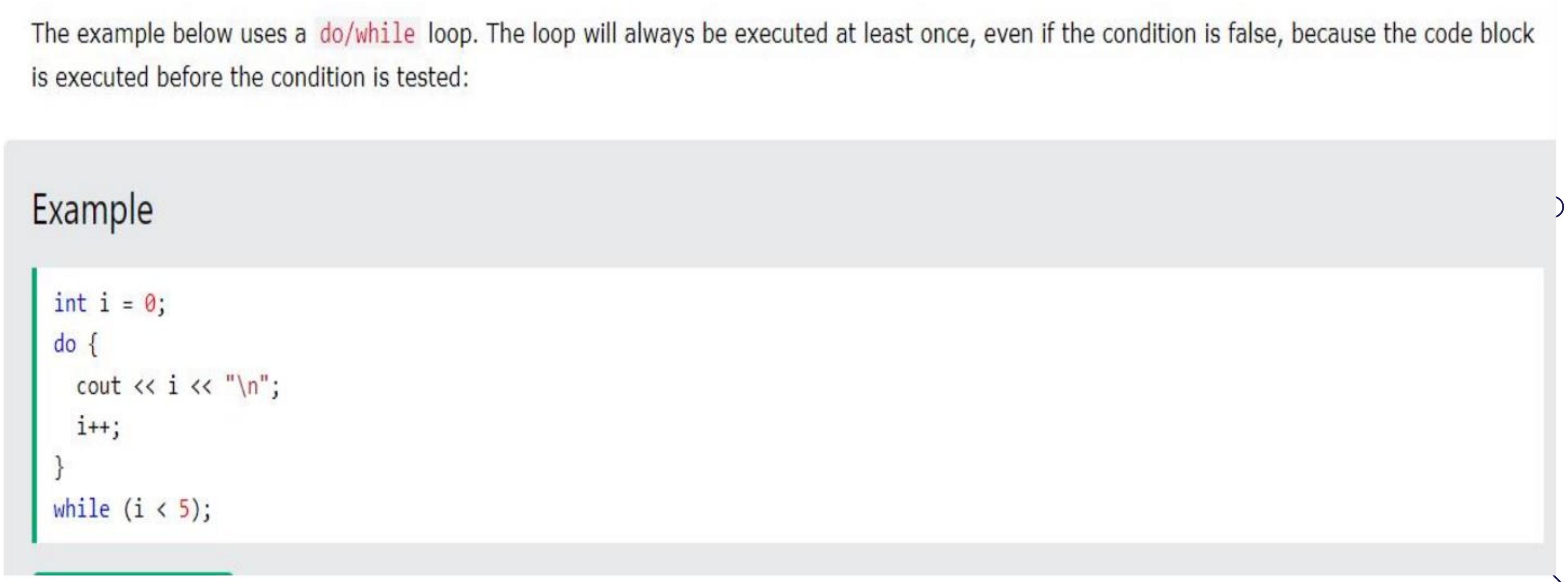


The Do/While Loop

The `do/while` loop is a variant of the `while` loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

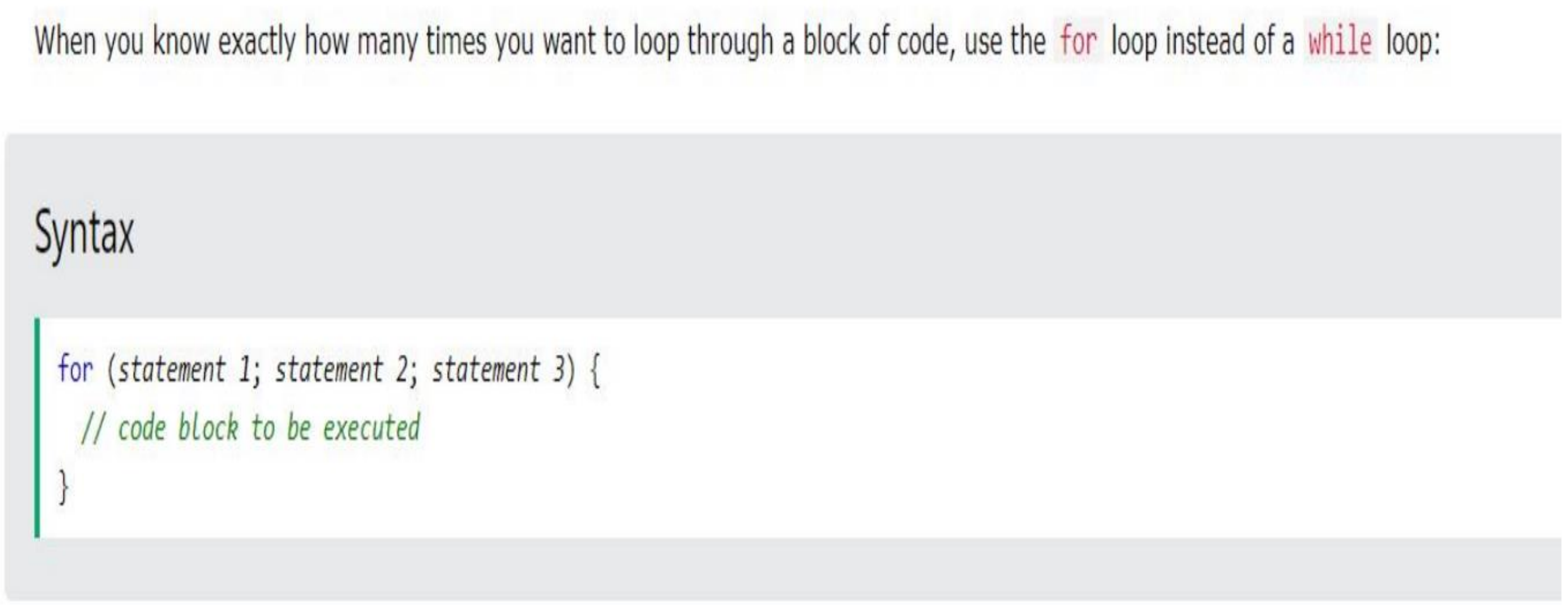


The example below uses a `do/while` loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Example

```
int i = 0;
do {
  cout << i << "\n";
  i++;
}
while (i < 5);
```

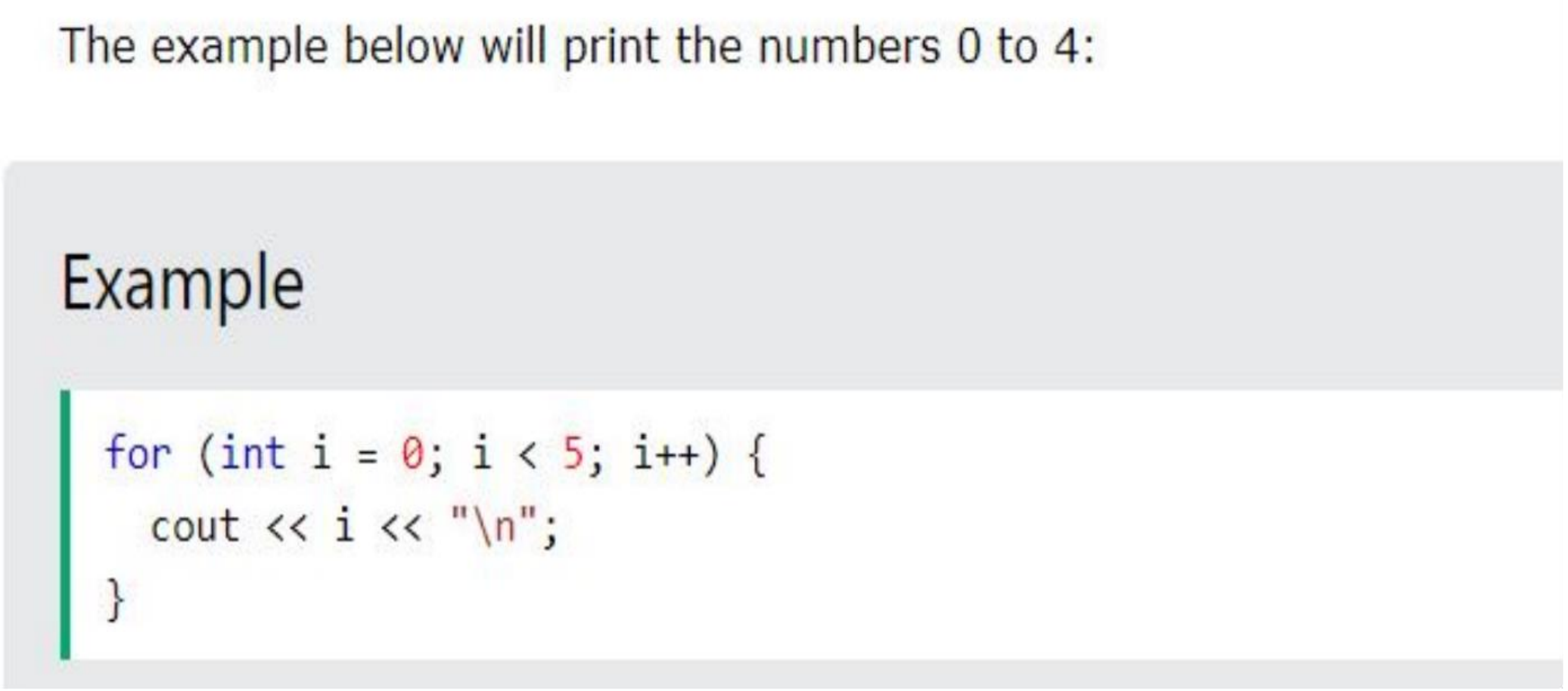
For Loop



When you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop:

Syntax

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```



The example below will print the numbers 0 to 4:

Example

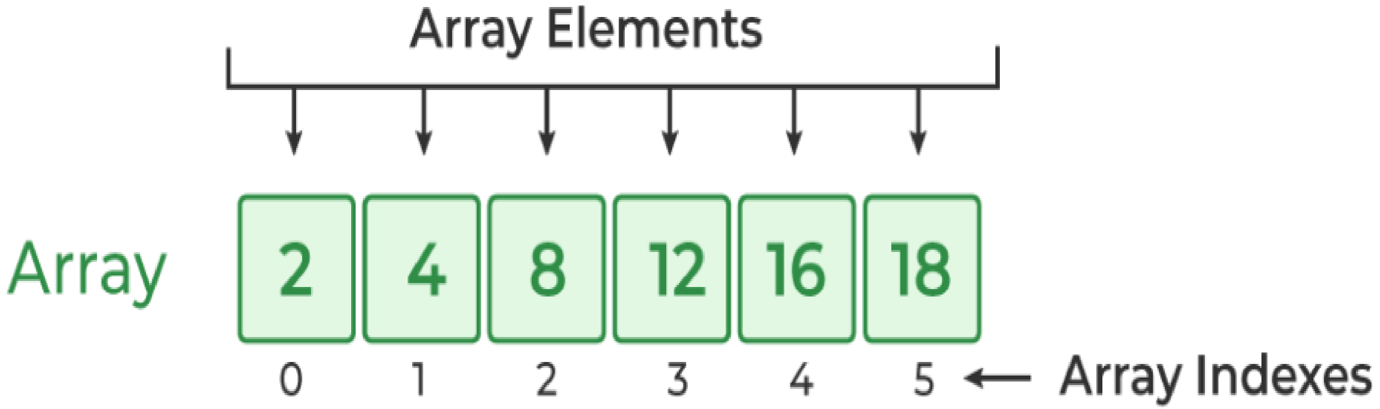
```
for (int i = 0; i < 5; i++) {
  cout << i << "\n";
}
```

Nested Loops

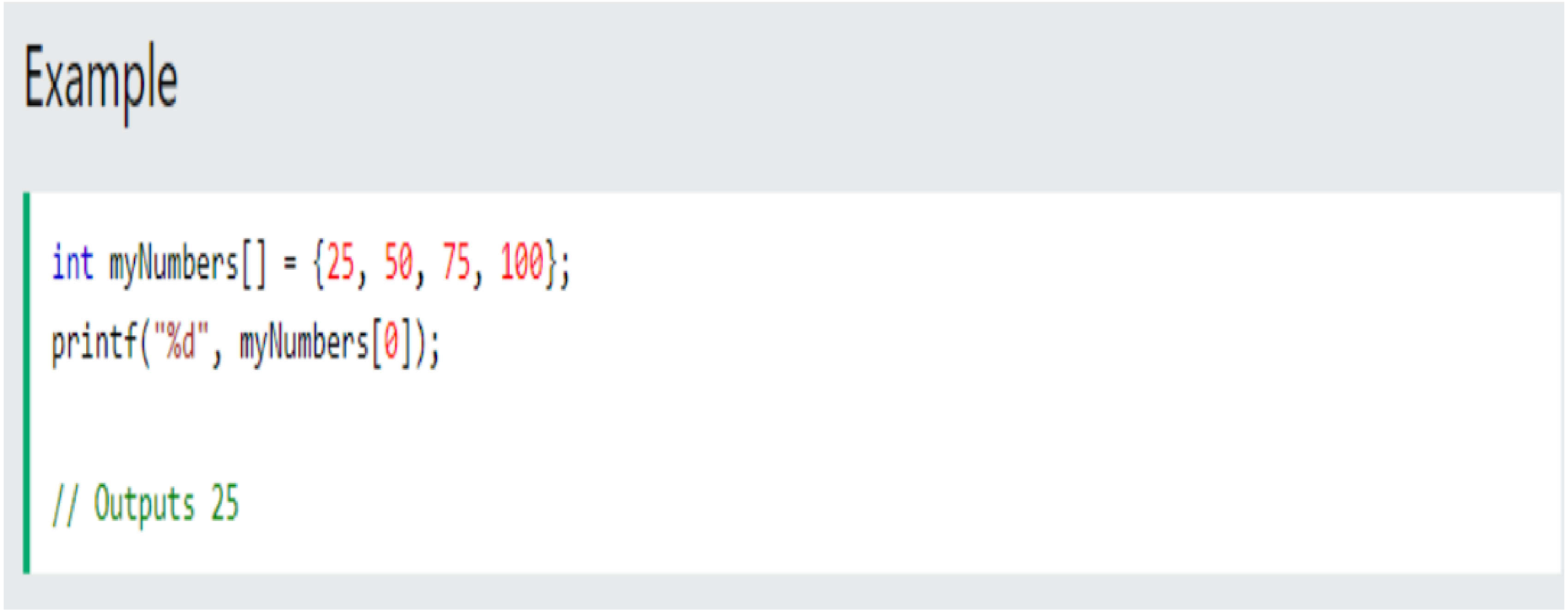
```
int i, j;

for (int i = 1; i <= 9; i++)
 {
    for (int j = 1; j <= 9; j++)
     {
  // Display the product and align properly
          cout << " " << i * j;
     }
       cout << "\n";

}
```

Arrays

Array in C



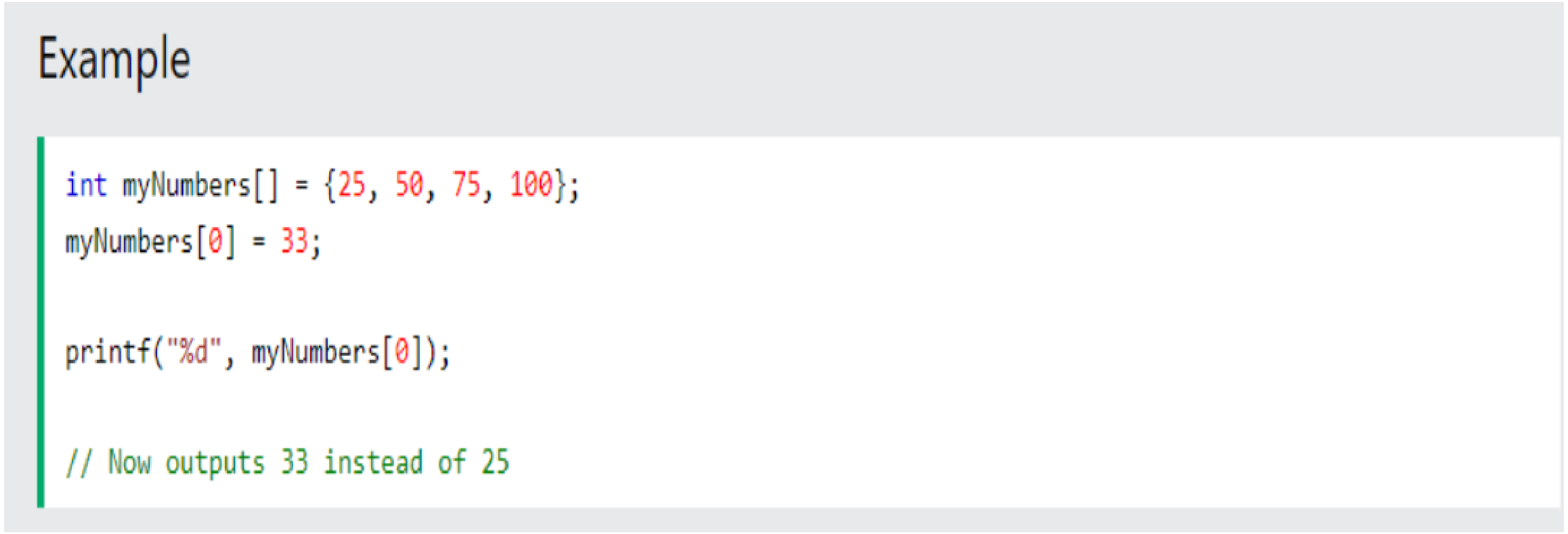
Access the Elements of an Array



Example

```
int myNumbers[] = {25, 50, 75, 100};
printf("%d", myNumbers[0]);


// Outputs 25
```

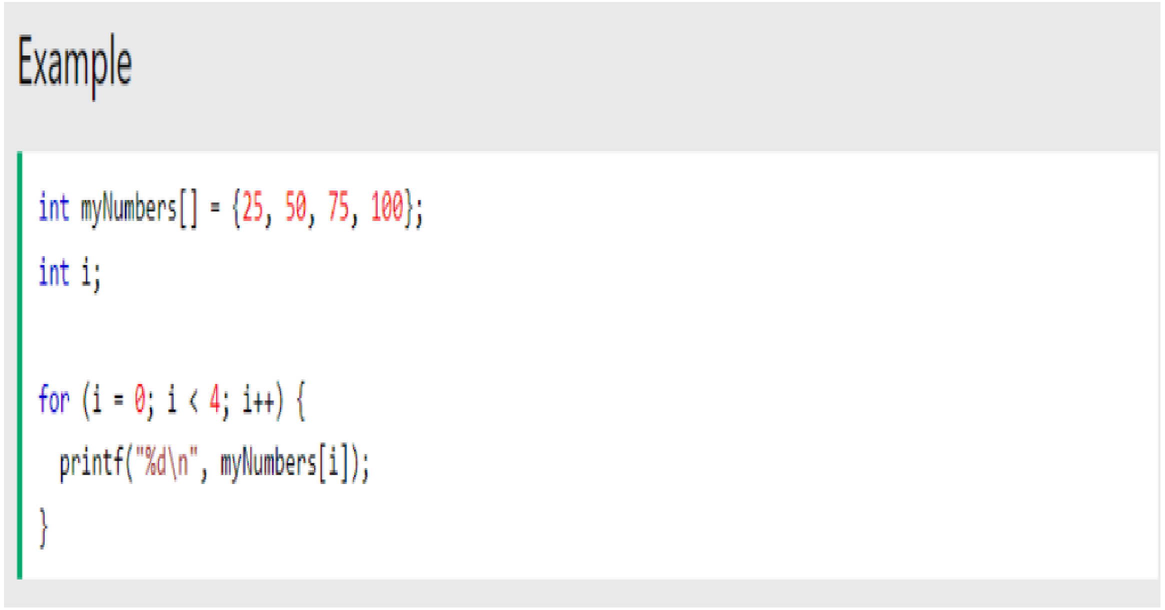
Change an Array Element



Example

```c
int myNumbers[] = {25, 50, 75, 100};
myNumbers[0] = 33;

printf("%d", myNumbers[0]);

// Now outputs 33 instead of 25
```

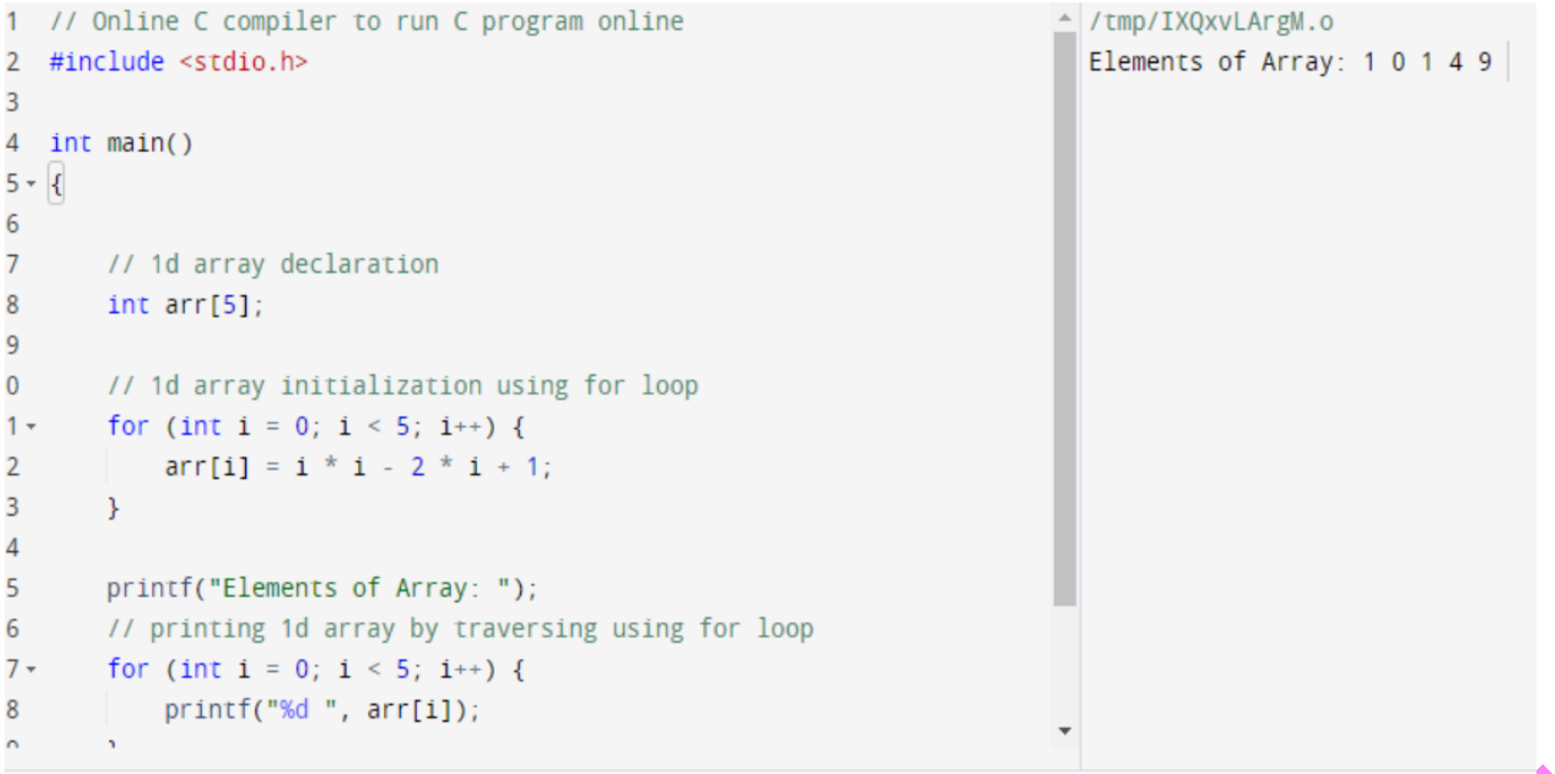
Loop Through an Array



Example

```
int myNumbers[] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

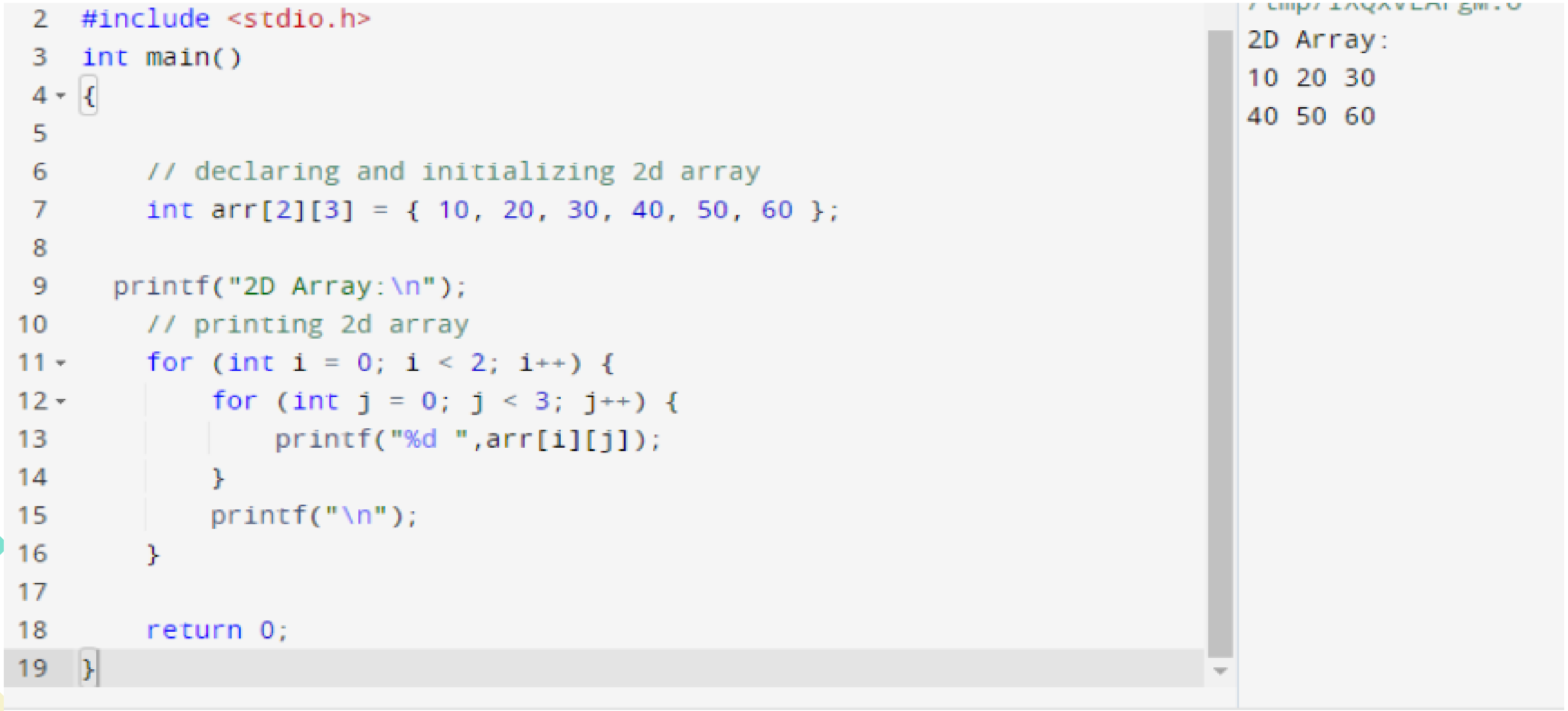
Example of 1D Array



```
// Online C compiler to run C program online
#include <stdio.h>

int main()
{

    // 1d array declaration
    int arr[5];

    // 1d array initialization using for loop
    for (int i = 0; i < 5; i++) {
        arr[i] = i * i - 2 * i + 1;
    }

    printf("Elements of Array: ");
    // printing 1d array by traversing using for loop
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
```

```
/tmp/IXQxvLArgM.o
Elements of Array: 1 0 1 4 9
```

Example of 2D Array



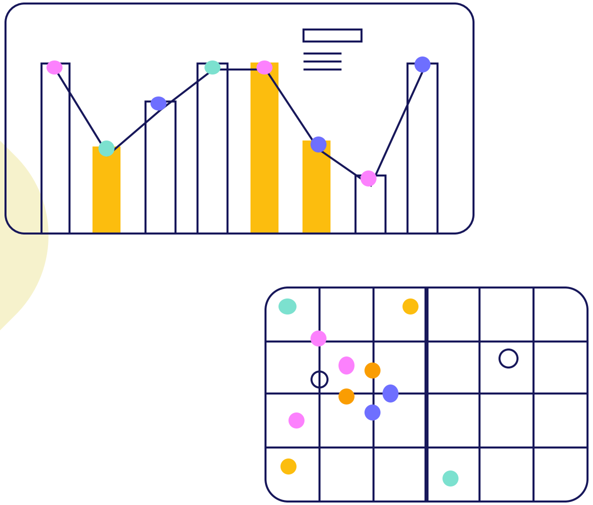
```c
#include <stdio.h>
int main()
{

    // declaring and initializing 2d array
    int arr[2][3] = { 10, 20, 30, 40, 50, 60 };

  printf("2D Array:\n");
    // printing 2d array
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%d ",arr[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
2D Array:
10 20 30
40 50 60
```

Task : Write a program that takes a number of elements from the user and stores them in an array. Then, it counts how many numbers are even and how many are odd.

```
#include <iostream>
using namespace std;

int main() {
int n;

cout << "Enter number of elements: ";
cin >> n;

int arr[n];
cout << "Enter the elements: ";
for (int i = 0; i < n; i++) {
cin >> arr[i];
 }
```

```
int evenCount = 0, oddCount = 0;
for (int i = 0; i < n; i++) {
if (arr[i] % 2 == 0) {
evenCount++;
 } else {
oddCount++;
}
}

// Output the results
cout << "Even numbers count: " << evenCount <<
endl;
cout << "Odd numbers count: " << oddCount <<
endl;
return 0; }
```



Thank You