Software Unit Testing, part 3
C++ - Generic Testing, GTEST

1 Introduction

In part 1 we discussed how Software Unit Testing is different from End-to-End testing. We briefly discussed Test Coverage and why you need it. There was a discussion of how you did automated software testing in the early days using a generic test framework, and an explanation of how automated software testing would be performed using JavaScript with the modern Web testing framework Cypress. In part 2 we discussed how to do Software Unit Testing in the C# language using .NET and the modern xUnit test framework. This article will discuss how to do Software Unit Testing in the C++ language using a generic test framework as well as the modern GTEST framework.

2 C++

C++ was created by Bjarne Stroustrup. [191] The first commercial implementation of C++ was released in 1985. [192]

3 Test Scenario

For our testing example let's use the children's game "FizzBuzz", which is sometimes used as a coding test. FizzBuzz works like this:

Output the numbers from 1 to 100.

1. If the number is not divisible by either 3 or 5, just display the number.
2. If the number is divisible by 3, display the word "FIZZ" instead of the number.
3. If the number is divisible by 5, display the word "BUZZ" instead of the number.
4. If the number is divisible by both 3 and 5, display the word "FIZZBUZZ" instead of the number.

So, our code must meet the four requirements listed above. In order to do so, the code must make several decisions based on the input value. We would therefore need a minimum of 4 test cases to do testing based on the requirements. This function has the following code paths:

path 1 - the input IS NOT divisible by EITHER 3 or 5, return the input value
path 2 - the input IS divisible by BOTH 3 and 5, return "FIZZBUZZ"
path 3 - the input IS divisible by 3 but IS NOT divisible by 5, return "FIZZ"
path 4 - the input IS NOT divisible by 3 but IS divisible by 5, return "BUZZ"

If we test to meet the requirements, we would expect to see 100% path coverage of the main section of the code with the four test cases needed to meet the requirements. We would add a fifth test case to verify that errors are handled correctly.

4 Unit Testing - Generic Application

Let's write a function called ***fizzBuzz*** that we want to use in an application. We want to test this function to make sure it works correctly. Our function would look like this:

```
std::string fizzBuzz( int counter ) {

       //  Input a number between 1 and 100.
       //  If the number is divisible by 3, output "FIZZ".
       //  If the number is divisible by 5, output "BUZZ".
       //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
       //  For all other numbers, output the original number.

              std::string resultStr;       //  declare the variable

               resultStr = std::to_string(counter);                // default - just return the counter

               if(counter % 3 == 0) {         //  divisible by 3
                  resultStr = "FIZZ";
               }

               if(counter % 5 == 0) {       //  divisible by 5
                  resultStr = "BUZZ";
               }

               if( (counter % 3 == 0) && (counter % 5 == 0) ) {   //  divisible by both 3 and  5
                  resultStr =  "FIZZBUZZ";
               }

             return resultStr;


     //  end      std::string fizzBuzz
}
```

Figure 1: C++ code for the fizzBuzz function.

5 C++ Compilers

Various compilers for C++ are available. Some of the choices are as follows:

Platform	Compilers	Architectures
Windows	MSVC, Clang, GCC	x64, x86, arm64, arm
Linux	Clang, GCC	x64, x86, arm64, arm
macOS	Clang, GCC	x64, x86, arm64 [167]

Common compilers that already come preinstalled on some platforms are the GNU Compiler Collection (GCC) on Linux and the Clang tools with Xcode on macOS. [167] [171]

Visual Studio includes a C++ compiler. If you don't have Visual Studio installed on your Windows computer, there are a couple of ways to get C++ working on your computer. You can install the **Windows Subsystem for Linux** [168] [169] [170], or you can install the **MinGW-w64 toolchain** under MSYS2. [167]

This article will focus on using C++ under Windows by installing GCC. If you have installed the Windows Subsystem for Linux, you can check if GCC is already installed. Open a Command Prompt window, or if you have VS Code with the C/C++ extension [167] [172] installed, open a new VS Code terminal window using (Ctrl+`). Use the following command to check for the GCC C++ compiler which is called g++:

```
C:\> g++ --version
g++ (Rev2, Built by MSYS2 project) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The output should show you the compiler version and details. If it is not found, make sure your compiler executable location is in your platform path (%PATH on Windows, $PATH on Linux and macOS) so that Windows or the VS Code C/C++ extension can find it. [177]

If you want to install the MinGW-w64 toolchain, get the latest version of MinGW-w64 via MSYS2 [173] , which provides up-to-date native builds of GCC, MinGW-w64, and other helpful C++ tools and libraries. This will provide you with the necessary tools to compile your code and debug it. [167] The size of the installation varies according to the packages which are installed. A typical base installation of MSYS2 would be around 1.3 GB.

Once you have installed MSYS2, you can use pacman to update or install any software packages that you might need.

```
$ pacman -h
usage: pacman [...]
operations:
pacman {-h --help}
pacman {-V --version}
pacman {-D --database} <options> <package (s)>
pacman {-F --files} [options] [file(s)]
pacman {-Q --query} [options] [package(s)]
pacman {-R --remove} [options] <package(s)>
pacman {-S --sync} [options] [package(s)]
pacman {-T --deptest} [options] [package(s)]
pacman {-U --upgrade} [options] <file(s)>

use 'pacman {-h --help}' with an operation for available options
```

You will need to install [174] [175] [176] the following packages:

GNU Compiler Collection (C,C++,OpenMP) for MinGW-w64
(Total Installed Size: 418.89 MB)

To install the package [177], enter the command

```
pacman -S mingw-w64-ucrt-x86_64-gcc
```

GNU Debugger (mingw-w64)
(Total Installed Size: 339.91 MB)

To install the package [177], enter the command

```
pacman -S mingw-w64-ucrt-x86_64-gdb
```

Google Testing and Mocking Framework (mingw-w64)
(Total Installed Size: 3.34 MB)

To install the package, enter the command

```
pacman -S mingw-w64-ucrt-x86_64-gtest
```

lcov - a front-end for GCC's coverage testing tool gcov
(Total Installed Size: 203.14 MB)

To install the package, enter the command

```
pacman -S mingw-w64-ucrt-x86_64-lcov
```

Add the MSYS2 binary directory (C:\msys64\ucrt64\bin) to the PATH environment variable. [177] You will need to run lcov in an MSYS2 terminal window for the UCRT64 environment if you did not add the MSYS2 binary directory to the PATH environment variable.

The tools should now be ready to use. You can check that the tools are correctly installed and available. Open a Command Prompt and type:

```
C:\> gcc --version
gcc (Rev2, Built by MSYS2 project) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
C:\> g++ --version
g++ (Rev2, Built by MSYS2 project) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
C:\> gdb --version
GNU gdb (GDB) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law. [168]
```

You will need to run lcov in an MSYS2 terminal window for the UCRT64 environment. We won't use lcov in this article.

```
$ lcov -version
lcov: LCOV version 1.16
```

Installing MSYS2 and the modules listed above used around 4 GB of storage space. This is the best option if you want to use the least storage space. The next best option would be to use C++ within VS Code. The basic installation of Visual Studio Code requires around 0.5 GB, while Microsoft C++ (Build Tools for Visual Studio 2022) requires around 6 GB. [164] It can only be run from a developer command prompt. In addition, you must have a valid Visual Studio license. [159] Compare this to the other options.

```
Product                    Typical Installation            Maximum

Visual Studio 2022          20 - 50 GB                     210 GB        [158]

Visual Studio 2019          20 - 50 GB                     210 GB        [160]

Visual Studio 2017          20 - 50 GB                     130 GB        [161] [166]

Visual Studio Code              0.5 GB

      *** Base install. C++ must be installed separately. [162] [163]

VS Code Microsoft C++             6 GB

      *** Build Tools for Visual Studio 2022. [164]

      *** Must run from a Developer Command Prompt. [164]

      *** requires a valid Visual Studio license. [159]
```

Figure 2: Storage Space Required by Visual Studio and Visual Studio Code.

This article will focus on using GCC under MSYS2 since it is the option that requires the least amount of storage space.

6 Setting Up the Project

Create a directory for the project.

mkdir testFizzBuzzCpp

Change directory to the project directory.

cd testFizzBuzzCpp

Create a directory for libraries.

mkdir library

Create a directory for a console app.

mkdir console_app

Create a directory for Google Test.

mkdir gtest

7 Build the Console App

Change directory to the console app directory.

cd console_app

Create the initial scaffolding for a console application by creating two files;

testFizzBuzzCpp.h and ***testFizzBuzzCpp.cpp***

testFizzBuzzCpp.h looks like this:

```
// testFizzBuzzCpp.h: the application header code.
// Additional source code to include.
```

Figure 3: C++ code for testFizzBuzzCpp.h

testFizzBuzzCpp.cpp looks like this:

```
// testFizzBuzzCpp.cpp : the application source code.
//  Defines the entry point for the console application.
//

#include <iostream>
#include <string>
#include "testFizzBuzzCpp.h"

int main(int argc, char* argv[]) {

   if (argc > 1) {
      //  begin   user asked for help
      std::cout << "testFizzBuzzCpp:\n";
      std::cout << "Help is provided here.\n";

      //  end    user asked for help
   }
   else
   {
      //  begin   run the program normally

          std::cout << "testFizzBuzzCpp:\n";
          std::cout << "For help, type testFizzBuzzCpp -h.\n";
          std::cout << "\n";
          std::cout << "testFizzBuzzCpp results:\n";

          //  end     run the program normally
   }

   std::cout << "End of program.\n";

   //  exit the program woth a return value of 0
   return 0;

   //  end       int main
}
```

Figure 4: C++ code for testFizzBuzzCpp.cpp

Make sure that the software builds correctly. Compile and link ***testFizzBuzzCpp.cpp*** to generate an executable file named ***testFizzBuzzCpp.exe*** .

g++ testFizzBuzzCpp.cpp -o testFizzBuzzCpp.exe

Change directory back to the previous directory.

cd ..

8 Build the library

Change directory to the library directory.

cd library

Create the initial scaffolding for a class library by creating two files:

Class1.h and ***Class1.cpp***

Class1.h looks like this:

```
// Class1.h

#pragma once
class Class1
{
   public:
      Class1();
      std::string fizzBuzz(int);
      ~Class1();
};
```

Figure 5: C++ code for Class1.h

Class1.cpp looks like this:

```
// Class1.cpp

#include <string>
#include "Class1.h"


Class1::Class1()
{
}

std::string Class1::fizzBuzz( int counter )
{
          //  Input a number between 1 and 100.
          //  If the number is divisible by 3, output "FIZZ".
          //  If the number is divisible by 5, output "BUZZ".
          //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
          //  For all other numbers, output the original number.

                 std::string resultStr;                         //  declare the variable

                  resultStr = std::to_string(counter);

                  if(counter % 3 == 0) {     //  divisible by 3
                     resultStr = "FIZZ";
                  }

                  if(counter % 5 == 0) {     //  divisible by 5
                     resultStr = "BUZZ";
                  }

                  if( (counter % 3 == 0) && (counter % 5 == 0) ) {     //  divisible by both 3 and  5
                     resultStr =  "FIZZBUZZ";
                  }

                return resultStr;


        //  end      string fizzBuzz
}


Class1::~Class1()
{
}
```

Figure 6: C++ code for Class1.cpp

Make sure that the software builds correctly. Compile and link **Class1.cpp** to create a class library object code file named **Class1.o** .

g++ -c Class1.cpp

Change directory back to the previous directory.

cd ..

9 Modify the Console App

Change directory to the console app directory.

cd console_app

Modify testFizzBuzzCpp.cpp so it looks like this:

```
// testFizzBuzzCpp.cpp : the application source code.

#include <iostream>
#include <string>
#include "testFizzBuzzCpp.h"
#include "../library/Class1.h"

   std::string localFizzBuzz( int counter ) {

          //  Input a number between 1 and 100.
          //  If the number is divisible by 3, output "FIZZ".
          //  If the number is divisible by 5, output "BUZZ".
          //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
          //  For all other numbers, output the original number.

                 std::string resultStr;                         //  declare the variable

                  resultStr = std::to_string(counter);

                  if(counter % 3 == 0) {     //  divisible by 3
                     resultStr = "FIZZ";
                  }

                  if(counter % 5 == 0) {     //  divisible by 5
                     resultStr = "BUZZ";
                  }

                  if( (counter % 3 == 0) && (counter % 5 == 0) ) {     //  divisible by both 3 and  5
                     resultStr =  "FIZZBUZZ";
                  }

                return resultStr;

        //  end      string localFizzBuzz
   }

   void testCase ( int testCaseNumber,
                  int inputValue,
                  std::string expectedResultStr ) {

         std::string resultStr;                //  declare the variable

         Class1 clsObj;     //  instantiate Class1
         resultStr = clsObj.fizzBuzz(inputValue);

         std::cout << "Test case ";
         std::cout << testCaseNumber;
         std::cout << " ";
         std::cout << "fizzBuzz with input of ";
         std::cout << inputValue;
         std::cout << " ";
         std::cout << "expecting ";
         std::cout << expectedResultStr;
         std::cout << " ";
         std::cout << "returned ";
         std::cout << resultStr;
         std::cout << " ";

         if(expectedResultStr == resultStr) {
               std::cout << "PASSED\n";
         }
         else
         {
               std::cout << "FAILED\n";
         }

        //  end       void testCase
   }

  void localTestCase ( int testCaseNumber,
                  int inputValue,
                  std::string expectedResultStr ) {

         std::string resultStr;                //  declare the variable

         resultStr = localFizzBuzz (inputValue);

         std::cout << "Local test case ";
         std::cout << testCaseNumber;
         std::cout << " ";
         std::cout << "fizzBuzz with input of ";
         std::cout << inputValue;
         std::cout << " ";
         std::cout << "expecting ";
         std::cout << expectedResultStr;
         std::cout << " ";
         std::cout << "returned ";
         std::cout << resultStr;
         std::cout << " ";
```

Continued on the next page.

```
         if(expectedResultStr == resultStr) {
               std::cout << "PASSED\n";
         }
         else
         {
               std::cout << "FAILED\n";
         }

        //  end       void localTestCase
   }

int main(int argc, char* argv[]) {

   if (argc > 1) {
      //  begin   user asked for help
      std::cout << "testFizzBuzzCpp:\n";
      std::cout << "If the number is divisible by 3, output FIZZ.\n";
      std::cout << "If the number is divisible by 5, output BUZZ.\n";
      std::cout << "If the number is divisible by BOTH 3 and 5, output FIZZBUZZ.\n";
      std::cout << "If the number is NOT divisible by EITHER 3 or 5, output the number.\n";

      //  end    user asked for help
   }
   else
   {
      //  begin   run the program normally

        std::cout << "testFizzBuzzCpp:\n";
        std::cout << "For help, type testFizzBuzzCpp -h.\n";
        std::cout << "\n";
        std::cout << "testFizzBuzzCpp results:\n";


        //  testCase (int testCaseNumber, int inputValue, string expectedResultStr);

        testCase(1, 1, "1");
        testCase(2, 2, "2");
        testCase(3, 3, "FIZZ");
        testCase(4, 4, "4");
        testCase(5, 5, "BUZZ");
        testCase(6,15, "FIZZBUZZ");
        testCase(7, -9, "FIZZ");
        testCase(8, 16, "17");                  // *** should fail ***

        std::cout << "\n";

        //  localTestCase (int testCaseNumber, int inputValue, string expectedResultStr);

        localTestCase(1, 1, "1");
        localTestCase(2, 2, "2");
        localTestCase(3, 3, "FIZZ");
        localTestCase(4, 4, "4");
        localTestCase(5, 5, "BUZZ");
        localTestCase(6, 15, "FIZZBUZZ");
        localTestCase(7, -9, "FIZZ");
        localTestCase(8, 16, "17");             // *** should fail ***

        //  end     run the program normally
   }


   std::cout << "End of program.\n";

   //  exit the program with a return value of 0
   return 0;

   //  end       int main
}
```

Figure 7: C++ code for testFizzBuzzCpp.cpp

10 Build the Console App

Make sure that the software builds correctly. Compile and link **testFizzBuzzCpp.cpp** to create an executable file named **testFizzBuzzCpp.exe** .

```
g++ -o testFizzBuzzCpp.exe testFizzBuzzCpp.cpp ../library/Class1.cpp
```

Run testFizzBuzzCpp.exe. The output should look like this:

```
C:\ ... \console_app> testfizzbuzzcpp
testFizzBuzzCpp:
For help, type testFizzBuzzCpp -h.

testFizzBuzzCpp results:
Test case 1 fizzBuzz with input of 1 expecting 1 returned 1 PASSED
Test case 2 fizzBuzz with input of 2 expecting 2 returned 2 PASSED
Test case 3 fizzBuzz with input of 3 expecting FIZZ returned FIZZ PASSED
Test case 4 fizzBuzz with input of 4 expecting 4 returned 4 PASSED
Test case 5 fizzBuzz with input of 5 expecting BUZZ returned BUZZ PASSED
Test case 6 fizzBuzz with input of 15 expecting FIZZBUZZ returned FIZZBUZZ PASSED
Test case 7 fizzBuzz with input of -9 expecting FIZZ returned FIZZ PASSED
Test case 8 fizzBuzz with input of 16 expecting 17 returned 16 FAILED

Local test case 1 fizzBuzz with input of 1 expecting 1 returned 1 PASSED
Local test case 2 fizzBuzz with input of 2 expecting 2 returned 2 PASSED
Local test case 3 fizzBuzz with input of 3 expecting FIZZ returned FIZZ PASSED
Local test case 4 fizzBuzz with input of 4 expecting 4 returned 4 PASSED
Local test case 5 fizzBuzz with input of 5 expecting BUZZ returned BUZZ PASSED
Local test case 6 fizzBuzz with input of 15 expecting FIZZBUZZ returned FIZZBUZZ PASSED
Local test case 7 fizzBuzz with input of -9 expecting FIZZ returned FIZZ PASSED
Local test case 8 fizzBuzz with input of 16 expecting 17 returned 16 FAILED
End of program.
```

Build the app with the test coverage flags, **-fprofile-arcs** and **-ftest-coverage**.

```
g++ -o testFizzBuzzCpp.exe testFizzBuzzCpp.cpp ../library/Class1.cpp --coverage -fprofile-arcs -ftest-coverage
```

After successful compilation, we should see the original files **testFizzBuzzCpp.cpp** and **testFizzbuzzCpp.h** . In addition, we should see the executable program **testFizzBuzzCpp.exe** and two files named **testFizzBuzzCpp-Class1.gcno** and **testFizzBuzzCpp-testFizzBuzzCpp.gcno** . The .gcno files are generated because we added the compile option **-ftest-coverage** , which adds information for reconstructing the base block map and assigning source line numbers to blocks during the compilation process. [180] [182]

Run the executable.

```
C:\ ... \console_app> testfizzbuzzcpp
testFizzBuzzCpp:
For help, type testFizzBuzzCpp -h.

testFizzBuzzCpp results:
Test case 1 fizzBuzz with input of 1 expecting 1 returned 1 PASSED
Test case 2 fizzBuzz with input of 2 expecting 2 returned 2 PASSED
Test case 3 fizzBuzz with input of 3 expecting FIZZ returned FIZZ PASSED
Test case 4 fizzBuzz with input of 4 expecting 4 returned 4 PASSED
Test case 5 fizzBuzz with input of 5 expecting BUZZ returned BUZZ PASSED
Test case 6 fizzBuzz with input of 15 expecting FIZZBUZZ returned FIZZBUZZ PASSED
Test case 7 fizzBuzz with input of -9 expecting FIZZ returned FIZZ PASSED
Test case 8 fizzBuzz with input of 16 expecting 17 returned 16 FAILED

Local test case 1 fizzBuzz with input of 1 expecting 1 returned 1 PASSED
Local test case 2 fizzBuzz with input of 2 expecting 2 returned 2 PASSED
Local test case 3 fizzBuzz with input of 3 expecting FIZZ returned FIZZ PASSED
Local test case 4 fizzBuzz with input of 4 expecting 4 returned 4 PASSED
Local test case 5 fizzBuzz with input of 5 expecting BUZZ returned BUZZ PASSED
Local test case 6 fizzBuzz with input of 15 expecting FIZZBUZZ returned FIZZBUZZ PASSED
Local test case 7 fizzBuzz with input of -9 expecting FIZZ returned FIZZ PASSED
Local test case 8 fizzBuzz with input of 16 expecting 17 returned 16 FAILED
End of program.
```

When the executable is run, the results are recorded in .gcda data files. We should see two additional files: **testFizzBuzzCpp-Class1.gcda** and **testFizzBuzzCpp-testFizzBuzzCpp.gcda** . These data files are generated because the program was compiled with the **-fprofile-arcs** option. The .gcda data files contain arc transition counts, value distribution counts, and some summary information. [180] [182]

Now we are going to use gcov to find out how much of the code in the fizzBuzz function was covered.

11 Using GCOV on the CONSOLE App

```
C:\ ... \console_app> gcov -version
gcov (Rev2, Built by MSYS2 project) 13.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

The gcov command produces an annotated version of the original source file, with the file extension '.gcov'. The file generated by the tool shows a count of the number of times each line was executed. The line counts can be seen in the first column of the output. **Lines which were not executed are marked with hashes (######)** . [183]

Run gcov on testFizzBuzzCpp.

```
C:\ ... \console_app> gcov testFizzBuzzCpp-testFizzBuzzCpp

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/basic_string.tcc'
Lines executed:83.33% of 12
Creating 'basic_string.tcc.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/stl_iterator_base_types.h'
Lines executed:100.00% of 1
Creating 'stl_iterator_base_types.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/stl_iterator_base_funcs.h'
Lines executed:100.00% of 3
Creating 'stl_iterator_base_funcs.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/basic_string.h'
Lines executed:95.24% of 21
Creating 'basic_string.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/move.h'
Lines executed:0.00% of 2
Creating 'move.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/new_allocator.h'
Lines executed:100.00% of 2
Creating 'new_allocator.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/char_traits.h'
Lines executed:28.00% of 25
Creating 'char_traits.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/charconv.h'
Lines executed:65.38% of 26
Creating 'charconv.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/allocator.h'
Lines executed:100.00% of 2
Creating 'allocator.h.gcov'

File 'testFizzBuzzCpp.cpp'
Lines executed:92.50% of 80
Creating 'testFizzBuzzCpp.cpp.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/x86_64-w64-mingw32/bits/c++config.h'
Lines executed:100.00% of 2
Creating 'c++config.h.gcov'

Lines executed:78.41% of 176
```

testFizzBuzzCpp.cpp.gcov should look like this:

```
        -:    0:Source:testFizzBuzzCpp.cpp
        -:    0:Graph:testFizzBuzzCpp-testFizzBuzzCpp.gcno
        -:    0:Data:testFizzBuzzCpp-testFizzBuzzCpp.gcda
        -:    0:Runs:1
        -:    1:// testFizzBuzzCpp.cpp : the application source code.
        -:    2:
        -:    3:#include <iostream>
        -:    4:#include <string>
        -:    5:#include "testFizzBuzzCpp.h"
        -:    6:#include "../library/Class1.h"
        -:    7:
        -:    8:
        8:    9:   std::string localFizzBuzz( int counter ) {
        -:   10:
        -:   11:          //  Input a number between 1 and 100.
        -:   12:          //  If the number is divisible by 3, output "FIZZ".
        -:   13:          //  If the number is divisible by 5, output "BUZZ".
        -:   14:          //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
        -:   15:          //  For all other numbers, output the original number.
        -:   16:
        8:   17:                 std::string resultStr;                         //  declare the variable
        -:   18:
        8:   19:                  resultStr = std::to_string(counter);
        -:   20:
```

Continued on the next page.

```
        8:   21:                  if(counter % 3 == 0) {     //  divisible by 3
        3:   22:                     resultStr = "FIZZ";
        -:   23:                  }
        -:   24:
        8:   25:                  if(counter % 5 == 0) {     //  divisible by 5
        2:   26:                     resultStr = "BUZZ";
        -:   27:                  }
        -:   28:
        8:   29:                  if( (counter % 3 == 0) && (counter % 5 == 0) ) {     //  divisible by both 3 and  5
        1:   30:                     resultStr =  "FIZZBUZZ";
        -:   31:                  }
        -:   32:
        8:   33:                return resultStr;
        -:   34:
        -:   35:        //  end      string localFizzBuzz
    =====:   36:   }
        -:   37:
        -:   38:
        -:   39:
        8:   40:   void testCase ( int testCaseNumber,
        -:   41:                  int inputValue,
        -:   42:                  std::string expectedResultStr ) {
        -:   43:
        8:   44:         std::string resultStr;                 //  declare the variable
        -:   45:
        8:   46:         Class1 clsObj;     //  instantiate Class1
        8:   47:         resultStr = clsObj.fizzBuzz(inputValue);
        -:   48:
        8:   49:         std::cout << "Test case ";
        8:   50:         std::cout << testCaseNumber;
        8:   51:         std::cout << " ";
        8:   52:         std::cout << "fizzBuzz with input of ";
        8:   53:         std::cout << inputValue;
        8:   54:         std::cout << " ";
        8:   55:         std::cout << "expecting ";
        8:   56:         std::cout << expectedResultStr;
        8:   57:         std::cout << " ";
        8:   58:         std::cout << "returned ";
        8:   59:         std::cout << resultStr;
        8:   60:         std::cout << " ";
        -:   61:
        8:   62:         if(expectedResultStr == resultStr) {
        7:   63:               std::cout << "PASSED\n";
        -:   64:         }
        -:   65:         else
        -:   66:         {
        1:   67:               std::cout << "FAILED\n";
        -:   68:         }
        -:   69:
        -:   70:        //  end       void testCase
        8:   71:   }
        -:   72:
        -:   73:
        8:   74:   void localTestCase ( int testCaseNumber,
        -:   75:                  int inputValue,
        -:   76:                  std::string expectedResultStr ) {
        -:   77:
        8:   78:         std::string resultStr;                 //  declare the variable
        -:   79:
        8:   80:         resultStr = localFizzBuzz (inputValue);
        -:   81:
        8:   82:         std::cout << "Local test case ";
        8:   83:         std::cout << testCaseNumber;
        8:   84:         std::cout << " ";
        8:   85:         std::cout << "fizzBuzz with input of ";
        8:   86:         std::cout << inputValue;
        8:   87:         std::cout << " ";
        8:   88:         std::cout << "expecting ";
        8:   89:         std::cout << expectedResultStr;
        8:   90:         std::cout << " ";
        8:   91:         std::cout << "returned ";
        8:   92:         std::cout << resultStr;
        8:   93:         std::cout << " ";
        -:   94:
        8:   95:         if(expectedResultStr == resultStr) {
        7:   96:               std::cout << "PASSED\n";
        -:   97:         }
        -:   98:         else
        -:   99:         {
        1:  100:               std::cout << "FAILED\n";
        -:  101:         }
        -:  102:
        -:  103:        //  end       void localTestCase
        8:  104:   }
```

Continued on the next page.

```
        -:  105:
        -:  106:
        -:  107:
        -:  108:
        -:  109:
        1:  110:int main(int argc, char* argv[]) {
        -:  111:
        1:  112:   if (argc > 1) {
        -:  113:      //  begin   user asked for help
    #####:  114:      std::cout << "testFizzBuzzCpp:\n";
    #####:  115:      std::cout << "If the number is divisible by 3, output FIZZ.\n";
    #####:  116:      std::cout << "If the number is divisible by 5, output BUZZ.\n";
    #####:  117:      std::cout << "If the number is divisible by BOTH 3 and 5, output FIZZBUZZ.\n";
    #####:  118:      std::cout << "If the number is NOT divisible by EITHER 3 or 5, output the number.\n";
        -:  119:
        -:  120:      //  end    user asked for help
        -:  121:   }
        -:  122:   else
        -:  123:   {
        -:  124:      //  begin   run the program normally
        -:  125:
        1:  126:        std::cout << "testFizzBuzzCpp:\n";
        1:  127:        std::cout << "For help, type testFizzBuzzCpp -h.\n";
        1:  128:        std::cout << "\n";
        1:  129:        std::cout << "testFizzBuzzCpp results:\n";
        -:  130:
        -:  131:
        -:  132:        //  testCase (int testCaseNumber, int inputValue, string expectedResultStr);
        -:  133:
        2:  134:        testCase(1, 1, "1");
        2:  135:        testCase(2, 2, "2");
        2:  136:        testCase(3, 3, "FIZZ");
        2:  137:        testCase(4, 4, "4");
        2:  138:        testCase(5, 5, "BUZZ");
        2:  139:        testCase(6,15, "FIZZBUZZ");
        2:  140:        testCase(7, -9, "FIZZ");
        1:  141:        testCase(8, 16, "17");                  // *** should fail ***
        -:  142:
        1:  143:        std::cout << "\n";
        -:  144:
        -:  145:        //  localTestCase (int testCaseNumber, int inputValue, string expectedResultStr);
        -:  146:
        2:  147:        localTestCase(1, 1, "1");
        2:  148:        localTestCase(2, 2, "2");
        2:  149:        localTestCase(3, 3, "FIZZ");
        2:  150:        localTestCase(4, 4, "4");
        2:  151:        localTestCase(5, 5, "BUZZ");
        2:  152:        localTestCase(6, 15, "FIZZBUZZ");
        2:  153:        localTestCase(7, -9, "FIZZ");
        2:  154:        localTestCase(8, 16, "17");             // *** should fail ***
        -:  155:
        -:  156:        //  end     run the program normally
        -:  157:   }
        -:  158:
        -:  159:
        1:  160:   std::cout << "End of program.\n";
        -:  161:
        -:  162:   //  exit the program with a return value of 0
        1:  163:   return 0;
        -:  164:
        -:  165:   //  end       int main
        -:  166:}
```

Figure 8:   gcov output for testFizzBuzzCpp.cpp.gcov

Run gcov on Class1.

C:\ ... \console_app> gcov testFizzBuzzCpp-Class1

```
File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/move.h'
Lines executed:0.00% of 2
Creating 'move.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/new_allocator.h'
Lines executed:0.00% of 2
Creating 'new_allocator.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/charconv.h'
Lines executed:0.00% of 26
Creating 'charconv.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/basic_string.h'
Lines executed:0.00% of 10
Creating 'basic_string.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/allocator.h'
Lines executed:0.00% of 2
Creating 'allocator.h.gcov'

File '../library/Class1.cpp'
Lines executed:93.33% of 15
Creating 'Class1.cpp.gcov'

Lines executed:24.56% of 57
```

Class1.cpp.gcov should look like this:

```
    -:    0:Source:../library/Class1.cpp
    -:    0:Graph:testFizzBuzzCpp-Class1.gcno
    -:    0:Data:testFizzBuzzCpp-Class1.gcda
    -:    0:Runs:1
    -:    1:
    -:    2:
    -:    3:// Class1.cpp
    -:    4:
    -:    5:#include <string>
    -:    6:#include "Class1.h"
    -:    7:
    -:    8:
    8:    9:Class1::Class1()
    -:   10:{
    8:   11:}
    -:   12:
    8:   13:std::string Class1::fizzBuzz( int counter )
    -:   14:{
    -:   15:          //  Input a number between 1 and 100.
    -:   16:          //  If the number is divisible by 3, output "FIZZ".
    -:   17:          //  If the number is divisible by 5, output "BUZZ".
    -:   18:          //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
    -:   19:          //  For all other numbers, output the original number.
    -:   20:
    8:   21:                 std::string resultStr;                         //  declare the variable
    -:   22:
    8:   23:                  resultStr = std::to_string(counter);
    -:   24:
    8:   25:                  if(counter % 3 == 0) {     //  divisible by 3
    3:   26:                     resultStr = "FIZZ";
    -:   27:                  }
    -:   28:
    8:   29:                  if(counter % 5 == 0) {     //  divisible by 5
    2:   30:                     resultStr = "BUZZ";
    -:   31:                  }
    -:   32:
    8:   33:                  if( (counter % 3 == 0) && (counter % 5 == 0) ) {     //  divisible by both 3 and  5
    1:   34:                     resultStr =  "FIZZBUZZ";
    -:   35:                  }
    -:   36:
    8:   37:                return resultStr;
    -:   38:
    -:   39:
    -:   40:        //  end      string fizzBuzz
=====:   41:}
    -:   42:
    -:   43:
    8:   44:Class1::~Class1()
    -:   45:{
    8:   46:}
    -:   47:
```

Figure 9: gcov output for Class1.cpp.gcov

12 Generate the report using reportgenerator

If we have the **.NET SDK** installed on our system, we can also use the report generator tool for .NET to generate the report.

13 .NET CLI

The .NET framework includes a command line interface (CLI). The .NET command-line interface (CLI) is a cross-platform toolchain for developing, building, running, and publishing .NET applications. The .NET CLI is included with the .NET SDK. [123] After installing the SDK, CLI commands are run by opening a Command Prompt window and entering the commands at the command prompt. [122] [124] [134] Once the .NET SDK is installed, further help on using the CLI may be obtained by typing

dotnet --help or dotnet -h

To determine the version of the .NET SDK which is installed, type

```
dotnet --info
```

If we are using a **.NET SDK earlier than version 8** , we may need to add some software in order to generate coverage reports. [145] [146] [151]

```
dotnet new tool-manifest
```

```
dotnet add package coverlet.collector
```

```
dotnet add package coverlet.msbuild
```

```
dotnet tool install coverlet.console
```

```
dotnet tool install -g dotnet-reportgenerator-globaltool
```

If we are using **.NET SDK 8** , we may need to install the following software: [146] [147]

```
dotnet tool install --global dotnet-coverage
```

The report generator converts coverage reports generated by coverlet, OpenCover, dotCover, Visual Studio, NCover, Cobertura, JaCoCo, Clover, gcov or lcov into human readable reports in various formats. The reports show which lines of your source code have been covered. ReportGenerator is available under the Apache [136] License. [148]

Some of the many report types available include: [157]

Html	**A summary (index.html) and detailed reports for each class.**
Html_Light	**Html with a light theme.**
Html_Dark	**Html with a dark theme.**
HtmlSummary	**A single HTML file (summary.html) without links.**
TextSummary	**A single TXT file containing coverage information per class.**
SvgChart	**A single SVG file containing a chart with historic coverage information.**
Badges	**SVG files that show line and/or branch coverage information.**

14 Using the Reportgenerator Installed Under the Dotnet CLI

Use the dotnet reportgenerator tool to create the coverage report for testFizzBuzzCpp.cpp.

```
C:\ ... \console_app> reportgenerator -reports:testFizzBuzzCpp.cpp.gcov -targetdir:coverage-report1 -reporttypes:"Html_Dark"
```

```
2025-03-16T14:37:03: Arguments
2025-03-16T14:37:03:  -reports:testFizzBuzzCpp.cpp.gcov
2025-03-16T14:37:03:  -targetdir:coverage-report1
2025-03-16T14:37:03:  -reporttypes:Html_Dark
2025-03-16T14:37:04:   No source directories supplied for 'GCov' coverage file
2025-03-16T14:37:04: Writing report file 'coverage-report1\index.html'
```

Use the dotnet reportgenerator tool to create the coverage report for Class1.cpp.

```
C:\ ... \console_app> reportgenerator -reports:Class1.cpp.gcov -targetdir:coverage-report2 -reporttypes:"Html_Dark"
```

```
2025-03-16T14:38:01: Arguments
2025-03-16T14:38:01:  -reports:Class1.cpp.gcov
2025-03-16T14:38:01:  -targetdir:coverage-report2
2025-03-16T14:38:01:  -reporttypes:Html_Dark
2025-03-16T14:38:01:   No source directories supplied for 'GCov' coverage file
2025-03-16T14:38:02: Writing report file 'coverage-report2\index.html'
```

We wrote a coverage report for testFizzBuzzCpp.cpp to **coverage-report1\index.html** . We also wrote a coverage report for Class1.cpp to **coverage-report2\index.html** . We can open the files in a browser.

15 Coverage Report

The coverage reports created by reportgenerator show that we have the following coverage:

```
File(s):        testFizzBuzzCpp.cpp

Covered lines:  74
Uncovered lines:        6
Coverable lines:        80
Total lines:    166
Line coverage:  92.5%
```

We have 6 uncovered lines of code. The first thing we see is that the closing brace at line 36 is not covered.

```
          32
  8       33                              return resultStr;
          34
          35                      //  end      string localFizzBuzz
  0       36                 }
          37
          38
          39
```

The information can also be seen in the gcov file for testFizzBuzzCpp.cpp (testFizzBuzzCpp.cpp.gcov).

```
        -:   32:
        8:   33:                return resultStr;
        -:   34:
        -:   35:        //  end      string localFizzBuzz
    =====:   36:   }
        -:   37:
        -:   38:
        -:   39:
```

The closing brace at line 36 is not covered due to the return statement immediately preceding it.

We have 5 uncovered lines of code at line 114 through 118 because the user did not ask for help when they ran the program.

```
     1       112                 if (argc > 1) {
             113                    //  begin   user asked for help
     0       114                    std::cout << "testFizzBuzzCpp:\n";
     0       115                    std::cout << "If the number is divisible by 3, output FIZZ.\n";
     0       116                    std::cout << "If the number is divisible by 5, output BUZZ.\n";
     0       117                    std::cout << "If the number is divisible by BOTH 3 and 5, output FIZZBUZZ.\n";
     0       118                    std::cout << "If the number is NOT divisible by EITHER 3 or 5, output the number.\n";
             119
             120                    //  end    user asked for help
             121                 }
```

The information can also be seen in the gcov file for testFizzBuzzCpp.cpp (testFizzBuzzCpp.cpp.gcov).

```
        1:  112:   if (argc > 1) {
        -:  113:      //  begin   user asked for help
    #####:  114:      std::cout << "testFizzBuzzCpp:\n";
    #####:  115:      std::cout << "If the number is divisible by 3, output FIZZ.\n";
    #####:  116:      std::cout << "If the number is divisible by 5, output BUZZ.\n";
    #####:  117:      std::cout << "If the number is divisible by BOTH 3 and 5, output FIZZBUZZ.\n";
    #####:  118:      std::cout << "If the number is NOT divisible by EITHER 3 or 5, output the number.\n";
        -:  119:
        -:  120:      //  end    user asked for help
        -:  121:   }
```

We could run the program again and ask for help in order to cover those lines of code. We have now accounted for the 6 uncovered lines of code. We have effectively achieved 100% line coverage.

Let's look at the coverage for our library.

```
File(s):        ../library/Class1.cpp

Covered lines:  14
Uncovered lines:        1
Coverable lines:        15
Total lines:    47
Line coverage:  93.3%
```

We have 1 uncovered line of code.

```
                36
        8       37                              return resultStr;
                38
                39
                40                      //  end      string fizzBuzz
        0       41              }
                42
                43
```

The information can also be seen in the gcov file for Class1.cpp (Class1.cpp.gcov).

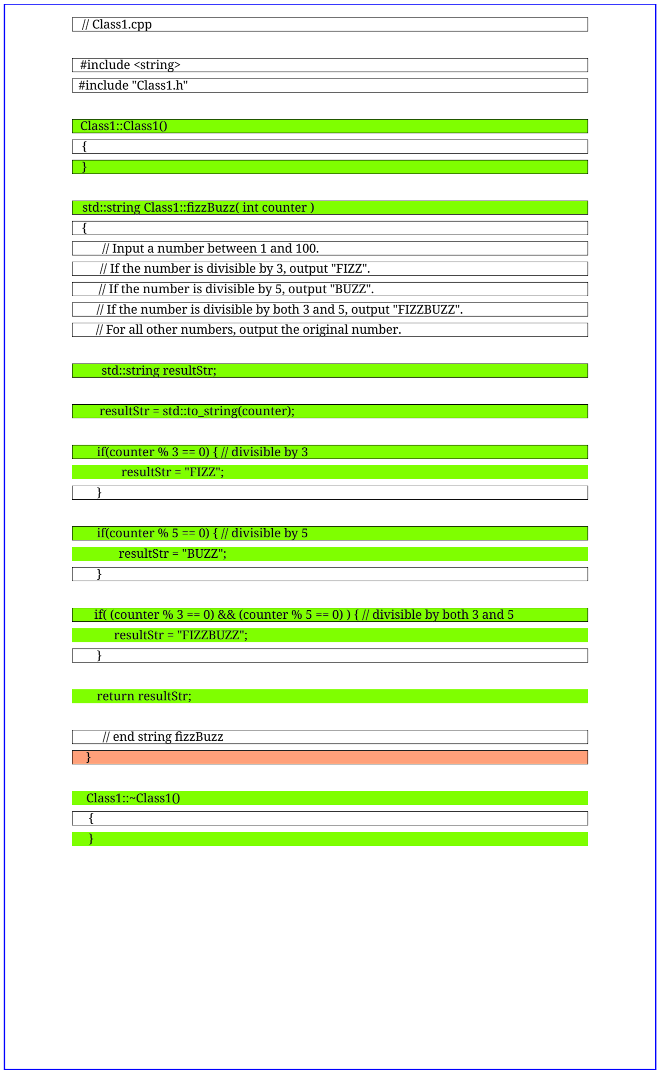
```
        -:   36:
        8:   37:                return resultStr;
        -:   38:
        -:   39:
        -:   40:        //  end      string fizzBuzz
    =====:   41:}
        -:   42:
        -:   43:
```

The closing brace at line 41 is not covered due to the return statement immediately preceding it. We have effectively achieved 100% line coverage.

16 Coverage Report for the CONSOLE App Compared to the Coverage Report when using GTEST

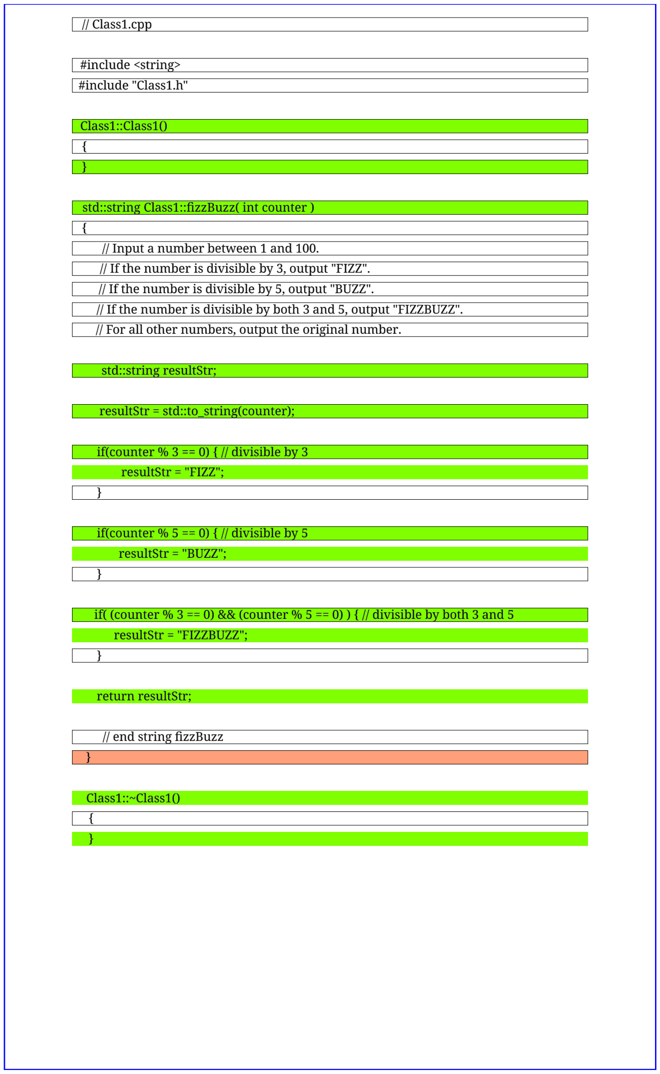
On the following pages are sample reports similar to the output of reportgenerator. The first report shows the coverage we obtain when we run the CONSOLE app. The second report shows the coverage we obtain when we use GTEST. This is followed by an explanation of how to use GTEST.

In both cases, the closing brace at the end of fizzBuzz is not covered due to the return statement immediately preceding it. We have effectively achieved 100% line coverage.



```
// Class1.cpp

#include <string>
#include "Class1.h"

Class1::Class1()
{
}

std::string Class1::fizzBuzz( int counter )
{
    // Input a number between 1 and 100.
    // If the number is divisible by 3, output "FIZZ".
    // If the number is divisible by 5, output "BUZZ".
    // If the number is divisible by both 3 and 5, output "FIZZBUZZ".
    // For all other numbers, output the original number.

    std::string resultStr;

    resultStr = std::to_string(counter);

    if(counter % 3 == 0) { // divisible by 3
        resultStr = "FIZZ";
    }

    if(counter % 5 == 0) { // divisible by 5
        resultStr = "BUZZ";
    }

    if( (counter % 3 == 0) && (counter % 5 == 0) ) { // divisible by both 3 and 5
        resultStr = "FIZZBUZZ";
    }

    return resultStr;

    // end string fizzBuzz
}

Class1::~Class1()
{
}
```

Figure 11: Coverage report when running the CONSOLE app.



```
// Class1.cpp

#include <string>
#include "Class1.h"

Class1::Class1()
{
}

std::string Class1::fizzBuzz( int counter )
{
    // Input a number between 1 and 100.
    // If the number is divisible by 3, output "FIZZ".
    // If the number is divisible by 5, output "BUZZ".
    // If the number is divisible by both 3 and 5, output "FIZZBUZZ".
    // For all other numbers, output the original number.

    std::string resultStr;

    resultStr = std::to_string(counter);

    if(counter % 3 == 0) { // divisible by 3
        resultStr = "FIZZ";
    }

    if(counter % 5 == 0) { // divisible by 5
        resultStr = "BUZZ";
    }

    if( (counter % 3 == 0) && (counter % 5 == 0) ) { // divisible by both 3 and 5
        resultStr = "FIZZBUZZ";
    }

    return resultStr;

    // end string fizzBuzz
}

Class1::~Class1()
{
}
```

Figure 12: Coverage report when running the GTEST app.

17 Using GTEST

We can use Google Test to build test cases for fizzBuzz.

Change directory to the gtest directory.

cd gtest

Create a gtest application by creating two files;

fizzBuzzCppGtest.h and ***fizzBuzzCppGtest.cpp***

fizzBuzzCppGtest.h looks like this:

```
// fizzBuzzCppGtest.h: the application header code.
/* Additional source code to include. */
```

Figure 13: C++ source code for fizzBuzzCppGtest.h

fizzBuzzCppGtest.cpp looks like this:

```
// fizzBuzzCppGtest.cpp : the application source code.

#include "gtest/gtest.h"
#include <iostream>
#include <string>
#include "fizzBuzzCppGtest.h"
#include "../library/Class1.h"

TEST(fizzBuzz_Test1,InputIs1_ReturnOriginalNumber){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("1", clsObj.fizzBuzz(1));
};

TEST(fizzBuzz_Test2, InputIs2_ReturnOriginalNumber){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("2", clsObj.fizzBuzz(2));
};

TEST(fizzBuzz_Test3,InputIs3_DivisibleBy3ReturnFizz){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("FIZZ", clsObj.fizzBuzz(3));
};

TEST(fizzBuzz_Test4,InputIs4_ReturnOriginalNumber){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("4", clsObj.fizzBuzz(4));
};

TEST(fizzBuzz_Test5,InputIs5_DivisibleBy5ReturnBuzz){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("BUZZ", clsObj.fizzBuzz(5));
};

TEST(fizzBuzz_Test6,InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("FIZZBUZZ", clsObj.fizzBuzz(15));
};

TEST(fizzBuzz_Test7,InputIsNegative9_DivisibleBy3ReturnFizz){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("FIZZ", clsObj.fizzBuzz(-9));
};

TEST(fizzBuzz_Test8,InputIs16_Expect17ShouldFail){
   Class1 clsObj;     //  instantiate Class1
   EXPECT_EQ("17", clsObj.fizzBuzz(16));
};

int main(int argc, char**argv) {

        testing::InitGoogleTest(&argc, argv);
        return RUN_ALL_TESTS();

   //  end       int main
}
```

Figure 14: C++ source code for fizzBuzzCppGtest.cpp

Compile the GTEST application.

C:\ ... \gtest> g++ -o fizzBuzzCppGtest.exe fizzBuzzCppGtest.cpp ../library/Class1.cpp -L /user/lib -lgtest -lgtest_main -pthread

After the compile is successful, run the tests.

C:\ ... \gtest> fizzbuzzcppgtest

```
[==========] Running 8 tests from 8 test suites.
[----------] Global test environment set-up.
[----------] 1 test from fizzBuzz_Test1
[ RUN      ] fizzBuzz_Test1.InputIs1_ReturnOriginalNumber
[       OK ] fizzBuzz_Test1.InputIs1_ReturnOriginalNumber (0 ms)
[----------] 1 test from fizzBuzz_Test1 (4 ms total)

[----------] 1 test from fizzBuzz_Test2
[ RUN      ] fizzBuzz_Test2.InputIs2_ReturnOriginalNumber
[       OK ] fizzBuzz_Test2.InputIs2_ReturnOriginalNumber (0 ms)
[----------] 1 test from fizzBuzz_Test2 (6 ms total)

[----------] 1 test from fizzBuzz_Test3
[ RUN      ] fizzBuzz_Test3.InputIs3_DivisibleBy3ReturnFizz
[       OK ] fizzBuzz_Test3.InputIs3_DivisibleBy3ReturnFizz (0 ms)
[----------] 1 test from fizzBuzz_Test3 (6 ms total)

[----------] 1 test from fizzBuzz_Test4
[ RUN      ] fizzBuzz_Test4.InputIs4_ReturnOriginalNumber
[       OK ] fizzBuzz_Test4.InputIs4_ReturnOriginalNumber (0 ms)
[----------] 1 test from fizzBuzz_Test4 (4 ms total)

[----------] 1 test from fizzBuzz_Test5
[ RUN      ] fizzBuzz_Test5.InputIs5_DivisibleBy5ReturnBuzz
[       OK ] fizzBuzz_Test5.InputIs5_DivisibleBy5ReturnBuzz (0 ms)
[----------] 1 test from fizzBuzz_Test5 (7 ms total)

[----------] 1 test from fizzBuzz_Test6
[ RUN      ] fizzBuzz_Test6.InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz
[       OK ] fizzBuzz_Test6.InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz (0 ms)
[----------] 1 test from fizzBuzz_Test6 (6 ms total)

[----------] 1 test from fizzBuzz_Test7
[ RUN      ] fizzBuzz_Test7.InputIsNegative9_DivisibleBy3ReturnFizz
[       OK ] fizzBuzz_Test7.InputIsNegative9_DivisibleBy3ReturnFizz (0 ms)
[----------] 1 test from fizzBuzz_Test7 (6 ms total)

[----------] 1 test from fizzBuzz_Test8
[ RUN      ] fizzBuzz_Test8.InputIs16_Expect17ShouldFail
fizzBuzzCppGtest.cpp:54: Failure
Expected equality of these values:
  "17"
  clsObj.fizzBuzz(16)
    Which is: "16"

[  FAILED  ] fizzBuzz_Test8.InputIs16_Expect17ShouldFail (14 ms)
[----------] 1 test from fizzBuzz_Test8 (19 ms total)

[----------] Global test environment tear-down
[==========] 8 tests from 8 test suites ran. (125 ms total)
[  PASSED  ] 7 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] fizzBuzz_Test8.InputIs16_Expect17ShouldFail

 1 FAILED TEST
```

Build the app with the test coverage flags.

C:\ ... \gtest> g++ -o fizzBuzzCppGtest.exe fizzBuzzCppGtest.cpp ../library/Class1.cpp --coverage -fprofile-arcs -ftest-coverage -L /user/lib -lgtest -lgtest_main -pthread

After successful compilation, we should see the original files **fizzBuzzCppGtest.cpp** and **fizzbuzzCppGtest.h** . In addition, we should see the executable program **fizzBuzzCppGtest.exe** and two files named **fizzBuzzCppGtest-Class1.gcno** and **fizzBuzzCppGtest-fizzBuzzCppGtest.gcno** . The .gcno files are generated because we added the compile option **-ftest-coverage** , which adds information for reconstructing the base block map and assigning source line numbers to blocks during the compilation process. [180] [182]

Run the executable.

C:\ ... \gtest> fizzbuzzcppgtest

```
[==========] Running 8 tests from 8 test suites.
[----------] Global test environment set-up.
[----------] 1 test from fizzBuzz_Test1
[ RUN      ] fizzBuzz_Test1.InputIs1_ReturnOriginalNumber
[       OK ] fizzBuzz_Test1.InputIs1_ReturnOriginalNumber (0 ms)
[----------] 1 test from fizzBuzz_Test1 (4 ms total)

[----------] 1 test from fizzBuzz_Test2
[ RUN      ] fizzBuzz_Test2.InputIs2_ReturnOriginalNumber
[       OK ] fizzBuzz_Test2.InputIs2_ReturnOriginalNumber (0 ms)
[----------] 1 test from fizzBuzz_Test2 (10 ms total)

[----------] 1 test from fizzBuzz_Test3
[ RUN      ] fizzBuzz_Test3.InputIs3_DivisibleBy3ReturnFizz
[       OK ] fizzBuzz_Test3.InputIs3_DivisibleBy3ReturnFizz (0 ms)
[----------] 1 test from fizzBuzz_Test3 (8 ms total)

[----------] 1 test from fizzBuzz_Test4
[ RUN      ] fizzBuzz_Test4.InputIs4_ReturnOriginalNumber
[       OK ] fizzBuzz_Test4.InputIs4_ReturnOriginalNumber (0 ms)
[----------] 1 test from fizzBuzz_Test4 (7 ms total)

[----------] 1 test from fizzBuzz_Test5
[ RUN      ] fizzBuzz_Test5.InputIs5_DivisibleBy5ReturnBuzz
[       OK ] fizzBuzz_Test5.InputIs5_DivisibleBy5ReturnBuzz (0 ms)
[----------] 1 test from fizzBuzz_Test5 (4 ms total)

[----------] 1 test from fizzBuzz_Test6
[ RUN      ] fizzBuzz_Test6.InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz
[       OK ] fizzBuzz_Test6.InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz (0 ms)
[----------] 1 test from fizzBuzz_Test6 (6 ms total)

[----------] 1 test from fizzBuzz_Test7
[ RUN      ] fizzBuzz_Test7.InputIsNegative9_DivisibleBy3ReturnFizz
[       OK ] fizzBuzz_Test7.InputIsNegative9_DivisibleBy3ReturnFizz (0 ms)
[----------] 1 test from fizzBuzz_Test7 (5 ms total)

[----------] 1 test from fizzBuzz_Test8
[ RUN      ] fizzBuzz_Test8.InputIs16_Expect17ShouldFail
fizzBuzzCppGtest.cpp:54: Failure
Expected equality of these values:
  "17"
  clsObj.fizzBuzz(16)
    Which is: "16"

[  FAILED  ] fizzBuzz_Test8.InputIs16_Expect17ShouldFail (6 ms)
[----------] 1 test from fizzBuzz_Test8 (18 ms total)

[----------] Global test environment tear-down
[==========] 8 tests from 8 test suites ran. (132 ms total)
[  PASSED  ] 7 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] fizzBuzz_Test8.InputIs16_Expect17ShouldFail

 1 FAILED TEST
```

We should see two additional files named **fizzBuzzCppGtest-Class1.gcda** and **fizzBuzzCppGtest-fizzBuzzCppGtest.gcda** . These data files are generated because the program was compiled with the **-fprofile-arcs** option. The .gcda data files contain arc transition counts, value distribution counts, and some summary information. [180] [182]

Now we are going to use gcov to find out how much of the code in the fizzBuzz function was covered.

Run gcov on fizzBuzzCppGtest.

C:\ ... \gtest> gcov fizzBuzzCppGtest-fizzBuzzCppGtest

```
File 'fizzBuzzCppGtest.cpp'
Lines executed:100.00% of 91
Creating 'fizzBuzzCppGtest.cpp.gcov'

File 'C:/msys64/ucrt64/include/gtest/internal/gtest-internal.h'
Lines executed:88.89% of 27
Creating 'gtest-internal.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/new_allocator.h'
Lines executed:100.00% of 2
Creating 'new_allocator.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/allocator.h'
Lines executed:100.00% of 2
Creating 'allocator.h.gcov'

File 'C:/msys64/ucrt64/include/gtest/gtest-printers.h'
Lines executed:64.79% of 71
Creating 'gtest-printers.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/tuple'
Lines executed:100.00% of 38
Creating 'tuple.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/basic_string.h'
Lines executed:92.31% of 13
Creating 'basic_string.h.gcov'

File 'C:/msys64/ucrt64/include/gtest/gtest.h'
Lines executed:67.39% of 46
Creating 'gtest.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/unique_ptr.h'
Lines executed:100.00% of 48
Creating 'unique_ptr.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/basic_string.tcc'
Lines executed:66.67% of 36
Creating 'basic_string.tcc.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/stl_iterator_base_types.h'
Lines executed:100.00% of 1
Creating 'stl_iterator_base_types.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/stl_iterator_base_funcs.h'
Lines executed:100.00% of 3
Creating 'stl_iterator_base_funcs.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/move.h'
Lines executed:75.00% of 8
Creating 'move.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/char_traits.h'
Lines executed:28.00% of 25
Creating 'char_traits.h.gcov'

File 'C:/msys64/ucrt64/include/gtest/gtest-assertion-result.h'
Lines executed:100.00% of 4
Creating 'gtest-assertion-result.h.gcov'

File 'C:/msys64/ucrt64/include/gtest/gtest-message.h'
No executable lines
Removing 'gtest-message.h.gcov'

File 'C:/msys64/ucrt64/include/gtest/internal/gtest-port.h'
Lines executed:0.00% of 1
Creating 'gtest-port.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/x86_64-w64-mingw32/bits/c++config.h'
Lines executed:100.00% of 2
Creating 'c++config.h.gcov'

Lines executed:81.58% of 418
```

fizzBuzzCppGtest.cpp.gcov should look like this:

```
        -:    0:Source:fizzBuzzCppGtest.cpp
        -:    0:Graph:fizzBuzzCppGtest-fizzBuzzCppGtest.gcno
        -:    0:Data:fizzBuzzCppGtest-fizzBuzzCppGtest.gcda
        -:    0:Runs:1
        -:    1:// fizzBuzzCppGtest.cpp : the application source code.
        -:    2:
        -:    3:#include "gtest/gtest.h"
        -:    4:#include <iostream>
        -:    5:#include <string>
        -:    6:#include "fizzBuzzCppGtest.h"
        -:    7:#include "../library/Class1.h"
        -:    8:
        -:    9:
        -:   10:
        4:   11:TEST(fizzBuzz_Test1,InputIs1_ReturnOriginalNumber){
        1:   12:   Class1 clsObj;     //  instantiate Class1
       1*:   13:   EXPECT_EQ("1", clsObj.fizzBuzz(1));
        1:   14:};
------------------
_ZN49fizzBuzz_Test1_InputIs1_ReturnOriginalNumber_TestC1Ev:
        1:   11:TEST(fizzBuzz_Test1,InputIs1_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test1_InputIs1_ReturnOriginalNumber_TestD0Ev:
        1:   11:TEST(fizzBuzz_Test1,InputIs1_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test1_InputIs1_ReturnOriginalNumber_TestD1Ev:
        1:   11:TEST(fizzBuzz_Test1,InputIs1_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test1_InputIs1_ReturnOriginalNumber_Test8TestBodyEv:
        1:   11:TEST(fizzBuzz_Test1,InputIs1_ReturnOriginalNumber){
        1:   12:   Class1 clsObj;     //  instantiate Class1
       1*:   13:   EXPECT_EQ("1", clsObj.fizzBuzz(1));
        1:   14:};
------------------
        -:   15:
        4:   16:TEST(fizzBuzz_Test2, InputIs2_ReturnOriginalNumber){
        1:   17:   Class1 clsObj;     //  instantiate Class1
       1*:   18:   EXPECT_EQ("2", clsObj.fizzBuzz(2));
        1:   19:};
------------------
_ZN49fizzBuzz_Test2_InputIs2_ReturnOriginalNumber_TestC1Ev:
        1:   16:TEST(fizzBuzz_Test2, InputIs2_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test2_InputIs2_ReturnOriginalNumber_TestD0Ev:
        1:   16:TEST(fizzBuzz_Test2, InputIs2_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test2_InputIs2_ReturnOriginalNumber_TestD1Ev:
        1:   16:TEST(fizzBuzz_Test2, InputIs2_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test2_InputIs2_ReturnOriginalNumber_Test8TestBodyEv:
        1:   16:TEST(fizzBuzz_Test2, InputIs2_ReturnOriginalNumber){
        1:   17:   Class1 clsObj;     //  instantiate Class1
       1*:   18:   EXPECT_EQ("2", clsObj.fizzBuzz(2));
        1:   19:};
------------------
        -:   20:
        -:   21:
        4:   22:TEST(fizzBuzz_Test3,InputIs3_DivisibleBy3ReturnFizz){
        1:   23:   Class1 clsObj;     //  instantiate Class1
       1*:   24:   EXPECT_EQ("FIZZ", clsObj.fizzBuzz(3));
        1:   25:};
------------------
_ZN51fizzBuzz_Test3_InputIs3_DivisibleBy3ReturnFizz_TestC1Ev:
        1:   22:TEST(fizzBuzz_Test3,InputIs3_DivisibleBy3ReturnFizz){
------------------
_ZN51fizzBuzz_Test3_InputIs3_DivisibleBy3ReturnFizz_TestD0Ev:
        1:   22:TEST(fizzBuzz_Test3,InputIs3_DivisibleBy3ReturnFizz){
------------------
_ZN51fizzBuzz_Test3_InputIs3_DivisibleBy3ReturnFizz_TestD1Ev:
        1:   22:TEST(fizzBuzz_Test3,InputIs3_DivisibleBy3ReturnFizz){
------------------
_ZN51fizzBuzz_Test3_InputIs3_DivisibleBy3ReturnFizz_Test8TestBodyEv:
        1:   22:TEST(fizzBuzz_Test3,InputIs3_DivisibleBy3ReturnFizz){
        1:   23:   Class1 clsObj;     //  instantiate Class1
       1*:   24:   EXPECT_EQ("FIZZ", clsObj.fizzBuzz(3));
        1:   25:};
------------------
        -:   26:
        -:   27:
        4:   28:TEST(fizzBuzz_Test4,InputIs4_ReturnOriginalNumber){
        1:   29:   Class1 clsObj;     //  instantiate Class1
       1*:   30:   EXPECT_EQ("4", clsObj.fizzBuzz(4));
        1:   31:};
------------------
_ZN49fizzBuzz_Test4_InputIs4_ReturnOriginalNumber_TestC1Ev:
        1:   28:TEST(fizzBuzz_Test4,InputIs4_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test4_InputIs4_ReturnOriginalNumber_TestD0Ev:
        1:   28:TEST(fizzBuzz_Test4,InputIs4_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test4_InputIs4_ReturnOriginalNumber_TestD1Ev:
        1:   28:TEST(fizzBuzz_Test4,InputIs4_ReturnOriginalNumber){
------------------
_ZN49fizzBuzz_Test4_InputIs4_ReturnOriginalNumber_Test8TestBodyEv:
        1:   28:TEST(fizzBuzz_Test4,InputIs4_ReturnOriginalNumber){
        1:   29:   Class1 clsObj;     //  instantiate Class1
       1*:   30:   EXPECT_EQ("4", clsObj.fizzBuzz(4));
        1:   31:};
------------------
        -:   32:
```

```
        -:   33:
        4:   34:TEST(fizzBuzz_Test5,InputIs5_DivisibleBy5ReturnBuzz){
        1:   35:   Class1 clsObj;     //  instantiate Class1
       1*:   36:   EXPECT_EQ("BUZZ", clsObj.fizzBuzz(5));
        1:   37:};
------------------
_ZN51fizzBuzz_Test5_InputIs5_DivisibleBy5ReturnBuzz_TestC1Ev:
        1:   34:TEST(fizzBuzz_Test5,InputIs5_DivisibleBy5ReturnBuzz){
------------------
_ZN51fizzBuzz_Test5_InputIs5_DivisibleBy5ReturnBuzz_TestD0Ev:
        1:   34:TEST(fizzBuzz_Test5,InputIs5_DivisibleBy5ReturnBuzz){
------------------
_ZN51fizzBuzz_Test5_InputIs5_DivisibleBy5ReturnBuzz_TestD1Ev:
        1:   34:TEST(fizzBuzz_Test5,InputIs5_DivisibleBy5ReturnBuzz){
------------------
_ZN51fizzBuzz_Test5_InputIs5_DivisibleBy5ReturnBuzz_Test8TestBodyEv:
        1:   34:TEST(fizzBuzz_Test5,InputIs5_DivisibleBy5ReturnBuzz){
        1:   35:   Class1 clsObj;     //  instantiate Class1
       1*:   36:   EXPECT_EQ("BUZZ", clsObj.fizzBuzz(5));
        1:   37:};
------------------
        -:   38:
        -:   39:
        4:   40:TEST(fizzBuzz_Test6,InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz){
        1:   41:   Class1 clsObj;     //  instantiate Class1
       1*:   42:   EXPECT_EQ("FIZZBUZZ", clsObj.fizzBuzz(15));
        1:   43:};
------------------
_ZN62fizzBuzz_Test6_InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz_TestC1Ev:
        1:   40:TEST(fizzBuzz_Test6,InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz){
------------------
_ZN62fizzBuzz_Test6_InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz_TestD0Ev:
        1:   40:TEST(fizzBuzz_Test6,InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz){
------------------
_ZN62fizzBuzz_Test6_InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz_TestD1Ev:
        1:   40:TEST(fizzBuzz_Test6,InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz){
------------------
_ZN62fizzBuzz_Test6_InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz_Test8TestBodyEv:
        1:   40:TEST(fizzBuzz_Test6,InputIs15_DivisibleBy3AndBy5ReturnFizzBuzz){
        1:   41:   Class1 clsObj;     //  instantiate Class1
       1*:   42:   EXPECT_EQ("FIZZBUZZ", clsObj.fizzBuzz(15));
        1:   43:};
------------------
        -:   44:
        -:   45:
        4:   46:TEST(fizzBuzz_Test7,InputIsNegative9_DivisibleBy3ReturnFizz){
        1:   47:   Class1 clsObj;     //  instantiate Class1
       1*:   48:   EXPECT_EQ("FIZZ", clsObj.fizzBuzz(-9));
        1:   49:};
------------------
_ZN59fizzBuzz_Test7_InputIsNegative9_DivisibleBy3ReturnFizz_TestC1Ev:
        1:   46:TEST(fizzBuzz_Test7,InputIsNegative9_DivisibleBy3ReturnFizz){
------------------
_ZN59fizzBuzz_Test7_InputIsNegative9_DivisibleBy3ReturnFizz_TestD0Ev:
        1:   46:TEST(fizzBuzz_Test7,InputIsNegative9_DivisibleBy3ReturnFizz){
------------------
_ZN59fizzBuzz_Test7_InputIsNegative9_DivisibleBy3ReturnFizz_TestD1Ev:
        1:   46:TEST(fizzBuzz_Test7,InputIsNegative9_DivisibleBy3ReturnFizz){
------------------
_ZN59fizzBuzz_Test7_InputIsNegative9_DivisibleBy3ReturnFizz_Test8TestBodyEv:
        1:   46:TEST(fizzBuzz_Test7,InputIsNegative9_DivisibleBy3ReturnFizz){
        1:   47:   Class1 clsObj;     //  instantiate Class1
       1*:   48:   EXPECT_EQ("FIZZ", clsObj.fizzBuzz(-9));
        1:   49:};
------------------
        -:   50:
        -:   51:
        4:   52:TEST(fizzBuzz_Test8,InputIs16_Expect17ShouldFail){
        1:   53:   Class1 clsObj;     //  instantiate Class1
        1:   54:   EXPECT_EQ("17", clsObj.fizzBuzz(16));
        1:   55:};
------------------
_ZN48fizzBuzz_Test8_InputIs16_Expect17ShouldFail_TestC1Ev:
        1:   52:TEST(fizzBuzz_Test8,InputIs16_Expect17ShouldFail){
------------------
_ZN48fizzBuzz_Test8_InputIs16_Expect17ShouldFail_TestD0Ev:
        1:   52:TEST(fizzBuzz_Test8,InputIs16_Expect17ShouldFail){
------------------
_ZN48fizzBuzz_Test8_InputIs16_Expect17ShouldFail_TestD1Ev:
        1:   52:TEST(fizzBuzz_Test8,InputIs16_Expect17ShouldFail){
------------------
_ZN48fizzBuzz_Test8_InputIs16_Expect17ShouldFail_Test8TestBodyEv:
        1:   52:TEST(fizzBuzz_Test8,InputIs16_Expect17ShouldFail){
        1:   53:   Class1 clsObj;     //  instantiate Class1
        1:   54:   EXPECT_EQ("17", clsObj.fizzBuzz(16));
        1:   55:};
------------------
        -:   56:
        -:   57:
        -:   58:
        -:   59:
        -:   60:
```

```
        -:   61:
        1:   62:int main(int argc, char**argv) {
        -:   63:
        1:   64:        testing::InitGoogleTest(&argc, argv);
        1:   65:        return RUN_ALL_TESTS();
        -:   66:
        -:   67:   //  end       int main
        -:   68:}
```

Run gcov on Class1.

C:\ ... \gtest> gcov fizzBuzzCppGtest-Class1

```
File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/move.h'
Lines executed:0.00% of 2
Creating 'move.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/new_allocator.h'
Lines executed:50.00% of 2
Creating 'new_allocator.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/charconv.h'
Lines executed:65.38% of 26
Creating 'charconv.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/basic_string.h'
Lines executed:100.00% of 10
Creating 'basic_string.h.gcov'

File 'C:/msys64/ucrt64/include/c++/13.2.0/bits/allocator.h'
Lines executed:100.00% of 2
Creating 'allocator.h.gcov'

File '../library/Class1.cpp'
Lines executed:93.33% of 15
Creating 'Class1.cpp.gcov'

Lines executed:77.19% of 57
```

Class1.cpp.gcov should look like this:

```
        -:    0:Source:../library/Class1.cpp
        -:    0:Graph:fizzBuzzCppGtest-Class1.gcno
        -:    0:Data:fizzBuzzCppGtest-Class1.gcda
        -:    0:Runs:1
        -:    1:
        -:    2:
        -:    3:// Class1.cpp
        -:    4:
        -:    5:#include <string>
        -:    6:#include "Class1.h"
        -:    7:
        -:    8:
        8:    9:Class1::Class1()
        -:   10:{
        8:   11:}
        -:   12:
        8:   13:std::string Class1::fizzBuzz( int counter )
        -:   14:{
        -:   15:          //  Input a number between 1 and 100.
        -:   16:          //  If the number is divisible by 3, output "FIZZ".
        -:   17:          //  If the number is divisible by 5, output "BUZZ".
        -:   18:          //  If the number is divisible by both 3 and 5, output "FIZZBUZZ".
        -:   19:          //  For all other numbers, output the original number.
        -:   20:
        8:   21:                 std::string resultStr;                         //  declare the variable
        -:   22:
        8:   23:                  resultStr = std::to_string(counter);
        -:   24:
        8:   25:                  if(counter % 3 == 0) {     //  divisible by 3
        3:   26:                     resultStr = "FIZZ";
        -:   27:                  }
        -:   28:
        8:   29:                  if(counter % 5 == 0) {     //  divisible by 5
        2:   30:                     resultStr = "BUZZ";
        -:   31:                  }
        -:   32:
        8:   33:                  if( (counter % 3 == 0) && (counter % 5 == 0) ) {     //  divisible by both 3 and  5
        1:   34:                     resultStr =  "FIZZBUZZ";
        -:   35:                  }
        -:   36:
        8:   37:                return resultStr;
        -:   38:
        -:   39:
        -:   40:        //  end      string fizzBuzz
    =====:   41:}
        -:   42:
        -:   43:
        8:   44:Class1::~Class1()
        -:   45:{
        8:   46:}
        -:   47:
```

Use the dotnet reportgenerator tool to create the coverage report for testFizzBuzzCpp.cpp.

C:\ ... \gtest> reportgenerator -reports:fizzBuzzCppGtest.cpp.gcov -targetdir:coverage-report1 -reporttypes:"Html_Dark"

```
2025-03-16T15:09:58: Arguments
2025-03-16T15:09:58:  -reports:fizzBuzzCppGtest.cpp.gcov
2025-03-16T15:09:58:  -targetdir:coverage-report1
2025-03-16T15:09:58:  -reporttypes:Html_Dark
2025-03-16T15:09:58:   No source directories supplied for 'GCov' coverage file
2025-03-16T15:09:58: Writing report file 'coverage-report1\index.html'
```

Use the dotnet reportgenerator tool to create the coverage report for Class1.cpp.

C:\ ... \gtest> reportgenerator -reports:Class1.cpp.gcov -targetdir:coverage-report2 -reporttypes:"Html_Dark"

```
2025-03-16T15:10:37: Arguments
2025-03-16T15:10:37:  -reports:Class1.cpp.gcov
2025-03-16T15:10:37:  -targetdir:coverage-report2
2025-03-16T15:10:37:  -reporttypes:Html_Dark
2025-03-16T15:10:37:   No source directories supplied for 'GCov' coverage file
2025-03-16T15:10:37: Writing report file 'coverage-report2\index.html'
```

We wrote a coverage report for fizzBuzzCppGtest.cpp to **coverage-report1\index.html** . We wrote a coverage report for Class1.cpp to **coverage-report2\index.html** . We can open the files in a browser.

18 Coverage Report

The coverage reports created by reportgenerator show that we have the following coverage:

```
File(s):        fizzBuzzCppGtest.cpp

Covered lines:  28
Uncovered lines:        0
Coverable lines:        28
Total lines:    68
Line coverage:  100%
```

We have 0 uncovered lines of code. We have achieved 100% line coverage.

Let's look at the coverage for our library.

```
File(s):        ../library/Class1.cpp

Covered lines:  14
Uncovered lines:        1
Coverable lines:        15
Total lines:    47
Line coverage:  93.3%
```

We have 1 uncovered line of code.

```
                36
        8       37                              return resultStr;
                38
                39
                40                      //  end      string fizzBuzz
        0       41              }
                42
                43
```

The information can also be seen in the gcov file for Class1.cpp (Class1.cpp.gcov).

```
        -:   36:
        8:   37:                return resultStr;
        -:   38:
        -:   39:
        -:   40:        //  end      string fizzBuzz
    =====:   41:}
        -:   42:
        -:   43:
```

The closing brace at line 41 is not covered due to the return statement immediately preceding it. We have effectively achieved 100% line coverage.

This page intentionally left blank.

References

[1] G. Warren *et al.*, “Overview of .NET Framework.” *Microsoft Docs*, 21-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview. [Accessed: 21-Jan-2021].

[2] G. Warren *et al.*, “What is managed code?” *Microsoft Docs*, 20-Jun-2016. [Online].Available: https://docs.microsoft.com/en-us/dotnet/standard/managed-code. [Accessed: 21-Jan-2021].

[3] G. Warren *et al.*, “Common Language Runtime (CLR) overview.” *Microsoft Docs*, 22-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/clr. [Accessed: 28-Jan-2021].

[4] G. Warren, “gewarren - Overview,” *GitHub*. [Online]. Available: https://github.com/gewarren. [Accessed: 22-Jan-2021].

[5] B. Wagner, “BillWagner - Overview,” *GitHub*. [Online]. Available: https://github.com/BillWagner. [Accessed: 26-Jan-2021].

[6] D. Lee, “DennisLee-DennisLee - Overview,” *GitHub*. [Online]. Available: https://github.com/DennisLee-DennisLee. [Accessed: 10-Feb-2021].

[7] nxtn, “nxtn - Overview,” *GitHub*. [Online]. Available: https://github.com/nxtn. [Accessed: 10-Feb-2021].

[8] J. Kotas, “jkotas - Overview,” *GitHub*. [Online]. Available: https://github.com/jkotas. [Accessed: 10-Feb-2021].

[9] M. Wenzel, “mairaw - Overview,” *GitHub*. [Online]. Available: https://github.com/mairaw. [Accessed: 10-Feb-2021].

[10] N. Schonning, “nschonni - Overview,” *GitHub*. [Online]. Available: https://github.com/nschonni. [Accessed: 10-Feb-2021].

[11] A. Aymeric, “nemrism - Overview,” *GitHub*. [Online]. Available: https://github.com/nemrism. [Accessed: 10-Feb-2021].

[12] C. Maddock, “ChrisMaddock - Overview,” *GitHub*. [Online]. Available: https://github.com/ChrisMaddock. [Accessed: 10-Feb-2021].

[13] xaviex, “xaviex - Overview,” *GitHub*. [Online]. Available: https://github.com/xaviex. [Accessed: 10-Feb-2021].

[14] M. Jones, “Mikejo5000 - Overview,” *GitHub*. [Online]. Available: https://github.com/Mikejo5000. [Accessed: 10-Feb-2021].

[15] L. Latham, “guardrex - Overview,” *GitHub*. [Online]. Available: https://github.com/guardrex. [Accessed: 10-Feb-2021].

[16] tompratt-AQ, “tompratt-AQ - Overview,” *GitHub*. [Online]. Available: https://github.com/tompratt-AQ. [Accessed: 10-Feb-2021].

[17] yishengjin1413, “yishengjin1413 - Overview,” *GitHub*. [Online]. Available: https://github.com/yishengjin1413. [Accessed: 10-Feb-2021].

[18] S. Hoag, “stevehoag - Overview,” *GitHub*. [Online]. Available: https://github.com/stevehoag. [Accessed: 10-Feb-2021].

[19] P. Onderka, “svick - Overview,” *GitHub*. [Online]. Available: https://github.com/svick. [Accessed: 10-Feb-2021].

[20] T. Dykstra, “tdykstra - Overview,” *GitHub*. [Online]. Available: https://github.com/tdykstra. [Accessed: 10-Feb-2021].

[21] A. De George, “Desktop Guide (Windows Forms .NET).” *Microsoft Docs*, 26-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/. [Accessed: 19-Jan-2021].

[22] A. De George and M. Wenzel, “Tutorial: Create a new WinForms app (Windows Forms .NET).” *Microsoft Docs*, 26-Oct-2020. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/get-started/create-app-visual-studio. [Accessed: 28-Jan-2021].

[23] A. De George, “adegeo - Overview,” *GitHub*. [Online]. Available: https://github.com/adegeo. [Accessed: 22-Jan-2021].

[24] O. Altunyan, G. Hogenson, D. Coulter, Y. Victor, and T. Lee, “Create a Windows Forms app with C# - Visual Studio.” *Microsoft Docs*, 26-Sep-2019. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio. [Accessed: 28-Jan-2021].

[25] O. Altunyan, “ornellaalt - Overview,” *GitHub*. [Online]. Available: https://github.com/ornellaalt. [Accessed: 22-Jan-2021].

[26] E. Nakamura, “eddynaka - Overview,” *GitHub*. [Online]. Available: https://github.com/eddynaka. [Accessed: 13-Feb-2021].

[27] D. Mabee, “damabe - Overview,” *GitHub*. [Online]. Available: https://github.com/damabe. [Accessed: 13-Feb-2021].

[28] A. Dev, “amaldevv - Overview,” *GitHub*. [Online]. Available: https://github.com/amaldevv. [Accessed: 13-Feb-2021].

[29] M. Blome, “mikeblome - Overview,” *GitHub*. [Online]. Available: https://github.com/mikeblome. [Accessed: 14-Feb-2021].

[30] A. Sal, “abdullahsalem - Overview,” *GitHub*. [Online]. Available: https://github.com/abdullahsalem. [Accessed: 14-Feb-2021].

[31] M. Koudelka, “v-makoud - Overview,” *GitHub*. [Online]. Available: https://github.com/v-makoud. [Accessed: 14-Feb-2021].

[32] B. Wagner, “.NET documentation.” *Microsoft Docs*, 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/. [Accessed: 14-Feb-2021].

[33] G. Hogenson, “ghogen - Overview,” *GitHub*. [Online]. Available: https://github.com/ghogen. [Accessed: 15-Feb-2021].

[34] Y. Victor, “Youssef1313 - Overview,” *GitHub*. [Online]. Available: https://github.com/Youssef1313. [Accessed: 15-Feb-2021].

References (continued)

[35] T. Lee, “TerryGLee - Overview,” *GitHub*. [Online]. Available: https://github.com/TerryGLee. [Accessed: 15-Feb-2021].

[36] D. Coulter, “DCtheGeek - Overview,” *GitHub*. [Online]. Available: https://github.com/DCtheGeek. [Accessed: 04-Feb-2021].

[37] “About Mono | Mono.” *Mono Project*. [Online]. Available: https://www.mono-project.com/docs/about-mono/. [Accessed: 27-Feb-2021].

[38] “Home | Mono.” *Mono Project*, 2021. [Online]. Available: https://www.mono-project.com/. [Accessed: 28-Feb-2021].

[39] .NET Foundation and Contributors, “.NET Foundation.” *.NET Foundation*, 2021. [Online]. Available: https://dotnetfoundation.org/. [Accessed: 28-Feb-2021].

[40] “Xamarin | Open-source mobile app platform for .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/apps/xamarin. [Accessed: 27-Feb-2021].

[41] “What is Xamarin? | .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin. [Accessed: 27-Feb-2021].

[42] “Cross-platform with Xamarin | .NET,” *Microsoft*. [Online]. Available: https://dotnet.microsoft.com/apps/xamarin/cross-platform. [Accessed: 27-Feb-2021].

[43] “ECMA-335,” *Ecma International*, 2012. [Online]. Available: https://www.ecma-international.org/publications-and-standards/standards/ecma-335/. [Accessed: 28-Feb-2021].

[44] “ECMA-334,” *Ecma International*, 2017. [Online]. Available: https://www.ecma-international.org/publications-and-standards/standards/ecma-334/. [Accessed: 28-Feb-2021].

[45] “Microsoft Announces Visual Studio 97, A Comprehensive Suite of Microsoft Visual Development Tools,” *Stories*, 28-Jan-1997. [Online]. Available: https://news.microsoft.com/1997/01/28/microsoft-announces-visual-studio-97-a-comprehensive-suite-of-microsoft-visual-development-tools/. [Accessed: 05-Mar-2021].

[46] “WineHQ - Visual Studio 2002 (7.0).” *WINE^{HQ}*. [Online]. Available: https://appdb.winehq.org/objectManager.php?sClass=version&iId=15422. [Accessed: 05-Mar-2021].

[47] P. Morlion, “The Death of WinForms Has Been Greatly Exaggerated - SubMain Blog,” *Software Quality Blog - SubMain Software*, 28-Aug-2018. [Online]. Available: https://blog.submain.com/death-winforms-greatly-exaggerated/. [Accessed: 05-Mar-2021].

[48] T. Segal, “Profit Center: Characteristics vs. a Cost Center, With Examples,” *Investopedia*, 07-Dec-2020. [Online]. Available: https://www.investopedia.com/terms/p/profitcentre.asp. [Accessed: 31-May-2024].

[49] “Test Coverage Tutorial: Comprehensive Guide With Best Practices.” *lambdatest.com*. [Online]. Available: https://www.lambdatest.com/learning-hub/test-coverage. [Accessed: 31-May-2024].

[50] H. Shah, “A Detailed Guide on Test Coverage,” *Simform - Product Engineering Company*, 28-Jun-2021. [Online]. Available: https://www.simform.com/blog/test-coverage/. [Accessed: 31-May-2024].

[51] T. Hamilton, “Test Coverage in Software Testing,” *Guru99*, 13-Apr-2024. [Online]. Available: https://www.guru99.com/test-coverage-in-software-testing.html. [Accessed: 31-May-2024].

[52] “What is DO-178B?,” *AdaCore*. [Online]. Available: https://www.adacore.com/industries/avionics/what-is-do-178b. [Accessed: 31-May-2024].

[53] D. Goswami, “Why does your Avionics Software Need DO-178B/C Compliance?,” *Qualitest*, 24-Dec-2020. [Online]. Available: https://medium.com/qualitest/why-does-your-avionics-software-need-do-178b-c-compliance-4d1cba8c94a0. [Accessed: 31-May-2024].

[54] V. Nanda, “Test Coverage in Software Testing.” *tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/test-coverage-in-software-testing. [Accessed: 31-May-2024].

[55] “Manual vs Automated Testing | Cypress Testing Tools.” *cypress.io*. [Online]. Available: https://learn.cypress.io/testing-foundations/manual-vs-automated-testing. [Accessed: 30-May-2024].

[56] “End-to-End Testing: Your First Test with Cypress | Cypress Documentation,” *Cypress Documentation*, 09-May-2024. [Online]. Available: https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test. [Accessed: 30-May-2024].

[57] “From Manual Testing to Automated Testing With Cypress - Our Story | Cogworks Blog.” *Cogworks Blog*. [Online]. Available: https://www.wearecogworks.com/innerworks/cogworks-blog-archive/from-manual-to-automated-testing-with-cypress-our-story. [Accessed: 30-May-2024].

[58] “Best Practices | Cypress Documentation,” *Cypress Documentation*, 11-Apr-2024. [Online]. Available: https://docs.cypress.io/guides/references/best-practices. [Accessed: 30-May-2024].

[59] “Testing Frameworks for Javascript | Write, Run, Debug | Cypress.” *cypress.io*. [Online]. Available: https://www.cypress.io/. [Accessed: 30-May-2024].

[60] “Comprehensive Cypress Test Automation Guide | Cypress Documentation,” *Cypress Documentation*, 11-Apr-2024. [Online]. Available: https://docs.cypress.io/guides/overview/why-cypress. [Accessed: 30-May-2024].

[61] “Key Differences | Cypress Documentation,” *Cypress Documentation*, 14-Dec-2022. [Online]. Available: https://docs.cypress.io/guides/overview/key-differences. [Accessed: 30-May-2024].

[62] “How Cypress Works | End to end and component testing tools.” *cypress.io*. [Online]. Available: https://www.cypress.io/how-it-works. [Accessed: 30-May-2024].

[63] G. Hegde, “Getting Started with Cypress Test Automation : Tutorial,” *BrowserStack*. [Online]. Available: https://browserstack.wpengine.com/guide/cypress-automation-tutorial/. [Accessed: 30-May-2024].

[64] G. Hegde and P. Bhat, “How to perform Component Testing using Cypress,” *BrowserStack*, 03-Oct-2022. [Online]. Available: https://browserstack.wpengine.com/guide/cypress-component-testing/. [Accessed: 30-May-2024].

References (continued)

[65] K. Pathak, “How to Perform Component Testing using Cypress,” *Medium*, 29-Jan-2023. [Online]. Available: https://kailash-pathak.medium.com/how-to-perform-component-testing-using-cypress-3bbe74af7492. [Accessed: 30-May-2024].

[66] F. Hric, “Component testing in Cypress: What is it and why it’s important,” *Automated Visual Testing | Applitools*, 15-Dec-2022. [Online]. Available: https://applitools.com/blog/component-testing-in-cypress-what-is-it-and-why-its-important/. [Accessed: 30-May-2024].

[67] Technocrat, “How to write Test Case in Cypress: (with testing example),” *BrowserStack*. [Online]. Available: https://browserstack.wpengine.com/guide/how-to-write-test-case-in-cypress/. [Accessed: 30-May-2024].

[68] D. Greene, “Wow! Cypress can run unit tests! 🥳,” *DEV Community*, 13-Nov-2020. [Online]. Available: https://dev.to/dgreene1/wow-cypress-can-run-unit-tests-15l5. [Accessed: 29-May-2024].

[69] “Recipes | Cypress Documentation,” *Cypress Documentation*, 11-Apr-2024. [Online]. Available: https://docs.cypress.io/examples/recipes. [Accessed: 29-May-2024].

[70] “Code Coverage | Cypress Documentation,” *Cypress Documentation*, 25-Mar-2024. [Online]. Available: https://docs.cypress.io/guides/tooling/code-coverage. [Accessed: 29-May-2024].

[71] G. Bahmutov, “Readable Cypress.io tests,” *Better world by better software*, 01-Apr-2019. [Online]. Available: https://glebbahmutov.com/blog/readable-tests/index.html. [Accessed: 29-May-2024].

[72] “cypress-example-recipes/examples/unit-testing__application-code at master · cypress-io/cypress-example-recipes,” *GitHub*. [Online]. Available: https://github.com/cypress-io/cypress-example-recipes/tree/master/examples/unit-testing__application-code. [Accessed: 29-May-2024].

[73] “Cypress Testing Framework Tutorial With Examples | Sauce Labs.” *Sauce Labs*. [Online]. Available: https://saucelabs.com/resources/blog/getting-started-with-cypress. [Accessed: 29-May-2024].

[74] Morrismoses, “Cypress For Test Automation: A Step-by-Step Guide,” *Medium*, 22-Apr-2024.[Online]. Available: https://medium.com/@morrismoses149/cypress-for-test-automation-a-step-by-step-guide-93cb47163d05. [Accessed: 29-May-2024].

[75] S. Nadeesha, “Making a simple Test Automation Framework with Cypress.io.” *LinkedIn*. [Online]. Available: https://www.linkedin.com/pulse/making-simple-test-automation-framework-cypressio-supun-nadeesha. [Accessed: 29-May-2024].

[76] “Selenium vs Cypress vs alternatives: choosing the right framework for web testing.” *zebrunner.com*. [Online]. Available: https://zebrunner.com/blog-posts/what-is-cypress-and-is-it-a-real-alternative-to-selenium. [Accessed: 29-May-2024].

[77] “How to use Cypress Testing Framework?,” *Testbytes Softwares*. [Online]. Available: https://www.testbytes.net/blog/cypress-testing/. [Accessed: 29-May-2024].

[78] “Reporters | Cypress Documentation,” 25-Mar-2024. *Cypress Documentation*. [Online]. Available: https://docs.cypress.io/guides/tooling/reporters. [Accessed: 29-May-2024].

[79] “Mocha - the fun, simple, flexible JavaScript test framework.” *mochajs.org*. [Online]. Available: https://mochajs.org/. [Accessed: 29-May-2024].

[80] “Jest vs Mocha: Which one Should You Choose?,” *GeeksforGeeks*, 27-Feb-2024. [Online]. Available: https://www.geeksforgeeks.org/jest-vs-mocha-which-one-should-you-choose/. [Accessed: 29-May-2024].

[81] “Bundled Libraries | Cypress Documentation,” *Cypress Documentation*, 15-Dec-2022. [Online]. Available: https://docs.cypress.io/guides/references/bundled-libraries. [Accessed: 29-May-2024].

[82] “Chai.” *chaijs.com*. [Online]. Available: https://www.chaijs.com/. [Accessed: 29-May-2024].

[83] D. Vasudevan, “How can you use Jest or Cypress to ensure front-end code quality?” *LinkedIn*. [Online]. Available: https://www.linkedin.com/advice/3/how-can-you-use-jest-cypress-ensure-front-end-code-k6bnf. [Accessed: 29-May-2024].

[84] “Cypress vs Jest | Top 15 Key Differences,” *Testsigma Blog*, 02-Apr-2024. [Online]. Available: https://testsigma.com/blog/cypress-vs-jest/. [Accessed: 29-May-2024].

[85] “Testing Types | Cypress Documentation,” *Cypress Documentation*, 30-Jan-2023. [Online]. Available: https://docs.cypress.io/guides/core-concepts/testing-types. [Accessed: 29-May-2024].

[86] “Karma - Spectacular Test Runner for Javascript.” [Online]. Available: https://karma-runner.github.io/latest/index.html. [Accessed: 29-May-2024].

[87] K. Choksi, “How to write unit tests with Jasmine & Karma?,” *Simform Engineering*, 01-Jun-2023. [Online]. Available: https://medium.com/simform-engineering/how-to-write-unit-tests-with-jasmine-karma-f1908bdeb617. [Accessed: 29-May-2024].

[88] Testim, “Karma JS Testing: What, Why, and How to Get Going Right Now,” *AI-driven E2E automation with code-like flexibility for your most resilient tests*, 11-Dec-2021. [Online]. Available: https://www.testim.io/blog/karma-js-testing-what-why-and-how-to-get-going-right-now/. [Accessed: 29-May-2024].

[89] “How to Install Cypress for Test Automation,” *BrowserStack*. [Online]. Available: https://browserstack.wpengine.com/guide/cypress-installation-for-test-automation/. [Accessed: 30-May-2024].

[90] “Code Coverage | Cypress Documentation,” *Cypress Documentation*, 25-Mar-2024. [Online]. Available: https://docs.cypress.io/guides/tooling/code-coverage. [Accessed: 30-May-2024].

[91] “Open the App with Cypress: Step-by-Step Guide | Cypress Documentation,” *Cypress Documentation*, 19-Apr-2024. [Online]. Available: https://docs.cypress.io/guides/getting-started/opening-the-app. [Accessed: 30-May-2024].

[92] “End-to-End Testing: Your First Test with Cypress | Cypress Documentation,” *Cypress Documentation*, 09-May-2024. [Online]. Available: https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test. [Accessed: 30-May-2024].

References (continued)

[93] “Installing Cypress and writing your first test | Cypress Testing Tools.” [Online]. Available: https://learn.cypress.io/testing-your-first-application/installing-cypress-and-writing-your-first-test. [Accessed: 30-May-2024].

[94] admin, “Cypress Code Coverage,” *testomat.io*, 07-Feb-2024. [Online]. Available: https://testomat.io/blog/cypress-code-coverage/. [Accessed: 29-May-2024].

[95] “Setting up Cypress Code Coverage,” *BrowserStack*. [Online]. Available: https://browserstack.wpengine.com/guide/cypress-code-coverage/. [Accessed: 29-May-2024].

[96] “Node.js — Run JavaScript Everywhere.” *nodejs.org*. [Online]. Available: https://nodejs.org/en. [Accessed: 31-May-2024].

[97] “Node.js Tutorial.” *W3Schools*. [Online]. Available: https://www.w3schools.com/nodejs/. [Accessed: 31-May-2024].

[98] “Node.js — Introduction to Node.js.” *nodejs.org*. [Online]. Available: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs. [Accessed: 31-May-2024].

[99] M. Wankhede, “Automation testing with Cypress, Mocha, and JavaScript,” *TO THE NEW BLOG*, 17-Feb-2023. [Online]. Available: https://www.tothenew.com/blog/automation-testing-with-cypress-mocha-and-javascript/. [Accessed: 06-Jun-2024].

[100] C. Adams, “Cypress Limitations you should be aware of,” *Codoid*, 12-Jan-2023. [Online]. Available: https://codoid.com/automation-testing/cypress-limitations-you-should-be-aware-of/. [Accessed: 06-Jun-2024].

[101] “Mocha vs Cypress.io comparison of testing frameworks.” *Knapsack Pro*. [Online]. Available: https://knapsackpro.com/testing_frameworks/difference_between/mochajs/vs/cypress-io. [Accessed: 06-Jun-2024].

[102] K. Halder, “Automated Testing with Cypress,” *The Startup*, 03-May-2020. [Online]. Available: https://medium.com/swlh/automated-testing-with-cypress-17bf74bfd97d. [Accessed: 06-Jun-2024].

[103] “Comparing cypress vs. jasmine vs. jest vs. karma vs. mocha,” *NPMCompare*. [Online]. Available: https://npmcompare.com/compare/cypress,jasmine,jest,karma,mocha. [Accessed: 06-Jun-2024].

[104] “Comparing cucumber vs. cypress vs. jasmine vs. mocha vs. nightwatch,” *NPMCompare*. [Online]. Available: https://npmcompare.com/compare/cucumber,cypress,jasmine,mocha,nightwatch. [Accessed: 06-Jun-2024].

[105] R. Kumari, “Cypress vs Cucumber - Which One to Choose?,” *Testsigma Blog*, 03-Feb-2024. [Online]. Available: https://testsigma.com/blog/cypress-vs-cucumber/. [Accessed: 06-Jun-2024].

[106] “Cucumber vs Cypress.io comparison of testing frameworks.” *Knapsack Pro*. [Online]. Available: https://knapsackpro.com/testing_frameworks/difference_between/cucumber/vs/cypress-io. [Accessed: 06-Jun-2024].

[107] “Selenium,” *Selenium*. [Online]. Available: https://www.selenium.dev/. [Accessed: 09-Jun-2024].

[108] “WebDriver,” *Selenium*, 29-Mar-2024. [Online]. Available: https://www.selenium.dev/documentation/webdriver/. [Accessed: 09-Jun-2024].

[109] “WebDriver.” *W3C*, 2018. [Online]. Available: https://www.w3.org/TR/webdriver1/. [Accessed: 09-Jun-2024].

[110] “Getting started,” *Selenium*. [Online]. Available: https://www.selenium.dev/documentation/webdriver/getting_started/. [Accessed: 09-Jun-2024].

[111] “Write your first Selenium script,” *Selenium*. [Online]. Available: https://www.selenium.dev/documentation/webdriver/getting_started/first_script/. [Accessed: 09-Jun-2024].

[112] “Supported Browsers,” *Selenium*, 20-Sep-2022. [Online]. Available: https://www.selenium.dev/documentation/webdriver/browsers/. [Accessed: 09-Jun-2024].

[113] “Downloads,” *Selenium*. [Online]. Available: https://www.selenium.dev/downloads/. [Accessed: 09-Jun-2024].

[114] “Selenium WebDriver Tutorial - javatpoint,” *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/selenium-webdriver. [Accessed: 09-Jun-2024].

[115] “WebDriver For Mobile Browsers,” *Selenium*, 12-Jan-2022. [Online]. Available: https://www.selenium.dev/documentation/legacy/selenium_2/mobile/. [Accessed: 09-Jun-2024].

[116] C. Tozzi , “Can Selenium Be Used for Mobile Testing?” *Sauce Labs Inc.*, 04-Feb-2023. [Online]. Available: https://saucelabs.com/resources/blog/can-selenium-be-used-for-mobile-testing.
[Accessed: 09-Jun-2024].

[117] “Welcome - Appium Documentation.” *Appium Documentation*. [Online]. Available: https://appium.io/docs/en/latest/. [Accessed: 10-Jun-2024].

[118] “Appium in a Nutshell - Appium Documentation.” *Appium Documentation*. [Online]. Available: https://appium.io/docs/en/latest/intro/. [Accessed: 10-Jun-2024].

[119] “Selendroid: Selenium for Android.” *selendroid.io*. [Online]. Available: http://selendroid.io/. [Accessed: 10-Jun-2024].

[120] “Mobile Testing - Selendroid Framework.” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/mobile_testing/mobile_testing_selendroid_framework.htm#:~:text=Selendroid%20is%20a%20test%20automation,the%20Selenium%20Webdriver%20client%20API. [Accessed: 10-Jun-2024].

[121] “Appium vs Selendroid,” *GeeksforGeeks*, 10-Dec-2023. [Online]. Available: https://www.geeksforgeeks.org/appium-vs-selendroid/. [Accessed: 10-Jun-2024].

[122] adegeo, “Install .NET on Windows - .NET,” *Microsoft Learn*, 20-Jun-2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/install/windows. [Accessed: 29-Aug-2024].

References (continued)

[123] tdykstra, “.NET CLI,” 20-Jun-2024. *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/tools/. [Accessed: 29-Aug-2024].

[124] tdykstra, “dotnet command - .NET CLI,” *Microsoft Learn*, 11-Oct-2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet. [Accessed: 29-Aug-2024].

[125] tdykstra, “dotnet test command - .NET CLI,” *Microsoft Learn*, 27-Mar-2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/tools/dotnet-test. [Accessed: 29-Aug-2024].

[126] “Using xUnit to Test your C# Code,” *Auth0 - Blog*. [Online]. Available: https://auth0.com/blog/xunit-to-test-csharp-code/. [Accessed: 29-Aug-2024].

[127] B. Code, “Unit Testing in C# with xUnit: Complete guide,” *Medium*, 11-Apr-2024. [Online]. Available: https://medium.com/@codebob75/unit-testing-in-c-with-xunit-complete-guide-18ee2b919b05. [Accessed: 29-Aug-2024].

[128] “Getting started: .NET Core with command line,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/getting-started/v2/netcore/cmdline. [Accessed: 29-Aug-2024].

[129] “Getting started: .NET Framework with command line,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/getting-started/v2/netfx/cmdline. [Accessed: 29-Aug-2024].

[130] “Capturing Output,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/capturing-output. [Online]. [Accessed: 29-Aug-2024].

[131] erikdietrich, “The history of C#,” *Microsoft Learn*, 07-Mar-2024. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history. [Accessed: 29-Aug-2024].

[132] “Download - Stable | Mono.” [Online]. Available: https://www.mono-project.com/download/stable/. [Accessed: 29-Aug-2024].

[133] “Microsoft Offloads The Mono Project To Wine.” [Online]. Available: https://www.phoronix.com/news/Microsoft-Gives-Mono-To-Wine. [Accessed: 29-Aug-2024].

[134] “How YOU can get started with .NET Core and C# in VS Code.” [Online]. Available: https://softchris.github.io/pages/dotnet-core.html#resources. [Accessed: 30-Aug-2024].

[135] “Home,” *xUnit.net*. [Online]. Available: https://xunit.net/. [Accessed: 03-Sept-2024].

[136] “Apache License, Version 2.0,” *Open Source Initiative*. [Online]. Available: https://opensource.org/license/apache-2-0. [Accessed: 03-Sept-2024].

[137] “Why Did We Build xUnit 1.0?,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/why-did-we-build-xunit-1.0. [Accessed: 03-Sept-2024].

[138] “Using xUnit to Test your C# Code,” *Auth0 - Blog*. [Online]. Available: https://auth0.com/blog/xunit-to-test-csharp-code/. [Accessed: 03-Sept-2024].

[139] “Getting Started: .NET Framework with Visual Studio,” *xUnit.net*. [Online]. Available: https://xunit.net/docs/getting-started/v2/netfx/visual-studio. [Accessed: 03-Sept-2024].

[140] S. Dhakne, “Why Should You Use xUnit? A Unit Testing Framework For .NET.” [Online]. Available: https://www.clariontech.com/blog/why-should-you-use-xunit-a-unit-testing-framework-for-.net. [Accessed: 03-Sept-2024].

[141] “NUnit vs xUnit vs MSTest: .NET Unit Testing Framework Comparison.” [Online]. Available: https://daily.dev/blog/nunit-vs-xunit-vs-mstest-net-unit-testing-framework-comparison. [Accessed: 03-Sept-2024].

[142] A. Kumar, “Unit Testing with xUnit.” [Online]. Available: https://www.c-sharpcorner.com/article/unit-testing-with-xunit-in-net/. [Accessed: 03-Sept-2024].

[143] “Testing with the xUnit Framework - Overview (2 of 12),” *Microsoft Learn*, 29-Nov-2022. [Online]. Available: https://learn.microsoft.com/en-us/shows/visual-studio-toolbox/testing-with-the-xunit-framework-overview-2-of-12-automated-software-testing. [Accessed: 03-Sept-2024].

[144] “Testing with the xUnit Framework - Overview (2 of 12),” *Microsoft Learn*, 29-Nov-2022. [Online]. Available: https://learn.microsoft.com/en-us/shows/visual-studio-toolbox/testing-with-the-xunit-framework-overview-2-of-12-automated-software-testing. [Accessed: 03-Sept-2024].

[145] V. Magalhães, “Test Coverage Analysis with Coverlet in .NET,” *Medium*, 23-Sept-2023. [Online]. Available: https://victormagalhaes-dev.medium.com/test-coverage-analysis-with-coverlet-in-net-2e38df3c6ed7. [Accessed: 03-Sept-2024].

[146] G. Barré, “Computing code coverage for a .NET project - Gérald Barré,” *Meziantou’s blog*, 15-Apr-2024. [Online]. Available: https://www.meziantou.net/computing-code-coverage-for-a-dotnet-project.htm. [Accessed: 03-Sept-2024].

[147] gewarren, “dotnet-coverage code coverage tool - .NET CLI - .NET”, 24-Feb-2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/additional-tools/dotnet-coverage. [Accessed: 04-Sept-2024].

[148] “ReportGenerator - Code coverage reports.” [Online]. Available: https://reportgenerator.io/. [Accessed: 04-Sept-2024].

[149] E. Dahl, "Code Coverage Reports for .NET Projects.” [Online]. Available: https://knowyourtoolset.com/2024/01/coverage-reports/. [Accessed: 04-Sept-2024].

[150] “Code Coverage in .NET,” *my tech ramblings*, 10-Jul-2024. [Online]. Available: https://www.mytechramblings.com/posts/code-coverage-in-dotnet/. [Accessed: 04-Sept-2024].

[151] C. Allen, “Generating Code Coverage Metrics for .NET Framework Applications,” *Coding with Calvin - Calvin Allen*, 18-Aug-2022. [Online]. Available: https://www.codingwithcalvin.net/generating-code-coverage-metrics-for-net-framework-applications/. [Accessed: 04-Sept-2024].

[152] J. Carracedo, “Code coverage in .Net Core,” *Medium*, 09-May-2021. [Online]. Available: https://jke94.medium.com/code-coverage-in-net-core-840bcd96b1e9. [Accessed: 04-Sept-2024].

References (continued)

[153] “NET Core Code Coverage,” *DEV Community*, 07-Mar-2021. [Online]. Available: https://dev.to/eduardstefanescu/net-core-code-coverage-1dfn. [Accessed: 04-Sept-2024].

[154] L. Bennett, “9 Best Code Coverage Tools for Java, Python, C, C++, C#, .NET,” 13-Aug-2024. [Online]. Available: https://www.guru99.com/code-coverage-tools.html. [Accessed: 04-Sept-2024].

[155] “Cobertura.” [Online]. Available: https://cobertura.github.io/cobertura/. [Accessed: 04-Sept-2024].

[156] R. Singh, “What is Cobertura and use cases of Cobertura ?,” *DevOpsSchool.com*, 15-Feb-2024. [Online]. Available: https://www.devopsschool.com/blog/what-is-cobertura-and-use-cases-of-cobertura/. [Accessed: 04-Sept-2024].

[157] “ReportGenerator - Code coverage reports.” [Online]. Available: https://reportgenerator.io/usage. [Accessed: 04-Sept-2024].

[158] DeDiv-VR, “Visual Studio 2022 System Requirements,” *Microsoft Learn*, 09-Apr-2024. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/releases/2022/system-requirements. [Accessed: 30-Sept-2024].

[159] “Download Visual Studio Tools - Install Free for Windows, Mac, Linux,” *Visual Studio*. [Online]. Available: https://visualstudio.microsoft.com/downloads/. [Accessed: 30-Sept-2024].

[160] DeDiv-VR, “Visual Studio 2019 System Requirements,” *Microsoft Learn*, 22-Jan-2024. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/releases/2019/system-requirements. [Accessed: 30-Sept-2024].

[161] DeDiv-VR, “Visual Studio 2017 System Requirements,” 22-Jan-2024. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/releases/2017/vs2017-system-requirements-vs. [Accessed: 30-Sept-2024].

[162] “Requirements for Visual Studio Code.” [Online]. Available: https://code.visualstudio.com/docs/supporting/requirements. [Accessed: 30-Sept-2024].

[163] K. Buzdar, “How to Install Visual Studio Code on Windows,” *Ultahost Knowledge Base*, 25-May-2024. [Online]. Available: https://ultahost.com/knowledge-base/install-visual-studio-code-windows/. [Accessed: 30-Sept-2024].

[164] “Configure Visual Studio Code for Microsoft C++.” [Online]. Available: https://code.visualstudio.com/docs/cpp/config-msvc. [Accessed: 30-Sept-2024].

[165] TylerMSFT, “Use the Microsoft C++ toolset from the command line,” 02-Mar-2023. [Online]. Available: https://learn.microsoft.com/en-us/cpp/build/building-on-the-command-line?view=msvc-170. [Accessed: 30-Sept-2024].

[166] “Developer Community.” [Online]. Available: https://developercommunity.visualstudio.com/t/always-show-it-looks-like-you-dont-have-enough-dis/26279. [Accessed: 30-Sept-2024].

[167] “C/C++ - Visual Studio Marketplace.” [Online]. Available: https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools. [Accessed: 30-Sept-2024].

[168] “Get Started with C++ and Windows Subsystem for Linux in Visual Studio Code.” [Online]. Available: https://code.visualstudio.com/docs/cpp/config-wsl. [Accessed: 30-Sept-2024].

[169] “WSL - Visual Studio Marketplace.” [Online]. Available: https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl. [Accessed: 30-Sept-2024].

[170] craigloewen-msft, “Install WSL,” 28-Aug-2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/wsl/install. [Accessed: 30-Sept-2024].

[171] “C++ programming with Visual Studio Code.” [Online]. Available: https://code.visualstudio.com/docs/languages/cpp. [Accessed: 30-Sept-2024].

[172] “Download Visual Studio Code - Mac, Linux, Windows.” [Online]. Available: https://code.visualstudio.com/Download. [Accessed: 30-Sept-2024].

[173] “MSYS2.” [Online]. Available: https://www.msys2.org/. [Accessed: 02-Oct-2024].

[174] “MSYS2 - Updating MSYS2.” [Online]. Available: https://www.msys2.org/docs/updating/. [Accessed: 02-Oct-2024].

[175] “MSYS2 - Package Management.” [Online]. Available: https://www.msys2.org/docs/package-management/. [Accessed: 02-Oct-2024].

[176] “pacman/Tips and tricks - ArchWiki.” [Online]. Available: https://wiki.archlinux.org/title/Pacman/Tips_and_tricks. [Accessed: 02-Oct-2024].

[177] S. Ifti, “How to Install GCC and GDB on Windows Using MSYS2 — Tutorial,” *Medium*, 23-Nov-2023. [Online]. Available: https://sajidifti.medium.com/how-to-install-gcc-and-gdb-on-windows-using-msys2-tutorial-0fceb7e66454. [Accessed: 02-Oct-2024].

[178] “Compiling with g++,” *GeeksforGeeks*, 19-Feb-2019. [Online]. Available: https://www.geeksforgeeks.org/compiling-with-g-plus-plus/. [Accessed: 07-Dec-2024].

[179] “An Overview of Functions .” [Online]. Available: https://www.umsl.edu/~lawtonb/224/functions4.html. [Accessed: 07-Dec-2024].

[180] P. Shen, “Code coverage testing of C/C++ projects using Gcov and LCOV,” *Medium*, 28-Jan-2022. [Online]. Available: https://medium.com/@xianpeng.shen/use-gcov-and-lcov-to-perform-code-coverage-testing-for-c-c-projects-c85708b91c78. [Accessed: 07-Dec-2024].

[181] “Code coverage with lcov — The Vector Packet Processor v22.10-0-g07e0c05e6 documentation.” [Online]. Available: https://s3-docs.fd.io/vpp/22.10/developer/extras/lcov.html. [Accessed: 07-Dec-2024].

[182] Naveenkhasyap, "Generating Code Coverage Report Using GNU Gcov & Lcov.," *DEV Community*, 16-Feb-2024. [Online]. Available: https://dev.to/naveenkhasyap/generating-code-coverage-report-using-gnu-gcov-lcov-59p. [Accessed: 07-Dec-2024].

References (continued)

[183] S. Jobing, “Code Coverage Analysis with GCC.” *LinkedIn*. [Online]. Available: https://www.linkedin.com/pulse/code-coverage-analysis-sander-jobing. [Accessed: 07-Dec-2024].

[184] “GoogleTest User’s Guide,” *GoogleTest*. [Online]. Available: http://google.github.io/googletest/. [Accessed: 20-Dec-2024].

[185] “GoogleTest Primer,” *GoogleTest*. [Online]. Available: http://google.github.io/googletest/primer.html. [Accessed: 20-Dec-2024].

[186] “Assertions Reference,” *GoogleTest*. [Online]. Available: http://google.github.io/googletest/reference/assertions.html. [Accessed: 20-Dec-2024].

[187] “GTest Framework,” *GeeksforGeeks*, 25-Jan-2021. [Online]. Available: https://www.geeksforgeeks.org/gtest-framework/. [Accessed: 20-Dec-2024].

[188] C. Sethi, “Introduction to Unit Testing with Google Test (GTest) in C++,” *Medium*, 28-May-2023. [Online]. Available: https://medium.com/@chittaranjansethi/introduction-to-unit-testing-with-google-test-gtest-in-c-344a89b8eb4. [Accessed: 20-Dec-2024].

[189] TylerMSFT, “Create and run unit tests with Google Test for C++ - Visual Studio (Windows),” *Microsoft Learn*, 12-Jan-2024. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/test/how-to-use-google-test-for-cpp?view=vs-2022. [Accessed: 20-Dec-2024].

[190] O. S, “Introduction to GoogleTest (gtest),” *Medium*, 14-Dec-2022. [Online]. Available: https://levelup.gitconnected.com/introduction-to-googletest-gtest-1ece21f55d0e. [Accessed: 20-Dec-2024].

[191] B. Stroustrup, "Bjarne Stroustrup’s Homepage.” [Online]. Available: https://www.stroustrup.com/. [Accessed: 22-Mar-2025].

[192] B. Stroustrup, “Bjarne Stroustrup’s FAQ,” *Bjarne Stroustrup’s FAQ*, 26-May-2024. [Online]. Available: https://www.stroustrup.com/bs_faq.html#invention. [Accessed: 22-Mar-2025].

[193] B. Stroustrup, “Thriving in a crowded and changing world: C++ 2006–2020,” *Proc. ACM Program. Lang.*, vol. 4, no. HOPL, pp. 1–168, Jun. 2020, doi: 10.1145/3386320. [Online]. Available: https://dl.acm.org/doi/10.1145/3386320. [Accessed: 22-Mar-2025].

[194] “C++ Introduction.” *www.w3schools.com*. [Online]. Available: https://www.w3schools.com/cpp/cpp_intro.asp. [Accessed: 22-Mar-2025].

[195] TylerMSFT, “Microsoft C/C++ Documentation." *Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/cpp/?view=msvc-170. [Accessed: 22-Mar-2025].

This page intentionally left blank.

End of document