Source Code Samples - FizzBuzz

1 Introduction

When you apply for a job, companies will sometimes ask to see projects and software source code that you have written. This is the source code for a demo of the game FizzBuzz written in various programming languages. It illustrates basic iteration and conditional statements.

How does FizzBuzz work?

Output the numbers from 1 to 100.

1. If the number is divisible by 3, display the word "FIZZ".

2. If the number is divisible by 5, display the word "BUZZ".

3. If the number is divisible by both 3 and 5, display the word "FIZZBUZZ".

4. If the number is NOT divisible by either 3 or 5, display the original number.

2 Ada

```
Ada


--  Ada, which first appeared in 1980, is a strongly typed language which
--  evolved from Pascal and other languages. Ada was originally used
--  in embedded system programming. It is used for safety critical
--  development such as aviation and aerospace. Avionics software
--  development sometimes uses Ada in conjunction with DO-178B "Software
--  Considerations in Airborne Systems and Equipment Certification" ( 1992 ) or the
--  later DO-178C ( 2011 ) to maintain product safety. Ada was named after Ada
--  Lovelace (1815–1852).
--
--  This code sample was compiled using the GNAT Compiler ( x86_64-linux-gnu-gcc-9 ),
--  which is part of the GNU Compiler Collection (GCC). The executable version of the
--  software was created using GNATMAKE ( x86_64-linux-gnu-gnatmake-9 ).
--  These tools were available online at
--
--          https://onecompiler.com/ada
--


with
Ada.Text_IO,
Ada.Command_Line;

use
Ada.Text_IO,
Ada.Command_Line;


procedure Main is

   count: Integer := 0;

begin

    if(Argument_Count > 0) then
             Ada.Text_IO.Put_Line ("FizzBuzz:");
             Ada.Text_IO.Put_Line ("Output the numbers 1 to 100.");
             Ada.Text_IO.Put_Line ("If the number is divisible by 3, output FIZZ.");
             Ada.Text_IO.Put_Line ("If the number is divisible by 5, output BUZZ.");
             Ada.Text_IO.Put_Line ("If the number is divisible by both 3 and 5, output FIZZBUZZ.");
    else
             --  Argument_Count = 0
             Ada.Text_IO.Put_Line ("FizzBuzz");
             Ada.Text_IO.Put_Line ("For help, type FizzBuzz -h.");
             Ada.Text_IO.Put_Line ("");
             Ada.Text_IO.Put_Line ("FizzBuzz results:");


             for count in 1 .. 100 loop
                   Ada.Text_IO.Put ("The count is " & Integer'Image(count) & "   ");

                   if count mod 3 = 0 then      -- divisible by 3
                         Ada.Text_IO.Put ("FIZZ");
                      end if;

                   if count mod 5 = 0 then     -- divisible by 5
                         Ada.Text_IO.Put ("BUZZ");
                      end if;

                   Ada.Text_IO.Put_Line ("");
               end loop;


       end if;    --  Argument_Count = 0


       Ada.Text_IO.Put_Line ("End of program.");


 end Main;
```

3 C

```
#include <stdio.h>

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

   if(argc > 1) {
      printf ("FizzBuzz:\n");
      printf ("Output the numbers 1 to 100.\n");
      printf ("If the number is divisible by 3, output FIZZ.\n");
      printf ("If the number is divisible by 5, output BUZZ.\n");
      printf ("If the number is divisible by both 3 and 5, output FIZZBUZZ.\n");
   }
   else
   {
      printf ("FizzBuzz\n");
      printf ("For help, type FizzBuzz -h.\n");
      printf ("\n");
      printf ("FizzBuzz results:\n");

      for (int count = 1; count <= 100; count++) {

         printf ("The count is %d   ", count);

         if(count % 3 == 0) {     //  divisible by 3
            printf ("FIZZ");
         }

         if(count % 5 == 0) {     //  divisible by 5
            printf ("BUZZ");
         }

         printf ("\n");

         // end   for loop
      }

      //  end   else
   }

   printf ("End of program.\n");

   //   end     main
}
```

4 C++

```
#include <iostream>


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

    if(argc > 1) {
        std::cout <<  "FizzBuzz:\n";
        std::cout <<  "Output the numbers 1 to 100.\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";
    }
    else
    {
        std::cout <<  "FizzBuzz\n";
        std::cout <<  "For help, type FizzBuzz -h.\n";
        std::cout <<  "\n";
        std::cout <<  "FizzBuzz results:\n";

        for (int count = 1; count <= 100; count++) {

            std::cout <<  "The count is " << count << "   ";

            if(count % 3 == 0) {     //  divisible by 3
                std::cout <<  "FIZZ";
            }

            if(count % 5 == 0) {     //  divisible by 5
                std::cout <<  "BUZZ";
            }

            std::cout <<  "\n";

            // end   for loop
        }

        //  end   else
    }

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

    return 0;

    //  end     main
}
```

5 C#

```
class FizzBuzz
{
   static void Main(string[] args) {

       if(args.Length > 0) {

          Console.WriteLine("FizzBuzz:");
          Console.WriteLine("Output the numbers 1 to 100.");
          Console.WriteLine("If the number is divisible by 3, output FIZZ.");
          Console.WriteLine("If the number is divisible by 5, output BUZZ.");
          Console.WriteLine("If the number is divisible by both 3 and 5, output FIZZBUZZ.");

       }
       else
       {
          Console.WriteLine("FizzBuzz");
          Console.WriteLine("For help, type FizzBuzz -h.");
          Console.WriteLine();
          Console.WriteLine("FizzBuzz results:");

          for (int count = 1; count <= 100; count++) {

             Console.Write("The count is " +
                count.ToString() + "   ");

             if(count % 3 == 0) {     //  divisible by 3
                Console.Write("FIZZ");
             }

             if(count % 5 == 0) {     //  divisible by 5
                Console.Write("BUZZ");
             }

             Console.WriteLine();

             // end   for loop
          }

          //  end   else
       }

       Console.WriteLine("End of program.");

      //  end     void Main
   }

   //   end     class FizzBuzz
}
```

6 Java

```
public class FizzBuzz {
   public static void main(String[] args) {

      if(args.length > 0) {
         System.out.println("FizzBuzz:");
         System.out.println("Output the numbers 1 to 100.");
         System.out.println("If the number is divisible by 3, output FIZZ.");
         System.out.println("If the number is divisible by 5, output BUZZ.");
         System.out.println("If the number is divisible by both 3 and 5, output FIZZBUZZ.");
      }
      else
      {
         System.out.println("FizzBuzz");
         System.out.println("For help, type FizzBuzz -h.");
         System.out.println();
         System.out.println("FizzBuzz results:");

         for (int count = 1; count <= 100; count++) {

            System.out.print("The count is " +
               Integer.toString(count) + "   ");

            if(count % 3 == 0) {     //  divisible by 3
               System.out.print("FIZZ");
            }

            if(count % 5 == 0) {     //  divisible by 5
               System.out.print("BUZZ");
            }

            System.out.println();

            // end   for loop
         }

         System.out.println("End of program.");

         //  end   else
      }

      //  end     void main
   }

   //  end class FizzBuzz
}
```

7 JavaScript

```
var text2display = "";

var counter= 0;     // counter fom 1 to 100

var divisibleBy3 = false;     // variable used to improve code readability

var divisibleBy5 = false;     // variable used to improve code readability

// begin building the text to display
text2display =
"Results of the code execution." +
"<br><br>";

// Set up the loop
for (counter = 1; counter <= 100; counter++) {

   divisibleBy3 = false;     // initialize vars to false
   divisibleBy5 = false;

   // display the count
   text2display = text2display +
   "The count is " +  counter;

   // is the count divisible by 3?
   if ( (counter%3) === 0 )
   {
      divisibleBy3 = true;
   }

   // is the count divisible by 5?
   if ( (counter%5) === 0 )
   {
      divisibleBy5 = true;
   }

   // count is divisible by BOTH 3 and 5
   if ( divisibleBy3 && divisibleBy5 ) {
      //  Start   divisible by both 3 and 5

      // output the string FIZZBUZZ
      text2display = text2display +
      '   ' +
      '<b>' +
      "FIZZBUZZ" +
      '</b>';

      //  End   divisible by both 3 and 5
   }

   // count is divisible by 3 only
   if ( divisibleBy3 && !divisibleBy5 ) {
      //  Start   divisible by 3 only

      // output the string FIZZ
      text2display = text2display +
      '   ' +
      '<b>' +
      "FIZZ" +
      '</b>';

      //  End   divisible by 3 only
   }
```

(Continued on the next page.)

```
   // count is divisible by 5 only
   if ( !divisibleBy3 && divisibleBy5 ) {
      //  Start   divisible by 5 only

      // output the string BUZZ
      text2display = text2display +
      '   ' +
      '<b>' +
      "BUZZ" +
      '</b>';

      //  End   divisible by 5 only
   }


    // terminate the line of text
    text2display = text2display +
    "<br>";

    //   end for loop
}


   // terminate the line of text
   text2display = text2display +
   "<br>";

// display the text
document.getElementById("textOutput").innerHTML =
text2display + '<br>';
```

8 Pascal

```
(*
The Pascal programming language was created in the 1970s by
Niklaus Wirth, who passed away January 01, 2024. Before the
creation of Pascal, languages such as BASIC allowed the use of the
GOTO statement. This meant that program execution could jump
with almost no restrictions to any place within the code. This could
result in what people called "spaghetti code", which was difficult to
read and maintain.

Pascal was an attempt to add logical structure to the software of the
1970s. It became very popular in the 1980s with the release of Borland's
Turbo Pascal for computers compatible with the IBM Personal Computer.

Pascal predated Object-Oriented Programming, although in the 1980s a
version called Object Pascal was adopted by Apple Computer. Support for
Objects was added to Turbo Pascal in the late 1980s. Borland Delphi, which
is still used today, also included support for Objects.

This code sample was compiled using the
        Free Pascal Compiler version 3.2.2 (x86_64),
available at
        www.freepascal.org

Free Pascal supports its own version of Object Pascal, as well as Turbo Pascal /
Borland Pascal and Borland (Embarcadero) Delphi, among others.
*)

program fizzBuzz(input, output, stdErr);

uses sysUtils;

var
   count : integer;

begin    (* Main *)

    if(paramCount() > 0) then
       begin
            writeLn('FizzBuzz:');
            writeLn('Output the numbers 1 to 100.');
            writeLn('If the number is divisible by 3, output FIZZ.');
            writeLn('If the number is divisible by 5, output BUZZ.');
            writeLn('If the number is divisible by both 3 and 5, output FIZZBUZZ.');
        end    (* paramCount > 0 *)
        else
        begin    (* paramCount = 0 *)
            writeLn('FizzBuzz');
            writeLn('For help, type FizzBuzz -h.');
            writeLn();
            writeLn('FizzBuzz results:');

            for count := 1 to 100 do
               begin
                  write('The count is ', inttostr( count ), '   ');

                  if(count mod 3 = 0) then      (*  divisible by 3 *)
                     begin
                        write('FIZZ');
                     end;

                  if(count mod 5 = 0) then     (*  divisible by 5 *)
                     begin
                        write('BUZZ');
                     end;

                  writeLn();
               end;    (* end   for loop *)

         end;    (* paramCount = 0 *)

    writeLn('End of program.');

 end.    (* Main *)
```

9 PHP

```
//  Set up the loop
for ($counter = 1; $counter <= 100; $counter++) {

    //  Display the count
    echo "The count is $counter";

    //  Is the count divisible by BOTH 3 and 5?
    if ( (($counter%3) ==0) && (($counter%5) ==0) ) {
        echo "FIZZBUZZ";
    }
    elseif ( ($counter%3) ==0 ) {     //  Is the count divisible by 3?
        echo "FIZZ";
    }
    elseif ( ($counter%5) ==0 ) {     //  Is the count divisible by 5?
        echo "BUZZ";
    }

    // end for loop
}
```

10 Python

```
import sys

if len(sys.argv) > 1:
     print("FizzBuzz:")
     print("Output the numbers 1 to 100.")
     print("If the number is divisible by 3, output FIZZ.")
     print("If the number is divisible by 5, output BUZZ.")
     print("If the number is divisible by both 3 and 5, output FIZZBUZZ.")
else:
     print("FizzBuzz")
     print("For help, type FizzBuzz -h.")
     print()
     print("FizzBuzz results:")
     print()

     for count in range (1, 101):
        print("The count is ", count, "   ", end = "")

        #  divisible by 3
        if count % 3 == 0:
           print("FIZZ", end = "")

        #  divisible by 5
        if count % 5 == 0:
           print("BUZZ", end = "")

        print()


print("End of program.")
```

11 Rust

```
//  This code sample was written using the
//  Rust Playground  ( https://play.rust-lang.org )
//
//  The code was formatted using the Rustfmt tool.
//  The code was Linted using the Clippy tool.
//  Mid-level intermediate representation (MIR) verified
//  using the Miri interpreter.
//

use std::env;

fn main() {
    //  collect the command-line arguments into an array of strings
    //  the length of the array indicates the number of arguments
    //  The first argument is the path to call the program, so there
    //  are no arguments unless the count is greater than 1.

    let args: Vec = env::args().collect();

    if args.len() > 1 {
        //  there are command-line arguments, explain how the program works
        println!("FizzBuzz");
        println!("'Output the numbers 1 to 100.");
        println!("'If the number is divisible by 3, output FIZZ.");
        println!("If the number is divisible by 5, output BUZZ.");
        println!("If the number is divisible by both 3 and 5, output FIZZBUZZ.");

        //   end     command-line arguments, explain how the program works
    } else {
        //  no command-line arguments, normal program operation
        println!("FizzBuzz");
        println!("For help, type FizzBuzz -h.");
        println!();
        println!("FizzBuzz results:");

        for n in 1..=100 {
            print!("The count is {:?}   ", n);

            if n % 3 == 0 {
                //  divisible by 3
                print!("FIZZ");
            }

            if n % 5 == 0 {
                //  divisible by 5
                print!("BUZZ");
            }

            println!();     //  end the line of text

            //   end     for loop
        }

        //   end     no command-line arguments
    }

    //  Notify the user that the program has completed execution.
    println!("End of program.");

    //   end     fn main
}
```

12 80x86 Assembly Language (Win32)

```
; This is the game of FizzBuzz in 80x86 assembly language under Win32.
;
; Assemble the code using NASM.
;

        global _main
        extern _printf
        extern ExitProcess

        segment .data

        countmsg         db      'The count is ', 0
        fizzmsg          db       'Fizz', 0
        buzzmsg          db       'Buzz', 0
        crlfmsg          db       0dh,0ah,0
        exitmsg:         db       'Exiting the program ...', 0
        counter:
        byte1:           db     '0'     ; 3 bytes to display the count value
        byte2:           db     '0'
        byte3:           db     '0'
                         db     '   '   ; space for formatting
        byte4:           db      0      ; the string needs to be terminated with a 0



        ;  code
        section .text

_main:

start:
        mov cx, 1                ; initialize cx for the counter, 1 to 100

mainloop:
        push cx                  ;  save cx

        call getcount

        push cx
        push countmsg            ;  display the count message
        call _printf
        add esp, 4
        pop cx

        push cx
        call printcount          ;  display the count value
        pop cx

divisibleby3:

        mov ax, cx              ; copy the count in cx to ax to perform the Modulo operation

        mov  bl, 03h            ; is ax divisible by 3?
        div bl                        ; al = quotient  ( ax / divisor )
                                    ; ah =  remainder  ( ax MOD divisor )
        cmp ah, 00h           ; if MOD = 0, then the count is divisible by 3
        je dividesby3
```

(Continued on the next page.)

```
divisibleby5:

        mov ax, cx              ; copy the count in cx to ax to perform the Modulo operation

        mov  bl, 05h            ; is ax divisible by 5?
        div bl                       ; al = quotient  ( ax / divisor )
                                   ; ah =  remainder  ( ax MOD divisor )
        cmp  ah, 00h                ; if MOD = 0, then the count is divisible by 5
        je dividesby5


incrementindex:

        call printcrlf          ; print the  sequence to terminate the line of text


        pop cx                  ; restore the count in cx which we saved at the top of the loop

        inc cx
        cmp cx, 100                ; has the count reached 100?
        jbe mainloop            ; if the count has not exceeded 100, then repeat the loop
        jmp exit                        ; the count has exceeded 100, exit the program


dividesby3:

        call printfizz               ; divisible by 3, so print 'FIZZ'
        jmp divisibleby5        ; go check now if divisible by 5

dividesby5:

        call printbuzz                  ; divisible by 5, so print 'BUZZ'
        jmp incrementindex      ; go check now if we have reached 100


printfizz:

        push cx
        push fizzmsg
        call _printf
        add esp, 4
        pop cx
        ret

printbuzz:

        push buzzmsg
        call _printf
        add esp, 4
        ret

printcrlf:

        push crlfmsg            ; print the  sequence to terminate the line of text
        call _printf
        add esp, 4
        ret
```

(Continued on the next page.)

```
printcount:
; prints the value of the counter to the screen

        push counter
        call _printf
        add esp, 4
        ret



getcount:
; converts the count in cx to a format which
; can be displayed on the screen
; al must contain the byte to display

        mov ax, cx              ; get the count
        mov bl, 100             ; divisible by 100?
        div bl                        ; al = al / 100
        add al, 30h              ; add offset for printing to the screen
        mov [byte1], al       ; copy the value to byte1 of the counter

        sub al, 30h             ; remove the offset from al
        mov bl, 100
        mul bl                      ; al = al * 100
        mov bl, al                ; move al to bl
        mov ax, cx              ; restore the count to al
        sub al, bl                 ; subtract bl   ( al = al - (al * 100) )

        push ax                 ; save the count minus the hundreds

        mov bl, 10              ; divisible by 10?
        div bl                      ; al = al / 10
        add al, 30h             ; add offset for printing to the screen
        mov [byte2], al      ; copy the value to byte2 of the counter

        sub al, 30h             ; remove the offset from al
        mov bl, 10
        mul bl                     ; al = al * 10
        mov bl, al              ; move al to bl
        pop ax                   ; restore the count minus the hundreds
        sub al, bl               ; subtract bl   ( al = al - (al * 10) )

        add al, 30h              ; add offset for printing to the screen
        mov [byte3], al        ; copy the value to byte3 of the counter

        ret



exit:

        push exitmsg            ; display message to indicate the end of the program
        call _printf
        add esp, 4

        push crlfmsg            ; print the  sequence to terminate the line of text
        call _printf
        add esp, 4

        push 0                  ;  terminate the program
        call    ExitProcess
```

13 80x86 Assembly Language (MS-DOS)

```
; Talk about a blast from the past.
;
; This is the game of FizzBuzz in 80x86 assembly language.
; Assemble the code using NASM ( https://www.nasm.us/ ).
; It generates a COM file. COM files were a simple executable
; file format for IBM PC DOS or Microsoft MS-DOS. COM files
; are limited in that the program code and data must fit into a
; single 64K byte segment. They do not contain a header with
; information for relocation, so they are no longer supported by
; recent versions of Microsoft Windows. COM files were often used
; from the early 1980s until Microsoft Windows no longer supported
; them beginning in the early 2000's.
;
; 32-bit Windows still appears to have support for emulating
; MS-DOS, but modern Windows versions (64-bit) no longer support
; the format, so it is necessary to use an external DOS emulator
; in order to run this type of executable file.
;
; DOS emulators such as DOSBox ( https://www.dosbox.com/ ), or
; FreeDOS ( https://www.freedos.org/ ) are available for this.
;
;
;
org 100h        ; required entry point for COM executables

start:
        mov cx, 1               ; initialize cx for the counter,
                                ; going from 1 to 100

mainloop:

        call getcount           ; get the counter value and convert
                                ; it to a string for display
        lea dx, [countmsg]      ; print the count message
        call printmsg

        call printcount         ; print the counter as a string

divisibleby3:
        mov ax, cx              ; copy cx to ax to perform the Modulo operation

        mov  bl, 03h            ; is ax divisible by 3?
        div bl                  ; al = quotient  ( ax / divisor )
                                ; ah =  remainder  ( ax MOD divisor )
        cmp ah, 00h             ; if it's MOD 0, then it's divisible by 3
        je dividesby3           ; jump to the routine to print the message


divisibleby5:
        mov ax, cx              ; copy cx to ax to perform the Modulo operation

        mov  bl, 05h            ; is ax divisible by 5?
        div bl                  ; al = quotient  ( ax / divisor )
                                ; ah =  remainder  ( ax MOD divisor )
        cmp  ah, 00h            ; if it's MOD 0, then it's divisible by 5
        je dividesby5           ; jump to the routine to print the message
```

(Continued on the next page.)

```
incrementindex:
        call printcrlf          ; print the sequence  to end the line
                                ; of text

        inc cx                  ; increment the loop counter
        cmp cx, 100             ; have we reached 100 yet?
        jbe mainloop            ; not yet, so loop again
        jmp exit                ; we've reached 100, so exit the program


dividesby3:
        call printfizz          ; print "FIZZ" since it's divisible by 3
        jmp divisibleby5        ; jump to the next action to perform
                                ; ie; check if it's divisible by 5

dividesby5:
        call printbuzz          ; print "BUZZ" since it's divisible by 5
        jmp incrementindex      ; jump to the next action to perform
                                ; ie; check if we've reached 100 yet


printfizz:                      ; print the message showing divisible by 3
        lea dx, [fizzmsg]       ; the address of the message to print in dx
        call printmsg           ; call the subroutine to print the message
        ret

printbuzz:                      ; print the message showing divisible by 5
        lea dx, [buzzmsg]       ; the address of the message to print in dx
        call printmsg           ; call the subroutine to print the message
        ret

printcrlf:                      ; print the sequence  to end the line
                                ; of text
        lea dx, [crlfmsg]       ; the address of the message to print in dx
        call printmsg           ; call the subroutine to print the message
        ret



printmsg:                       ; print a message to the screen
        mov  ah, 9              ; the BDOS function to print a $ terminated
                                ; string
                                ; DX must point to the string before calling
                                ; this subroutine
        int  21h                ; call the BDOS
        ret



printcount:
; prints the value of the counter to the screen
; before calling this, we must call getcount in
; order to convert the counter value to a 3 byte string
; ( 000 to 255 )
;
        lea dx, [counter]      ;  print the counter string
        call printmsg
        ret
```

(Continued on the next page.)

```
getcount:
; Converts the count in cx to a format which
; can be displayed on the screen.
; al must contain the byte to display.
;
; The value of the byte ranges from 0 to 255.
; We store the value as a 3 byte text string
; which is terminated by a "$". We insert 3
; spaces   ( db    "   " ) before the "$" in
; order to format the text prior to announcing
; "FIZZ", "BUZZ" or "FIZZBUZZ".
;
; In order to obtain the hundreds digit, we divide
; the counter value by one hundred. We convert the
; result to a displayable character and store it as
; the first digit of the string.
;
; The we subtract the hundreds from the counter.
; For example:   255 - 200 = 55.
;
; In order to obtain the tens digit, we divide
; the new counter value by ten. We convert the
; result to a displayable character and store it
; as the second digit of the string.
;
; The we subtract the tens from the counter.
; For example:   55 - 50 = 5.
;
; We convert the result to a displayable character
; and store it as the third digit of the string.
;
;
;
        mov ax, cx              ; get the count
        mov bl, 100             ; divisible by 100?
        div bl                  ; al = al / 100
        add al, 30h             ; add offset for printing to the screen
        mov [byte1], al         ; copy the value to byte1 of the counter

        sub al, 30h             ; remove the offset from al
        mov bl, 100
        mul bl                  ; al = al * 100
        mov bl, al              ; move al to bl
        mov ax, cx              ; restore the count to al
        sub al, bl              ; subtract bl   ( al = al - 100s )

        push ax                 ; save the count minus the hundreds

        mov bl, 10              ; divisible by 10?
        div bl                  ; al = al / 10
        add al, 30h             ; add offset for printing to the screen
        mov [byte2], al         ; copy the value to byte2 of the counter

        sub al, 30h             ; remove the offset from al
        mov bl, 10
        mul bl                  ; al = al * 10
        mov bl, al              ; move al to bl

        pop ax                  ; restore the count minus the hundreds

        sub al, bl              ; subtract bl   ( al = al - tens )

        add al, 30h             ; add offset for printing to the screen
        mov [byte3], al         ; copy the value to byte3 of the counter

        ret
```

(Continued on the next page.)

```
exit:
        lea dx, [exitmsg]       ;  print the exit message since we're finished
        call printmsg

        call printcrlf          ; print the sequence  to end the line
                                ; of text

        mov  ah, 4ch            ; terminate the program
        int  21h                ; call the  BDOS


        countmsg:       db    "The count is $"
        fizzmsg:        db    "Fizz$"
        buzzmsg:        db    "Buzz$"
        exitmsg:        db    "Exiting the program.$"
        crlfmsg:        db     0dh,0ah,"$"
        counter:
        byte1:          db    "0"     ; 3 bytes to display the count value
        byte2:          db    "0"
        byte3:          db    "0"
                        db    "   "   ; used to format the text on screen
        byte4:          db    "$"     ; the string needs to be terminated with a $
```

This page intentionally left blank.

References

[1] “https://leetcode.com/problems/fizz-buzz/description,” *LeetCode*. [Online]. Available: https://leetcode.com/problems/fizz-buzz/description. [Accessed: 02-Jul-2023].

[2] “Fizz Buzz Implementation,” *GeeksforGeeks*, 23-May-2017. [Online]. Available: https://www.geeksforgeeks.org/fizz-buzz-implementation/. [Accessed: 02-Jul-2023].

This page intentionally left blank.

End of document.