Polyglottal Programing

Waqqas Ibrahim

Contents

Programming Languages

The FizzBuzz Problem is a classic programming interview question.

The task is to create an algorithm that prints the numbers `1` to `n`, such that if the integer is divisible by three *“Fizz”* is printed, if divisible by 5 *“Buzz”* is printed, and both if both are apparent.

Python

```python
def FizzBuzz(n):
  for i in range(0, n):
    res=""
    if (n % 3) == 0:
      res+="Fizz"
    if (n % 5) == 0:
      res+="Buzz"
    if len(res) > 0:
      print(res)
    n=n+1
FizzBuzz(15)
```

```
## FizzBuzz
## Fizz
## Buzz
## Fizz
## Fizz
## Buzz
## Fizz
```

JavaScript

```
const FizzBuzz = (n) => {
  for (let i = 0; i <= n; i++) {
    const res = (n % 3 == 0 ? 'Fizz' : '') + (n % 5 == 0 ? 'Buzz' : '')
    if (res) console.log(res)
  }
}
```

Typescript

```
function FizzBuzz(n: number): void {
  for (let i: number = 0; i <= n; i++) {
    const res: string = (n % 3 == 0 ? 'Fizz' : '') + (n % 5 == 0 ? 'Buzz' : '')
    if (res) console.log(res)
  }
}
```

R Programming Language [1]

```
FizzBuzz <- 1:15
output <- vector()

for (i in FizzBuzz) {
  output[i] <- ""

  if (i %% 3 == 0) { output[i] <- paste0(output[i], "Fizz") }
  if (i %% 5 == 0) { output[i] <- paste0(output[i], "Buzz") }

  if (output[i] == "") {output[i] <- i}
}

print(output)
```

```
##  [1] "1"        "2"        "Fizz"     "4"        "Buzz"     "Fizz"
##  [7] "7"        "8"        "Fizz"     "Buzz"     "11"       "Fizz"
## [13] "13"       "14"       "FizzBuzz"
```

Bash Shell Scripting

```
read n
echo "input a number"
if [[ -z ${n+1} ]]; then n = 15; fi
seq 5 | sed '0~5s/.*/Buzz/;0~3s/.*/Fizz/;0~15s/.*/FizzBuzz/'
```

[1] Assumed that n is 15

```
## input a number
## 1
## 2
## Fizz
## 4
## Buzz
```

Lua

```
repeat
  io.write("Input a number: \n")
  num = io.read()
until tonumber(num)

n = tonumber(num)
for i = 1, n, 1 do
  local res = "\n"
  if i % 3 == 0 then
    res = res .. "Fizz"
  end
  if i % 5 == 0 then
    res = res .. "Buzz"
  end
  if res == "\n" then
    res = res .. tostring(i)
  end
  io.write(res)
end
```

C Programing Language

```

#include <stdio.h>

int main(int argc, char** argv) {
    if (argc != 2) {
        printf("Need exactly one argument.");
        return -1;
    }

    int num;
    sscanf(argv[1], "%d", &num);

    for (int i = 1; i <= num; i++) {
        printf("\n");
        if (i % 3 == 0) printf("Fizz");
        if (i % 5 == 0) printf("Buzz");
        if ((i % 3 && i % 5) != 0) printf("%d", i);
    }

    return 0;
}
```

```
## gcc -I"/usr/include/R/" -DNDEBUG   -D_FORTIFY_SOURCE=2   -fpic  -march=x86-64 -mtune=generic -O2 -pi
## gcc -shared -L/usr/lib64/R/lib -Wl,-O1,--sort-common,--as-needed,-z,relro,-z,now -o c10bb914bb01dc.s
```

C++

```cpp
#include <iostream>

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

  std::cout << "Input an integer";

  int num;
  std::cin >> num;

  if (!num) {
    std::cout << "Need a number arg";
    return -1;
  }
  for (int i = 1; i <= num; i++) {
    std::cout << "\n";
    if (i % 3 == 0)
      std::cout << "Fizz";
    if (i % 5 == 0)
      std::cout << "Buzz";
    if ((i % 3 && i % 5) != 0)
      std::cout << i;
  }

  return 0;
}
```

x86-64 Assembly Language [2]

```
   0x0000000000001159 <+0>: push   rbp
   0x000000000000115a <+1>: push   rbx
   0x000000000000115b <+2>: sub    rsp,0x18
   0x000000000000115f <+6>: cmp    edi,0x2
   0x0000000000001162 <+9>: jne    0x1197 <main+62>
   0x0000000000001164 <+11>:    lea    rdx,[rsp+0xc]
   0x0000000000001169 <+16>:    mov    rdi,QWORD PTR [rsi+0x8]
   0x000000000000116d <+20>:    lea    rsi,[rip+0xeab]        # 0x201f
   0x0000000000001174 <+27>:    mov    eax,0x0
   0x0000000000001179 <+32>:    call   0x1050 <__isoc99_sscanf@plt>
   0x000000000000117e <+37>:    cmp    DWORD PTR [rsp+0xc],0x0
   0x0000000000001183 <+42>:    jle    0x1249 <main+240>
   0x0000000000001189 <+48>:    mov    ebx,0x1
   0x000000000000118e <+53>:    lea    rbp,[rip+0xe8a]        # 0x201f
   0x0000000000001195 <+60>:    jmp    0x11e8 <main+143>
   0x0000000000001197 <+62>:    lea    rdi,[rip+0xe66]        # 0x2004
   0x000000000000119e <+69>:    mov    eax,0x0
   0x00000000000011a3 <+74>:    call   0x1040 <printf@plt>
   0x00000000000011a8 <+79>:    mov    eax,0xffffffff
```

[2]Compiled from C code and disassembled

```
   0x00000000000011ad <+84>:    jmp    0x1242 <main+233>
   0x00000000000011b2 <+89>:    lea    rdi,[rip+0xe69]        # 0x2022
   0x00000000000011b9 <+96>:    mov    eax,0x0
   0x00000000000011be <+101>:   call   0x1040 <printf@plt>
   0x00000000000011c3 <+106>:   movsxd rax,ebx
   0x00000000000011c6 <+109>:   imul   rax,rax,0x66666667
   0x00000000000011cd <+116>:   sar    rax,0x21
   0x00000000000011d1 <+120>:   mov    edx,ebx
   0x00000000000011d3 <+122>:   sar    edx,0x1f
   0x00000000000011d6 <+125>:   sub    eax,edx
   0x00000000000011d8 <+127>:   lea    eax,[rax+rax*4]
   0x00000000000011db <+130>:   cmp    ebx,eax
   0x00000000000011dd <+132>:   je     0x122a <main+209>
   0x00000000000011df <+134>:   add    ebx,0x1
   0x00000000000011e2 <+137>:   cmp    DWORD PTR [rsp+0xc],ebx
   0x00000000000011e6 <+141>:   jl     0x123d <main+228>
   0x00000000000011e8 <+143>:   mov    edi,0xa
   0x00000000000011ed <+148>:   call   0x1030 <putchar@plt>
   0x00000000000011f2 <+153>:   movsxd rax,ebx
   0x00000000000011f5 <+156>:   imul   rax,rax,0x55555556
   0x00000000000011fc <+163>:   shr    rax,0x20
   0x0000000000001200 <+167>:   mov    edx,ebx
   0x0000000000001202 <+169>:   sar    edx,0x1f
   0x0000000000001205 <+172>:   sub    eax,edx
   0x0000000000001207 <+174>:   lea    eax,[rax+rax*2]
   0x000000000000120a <+177>:   cmp    ebx,eax
   0x000000000000120c <+179>:   je     0x11b2 <main+89>
   0x000000000000120e <+181>:   movsxd rax,ebx
   0x0000000000001211 <+184>:   imul   rax,rax,0x66666667
   0x0000000000001218 <+191>:   sar    rax,0x21
   0x000000000000121c <+195>:   mov    edx,ebx
   0x000000000000121e <+197>:   sar    edx,0x1f
   0x0000000000001221 <+200>:   sub    eax,edx
   0x0000000000001223 <+202>:   lea    eax,[rax+rax*4]
   0x0000000000001226 <+205>:   cmp    ebx,eax
   0x0000000000001228 <+207>:   jne    0x1250 <main+247>
   0x000000000000122a <+209>:   lea    rdi,[rip+0xdf6]        # 0x2027
   0x0000000000001231 <+216>:   mov    eax,0x0
   0x0000000000001236 <+221>:   call   0x1040 <printf@plt>
   0x000000000000123b <+226>:   jmp    0x11df <main+134>
   0x000000000000123d <+228>:   mov    eax,0x0
   0x0000000000001242 <+233>:   add    rsp,0x18
   0x0000000000001246 <+237>:   pop    rbx
   0x0000000000001247 <+238>:   pop    rbp
   0x0000000000001248 <+239>:   ret
   0x0000000000001249 <+240>:   mov    eax,0x0
   0x000000000000124e <+245>:   jmp    0x1242 <main+233>
   0x0000000000001250 <+247>:   mov    esi,ebx
   0x0000000000001252 <+249>:   mov    rdi,rbp
   0x0000000000001255 <+252>:   mov    eax,0x0
   0x000000000000125a <+257>:   call   0x1040 <printf@plt>
   0x000000000000125f <+262>:   jmp    0x11df <main+134>
```

Data Presentation and Serialisation Languages

Based on the Iris dataset from the R data science language.

‘

Edgar Anderson’s Iris Data

Description:

```
This famous (Fisher's or Anderson's) iris data set gives the
measurements in centimeters of the variables sepal length and
width and petal length and width, respectively, for 50 flowers
from each of 3 species of iris.  The species are _Iris setosa_,
_versicolor_, and _virginica_.
```

‘

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

R Data Manipulation

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

JSON

```
{
  "Sepal.Length": [5.1, 4.9, 4.7, 4.6, 5.0, 5.4],
  "Sepal.Width": [3.5, 3.0, 3.2, 3.1, 3.6, 3.9],
  "Petal.Length": [1.4, 1.4, 1.3, 1.5,, 1.4, 1.7],
  "Petal.Width": [0.2, 0.2, 0.2, 0.2, 0.2, 0.4],
  "Species": ["setosa","setosa","setosa","setosa","setosa","setosa"]
}
```

XML [3]

```xml
<Iris>
  <Sepal>
    <Length>5.1</Length>
    <Length>4.9</Length>
    <Length>4.6</Length>
    <Length>4.6</Length>
    <Length>5.0</Length>
    <Length>5.4</Length>
  </Sepal>
  <Petal>
    <Width>0.2</Width>
    <Width>0.2</Width>
    <Width>0.2</Width>
    <Width>0.2</Width>
    <Width>0.2</Width>
    <Width>0.4</Width>
  </Petal>
</Iris>
```

YAML

```yaml
Iris:
  Sepal.Length:
    - 5.1
    - 4.9
    - 4.7
    - 4.6
    - 5.0
    - 5.4
  Petal.Width:
    - 0.2
    - 0.2
    - 0.2
    - 0.2
    - 0.2
    - 0.4
```

Notes

This document is written in *R Markdown*, which is a plain text markup language that combines traditional Markdown with LaTeX, code engines, and the R data science language.

Source code for this document can be found at https://github.com/WaqqasI/polyglottal-programming/blob/master/polyglottal.Rmd

[3] Dataset reduced for space