

# Programacion estructurada.

Daniel Reyes Barrera

19 de noviembre de 2020

## Resumen

En este documento se ha resuelto algunos problemas computacionales utilizando las herramientas aprendidas de la clase 4 – Programación estructurada del curso de programación C++, las cuales son: Estructura de control condicionada (`if-else` o `switch-case`), estructuras de control por repetición (`for`, `while`, `do-while`).

## 1. Ejercicio 1.

Escriba un programa que lea un año y determine si se trata de un año bisiesto.

### 1.1. Problema computacional.

**Objetivo:** Dado un número entero como año determinar si es un año bisiesto o no.

**Entrada:** Un número entero mayor que 0 representando un año.

**Salida:** La respuesta de si el número dado es un año bisiesto o no.

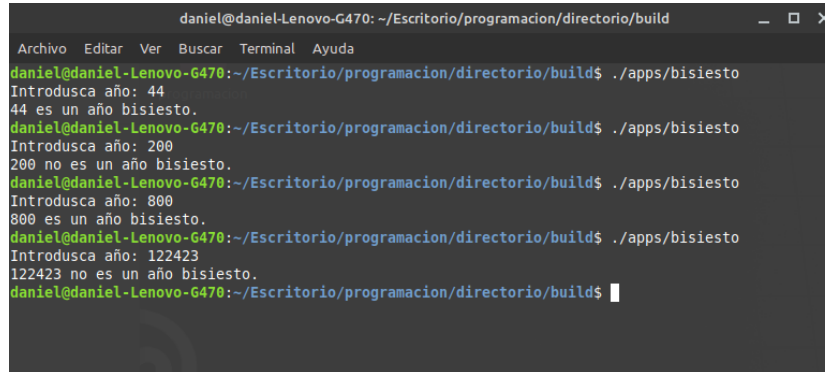
### 1.2. Algoritmo.

Para solucionar el problema computacional, partimos de la definición de un año bisiesto. Un año se denomina bisiesto si es múltiplo de 4, a excepción de que sí es múltiplo de 100 también y no de 400 entonces no es bisiesto, por tanto necesitaremos utilizar el condicional `if-else` para validar que se cumplan las condiciones para que el año sea bisiesto o no.

El código fuente está disponible en mi repositorio de git hub. [1]

### 1.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: 44, 200, 800 y 122423. La salida del programa se observa en la Figura 1.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/bisiesto
Introduzca año: 44
44 es un año bisiesto.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/bisiesto
Introduzca año: 200
200 no es un año bisiesto.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/bisiesto
Introduzca año: 800
800 es un año bisiesto.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/bisiesto
Introduzca año: 122423
122423 no es un año bisiesto.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 1: Ejecución de algunas instancias del problema.

## 2. Ejercicio 2.

Escriba un programa que capture los coeficientes  $a$ ,  $b$ ,  $c$  de un polinomio de segundo orden, e imprima las raíces del polinomio.

### 2.1. Problema computacional.

**Objetivo:** Calcular las raíces de un polinomio de segundo orden de manera general.

**Entrada:** Tres números reales que representan los coeficientes del polinomio.

**Salida:** Las raíces (reales o complejas) del polinomio.

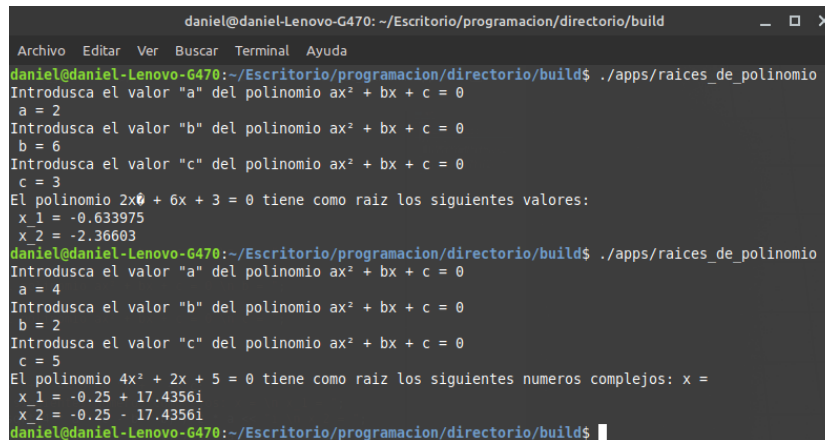
### 2.2. Algoritmo.

Para solucionar el problema computacional, sabemos que por la formula general  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$  podemos hallar las raíces de un polinomio de segundo orden. Si  $\sqrt{b^2 - 4ac} > 0$  entonces el polinomio tiene raíces reales, en caso contrario tiene raíces complejas.

El código fuente está disponible en mi repositorio de git hub. [6]

## 2.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron los siguientes polinomios:  $2x^2 + 6x + 3 = 0$  y  $4x^2 + 2x + 5 = 0$ . Donde el primer polinomio tiene raíces reales, y el segundo raíces complejas. La salida del programa se observa en la Figura 2.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/raices_de_polinomio
Introduzca el valor "a" del polinomio ax^2 + bx + c = 0
a = 2
Introduzca el valor "b" del polinomio ax^2 + bx + c = 0
b = 6
Introduzca el valor "c" del polinomio ax^2 + bx + c = 0
c = 3
El polinomio 2x^2 + 6x + 3 = 0 tiene como raiz los siguientes valores:
x_1 = -0.633975
x_2 = -2.36603
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/raices_de_polinomio
Introduzca el valor "a" del polinomio ax^2 + bx + c = 0
a = 4
Introduzca el valor "b" del polinomio ax^2 + bx + c = 0
b = 2
Introduzca el valor "c" del polinomio ax^2 + bx + c = 0
c = 5
El polinomio 4x^2 + 2x + 5 = 0 tiene como raiz los siguientes numeros complejos: x =
x_1 = -0.25 + 17.4356i
x_2 = -0.25 - 17.4356i
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 2: Ejecución de algunas instancias del problema.

## 3. Ejercicio 3.

Escriba un programa que lea dos números y uno de los siguientes operadores: +, -, \*, /. El programa debe aplicar y calcular la operación seleccionada a los números introducidos e imprimir el resultado.

### 3.1. Problema computacional.

**Objetivo:** Dado dos número y un operador mostrar el resultado de la operación.

**Entrada:** Dos números reales y un caracter (+, -, \* ó /).

**Salida:** El resultado de la operación.

### 3.2. Algoritmo.

Para solucionar el problema computacional, se utilizo la estructura de control condicionada `switch` para realizar la operación seleccionada.

El código fuente está disponible en mi repositorio de git hub. [7]

### 3.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: 34, 56, \* y 200, 100, -. La salida del programa se observa en la Figura 3.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/calcular_operacion
Primer valor: 34
Segundo valor: 56
Operacion a realizar: *
34*56 = 1904
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/calcular_operacion
Primer valor: 200
Segundo valor: 100
Operacion a realizar: -
200-100 = 100
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 3: Ejecución de algunas instancias del problema.

## 4. Ejercicio 4.

Escriba un programa que lea una letra y determine si es vocal o consonante. Asuma que el usuario no puede introducir números ni caracteres especiales.

### 4.1. Problema computacional.

**Objetivo:** Dado una letra, determinar si es vocal o consonante.

**Entrada:** Una letra del abecedario.

**Salida:** La respuesta de si la letra es una vocal o consonante.

### 4.2. Algoritmo.

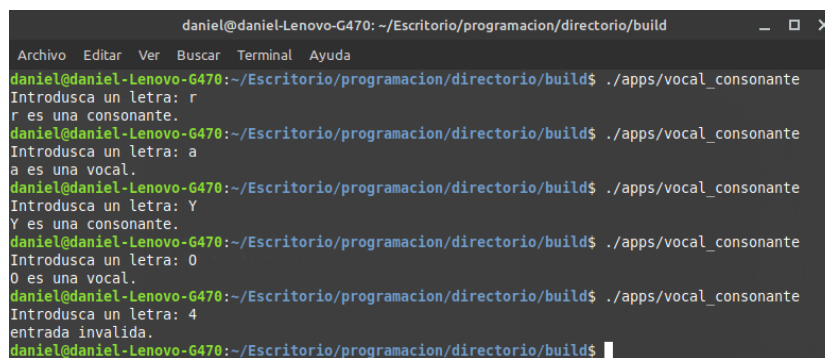
Para solucionar el problema computacional, se utilizó el método `toupper()` para tener en cuenta las letras minúsculas introducidas

por el usuario, posteriormente se utilizó la estructura de control condicionada `switch` para validar si la letra introducida por el usuario coincide con una vocal.

El código fuente está disponible en mi repositorio de git hub. [8]

### 4.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: r, a, Y, O y 4. La salida del programa se observa en la Figura 4.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/vocal_consonante
Introduzca un letra: r
r es una consonante.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/vocal_consonante
Introduzca un letra: a
a es una vocal.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/vocal_consonante
Introduzca un letra: Y
Y es una consonante.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/vocal_consonante
Introduzca un letra: 0
0 es una vocal.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/vocal_consonante
Introduzca un letra: 4
entrada invalida.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 4: Ejecución de algunas instancias del problema.

## 5. Ejercicio 5.

En el juego para dos personas llamado "Roca, papel ó tijera" cada jugador escoge "R", "P" o "T" respectivamente. El ganador se determina así: roca rompe tijeras, las tijeras cortan el papel, el papel cubre la roca, el juego es un empate si ambos jugadores eligen la misma opción. Elaborar un programa en que un jugador sea el usuario y el otro la computadora. El programa debe leer la entrada del usuario (R,P, ó T) y generar la opción elegida por la computadora de forma aleatoria. La salida debe mostrarse de la siguiente forma: "T-R Roca rompe tijeras: Gana el usuario" ó "P-R Papel cubre roca: Gana la computadora".

### 5.1. Problema computacional.

**Objetivo:** Programar el juego de "Roca, papel ó tijeras" donde el usuario juegue contra la computadora.

**Entrada:** Una letra (R, P ó T) representando la elección del usuario.

**Salida:** El resultado del juego comparando con la elección de la computadora.

### 5.2. Algoritmo.

Para solucionar el problema computacional, en primer lugar el código del juego esta siendo ejecutado en el ciclo `do-while` preguntandole al usuario si desea seguir jugando. Despues utilizamos la libreria `<ctime>` para tener una semilla diferente `srand(time(0))` en cada ejecución del programa y asi obtener valores diferentes en `rand()` para la elección de la computadora.

El código fuente está disponible en mi repositorio de git hub. [9]

### 5.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Primera elección del usuario: R. Segunda elección del usuario: t. La salida del programa se observa en la Figura 5.

```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/rock_piper
Roca, papel o tijeras
Elija:
R: Roca
P: Papel
T: Tijeras
Eleccion: R
T-R Roca rompe tijera Gana el usuario!

Desea seguir jugando?: (s/n)
Respuesta: s
Roca, papel o tijeras
Elija:
R: Roca
P: Papel
T: Tijeras
Eleccion: t
R-T Roca rompe tijera Gana la computadora

Desea seguir jugando?: (s/n)
Respuesta: n
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 5: Ejecución de algunas instancias del problema.

## 6. Ejercicio 6.

Escriba un programa que lea un número entero  $n$ , y calcule el resultado de la serie geométrica:

$$s = \sum_{k=0}^{n-1} \left(\frac{1}{2}\right)^k = 1 + \left(\frac{1}{2}\right)^1 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^3 + \left(\frac{1}{2}\right)^4 + \left(\frac{1}{2}\right)^5 + \dots + \left(\frac{1}{2}\right)^{n-1}$$

### 6.1. Problema computacional.

**Objetivo:** Dado un número entero positivo  $n$  calcular el resultado de la serie geométrica hasta el  $n$ -simo término.

**Entrada:** Un número entero positivo  $n$ .

**Salida:** El resultado de la suma hasta el  $n$ -simo término.

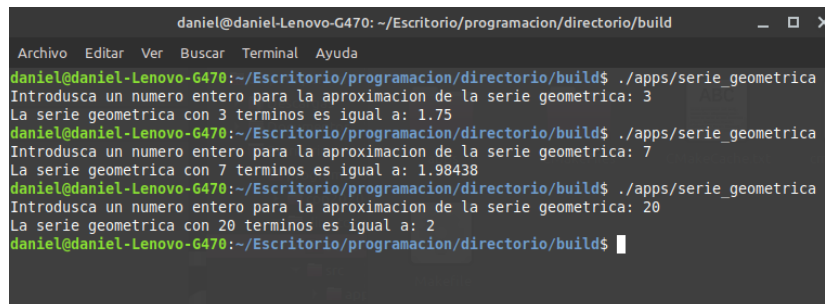
### 6.2. Algoritmo.

Para solucionar el problema computacional, se utilizo la estructura de control por repetición **for** para ir sumando  $n$  veces el resultado de cada termino a una variable de la suma total.

El código fuente está disponible en mi repositorio de git hub. [10]

### 6.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entradas:  $n = 3$ ,  $n = 7$  y  $n = 20$ . La salida del programa se observa en la Figura 6.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/serie_geometrica
Introduzca un numero entero para la aproximacion de la serie geometrica: 3
La serie geometrica con 3 terminos es igual a: 1.75
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/serie_geometrica
Introduzca un numero entero para la aproximacion de la serie geometrica: 7
La serie geometrica con 7 terminos es igual a: 1.98438
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/serie_geometrica
Introduzca un numero entero para la aproximacion de la serie geometrica: 20
La serie geometrica con 20 terminos es igual a: 2
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 6: Ejecución de algunas instancias del problema.

## 7. Ejercicio 7.

La sucesión de Fibonacci es un conjunto infinito de números ordenados, donde cada elemento de la sucesión es igual a la suma de los dos elementos anteriores. La sucesión comienza con los elementos 0, 1 y continúa como se muestra a continuación:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Escriba un programa que imprima los primeros 100 términos de la serie de Fibonacci.

### 7.1. Problema computacional.

**Objetivo:** Calcular los primeros 100 elementos de la serie de fibonacci.

**Entrada:** Un número entero representando la cantidad de términos deseados.

**Salida:** Los primeros 100 terminos de la serie de fibonnaci.



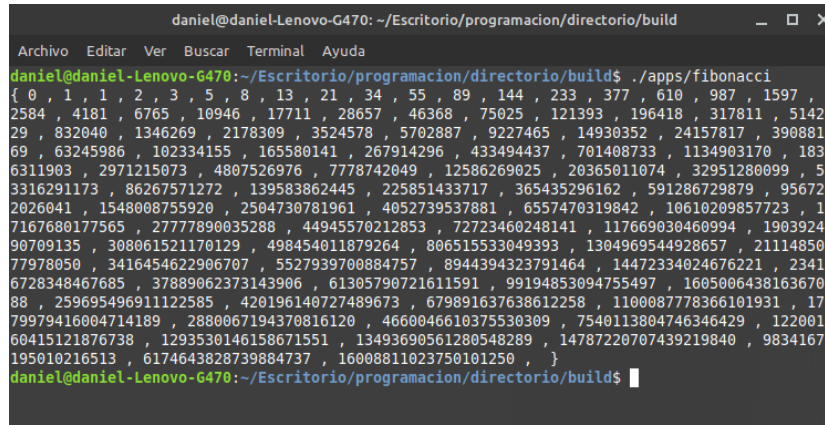
## 7.2. Algoritmo.

Para solucionar el problema computacional, se utilizó la estructura de control por repetición `for` para ir sumando los dos terminos anteriores para hallar el siguiente  $n = 100$  veces .

El código fuente está disponible en mi repositorio de git hub. [11]

## 7.3. Instancia del problema.

Como prueba de escritorio, se seleccionó la siguiente instancia del problema.  $n = 100$  La salida del programa se observa en la Figura 7.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/fibonacci
{ 0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , 144 , 233 , 377 , 610 , 987 , 1597 ,
2584 , 4181 , 6765 , 10946 , 17711 , 28657 , 46368 , 75025 , 121393 , 196418 , 317811 , 5142
29 , 832040 , 1346269 , 2178309 , 3524578 , 5702887 , 9227465 , 14930352 , 24157817 , 390881
69 , 63245986 , 102334155 , 165580141 , 267914296 , 433494437 , 701408733 , 1134903170 , 183
6311903 , 2971215073 , 4807526976 , 7778742049 , 12586269025 , 20365011074 , 32951280099 , 5
3316291173 , 86267571272 , 139583862445 , 225851433717 , 365435296162 , 591286729879 , 95672
2026041 , 1548008755920 , 2504730781961 , 4052739537881 , 6557470319842 , 10610209857723 , 1
7167680177565 , 27777890035288 , 44945570212853 , 72723460248141 , 117669030460994 , 1903924
90709135 , 308061521170129 , 498454011879264 , 806515533049393 , 1304969544928657 , 21114850
77978050 , 3416454622906707 , 5527939700884757 , 8944394323791464 , 14472334024676221 , 2341
6728348467685 , 37889062373143906 , 61305790721611591 , 99194853094755497 , 1605006438163670
88 , 259695496911122585 , 420196140727489673 , 679891637638612258 , 1100087778366101931 , 17
79979416004714189 , 2880067194370816120 , 4660046610375530309 , 7540113804746346429 , 122001
60415121876738 , 1293530146158671551 , 13493690561280548289 , 14787220707439219840 , 9834167
195010216513 , 6174643828739884737 , 10008811023750101250 , }
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 7: Ejecución de algunas instancias del problema.

## 8. Ejercicio 8.

En 1682, el matemático alemán, Gottfried Leibniz, propuso una fórmula para aproximar el valor del número  $\pi$ , de la siguiente forma:

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Esta fórmula, es una serie infinita. Sumando un conjunto infinito de términos, es posible aproximar el valor de  $\pi$  con una precisión razonable. Escriba un programa que calcule el valor aproximado de  $\pi$ . El programa debe preguntar al usuario el número de términos con los cuales desea calcular el valor de  $\pi$ .

## 8.1. Problema computacional.

**Objetivo:** Dado un número entero positivo calcular una aproximación de  $\pi$ .

**Entrada:** Un número entero positivo  $n$ .

**Salida:** Una aproximación de  $\pi$  a  $n$  términos.

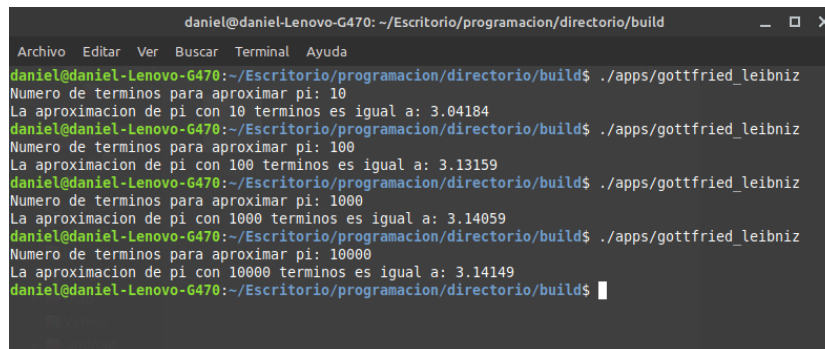
## 8.2. Algoritmo.

Para solucionar el problema computacional, se despejo el valor de  $\pi$  de la ecuación de Leibniz y utilizo la estructura de control por repetición `for` para ir sumando  $n$  veces los terminos de la ecuación.

El código fuente está disponible en mi repositorio de git hub. [12]

## 8.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada:  $n = 10$ ,  $n = 100$ ,  $n = 1000$  y  $n = 10000$ . La salida del programa se observa en la Figura 8.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/gottfried_leibniz
Numero de terminos para aproximar pi: 10
La aproximacion de pi con 10 terminos es igual a: 3.04184
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/gottfried_leibniz
Numero de terminos para aproximar pi: 100
La aproximacion de pi con 100 terminos es igual a: 3.13159
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/gottfried_leibniz
Numero de terminos para aproximar pi: 1000
La aproximacion de pi con 1000 terminos es igual a: 3.14059
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/gottfried_leibniz
Numero de terminos para aproximar pi: 10000
La aproximacion de pi con 10000 terminos es igual a: 3.14149
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 8: Ejecución de algunas instancias del problema.

## 9. Ejercicio 9.

El problema de Basilea, consiste en calcular la suma exacta de los inversos de los cuadrados de los números enteros positivos. En 1735, el matemático suizo Leonhard Euler solucionó dicho problema, de tal

forma que:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Escriba un programa que utilizando esta fórmula calcule el valor aproximado de  $\pi$ . El programa debe preguntar al usuario el número de términos con los cuales desea calcular el valor de  $\pi$ .

### 9.1. Problema computacional.

**Objetivo:** Dado un número entero positivo calcular una aproximación de  $\pi$ .

**Entrada:** Un número entero positivo  $n$ .

**Salida:** Una aproximación de  $\pi$  con  $n$  términos.

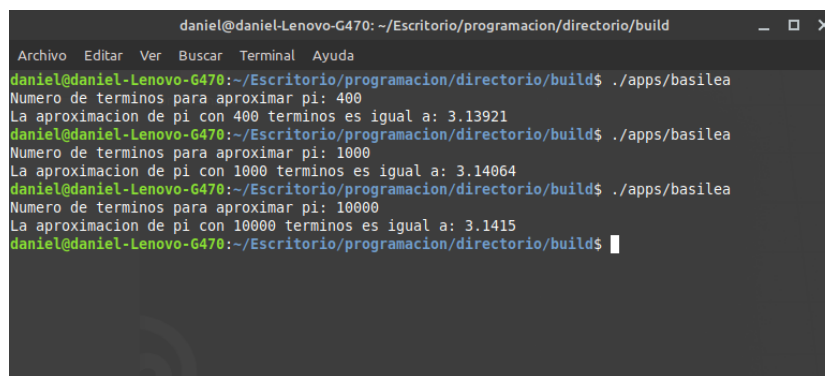
### 9.2. Algoritmo.

Para solucionar el problema computacional, se hizo un procedimiento similar al ejercicio 8

El código fuente está disponible en mi repositorio de git hub. [13]

### 9.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada:  $n = 400$ ,  $n = 1000$  y  $n = 10000$ . La salida del programa se observa en la Figura 9.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/basilea
Numero de terminos para aproximar pi: 400
La aproximacion de pi con 400 terminos es igual a: 3.13921
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/basilea
Numero de terminos para aproximar pi: 1000
La aproximacion de pi con 1000 terminos es igual a: 3.14064
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/basilea
Numero de terminos para aproximar pi: 10000
La aproximacion de pi con 10000 terminos es igual a: 3.1415
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 9: Ejecución de algunas instancias del problema.

## 10. Ejercicio 10.

La siguiente se llama conjetura de ULAM:

- Comience con cualquier número entero positivo.
- Si es par divídalo entre 2, si es impar multiplíquelo por 3 y agréguele 1.
- Obtenga enteros sucesivamente repitiendo el proceso.
- Al final, obtendrá el número 1, independientemente del entero inicial.

Escriba un programa que lea un número entero positivo y obtenga e imprima la sucesión de ULAM.

### 10.1. Problema computacional.

**Objetivo:** Dado un número entero positivo imprimir en pantalla la sucesión de ULAM empezando por dicho número.

**Entrada:** Un número entero positivo.

**Salida:** La sucesión de ULAM.

### 10.2. Algoritmo.

Para solucionar el problema computacional, se utilizó la estructura de control por repetición `do-while` el cual repite el ciclo siempre y cuando  $n! = 1$

El código fuente está disponible en mi repositorio de git hub. [2]

### 10.3. Instancia del problema.

Como prueba de escritorio, se seleccionó la siguiente instancia del problema. Entrada:  $n = 78$ . La salida del programa se observa en la Figura 10.

```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/ULAM
Ingrese un numero entero positivo: 78
La sucesion de ULAM es:
78
39
118
59
178
89
268
134
67
202
101
304
152
76
38
19
58
29
88
44
22
11
34
17
52
26
13
40
20
10
5
16
8
4
2
1
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 10: Ejecución de algunas instancias del problema.

## 11. Ejercicio 11.

Un triángulo recto puede tener lados cuyas longitudes sean valores enteros. El conjunto de tres valores enteros para las longitudes de los lados de un triángulo recto se conoce como triple de Pitágoras. Las longitudes de los tres lados deben satisfacer la relación que establece que la suma de los cuadrados de dos lados es igual al cuadrado de la hipotenusa. Escriba una aplicación para encontrar todos los triples de Pitágoras, donde el lado 1, lado 2 y la hipotenusa no sean mayores de 500.

### 11.1. Problema computacional.

**Objetivo:** Imprimir en pantalla ternas que sean triple de pítagoras tal que ningun valor no sea mayor de 500.

**Entrada:** Un número entero mayor que 0 que delimite la hipotenusa.

**Salida:** Ternas que son triples de pitágoras.

## 11.2. Algoritmo.

Para solucionar el problema computacional, utilizamos un doble ciclo `for`, el primer ciclo delimitado por 354 ya que esta cerca del valor maximo que deben de tener los catetos del triangulo rectangulo:  $354 \approx \sqrt{\frac{500^2}{2}}$ . El segundo ciclo delimitado por 500 en el cual se va evaluando el condicional `if` si la suma del cuadrado de los catetos es un número entero y si es menor que 500. Y de esta manera el programa es de tipo  $O(n^2)$  y no  $O(n^3)$ .

El código fuente está disponible en mi repositorio de git hub. [3]

## 11.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada: 40, 100 y 800. La salida del programa se observa en la Figura 11.

```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda

daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/triple de pitagoras (3, 4, 5), (5, 12, 13), (6, 8, 10), (7, 24, 25), (8, 15, 17), (9, 12, 15), (9, 40, 41), (10, 24, 26), (11, 60, 61), (12, 16, 20), (12, 35, 37), (13, 84, 85), (14, 48, 50), (15, 20, 25), (15, 36, 39), (15, 112, 113), (16, 30, 34), (16, 63, 65), (17, 144, 145), (18, 24, 30), (18, 80, 82), (19, 180, 181), (20, 21, 29), (20, 48, 52), (20, 99, 101), (21, 28, 35), (21, 72, 75), (21, 220, 221), (22, 120, 122), (23, 264, 265), (24, 32, 40), (24, 45, 51), (24, 70, 74), (24, 143, 145), (25, 60, 65), (25, 312, 313), (26, 168, 170), (27, 36, 45), (27, 126, 123), (27, 364, 365), (28, 45, 53), (28, 96, 100), (28, 195, 197), (29, 420, 421), (30, 40, 50), (30, 72, 78), (30, 224, 226), (31, 480, 481), (32, 40, 68), (32, 126, 130), (32, 355, 357), (33, 44, 55), (33, 56, 65), (33, 180, 183), (34, 288, 290), (35, 84, 91), (35, 120, 125), (36, 48, 60), (36, 77, 85), (36, 105, 111), (36, 168, 164), (36, 323, 325), (38, 360, 362), (39, 52, 65), (39, 80, 89), (39, 252, 255), (40, 42, 58), (40, 75, 85), (40, 96, 104), (40, 198, 202), (40, 399, 401), (42, 56, 7), (42, 144, 150), (42, 440, 442), (44, 112, 125), (44, 240, 244), (44, 483, 485), (45, 60, 75), (45, 108, 117), (45, 200, 205), (45, 336, 339), (48, 55, 73), (48, 64, 80), (48, 90, 102), (48, 140, 148), (48, 180, 195), (48, 288, 296), (49, 168, 175), (50, 120, 130), (51, 68, 85), (51, 140, 149), (51, 432, 435), (52, 165, 173), (52, 336, 340), (54, 72, 90), (54, 240, 246), (55, 132, 143), (55, 360, 365), (56, 96, 106), (56, 165, 119), (56, 192, 208), (56, 390, 394), (57, 76, 95), (57, 1, 76, 183), (60, 45, 87), (60, 80, 100), (60, 91, 109), (60, 144, 156), (60, 175, 185), (60, 221, 229), (60, 297, 303), (60, 448, 452), (63, 84, 105), (63, 216, 225), (63, 288, 297), (64, 120, 136), (64, 252, 260), (65, 72, 97), (65, 156, 169), (65, 420, 425), (66, 88, 110), (66, 112, 130), (66, 360, 366), (68, 285, 293), (69, 9, 12, 115), (69, 260, 269), (70, 168, 182), (70, 240, 258), (72, 96, 120), (72, 135, 153), (72, 154, 170), (72, 216, 222), (72, 326, 328), (72, 429, 435), (75, 108, 12), (75, 180, 195), (75, 300, 317), (76, 357, 365), (77, 264, 275), (77, 420, 427), (78, 104, 130), (78, 168, 178), (80, 84, 110), (80, 150, 170), (80, 192, 208), (80, 315, 325), (80, 396, 404), (81, 108, 135), (81, 360, 369), (84, 112, 140), (84, 135, 159), (84, 187, 205), (84, 245, 259), (84, 288, 300), (84, 437, 445), (85, 132, 157), (85, 204, 221), (87, 116, 145), (87, 416, 425), (88, 105, 137), (88, 165, 187), (88, 234, 250), (88, 480, 488), (90, 120, 150), (90, 216, 234), (90, 400, 410), (91, 312, 325), (93, 124, 155), (93, 476, 485), (95, 168, 195), (95, 228, 247), (96, 120, 140), (96, 136, 160), (96, 180, 204), (96, 247, 265), (96, 288, 296), (96, 378, 390), (98, 336, 350), (99, 132, 165), (99, 168, 195), (99, 440, 451), (100, 105, 145), (100, 240, 260), (102, 136, 170), (102, 288, 298), (104, 153, 18), (104, 195, 221), (104, 336, 340), (105, 140, 175), (105, 288, 233), (105, 252, 273), (105, 360, 375), (108, 144, 180), (108, 231, 255), (108, 315, 333), (108, 4, 80, 492), (110, 264, 266), (111, 140, 183), (112, 180, 212), (112, 210, 238), (112, 384, 400), (112, 441, 453), (114, 152, 190), (114, 352, 370), (115, 252, 277), (115, 276, 299), (117, 156, 195), (117, 240, 267), (119, 120, 169), (119, 488, 425), (120, 126, 174), (120, 160, 200), (120, 182, 218), (120, 209, 241), (120, 225, 255), (120, 288, 312), (120, 350, 370), (120, 391, 409), (120, 442, 458), (123, 164, 205), (125, 300, 325), (126, 168, 210), (126, 432, 450), (128, 240, 272), (129, 172, 215), (130, 144, 184), (130, 312, 330), (132, 176, 220), (132, 224, 260), (132, 351, 375), (132, 385, 407), (132, 476, 493), (133, 156, 203), (133, 456, 475), (135, 180, 225), (135, 324, 351), (135, 352, 377), (136, 255, 289), (136, 275, 305), (138, 184, 230), (140, 147, 203), (140, 171, 221), (140, 225, 265), (140, 336, 364), (141, 188, 235), (144, 165, 219), (144, 192, 240), (144, 276, 306), (144, 380, 340), (144, 420, 444), (145, 348, 377), (145, 480, 433), (147, 196, 245), (150, 200, 250), (150, 360, 390), (152, 205, 253), (152, 345, 377), (153, 204, 253), (155, 420, 447), (155, 372, 403), (155, 468, 493), (156, 288, 260), (156, 320, 355), (156, 455, 461), (159, 212, 265), (160, 168, 232), (160, 231, 261), (160, 300, 340), (160, 384, 416), (161, 240, 289), (162, 216, 270), (165, 220, 275), (165, 280, 325), (165, 396, 429), (168, 224, 280), (168, 270, 318), (168, 315, 357), (168, 374, 410), (168, 425, 457), (170, 264, 314), (170, 408, 442), (171, 228, 285), (174, 222, 290), (175, 300, 337), (175, 420, 453), (176, 210, 274), (176, 330, 374), (177, 236, 295), (180, 180, 201), (180, 240, 300), (180, 272, 327), (180, 299, 340), (180, 385, 425), (180, 432, 468), (183, 244, 305), (184, 345, 391), (185, 444, 481), (186, 248, 310), (189, 252, 315), (189, 340, 389), (190, 336, 386), (190, 456, 494), (192, 220, 292), (192, 256, 320), (192, 360, 408), (195, 216, 291), (195, 260, 325), (195, 400, 445), (196, 315, 371), (198, 264, 330), (198, 336, 390), (200, 210, 290), (200, 370, 425), (201, 288, 335), (203, 396, 445), (204, 252, 325), (204, 272, 340), (207, 224, 305), (207, 276, 345), (208, 306, 370), (208, 396, 442), (210, 280, 350), (210, 416, 466), (213, 284, 355), (216, 288, 360), (216, 405, 459), (219, 292, 365), (220, 231, 319), (222, 296, 370), (224, 360, 424), (224, 42, 0, 476), (225, 272, 353), (225, 300, 375), (228, 384, 380), (228, 325, 397), (231, 308, 385), (231, 392, 455), (232, 435, 493), (234, 312, 390), (237, 316, 395), (238, 240, 331), (240, 252, 340), (240, 275, 365), (240, 320, 400), (240, 364, 430), (240, 416, 482), (243, 324, 405), (246, 328, 410), (249, 332, 415), (252, 276, 37), (252, 336, 420), (252, 405, 477), (255, 340, 425), (255, 396, 471), (258, 344, 430), (260, 273, 377), (260, 288, 380), (261, 348, 435), (261, 380, 461), (264, 3, 15, 411), (264, 352, 448), (266, 312, 410), (267, 356, 445), (270, 360, 450), (273, 364, 455), (276, 368, 460), (279, 372, 465), (280, 294, 406), (280, 342, 442), (280, 351, 440), (282, 376, 470), (285, 380, 475), (288, 330, 430), (289, 364, 480), (291, 388, 465), (294, 392, 490), (297, 304, 423), (297, 396, 495), (300, 315, 435), (319, 360, 481), (320, 336, 464), (325, 360, 485), (340, 357, 493),
```

Figura 11: Ejecución de algunas instancias del problema.

## 12. Ejercicio 12.

Escriba un programa capture un número e imprima un mensaje indicando si es un número primo.

### 12.1. Problema computacional.

**Objetivo:** Dado un número entero mayor que cero, imprimir si es primo o no.

**Entrada:** Un número entero mayor que 0.

**Salida:** La respuesta de si el número dado es primo o no.

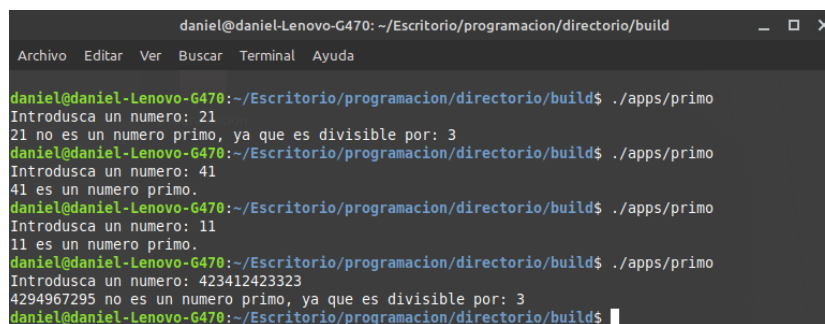
### 12.2. Algoritmo.

Para solucionar el problema computacional, partimos de la definición de número primo el cual un número se denomina primo si solo es divisible por él mismo y por 1. En este caso primero verificamos si es divisible por 2, si no, entonces mediante un ciclo **for** recorriendo solo números impares delimitado por  $n/2$  verificamos si  $n$  es divisible.

El código fuente está disponible en mi repositorio de git hub. [4]

### 12.3. Instancia del problema.

Como prueba de escritorio, se seleccionaron las siguientes instancias del problema. Entrada:  $n = 21$ ,  $n = 41$ ,  $n = 11$  y  $n = 423412423323$ . La salida del programa se observa en la Figura 12.



```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo Editar Ver Buscar Terminal Ayuda

daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/primo
Introduzca un numero: 21
21 no es un numero primo, ya que es divisible por: 3
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/primo
Introduzca un numero: 41
41 es un numero primo.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/primo
Introduzca un numero: 11
11 es un numero primo.
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/primo
Introduzca un numero: 423412423323
4294967295 no es un numero primo, ya que es divisible por: 3
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 12: Ejecución de algunas instancias del problema.

## 13. Ejercicio 13.

Escriba un programa que capture 15 números y los imprima ordenados de menor a mayor.

### 13.1. Problema computacional.

**Objetivo:** Ordenar una cantidad de  $n$  números dados por el usuario.

**Entrada:** Una serie de números.

**Salida:** Los números dados por el usuario impresos de menor a mayor.

### 13.2. Algoritmo.

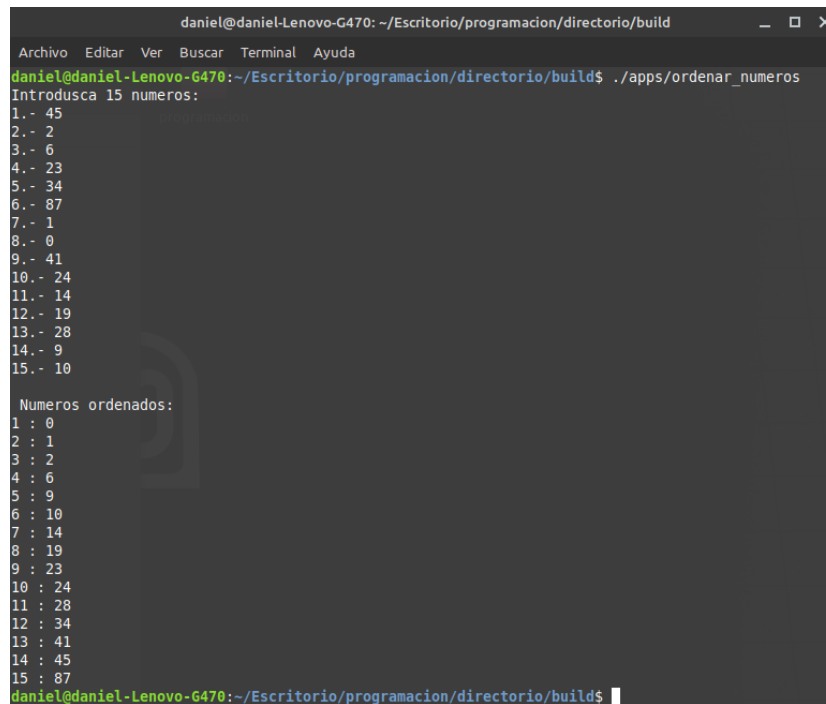
Para solucionar el problema computacional, se utilizó un array para ser más práctico el programa y se aplicó un algoritmo de ordenamiento conocido como Selection Sort.

El código fuente está disponible en mi repositorio de git hub. [5]

### 13.3. Instancia del problema.

Como prueba de escritorio, se seleccionó la siguiente instancia del problema. Entrada: 45, 2, 6, 23, 34, 87, 1, 0, 41, 24, 14, 19, 28, 9 y 10. La salida del programa se observa en la Figura 13.





```
daniel@daniel-Lenovo-G470: ~/Escritorio/programacion/directorio/build
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$ ./apps/ordenar_numeros
Introduzca 15 numeros:
1.- 45
2.- 2
3.- 6
4.- 23
5.- 34
6.- 87
7.- 1
8.- 0
9.- 41
10.- 24
11.- 14
12.- 19
13.- 28
14.- 9
15.- 10

Numeros ordenados:
1 : 0
2 : 1
3 : 2
4 : 6
5 : 9
6 : 10
7 : 14
8 : 19
9 : 23
10 : 24
11 : 28
12 : 34
13 : 41
14 : 45
15 : 87
daniel@daniel-Lenovo-G470:~/Escritorio/programacion/directorio/build$
```

Figura 13: Ejecución de algunas instancias del problema.

## 14. Conclusiones.

Las Estructura de control condicionada ( `if-else` o `switch-case`), estructuras de control por repetición (`for`, `while`, `do-while`) pueden resolver muchos problemas en el cual se requieran ciclos o codiciones, pero si incluimos los array los problemas abarcables son innumerables.

## Referencias

- [1] Daniel Reyes Barrera. *Problema 1*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/a%C3%B1o\\_bisiesto.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/a%C3%B1o_bisiesto.cpp) (visitado 16-11-2020).
- [2] Daniel Reyes Barrera. *Problema 10*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/ULAM.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/ULAM.cpp) (visitado 16-11-2020).

- [3] Daniel Reyes Barrera. *Problema 11*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/triple\\_de\\_pitagoras.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/triple_de_pitagoras.cpp) (visitado 16-11-2020).
- [4] Daniel Reyes Barrera. *Problema 12*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/primo.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/primo.cpp) (visitado 16-11-2020).
- [5] Daniel Reyes Barrera. *Problema 13*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/ordenar\\_numeros.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/ordenar_numeros.cpp) (visitado 16-11-2020).
- [6] Daniel Reyes Barrera. *Problema 2*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/raices\\_de\\_polinomio.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/raices_de_polinomio.cpp) (visitado 16-11-2020).
- [7] Daniel Reyes Barrera. *Problema 3*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/calcular\\_operacion.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/calcular_operacion.cpp) (visitado 16-11-2020).
- [8] Daniel Reyes Barrera. *Problema 4*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/vocal\\_consonante.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/vocal_consonante.cpp) (visitado 16-11-2020).
- [9] Daniel Reyes Barrera. *Problema 5*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/rock\\_piper.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/rock_piper.cpp) (visitado 16-11-2020).
- [10] Daniel Reyes Barrera. *Problema 6*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/serie\\_geometrica.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/serie_geometrica.cpp) (visitado 16-11-2020).
- [11] Daniel Reyes Barrera. *Problema 7*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/fibonacci.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/fibonacci.cpp) (visitado 16-11-2020).
- [12] Daniel Reyes Barrera. *Problema 8*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/gottfried\\_leibniz.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/gottfried_leibniz.cpp) (visitado 16-11-2020).
- [13] Daniel Reyes Barrera. *Problema 9*. 2020. URL: [https://github.com/danield877/cpp2020/blob/master/Clase\\_4\\_Tareas/basilea.cpp](https://github.com/danield877/cpp2020/blob/master/Clase_4_Tareas/basilea.cpp) (visitado 16-11-2020).