

Proyecto 2

Planteamiento del Problema

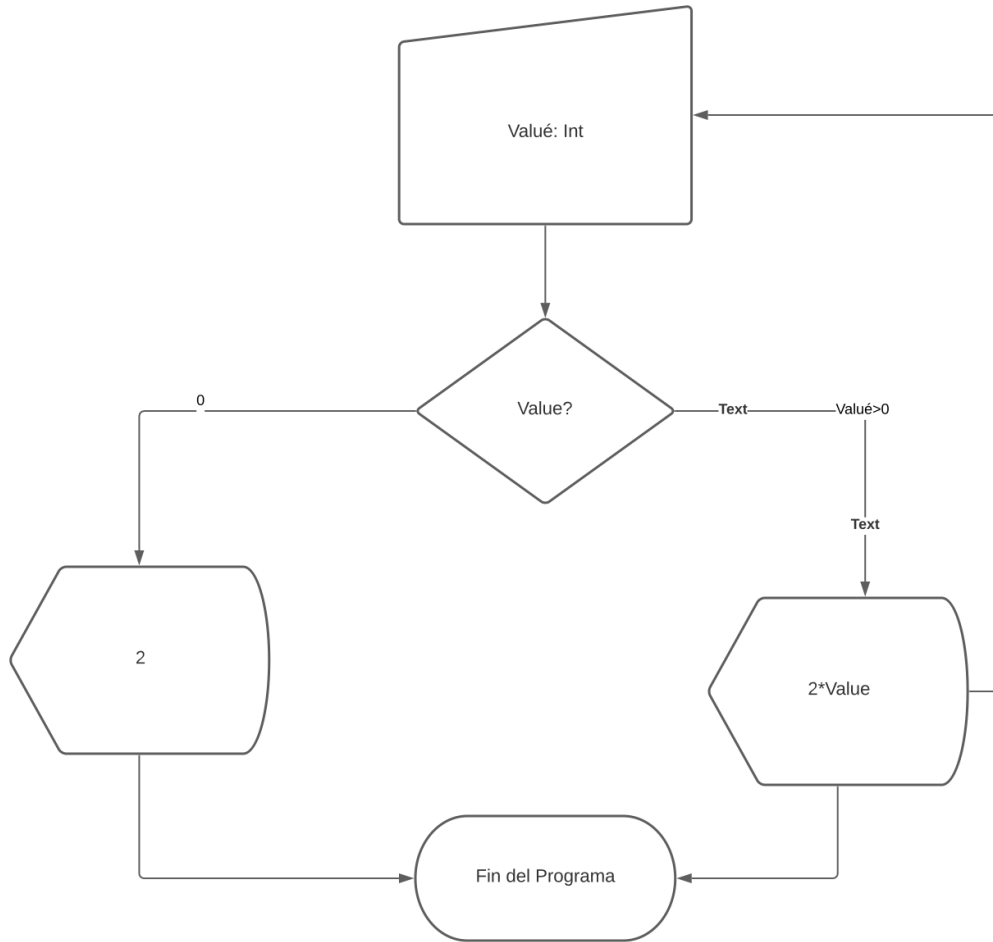
Para esta actividad se plantearon 5 problemas, cada uno de estos conllevaron un pensamiento diferente y una situación diferente a nivel de código, sin embargo, todas estas se presentan como funciones por cada problema, estas engloban una clase general llamada *Proyecto 2*

Ejercicio 1

Planteamiento del problema: Desarrolle el algoritmo que calcule 2^N donde N es un número entero mayor o igual a cero, el algoritmo debe ser desarrollado utilizando exclusivamente recursividad

Diagrama de Flujo

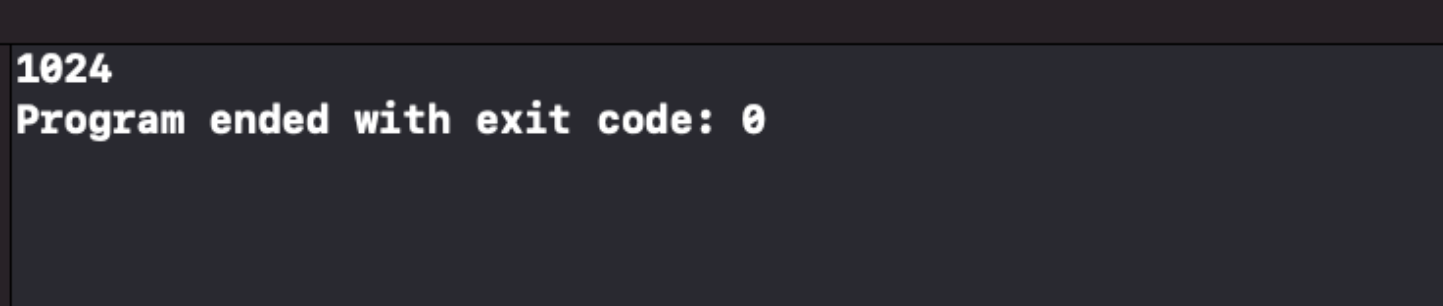
Planteamiento de la Solución: Para este problema se usó un simple if, de esta forma podemos usar la recursividad para el caso base, donde el valor de entrada es mayor a 0



Código

```
// Ejercicio 1
func ejercicio1(value: Int) -> Int{
    let base = 2
    var pot: Int
    if (value == 0) {
        pot = 1
    } else if (value > 0) {
        pot = base * ejercicio1(value: value - 1)
    } else {
        pot = 0
    }
    return pot
}
```

Capturas de Pantalla



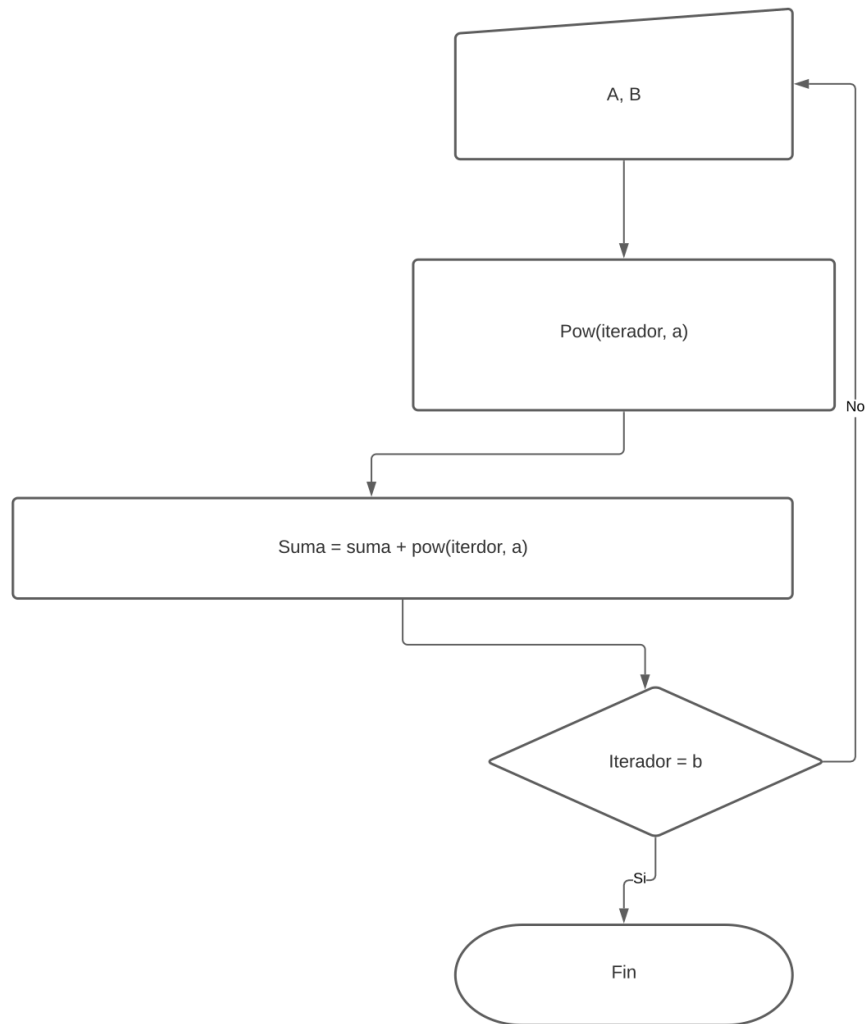
```
1024
Program ended with exit code: 0
```

Ejercicio 2

Planteamiento del problema: Dado a y b como números enteros, realizar la sumatoria de:
 $1^a + 2^a + 3^a \dots b^a$

Diagrama de Flujo

Planteamiento de la Solución: Para este problema se usó un for, en donde se recorriera de 1 hasta el valor de b , se eleva a la potencia indicada y se guarda en una variable donde se suma.



Código

```
// Ejercicio 2

func ejercicio2(a: Int, b:Int) -> Decimal{
    let a1 = Decimal(a)
    var suma: Decimal = 0
    for i in 1...b{
        let i2 = Decimal(i)
        print("b[\(i)]^a[\(a)]", pow(i2, a))
        suma = suma + pow(i2, a)
    }
    return suma
}
```

Capturas de Pantalla

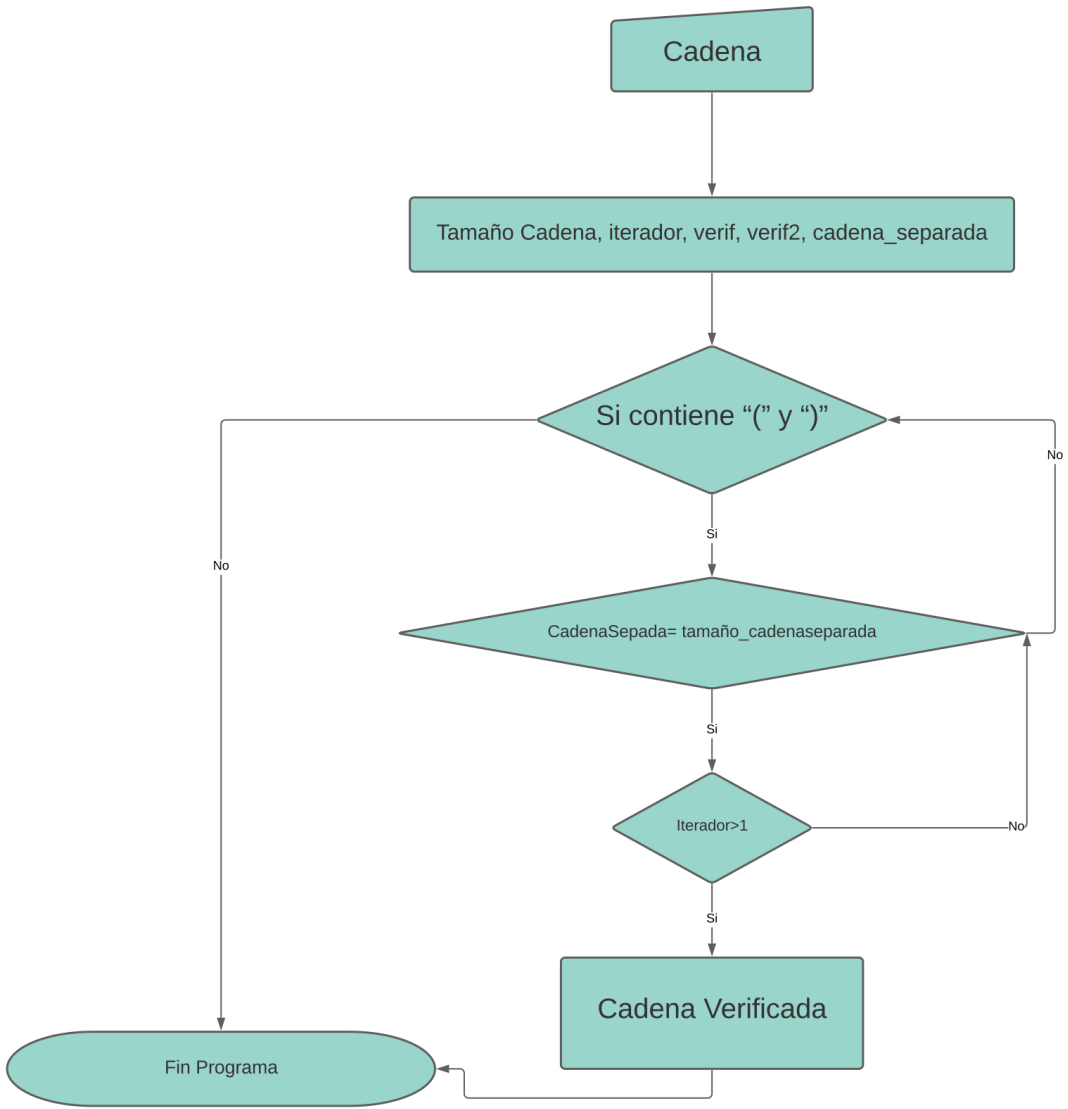
```
b[1]^a[10] 1
b[2]^a[10] 1024
b[3]^a[10] 59049
b[4]^a[10] 1048576
1108650
Program ended with exit code: 0
```

Ejercicio 3

Planteamiento del problema: Dada una cadena introducida por el usuario validar si es un número telefónico con la siguiente estructura (nn) - nn - nnnn - nnnn

Diagrama de Flujo

Planteamiento de la Solución: Para este problema se penso primero en cómo dividir las cadenas, estas deben de estar separadas principalmente por el delimitador -, a su vez para validar que la cadena sea válida, debe de contar con los parentesis que envuelven a los 2 primeros dígitos (n), posteriormente se aceptan los números de dígitos. En caso de no contar



Código

```
// Ejercicio 3
func ejercicio3(){
    print("Inserte la cadena a Verificar>> ")
    let b = String (readLine() ?? "?")
    let a = b+" "
    print("Cadena a Verificar: ", a, " Tamaño de la cadena: ",
a.count)
    let tam = a.count
    var cadenaSeparada = a.split(separator: "-")
    var iterador:Int = 0
    var verif:Int = 0
    var verif2:Bool
    var cademaSeparada2: String
    if a.hasPrefix("(") && a.contains(")") && a.contains(" - "){
        print("Si esta")
        for i in cadenaSeparada{
            iterador = iterador + 1
            if iterador > 1{
                verif = i.count
            }
        }
        if verif == 8{
            print("Cadena Validada! Cumple el formato (nn) - nn - nnnn
- nnnn")
        }
    }else{
        print("La cadena no contiene la estructura Valida")
    }
}
```


Capturas de Pantalla

```
Inserte la cadena a Verificar>>
55 - 66-66666
Cadena a Verificar: 55 - 66-66666      Tamaño de la cadena: 14
La cadena no contiene la estructura Valida
Program ended with exit code: 0
```

```
Inserte la cadena a Verificar>>
(55) - 55 - 5555
Cadena a Verificar: (55) - 55 - 5555    Tamaño de la cadena: 14
Si esta
Program ended with exit code: 0
```

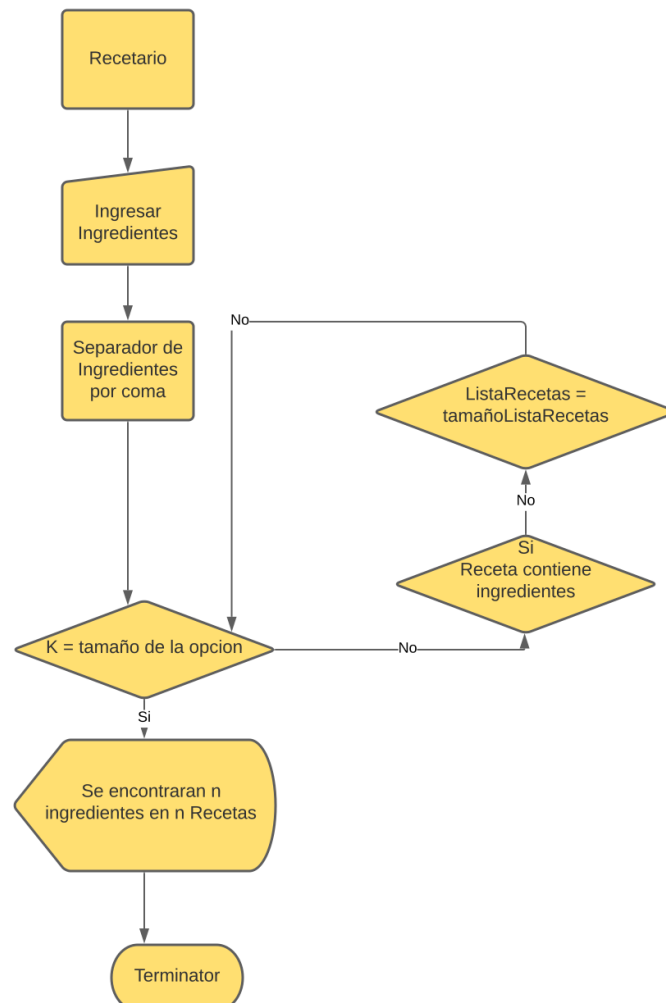
Ejercicio 4

Planteamiento del problema: Dada una lista de recetas (mínimo 10 recetas - texto) encontrar las recetas que contengan alguno o algunos ingredientes. El usuario solo verá las recetas que contengan los ingredientes que el captura desde la línea de comandos. Es importante que las recetas no solo indiquen instrucciones, sino también ingredientes. El usuario puede hacer la siguiente captura:

- Introduce ingredientes > huevos
- Introduce ingredientes > res, huevos, sal
- Introduce ingredientes > harina, sal, leche

Diagrama de Flujo

Planteamiento de la Solución: Para este Planteamiento se toma en cuenta tener una lista de recetas previamente cargadas, esta lista es una estructura de tipo *diccionario*, dónde está como llave el nombre de la receta y como valor se tiene la receta en sí. *llave(Nombre de la receta)->"Bisquets":Valor(Contenido de la receta)*



Código

```
// Ejercicio 4
func ejercicio4(){
    print("Existen ", self.listaRecetas.count, " en el recetario")
}
```

```

print("Escriba los ingredientes que desea Buscar>> ")
var opcion = String (readLine() ?? "?")
var opcion2 = opcion.lowercased().split(separator: ",")
var contador: Int = 0
for k in opcion2{
    for (i,j) in self.listaRecetas{
        if j.lowercased().contains(k){
            print("La Receta de: ", i, " Contiene el ingrediente:
", k)

            print(j)
            contador = contador + 1
        }
    }
}
print("Se encontraron \n(contador) coincidencias en \n
(self.listaRecetas.count) Listas")
}

```

Capturas de Pantalla

minutos. En una batidora agrega la harina con el azúcar, la sal, la leche, la levadura hidratada, los huevos, la esencia de vainilla y bate hasta tener una masa homogénea. Agrega la mantequilla y bate hasta que se incorpore por completo. Coloca la masa en un bowl engrasado y tápala, deja reposar por 2 horas o hasta que doble su volumen. Para la costra: coloca en la batidora la harina con el azúcar glass, agrega la manteca vegetal y mezcla, hasta obtener una masa homogénea. Separa la mitad de la masa, agrega a una la cocoa en polvo y a la otra la esencia de vainilla. Tapa y reserva. Sobre una superficie lisa, pesa bolitas de 80 gramos, coloca en una charola y aplana con la palma de la mano. Toma un poco de la masa de la costra de aproximadamente 15 gramos y con ayuda de un rodillo aplana hasta obtener un diámetro igual al de la concha. Cubre las conchas con la costra y marca con un cortador. Deja reposar hasta que dupliquen su volumen y hornea por 15 minutos.

La Receta de: 1.- Bisquets Contiene el ingrediente: leche

Precalienta el horno a 180°C, En un bowl mezclamos la harina, el polvo para hornear, media cucharadita de sal y una cucharada de azúcar. Añadimos la mantequilla cortada en cuadros (tiene que estar fría). Mezcla presionando con los dedos hasta que tenga consistencia de pan rallado. Se añade el huevo batido y la leche. Mezcla con una cuchara hasta conseguir una masa. Pon harina en tu superficie de trabajo. Coloca la masa salpica de harina y se trabaja para formar una bola. Estira la bola con el rodillo (1.5 cm de grosor) y corta los biscuits con moldes circulares. Coloca las ruedas en una bandeja con papel de hornear y se pinta la parte superior con la mezcla de huevo y leche. Mete al horno durante 15 minutos o hasta que se doren.

Se encontraron 10 coincidencias en 10 Listas

Existen 10 en el recetario

Mientras realizas la preparación de las mantecadas, te sugerimos que precalientes el horno a 180°C. En un recipiente debes cernir la harina junto con el polvo para hornear, la pizca de sal, mezcla bien, reserva. En otro recipiente coloca los huevos y bátelos a velocidad baja, durante 5 min. Agrega la taza de azúcar y sigue batiendo otro minuto más, una vez pasado el tiempo agrega el aceite y la esencia de vainilla, sigue batiendo hasta incorporar bien. Adiciona poco a poco la mezcla de harinas reservadas previamente, cada vez que incorpores esta mezcla ve agregando un poco de leche, sigue batiendo a velocidad baja hasta incorporar el resto. Bate un poco hasta integrar todos los ingredientes. Coloca los capacillos sobre el molde para hornear y agrega la mezcla a $\frac{1}{2}$ de su capacidad. Hornea a 180 °C durante 25 min. (para verificar si están listas puedes hacer la prueba del palillo que al introducirlo en una de las mantecadas este saldrá seco). Una vez listas, deja reposar las mantecadas antes de desmoldarlas. Para preparar el jarabe, coloca el azúcar en el agua caliente y disuelve bien. Desmolda las mantecadas y con ayuda de una brocha para cocina coloca el jarabe sobre ellas y barniza cada una. Déjalas reposar unos minutos y estarán listas

La Receta de: 9.- Conchas de pan Contiene el ingrediente: huevo

Precalienta el horno a 180°C. Hidrata la levadura con la leche y deja reposar 10 minutos. En una batidora agrega la harina con el azúcar, la sal, la leche, la levadura hidratada, los huevos, la esencia de vainilla y bate hasta tener una masa homogénea. Agrega la mantequilla y bate hasta que se incorpore por completo. Coloca la masa en un bowl engrasado y tápala, deja reposar por 2 horas o hasta que doble su volumen. Para la costra: coloca en la batidora la harina con el azúcar glass, agrega la manteca vegetal y mezcla, hasta obtener una masa homogénea. Separa la mitad de la masa, agrega a una la cocoa en polvo y a la otra la esencia de vainilla. Tapa y reserva. Sobre una superficie lisa, pesa

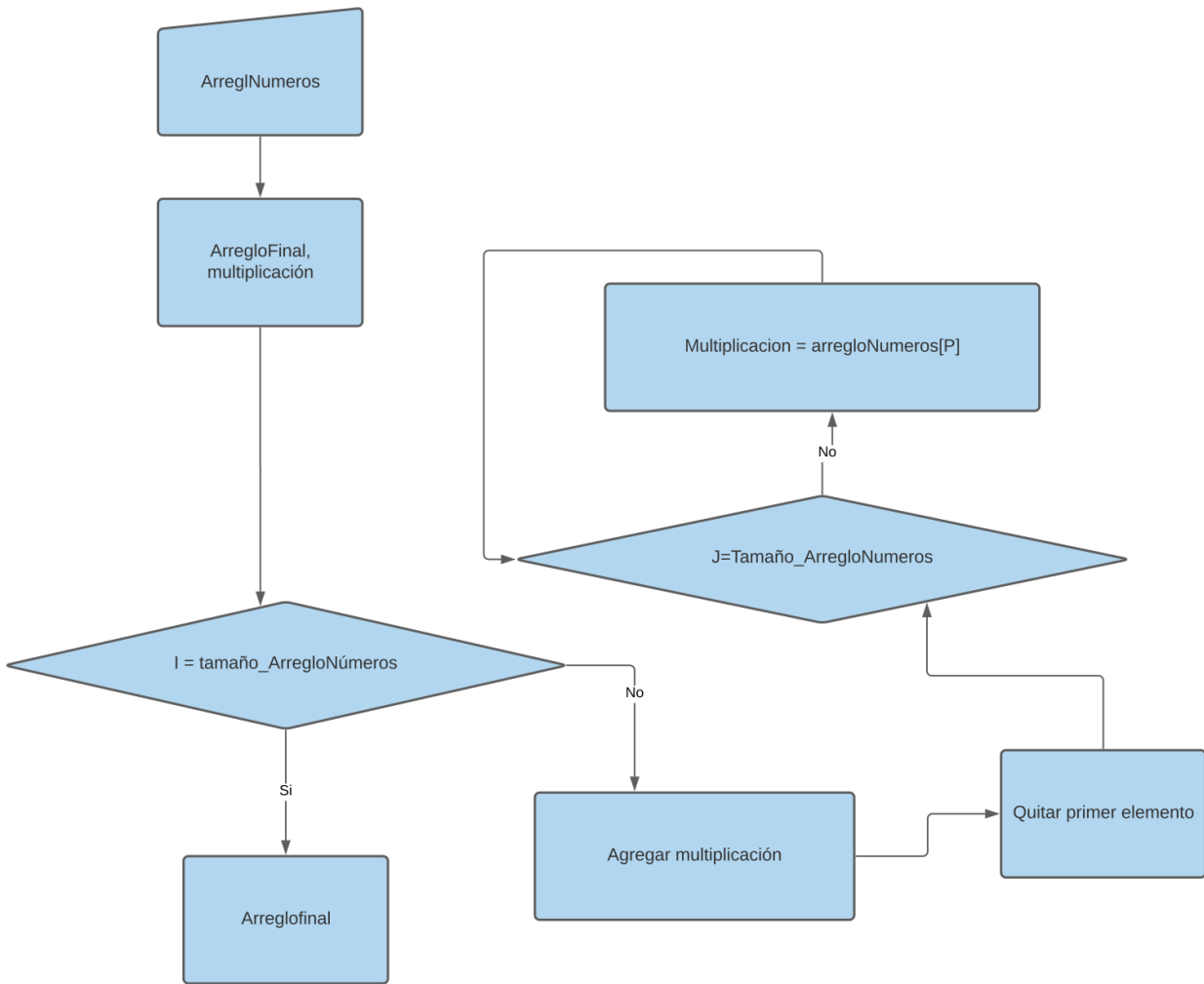
```
Existen 10 en el recetario
Escriba los ingredientes que desea Buscar>>
Teclados
Se encontraron 0 coincidencias en 10 Listas
Program ended with exit code: 0
```

Ejercicio 5

Planteamiento del problema: Dado un arreglo de números enteros, regresar un nuevo arreglo tal que cada elemento en la posición "i" del nuevo arreglo es el producto de todos los números del arreglo original menos el de la posición i, por ejemplo: [1, 2, 3, 4, 5] nos dará el nuevo arreglo con los valores [120, 60, 40, 30, 24]

Diagrama de Flujo

Planteamiento de la Solución: Para este problema se pensaron varias formas de cómo resolverlo, sin embargo se optó por usar 2 opciones particulares de los arreglos o bien, las listas, se usa la operación de tipo *remove* y la operación de tipo *append*, estas cumplen las funciones de quitar el primer elemento para agregarlo al final de el arreglo, haciendo que se forme la estructura deseada.



Código

```
// Ejercicio 5
func ejeercicio5(arregloNumeros: [Int]) -> [Int] {
    var arregloFinal: [Int] = []
    var multiplicacion: Int = 1
    var arregloNumeros2 = arregloNumeros
    var c: Int = 0
    for i in 1...arregloNumeros.count{
        for j in 2...arregloNumeros2.count{
            multiplicacion = multiplicacion * arregloNumeros2[j-1]
        }
    }
}
```

```

        arregloFinal.append(multiplicacion)
        c = arregloNumeros2.remove(at: 0)
        arregloNumeros2.append(c)
        multiplicacion = 1
    }
    return arregloFinal
}
}

```

Capturas de Pantalla

```

var c = ejeercicio5(arregloNumeros: [1,2,3,4,5])
print(c)
var d = ejeercicio5(arregloNumeros: [90,1,5,32,7])
print(d)

```

	[120, 60, 40, 30, 24]
	[1120, 100800, 20160, 3150, 14400]
	Program ended with exit code: 0

Código Final y ejecución

Finalmente se puso todo el código en una clase llamada **Proyecto 2**, esta contiene a manera de métodos, todas las funciones anteriormente explicadas.

```

/*
    @name: Cabrera Garibaldi Hernan Galileo
    @Date: 10/Diciembre/2020 - 6/Enero/2021
    @Descripcion: Project 2 of CM
*/
import Foundation

class Proyecto2{
    //    Variables para el ejercicio 4

```



```

func ejercicio2(a: Int, b: Int) -> Decimal{
    let a1 = Decimal(a)
    var suma: Decimal = 0
    for i in 1...b{
        let i2 = Decimal(i)
//        print("b[\(i)]^a[\(a)]", pow(i2, a))
        suma = suma + pow(i2, a)
    }
    return suma
}

// Ejercicio 3
func ejercicio3(){
    print("Inserte la cadena a Verificar>> ")
    let b = String (readLine() ?? "?")
    let a = b + " "
    print("Cadena a Verificar: ", a, " Tamaño de la cadena: ",
a.count)

    let tam = a.count
    var cadenaSeparada = a.split(separator: "-")
    var iterador: Int = 0
    var verif: Int = 0
    var verif2: Bool
    var cademaSeparada2: String
    if a.hasPrefix("(") && a.contains(")") && a.contains(" - "){
        print("Si esta")
        for i in cadenaSeparada{
            iterador = iterador + 1
            if iterador > 1{
                verif = i.count
            }
        }
        if verif == 8{

```

```

        print("Cadena Validada! Cumple el formato (nn) - nn - nnnn
- nnnn")
    }
}else{
    print("La cadena no contiene la estructura Valida")
}
}

```

// Ejercicio 4

```

func ejercicio4(){
    print("Existen ", self.listaRecetas.count, " en el recetario")
    print("Escriba los ingredientes que desea Buscar>> ")
    var opcion = String (readLine() ?? "?")
    var opcion2 = opcion.lowercased().split(separator: ",")
    var contador: Int = 0
    for k in opcion2{
        for (i,j) in self.listaRecetas{
            if j.lowercased().contains(k){
                print("La Receta de: ", i, " Contiene el ingrediente:
", k)

                print(j)
                contador = contador + 1
            }
        }
    }
    print("Se encontraron \n(contador) coincidencias en \n
(self.listaRecetas.count) Listas")
}

```

// Ejercicio 5

```

func ejeercicio5(arregloNumeros: [Int]) -> [Int] {
    var arregloFinal: [Int] = []
    var multiplicacion: Int = 1
    var arregloNumeros2 = arregloNumeros
    var c: Int = 0

```

```

        for i in 1...arregloNumeros.count{
            for j in 2...arregloNumeros2.count{
                multiplicacion = multiplicacion * arregloNumeros2[j-1]
            }
            arregloFinal.append(multiplicacion)
            c = arregloNumeros2.remove(at: 0)
            arregloNumeros2.append(c)
            multiplicacion = 1
        }
        return arregloFinal
    }
}

let proyecto2 = Proyecto2()
while true {
    print("Elija el numero del ejercicio que desea visualizar\n1-Ejercicio
1\n2.-Ejercicio 2\n3.-Ejercicio 3\n4.-Ejercicio 4\n5.- Ejercicio 5\n0.-
Salir")
    var iterations = Int (readLine() ?? "0")
    switch iterations {
        case 0:
            exit(0)
        case 1:
            print("Ejercicio 1\nDesarrolle el algoritmo que calcule 2 a la N
donde N es un número entero mayor o igual a cero, el algoritmo debe ser
desarrollado utilizando exclusivamente recursividad")
            var c = proyecto2.ejercicio1(value: 4)
            print(c)
        case 2:
            print("Ejercicio 2\n Dado a y b como números enteros, realizar la
sumatoria de: 1^a + 2^a + 3^a ... b^a")
            var op = proyecto2.ejercicio2(a: 3, b: 4)
            print(op)
    }
}

```

```

    case 3:
        print("Ejercicio 3: Dada una cadena introducida por el usuario
validar si es un número telefónico con la siguiente estructura (nn) - nn -
nnnn - nnnn\n")
        var op2 = proyecto2.ejercicio3()
    case 4:
        print("Ejercicio 4")
        var op3 = proyecto2.ejercicio4()
    case 5:
        print("Ejercicio 5")
        var op4 = proyecto2.ejeercicio5(arregloNumeros: [1,2,3,4,5])
    default:
        print("Opcion No valida")
}
}

```

Eliga el numero del ejercicio que desea visualizar

```

1-Ejercicio 1
2.-Ejercicio 2
3.-Ejercicio 3
4.-Ejercicio 4
5.- Ejercicio 5
0.-Salir
1
- . . .

```

Eliga el numero del ejercicio que desea visualizar

```

1-Ejercicio 1
2.-Ejercicio 2
3.-Ejercicio 3
4.-Ejercicio 4
5.- Ejercicio 5
0.-Salir
2
Ejercicio 2
Dado a y b como números enteros, realizar la sumatoria de: 1^a + 2^a + 3^a ... b^a
10
20
24163571680850
- . . .

```


Eliga el numero del ejercicio que desea visualizar

- 1-Ejercicio 1
- 2.-Ejercicio 2
- 3.-Ejercicio 3
- 4.-Ejercicio 4
- 5.- Ejercicio 5
- 0.-Salir

3

Ejercicio 3: Dada una cadena introducida por el usuario validar si es un número telefónico con la siguiente estructura (nn) - nn - nnnn - nnnn

Inserte la cadena a Verificar>>

Hola

Cadena a Verificar: Hola Tamaño de la cadena: 6

La cadena no contiene la estructura Valida

Eliga el numero del ejercicio que desea visualizar

- 1-Ejercicio 1
- 2.-Ejercicio 2
- 3.-Ejercicio 3
- 4.-Ejercicio 4
- 5.- Ejercicio 5
- 0.-Salir

4

Ejercicio 4

Existen 10 en el recetario

Escriba los ingredientes que desea Buscar>>

Tocino

Se encontraron 0 coincidencias en 10 Listas

Eliga el numero del ejercicio que desea visualizar

- 1-Ejercicio 1

```
var op4 = proyecto2.ejeercicio5(arregloNumeros: [1,2,3,4,5])  
print(op4)  
default:  
    print("Opcion No valida")
```

Variable 'op4' was never m

Actividad2

Eliga el numero del ejercicio que desea visualizar

- 1-Ejercicio 1
- 2.-Ejercicio 2
- 3.-Ejercicio 3
- 4.-Ejercicio 4
- 5.- Ejercicio 5
- 0.-Salir

5

Ejercicio 5

[120, 60, 40, 30, 24]

Conclusiones

Para este proyecto ya tenía mayor dominio y facilidad en el lenguaje de programación Swift, por lo que fue mas sencillo realizar estos 5 programas, sólo tuve algunos tropiezos con mi lógica propia

[Repositorio de Github](#)