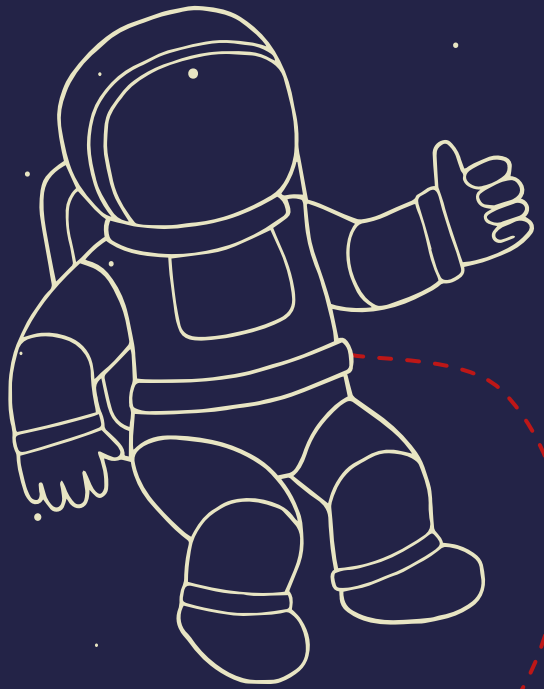
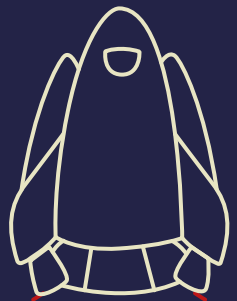
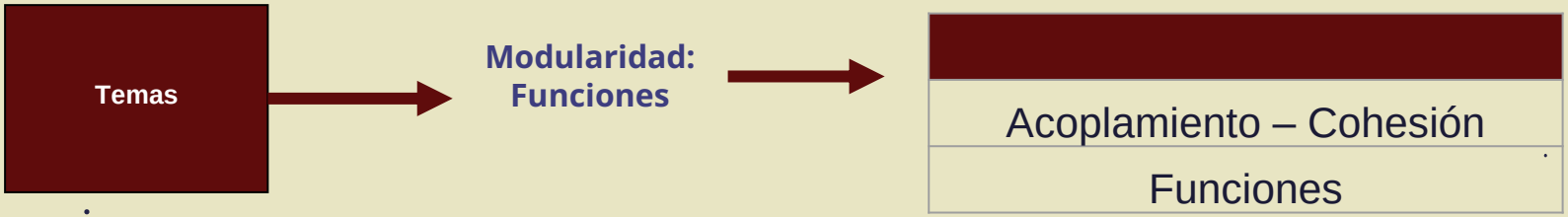


Programa académico CAMPUS

Ciclo 1: Fundamentos de Programación



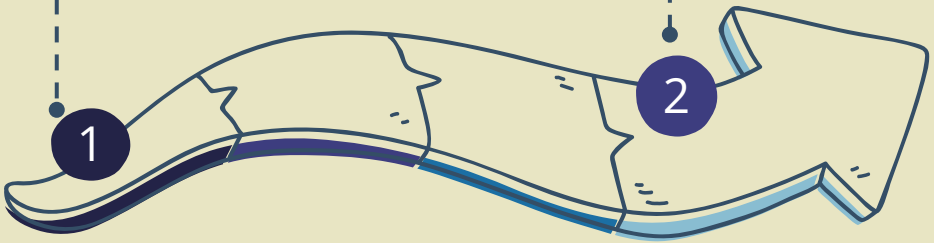
Presentación Ciclo 1 – Fundamentos de Programación



Funciones

Modularidad: Funciones

Taller



Funciones

Conceptualización

¿Has visto alguna vez una carrera de autos de fórmula uno? Pues bien, hay un momento en la competición en la que los autos deben entrar a pits. La razón es que al auto se le debe hacer un mantenimiento a las llantas y se le debe suministrar combustibles. Ambas funciones deben llevarse a cabo luego de un determinado número de vueltas, cuando el ingeniero automovilístico encargado lo determine. Supongamos que la función general de la entrada a pits es realizar ambas tareas (cambio de llantas y suministrar combustible) ejecutadas una seguida de la otra. Así pues, una función se puede definir como una secuencia de instrucciones que tiene como finalidad llevar a cabo una tarea específica; como por ejemplo, realizar la suma de dos números, contar las palabras de una cadena de caracteres, etc.

Funciones

Conceptualización

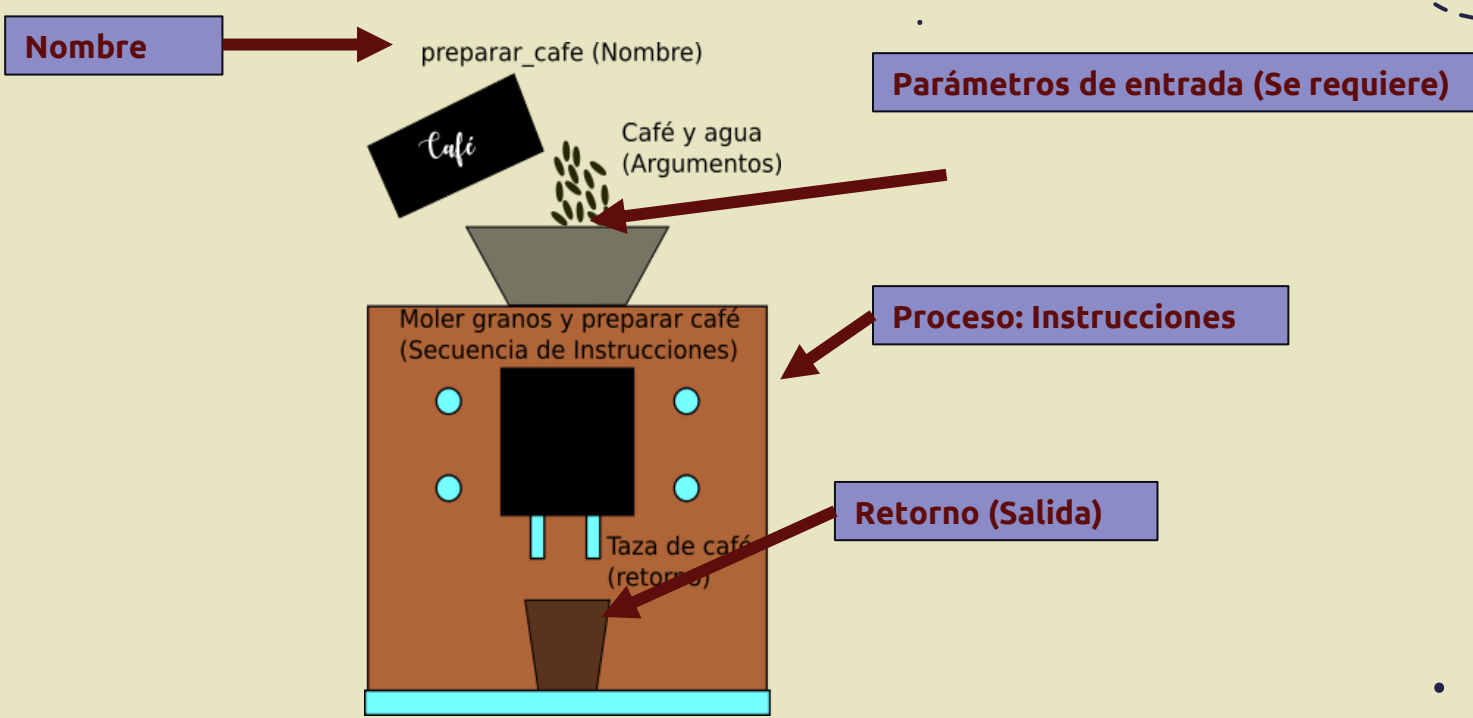
- ✓ Las **funciones en programación reciben un nombre que debe ser coherente con su función.**

- ✓ Por lo general, un programa es dividido por diferentes tipos de funciones, que llevan a cabo diferentes tipos de tareas,

- ✓ Se logra la solución de un problema más grande.

Funciones

Estructura de una Función



Estructura básica de una función haciendo analogía a la preparación de una taza de café.

Funciones

Modularidad

Uno de los aspectos fundamentales de la programación moderna, base de los nuevos paradigmas, es sin duda alguna la **modularidad**, entendida como la generación de módulos o segmentos funcionales e independientes que permitan una mejor organización y compresión de un programa. Este aspecto se basa en la aplicación de dos técnicas propias de la ingeniería del software, denominadas Acoplamiento de módulos y Cohesión de módulos que definen unas guías en la definición de un módulo.

Funciones

Acoplamiento – Cohesión de Módulos

Cohesión de módulos

La técnica de la ingeniería del software, denominada Cohesión de Módulos busca medir el grado de relación o dependencia que existe entra las actividades propias de un proceso o módulo. La finalidad es generar módulos que realicen un proceso determinado y por consiguiente las actividades o instrucciones que contine están todas relacionadas con el objetivo del módulo. Por ejemplo, un módulo de liquidación de comisiones, solo debe contener las instrucciones que permitan el calculo del valor de la comisión y no incluir otro tipo instrucciones, como las de incrementar contadores y sumadores.

Acoplamiento de módulos

La técnica La técnica del Acoplamiento de Módulos que se aplica después de la cohesión, tiene como objetivo la generación de módulos independientes dentro de un proceso, en los cuales, cada uno de ellos define sus propias variables y la comunicación con ellos se realice a través de parámetros, o sea, variables (argumentos) que recibe el módulo que le permitan realizar la función especifica para lo que fue definido. Los módulos independientes, que reciben parámetros de entrada y retornan una salida específica, permiten su reutilización en otros programas y procesos, lo que facilita el desarrollo de software.

Funciones

Ejercicio

La empresa de teléfonos de la ciudad necesita realizar su proceso de facturación en forma automática, contando con los N abonados, de los cuales conoce el nombre, estrato, que puede ser (1, 2, 3, 4, 5), cantidad de impulsos del mes (N es suministrado). Además la empresa nos informa que para la liquidación de la factura se debe tener en cuenta el valor de la tarifa básica, de acuerdo al estrato, que depende de la siguiente tabla:

Estrato	Ttarifa Básica
1	$10.000
2	$15.000
3	$20.000
4	$25.000
5	$30.000

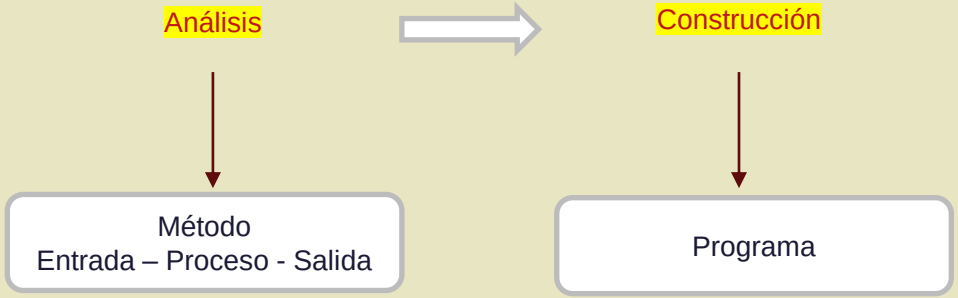
Además se debe calcular el valor de los impulsos, con base en la cantidad de impulsos del mes, conociendo que cada impulso tiene un valor de $100. Con esta información, se desea:

- ☑ Valor a pagar de cada abonado con el nombre**. También se debe visualizar la tarifa básica y el valor de los impulsos**
- ☑ Valor total a pagar(Todos los abonados)

Funciones

Ejercicio

Metodología -> Pensamiento lógico estructurado



Funciones

Ejercicio

Análisis –> Ejercicio funciones

Entrada LEER

Proceso

Salida IMPRIMIR

1

N
Iniciar sumador

nombre,estrato,impulsos

3

2

CICLO: Proceso de facturación para cada abonado=>FOR

4

CONDICIONAL: estrato, para calcular tarifa básica

valorImpulsos=impulsos*100

valorPagar=tarifaBasica+valorImpulsos

Incrementar sumador

5

6

nombre,tarifaBasica,valorImpulsos, valorPagar

totalPagar => sumador

7

Funciones

Ejercicio

Análisis - Modularidad

Parámetros de entrada

estrato
impulsos

Nombre: facturación:

4

Parte de 5

Parámetros de salida

valorPagar

FUNCIÓN: Retorno

Funciones

Programa -Versión 1 -Función

Ejercicio

```
# Programa para manejo de funciones
# Autor: Sergio Medina
# Fecha: 15/06/2022

# Definición de las funciones
def facturacion_abonado(estrato,impulsos):
    if estrato==1:
        tarifa_basica=10000
    elif estrato==2:
        tarifa_basica=15000
    elif estrato==3:
        tarifa_basica=20000
    elif estrato==4:
        tarifa_basica=25000
    else:
        tarifa_basica=30000
    valor_impulsos=impulsos*100
    valor_abonado=tarifa_basica+valor_impulsos
    return valor_abonado
#Programa principal
```

Funciones

Programa Versión 1: Principal

Ejercicio

```
#Programa principal
N=int(input("Cantidad de abonados: "))
total_abonados=0
for i in range(N):
    nombre=input("Nombre abonado: ")
    estrato=int(input("Estrato(1,2,3,4,5,6): "))
    impulsos=int(input("Impulsos: "))
    #Llamado o la ejecución de la función
    valor_abonado=facturacion_abonado(estrato,impulsos)
    total_abonados+=valor_abonado
    print("Nombre abonado: ",nombre)
    print("Valor a pagar abonado: ","{:,.2f}".format(valor_abonado))
print("Toal a pagar por todos los abonados: ","{:,.2f}".format(total_abonados))
```

Programa Versión 2 (Validaciones) Funciones

Ejercicio

```
# Definición de las funciones
def facturacion_abonado(estrato,impulsos):
    if estrato==1:
        tarifa_basica=10000
    elif estrato==2:
        tarifa_basica=15000
    elif estrato==3:
        tarifa_basica=20000
    elif estrato==4:
        tarifa_basica=25000
    else:
        tarifa_basica=30000
    valor_impulsos=impulsos*100
    valor_abonado=tarifa_basica+valor_impulsos
    return valor_abonado
```

```
def valida_entero(etiqueta):
    while True:
        try:
            dato=int(input(etiqueta))
            break
        except ValueError:
            print(etiqueta," debe ser un dato entero")
    return dato
def valida_estrato(etiqueta):
    while True:
        try:
            estrato=int(input(etiqueta))
            if estrato<1 or estrato>5:
                print(etiqueta," debe estar entre 1 y 5")
                continue
            break
        except ValueError:
            print(etiqueta," debe ser un dato entero")
    return estrato
```

Funciones

Programa Versión 2 (Validaciones) Principal

Ejercicio

```
#Programa principal
N=valida_entero("Cantidad de abonados ")
total_abonados=0
for i in range(N):
    nombre=input("Nombre abonado: ")
    estrato=valida_estrato("Estrato(1,2,3,4,5) ")
    impulsos=valida_entero("Impulsos ")
    #Llamado o la ejecución de la función
    valor_abonado=facturacion_abonado(estrato,impulsos)
    total_abonados+=valor_abonado
    print("Nombre abonado: ",nombre)
    print("Valor a pagar abonado: ","{:,.2f}".format(valor_abonado))
print("Toal a pagar por todos los abonados: ","{:,.2f}".format(total_abonados))
```

Funciones

Programa Versión 3 (Retorno varios valores) Funciones

Ejercicio

```
# Programa para manejo de funciones
# Autor: Sergio Medina
# Fecha: 15/06/2022

# Definición de las funciones
def facturacion_abonado(estrato,impulsos):
    if estrato==1:
        tarifa_basica=10000
    elif estrato==2:
        tarifa_basica=15000
    elif estrato==3:
        tarifa_basica=20000
    elif estrato==4:
        tarifa_basica=25000
    else:
        tarifa_basica=30000
    valor_impulsos=impulsos*100
    valor_abonado=tarifa_basica+valor_impulsos
    return tarifa_basica,valor_impulsos,valor_abonado
```

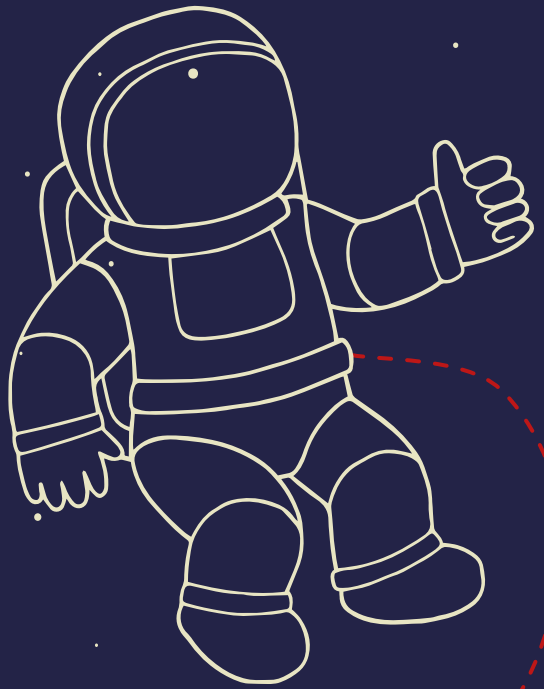
```
def valida_entero(etiqueta):
    while True:
        try:
            dato=int(input(etiqueta))
            break
        except ValueError:
            print(etiqueta," debe ser un dato entero")
    return dato
def valida_estrato(etiqueta):
    while True:
        try:
            estrato=int(input(etiqueta))
            if estrato<1 or estrato>5:
                print(etiqueta," debe estar entre 1 y 5")
                continue
            break
        except ValueError:
            print(etiqueta," debe ser un dato entero")
    return estrato
```

Funciones

Programa Versión 3 ((Retorno varios valores)) Principal

Ejercicio

```
#Programa principal
N=valida_entero("Cantidad de abonados ")
total_abonados=0
for i in range(N):
    nombre=input("Nombre abonado: ")
    estrato=valida_estrato("Estrato(1,2,3,4,5) ")
    impulsos=valida_entero("Impulsos ")
    #Llamado o la ejecución de la función
    tarifa_basica,valor_impulsos,valor_abonado=facturacion_abonado(estrato,impulsos)
    total_abonados+=valor_abonado
    print("Nombre abonado: ",nombre)
    print("Tarifa Básica: ","{:,.2f}".format(tarifa_basica))
    print("Valor impulsos: ","{:,.2f}".format(valor_impulsos))
    print("Valor a pagar abonado: ","{:,.2f}".format(valor_abonado))
print("Toal a pagar por todos los abonados: ","{:,.2f}".format(total_abonados))
```



Programa académico CAMPUS

Ciclo 1: Fundamentos de Programación