ICF 122 - Fundamentos de Programación
Ingeniería Civil Informática

Unidad 3: Estructuras algorítmicas

Facultad de Ingeniería

Universidad Andrés Bello

Los objetivos de esta unidad son:

- Aprender sobre el paradigma de la **programación estructurada**.
- Aprender a crear **estructuras condicionales**.
- Aprender a crear **estructuras de repetición**.

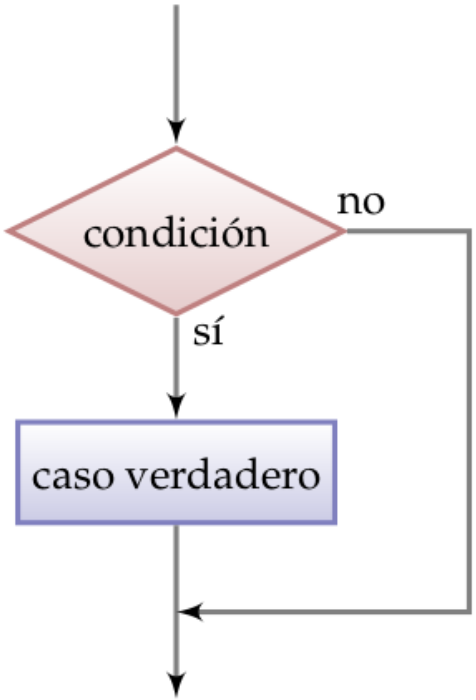
Las reglas de la programación estructurada son:

- Todo programa consiste en una serie de **sentencias** que se ejecutan en **secuencia**, una tras otra.
- Toda sentencia consiste en una **estructura de control**.
- Se consideran tres estructuras de control:
  - la **secuencia**,
  - la **selección**, y
  - la **repetición**.
- En términos de sintaxis, toda sentencia se crea mediante un punto y coma (**;**) precedido de una expresión (asignación, declaración de una variable, llamada de una función, etc.)

Estructuras condicionales

Bifurcación abierta: `if`

- La sentencia `if` ejecuta las instrucciones solo si se cumple la condición.
- Si la condición es falsa, el programa continúa.



Estructuras condicionales

Bifurcación abierta: if

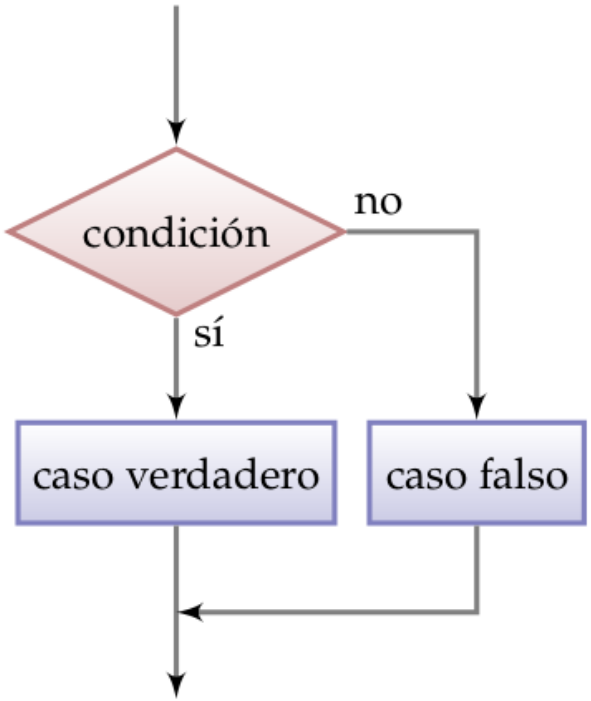
```c
#include <stdio.h>

main(){
   short num, den;
   printf("Ingrese numerador: ");
   scanf("%hd",&num);
   printf("Ingrese denominador: ");
   scanf("%hd",&den);
   if (den != 0){
      printf("%hu / %hu = %hu\n",num, den, num/den);
   }
}
```

Estructuras condicionales

Bifurcación cerrada: `if-else`

- La sentencia `if-else` decide qué instrucciones ejecutar dependiendo si una condición es verdadera o falsa.
- Si la condición es verdadera, ejecuta solo las instrucciones indicadas dentro del `if`.
- Si la condición es falsa, ejecuta solo las instrucciones indicadas dentro del `else`.



Estructuras condicionales

Bifurcación cerrada: `if-else`

```c
#include <stdio.h>

main(){
   short num, den;
   printf("Ingrese numerador: ");
   scanf("%hd",&num);
   printf("Ingrese denominador: ");
   scanf("%hd",&den);
   if (den != 0){
      printf("%hu / %hu = %hu\n",num, den, num/den);
   }
   else{
      printf("No se puede realizar division por 0.\n");
   }
}
```

Estructuras condicionales

Anidamiento

- Se produce **anidamiento** de estructuras de control, cuando una estructura de control aparece dentro de otra del mismo tipo.
- Tanto dentro del `if` como del `else`, los anidamientos pueden llegar a cualquier nivel.
- Estos anidamientos, nos permite poder escoger entre numerosas sentencias estableciendo las condiciones necesarias.
- Cada `else` se asocia al `if` más próximo en el bloque que se encuentre y que no tenga asociado un `else`.
- No está permitido utilizar un `else` sin un `if` previo, no tiene sentido.

Estructuras condicionales

Anidamiento

```
if(expresion_1)          /* primer if */
{
  if(expresion_2)        /* segundo if */
  {
    if(expresion_3)      /* tercer if */
      sentencia_1;
    else                 /* alternativa al tercer if */
      sentencia_2;
  }
  else                   /* alternativa al segundo if */
    sentencia_3;
}
else                     /* alternativa al primer if */
  sentencia_4;
```

Estructuras condicionales

Anidamiento

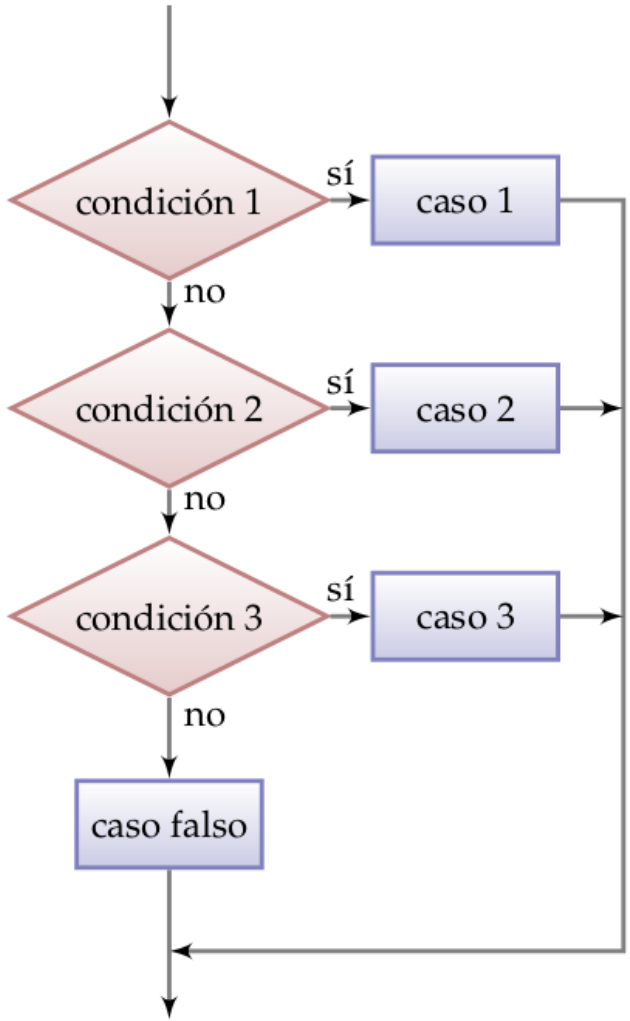
```c
#include <stdio.h>

main(){
  short num, den;
  char opcion;
  printf("Ingrese numerador: ");
  scanf("%hd",&num);
  printf("Ingrese denominador: ");
  scanf("%hd",&den);
  if (den != 0)
    printf("%hu / %hu = %hu\n",num, den, num/den);
  else{
    printf("No se puede realizar division por 0.\n");
    printf("Ingresa otro denominador? (s/n)");
    opcion = getchar();
    if(opcion == 's'){
      printf("\nNuevo denominador: ");
      scanf("%hd",&den);
      if(den != 0)
        printf("%hu / %hu = %hu\n",num, den, num/den);
      else
        printf("Nuevamente ha ingresado 0.\n");
    }
  }
}
```

Estructuras condicionales

Sentencia `if-else-if`

- La sentencia `if-else-if` depende de dos o más condiciones, que son evaluadas en orden.
- La primera que sea verdadera determina qué instrucciones serán ejecutadas.
- Si ninguna condición es verdadera, se ejecutan las instrucciones indicadas dentro del `else`.



Estructuras condicionales

Sentencia if-else-if

```c
#include <stdio.h>

main(){
  float nota;
  printf("Ingrese nota: ");
  scanf("%f",&nota);
  if (nota < 1.0 || nota > 7.0)
    printf("Nota invalida.\n");
  else if (nota < 2.0)
    printf("Su nota es pesima.\n");
  else if (nota < 4.0)
    printf("Su nota es mediocre.\n");
  else if (nota < 6.0)
    printf("Su nota es aceptable.\n");
  else
    printf("Su nota es excelente.\n");
}
```

Estructuras condicionales

Operador condicional

- Existe un operador, que de forma muy sencilla y bajo ciertas limitaciones, realiza la misma operación que una bifurcación cerrada.
- El operador se denomina **interrogante - dos puntos** (`?:`)
- Su sintaxis es

```
expresion_1 ? expresion_2 : expresion_3;
```

- Se evalúa `expresion_1`, si es verdadera, entonces se ejecuta la sentencia `expresion_2`, sino se ejecuta la sentencia `expresion_3`.
- Por ejemplo, el código

```
if(x >= 0)
  printf("Positivo\n");
else
  printf("Negativo\n");
```

equivale a

```
printf("%s\n", x >= 0 ? "Positivo": "Negativo");
```

Estructuras Condicionales

Selección múltiple: `switch`

- La sentencia `switch` ejecuta las sentencias, a partir de la etiqueta `case` que coincide con el valor de `var_switch`.

```
switch (var_switch){
  case expresionConstante1:
    sentencias
    break;
  case expresionConstante2:
    sentencias
    break;
  ...
  default:
    sentencias
}
```

- Una etiqueta `case` es una constante entera (`char`, `short` o `long`).
- Todas las expresiones constantes deben ser diferentes.
- El caso etiquetado `default` se ejecuta si ninguno de los casos anteriores coincide.

Estructuras Condicionales

Selección múltiple: `switch`

```c
#include <stdio.h>

main(){
  char letra;
  printf("Ingrese vocal:");
  scanf("%c",&letra);
  switch (letra){
    case 'a':
      printf("Ha ingresado a.");
      break;
    case 'e':
      printf("Ha ingresado e.");
      break;
    case 'i':
      printf("Ha ingresado i.");
      break;
    case 'o':
      printf("Ha ingresado o.");
      break;
    default:
      printf("Ha ingresado u.");
  }
}
/*En las distintas entradas case, no se permite expresiones logicas o
relacionales, ni expresiones aritmeticas, solo literales.*/
```

Ejercicio 1

Escriba un programa que determine si un número entero ingresado es múltiplo de 7.

```
Ingrese un numero: 46
Su numero es: 46

Ingrese un numero: 56
Es multiplo de 7
Su numero es: 56

Ingrese un numero: 343
Es multiplo de 7
Su numero es: 343
```

Ejercicio 2

Escriba un programa que determine si un número entero ingresado es par o impar.

```
Ingrese un numero: 4
Su numero es par

Ingrese un numero: 3
Su numero es impar

Ingrese un numero: 78
Su numero es par

Ingrese un numero: 9653
Su numero es impar
```

Ejercicio 3

Escriba un programa que determine si un número está dentro de un rango $[a, b]$, donde a y b son ingresados por el usuario.

```
Ingrese a: 3
Ingrese b: 9
Ingrese un numero: 8
Su numero esta en el rango

Ingrese a: 5
Ingrese b: 25
Ingrese un numero: 43
Su numero no esta en el rango
```

Ejercicio 4

Escriba un programa que determine si un número es negativo, positivo o cero.

```
Ingrese un numero: 45
Su numero es positivo.

Ingrese un numero: 0
Su numero es cero.

Ingrese un numero: -87
Su numero es negativo.
```

Ejercicio 5

La tasa de impuesto a pagar por una persona según su sueldo, viene dada por la Tabla 1. Escriba un programa que calcula el impuesto a pagar.

Sueldo	*Tasa de impuesto*
menos de 1000	0 %
$1000 \leq$ *sueldo* < 2000	5 %
$2000 \leq$ *sueldo* < 4000	10 %
4000 *o más*	12 %

Tabla 1: Tasa de impuesto en función del sueldo.

Ejercicio 6

Un año es bisiesto si es divisible por 4, excepto si es divisible por 100 y no por 400. Escriba un programa que indique si un año es bisiesto.

```
Ingrese un anio: 1988
1988 es bisiesto
```

```
Ingrese un anio: 2011
2011 no es bisiesto
```

```
Ingrese un anio: 1700
1700 no es bisiesto
```

```
Ingrese un anio: 2400
2400 es bisiesto
```

Ejercicio 7

Los tres lados a, b y c de un triángulo deben satisfacer la desigualdad triangular: cada uno de los lados no puede ser más largo que la suma de los otros dos. Escriba un programa que reciba como entrada los tres lados de un triángulo, e indique si acaso el triángulo es inválido; y si no lo es, qué tipo de triángulo es.

```
Ingrese a: 3.9
Ingrese b: 6.0
Ingrese c: 1.2
No es un triangulo valido.

Ingrese a: 1.9
Ingrese b: 2
Ingrese c: 2
El triangulo es isoceles.
```

Ejercicio 8

Escriba un programa que reciba como entrada ahora tres números, y los muestre ordenados de menor a mayor:

```
Ingrese numero: 8
Ingrese numero: 1
Ingrese numero: 4
1 4 8

Ingrese numero: 9
Ingrese numero: 11
Ingrese numero: 6
6 9 11
```

Estructuras de repetición

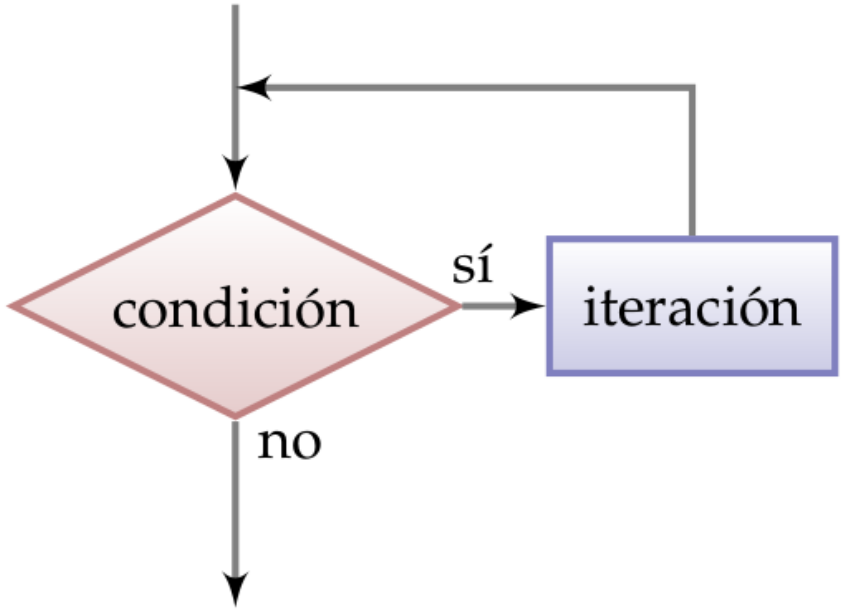
Introducción

- Una estructura de **repetición** o de **iteración** es aquella que nos permite repetir un conjunto de sentencias mientras que se cumpla una determinada condición.
- Existen tres estructuras de reptición en C: `do-while`, `while` y `for`.
- Todas estas estructuras permiten la anidación de unas dentro de otras a cualquier nivel.

Estructuras de repetición

Sentencia `while`

- La sentencia `while` ejecuta una secuencia de instrucciones mientras una condición sea verdadera.
- No se conoce por adelantado la cantidad de iteraciones.
- La condición es evaluada antes de cada iteración.
- Si la condición, inicialmente es falsa, las instrucciones no se ejecutarán ninguna vez.



Ejemplo 1

Tabla de multiplicar de un número entero dado.

```
#include <stdio.h>
main(){
   short int i, n;
   printf("Tabla de multiplicar de: ");
   scanf("%hd",&n);
   i = 0;
   while(i <= 10){
      printf("%3hu * %3hu = %3hu\n",i, n, i*n);
      i++;
   }
}
```

Ejemplo 2

Suma entre dos números enteros a y b.

```
#include <stdio.h>
main(){
   int a, b, i, suma = 0;
   printf("a: ");
   scanf("%d",&a);
   printf("b: ");
   scanf("%d",&b);
   i = a;
   while (i <= b){
      suma = suma + i;
      i++;
   }
   printf("La suma entre %d y %d es: %d",a, b, suma);
}
```

Ejemplo 3

Cálculo del factorial de un entero n, el cual viene dado por

$$n! = n \times (n-1) \times \cdots \times 2 \times 1 \tag{1}$$

```
#include <stdio.h>
main(){
  unsigned short n;
  unsigned long fact;
  printf("Ingrese n: ");
  scanf("%hu",&n);
  printf("El factorial de %hu es ", n);
  fact = 1;
  while(n != 0){
    fact = fact * n;
    n = n - 1;
  }
  printf("%lu.", fact);
}
```

- La sentencia `do-while` es muy similar a la anterior.
- La diferencia está en que las instrucciones se ejecutan **al menos una vez.**
- Si después de haberse ejecutado, la condición se cumple, entonces vuelve a ejecutarse, y así hasta que la condición sea falsa.

```
do{
   sentencia_1;
   sentencia_2;
   ...
   sentencia_N;
}while(condicion);
```

Ejemplo 4

Tabla de multiplicar de un número entero dado.

```
#include <stdio.h>
main(){
   short int i = 0, n;
   printf("Tabla de multiplicar de: ");
   scanf("%hd",&n);
   do
   {
      printf("%3hu * %3hu = %3hu\n",i, n, i*n);
      i++;
   }while(i <= 10);
}
```

Estructuras de repetición

Sentencia `for`

- Tiene una sintaxis notablemente distinta a las sentencias anteriores.
- Con la sentencia `for` podemos crear estructuras de control que se dicen **"controladas por variable"**.

```
for(sentencias_1; expresion; sentencias_2)
  sentencias_3;
```

- Donde `sentencias_3` son las **sentencias que se iteran**.
- Donde `sentencias_1` son las **sentencias que se ejecutan antes que ninguna otra** y siempre se ejecutan solamente una vez.
- Donde `expresion` es la **condición de permanencia** en la estructura. Siempre que se cumpla esta expresión se vuelve a ejecutar la sentencia de iteración.
- Donde `sentencias_2` son las sentencias que se ejecutan **después de la sentencia de iteración**.

La sentencia `for`

```
for(sentencias_1; expresion; sentencias_2)
  sentencias_3;
```

equivale a

```
sentencias_1;
while(expresion)
{
  sentencias_3;
  sentencias_2;
}
```

Ejemplo 5

Números pares del 1 al 100

```
#include <stdio.h>
main(){
  short i;
  for(i = 2; i <= 100; i += 2)
    printf("%5hd",i);
}
```

Ejemplo 6

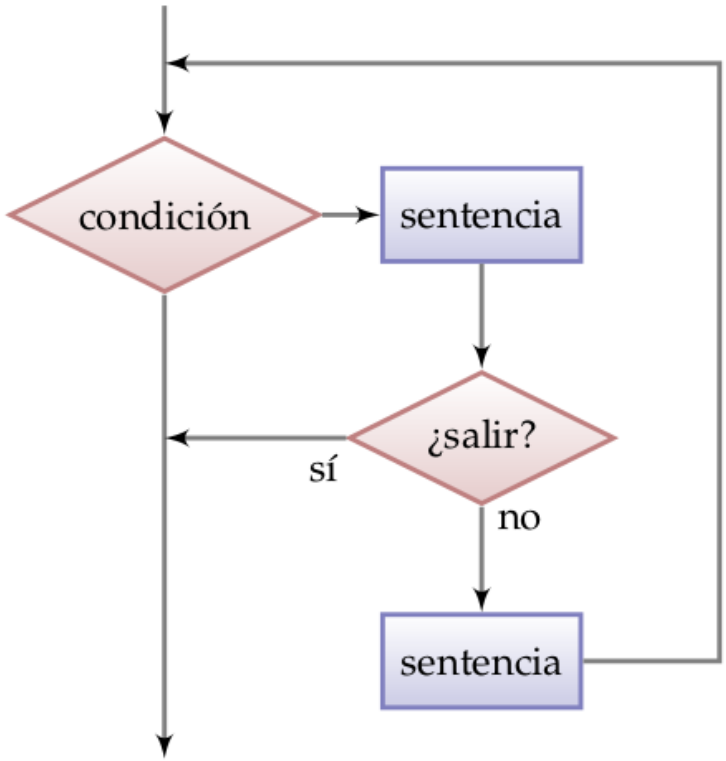
Tabla de multiplicar de un número entero dado.

```
#include <stdio.h>
main(){
  short int i, n;
  printf("Tabla de multiplicar de: ");
  scanf("%hd",&n);
  for(i = 0; i <= 10; i++)
    printf("%3hu * %3hu = %3hu\n",i, n, i*n);
}
```

Estructuras de repetición

Sentencia `break`

- Siempre es posible salir de una estructura de repetición, en medio de una iteración utilizando la sentencia `break`.
- Es lógico que el `break` aparezca siempre dentro de un `if`, sino el ciclo terminaría siempre en la primera iteración.



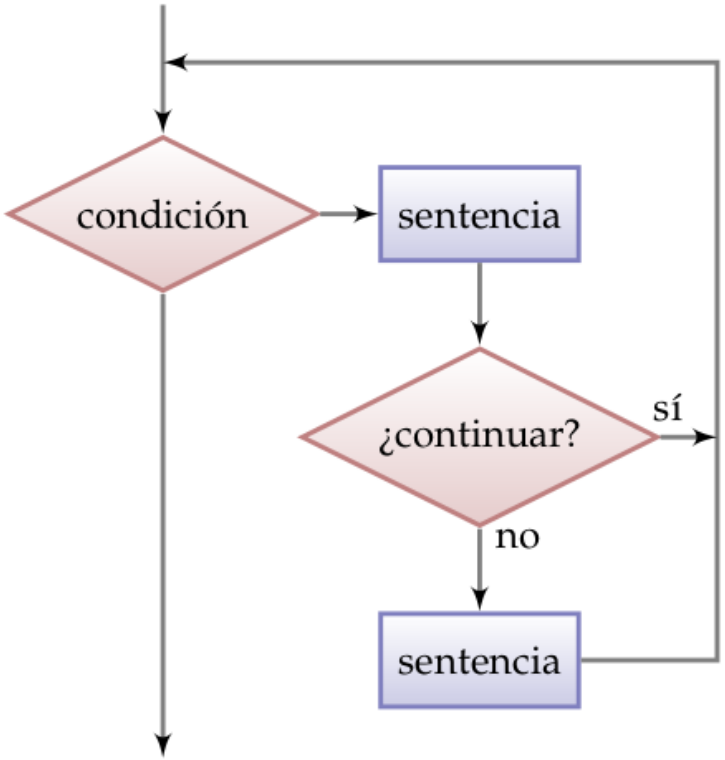
Ejemplo 7

Suma de los números ingresados hasta que alguno sea negativo.

```
#include <stdio.h>

main(){
   int num = 0, suma = 0;
   printf("Ingrese numeros: \n");
   while(1){
      scanf("%d",&num);
      if (num < 0)
         break;
      suma += num;
   }
   printf("%d\n",suma);
}
```

- La sentencia `continue` se utiliza para saltar a la siguiente iteración sin llegar al final de la que está en curso.



Ejemplo 8

Suma de los primeros `n` *números, sin considerar los múltiplos de 7.*

```c
#include <stdio.h>

main(){
  int i = 1, n, suma = 0;
  printf("Ingrese n: ");
  scanf("%d",&n);
  while(i <= n){
    if (i % 7 == 0){
      i++;
      continue;
    }
    printf("i = %d\n",i);
    suma += i;
    i++;
  }
  printf("%d\n",suma);
}
```

Ejercicio 9

Escriba un programa que calcule la suma de los pares positivos menores o igual a 200.

Ejercicio 10

Escriba un programa que calcule la media de todos los valores que ingrese el usuario. El programa debe dejar de solicitar valores cuando el usuario introduzca el valor 0.

```
Ingrese valores:
2
3
6
0
Media: 3.66
```

Ejercicio 11

Juan Carnicer está preocupado, ya que este fin de semana ha decidido realizar un tremendo asado entre amigos y familiares. El problema es que cada invitado llegará con una cantidad desconocida de carne. Juan, debe saber rápidamente cuántos kilos y gramos tendrá en total. Ayude a Juan, considerando que cada invitado llegará con `p` *kilos y* `q` *gramos de carne, a calcular el total obtenido.*

```
kilos: 2
gramos: 430
kilos: 3
gramos: 560
kilos: 1
gramos: 670
kilos: 0

Total: 7 kilos y 660 gramos.
```