

# CS1112: Programación II

## Unidad 2: Funciones y recursividad

### Sesión de Laboratorio - 3A

#### Profesores:

**María Hilda Bermejo** [mbermejo@utec.edu.pe](mailto:mbermejo@utec.edu.pe)  
**Estanislao Contreras** [econtreras@utec.edu.pe](mailto:econtreras@utec.edu.pe)  
**Jorge Villavicencio** [jvillavicencio@utec.edu.pe](mailto:jvillavicencio@utec.edu.pe)  
**Edson Mendiola** [emendiola@utec.edu.pe](mailto:emendiola@utec.edu.pe)  
**Ian Paul Brossard** [ibrossard@utec.edu.pe](mailto:ibrossard@utec.edu.pe)  
**Jose Chavez** [jchaveza@utec.edu.pe](mailto:jchaveza@utec.edu.pe)  
**Julio Yarasca** [jyarascam@utec.edu.pe](mailto:jyarascam@utec.edu.pe)  
**Percy Quevedo** [pquevedo@utec.edu.pe](mailto:pquevedo@utec.edu.pe)  
**Wilder Nina** [wnina@utec.edu.pe](mailto:wnina@utec.edu.pe)  
**José Fiestas** [jfiestas@utec.edu.pe](mailto:jfiestas@utec.edu.pe)

Material elaborado por:

**María Hilda Bermejo, Jaime Farfán**



# Índice:

- Unidad 2: Funciones y recursividad
  - Ámbito de una variable
  - Paso de parámetros a funciones. Transferencia por valor y por referencia

# 2.1 Funciones

The background of the slide is a photograph of a modern, multi-story building with a curved facade and many balconies. The entire image is covered with a semi-transparent blue filter. Overlaid on this background is the text '2.1 Funciones' in a large, white, sans-serif font. The '2.1' is significantly larger than the word 'Funciones'. In the bottom right corner of the building, the letters 'UTEC' are visible.

# Logro de la sesión:

Al finalizar la sesión, los alumnos:

- Entienden el concepto de variables globales.
- Comprenden el concepto de transferencia por valor y por referencia.
- Usan funciones recursivas y no-recursivas

# Ejemplo 1:

Desarrolle una función que reciba como parámetro una cantidad de segundos y realice la conversión a su equivalente en horas, minutos y segundos.

Ejemplo:

Segundos: 3850

Conversión: 1h 4m 10s

# Ejemplo 1: Solución

A continuación se proporciona la función principal para solicitar el total de segundos como dato de entrada y luego llama a la función *Convertir* para realizar la conversión a horas, minutos y segundos.

```
int main()
{
    long int segundos;
    long int horas, min, seg;
    do{
        cout << "Segundos : ";
        cin >> segundos;
    }while(segundos < 1);
    Convertir(segundos, horas, min, seg); //implementar esta función
    cout << "Equivalen a : ";
    cout << horas << ":" << min << ":" << seg;
}
```



# Ejemplo 1: Solución

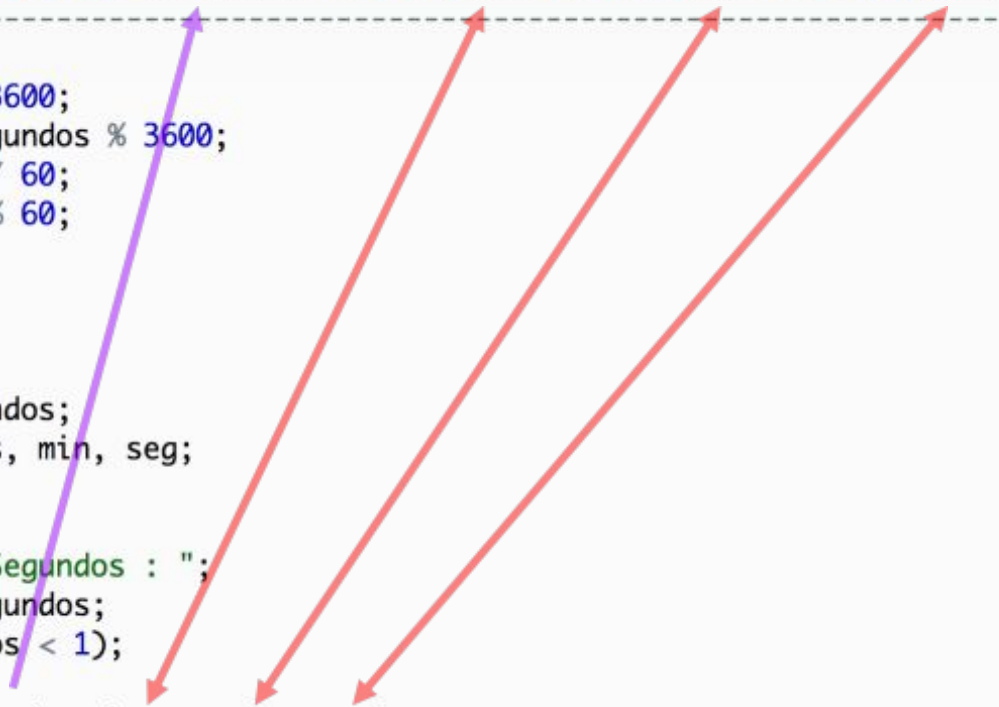
```
5  #include <iostream>
6  using namespace std;
7
8  void Convertir(long int segundos, long int &h, long int &m, long int &s )
9  //-----
10 {
11     h = segundos/3600;
12     segundos = segundos % 3600;
13     m = segundos / 60;
14     s = segundos % 60;
15 }
16
17 int main()
18 {
19     long int segundos;
20     long int horas, min, seg;
21
22     do{
23         cout << "Segundos : ";
24         cin >> segundos;
25     }while(segundos < 1);
26
27     Convertir(segundos, horas, min, seg);
28     cout << "Equivalen a : ";
29     cout << horas << ":" << min << ":" << seg;
30     return(0);
31 }
```

Se hace llamada por referencia a los argumentos de la función.

No sería posible realizar la función de otra manera, ya que en el nombre de la función, solo se puede devolver un solo valor y no 3 como es necesario en este caso.

# Ejemplo 1: Solución

```
5  #include <iostream>
6  using namespace std;
7
8  void Convertir(long int segundos, long int &h, long int &m, long int &s )
9  //-----
10 {
11     h = segundos/3600;
12     segundos = segundos % 3600;
13     m = segundos / 60;
14     s = segundos % 60;
15 }
16
17 int main()
18 {
19     long int segundos;
20     long int horas, min, seg;
21
22     do{
23         cout << "Segundos : ";
24         cin >> segundos;
25     }while(segundos < 1);
26
27     Convertir(segundos, horas, min, seg);
28     cout << "Equivalen a : ";
29     cout << horas << ":" << min << ":" << seg;
30     return(0);
31 }
```





## Ejemplo 2:

Construir una función que imprima en pantalla un menú y permita elegir una opción al usuario.

```
Elija una opción:  
S. Sumar  
R. Restar  
M. Media Aritmética
```

```
—
```

# Ejemplo 2:

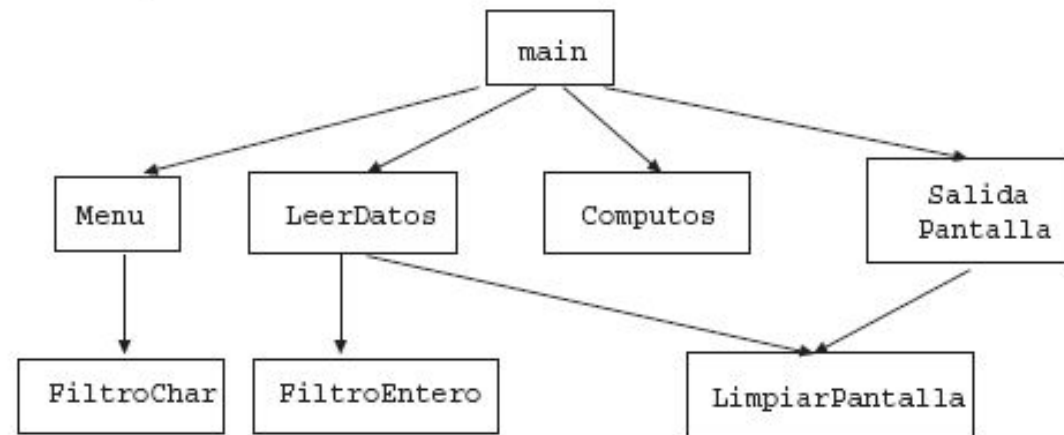
Construir una función que imprima en pantalla un menú y permita elegir una opción al usuario.

Con una función que devuelve un char.	Con una función void.
<pre>#include &lt;iostream&gt; using namespace std; char Menu(){     char tecla;     cout &lt;&lt; "\nElija una opción:\n";     cout &lt;&lt; "\nS. Sumar";     cout &lt;&lt; "\nR. Restar";     cout &lt;&lt; "\nM. Media Aritmética\n";     cin &gt;&gt; tecla;     return tecla; } int main(){     char opcion;     opcion = Menu();     switch (opcion)     ..... }</pre>	<pre>#include &lt;iostream&gt; using namespace std; void Menu(char &amp;tecla){     cout &lt;&lt; "\nElija una opción:\n";     cout &lt;&lt; "\nS. Sumar";     cout &lt;&lt; "\nR. Restar";     cout &lt;&lt; "\nM. Media Aritmética\n";     cin &gt;&gt; tecla; } int main(){     char opcion;     Menu(opcion);     switch (opcion)     ..... }</pre>

## Ejemplo 2:

Ahora el programa debe permitir sumar, restar o hallar la media aritmética de dos números enteros. Se debe considerar funciones de validación de los datos de entrada y un menú de opciones.

Descomposición modular:



# Ejemplo 2: Solución

Los prototipos de las funciones serían:

```
void LimpiarPantalla();  
char FiltroChar();  
int FiltroEntero();  
char Menu();  
void LeerDatos (int &primero, int &segundo);  
double Computos (char tecla, int primero, int segundo);  
void SalidaPantalla (double final);
```

# Ejemplo 2: Solución

Nivel 1: Construimos el esqueleto de la función main (todavía no se han definido las funciones)

```
#include <iostream>
using namespace std;
void LeerDatos (int &primero, int &segundo);
char Menu ();
double Computos (char tecla, int primero, int segundo);
void SalidaPantalla (double final);

int main(){
    int dato1, dato2;
    double final;
    char opcion;
    LeerDatos(dato1, dato2);
    opcion = Menu();
    final = Computos(opcion, dato1, dato2);
    SalidaPantalla(final);
}
```

# Ejemplo 2: Solución

Nivel 2: Construimos el cuerpo de cada una de las funciones llamadas en el main. Por ejemplo:

```
char FiltroChar();
```

```
char Menu(){  
    char tecla;  
    cout << "\nElija una opción\n";  
    cout << "\nS. Sumar";  
    cout << "\nR. Restar";  
    cout << "\nM. Media Aritmética\n";  
    tecla = FiltroChar();  
    return tecla;  
}
```



# Ejemplo 2: Solución

Nivel 2: Construimos el cuerpo de cada una de las funciones llamadas en el main. Por ejemplo:

```
void LimpiarPantalla();  
int FiltroEntero();  
  
void LeerDatos(int &primero, int &segundo){  
    LimpiarPantalla();  
    primero = FiltroEntero();  
    segundo = FiltroEntero();  
}
```

## Ejemplo 3:

Un número es capicúa, si se lee igual de izquierda a derecha que de derecha a izquierda, por ejemplo los siguientes números son capicúas 34443, 124421. La compañía telefónica “Movistar” desea saber cuáles de estos 10 números telefónicos son capicúas:

*(900464009, 914103031, 925979529 , 935935606, 963839025 ,966173734, 978119539, 982727289, 989950857, 999958817).*

Respuesta : Los siguientes son capicúas (900464009, 925979529, 982727289)

# Ejemplo 3: Solución

Construimos el cuerpo de cada una de las funciones llamadas en el main.

```
int invertir(int n){  
    // Escriba su código aquí  
}  
  
bool es_capicua(int n){  
    // Escriba su código aquí  
}  
  
int main (){  
    for( int i =0; i <10;i++){  
        int n;  
        cin >> n;  
        bool result = EsCapicua(n);  
        // Escriba su código aquí  
    }  
}
```

# Ejemplo 3: Solución

```
int invertir(int n){
    int result = 0;
    while(n!=0){
        result = result*10+n%10;
        n=n/10;
    }
    return result;
}

bool es_capicua(int n){
    return n == invertir(n);
}

int main (){
    for( int i =0; i <10;i++){
        int n;
        cin >> n;
        bool result = es_capicua(n);
        if(result)
            cout<<n<<endl;
    }
    return 0;
}
```

## Ejemplo 4:

Escriba un código que pida un número entero y determine el triple, la raíz cuadrada (con una precisión de dos decimales) y si el número es múltiplo de 3. El main() solo debe consistir en la declaración de variables y la llamada a funciones. Use paso por valor y referencia.

```
Ingrese un numero entero:  
6  
El triple es 18  
La raiz cuadrada es 2.45  
Múltiplo de 3: true
```

```
Ingrese un numero entero:  
20  
El triple es 60  
La raiz cuadrada es 4.47  
Múltiplo de 3: false
```

# Ejercicios





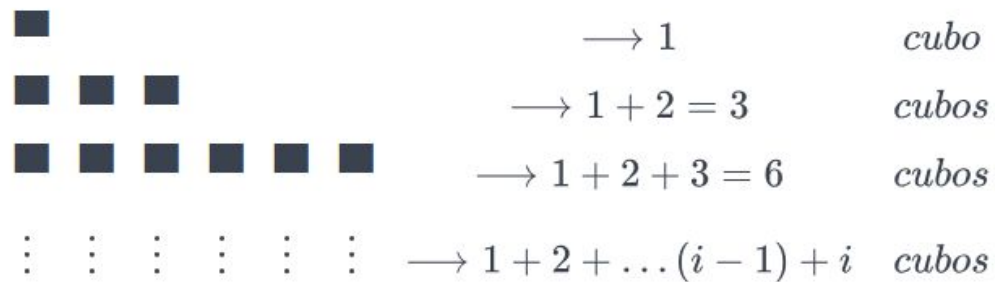
# Ejercicio 1:

Alessia es profesora del centro de estimulación “Bebochos” quien tiene  $n$  cubos del mismo tamaño y color, y con la ayuda de los niños decide construir una pirámide con ellos, la pirámide a construir debe tener la siguiente característica:

- El nivel superior de la pirámide debe consistir de 1 cubo.
- El segundo nivel debe consistir de  $1 + 2 = 3$  cubos.
- El tercer nivel debe tener  $1 + 2 + 3 = 6$  cubos.
- El cuarto nivel debe tener  $1 + 2 + 3 + 4 = 10$  cubos.
- El quinto nivel debe tener  $1 + 2 + 3 + 4 + 5 = 15$  cubos.

# Ejercicio 1: Solución

Tal como se puede observar en la siguiente gráfica:



donde :

$i$  = representa el  $i$  - ésimo nivel(altura) de la pirámide

Además, Alessia quiere saber cuál es la altura máxima de la pirámide que se puede hacer usando los  $N$  cubos que dispone y además saber cuántos cubos sobran (si las hay).

# Ejercicio 1: Solución

Los prototipos de las funciones serían:

```
1  #include <iostream>
2  using std::cout;
3  using std::cin;
4  using std::endl;
5
6  void imprimirCubos(int n);
7  int  calcularAltura(int n);
8  void totalCubosXAltura(int &totalCubos, int altura);
9
```

# Ejercicio 1: Solución

```
10 int main(){
11     int numero, altura=0, cubos=0;
12     cout<<"Ingrese numero de cubos: ";
13     cin >> numero;
14     altura = calcularAltura(numero);
15     totalCubosXAltura(cubos, altura);
16     imprimirCubos(altura);
17     cout << "La altura es de: " << altura << endl;
18     cout << "Cubos sobrantes: " << numero - cubos<< endl;
19     return 0;
20 }
```

## Ejercicio 2:

Pedir del usuario una nota desde 0 a 20, validarlo y finalmente mostrar una barra de progreso de 0% a 100%.

Ejemplo:

Mostrar Barra de Progreso:

0%|||||

70.0/100%

Mostrar Barra de Progreso:

0%|||||

65.0/100%

## Ejercicio 2: Solución

```
1  #include <iostream>
2  #include <string>
3  using std::cout;
4  using std::endl;
5  using std::cin;
6  using std::string;
7  using std::to_string;
8
9  void filtrarNota(int &nota);
10 string barraDeProgreso(float porcentaje);
```



## Ejercicio 2: Solución

```
35  int main() {  
36      int nota;  
37      filtrarNota(nota);  
38      cout<<"Mostrar Barra de Progreso:\n";  
39      float porcentaje = (float)nota/20;  
40      string barra = barraDeProgreso(porcentaje);  
41      cout<<barra<<"\n";  
42      return 0;  
43  }  
44
```

## Ejercicio 3:

El método de Newton para el cálculo de PI se basa en las siguientes ecuaciones:

$$\arcsen\left(\frac{1}{2}\right) = \frac{\pi}{6}$$

$$\arcsen(x) = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \cdot \frac{x^5}{5} + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot \frac{x^7}{7} + \dots$$

Desarrollar una función que calcule PI.

# Lo aprendido hoy:

- Las funciones nos permiten realizar una tarea específica.
- Las funciones reciben valores en los parámetros de entrada y retornan un valor.
- La recursividad es una característica que permite a un subprograma invocarse a sí mismo.
- Cualquier proceso iterativo puede expresarse en forma recursiva y viceversa.

# Bibliografía:

Deitel. P.J. and Deitel. H. M. (2016) C++ How to Program, Prentice Hall.

Stroustrup, Bjarne (2013). The C++ Programming Language, 4th Addison-Wesley.

Eckel, Bruce, 2000. Thinking in C++, Vol 1: Introduction to Standard C++, 2nd Edition, Prentice Hall

¡Nos vemos en la siguiente  
clase!

