

# Funciones y Arreglos

Programacion en C++

Taller de programación



**Departamento de Informática**  
Universidad Técnica Federico Santa María

# Operadores compuestos

Si  $a = 10$ ,  $b = 20$ ;

Operador	Expresión	Equivalente a	Resultado
+=	$a += b$	$a = a + b$	a vale 30, b vale 20
-=	$a -= b$	$a = a - b$	a vale -10, b vale 20
*=	$a *= b$	$a = a * b$	a vale 200, b vale 20
/=	$b /= a$	$b = b / a$	a vale 10, b vale 2

# Funciones

# Definición de funciones

Una función es una agrupación lógica de instrucciones:

---

```
tipo_de_retorno  nombre_de_funcion ( parametros )  
{  
    declaracion_de_variables_locales  
    instrucciones  
}
```

---

- ▶ Las funciones no se pueden anidar.
- ▶ Todas las funciones son externas, se pueden llamar desde cualquier punto del programa.

# Ejemplo

## Definición de una función:

---

```
int sumar(int a, int b)
{
    int r;
    r = a+b;
    return r;
}
```

---

- ▶ Tipo de retorno:  
`int` sumar(int a, int b) ← esta función retorna un entero.
- ▶ Nombre de función:  
int **sumar**(int a, int b) ← esta función se llama sumar.
- ▶ Parámetros de la función:  
int sumar(**int a, int b**) ← esta función recibe dos enteros como parámetros, a y b.
- ▶ Variables locales: `int r;` ← se declaro una variable entera llamada r.
- ▶ Valor retornado: `return r;` ← se retorna el valor de r.

En C++ siempre se ejecuta primero el código del main. Entonces, ¿qué imprime el siguiente programa?

```
#include <iostream>
```

```
using namespace std;
```

```
int sumar(int a, int b)
```

```
{
```

```
    int r;
```

```
    r = a+b;
```

```
    return r;
```

```
}
```

```
int main()
```

```
{
```

```
    int z;
```

```
    z = sumar(5,3);
```

```
    cout << "El resultado es " << z << endl;
```

```
    return 0;
```

```
}
```

main	sumar		
z	a	b	r
"sumar(5,3)"			

# Ruteo

En C++ siempre se ejecuta primero el código del `main`. Entonces, ¿qué imprime el siguiente programa?

```
#include <iostream>
using namespace std;

int sumar(int a, int b)
{
    int r;
    r = a+b;
    return r;
}

int main()
{
    int z;
    z = sumar(5,3);
    cout << "El resultado es " << z << endl;
    return 0;
}
```

main	sumar		
	a	b	r
z			
"sumar(5,3)"			
	5	3	
			8
8			

Respuesta: El resultado es 8

# Tipo de retorno

- Una función puede devolver otros tipos de valores: char, double, float, bool

---

```
float raiz1(int a, int b, int c) {  
    float det = b*b - 4*a*c;  
    if (det >= 0){  
        return ((-1 * b) + sqrt(det)) / (2 * a);  
    }  
}
```

---

- Una función puede no retornar valores: void

---

```
void imprimir_edad(int edad) {  
    if (edad >= 0){ cout << "Tienes " << edad << " annos" << endl; }  
    else { cout << "Error! edad mal ingresada!" << endl; }  
}  
int main() {  
    int edad;  
    cout << "Ingresa su edad" << endl;  
    cin >> edad;  
    imprimir_edad(edad);  
    return 0;  
}
```

---



# Variables locales y globales

- ▶ Una variable que se **declara fuera** de las funciones es una **variable global** y puede ser utilizada dentro de todas las funciones del programa.
- ▶ Una variable que se **declara dentro** de una función es una **variable local** y solamente puede ser utilizada dentro de dicha función. Al terminar la ejecución de la función desaparece.

---

```
int un_numero;           // un_numero es una variable global
float otro_numero;       // otro_numero es una variable global

int main (){
    int edad;  float anno;    // edad y anno son variables locales al main
    ...
    un_numero = 1;
    cout << un_numero << endl;
    ...
    cout << "Ingresa tu edad: ";
    cin >> edad;
    ...
}
```

---

# Declaración de funciones (prototipos)

Prototipo de una función:

---

```
tipo_de_retorno nombre_de_funcion ( parametros );
```

---

- ▶ Se pueden omitir los nombres de los parámetros y dejar solamente los tipos.
- ▶ No es necesario declarar las funciones si las definimos antes de usarlas.

# Ejemplo de prototipos

---

```
#include <iostream>
using namespace std;

void par(int a);
void impar(int a);

int main ()
{
    int i = -1;
    while (i != 0) {
        cout << "Ingresa un numero (0 para salir): ";
        cin >> i;
        impar(i);
    }
    return 0;
}
...
```

---

Faltan las definiciones de par, impar ...

# Ejemplo de prototipos

```
...  
void impar (int a)  
{  
    if ((a % 2) != 0) {  
        cout << "El numero es impar.\n";  
    }  
    else {  
        par(a);  
    }  
}  
  
void par (int a)  
{  
    if ((a % 2) == 0) {  
        cout << "El numero es par.\n";  
    }  
    else {  
        impar(a);  
    }  
}
```

# Ejercicios

1. Escriba la función `invertir_digitos(n)` que reciba un número entero `n` y entregue como resultado el número `n` con los dígitos en el orden inverso:

---

```
int invertir_digitos(int n){  
    ...  
}  
  
int main(){  
    int n;  
    cout << "Ingrese un numero: ";  
    cin >> n;  
    cout << invertir_digitos(n);  
    return 0;  
}
```

---

Por ejemplo, `cout << invertir_digitos(142);` imprime 241.

2. A continuación, escriba un programa que indique si el número ingresado es palíndromo o no, usando la función `invertir_digitos`:

---

```
Ingrese n: 81418  
Es palindromo
```

---

# Ejercicios

1. Escriba la función `es_divisible(n, d)` que indique si `n` es divisible por `d`:

---

```
bool es_divisible(int a, int b){ ... }  
...  
cout << es_divisible(15, 5);    // imprime 1 (representa true)  
cout << es_divisible(15, 6);    // imprime 0 (representa false)
```

---

2. Usando la función `es_divisible`, escriba una función `es_primo(n)` que determine si un número es primo o no:

---

```
bool es_primo(int a){ ... }  
...  
cout << es_primo(17);           // imprime 1 (representa true)  
cout << es_primo(221);          // imprime 0 (representa false)
```

---

3. Usando la función `es_primo`, escriba la función `i_esimo_primo(i)` que entregue el `i`-ésimo número primo:

---

```
int i_esimo_primo(int a){ ... }  
...  
cout << i_esimo_primo(1);        // imprime 2  
cout << i_esimo_primo(20);       // imprime 71
```

---

# Ejercicios

Un analista financiero lleva un registro del precio del dólar día a día, y desea saber cuál fue la mayor de las alzas en el precio diario a lo largo de ese período. Escriba un programa que pida al usuario ingresar el número  $n$  de días, y luego el precio del dólar para cada uno de los  $n$  días.

El programa debe entregar como salida cuál fue la mayor de las alzas de un día para el otro. Si en ningún día el precio subió, la salida debe decir: No hubo alzas.

---

```
Cuantos dias? 10
Dia 1: 496.96
Dia 2: 499.03
Dia 3: 496.03
Dia 4: 493.27
Dia 5: 488.82
Dia 6: 492.16
Dia 7: 490.32
Dia 8: 490.67
Dia 9: 490.89
Dia 10: 494.10
La mayor alza fue de 3.34 pesos
```

---

# Arreglos



# Arreglos

- Podemos usar arreglos para guardar series de datos. Por ejemplo:

---

```
int v[10];
```

---

crea 10 variables de tipo `int` en memoria:

0	1	2	3	4	5	6	7	8	9

- Podemos asignarle valores a estas variables:

---

```
int v[10]={41,23,87,19,-9};
```

---

quedando:

41	23	87	19	-9					
0	1	2	3	4	5	6	7	8	9

# Arreglos

- Podemos definir todos los valores del arreglo al mismo tiempo:

```
int v[]={41,23,87,19,-9,91,103,25,11,22};
```

quedando:

41	23	87	19	-9	91	103	25	11	22
0	1	2	3	4	5	6	7	8	9

- Podemos acceder a un valor del arreglo usando su índice ...

```
cout << v[3];    // imprime 19
```

- ...y podemos cambiar un valor del arreglo usando su índice:

```
int a = 3, b = 2;  
v[3] = a * b;
```

quedando:

41	23	87	<b>6</b>	-9	91	103	25	11	22
0	1	2	3	4	5	6	7	8	9

# Ejemplo

¿Qué hace el siguiente programa?

---

```
#include <iostream>
using namespace std;

int main(){

    int notas[3];
    for (int i = 0; i < 3; i++){
        cout << "Ingrese la nota del certamen " << i + 1 << ": ";
        cin >> notas[i];
    }

    float suma = 0;
    for (int i = 0; i < 3; i++){
        suma += notas[i];
    }

    cout << "Su promedio es: " << suma / 3 << endl;
    return 0;
}
```

---

# Ejemplo

¿Qué hace el siguiente programa?

---

```
#include <iostream>
using namespace std;

int main(){

    int notas[3];
    for (int i = 0; i < 3; i++){
        cout << "Ingrese la nota del certamen " << i + 1 << ": ";
        cin >> notas[i];
    }

    float suma = 0;
    for (int i = 0; i < 3; i++){
        suma += notas[i];
    }

    cout << "Su promedio es: " << suma / 3 << endl;
    return 0;
}
```

---

Calcula el promedio de las tres notas ingresados por el usuario

# Matrices

Podemos definir matrices, o arreglos bidimensionales:

---

```
int x[5][3]={33,21,47},  
          {82,91,95},  
          {50,72,45},  
          {36,79,63},  
          {53,60,74}};
```

---

resulta en:

	0	1	2
0	33	21	47
1	82	91	95
2	50	72	45
3	36	79	63
4	53	60	74

y la siguiente linea

---

```
cout << x[3][2];
```

---

imprime 63

## Ejercicio

El producto interno de dos arreglos de números es la suma de los productos de los términos correspondientes de ambas.

Por ejemplo, si:

---

```
int a[3] = {5, 1, 6};  
int b[3] = {1, -2, 8};
```

---

entonces el producto interno entre a y b es  $(5 * 1) + (1 * -2) + (6 * 8)$ .

1. Escriba la función `producto_interno(int a[], int b[], int n)` que entregue el producto interno de a y b, donde n representa el largo de estos dos arreglos.
2. Dos arreglos de números son **ortogonales** si su producto interno es cero. Escriba la función `son_ortogonales(int a[], int b[], int n)` que indique si a y b son ortogonales:

---

```
int a[2] = {2, 1};  
int b[2] = {-3, 6};  
cout << son_ortogonales(a, n);    // imprime 1 (representa true)
```

---

# El juego del gato

Usemos una matriz para implementar el juego del gato:

---

```
#include <iostream>
using namespace std;

int main()
{
    int i;
    int j;
    char gato[3][3];

    // inicializacion del tablero, inicialmente no hay jugadas
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 3; j++){
            gato[i][j] = ' ';
        }
    }
    ...
}
```

---

# El juego del gato

Deben programar la siguiente interfaz de usuario:

---

```
user@pc:~/Dropbox/Olimpiada/taller_2017/codigo$ ./gato
```

```
  |  |      1,1 | 1,2 | 1,3
-----
  |  |      2,1 | 2,2 | 2,3
-----
  |  |      3,1 | 3,2 | 3,3
```

Jugador 0, donde quiere jugar? (coordenadas para jugar, 0 0 para salir): 1 1

```
0 |  |      1,1 | 1,2 | 1,3
-----
  |  |      2,1 | 2,2 | 2,3
-----
  |  |      3,1 | 3,2 | 3,3
```

Jugador X, donde quiere jugar? (coordenadas para jugar, 0 0 para salir): 2 2

```
0 |  |      1,1 | 1,2 | 1,3
-----
  | X |      2,1 | 2,2 | 2,3
-----
  |  |      3,1 | 3,2 | 3,3
```

...

---