

Este documento contiene las pizarras y ejemplos de código que se utilizaron en las clases de Programación I del grupo G1M3, durante la segunda mitad el mes de octubre de 2025.

1 . Clase 17 octubre 2025

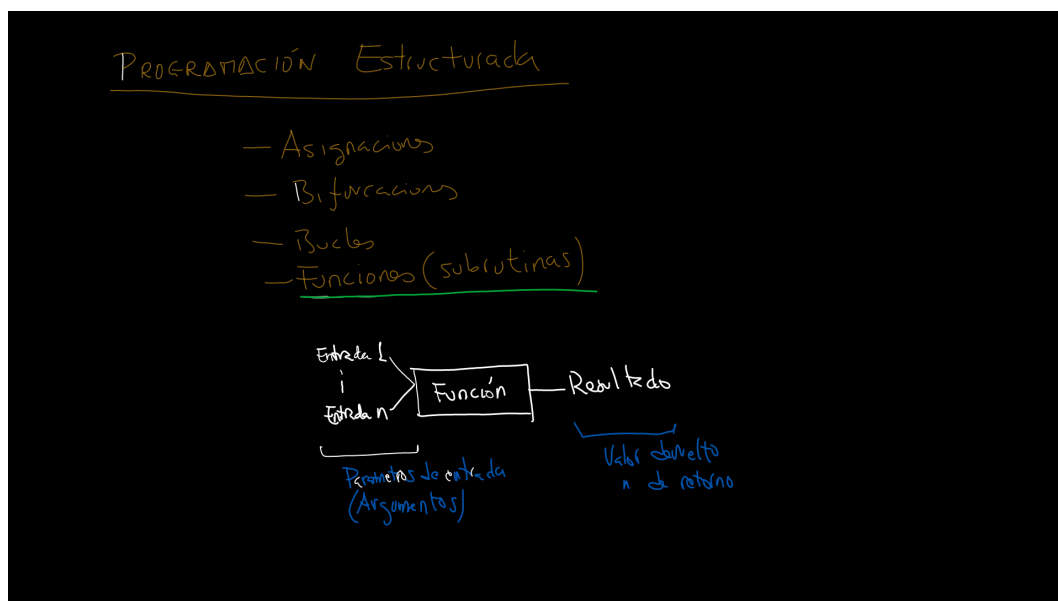


Figura 1: Las funciones son un elemento fundamental de la programación estructurada

```

<tipo_devuelto>  nombre (list de parámetros) {
    //
    //
    //
}

int suma (int n1, int n2) {           (signatura línea defín.)
    int result = n1 + n2;
    return result;
}
    
```

Figura 2: Definición de una función. Ejemplo de la función suma

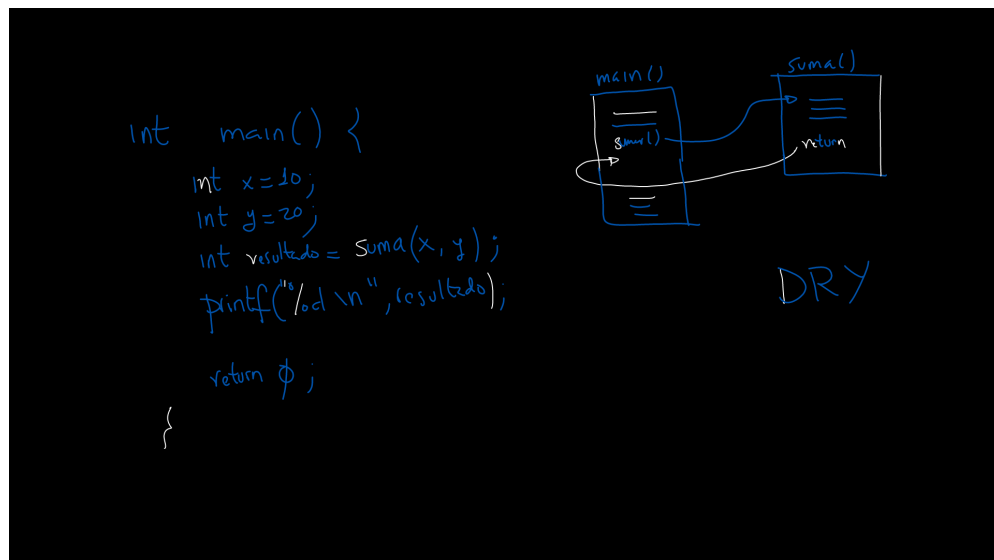


Figura 3: Cuando se llama a una función, el flujo del programa pasa a la función

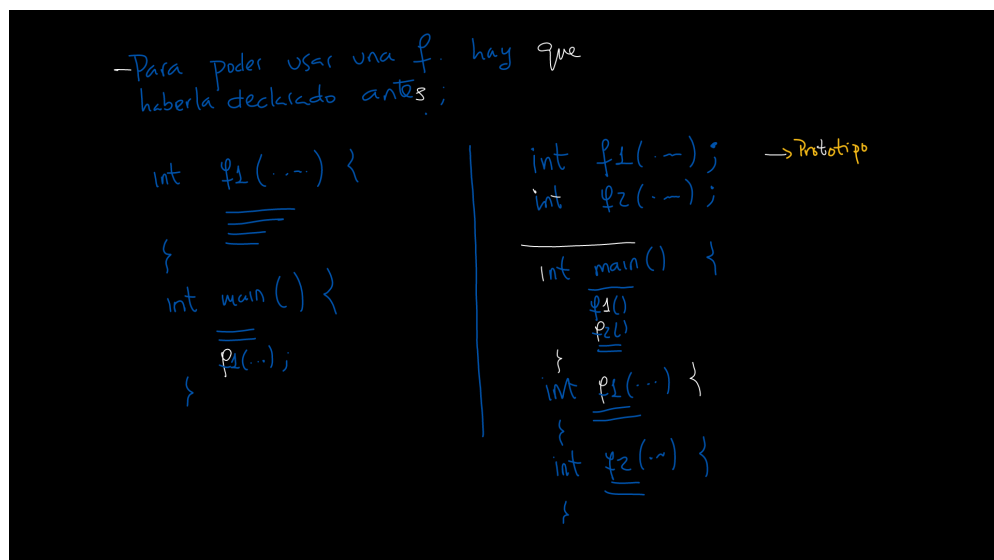


Figura 4: Izquierda: definición de las funciones antes de main(). Derecha: declaración mediante los prototipos antes de main(), poniendo el código completo después de main()

Ejemplo 1 Definición del código de la función antes de main()

```

#include <stdio.h>

double sumaf(double x1, double x2) {
    double result = x1 + x2;
    return result;
}

int main() {
    double x, y;
    printf("Teclea 2 números:");
    scanf("%lf", &x);
    scanf("%lf", &y); // scanf("%lf %lf", &x, &y);

    double result = sumaf(x, y);

    printf("Suma: %.2f", result);

    return 0;
}

```

Ejemplo 2 Prototipo antes de main() y definición, después de main()

```
#include <stdio.h>

double sumaf(double x1, double x2);
void saludo();

int main() {

    saludo();

    double x, y;
    printf("Teclea 2 números:");
    scanf("%lf", &x);
    scanf("%lf", &y); // scanf("%lf %lf", &x, &y);

    double result = sumaf(x, y);
    printf("Suma: %.2f", result);

    return 0;
}

void saludo() {
    printf("Hola\n");
}

double sumaf(double x1, double x2) {
    double result = x1 + x2;
    return result;
}
```

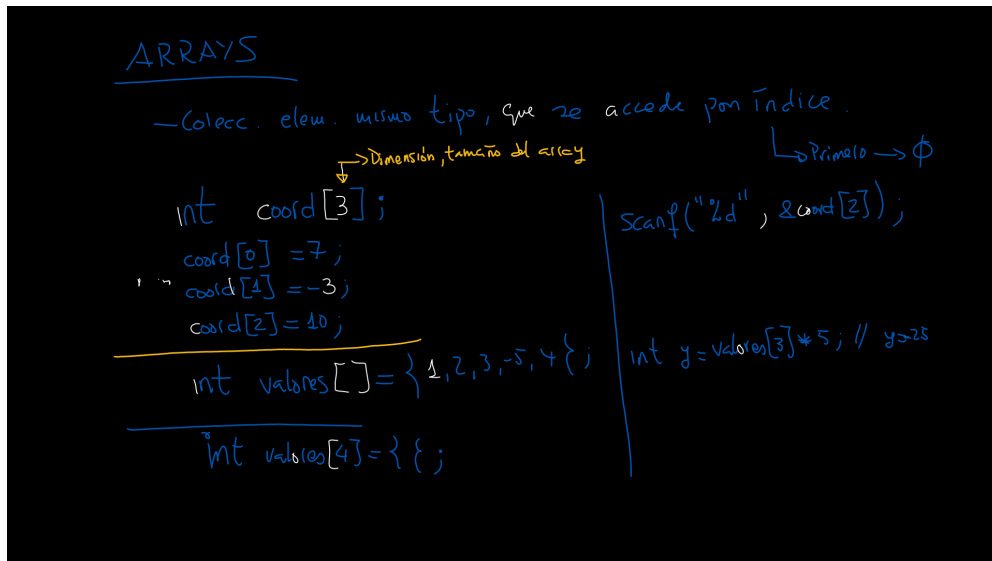


Figura 5: Conceptos sobre arrays: declaración, inicialización y acceso a los elementos individuales

Ejemplo 3 Declaración e inicialización de un array

```
#include <stdio.h>

int main() {

    double x[] = {3.14, 2.1e-3, .45};

    for(int i=0; i<3; i++) {
        printf("%.9f ", x[i]);
    }
    printf("\n");

    return 0;
}
```

TIPOS DE DATOS

- PRIMITIVOS (Básicos): int, double, bool, char
- COMPUESTOS:
 - ARRAYS: colección elem. del mismo tipo, guardados en posiciones consecutivas de memoria y que se accede por índice.
 - Estructuras: colección elementos que no tienen por qué ser del mismo tipo.

Figura 6: Tipos de datos: básicos (primitivos) y compuestos

ARRAYS

```
int v[3];
v[0] = 7;
v[1] = -2;
v[2] = 12;
printf("%d %d %d\n",
       v[0], v[1], v[2]);
v[3] = 25; → Puede dar error
```

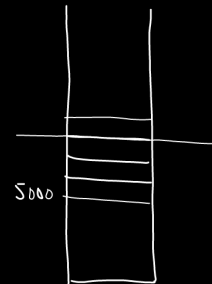


Figura 7: Acceso a los elementos individuales de un array. ¡Ojo, C no comprueba si el índice está dentro de los límites!

```
int v[] = {3, 2, -1, 0};  
int w[4] = { }; // Inicializa con ceros  
  
int p[3];  
p = {1, 2, 3}; ERROR!
```

Figura 8: No se puede inicializar dos veces un array

Ejemplo 4 Otro ejemplo de inicialización de array y acceso a los elementos

```
#include <stdio.h>  
#include <math.h>  
  
int main() {  
  
    double x[] = {3.14, -2.1e-3, -.45, 6.8, 100.0};  
  
    for(int i=0; i<5; i++) {  
        if(x[i] > 0.0) {  
            printf("%.8f ", sqrt(x[i]));  
        } else {  
            printf("%.8f ", x[i]*x[i]);  
        }  
    }  
    printf("\n");  
  
    return 0;  
}
```

ESTRUCTURAS

```

typedef struct {
    char id;
    int x;
    int y;
} Punto2D;

```

Campos (Propiedades)

Nombre del tipo

```

Punto2D p1;
p1.id = 'A';
p1.x = 17;
p1.y = -3;

Punto2D p2;
p2.id = 'C';
p2.x = 10;
p2.y = 12;

Punto2D p3 = {'F', 32, 65};

```

Figura 9: Definición de estructuras

```

typedef struct {
    int x, y;
} Punto;

```

Punto p;

~~p[0] = 7;~~ p.x = 7;

~~p[1] = -3;~~ p.y = -3;

size_t

```

if (x > 3)
    // no se que
else
    // no se como

```

```

if (x > 3) {
} else {
}

```

Figura 10: Izquierda: ejemplo estructura Punto. Derecha: en bifurcaciones y bucles, si dentro del bloque de código solo hay una línea, se pueden omitir las llaves

Ejemplo 5 Ejemplo de estructura

```

#include <stdio.h>

int main() {
    typedef struct {
        double x, y;
    } Punto2D;

    Punto2D p1;
    p1.x = 10;
    p1.y = 20;

    printf("%.4f %.4f\n", p1.x, p1.y);

    Punto2D p2;
    printf("x= ");
    scanf("%lf", &p2.x);
    printf("y= ");
    scanf("%lf", &p2.y);

    printf("%.1f %.1f\n", p2.x, p2.y);
    return 0;
}

```

3 . Comentarios sobre el Apartado 4 de la Práctica 2

- Hay que ir desarrollando el programa con avances pequeños. En cada avance, el programa debe ejecutarse sin errores. Si tras un avance, aparece un error al compilar, es probable que provenga del último código que se ha añadido.
- En la entrada de datos, tras conseguir hacer el bucle de entrada y que funcione, podemos convertirlo en un comentario y usar un array *mock data* para probar el programa.
- El menú, podemos sustituirlo por una simple palabra *Menú* durante las pruebas.
- Usamos un esqueleto del menú (*switch-case*) y probamos su funcionamiento: opción de salida, opciones erróneas.
- Cada opción se puede probar en un programa independiente, hasta que funcione. Cuando ya funcione, se puede hacer *copy-paste* dentro del bloque *case* de la opción correspondiente.

Ejemplo 6 Esqueleto de trabajo del Apartado 4 de la Práctica 2

```
#include <stdio.h>

#define DIM 10
int main() {

    // Entrada de datos
    /*
    int nums[DIM];
    for(int i=0; i<DIM; i++) {
        printf("nums[%d]= ", i);
        scanf("%d", &nums[i]);
    }
    */

    int nums[]={1,2,3,4,5,6,7,8,9,10};

    int opcion;
    do {
        printf("Menú\n"); // Menú mock para pruebas

        printf("\nSu opción:");
        scanf("%d", &opcion);

        switch(opcion) {
            case 1:
                printf("Opción 1\n");
                break;
            case 6:
                printf("Salir\n");
                break;
            default:
                printf("Opción no válida\n");
                break;
        }
        printf("Ha elegido: %d\n", opcion);
    } while(opcion!=6);

    printf("FIN\n");
    return 0;
}
```

Ejemplo 7 Programa independiente para probar opciones

```
#include <stdio.h>

int main() {
    int nums[]={1,2,3,4,5,6,7,8,9,10};

    // Pruebas
}
```