Este documento contiene las pizarras y ejemplos de código que se utilizaron en las clases de Programación I del grupo G1M3, el día 29 de octubre de 2025.

1 . Clase 29 octubre 2025

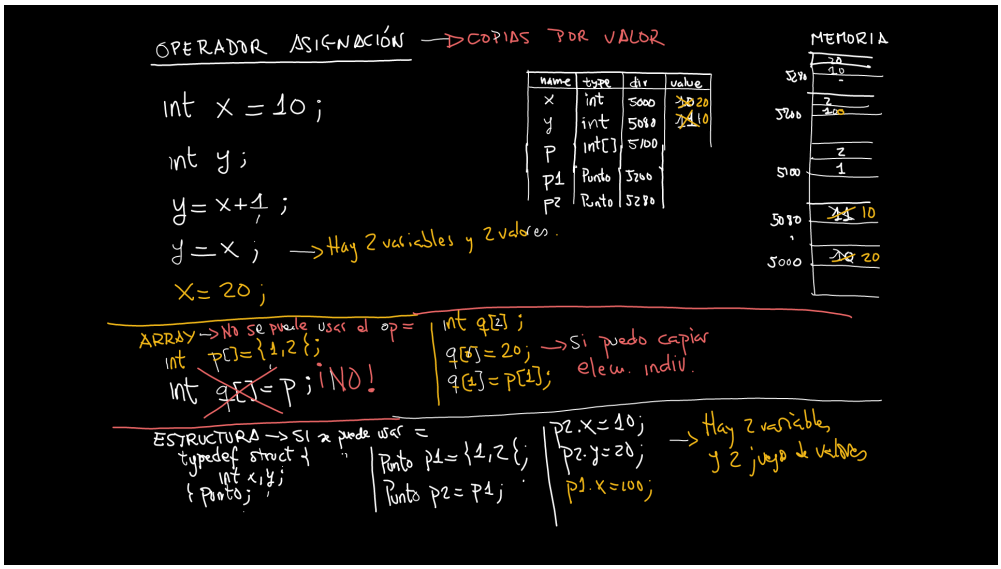

Figura 1: Operador de asignación con tipos primitivos, arrays o estructuras: con arrays no se puede usar, con tipos primitivos y estructuras, sí

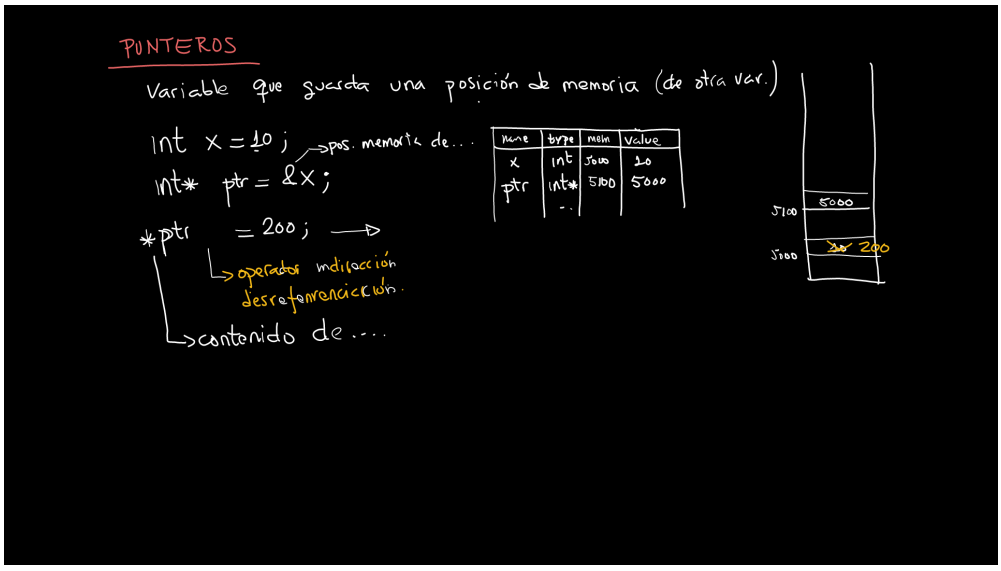

Figura 2: Punteros

Ejemplo 1 Ejemplo básico sobre punteros

```c
#include <stdio.h>

int main() {

   int x = 10;
   int y = x;  // Copia por valor
   y = 20;     // Hay dos variables y dos valores independientes en memoria

   int* ptr = &x;  // ptr apunta a x. Hay dos variables usando un solo valor en memoria
   *ptr = 255;     // El contenido de ptr es el valor de x

   int* ptr2 = &x; // ptr2 apunta a x. Hay tres variables usando el mismo valor en memoria
   *ptr2 = 127;    // el contenido de ptr2 es el valor de x

   // Imprimir x desde x, ptr o ptr2
   printf("%d\n", x);
   printf("%d\n", *ptr);
   printf("%d\n", *ptr2);

   // Imprimir punteros en formato puntero
   printf("%p\n", ptr);
   printf("%p %p\n", &x, ptr);
   printf("%p %p\n", &x, ptr2);

   return 0;
}
```

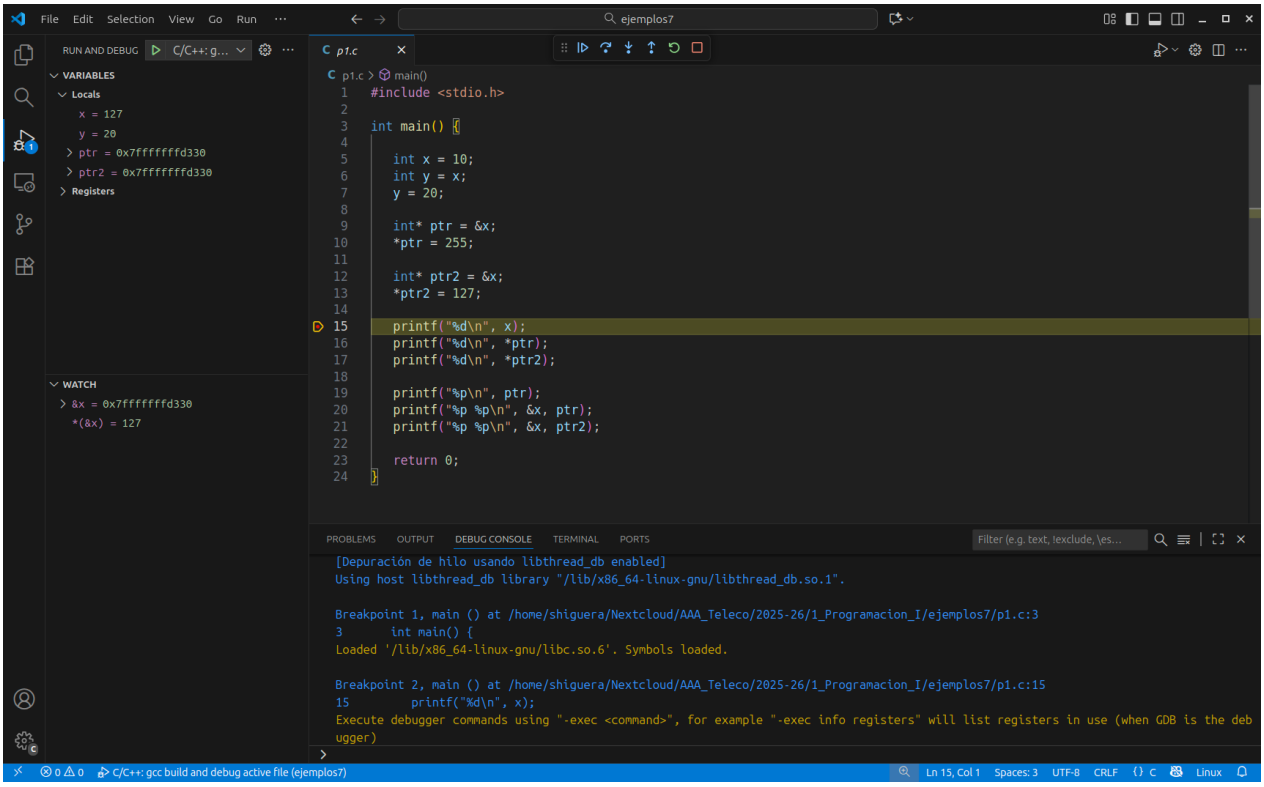

Figura 3: Salida del depurador para el Ejemplo 1. Observa la utilización de la ventana de inspección (*watch*) para comprobar el resultado de expresiones que puedan interesar y no aparecen en la ventana de variables locales

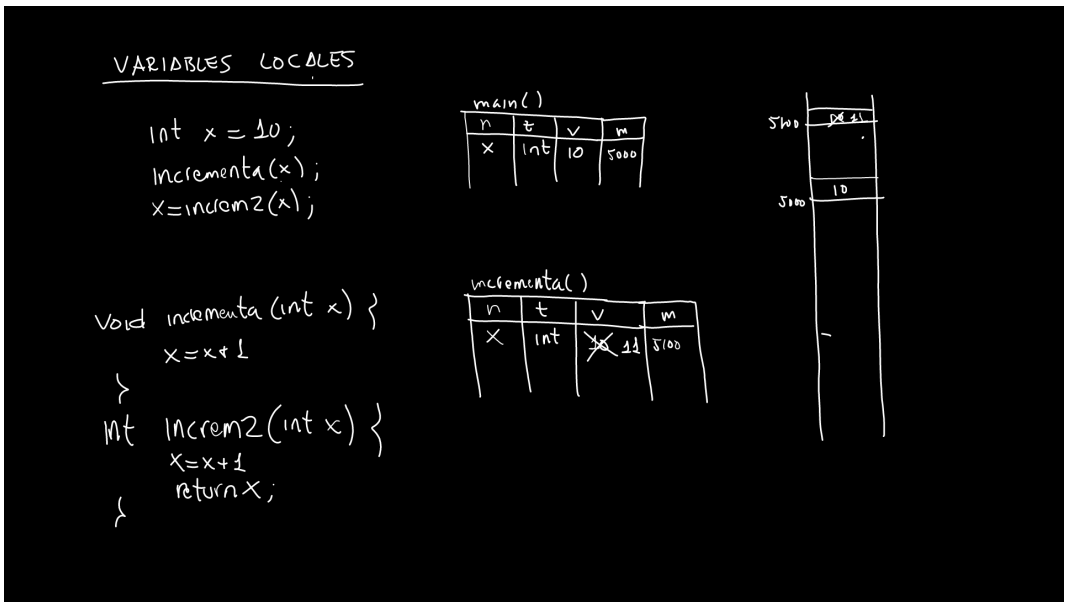

Figura 4: Variables locales. Cada función, tiene su propia tabla de variables. Si una variable se declara dentro de una función, solo es visible dentro de esa función. Aunque haya dos variables que se llamen igual en dos funciones distintas, las variables no son las mismas. La función *increm2()* devuelve el valor modificado, y serviría para modificar el valor de la variable *x* de la función *main()*

Ejemplo 2 Ejemplo para observar el funcionamiento de las variables locales

```c
#include <stdio.h>

void incrementa(int x);
int increm2(int x);

int main() {
   int x = 10;
   printf("Antes de incrementa(): %d\n", x);   // La x vale 10
   incrementa(x); // Con esta llamada, lo que llega a la función incrementa() es un 10
   printf("Después de incrementa(): %d\n", x); // La x sigue valiendo 10

   x = increm2(x);
   printf("%d\n", x); // Ahora la x sí que vale 11

   return 0;
}

void incrementa(int x) {
   // A través del parámetro, lo que llega a la función es un valor, no una variable
   // La variable x de la función incrementa() es local de la función
   // No es la misma ni tiene nada que ver con la variable x de la función main()
   x = x+1;
   printf("Dentro: %d\n", x);
}
int increm2(int x) {
   return x+1;
}
```

Ejemplo 3 Ejemplo de variables globales

```c
#include <stdio.h>

#define PI 3.1416

// Variables globales
int x = 20;
const double pi = 3.1416;

void incrementa() {
   // Esta función modifica la variable global
   x = x + 1;
   printf("Dentro: %d\n", x);
}

int main() {
   printf("Antes: %d\n", x);   // x es global
   incrementa();
   printf("Después: %d\n", x);

   printf("%.2f\n", 2*pi*10.0); // Uso de la constante global pi
   printf("%.2f\n", 2*PI*10.0); // Uso de la pseudo constante PI

   return 0;
}
```

Ejemplo 4 Ejemplo de función con parámetros de tipo *struct*

```c
#include <stdio.h>
#include <math.h>

// Definiciones de tipos
typedef struct {
   double x,y;
} Punto;

// Prototipos de funciones
double dist(Punto p1, Punto p2);

// Programa principal
int main() {
   Punto q1 ;
   q1.x = 0.0; q1.y = 0.0;
   Punto q2 = {1.0, 1.0};
   double d = dist(q1, q2);
   printf("%.2f\n", d);
   return 0;
}

// Código de las funciones
double dist(Punto p1, Punto p2) {
   // Esta función recibe como argumentos valores de tipo estructura
   // y devuelve un valor double
   double incx = p2.x - p1.x;
   double incy = p2.y - p1.y;
   double d = sqrt(incx*incx + incy*incy);
   return d;
}
```

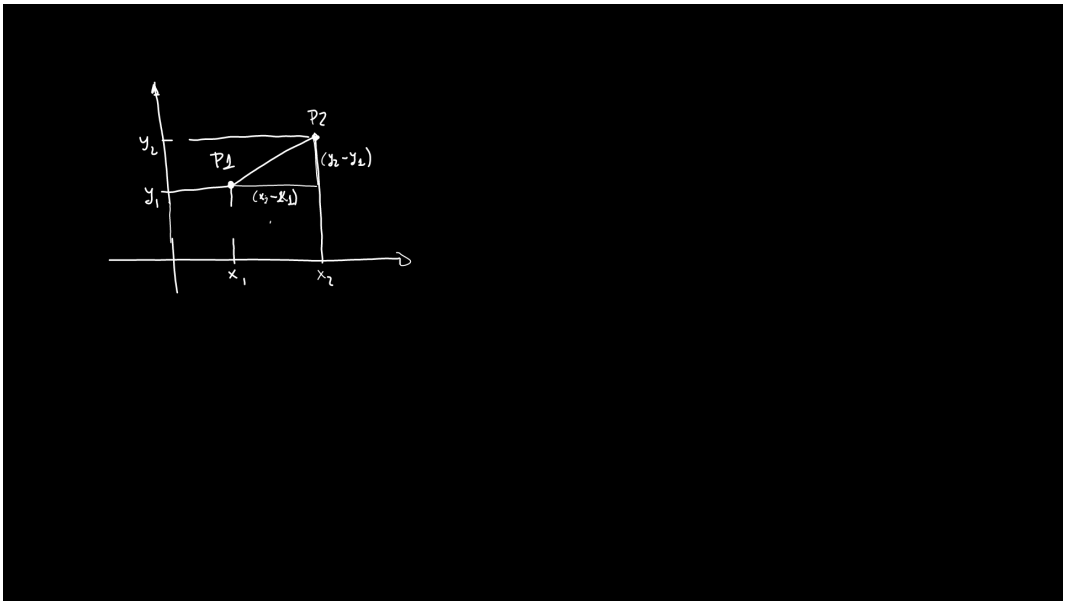

Figura 5: Distancia cartesiana entre dos puntos del plano