

UNIVERSIDAD SAN PABLO DE GUATEMALA

Facultad de Ciencias Empresariales

Escuela de Ingeniería

Ingeniería en Sistemas Ciencias y de la Computación



Examen Final Compiladores

Tarea presentada en el curso de Compiladores

Impartido por: MA. Ing. Walter Oswaldo Ordóñez Sierra

Yordy Estiven Rodríguez Morales - 2300321

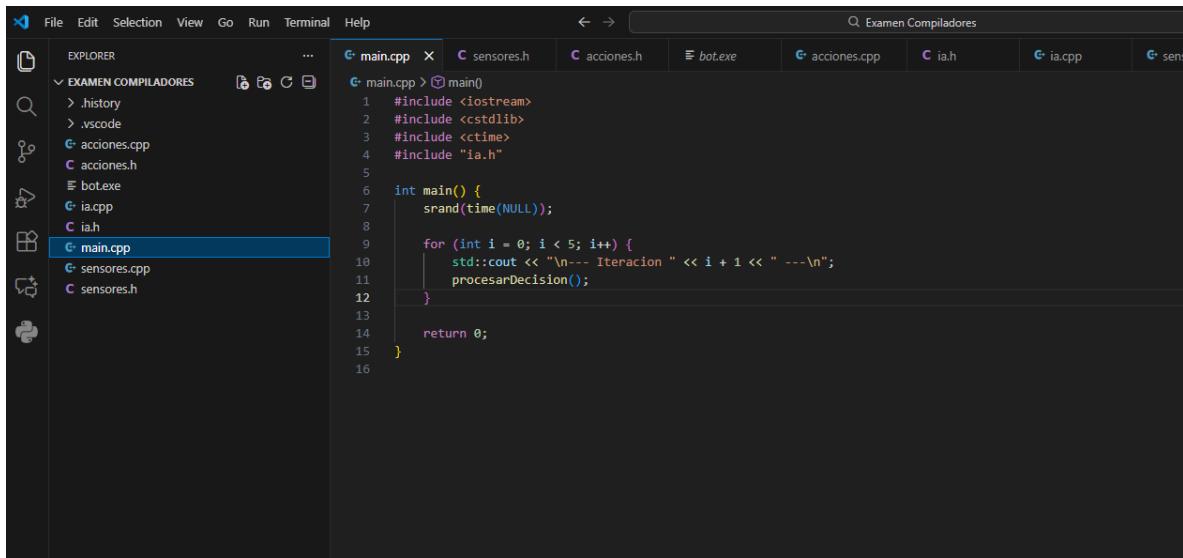
Guatemala, 29 de noviembre 2025

Explicación de cada archivo del proyecto

main.cpp

En este archivo está el punto de inicio del programa.

Aquí simplemente se configura el uso de números aleatorios para simular los sensores y se ejecuta un ciclo que corre varias iteraciones. En cada vuelta del ciclo se llama a la función que toma las decisiones del robot. Este archivo no hace nada de lógica compleja, solo organiza el orden en el que todo se ejecuta.



```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "ia.h"

int main() {
    srand(time(NULL));

    for (int i = 0; i < 5; i++) {
        std::cout << "\n--- Iteracion " << i + 1 << " ---\n";
        procesarDecision();
    }

    return 0;
}
```

sensores.cpp

Este archivo contiene todas las funciones que simulan los sensores del robot.

Cada función devuelve un valor aleatorio que representa si el sensor detectó o no un riesgo. No es una lectura real, solo una simulación para poder probar el resto del programa.

También está la función `detectarRiesgo()`, que revisa cada “sensor” en un orden específico. Si uno de los sensores activa un riesgo, la función devuelve un código que identifica ese peligro. Si ninguno detecta nada, devuelve un valor que indica que está todo normal. Esta parte es la base para que la IA sepa qué decisión tomar después.

```
#include <iostream>
#include <cstdlib>
#include "sensores.h"

class Sensores {
public:
    bool hayPersonaFrente() { return rand() % 10 < 2; }
    bool hayVehiculoDelante() { return rand() % 10 < 2; }
    bool vehiculoAtras() { return rand() % 10 < 2; }
    bool obstaculoLateralIzq() { return rand() % 10 < 2; }
    bool obstaculoLateralDer() { return rand() % 10 < 2; }
    bool terrenoResbaloso() { return rand() % 10 < 2; }
    bool zonaProhibida() { return rand() % 10 < 2; }
    bool objetoPequeno() { return rand() % 10 < 2; }

    int detectarRiesgo() {
        if (hayPersonaFrente()) return PERSONA_FRENT;
        if (hayVehiculoDelante()) return VEHICULO_ADELANTE;
        if (vehiculoAtras()) return VEHICULO_ATRAS;
        if (obstaculoLateralIzq()) return OBSTACULO_IQ;
        if (obstaculoLateralDer()) return OBSTACULO_DER;
        if (terrenoResbaloso()) return TERRENO_RESP;
        if (zonaProhibida()) return ZONA_PROHIBIDA_R;
        if (objetoPequeno()) return OBJETO_PEQ;

        return NINGUNO;
    }
};
```

acciones.cpp

Aquí están las funciones que representan lo que el robot puede hacer. Cada acción solo imprime un mensaje indicando lo que ejecutaría el robot en ese caso: frenar, girar, retroceder, reducir velocidad, avanzar, etc. No controla motores ni nada así; simplemente muestra las acciones que corresponderían según el riesgo que se detectó.

```
#include <iostream>
#include "acciones.h"

class Acciones {
public:
    void frenar() { std::cout << "Acción: Frenando inmediatamente.\n"; }
    void activarAlerta() { std::cout << "Acción: Activando alerta sonora.\n"; }
    void girarIzquierda() { std::cout << "Acción: Girando a la izquierda.\n"; }
    void girarDerecha() { std::cout << "Acción: Girando a la derecha.\n"; }
    void retroceder() { std::cout << "Acción: Retrocediendo con cuidado.\n"; }
    void reducirVelocidad() { std::cout << "Acción: Reduciendo velocidad por tracción baja.\n"; }
    void buscarRutaAlterna() { std::cout << "Acción: Buscando ruta alterna.\n"; }
    void avanzar() { std::cout << "Acción: Avanzando normalmente.\n"; }
};
```

ia.cpp

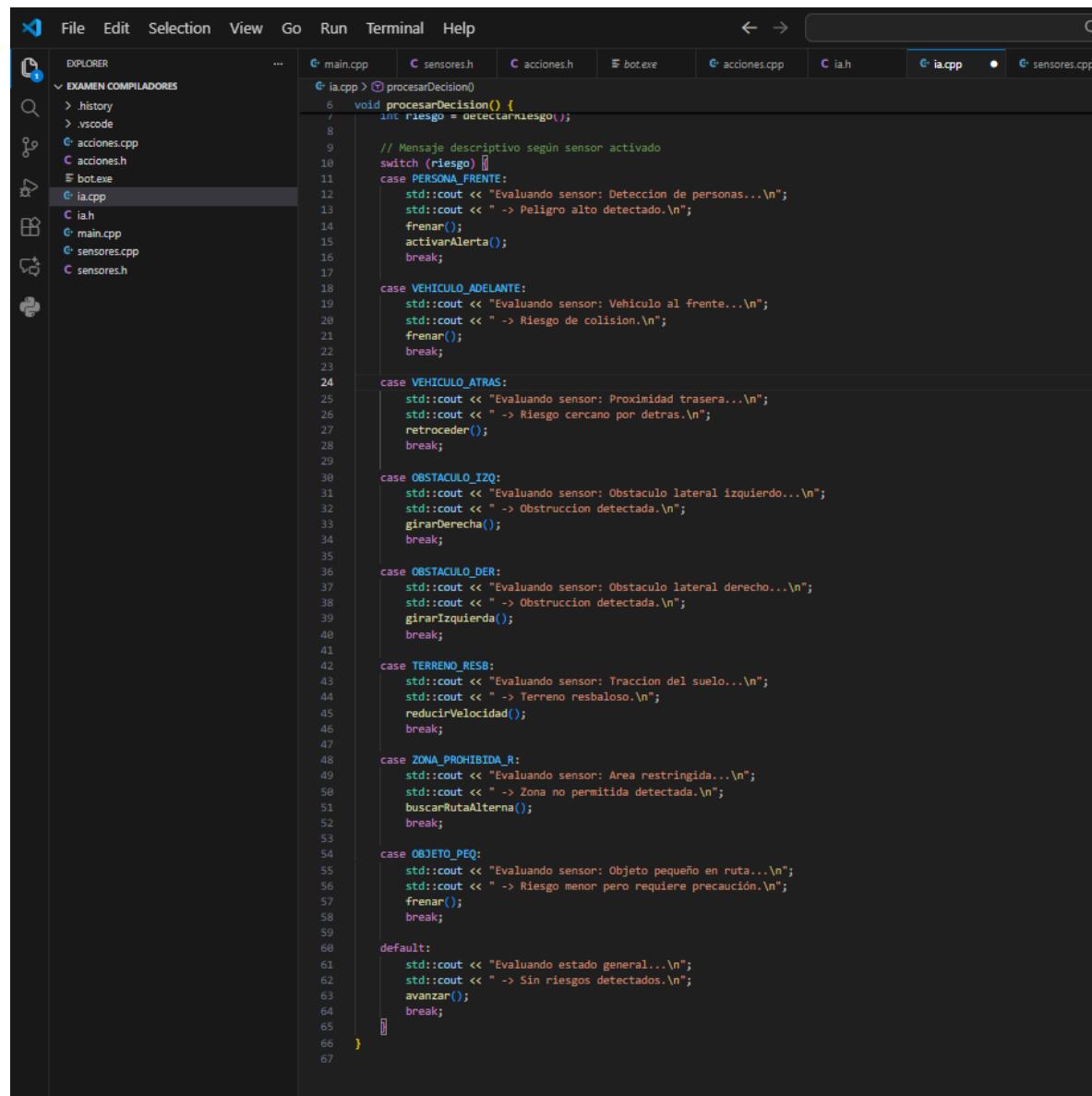
En este archivo está la lógica que decide qué hacer dependiendo del riesgo que detectan los sensores.

Lo primero que hace es llamar a la función `detectarRiesgo()`. Luego compara el resultado con una lista de casos posibles.

Cada caso activa una o varias acciones diferentes. Si el riesgo es una persona, frena y enciende la alerta. Si es un vehículo adelante, frena. Si el peligro está atrás, retrocede, y así con los demás.

Si no hay ningún riesgo, hace que el robot avance con normalidad.

Este archivo es donde se unen los sensores con las acciones.



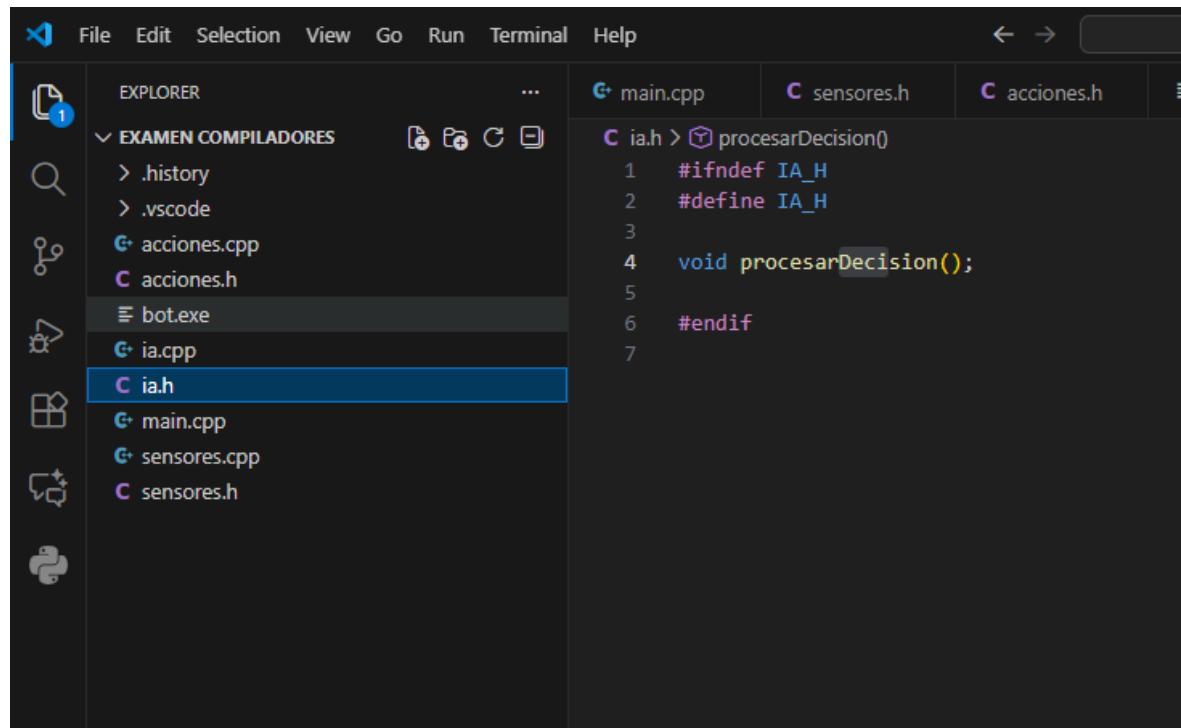
```
File Edit Selection View Go Run Terminal Help
EXPLORER main.cpp sensores.h acciones.h bot.exe acciones.cpp ia.h ia.cpp sensores.cpp
ia.cpp > procesarDecision()
6 void procesarDecision() {
7     int riesgo = detectarRiesgo();
8
9     // Mensaje descriptivo según sensor activado
10    switch (riesgo) {
11        case PERSONA_FRENT:
12            std::cout << "Evaluando sensor: Detecction de personas...\n";
13            std::cout << " -> Peligro alto detectado.\n";
14            frenar();
15            activarAlerta();
16            break;
17
18        case VEHICULO_ADELANTE:
19            std::cout << "Evaluando sensor: Vehiculo al frente...\n";
20            std::cout << " -> Riesgo de colision.\n";
21            frenar();
22            break;
23
24        case VEHICULO_ATRAS:
25            std::cout << "Evaluando sensor: Proximidad trasera...\n";
26            std::cout << " -> Riesgo cercano por detras.\n";
27            retroceder();
28            break;
29
30        case OBSTACULO_IZQ:
31            std::cout << "Evaluando sensor: Obstaculo lateral izquierdo...\n";
32            std::cout << " -> Obstruccion detectada.\n";
33            girarDerecha();
34            break;
35
36        case OBSTACULO_DER:
37            std::cout << "Evaluando sensor: Obstaculo lateral derecho...\n";
38            std::cout << " -> Obstruccion detectada.\n";
39            girarIzquierda();
40            break;
41
42        case TERRENO_RESB:
43            std::cout << "Evaluando sensor: Traccion del suelo...\n";
44            std::cout << " -> Terreno resbaloso.\n";
45            reducirVelocidad();
46            break;
47
48        case ZONA_PROHIBIDA_R:
49            std::cout << "Evaluando sensor: Area restringida...\n";
50            std::cout << " -> Zona no permitida detectada.\n";
51            buscarRutaAlternativa();
52            break;
53
54        case OBSTACULO_PEQ:
55            std::cout << "Evaluando sensor: Objeto pequeño en ruta...\n";
56            std::cout << " -> Riesgo menor pero requiere precaucion.\n";
57            frenar();
58            break;
59
60        default:
61            std::cout << "Evaluando estado general...\n";
62            std::cout << " -> Sin riesgos detectados.\n";
63            avanzar();
64            break;
65    }
66}
```

ia.h, acciones.h y sensores.h

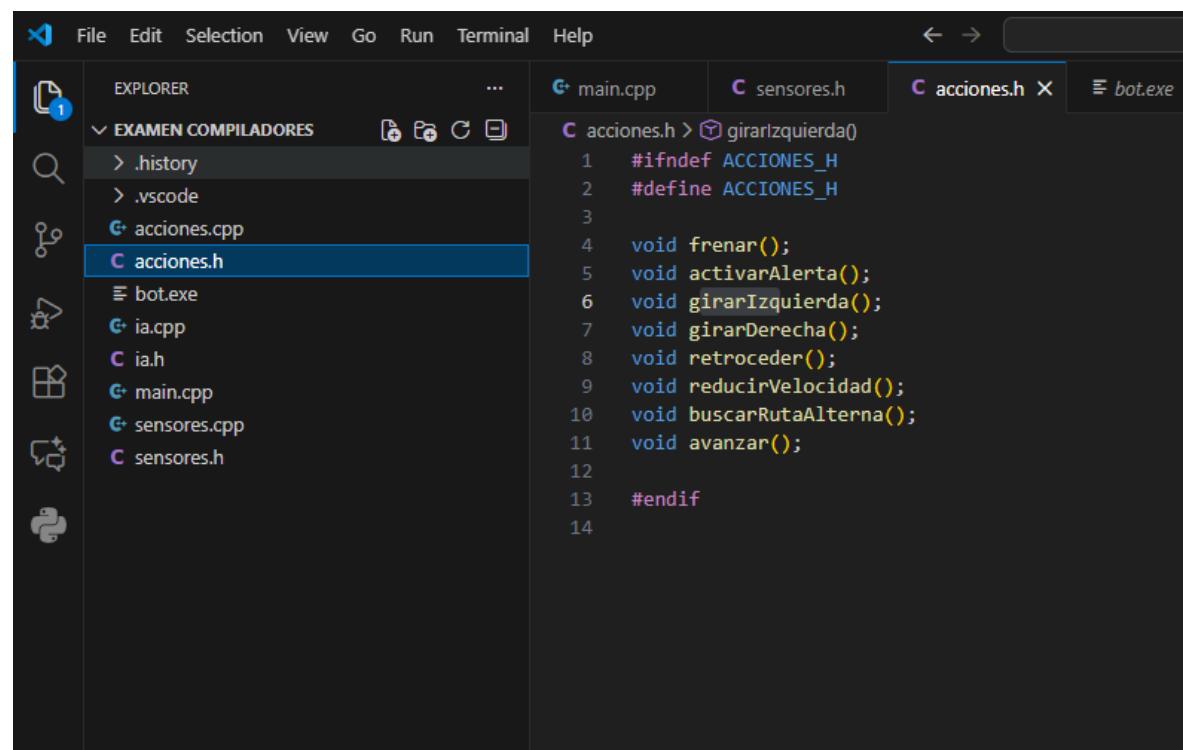
Estos archivos son los encabezados donde solo se declaran las funciones y los valores necesarios para que los otros archivos puedan trabajar juntos.

Sirven para organizar mejor el proyecto y mantener el código más claro.

No contienen lógica interna, solo las definiciones que permiten que cada archivo conozca las funciones y los nombres que necesita.



```
main.cpp    sensores.h    acciones.h  
ia.h  
1 #ifndef IA_H  
2 #define IA_H  
3  
4 void procesarDecision();  
5  
6#endif
```



```
main.cpp    sensores.h    acciones.h  
acciones.h  
1 #ifndef ACCIONES_H  
2 #define ACCIONES_H  
3  
4 void frenar();  
5 void activarAlerta();  
6 void girarIzquierda();  
7 void girarDerecha();  
8 void retroceder();  
9 void reducirVelocidad();  
10 void buscarRutaAlterna();  
11 void avanzar();  
12  
13#endif
```

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows a tree view of the project structure under "EXAMEN COMPILADORES". Files listed include ".history", ".vscode", "acciones.cpp", "acciones.h", "bot.exe", "ia.cpp", "ia.h", "main.cpp", "sensores.cpp", and "sensores.h".
- Editor:** The file "sensores.h" is open and selected. The code defines an enum for risks and various sensor detection functions.
- Terminal:** The terminal tab is visible at the top, showing navigation arrows and a search bar.

```
1 #ifndef SENSORES_H
2 #define SENSORES_H
3
4 // Identificadores de riesgos
5 enum Riesgos {
6     NINGUNO = 0,
7     PERSONA_FRENTRE,
8     VEHICULO_ADELANTE,
9     VEHICULO_ATRAS,
10    OBSTACULO_IZO,
11    OBSTACULO_DER,
12    TERRENO_RESB,
13    ZONA_PROHIBIDA_R,
14    OBJETO_PEQ
15 };
16
17 // Declaración de sensores
18 bool hayPersonaFrente();
19 bool hayVehiculoDelante();
20 bool vehiculoAtras();
21 bool obstaculoLateralIzq();
22 bool obstaculoLateralDer();
23 bool terrenoResbaloso();
24 bool zonaProhibida();
25 bool objetoPequeno();
26
27 // Función que decide qué riesgo hay
28 int detectarRiesgo();
29
30 #endif
31
```