

# Clase #02 de 27

# Introducción a Compiladores y al Trabajo #0

*Abril 1ro, Martes*  
*Abril 9, Miércoles*

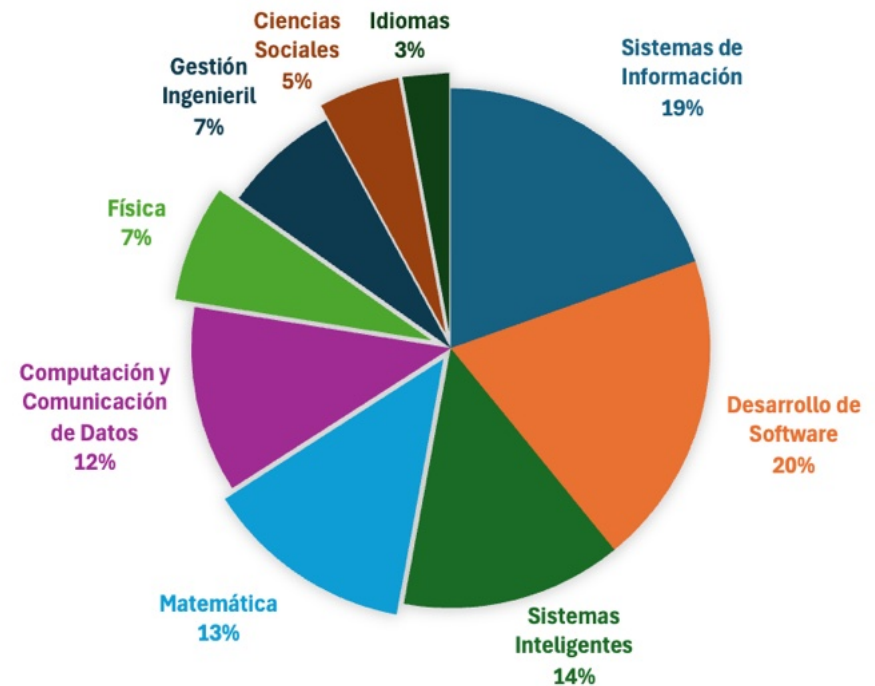
# Agenda para esta clase

- SSL y Relación con Otras Asignaturas
- Primer contacto con el compilador
- Continuación Trabajo #0

# SSL y Relación con Otras Asignaturas




# Incidencia de Cada Área

Área de conocimiento	Horas	%
Sistemas de Información	648	20%
Desarrollo de Software	648	20%
Sistemas Inteligentes	456	14%
Matemática	432	13%
Computación y Comunicación de Datos	384	12%
Física	240	7%
Gestión Ingenieril	240	7%
Ciencias Sociales	168	5%
Idiomas	96	3%
<b>Total</b>	<b>3312</b>	<b>100%</b>



- Departamento Básicas (28%)
  - Matemática
  - Física
  - Ciencias Sociales
  - Idiomas
- Departamento Ingeniería Sistemas de Información (72%)
  - Sistemas de Información
  - Desarrollo de Software
  - Sistemas Inteligentes
  - Computación y Comunicación de Datos
  - Gestión Ingenieril
- No incluye (680 h)
  - Electivas (480 h)
    - Tercer nivel: 96 h
    - Segundo nivel: 144 h
    - Quinto nivel 240 h
  - Práctica Profesional Supervisada: 200 h

# Integración con Otras Asignaturas (SSL)

-  Vertical Predecesoras
  - Algoritmos y Estructuras de Datos
  - Lógica y Estructuras Discretas (*ex Matemática Discreta*)
  - Correlativas recomendadas
    - Arquitectura de Computadoras
    - Sistemas y Procesos de Negocio (*ex Sistemas y Organizaciones*)
-  Horizontal (Paralelas)
  - Paradigmas de Programación
  - Sistemas Operativos
  - Análisis de Sistemas de Información (integradora)
-  Vertical Sucesoras
  - Base de Datos (*ex Gestión de Datos*)
  - Ingeniería y Calidad de Software

# Módulo B

- Conjuntos Numéricos
- Ecuaciones e Inecuaciones
- Funciones
- Vectores

```
// Conjuntos numéricos
unsigned u{21u};
    int i{42};
    double d{3.14};

#include<cassert>
#include<array>
using std::array;

int main(){
    int x{};

    // Ecuaciones
        x = 42      ; // Ecuación?
        x/2 == 21   ; // Ecuación?
    assert( x == 42 ); // Ecuación? Sí
    assert( x != 21 ); // Inecuación
    assert( x <= 42 ); // Inecuación
    assert( x >= 42 ); // Inecuación
    assert( x < 71  ); // Inecuación
    assert( x > 17  ); // Inecuación

    // Funciones
    double f(double); // Prototipo o declaración
    assert( f(1) == 0 ); // Invocación

    // Vectores
    double Norm(array<double,3>);
    array<double,3> v{3,7,5};
    assert( Norm(v) >= 9.11 and Norm(v) <= 9.12);
}

double f(double x){return 2*x+1;} // Definición

double Norm(array<double,3>v){
    return std::sqrt(
        v.at(0) * v.at(0) +
        v.at(1) * v.at(1) +
        v.at(2) * v.at(2)
    );
}
```



# Arquitectura de Computadoras

- Objetivos

- Comprender la representación de datos.
- Comprender las estructuras básicas de un computador y su funcionamiento.
- Distinguir la jerarquía de memoria y arquitecturas de microprocesadores.
- Comprender lenguajes de bajo nivel.
- Analizar los recursos computacionales a ser utilizados en el procesamiento, almacenamiento y comunicación de datos.

- Contenidos mínimos

- Sistemas numéricos de distintas bases.
- Operaciones y Conversiones.
- Circuitos lógicos y digitales, códigos y representaciones.
- Algebra de Boole.
- Tecnologías de almacenamiento y dispositivos de entrada y salida.
- Componentes de la arquitectura interna.
- Plataformas de microprocesadores.
- Programación en lenguajes de bajo nivel.

# Área Desarrollo de Software

- Objetivos
  - Formar e informar acerca de metodologías, técnicas y lenguajes de programación, como herramientas básicas para el desarrollo de software y el estudio de disciplinas que permitan crear nuevas tecnologías
- Asignaturas (648 hs)
  - Lógica y Estructuras Discretas
  - Algoritmos y Estructuras de Datos
  - Sintaxis y Semántica de los Lenguajes
  - Paradigmas de Programación
  - Bases de Datos
  - Desarrollo de Software
  - Ingeniería y Calidad de Software



# Lógica y Estructuras Discretas

- Objetivos

- Aplicar métodos inductivos, deductivos y recursivos en resolución de situaciones problemáticas.
- Caracterizar estructuras algebraicas y sus propiedades.
- Emplear la teoría de grafos, dígrafos y árboles en resolución de problemas.

- Contenidos Mínimos

- Lógica Simbólica Proposicional y de Predicados de Primer Orden.
- Inducción Matemática.
- Relaciones.
- Estructuras Algebraicas Finitas.
- Teoría de Grafos.
- Teoría de Conjuntos.
- Análisis Combinatorio.

# Algoritmos y Estructuras de Datos

- Objetivos
  - Identificar problemas algorítmicos.
  - Comprender el proceso de desarrollo de software.
  - Resolver problemas aplicando soluciones algorítmicas y estructuras de datos.
- Contenidos mínimos
  - Programación Imperativa y Concepto de algoritmo.
  - Concepto de Dato.
  - Tipos de Datos Simples.
  - Tipo Abstracto de datos.
- Estructuras de Control Básicas.
- Estrategias de Resolución de problemas.
- Estructuras de Datos.
- Abstracciones con procedimientos y funciones.
- Estructuras de Datos lineales y no lineales.
- Algoritmos de Búsqueda, Recorrido y Ordenamiento.
- Archivos de Acceso Secuencial y Aleatorio.
- Recursividad.

# Repaso de Conceptos de Asignaturas Anteriores

- Ingreso
  - Módulo B
    - Conjuntos Numéricos
    - Ecuaciones e Inecuaciones
    - Funciones
    - Vectores
- Área de Sistemas de Información
  - Sistemas y Procesos de Negocio (no correlativa)
    - Sistema
    - Organización
    - Información
    - Dato
    - Proceso
- Área Computación y Comunicación de Datos
  - Arquitectura de Computadoras
    - Representación de datos.
      - IEEE 754 Floating Point Standard  
<https://josemariasola.wordpress.com/reference/#ieee754>
    - Jerarquía de memoria y arquitecturas de microprocesadores.
    - Lenguajes de bajo nivel de abstracción  
<https://josemariasola.wordpress.com/ssl/reference/#assembler>
    - Sistemas numéricos de distintas bases
- Área de Desarrollo de Software
  - Lógica y Estructuras Discreta (correlativa)
    - Función
    - Lógica
    - Grafos
  - Algoritmos y Estructura de Datos (correlativa)
    - Algoritmo
    - Dato
    - Estructura de Datos
    - Estructuras básicas de control de flujo de ejecución  
<https://josemariasola.wordpress.com/aed/papers/#Iterations>  
<https://josemariasola.wordpress.com/aed/papers/#Sequences>
    - Programa
    - Lenguaje
    - Proceso
    - Procedimiento
    - Función  
<https://josemariasola.wordpress.com/aed/papers/#Functions>
    - Parámetro
    - Argumento
    - Recursividad  
<https://josemariasola.wordpress.com/aed/papers/#Recursion>
    - Layout de la Memoria Principal  
<https://josemariasola.wordpress.com/aed/papers/#MemoryLayout>
    - Heap: Reserva dinámica y manual de memoria  
<https://josemariasola.wordpress.com/aed/papers/#HeapAllocation>
    - Recursividad  
<https://josemariasola.wordpress.com/aed/papers/#Recursion>

# Integración con otras Asignaturas y Conceptos Principales

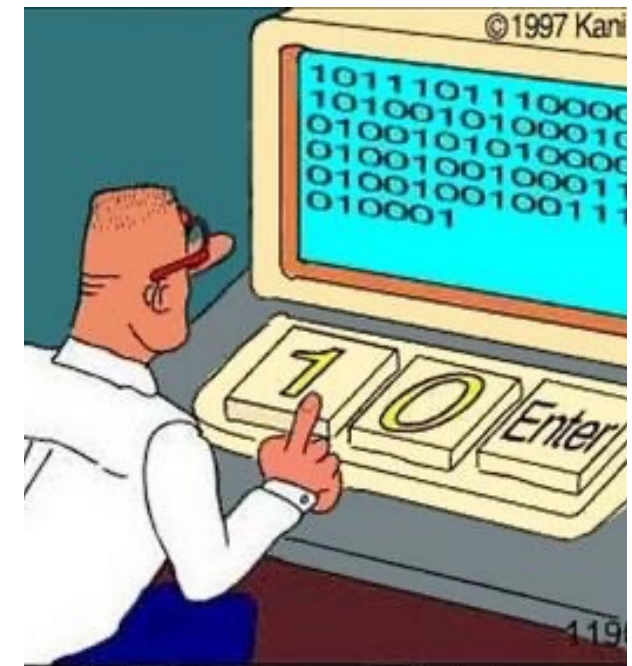
- Integración Vertical: Anteriores
  - Área Desarrollo de Software
    - Lógica y Estructuras Discretas
    - Algoritmos y Estructuras de Datos
- Integración Horizontal: Paralelas
  - Área de Desarrollo de Software
    - Paradigmas de Programación
  - Computación y Comunicación de Datos
    - Sistemas Operativos
  - Sistemas de Información
    - Sistemas y Organización
    - Análisis de Sistemas
    - Diseño de Sistemas
- Integración Vertical: Posteriores
  - Área de Desarrollo de Software
    - Gestión de Datos
    - Ingeniería en Software
- Abstracción -- Concepto fundamental
  - Separación, dejar de lado los detalles para enfocar en lo importante
- Tipo de Dato
  - Conjunto de Valores y conjunto de operaciones sobre ese conjunto de valores
- Función
  - Relación entre conjuntos: Existencia y Unicidad
- Orientación a Objetos
  - Objeto: entidad con comportamiento y que mantiene un estado.

# Primer Contacto con el Compilador

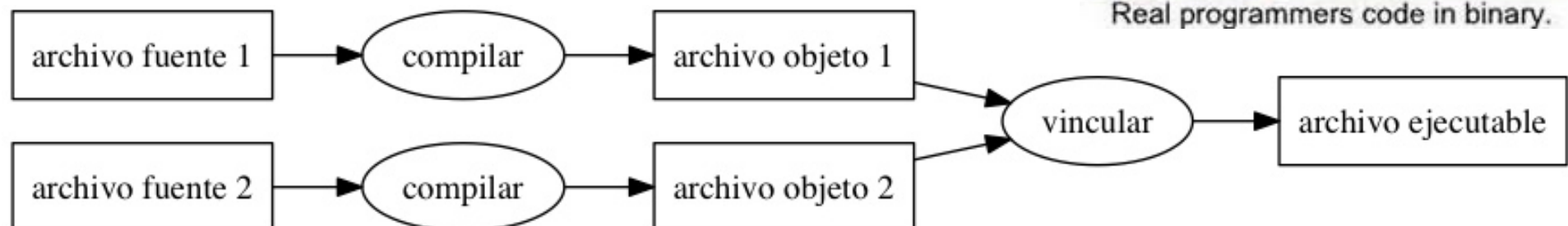
Lenguajes y Herramientas de Desarrollo

# ¿Qué es un Compilador?

- Programa que hace programas, un meta programa
- Traductor
- Baja el Nivel de Abstracción
- Función de Lenguaje a Lenguaje:  $C: L_1 \rightarrow L_2$
- Familia de Compiladores
- Par  $(L_1, L_2)$
- Proceso de compilación tiene varias etapas, agrupadas en Front End y Back End
- Compilaciones separadas, luego vinculadas



Real programmers code in binary.



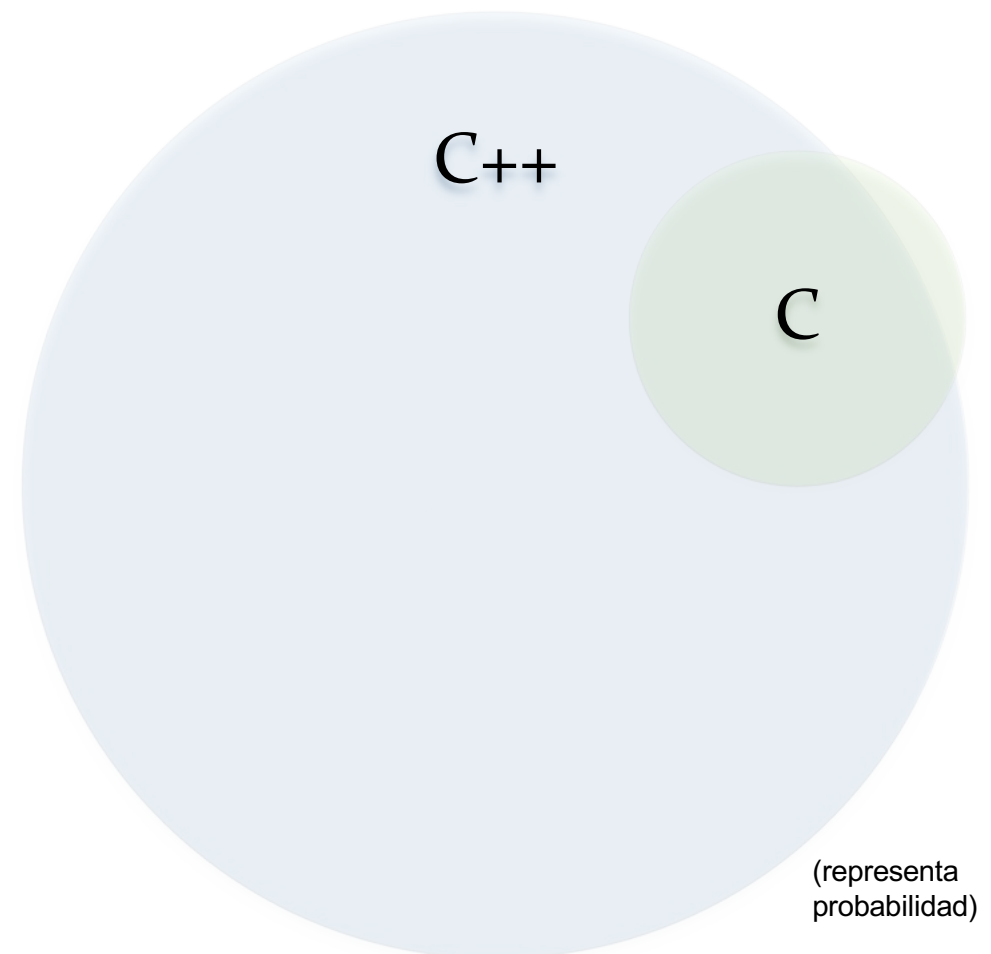


# Sobre los Lenguajes C y C++

## Historia

- 1970's
  - C
  - C With Classes
- 1980's
  - Comienza standard C
  - C++
- 1990's
  - Standard C90
  - Standard C++98
- 2000's
  - Standard C99
  - Standard C++03
- 2010's
  - Standard C11, C18
  - Standard C++11, 14, 17,
- 2020's
  - Standard C23
  - Standard C++23

Conjuntos de infinitos  
programas válidos de C++ y C



# "Hello, World!"

```
/* Hello.cpp
   C23
   JMS
   2015-2025
*/

#include <stdio.h>

int main(){
    printf("Hello, world!\n");
}
```

- Propósito
- Comentario encabezado
  - Qué
    - Título descriptivo
  - Quién
    - Número de Equipo e integrantes
  - Cuándo
    - Se actualizó por última vez

# Proceso básico para desarrollar programas

1. **Escribir** el programa con un editor de texto (e.g., vi, Notepad, TextPad, Sublime, TextMate, Notepad++, Notepad2). Es convención para los archivos fuente de C la extensión sea .c (e.g., hello.c)
2. **Compilar** el archivo fuente para producir el programa objeto (e.g., cc hello.c) ...  
... y **Vincular** (link) el programa con las bibliotecas para crear el programa ejecutable; generalmente ocurre junto con el punto anterior.
3. **Ejecutar** el programa (e.g., hello.exe ó ./a.out)
4. ¿Error en 2 ó 3? Volver a 1 y repetir.

# Ejemplo desde línea de comando macOS

1. Desde la línea de comando

1. `> vim hello.c` *crear el fuente*
2. `> cc hello.c` *compilar y crear el ejecutable, en realidad: Preprocesador → Compilador → Linker*
3. `> ./a.out` *ejecutar*  
Hello, world! *S alida*

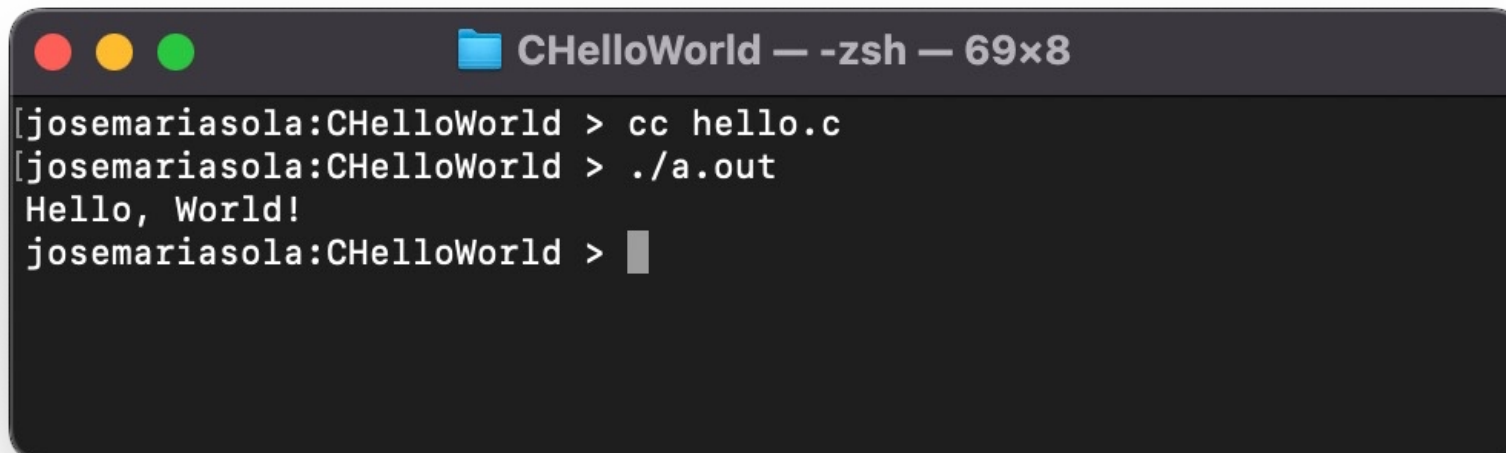
2. Si hay un error en el paso 2 ó 3, volver al 1 y repetir 2 y 3

- **Otras alternativas para cc son:**

- `> cc hello.c -o hello` Indica el nombre del ejecutable salida de la compilación (output), en vez del default `a.out`
- `> cc hello.c -std=c23` Compila según la *versión del estándar C23*

- **Opciones para controlar warnings (*pragmática*) y diagnóstico**

- `-Wall` Advierte sobre un conjunto amplio de posibles errores
- `-Wextra` Incluye más advertencias
- `-pedantic` Advierte sobre uso del lenguaje que no se adhiere al estándar
- `-weverything` (solo *clang*) Incluye todas las advertencias inclusive las experimentales.
- `-Werror` Trata las advertencias como errores y no genera código objeto
- `-pedantic-errors` Trata las no adherencias al estándar como errores y no genera código objeto



```
[josemariasola:CHelloWorld > cc hello.c  
[josemariasola:CHelloWorld > ./a.out  
Hello, World!  
josemariasola:CHelloWorld > ]
```

# Ejemplo desde línea de comando

## Compilador Microsoft (ejemplo en C++, no C)

1. Desde la línea de comando
  1. **> notepad hello.c** crear el fuente
  2. **> cl hello.c** crear el ejecutable  
en realidad: Preprocesador → Compilador → Linker
  3. **> hello.exe** ejecutar  
Hello, World! salida
2. Si hay un error en el paso 2 ó 3, volver al 1 y repetir 2 y 3



```
C:\Samples>cl Hello.cpp
Microsoft (R) C/C++ Optimizing Compiler Version 18.00.21005.1 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

Hello.cpp
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\INCLUDE\xlocale(337) : wa
rning C4530: C++ exception handler used, but unwind semantics are not enabled. S
pecify /EHsc
Microsoft (R) Incremental Linker Version 12.00.21005.1
Copyright (C) Microsoft Corporation. All rights reserved.

/out:Hello.exe
Hello.obj

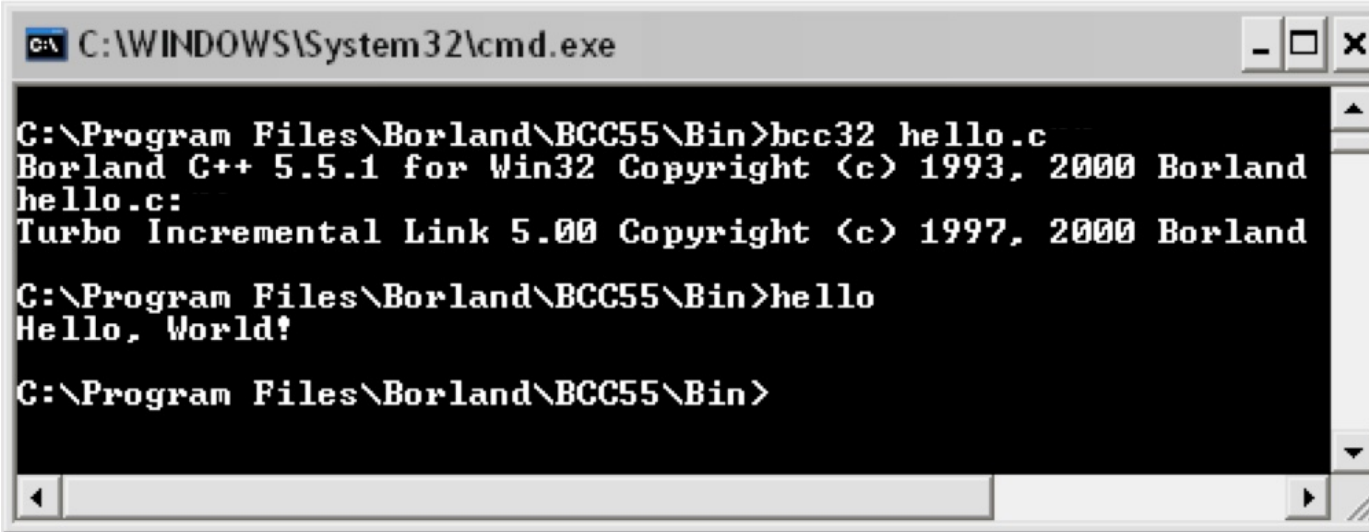
C:\Samples>hello
Hello World!

C:\Samples>
```

# Ejemplo desde línea de comando

## Compilador Borland

1. Desde la línea de comando
  1. > **notepad hello.c** crear el fuente
  2. > **bcc32 hello.c** crear el ejecutable  
en realidad: Preprocesador → Compilador → Linker
  3. > **hello.exe** ejecutar  
Hello, World! salida
2. Si hay un error en el paso 2 ó 3, volver al 1 y repetir 2 y 3



```
C:\WINDOWS\System32\cmd.exe

C:\Program Files\Borland\BCC55\Bin>bcc32 hello.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
hello.c:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

C:\Program Files\Borland\BCC55\Bin>hello
Hello, World!

C:\Program Files\Borland\BCC55\Bin>
```



# Herramientas de Desarrollo:

## Sobre el Compilador y el IDE

- Con IDE (*Integrated Development Enviroment*, Entorno Integrado de Desarrollo)
  - Ejemplos
    - Apple Xcode
    - Microsoft Visual Studio
    - Eclipse
- Sin IDE
  - Editor
  - Compilador.

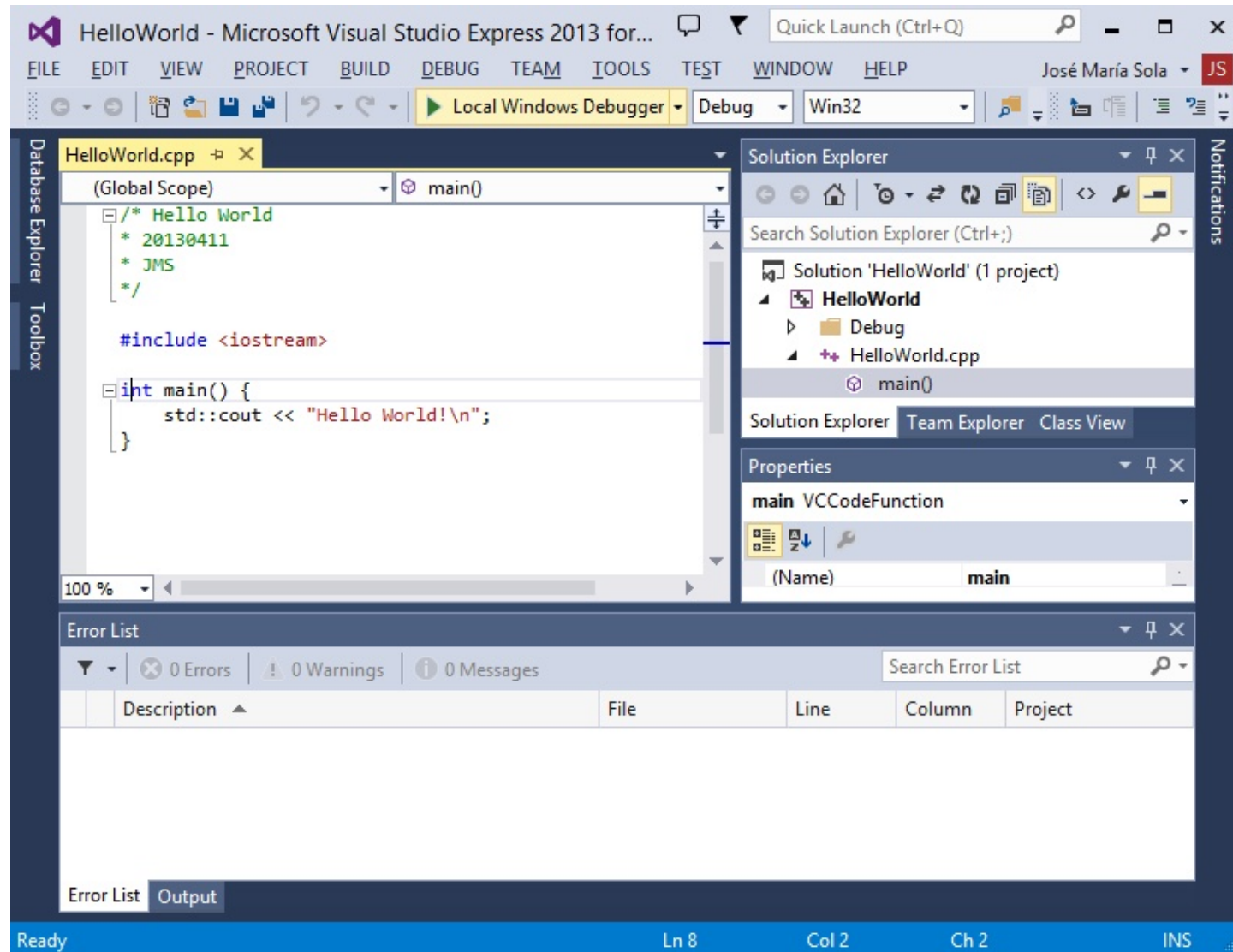
### Con IDE

- Editor
- Depurador
- Gestor de Proyectos y de configuraciones
- Ayuda
- y más...

**Sin IDE**  
Requiere editor

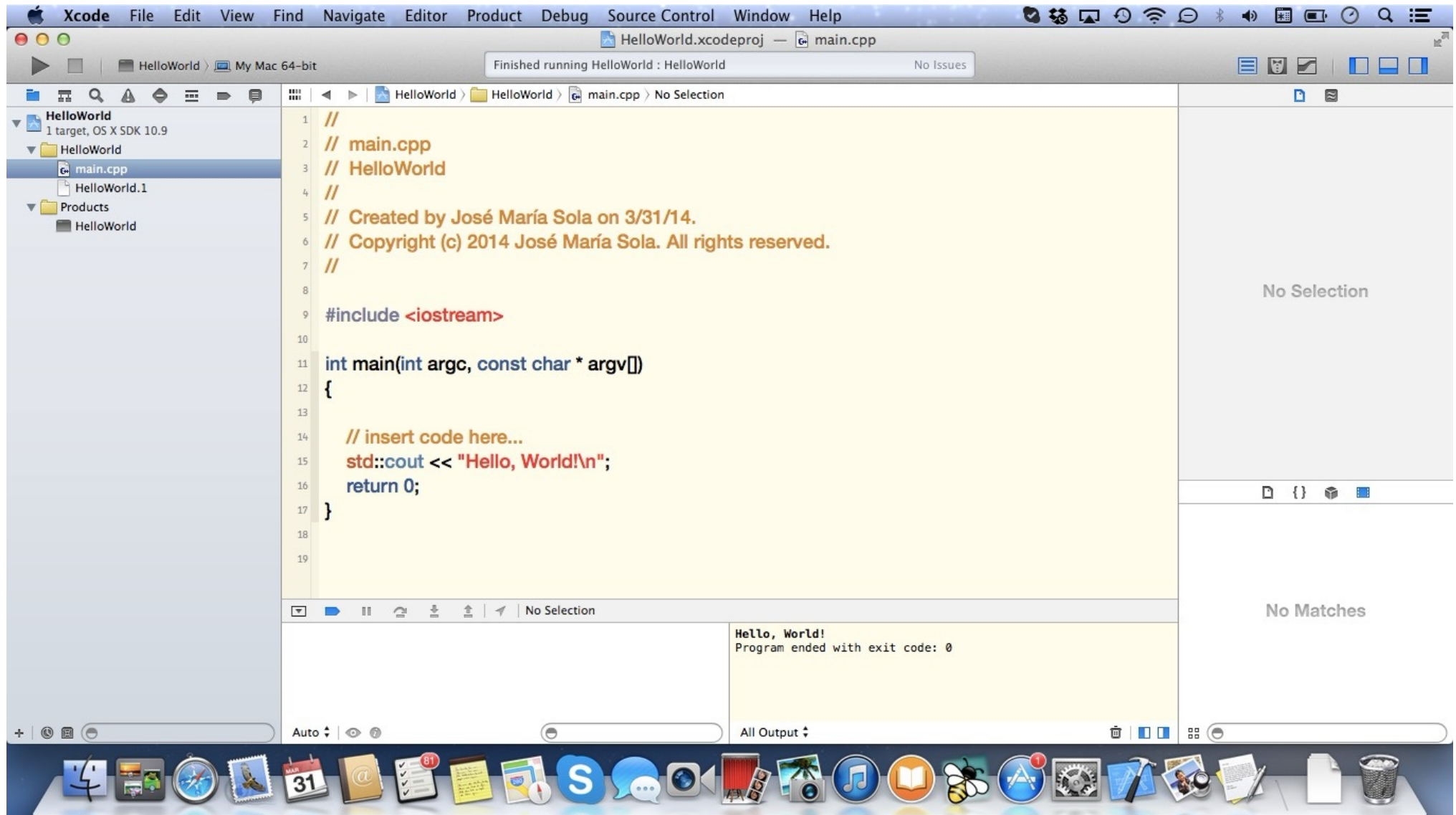
Compilador de C/C++

# Ejemplo con IDE Microsoft Visual Studio Express for Windows Desktop (Ejemplo en C++, no C)



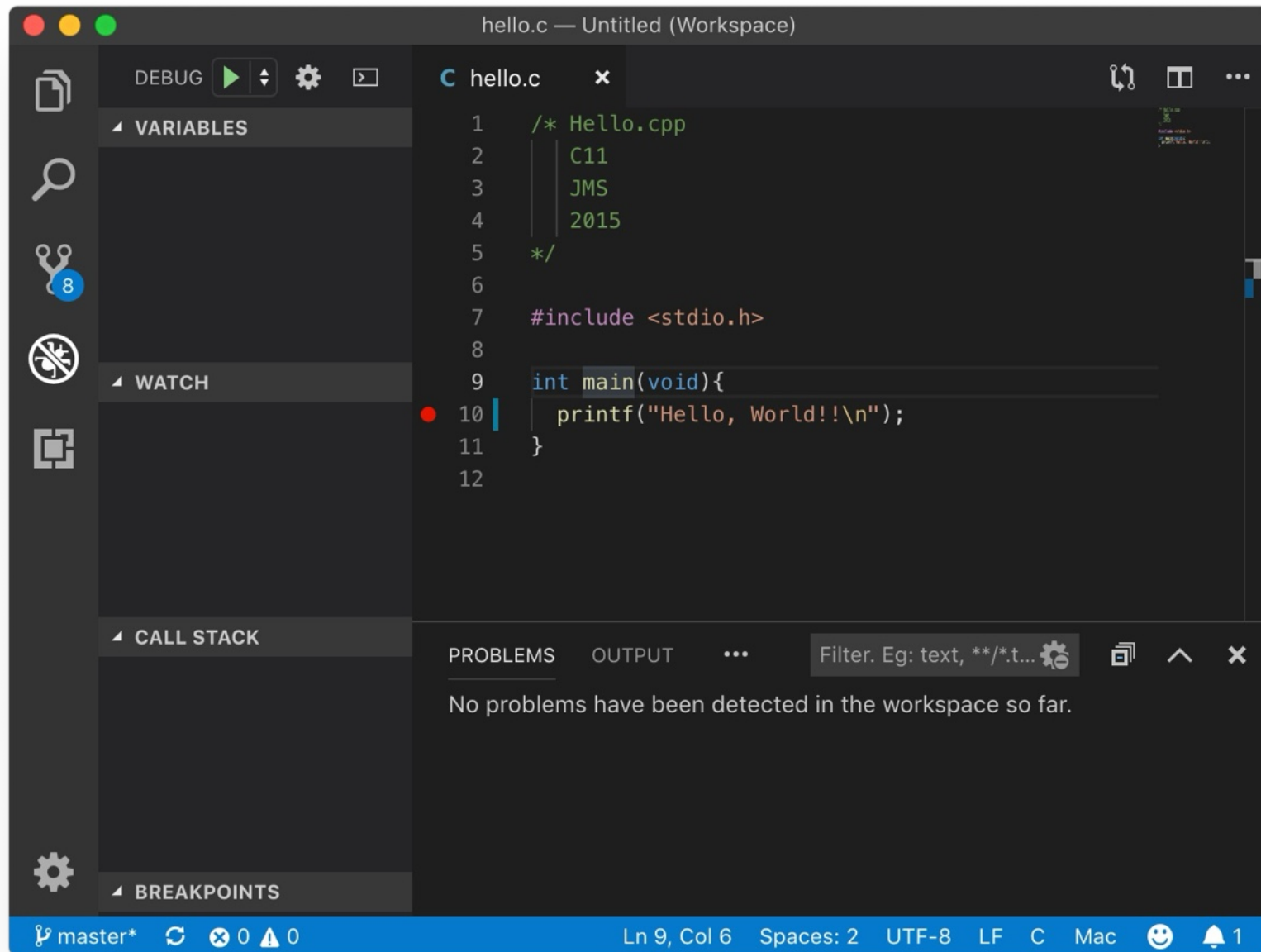
# Ejemplo con IDE

## Apple Xcode (ejemplo en C++, no C)



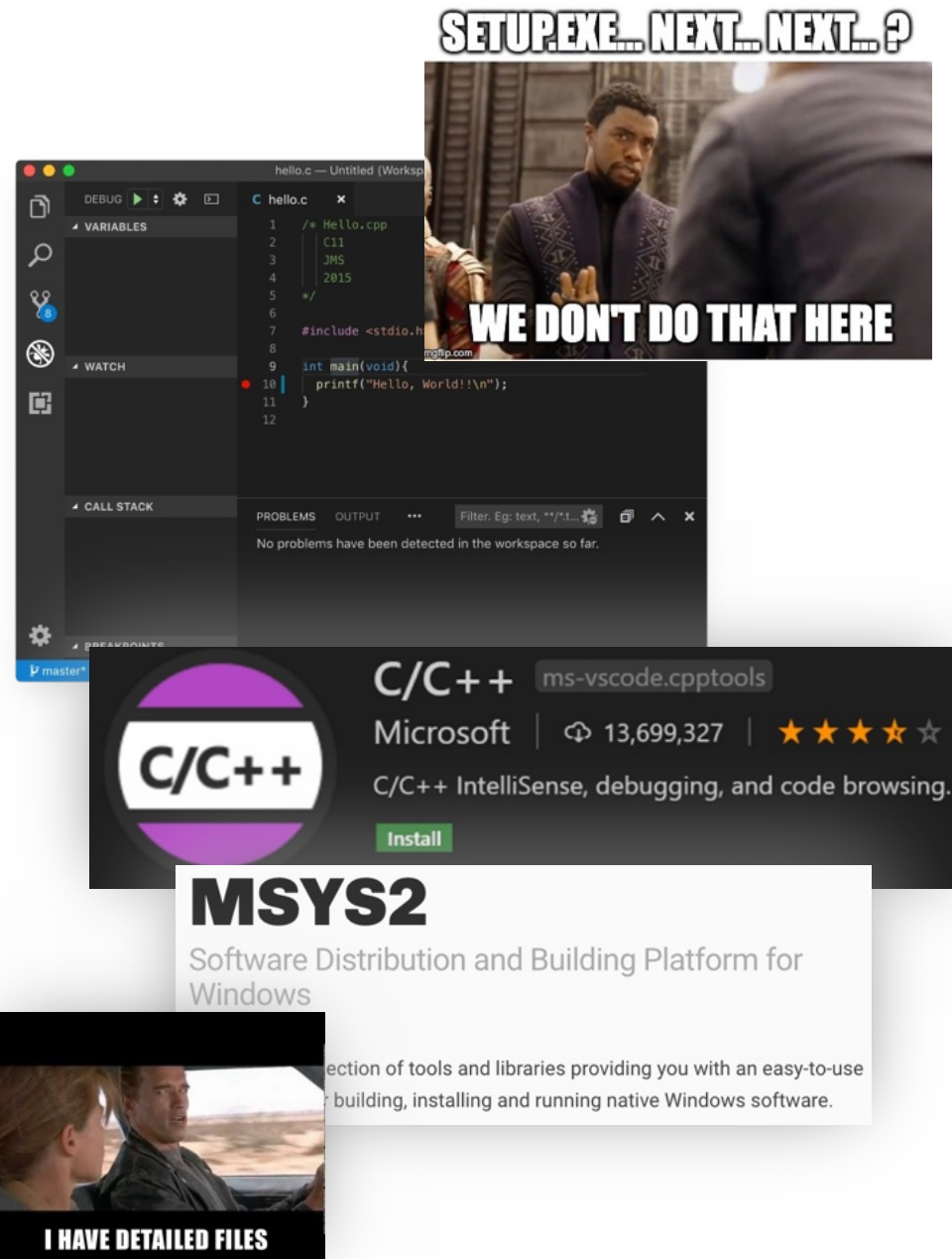
# Ejemplo con IDE

## Microsoft Visual Studio Code



# Instalación de Microsoft Visual Studio Code

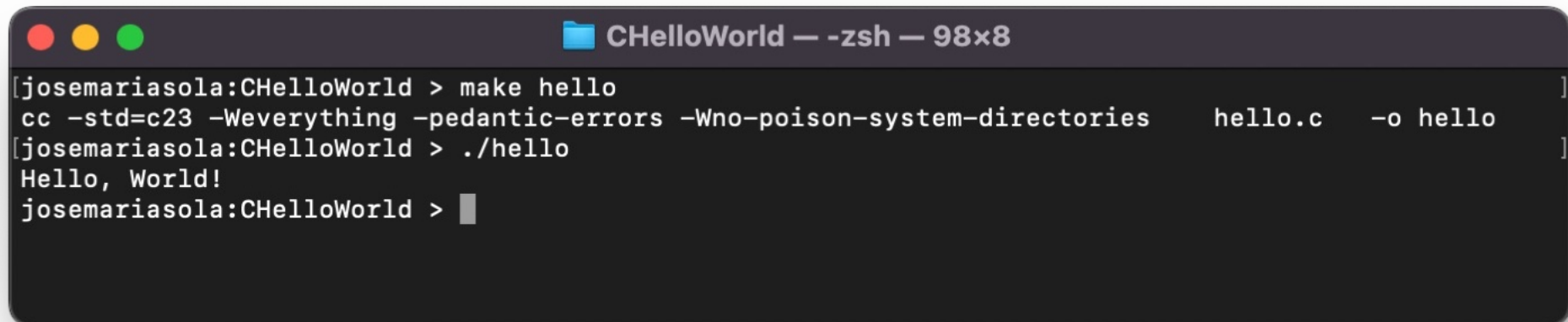
- Es más que un editor de texto, es un editor de código fuente
- Hoy en día es muy popular
- Agrega funcionalidad mediante un sistema de extensiones, para nuestro caso: C/C++
- La extensión no incluye un compilador ☹, por lo que hay que conseguir uno, por ejemplo gcc, clang, o cl
  - Los sistemas Windows pueden obtener gcc mediante MinGW (the Minimalist GNU for Windows)
  - MinGW se puede obtener desde [MSYS2](#)
- [La extensión C/C++ tiene documentación detallada sobre la instalación de todo el tool-chain](#)





# Make (C)

- Nuestro objetivo (*goal*) es fabricar (construir, hacer, *make*) la versión ejecutable de `hello.c`
- Desde la línea de comando podemos lograrlo con el comando `make`, pasándole como argumento el nombre del *goal*, en nuestro caso `hello`
- El comando `make` sabe *makear* un ejecutable a partir de un fuente
- En sistemas *Windows* el análogo es el comando `nmake.exe`, aunque también es posible utilizar el `make.exe` ó `mingw32-ake.exe` si instalamos *MinGW*.



```
CHelloWorld — -zsh — 98x8
[josemariasola:CHelloWorld > make hello
cc -std=c23 -Weverything -pedantic-errors -Wno-poison-system-directories    hello.c    -o hello
[josemariasola:CHelloWorld > ./hello
Hello, World!
josemariasola:CHelloWorld > █
```



# Links a Compiladores C/C++

Usar unos de estos compiladores o cualquier otro, siempre y cuando se lo configure para **C23**

- Con IDE y Línea de Comandos
  - Microsoft Visual Studio Community 2022
    - <https://www.visualstudio.com/vs/features/cppplusplus/>
  - Microsoft Code
    - <https://code.visualstudio.com/Download>
  - Apple Xcode
    - <https://developer.apple.com/xcode>
  - CodeLite
    - <https://codelite.org>
  - Eclipse IDE for C/C++ Developers
    - <https://www.eclipse.org/downloads/packages/release/2025-03/r/eclipse-ide-cc-developers>
  - Replit (IDE On-Line)
    - <https://replit.com/languages/c>
  - Más antiguos
    - Code::Blocks
      - <http://www.codeblocks.org/downloads/>
    - Dev-C++
      - <http://orwelldevcpp.blogspot.com>
- Sin IDE, solo Línea de Comandos
  - Si tu sistema es un UNIX (macOS, GNU, Linux) es probable que incluya un compilador, probá los comandos **cc** y **gcc** desde la línea de comandos
  - GNU C Compiler (ahora GNU Compiler Collection)
    - <http://gcc.gnu.org/install/binaries.html>
    - Para plataformas Windows
      - <http://www.mingw.org>
      - <http://mingw-w64.org/doku.php>
  - Clang
    - <http://releases.llvm.org/download.html>
  - Embarcadero Free C++ Compiler
    - <https://www.embarcadero.com/free-tools/ccompiler>
  - Más antiguos
    - Borland C++ Compiler version 5.5 Free Download
      - <http://edn.embarcadero.com/article/20633>
        - Using the Borland 5.5 Compiler and command-line tools
      - <http://edn.embarcadero.com/article/20997>
        - Borland C++ 5.5 Free Command-line Tools Supplementary Information
      - <http://edn.embarcadero.com/article/21205>
- Compilador On-line:
  - Compiler Explorer: <https://godbolt.org>

# Compiladores, Editores y Entornos de Desarrollo: Instalación, Configuración y Prueba

- <https://josemariasola.wordpress.com/papers/#CompiladoresInstalacion>
- Introducción a compilador, entornos de desarrollo
- Amar de entorno de desarrollo para C/C++ bajo un entorno Windows, basado en el compilador MinGW y el editor de código fuente Visual Studio Code.

# Trabajo #0

"Hello, World!" en C

# Trabajo #0 — "Hello, World!" en C

- Enunciado en <https://josemariasola.wordpress.com/ssl/assignments/>
- Secuencias de Tareas
  - Si no posee una cuenta GitHub, crearla
  - Crear un repositorio público llamado SSL
  - Escribir el archivo readme.md que actúa como front page del repositorio personal
  - Crear la carpeta oo-CHelloWorld
  - Escribir el archivo readme.md que actúa como front page de la resolución
  - Seleccionar, instalar, y configurar un compilador C23
  - Indicar en readme.md el compilador seleccionado
  - Probar compilador con hello.c que envíe a stdout la línea Hello, World! o similar
  - Ejecutar el programa, y capturar su salida en un archivo de texto output.txt
  - Publicar en repositorio personal SSL \ oo-CHelloWorld: readme.md, hello.c, y output.txt
  - Informar el usuario usuario GitHub en la lista indicada en el curso.
- Restricciones
  - La fecha y hora límite de entrega se publica en el calendario
  - La evaluación se hace con lo publicado en GitHub.

# Términos de la clase #02

Definir cada término con la bibliografía

- SSL y Relación con Otras Asignaturas
  - Áreas de Conocimiento de nuestro plan de estudio
  - Repaso Materias Anteriores
    - Conjuntos Numéricos
    - Ecuaciones e Inecuaciones
    - Función
    - Vector
    - Abstracción
    - Tipo de Dato
    - Orientación a Objetos
    - `assert`
    - `enum`
      - <https://josemariasola.wordpress.com/aed/papers/#Enums>
- Primer Contacto con el Compilador
  - Propósito de Compilador
  - Compilador como Función
  - Familia de Compiladores
- Proceso de compilación
- Lenguaje máquina (bajo nivel de abstracción)
- Lenguaje de Alto Nivel de Abstracción
- C
- C++
- Hello World (Kernighan)
- Proceso básico para desarrollar programas
- Línea de comandos
- Warnings
- IDE (Integrated Development Enviroment, Entorno Intedrado de Desarrollo)
- Utilidad Make

# Tareas para la próxima clase

1. Repasar conceptos de asignaturas anteriores
2. Leer de [MUCH]
  - $V_2C_3$
  - $V_2C_{3.1}$
3. Realizar ejercicios de [MUCH]  $V_2C_{3-1}$  y 2
4. Continuar con el Trabajo #0



# ¿Consultas?

**Fin de la clase**