UNIVERSITÀ DEGLI STUDI DI BERGAMO

# Informatica
# Modulo di Programmazione

## Informatica
## Modulo di Programmazione
## Getting Started with C++

mauro.pelucchi@gmail.com
Mauro Pelucchi

2023/2024

# Agenda

- C++ History and Motivation
- Building Programs with VS Code
- Debugging with VC Code
- C++ Basics

# Linguaggio di programmazione

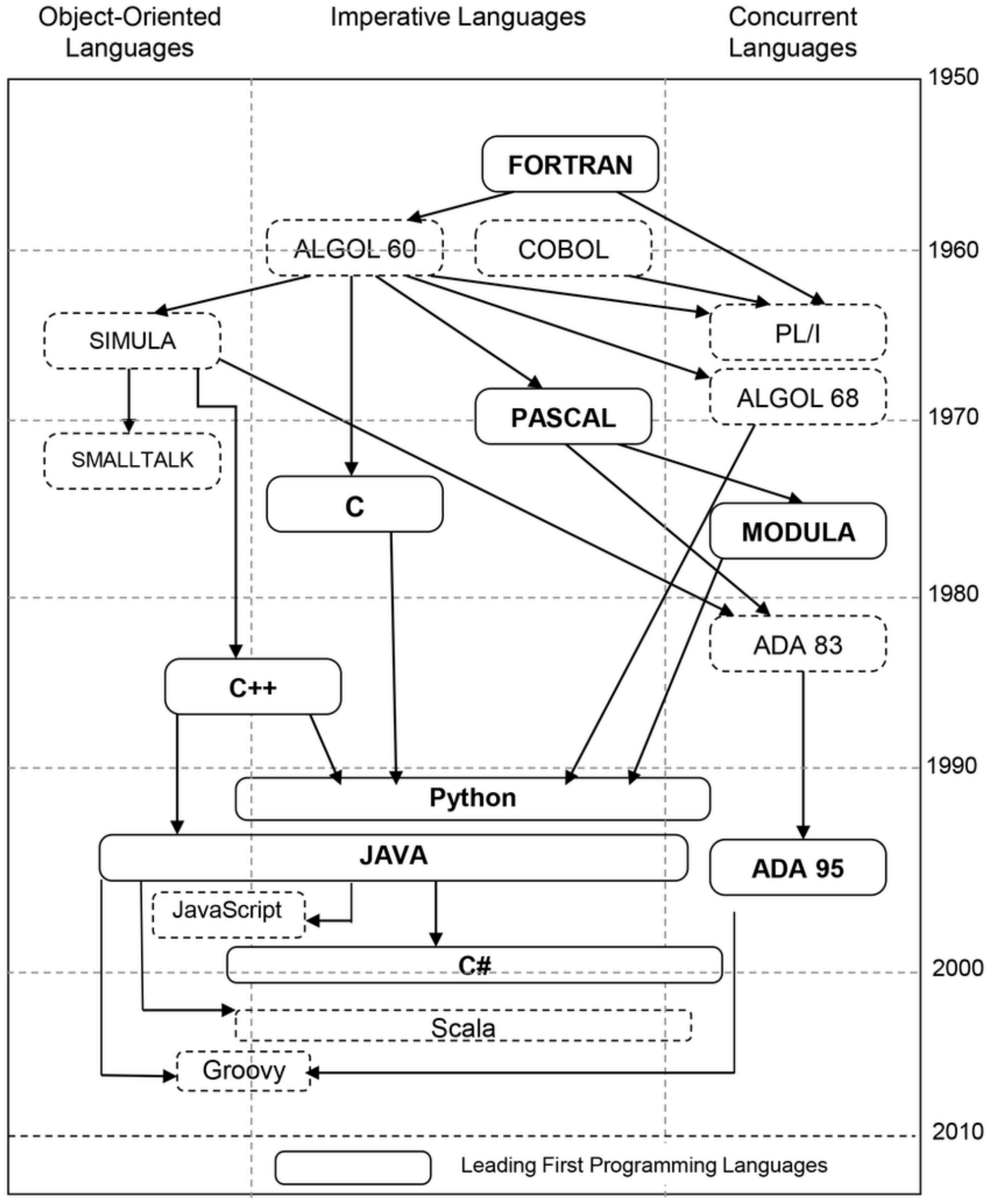
Un linguaggio di programmazione, formale (per la codifica) consente di scrivere un algoritmo sotto forma di programma eseguibile da un calcolatore

# Linguaggio di programmazione

- **Linguaggi macchina**
  - Ogni loro istruzione (vocabolo) è composta come una sequenza cifre binarie direttamente decodificabile dalla specifica macchina (CPU) per essere attuata
- **Linguaggi assembler**
  - Linguisticamente più vicini alle istruzioni eseguite direttamente dalla macchina (CPU)
  - 1 istruzione assembler equivale a 1 istruzione in linguaggio macchina)
- **Linguaggi di alto livello**
  - Linguisticamente più vicini al linguaggio naturale
  - 1 istruzione equivale a 2+, o anche 10+, istruzioni in linguaggio macchina

# Componenti di un linguaggio

| Vocabolario | Sintassi | Semantica |
| --- | --- | --- |
| Parole chiave del linguaggio riconosciute dal parser (analizzatore lessicale) | Regole per comporre i simboli del vocabolario<br><br>Il controllo della sintassi avviene tramite l'analizzatore sintattico | Significato delle espressioni<br><br>Un errore semantico, in genere, può essere rilevato solo a tempo di esecuzione |


Object-Oriented Languages
Imperative Languages
Concurrent Languages
1950
1960
1970
1980
1990
2000
2010
FORTRAN
ALGOL 60
COBOL
SIMULA
PL/I
ALGOL 68
PASCAL
SMALLTALK
C
MODULA
ADA 83
C++
Python
JAVA
ADA 95
JavaScript
C#
Scala
Groovy
Leading First Programming Languages

# Compilatori

- I **compilatori** sono software che traducono i programmi scritti in un linguaggio d'alto livello in codice macchina

- Il programma una volta che **è stato interamente tradotto** viene eseguito dal calcolatore

- Il maggior vantaggio della compilazione è senz'altro l'efficienza in termini di **prestazioni**

- Il codice tradotto in linguaggio macchina è valido solo per una piattaforma specifica (combinazione di architettura hardware e sistema operativo)

# Interpreti

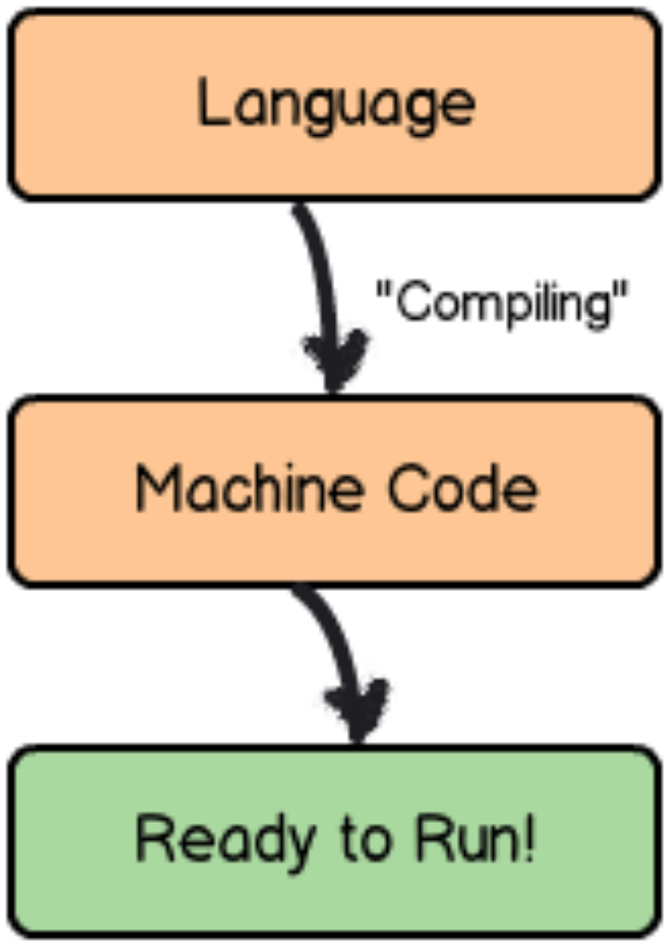
- Gli **interpreti** sono programmi che **traducono ed eseguono** ciascuna istruzione del programma scritto in un linguaggio d'alto livello in modo sequenziale.

- Un programma scritto in un linguaggio interpretato **non ha**, in linea di massima, **dipendenze** dalla specifica piattaforma su cui viene eseguito

- Ma è **più lento** e richiede più memoria in fase di esecuzione

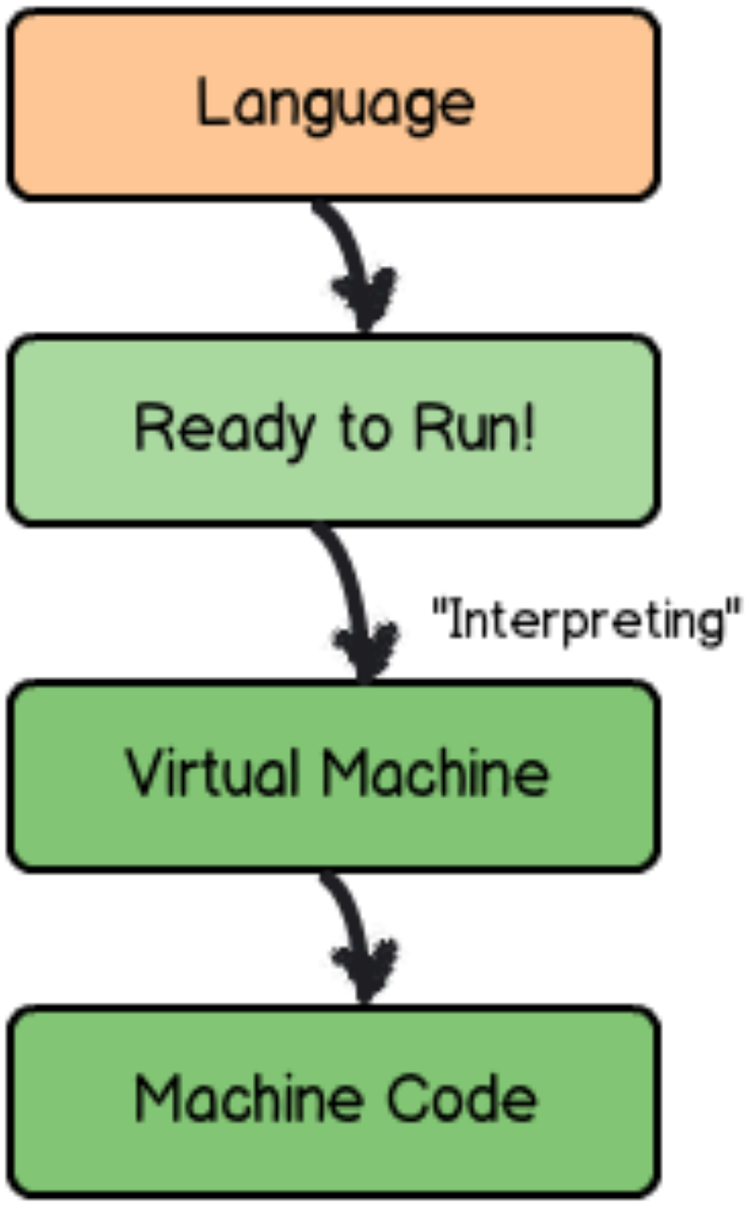
## Compiled

C, C++, Go, Fortran, Pascal

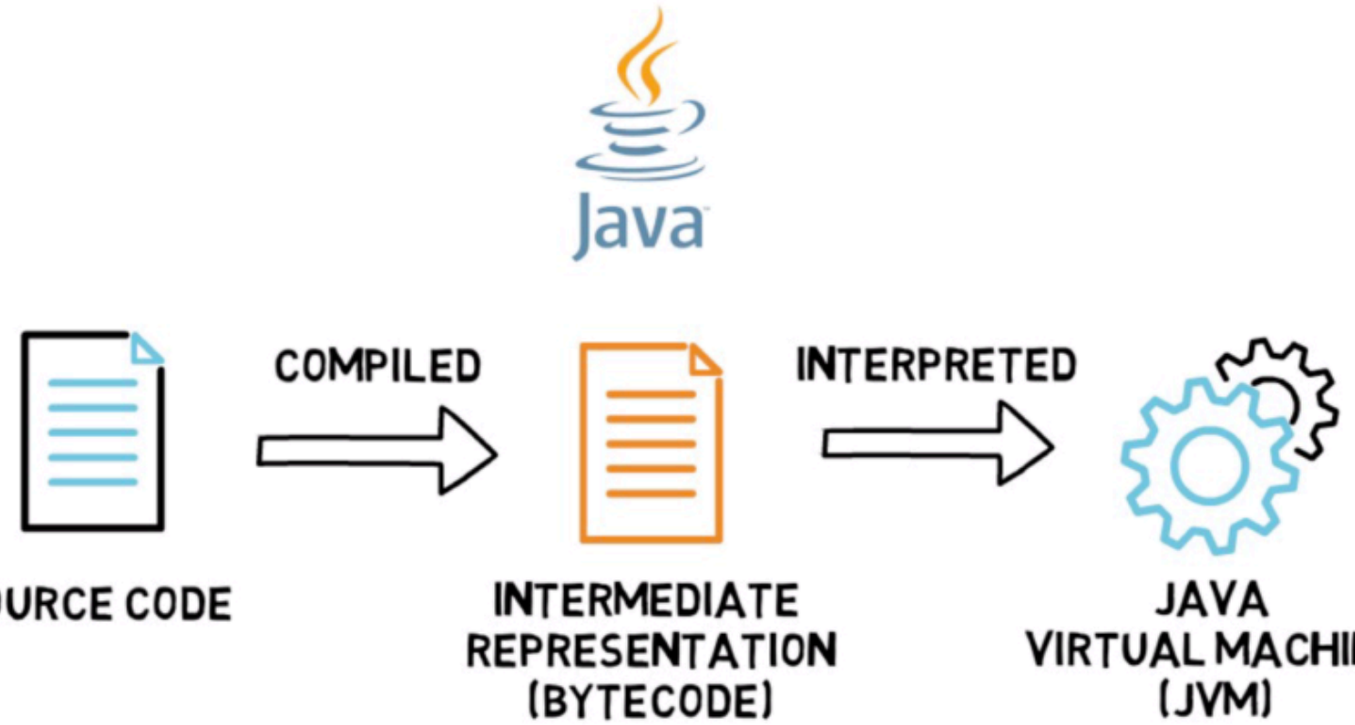



## Interpreted

Python, PHP, Ruby, JavaScript



# Bytecode




Il codice sorgente dei programmi non viene compilato in linguaggio macchina, ma viene tradotto in un **codice intermedio "ibrido"** destinato a venire interpretato al momento dell'esecuzione del programma

# Bytecode

- Il motivo di questo doppio passaggio è di
  - avere la portabilità dei linguaggi interpretati
  - grazie alla pre-compilazione, un interprete più semplice e quindi più veloce
- Il codice intermedio è più facile sia da interpretare che da compilare
  - Per questo motivo sia per Java che per i linguaggi .NET (es. C#) sono stati sviluppati i compilatori JIT (Just In Time) che al momento del lancio di un programma Java, VB o .NET compilano al volo il codice intermedio e mandano in esecuzione un normale codice macchina nativo, eliminando completamente la necessità dell'interprete e rendendo i programmi scritti in questi linguaggi veloci quasi quanto i normali programmi compilati

# C++ Goals

Support for effective program organization

Produce programs that run fast

Combine separately compilable units into program

Allow highly portable implementations

Model module structure of system and pattern of communication between modules

# Timeline C++

- 1980 —> initial implementation
- 1983 —> first real application of **C with Classes** was network simulators; renamed to C++
- 1987 —> first GNU C++ release
- 1992 —> first Microsoft C++ release; first IBM C++ release
- 2014 —> ISO/IEC 14882:2014 (informally known as C++14)

# Why C++ as starting point?

- Flexibilty
- Efficiency
- Availability
- Portability

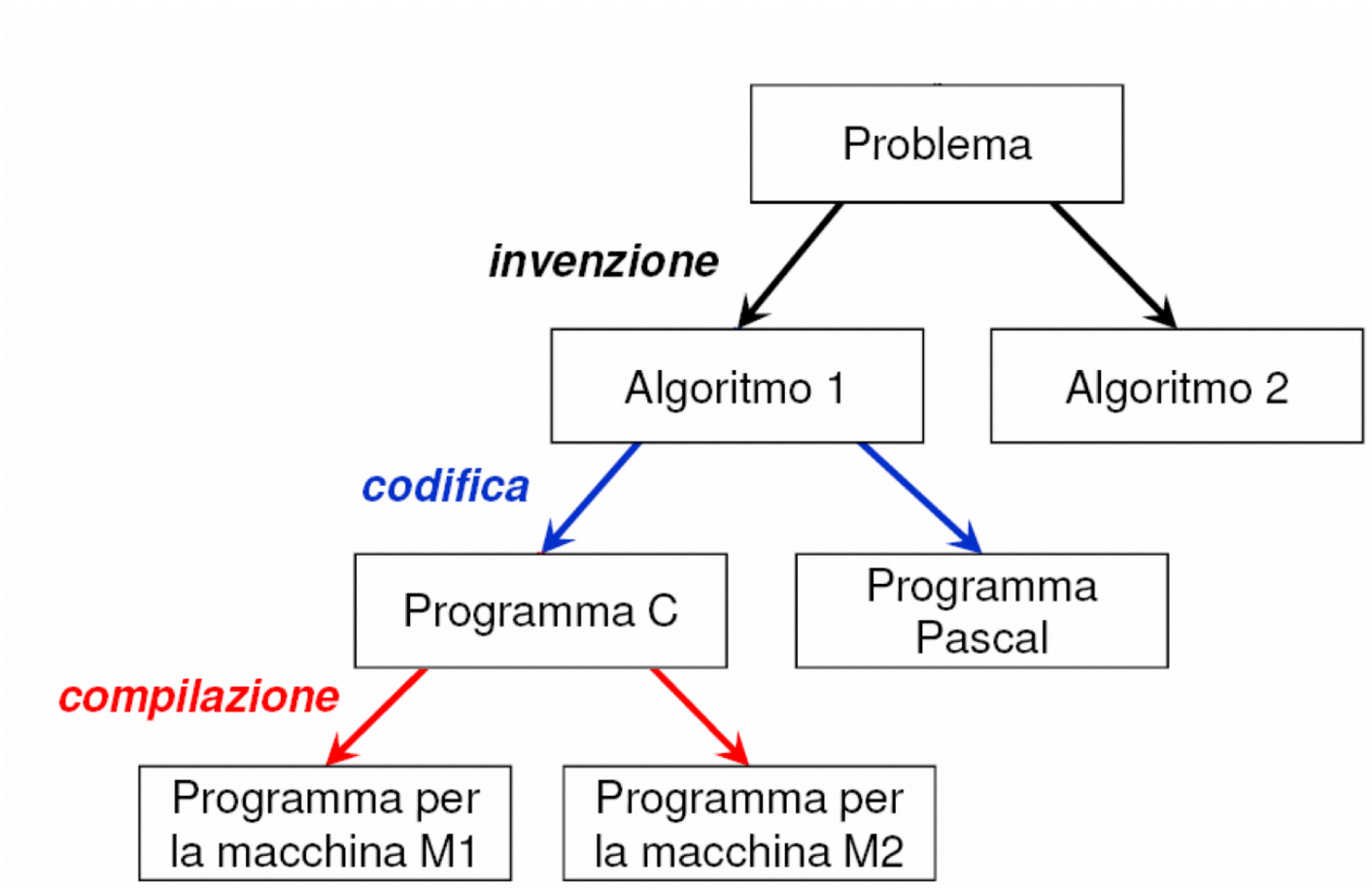
# Application Areas

- banking and financial
- classical systems programming (compilers, operating systems, device drivers, network layers, editors, database systems)
- small business applications (inventory systems)
- desktop publishing (document viewers/editors, image editing)
- embedded systems (cameras, cell phones, airplanes, medical systems, appliances)
- entertainment (games)
- GUI
- hardware design
- scientific and numeric computation (physics, engineering, simulations, data analysis, geometry processing)
- servers (web servers, billing systems)
- telecommunication systems (phones, networking, monitoring, billing, operations systems)

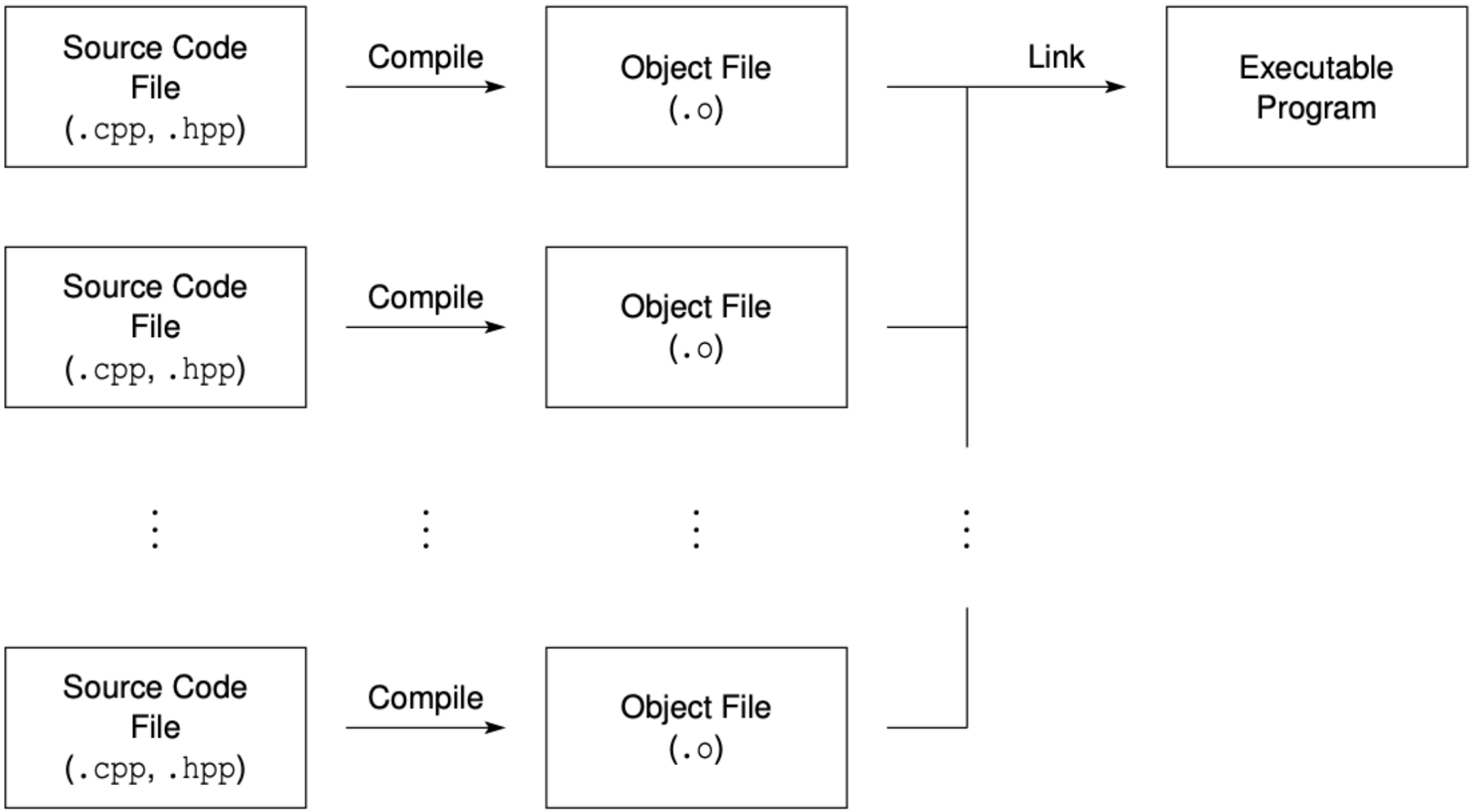
# Cosa è un algoritmo?

# Problemi, Algoritmi E Programmi

- Algoritmo: escogitare e formalizzare sequenze di passi che risolvono un problema
  - codificarli, con un linguaggio di programmazione, in programmi



# Software Build Process



# Software Build Process

1. Codifica di un algoritmo in un linguaggio di programmazione scelto dallo sviluppatore che prende il nome di **programma sorgente**.
    - Si tratta di un documento elettronico (file) contenente caratteri stampabili che rappresentano parole di un linguaggio che può essere sia
        - di basso livello: Assembly
        - sia di alto livello: C, C++, Java, …
2. Tramite opportuni strumenti si elabora il programma sorgente tramite due passaggi (in stretta sequenza)
    - Compilazione ( **Compiling** )
    - Collegamento ( **Linking** )
3. Si genera un altro documento elettronico (file) contenente anche caratteri NON stampabili con una precisa codifica binaria che verranno acquisiti dalla macchina preposta all'esecuzione.
    - Tale documento si dice essere scritto in linguaggio macchina ed è chiamato **programma eseguibile.**

# Software Build Process

Xxxx.cpp → Xxxx.obj → Xxxx.exe

| Editing | Compiling | Linking | Loading | Execution |
|---|---|---|---|---|
| Il testo del programma **sorgente**, costituito da una sequenza di caratteri, viene composto e modificato usando uno specifico programma: l'editor | Il sorgente viene tradotto usando uno specifico programma: il **compilatore**, in un Programma in **codice macchina** o Programma Oggetto costituito da una sequenza di byte | Il **linker** collega fra loro differenti programmi oggetto (**librerie**, che consentono di utilizzare sotto-programmi che eseguono un insieme di operazioni comuni a quasi tutti i software). Si genera un **Programma Eseguibile**. | Il **loader** individua una porzione della memoria centrale del calcolatore dove copiare il contenuto del programma eseguibile | Per eseguire il programma occorre fornire in ingresso i dati richiesti e in uscita riceveremo i risultati (su video o file o stampante) |

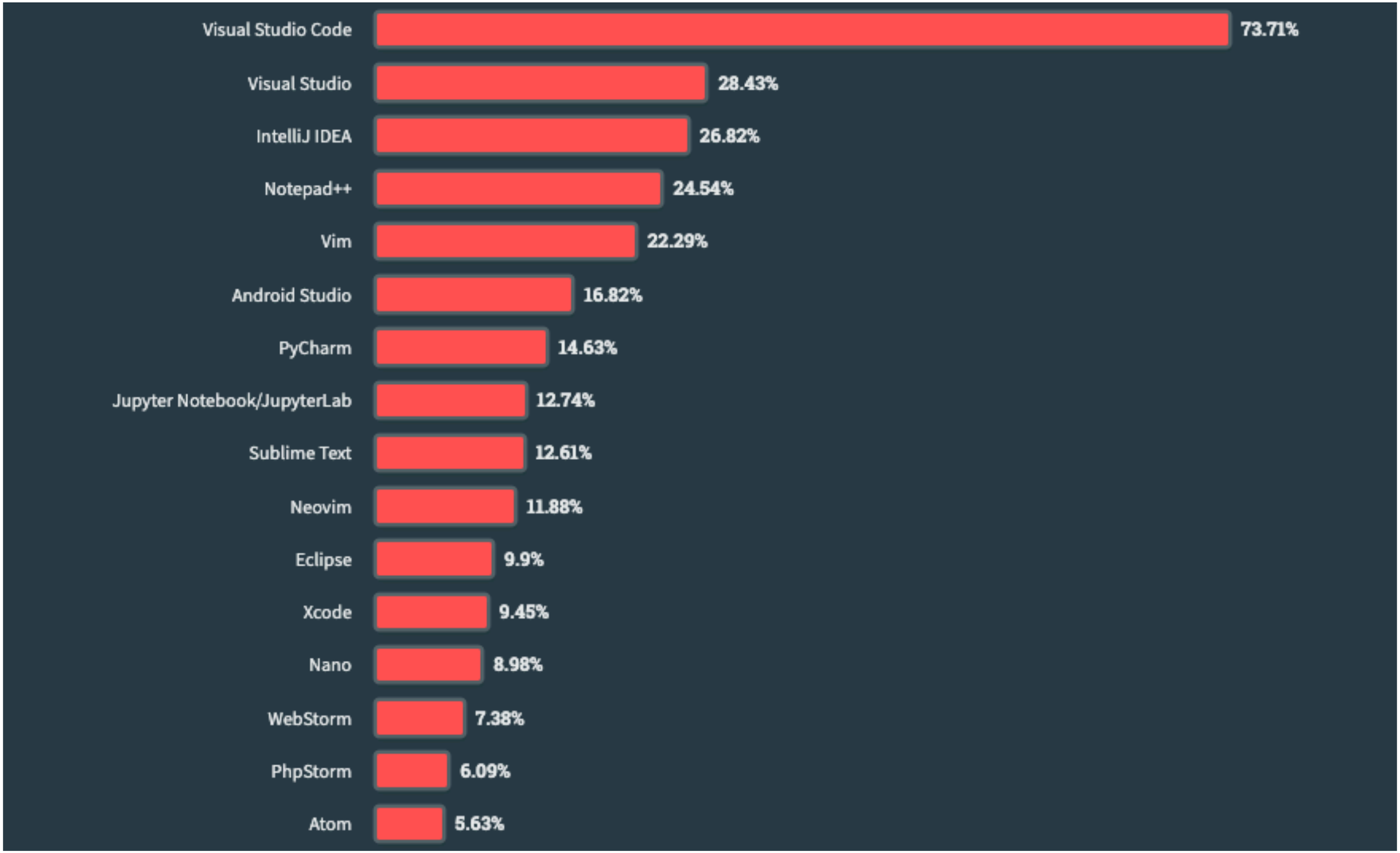
# IDE

## IDE = Integrated Development Environment

An integrated development environment (IDE) is software for building applications that combines common developer tools into a single graphical user interface (GUI).

An IDE typically consists of:

- **Source code editor:** A text editor that can assist in writing software code with features such as syntax highlighting with visual cues, providing language specific auto-completion, and checking for bugs as code is being written.
- **Local build automation:** Utilities that automate simple, repeatable tasks as part of creating a local build of the software for use by the developer, like compiling computer source code into binary code, packaging binary code, and running automated tests.
- **Debugger:** A program for testing other programs that can graphically display the location of a bug in the original code.

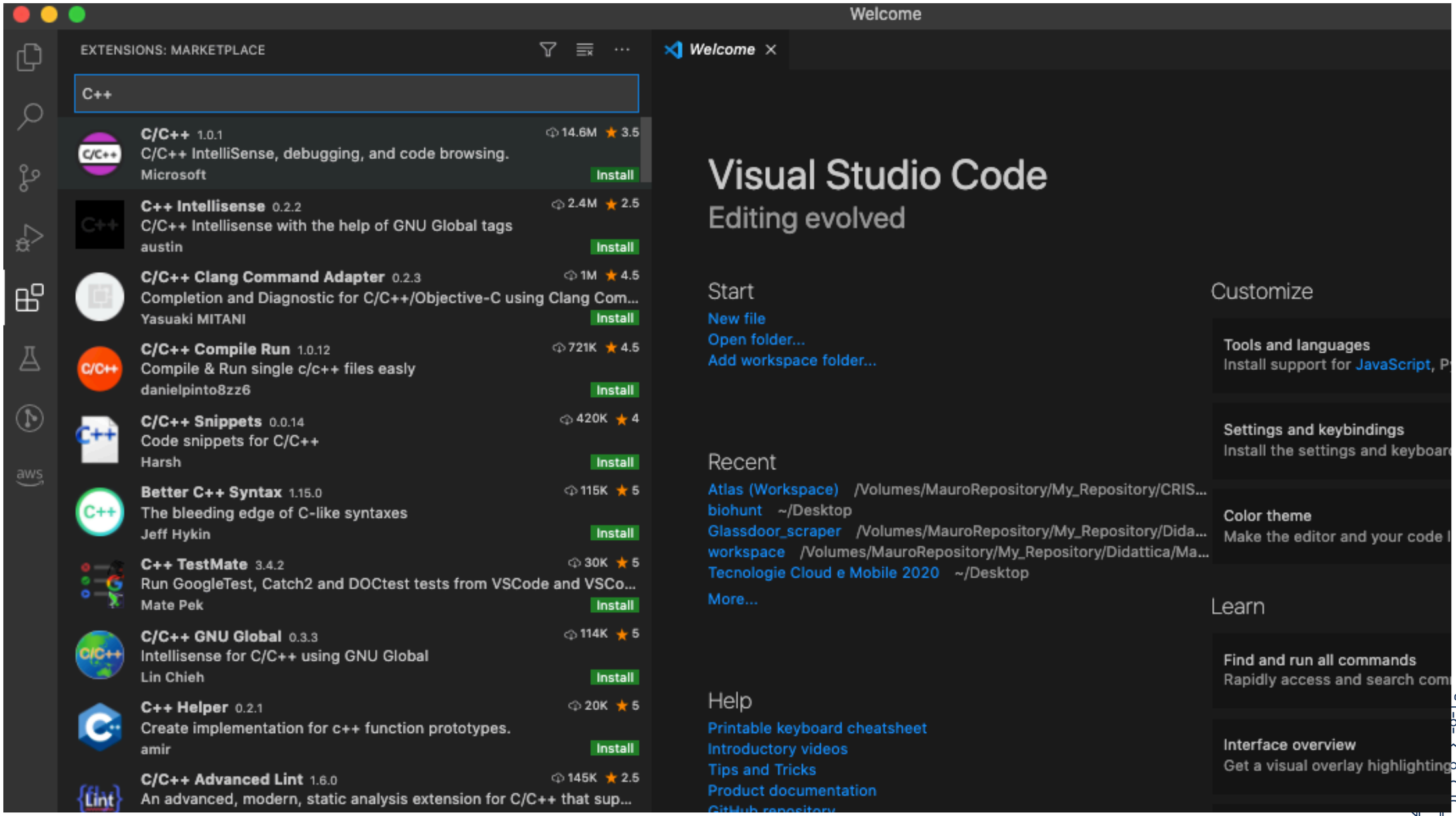



https://survey.stackoverflow.co/2023/#most-popular-technologies-new-collab-tools


# Visual Studio Code

# Visual Studio Code

## Aggiungiamo le estensioni per il C/C++



# Visual Studio Code

## C/C++ compiler and debugger

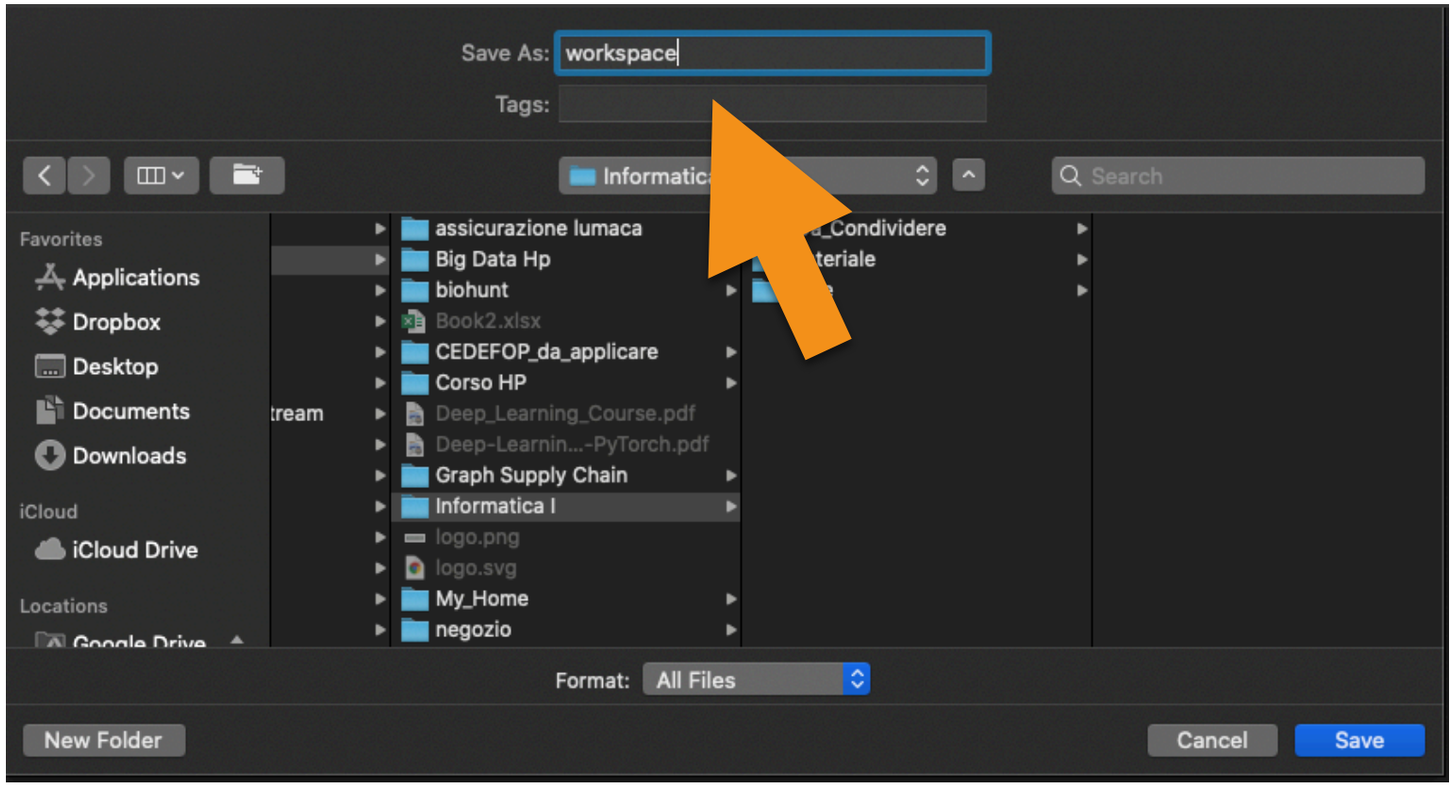
The C/C++ extension does not include a C++ compiler or debugger. You will need to install these tools or use those already installed on your computer.

Popular C++ compilers are:

- GCC on Linux
- GCC via Mingw-w64 on Windows
- Microsoft C++ compiler on Windows
- Clang for XCode on macOS

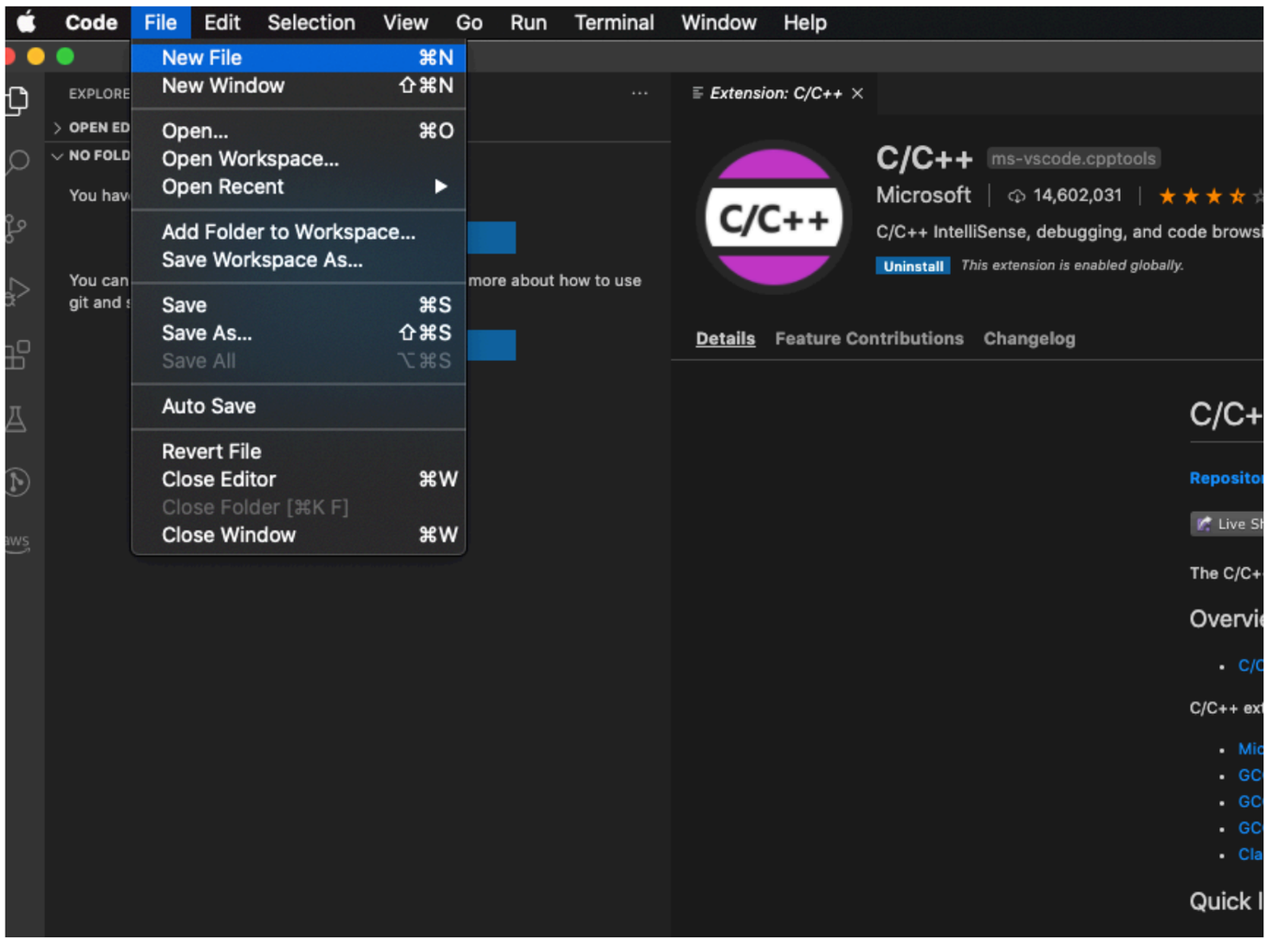
# Visual Studio Code Workspace

A Visual Studio Code workspace is a list of a project's folders and files. A workspace can contain multiple folders. You can customise the settings and preferences of a workspace.




Not use blank inside the full path of the workspace

# Visual Studio Code Workspace

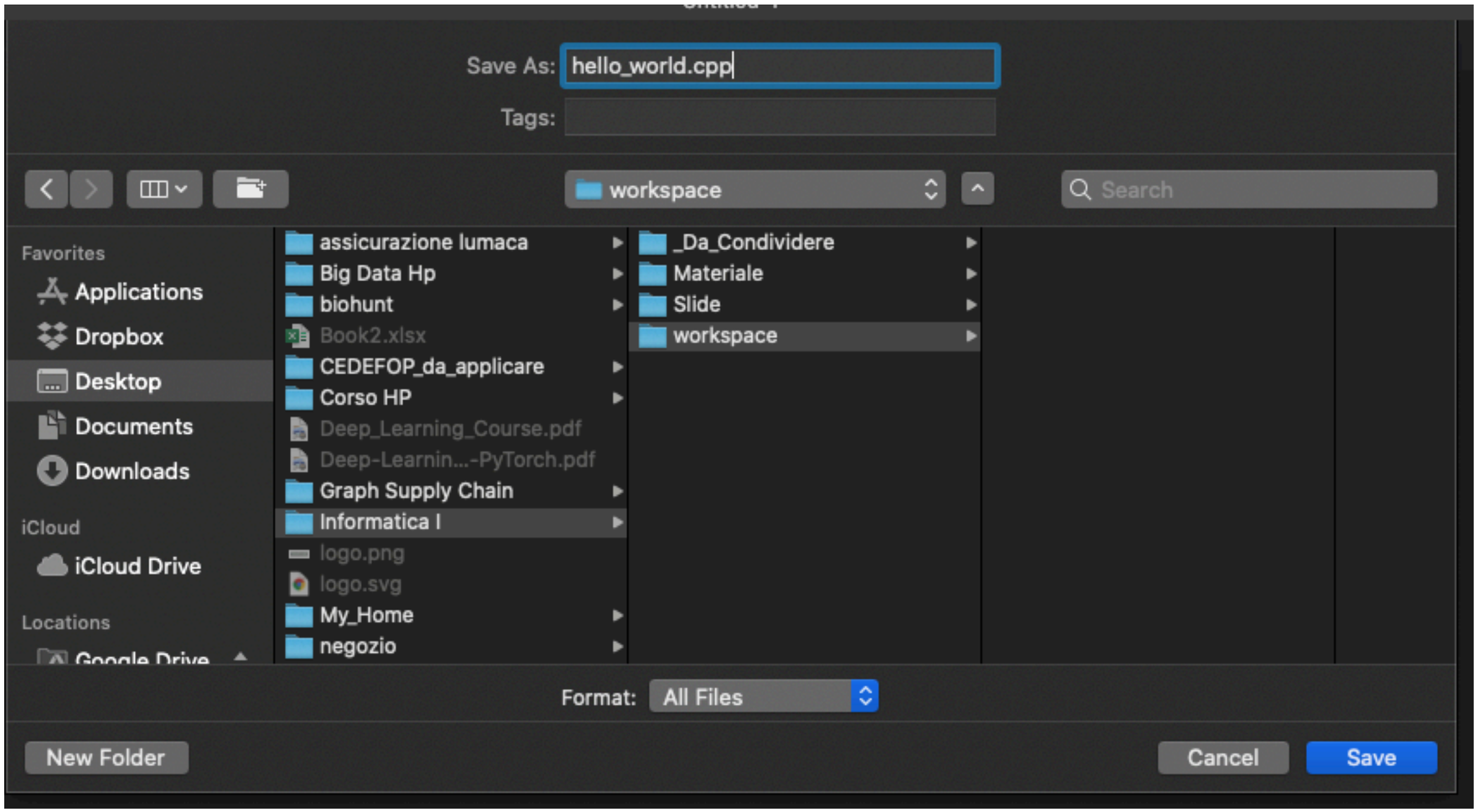



Creiamo un nuovo file

File > New File

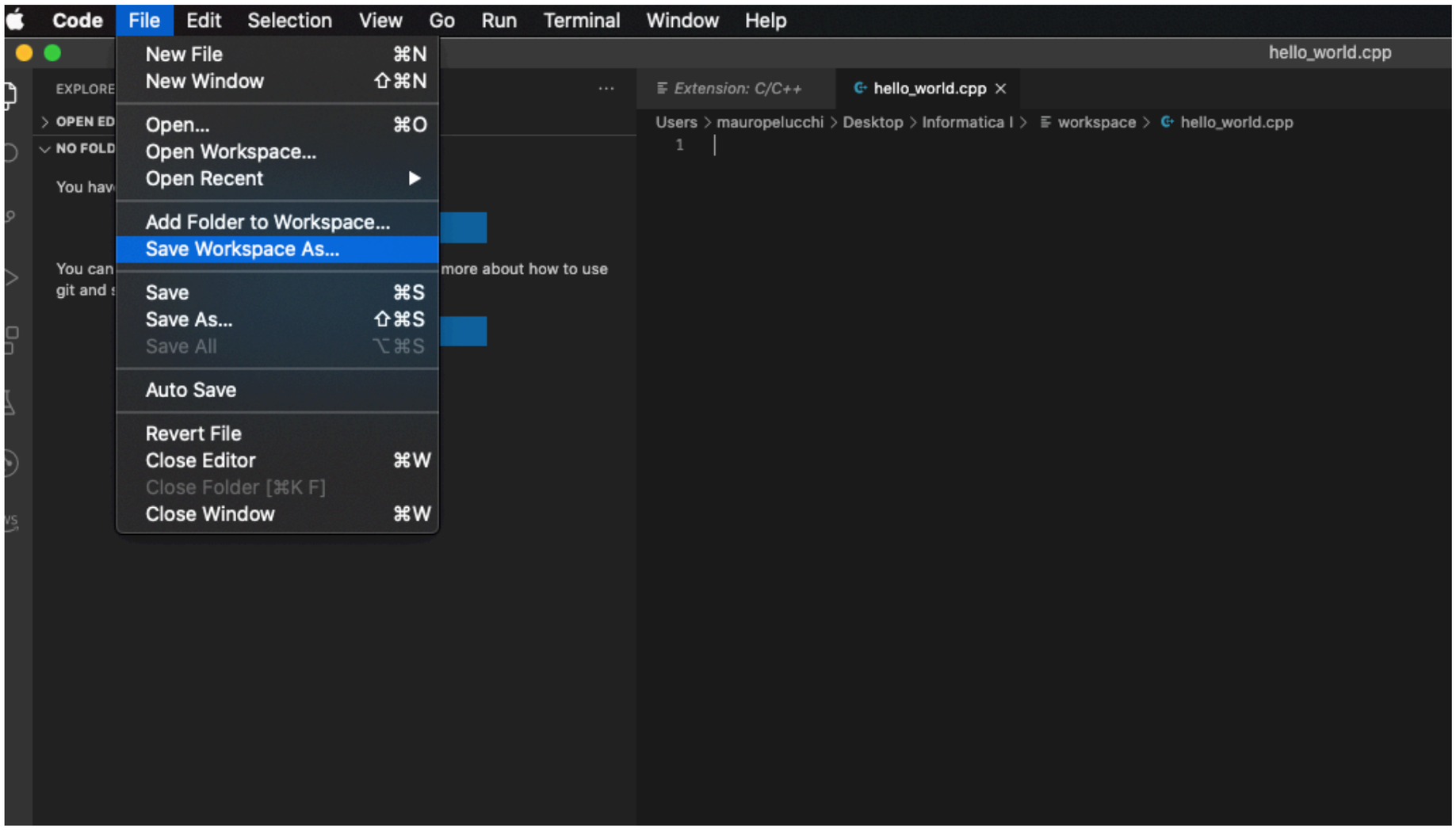
# Visual Studio Code
# Workspace

Salviamo il file nel nostro Workspace come hello_world.cpp

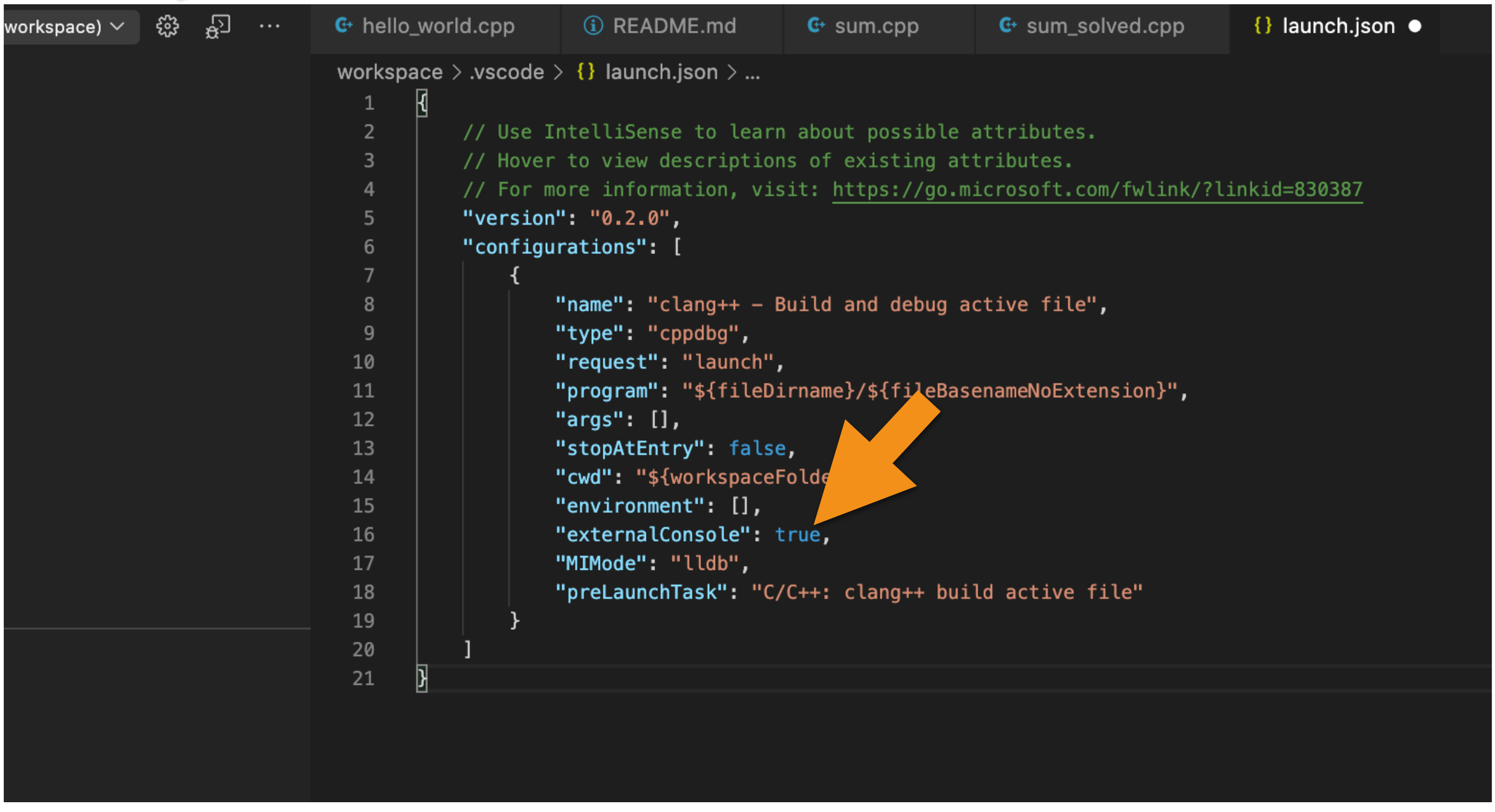


# Visual Studio Code Workspace

Salviamo la nostra workspace



# Crtl + Maiusc + D




https://github.com/microsoft/vscode-cpptools/issues/5079#issuecomment-626090192

# Hello World

```cpp
#include <cstdlib>
#include <iostream>

using namespace std;
/*
  example of multi-line comments
*/

int main(int argc, char *argv[])
{
    int i = 5; // single line comment
    char letter = 'A';

    cout << "Hello World !" << endl;
    cout << "An integer value (i: " << i << ")" << endl;
    cout << "A character value (letter: " << letter << ")" << endl;

    return EXIT_SUCCESS;
}
```

# Comments

- comment starts with // and proceeds to end of line
- comment starts with /* and proceeds to first */

```
// This is an example of a comment.

/* This is another example of a comment. */

/* This is an example of a comment that

spans

multiple lines. */
```

# Note

- Compilare ed eseguire il programma
  - selezionate Run > Start Debugging
  - in assenza di errori Dev-C++ compila il programma producendo l'eseguibile e lo esegue in un terminale
- Il ...main(...) manda in esecuzione il nostro programma

# Introduzione al C++

- Il codice sorgente dei programmi elencati di seguito ha l'obiettivo di riepilogare la struttura generica di un programma C++:
  - primitive base di I/O
  - dichiarazione e inizializzazione di variabili
  - uso di costrutti condizionali e/o iterativi con la relativa sintassi.

- Compilare  e correggere gli eventuali errori presenti nei sorgenti
  - warnings compresi – imparare a distinguere tra errori di sintassi, errori del linker, errori logici

- Si consiglia di leggerli accuratamente prima di trascriverli  con l'editor messo a disposizione dall'IDE per sviluppare le vostre capacità di analisi del codice.

- Le soluzioni mostrate sono solo una traccia e contengono degli errori appositamente inserti.

# Introduzione al C++

| | |
|---|---|
| **sum.cpp** | Esegue e visualizza a video il risultato della somma di 2 numeri inseriti da un utente |
| **sums.cpp** | Esegue la somma di più numeri con terminatore zero (quando l'utente inserisce la cifra 0 "zero" il programma termina e restituisce all'utente la somma) |
| **max.cpp** | Esegue e visualizza a video il risultato del massimo tra 2 numeri inseriti da un utente |
| **maxs.cpp** | Esegue il massimo di più numeri con terminatore zero (quando l'utente inserisce la cifra 0 "zero" il programma termina e restituisce all'utente il valore massimo inserito) |

# Identifiers

- identifiers used to name entities such as
  - types, objects (i.e., variables), and functions
- valid identifier is sequence of one or more letters, digits, and underscore characters that does not begin with a digit
- identifiers that begin with underscore (in many cases) or contain double underscores are reserved for use by C++ implementation and should be avoided
- examples of valid identifiers
  - event_counter
  - eventCounter
  - sqrt_2
- identifiers are case sensitive (e.g., counter and cOuNtEr are distinct identifiers)
- identifiers cannot be any of reserved keywords

# Reserved Keywords

```
alignas
alignof
and
and_eq
asm
auto
bitand
bitor
bool
break
case
catch
char
char16_t
char32_t
class
compl
const
constexpr
const_cast
continue
decltype
default
delete
do
double
dynamic_cast
else
enum
explicit
export
extern
false
float
for
friend
goto
if
inline
int
long
mutable
namespace
new
noexcept
not
not_eq
nullptr
operator
or
or_eq
private
protected
public
register
reinterpret_cast
return
short
signed
sizeof
static
static_assert
static_cast
struct
switch
template
this
thread_local
throw
true
try
typedef
typeid
typename
union
unsigned
using
virtual
void
volatile
wchar_t
while
xor
xor_eq
override*
final*
```

*Note: context sensitive

# Operators

Arithmetic Operators

| Operator Name | Syntax |
|---|---|
| addition | `a + b` |
| subtraction | `a - b` |
| unary plus | `+a` |
| unary minus | `-a` |
| multiplication | `a * b` |
| division | `a / b` |
| modulo (i.e., remainder) | `a % b` |
| pre-increment | `++a` |
| post-increment | `a++` |
| pre-decrement | `--a` |
| post-decrement | `a--` |

Bitwise Operators

| Operator Name | Syntax |
|---|---|
| bitwise NOT | `~a` |
| bitwise AND | `a & b` |
| bitwise OR | `a \| b` |
| bitwise XOR | `a ^ b` |
| arithmetic left shift | `a << b` |
| arithmetic right shift | `a >> b` |

# Operators

Assignment and
Compound-Assignment Operators

| Operator Name | Syntax |
|---|---|
| assignment | `a = b` |
| addition assignment | `a += b` |
| subtraction assignment | `a -= b` |
| multiplication assignment | `a *= b` |
| division assignment | `a /= b` |
| modulo assignment | `a %= b` |
| bitwise AND assignment | `a &= b` |
| bitwise OR assignment | `a \|= b` |
| bitwise XOR assignment | `a ^= b` |
| arithmetic left shift assignment | `a <<= b` |
| arithmetic right shift assignment | `a >>= b` |

# Operators

Logical/Relational Operators

| Operator Name | Syntax |
|---|---|
| equal | `a == b` |
| not equal | `a != b` |
| greater than | `a > b` |
| less than | `a < b` |
| greater than or equal | `a >= b` |
| less than or equal | `a <= b` |
| logical negation | `!a` |
| logical AND | `a && b` |
| logical OR | `a \|\| b` |

Member and Pointer Operators

| Operator Name | Syntax |
|---|---|
| array subscript | `a[b]` |
| indirection | `*a` |
| address of | `&a` |
| member selection | `a.b` |
| member selection | `a->b` |
| member selection | `a.*b` |
| member selection | `a->*b` |

| Alternative | Primary |
|---|---|
| **`and`** | `&&` |
| **`bitor`** | `\|` |
| **`or`** | `\|\|` |
| **`xor`** | `^` |
| **`compl`** | `~` |
| **`bitand`** | `&` |
| **`and_eq`** | `&=` |
| **`or_eq`** | `\|=` |
| **`xor_eq`** | `^=` |
| **`not`** | `!` |
| **`not_eq`** | `!=` |

sum.cpp

```cpp
#include <iostream> // include standard libraries
using namespace std;

void main(int argc, char *argv[])
{
  int num1, num2, sum;
  sum = 0;

  cout << "\nSum of 2 numbers\n\n";
  cout << "Insert 2 numbers" << endl << endl;
  cout << "First number: ";
  cin << num1;
  cout << "\nSecond number: ";
  cin >> num2;
  sum = num1 + num2;
  cout << "\n\nThe result is " << sum << "\n\n";

  std::cout << "press any key to exit...";
  // wait for user to hit enter or another key
}
```

sums.cpp

```
int main(int argc, char *argv[])
{
  int num, sum;

  sum = 0;
  num = 1;
  while (num != 0)
  {
      cout << "\n Digit a number (digit 0 to exit): ";
      cin >> num;
      sum = sum + num;
  }
  cout << "\nThe resuklt is " << sum << endl << endl;

  std::cout << "press any key to exit...";
  // wait for user to hit enter or another key
  return EXIT_SUCCESS;
}
```

max.cpp

```
#include <iostream> // include standard libraries

int main(int argc, char *argv[])
{
  int num1, num2;

  cout << "\nMax from 2 numbers\n\n";
  cout << "Digit the first number \n\n";
  count << "First number: ";
  cin >> num1;
  cout << "Second number: "
  cin >> num2;
  while (num1 < num2);
     cout << "\n" << num1 << " is greater than " << num2 << "\n\n";
  else {
     cout << "\n" << num1 << " is less or equal than;
     cout << num2 << "\n\n";
  }

  std::cout << "press any key to exit...";
  // wait for user to hit enter or another key
  return EXIT_SUCCESS;
}
```

maxs.cpp

```cpp
#include <iostream> // include standard libraries

int main(int argc, char *argv[])
{
  int num, max;

  max = 0;
  num = 1;
  while (num != 0);
  {
    std::cout << "Digit a number (digit 0 to exit):";
    std::cin >> num;
    if (num > max) max = num;
}
std::cout << "\nThe max is " << max << "\n";

std::cout << "press any key to exit...";
// wait for user to hit enter or another key
return EXIT_SUCCESS;
```

# If Statement

- allows conditional execution of code
- syntax has form:

```
if (expression)
    statement1
else
    statement2
```

- if expression *expression* is true, execute statement $statement_1$; otherwise, execute statement $statement_2$
- **`else`** clause can be omitted leading to simpler form:

```
if (expression)
    statement1
```

- conditional execution based on more than one condition can be achieved using construct like:

```
if (expression1)
    statement1
else if (expression2)
    statement2
...
else
    statementn
```

# While Statement

- looping construct
- syntax has form:

```
while (expression)
    statement
```

- if expression *expression* is true, statement *statement* is executed; this process repeats until expression *expression* becomes false
- to allow multiple statements to be executed in loop body, ***must group multiple statements*** into single statement with brace brackets

```
while (expression) {
    statement1
    statement2
    statement3
    ...
}
```

- advisable to ***always use brace brackets***, even when loop body consists of only one statement

# Esercizi introduttivi di programmazione

# Esercizio 1
# Costrutti condizionali e Casting delle variabili

- Realizzare un programma che, dato in ingresso un angolo specificato in gradi come un numero intero, fornisca la relativa conversione in radianti. L'angolo deve essere compreso tra 0 e 360 gradi, altrimenti il programma stampa un messaggio di errore e  termina.

# Esercizio 2
# Costrutti condizionali e Algoritmi

- Scrivere un programma che risolva il seguente problema.

- Letti tre numeri interi a, b, c dallo standard input, stampare a terminale la sequenza dei tre numeri in ordine non decrescente.

- Esempio: a = 10, b = 7, c = 9
  - deve dare in uscita -> 7, 9, 10

# Esercizio 2
# Costrutti condizionali e Algoritmi