

FEEL THE MEANING OF THE TRIP

青 / 春 / 不 / 老 / 梦 / 想 / 永 / 在

DREAM
MY DREAM WILL NEVER STOP

High-Level Language Programming

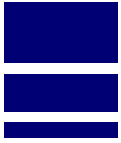
Experiment 1: Lab Environment and Basic Programming

GO!
TAKE YOU ON A TRIP



哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

探索 从未停止

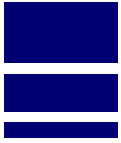


IDE- Code::Blocks

- ❑ Download link (latest version 25.03):
<http://www.codeblocks.org/downloads/>
- ❑ Single installation package, easy to install
- ❑ Free, open-source, and cross-platform
- ❑ Supports multiple compilers: GCC
- ❑ Powerful debugging features: full GDB support
- ❑ Intelligent code editor

Microsoft Windows (64 bit, default)

File	Download from
codeblocks-25.03-setup.exe	Sourceforge.net or dAppCDN.com
codeblocks-25.03-setup-nonadmin.exe	Sourceforge.net or dAppCDN.com
codeblocks-25.03-nosetup.zip	Sourceforge.net or dAppCDN.com
codeblocks-25.03mingw-setup.exe	Sourceforge.net or dAppCDN.com
codeblocks-25.03mingw-nosetup.zip	Sourceforge.net or dAppCDN.com



Code::Blocks

1. Create a New Project

2. Edit Code

3. Compile

4. Run

5. Open an Existing Project

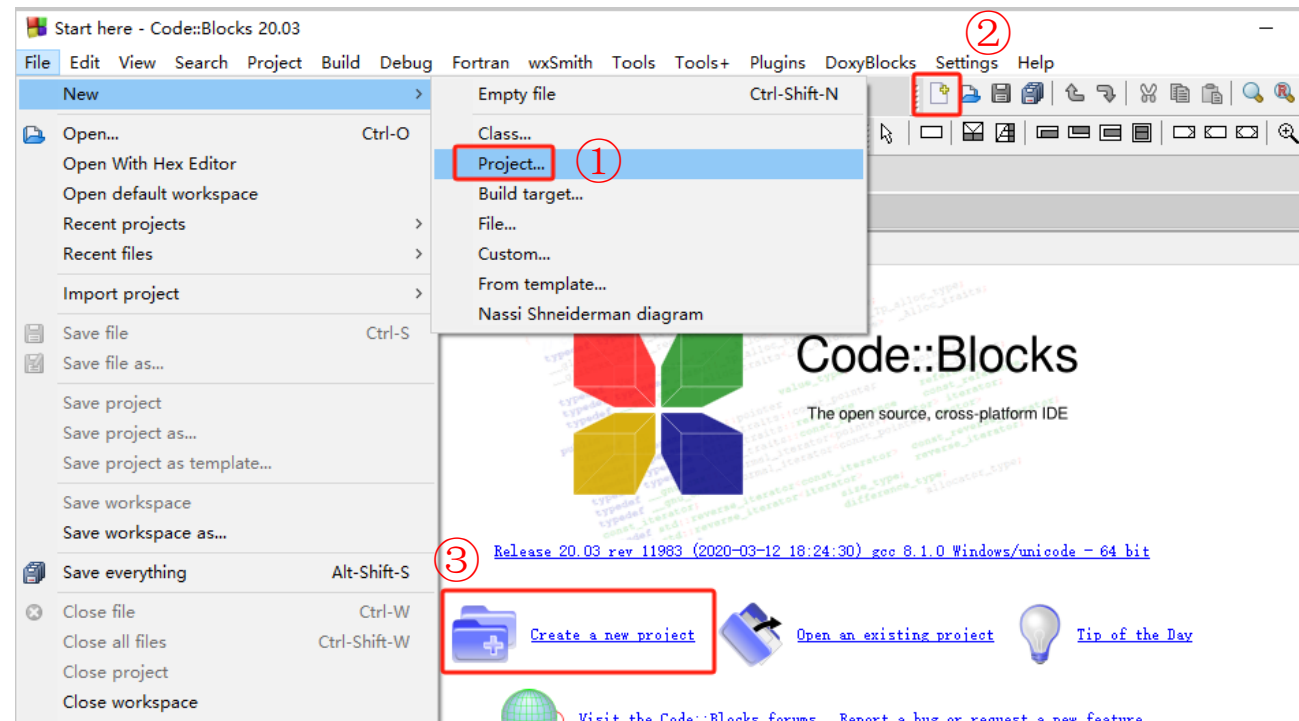
6. Activate Project

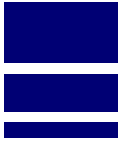
❑ Menu Bar: "File→New→Project "

❑ Tool bar: Project...



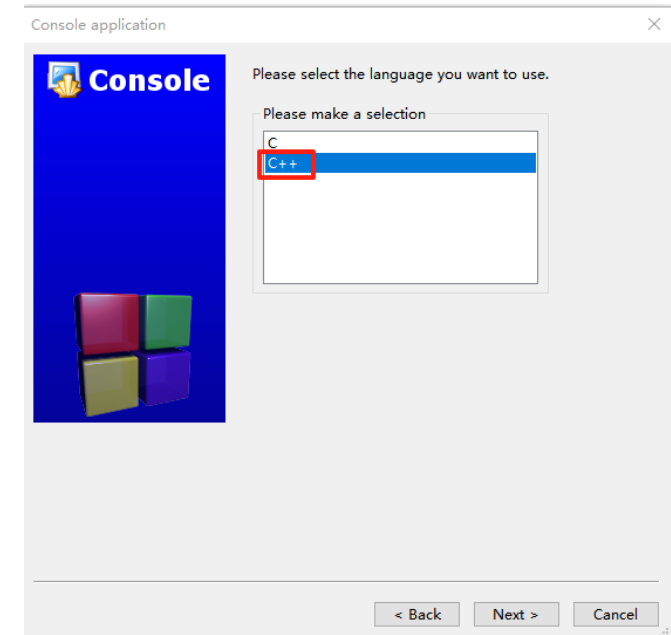
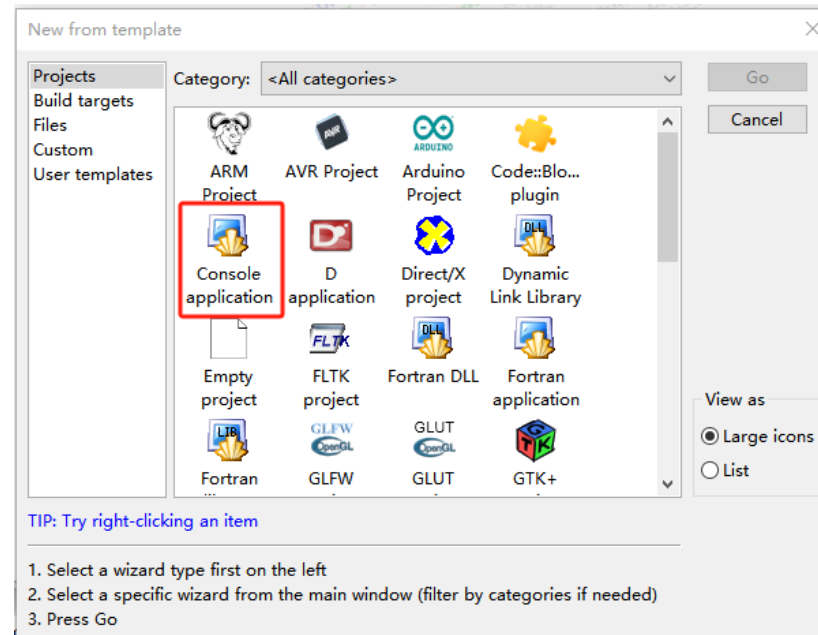
❑ Home Screen: "Create a new project"

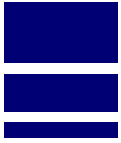




1. Create a New Project
2. Edit Code
3. Compile
4. Run
5. Open an Existing Project
6. Activate Project

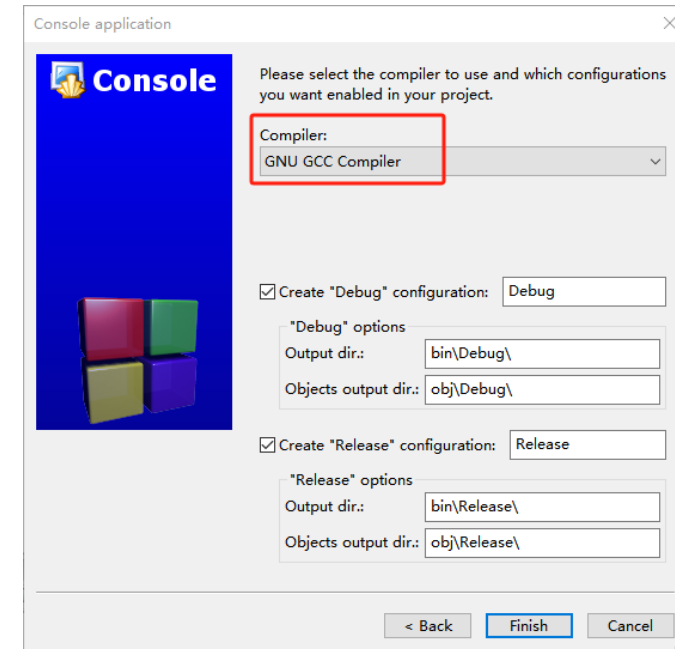
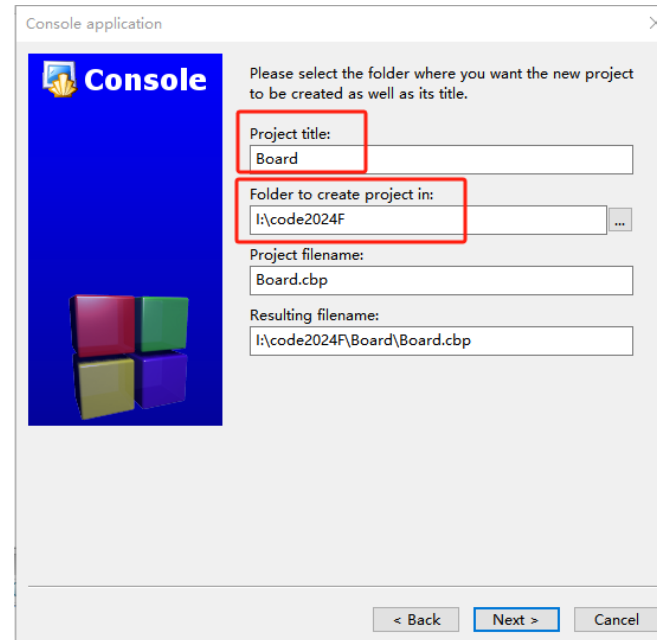
- ❑ Select "Concole application" ;
- ❑ Select "C++", and the program will run in the C++ compilation environment.;



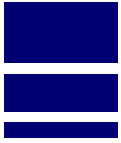


1. Create a New Project
2. Edit Code
3. Compile
4. Run
5. Open an Existing Project
6. Activate Project

- ❑ Fill in the project name and select the project storage location;
- ❑ Choose the default GCC compiler, and click OK



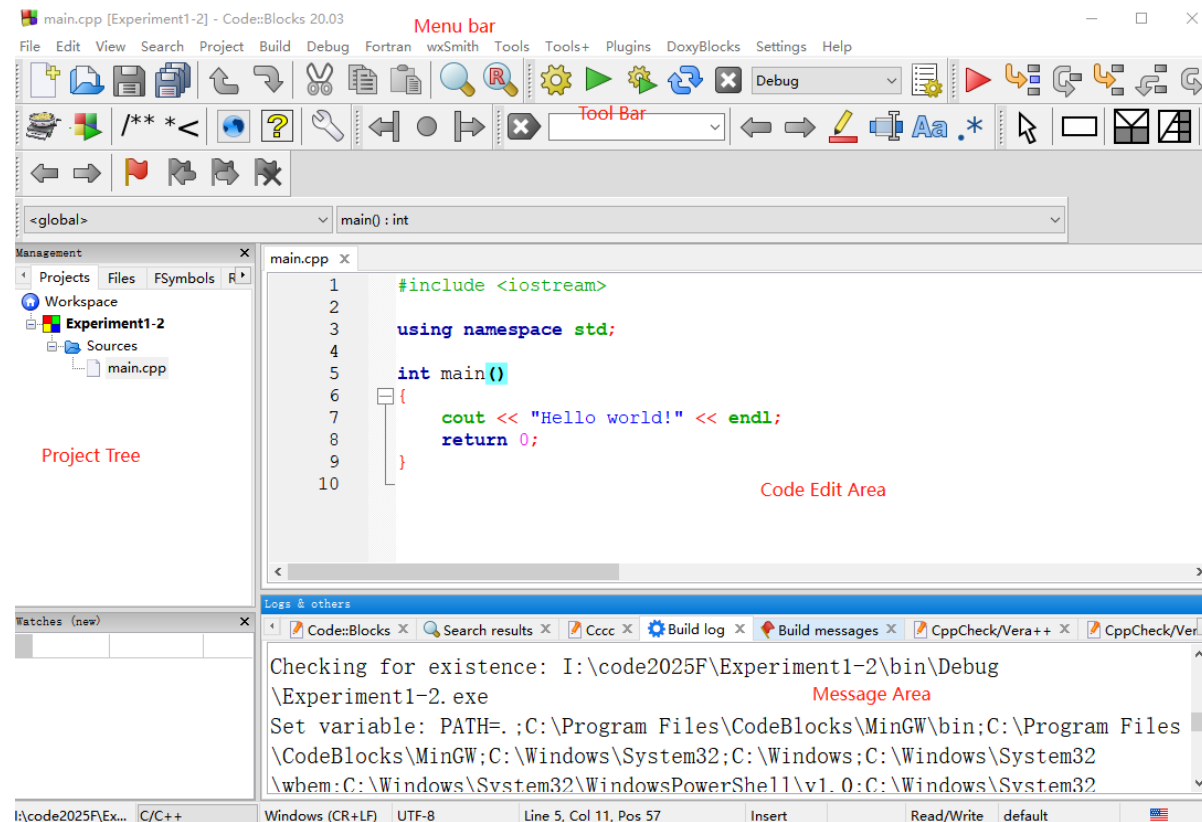
The project name and save path should not contain spaces.

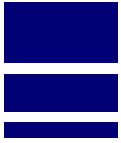


Code::Blocks

1. Create a New Project
2. Edit Code
3. Compile
4. Run
5. Open an Existing Project
6. Activate Project

- ❑ Double-click the main.cpp file in the Sources directory of the project, and edit the code in the code editor on the right.
- ❑ Click the Save button, or use the shortcut Ctrl+S to save.





Code::Blocks

1. Create a New Project


2. Edit Code

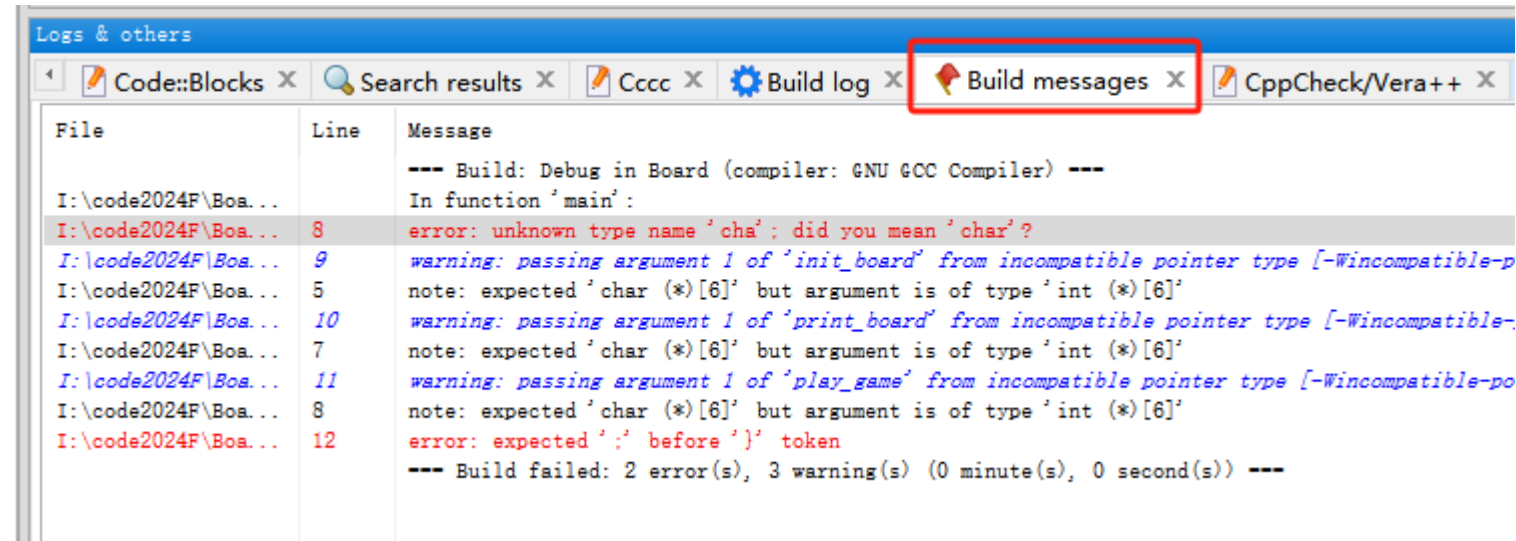
3. Compile

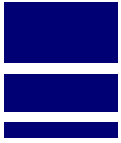
4. Run

5. Open an Existing Project

6. Activate Project

- ❑ Tool bar: 
- ❑ Menu Bar: Build→Build
- ❑ Shortcut Key: Ctrl + F9





1. Create a New Project


2. Edit Code

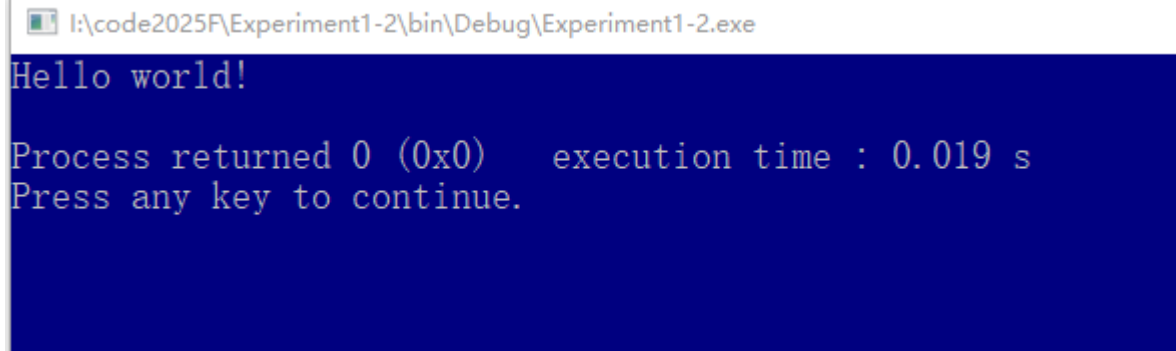
3. Compile

4. Run

5. Open an Existing Project

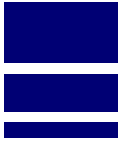
6. Activate Project

- ❑ Tool bar: 
- ❑ Menu bar : Build→Run
- ❑ Shortcut Key: Ctrl + F10



```
I:\code2025F\Experiment1-2\bin\Debug\Experiment1-2.exe
Hello world!


Process returned 0 (0x0)   execution time : 0.019 s
Press any key to continue.
```

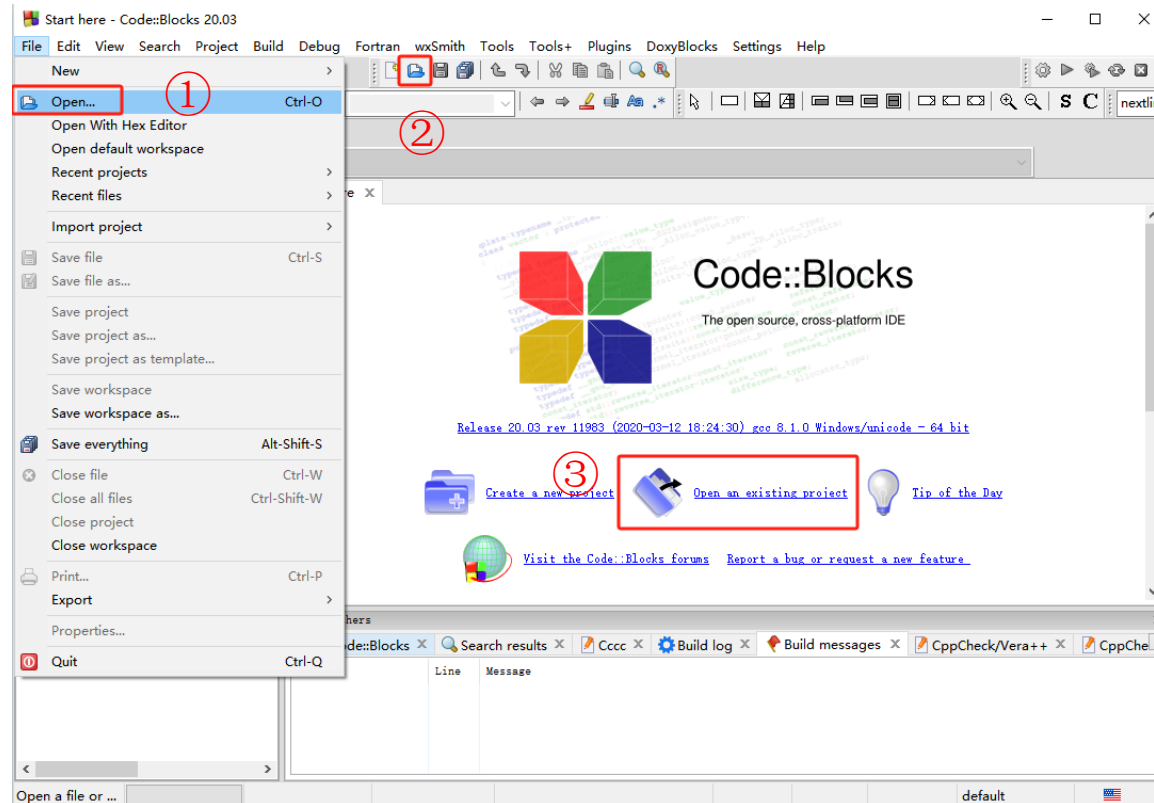
Code::Blocks

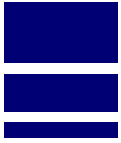
1. Create a New Project
2. Edit Code
3. Compile
4. Run
5. Open an Existing Project
6. Activate Project

❑ Menu bar: "File→Open→Project "

❑ Tool bar: 

❑ Home Screen: "Open an existing project"





1. Create a New Project

2. Edit Code

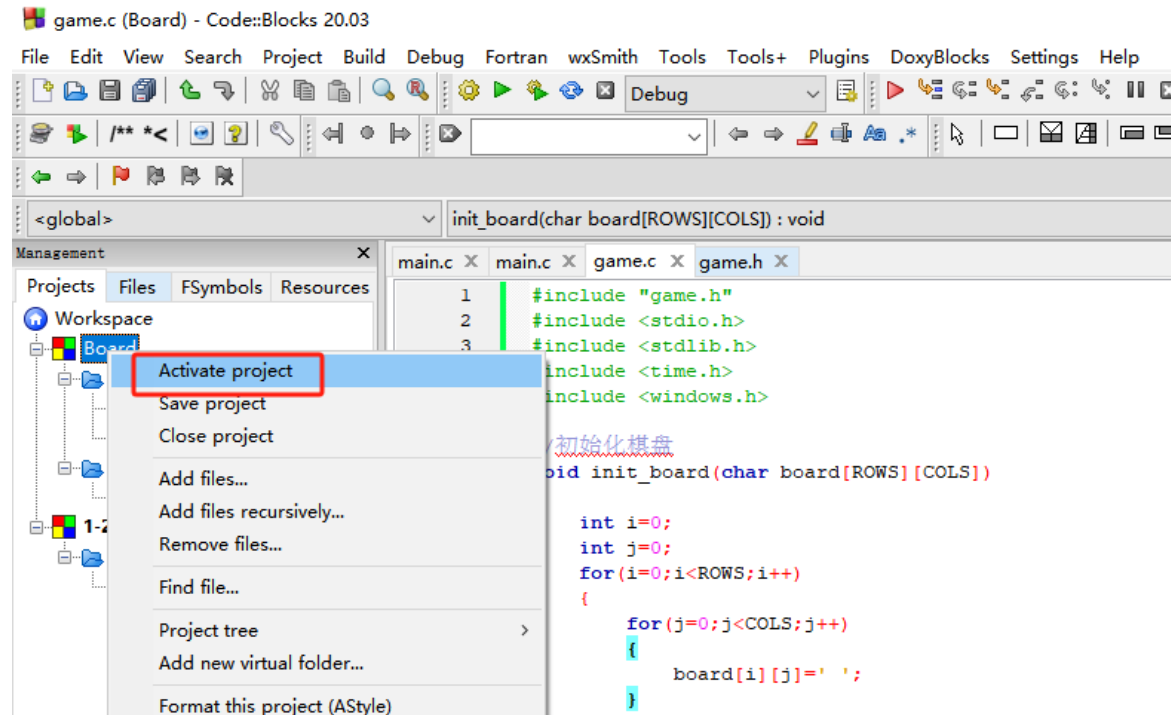
3. Compile

4. Run

5. Open an Existing Project

6. Activate Project

- ❑ Right-click and select "Activate Project"
- ❑ Activated projects are displayed in bold black text.
- ❑ You may open multiple projects simultaneously; however, only a single project can remain active.





How Programs Run

1. Preprocessing

2. Compilation

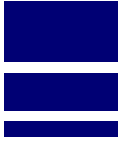
3. Assembly

4. Linking

5. Execution

Preprocessing: In this step, the preprocessor handles the directives in the source code that begin with #, such as `#include` and `#define`. The preprocessor can also perform conditional compilation, including or excluding certain parts of the code during the compilation process.

- ❑ Macro definition: `#define`
- ❑ File inclusion: `#include`
- ❑ Conditional compilation



How Programs Run

1. Preprocessing

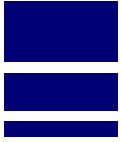
2. Compilation

3. Assembly

4. Linking

5. Execution

Compilation: In this step, the compiler translates the preprocessed source code into assembly language. The compiler performs lexical analysis, syntax analysis, and semantic analysis, and then generates the corresponding intermediate code.



How Programs Run

1. Preprocessing

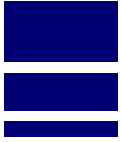
2. Compilation

3. **Assembly**

4. Linking

5. Execution

Assembly: In this step, the assembler converts the intermediate code into executable machine code. The assembler translates each assembly instruction into one or more machine instructions and generates an object file.



How Programs Run

1. Preprocessing

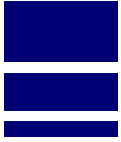
2. Compilation

3. Assembly

4. Linking

5. Execution

Linking: In this step, the linker combines object files and library files to generate an executable file. The linker resolves symbol references, places code and data segments at the correct locations in memory, and produces the executable file's metadata (such as the entry point address and symbol table).



How Programs Run

1. Preprocessing

2. Compilation

3. Assembly

4. Linking

5. Execution

Execution: The operating system loads the executable file into memory, locates the program's entry point (usually the address of the main function), and then begins execution.



Experimental Task

- 1. Implement the “Hello World” program.**
- 2. The definition and assignment of different types of variables.**
 - (1) Create a string variable, assign a string value to it.**
 - (2) Create an int variable, assign an integer value to it.**
 - (3) Create a float variable, assign a float value to it.**
 - (4) Print out the string, the int, and the float variables together.**

Experimental Environment:

Operating System: Windows

IDE: Code::Blocks
