CS100 Recitation 1

GKxx

Februrary 21, 2022

Contents

Editors, Compilers and IDEs

- A compiler translates the program written in a high-level language so that the computer can run it.
    - `GCC`, `Clang`, `Visual C++` compiler, ...

Editors, Compilers and IDEs

- A compiler translates the program written in a high-level language so that the computer can run it.
  - `GCC`, `Clang`, `Visual C++` compiler, ...
- An editor is something where you can edit text.
  - `Notepad`, `Word`, and even your phone memo.

Editors, Compilers and IDEs

- A compiler translates the program written in a high-level language so that the computer can run it.
  - `GCC`, `Clang`, `Visual C++` compiler, . . .
- An editor is something where you can edit text.
  - `Notepad`, `Word`, and even your phone memo.
  - But we need a code editor which provides more help for coding.
  - `Visual Studio Code`, `Vim`, `Sublime Text`, `Notepad++`, . . .

Editors, Compilers and IDEs

- A compiler translates the program written in a high-level language so that the computer can run it.
  - `GCC`, `Clang`, `Visual C++` compiler, ...
- An editor is something where you can edit text.
  - `Notepad`, `Word`, and even your phone memo.
  - But we need a code editor which provides more help for coding.
  - `Visual Studio Code`, `Vim`, `Sublime Text`, `Notepad++`, ...
- IDE: **I**ntegrated **D**evelopment **E**nvironment,
  - $=$ editor $+$ compilers $+$ debuggers $+ \cdots$.
  - `Visual Studio`, `Qt`, `CLion`, `Dev-C++`, ...

Contents

GCC and MinGW

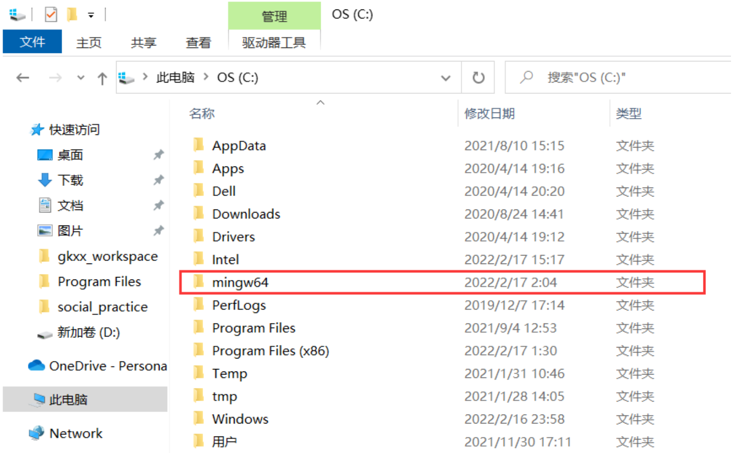
- GCC is the **G**NU **C**ompiler **C**ollection, an optimizing compiler produced by the GNU Project supporting various programming languages, hardware architectures and operating systems.
- MinGW is short for **Min**imalist **G**NU for **W**indows.

GCC and MinGW

- GCC is the **G**NU **C**ompiler **C**ollection, an optimizing compiler produced by the GNU Project supporting various programming languages, hardware architectures and operating systems.
- MinGW is short for **Min**imalist **G**NU for **W**indows.
- For Linux, install GCC directly is ok.
- For Windows, you may need MinGW (or, probably MinGW-w64).

MinGW

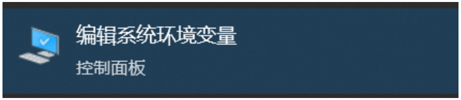
- Download the package provided in the Resources page.
- Unzip it and place the `mingw64` folder in the `C` drive.



名称	修改日期	类型
AppData	2021/8/10 15:15	文件夹
Apps	2020/4/14 19:16	文件夹
Dell	2020/4/14 20:20	文件夹
Downloads	2020/8/24 14:41	文件夹
Drivers	2020/4/14 19:12	文件夹
Intel	2022/2/17 15:17	文件夹
mingw64	2022/2/17 2:04	文件夹
PerfLogs	2019/12/7 17:14	文件夹
Program Files	2021/9/4 12:53	文件夹
Program Files (x86)	2022/2/17 1:30	文件夹
Temp	2021/1/31 10:46	文件夹
tmp	2021/1/28 14:05	文件夹
Windows	2022/2/16 23:58	文件夹
用户	2021/11/30 17:11	文件夹

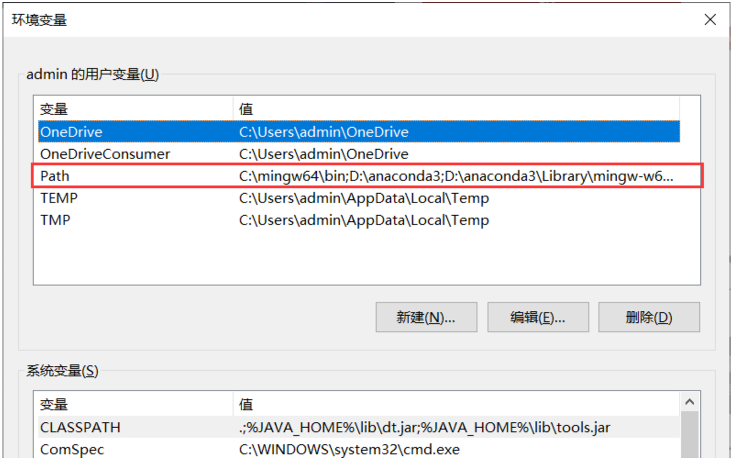
MinGW

- Now the compiler is installed, but it could not be invoked conveniently. We need to add it to the `Path` environment variable.
- Press `Win` and search 'env'. Choose 'Edit the system environment variables'.



- Click the 'Environment variables ...' button.

MinGW


环境变量

admin 的用户变量(U)

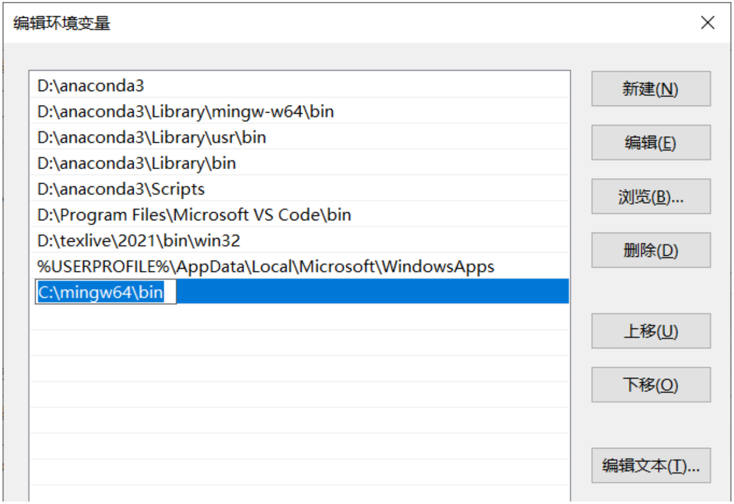
变量	值
OneDrive	C:\Users\admin\OneDrive
OneDriveConsumer	C:\Users\admin\OneDrive
Path	C:\mingw64\bin;D:\anaconda3;D:\anaconda3\Library\mingw-w6...
TEMP	C:\Users\admin\AppData\Local\Temp
TMP	C:\Users\admin\AppData\Local\Temp

系统变量(S)

变量	值
CLASSPATH	.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar
ComSpec	C:\WINDOWS\system32\cmd.exe

MinGW

- Add a new value ‘`C:\mingw64\bin`’.

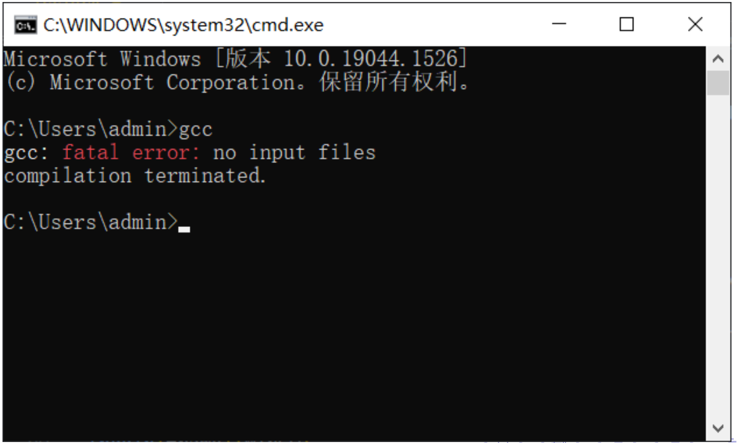


编辑环境变量

```
D:\anaconda3
D:\anaconda3\Library\mingw-w64\bin
D:\anaconda3\Library\usr\bin
D:\anaconda3\Library\bin
D:\anaconda3\Scripts
D:\Program Files\Microsoft VS Code\bin
D:\texlive\2021\bin\win32
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\mingw64\bin
```

MinGW

- Press `Win+r` to open a cmd.
- Type ‘gcc’ and press `Enter`. The following shows that gcc is correctly invoked.



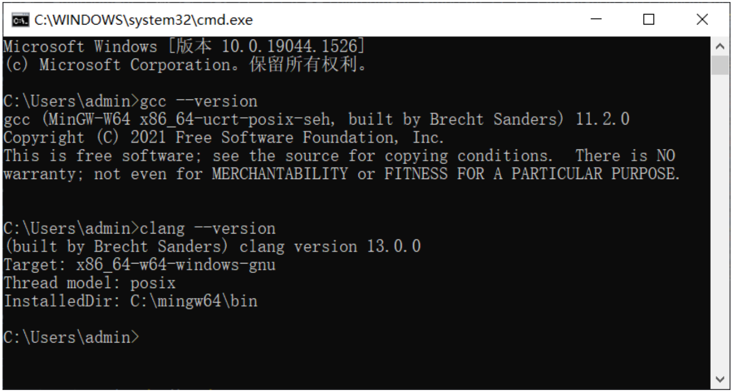
```
Microsoft Windows [版本 10.0.19044.1526]
(c) Microsoft Corporation。保留所有权利。

C:\Users\admin>gcc
gcc: fatal error: no input files
compilation terminated.

C:\Users\admin>_
```

MinGW

You can use '`--version`' to see more information about the compilers.



```
Microsoft Windows [版本 10.0.19044.1526]
(c) Microsoft Corporation。保留所有权利。

C:\Users\admin>gcc --version
gcc (MinGW-W64 x86_64-ucrt-posix-seh, built by Brecht Sanders) 11.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


C:\Users\admin>clang --version
(built by Brecht Sanders) clang version 13.0.0
Target: x86_64-w64-windows-gnu
Thread model: posix
InstalledDir: C:\mingw64\bin

C:\Users\admin>
```

For Linux (Ubuntu)

- ‘`sudo apt install build-essential`’ gets everything done.
- If you want compilers of newer versions:
  ```
  sudo add-apt-repository ppa:ubuntu-toolchain-r/test
  sudo apt update
  sudo apt install gcc-11
  ```
- You can search for more on your own.

For Mac OS X

Step #1: Install Xcode on a Apple Mac OS X

First, make sure Xcode is installed. If it is not installed on OS X, visit app store and install Xcode.

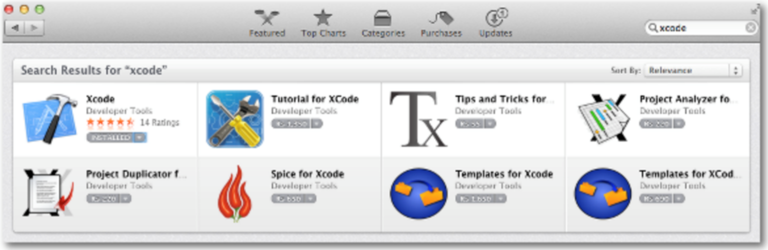


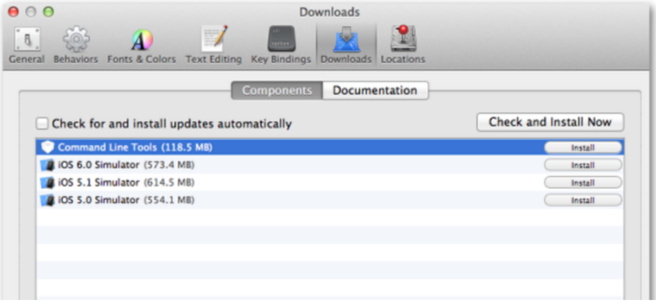
Fig.01: Make sure Xcode developer tools are install OS X

For Mac OS X

Step #2: Install gcc/LLVM compiler on OS X

Once installed, open Xcode and visit:

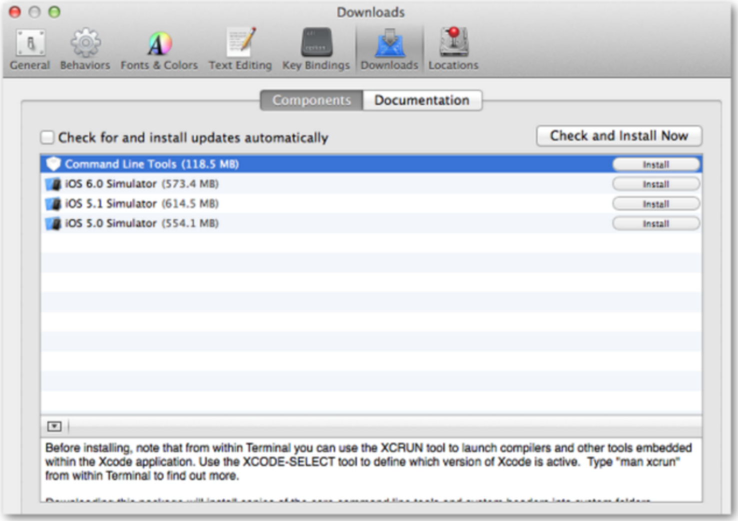
```
Xcode menu > Preferences > Downloads > choose "Command line tools" > Click
"Install" button:
```



☐ Check for and install updates automatically

Command Line Tools (118.5 MB)
iOS 6.0 Simulator (573.4 MB)
iOS 5.1 Simulator (614.5 MB)
iOS 5.0 Simulator (554.1 MB)

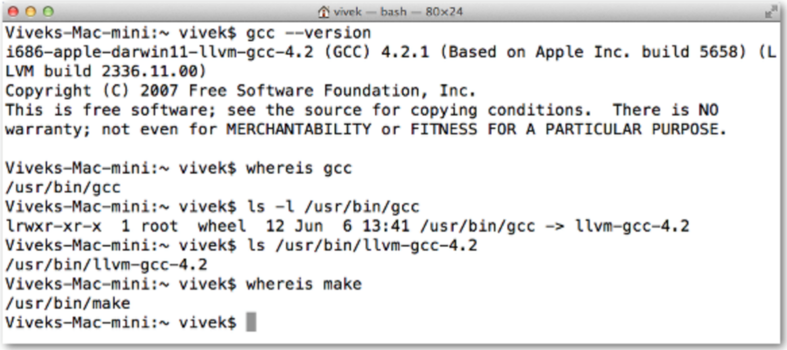
For Mac OS X


Downloads
General
Behaviors
Fonts & Colors
Text Editing
Key Bindings
Downloads
Locations
Components
Documentation
Check for and install updates automatically
Check and Install Now
Command Line Tools (118.5 MB) Install
iOS 6.0 Simulator (573.4 MB) Install
iOS 5.1 Simulator (614.5 MB) Install
iOS 5.0 Simulator (554.1 MB) Install

Before installing, note that from within Terminal you can use the XCRUN tool to launch compilers and other tools embedded within the Xcode application. Use the XCODE-SELECT tool to define which version of Xcode is active. Type "man xcrun" from within Terminal to find out more.

For Mac OS X

- Verify that it is working: '`gcc --version`'



```
Viveks-Mac-mini:~ vivek$ gcc --version
i686-apple-darwin11-llvm-gcc-4.2 (GCC) 4.2.1 (Based on Apple Inc. build 5658) (L
LVM build 2336.11.00)
Copyright (C) 2007 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Viveks-Mac-mini:~ vivek$ whereis gcc
/usr/bin/gcc
Viveks-Mac-mini:~ vivek$ ls -l /usr/bin/gcc
lrwxr-xr-x  1 root  wheel  12 Jun  6 13:41 /usr/bin/gcc -> llvm-gcc-4.2
Viveks-Mac-mini:~ vivek$ ls /usr/bin/llvm-gcc-4.2
/usr/bin/llvm-gcc-4.2
Viveks-Mac-mini:~ vivek$ whereis make
/usr/bin/make
Viveks-Mac-mini:~ vivek$ 
```

Fig.03: Verify gcc compiler installation on Mountain Lion OS X

Contents

Installation

- Install `VSCode` from `code.visualstudio.com`.
- For Linux users, **DO NOT** install it via `snap` or you may encounter trouble.

Installation

- Install `VSCode` from `code.visualstudio.com`.
- For Linux users, **DO NOT** install it via `snap` or you may encounter trouble.
- Run the installer. It is recommended to install it in the `D` or `E` drive, e.g. `D:\Program Files\Microsoft VS Code\`.

Extensions

Recommended extensions:

- `Code Runner`, `C/C++`, `C++ Intellisense`.
- `Bracket Pair Colorization Toggler`, `vscode-icons`.
- `One Dark Pro` and `GitHub Theme`: color themes.
- ~~`GlassIt-VSC`, `Cloudmusic`, `QQ`, `Zhihu On VSCode`, ...~~

You may also need `Chinese (Simplified) Language Pack for Visual Studio Code`.

Configuration

- Create a folder for CS100, e.g. `D:\CS100`. This will be viewed as a workspace.

Configuration

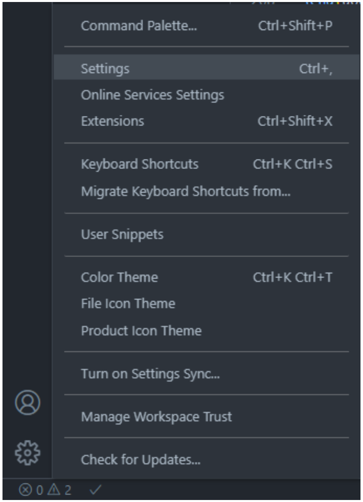
- Create a folder for CS100, e.g. `D:\CS100`. This will be viewed as a workspace.
- `VSCode` has '$n+1$' configurations, where n is the number of workspaces and the '$+1$' refers to the global (user's) one.
- The configuration of each workspace is done by some `json` files in a special folder `.vscode`.

Configuration

- Create a folder for CS100, e.g. `D:\CS100`. This will be viewed as a workspace.
- `VSCode` has '$n + 1$' configurations, where n is the number of workspaces and the '$+1$' refers to the global (user's) one.
- The configuration of each workspace is done by some `json` files in a special folder `.vscode`.
- Remember to always open `VSCode` first and then open the workspace, instead of open a single file directly. Otherwise your configuration for workspace wouldn't work.

Configuration

Global settings:



Configuration

- Code-runner: Save File Before Run true
- Code-runner: Run In Terminal true
- Code-runner: Ignore Selection true
- Editor: Format On Type true
- Editor: Accept Suggestion On Enter off

Configuration

- Create a folder `D:\CS100\.vscode` for your workspace configurations.
- Create two files `settings.json` and `c_cpp_properties.json`. Copy the contents from `https://www.luogu.com.cn/paste/scc7i5yq`.

Configuration

- Create a folder `D:\CS100\.vscode` for your workspace configurations.
- Create two files `settings.json` and `c_cpp_properties.json`. Copy the contents from `https://www.luogu.com.cn/paste/scc7i5yq`.
- Create a hello-world program somewhere in this workspace, e.g. `D:\CS100\tmp\hello.c`.
- There will be a ‘Run Code’ button on the top-right corner. Or you can press `Ctrl+Alt+N` to run the code.

Configuration

- Pressing this button, the `Code Runner` extension runs the command we wrote in `"code-runner.executorMap"` in `settings.json`.
- It is run in the terminal of `VSCode`, which is the same as in cmd.
- The `Code Runner` extension gets you free from typing the same compilation command manually over and over again. (You may have a try of typing it manually.)

Configuration

For the debugging part:

- Print statement debugging is effective, although `VSCode` says that it is 'a thing of the past'.
- To use the tools for debugging in `VSCode`, press `F5`.
- Choose 'GDB/LLDB', and then choose 'gcc'. If you wish to use the LLVM debuggers, you need to install 'lldb-mi' on your own.
- Wait a second and the default configuration files for debugging (`launch.json` and `tasks.json`) are generated automatically.

Configuration

For the debugging part:

- Print statement debugging is effective, although `VSCode` says that it is 'a thing of the past'.
- To use the tools for debugging in `VSCode`, press `F5`.
- Choose 'GDB/LLDB', and then choose 'gcc'. If you wish to use the LLVM debuggers, you need to install 'lldb-mi' on your own.
- Wait a second and the default configuration files for debugging (`launch.json` and `tasks.json`) are generated automatically.

⇒ An example: the "A+B" problem.

Where Do I Learn Things?

- More about `VSCode`, you can visit the official website `code.visualstudio.com`.
- **You should get used to reading official documentations**, not only for `VSCode`, but also for most programming languages and tools.

Where Do I Learn Things?

- More about `VSCode`, you can visit the official website `code.visualstudio.com`.
- **You should get used to reading official documentations**, not only for `VSCode`, but also for most programming languages and tools.
- The official documentation for `C/C++` is not suitable for newcomers. We recommend `cppreference.com`.

Where Do I Learn Things?

Apart from the course and slides, we can learn things from:

- `stackoverflow.com`, mostly for bug-fixing and trouble-shooting. (Also `stackexchange.com`)
- `cppreference.com` and **authoritative** textbooks like *C++ Primer*. One may use them as a dictionary.
- books like *Effective C++*, which helps you solve common problems and develop good coding habits.

Where Do I Learn Things?

The following websites do offer some help, but are not recommended:

- Wikipedia and Baidu Baike: Everyone can edit, and some contents are checked by experts.
- Zhihu, CSDN, Luogu, and some other blogs. Everyone can edit and no one checks.
- Baidu Zhidao, Baidu Jingyan, Xiao Hongshu: No experts would be willing to write things there!

Contents

Language Standards

- Standards of C: C89/90, C99, C11, C17, C23 (coming soon).
- Standards of C++: C++98/03, C++11, C++14, C++17, C++20, C++23 (coming soon),. . .

Language Standards

- Standards of C: C89/90, C99, C11, C17, C23 (coming soon).
- Standards of C++: C++98/03, C++11, C++14, C++17, C++20, C++23 (coming soon),...
- A new version of standard C++ comes out every three years.

Language Standards

- Standards of C: C89/90, C99, C11, C17, C23 (coming soon).
- Standards of C++: C++98/03, C++11, C++14, C++17, C++20, C++23 (coming soon),...
- A new version of standard C++ comes out every three years.
- To specify a standard for the compiler, use `-std=c`*x* or `-std=c++`*y*, e.g. `-std=c11`, `-std=c++17`.

Language Standards

- Standards of C: C89/90, C99, C11, C17, C23 (coming soon).
- Standards of C++: C++98/03, C++11, C++14, C++17, C++20, C++23 (coming soon),...
- A new version of standard C++ comes out every three years.
- To specify a standard for the compiler, use `-std=c`*x* or `-std=c++`*y*, e.g. `-std=c11`, `-std=c++17`.
- To see what language standard the compiler is using, check the macro `__STDC_VERSION__` in C and `__cplusplus` in C++. For example, `__cplusplus == 201703L` means that the program is compiled under C++17.

Contents

Integer Types

- `short (int)`, `signed short (int)`, `unsigned short (int)`
- `int`, `signed int`, `unsigned int`
- `long (int)`, `signed long (int)`, `unsigned long (int)`
- `long long (int)`, `signed long long (int)`, `unsigned long long (int)` (since C99)

Integer Types

- What's the size of a `short`? `int`? `long`? `long long`?

Integer Types

- What's the size of a `short`? `int`? `long`? `long long`?
  `short` and `int` are at least 16-bit. `long` is at least 32-bit. `long long` is at least 64-bit.
  `1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) <= sizeof(long long)`

Integer Types

- What's the size of a `short`? `int`? `long`? `long long`?
  `short` and `int` are at least 16-bit. `long` is at least 32-bit. `long long` is at least 64-bit.
  `1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) <= sizeof(long long)`
- Do `int` and `signed int` name the same type? What about others?

Integer Types

- What's the size of a `short`? `int`? `long`? `long long`?
  `short` and `int` are at least 16-bit. `long` is at least 32-bit. `long long` is at least 64-bit.
  `1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) <= sizeof(long long)`
- Do `int` and `signed int` name the same type? What about others?
  For any integer type `T`, `T` and `signed T` name the same type.

Integer Types

Interesting fact

As with all the type specifiers, any order is permitted: `unsigned long long int` and `long int unsigned long` name the same type.

Integer Types

Interesting fact

As with all the type specifiers, any order is permitted: `unsigned long long int` and `long int unsigned long` name the same type.

- For the exact choices made by each implementation about the sizes of the integer types, you may refer to `https://en.cppreference.com/w/c/language/arithmetic_types`.

Integer Types

Interesting fact

As with all the type specifiers, any order is permitted: `unsigned long long int` and `long int unsigned long` name the same type.

- For the exact choices made by each implementation about the sizes of the integer types, you may refer to `https://en.cppreference.com/w/c/language/arithmetic_types`.
- Exact-width integer types like `int32_t` are defined in `stdint.h` since C99.

Boolean Type

The boolean type in C is **different** than that in C++.

- The type `bool` (same as `_Bool`) is defined since C99, in the header `stdbool.h`.

Boolean Type

The boolean type in C is **different** than that in C++.

- The type `bool` (same as `_Bool`) is defined since C99, in the header `stdbool.h`.
- Type `bool` holds two possible values: `true` and `false`.
- `true` and `false` are `#defined` as `1` and `0` respectively (until C23), so they have type `int` instead of `bool`. Since C23, their type will become `bool`.

Boolean Type

The boolean type in C is **different** than that in C++.

- The type `bool` (same as `_Bool`) is defined since C99, in the header `stdbool.h`.
- Type `bool` holds two possible values: `true` and `false`.
- `true` and `false` are `#define`d as `1` and `0` respectively (until C23), so they have type `int` instead of `bool`. Since C23, their type will become `bool`.
- How does the conversion between `bool` and integer types behave?

Boolean Type

The boolean type in C is **different** than that in C++.

- The type `bool` (same as `_Bool`) is defined since C99, in the header `stdbool.h`.
- Type `bool` holds two possible values: `true` and `false`.
- `true` and `false` are `#defined` as `1` and `0` respectively (until C23), so they have type `int` instead of `bool`. Since C23, their type will become `bool`.
- How does the conversion between `bool` and integer types behave?
  Nonzero $\Rightarrow$ `true`, zero $\Rightarrow$ `false`.
  `true` $\Rightarrow$ 1, `false` $\Rightarrow$ 0.

Character Types

- `char`, `signed char`, `unsigned char`
- Other types for wide characters: `wchar_t`, `char16_t`, `char32_t`.

Character Types

- `char`, `signed char`, `unsigned char`
- Other types for wide characters: `wchar_t`, `char16_t`, `char32_t`.
- Do `char` and `signed char` name the same type?

Character Types

- `char`, `signed char`, `unsigned char`
- Other types for wide characters: `wchar_t`, `char16_t`, `char32_t`.
- Do `char` and `signed char` name the same type?
  NO. The type `char` is neither `signed char` nor `unsigned char`. Whether `char` is signed depends on the implementation, but it is a **distinct type** (unlike the relationship between `int` and `signed int`).

Character Types

- `char`, `signed char`, `unsigned char`
- Other types for wide characters: `wchar_t`, `char16_t`, `char32_t`.
- Do `char` and `signed char` name the same type?
  NO. The type `char` is neither `signed char` nor `unsigned char`. Whether `char` is signed depends on the implementation, but it is a **distinct type** (unlike the relationship between `int` and `signed int`).
  To know the exact choices made by each implementation, see `https://en.cppreference.com/w/cpp/language/types`.

Character Types

- `char`, `signed char`, `unsigned char`
- Other types for wide characters: `wchar_t`, `char16_t`, `char32_t`.
- Do `char` and `signed char` name the same type?
  NO. The type `char` is neither `signed char` nor `unsigned char`. Whether `char` is signed depends on the implementation, but it is a **distinct type** (unlike the relationship between `int` and `signed int`).
  To know the exact choices made by each implementation, see `https://en.cppreference.com/w/cpp/language/types`.
- How do you save the returned value of `getchar`?

Character Types

- `char`, `signed char`, `unsigned char`
- Other types for wide characters: `wchar_t`, `char16_t`, `char32_t`.
- Do `char` and `signed char` name the same type?
  NO. The type `char` is neither `signed char` nor `unsigned char`. Whether `char` is signed depends on the implementation, but it is a **distinct type** (unlike the relationship between `int` and `signed int`).
  To know the exact choices made by each implementation, see `https://en.cppreference.com/w/cpp/language/types`.
- How do you save the returned value of `getchar`?
  `int` is recommended because `EOF` is `-1`.

Which Type to Use?

- Use `int` for integer arithmetic. `int` should be integer type that target processor works with most efficiently. If `int` is not large enough, use `long long`.
- Use `bool` for boolean values, especially in C++.
- Use `double` for floating-point computations.

Which Type to Use?

- Use `int` for integer arithmetic. `int` should be integer type that target processor works with most efficiently. If `int` is not large enough, use `long long`.
- Use `bool` for boolean values, especially in C++.
- Use `double` for floating-point computations.
  - The precision of `float` is usually not enough.
  - The cost of double-precision calculations versus single-precision is negligible. (In fact, double-precision operations are even faster on certain machines.)
  - The precision offered by `long double` is usually unnecessary.

Contents

Define a Function

- `return-type function-name(parameters) { function-body }`
- How to return a value?

Define a Function

- `return-type function-name(parameters)` { `function-body` }
- How to return a value?
  The `return` statement.

Define a Function

- `return-type function-name(parameters) { function-body }`
- How to return a value?
  The `return` statement.
- How to define a function without return-value?

Define a Function

- `return-type function-name(parameters) { function-body }`
- How to return a value?
  The `return` statement.
- How to define a function without return-value?
  Set the return-type to `void`.

Define a Function

- `return-type function-name(parameters) { function-body }`
- How to return a value?
  The `return` statement.
- How to define a function without return-value?
  Set the return-type to `void`.
- What happens when a function returns?

Define a Function

- `return-type function-name(parameters) { function-body }`
- How to return a value?
  The `return` statement.
- How to define a function without return-value?
  Set the return-type to `void`.
- What happens when a function returns?
  - The control flow goes back to the caller.
  - Possibly a value is passed to the caller.

Define a Function

Notice

Be sure to discriminate between the return of a function and the output of a program! They have nothing to do with each other.

Define a Function

Notice

Be sure to discriminate between the return of a function and the output of a program! They have nothing to do with each other.

Notice

A non-void function without a return statement causes no error (although probably a warning) when it is compiled, but results in undefined behavior when running!

The `main` Function

- You might have seen some people/textbooks writing 'void main'...

The `main` Function

- You might have seen some people/textbooks writing 'void main'...

  The definition 'void main' is not and has never been in C++, nor has it even been in C. (Bjarne Stroustrup)

The `main` Function

- You might have seen some people/textbooks writing 'void main'...

  The definition 'void main' is not and has never been in C++, nor has it even been in C. (Bjarne Stroustrup)

- You might have seen some people/textbooks leaving out the return-type...

The `main` Function

- You might have seen some people/textbooks writing 'void main'...

  The definition 'void main' is not and has never been in C++, nor has it even been in C. (Bjarne Stroustrup)

- You might have seen some people/textbooks leaving out the return-type...

  In C89, the default return-type of a function is `int`. However, this rule is not in standard C++ and has been dropped since C99. Don't be lazy!

The `main` Function

- You might have seen some people/textbooks writing ‘void main’...

  The definition ‘void main’ is not and has never been in C++, nor has it even been in C. (Bjarne Stroustrup)

- You might have seen some people/textbooks leaving out the return-type...

  In C89, the default return-type of a function is `int`. However, this rule is not in standard C++ and has been dropped since C99. Don’t be lazy!

- You might have seen many people leaving out the `return` statement in `main`...

The `main` Function

- You might have seen some people/textbooks writing ‘void main’...

  The definition ‘void main’ is not and has never been in C++, nor has it even been in C. (Bjarne Stroustrup)

- You might have seen some people/textbooks leaving out the return-type...

  In C89, the default return-type of a function is `int`. However, this rule is not in standard C++ and has been dropped since C99. Don’t be lazy!

- You might have seen many people leaving out the `return` statement in `main`...

  This is ok because the compiler will impose a return-value 0 if the program exits successfully.

Contents

Precedence and Associativity

- How is `a + b * c + d` evaluated?
- How is `a - b + c` evaluated?
- How is `f() + g() + h()` evaluated?

Precedence and Associativity

- How is `a + b * c + d` evaluated?
- How is `a - b + c` evaluated?
- How is `f() + g() + h()` evaluated?

Node

The precedence and associativity do not necessarily determine the evaluation order!

Precedence and Associativity

- How is `a + b * c + d` evaluated?
- How is `a - b + c` evaluated?
- How is `f() + g() + h()` evaluated?

Node

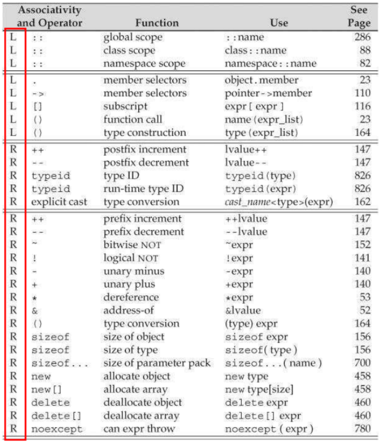
The precedence and associativity do not necessarily determine the evaluation order!

Typical undefined behavior: `printf("%d %d", a, ++a);`

Operator Precedence Table

Apart from the precedence of operators, you should also remember the associativities.

Table 4.4. Operator Precedence



Associativity and Operator		Function	Use	See Page
L	`::`	global scope	`::name`	286
L	`::`	class scope	`class::name`	88
L	`::`	namespace scope	`namespace::name`	82
L	`.`	member selectors	object.member	23
L	`->`	member selectors	pointer->member	110
L	`[]`	subscript	expr [expr]	116
L	`()`	function call	name (expr_list)	23
L	`()`	type construction	type (expr_list)	164
R	`++`	postfix increment	lvalue++	147
R	`--`	postfix decrement	lvalue--	147
R	`typeid`	type ID	`typeid` (type)	826
R	`typeid`	run-time type ID	`typeid` (expr)	826
R	explicit cast	type conversion	*cast_name*<type>(expr)	162
R	`++`	prefix increment	++lvalue	147
R	`--`	prefix decrement	--lvalue	147
R	`~`	bitwise NOT	~expr	152
R	`!`	logical NOT	!expr	141
R	`-`	unary minus	-expr	140
R	`+`	unary plus	+expr	140
R	`*`	dereference	*expr	53
R	`&`	address-of	&lvalue	52
R	`()`	type conversion	(type) expr	164
R	`sizeof`	size of object	`sizeof` expr	156
R	`sizeof`	size of type	`sizeof`(type)	156
R	`sizeof...`	size of parameter pack	`sizeof...`(name)	700
R	`new`	allocate object	`new` type	458
R	`new[]`	allocate array	`new` type[size]	458
R	`delete`	deallocate object	`delete` expr	460
R	`delete[]`	deallocate array	`delete[]` expr	460
R	`noexcept`	can expr throw	`noexcept` (expr)	780

Short-circuit Evaluation

Logical operators `&&` and `||` are short-circuited:

- Both `&&` and `||` evaluates their left operand first.
- If the left operand of `&&` evalutes `false`, the right operand will not be evaluated, and the whole expression evaluates `false`.
- If the left operand of `||` evalutes `true`, the right operand will not be evaluated, and the whole expression evaluates `true`.