# 2.x — Chapter 2 summary and quiz

ALEX DECEMBER 28, 2023

## Chapter Review

A **function** is a reusable sequence of statements designed to do a particular job. Functions you write yourself are called **user-defined** functions.

A **function call** is an expression that tells the CPU to execute a function. The function initiating the function call is the **caller**, and the function being called is the **callee** or **called** function. Do not forget to include parenthesis when making a function call.

The curly braces and statements in a function definition are called the **function body**.

A function that returns a value is called a **value-returning function**. The **return type** of a function indicates the type of value that the function will return. The **return statement** determines the specific **return value** that is returned to the caller. A return value is copied from the function back to the caller -- this process is called **return by value**. Failure to return a value from a non-void function will result in undefined behavior.

The return value from function main is called a **status code**, and it tells the operating system (and any other programs that called yours) whether your program executed successfully or not. By consensus a return value of 0 means success, and a non-zero return value means failure.

Practice **DRY** programming -- “don’t repeat yourself”. Make use of variables and functions to remove redundant code.

Functions with a return type of **void** do not return a value to the caller. A function that does not return a value is called a **void function** or **non-value returning function**. Void functions can’t be called where a value is required.

A return statement that is not the last statement in a function is called an **early return**. Such a statement causes the function to return to the caller immediately.

A **function parameter** is a variable used in a function where the value is provided by the caller of the function. An **argument** is the specific value passed from the caller to the function. When an argument is copied into the parameter, this is called **pass by value**.

Function parameters and variables defined inside the function body are called **local variables**. The time in which a variable exists is called its **lifetime**. Variables are created and destroyed at **runtime**, which is when the program is running. A variable’s **scope** determines where it can be seen and used. When a variable can be seen and used, we say it is **in scope**. When it can not be seen, it can not be used, and we say it is **out of scope**. Scope is a **compile-time** property, meaning it is enforced at compile time.

**Whitespace** refers to characters used for formatting purposes. In C++, this includes spaces, tabs, and newlines.

A **forward declaration** allows us to tell the compiler about the existence of an identifier before actually defining the identifier. To write a forward declaration for a function, we use a **function prototype**, which includes the function's return type, name, and parameters, but no function body, followed by a semicolon.

A **definition** actually implements (for functions and types) or instantiates (for variables) an identifier. A **declaration** is a statement that tells the compiler about the existence of the identifier. In C++, all definitions serve as declarations. **Pure declarations** are declarations that are not also definitions (such as function prototypes).

Most non-trivial programs contain multiple files.

When two identifiers are introduced into the same program in a way that the compiler or linker can't tell them apart, the compiler or linker will error due to a **naming collision**. A **namespace** guarantees that all identifiers within the namespace are unique. The std namespace is one such namespace.

The **preprocessor** is a process that runs on the code before it is compiled. **Directives** are special instructions to the preprocessor. Directives start with a # symbol and end with a newline. A **macro** is a rule that defines how input text is converted to a replacement output text.

**Header files** are files designed to propagate declarations to code files. When using the #include directive, the #include directive is replaced by the contents of the included file. When including headers, use angled brackets when including system headers (e.g. those in the C++ standard library), and use double quotes when including user-defined headers (the ones you write). When including system headers, include the versions with no .h extension if they exist.

**Header guards** prevent the contents of a header from being included more than once into a given code file. They do not prevent the contents of a header from being included into multiple different code files.

---

## Quiz time

Be sure to use your editor's auto-formatting feature to keep your formatting consistent and make your code easier to read.

**Question #1**

Write a single-file program (named main.cpp) that reads two separate integers from the user, adds them together, and then outputs the answer. The program should use three functions:

- A function named "readNumber" should be used to get (and return) a single integer from the user.
- A function named "writeAnswer" should be used to output the answer. This function should take a single parameter and have no return value.
- A main() function should be used to glue the above functions together.

Show Hint (javascript:void(0))

Show Hint (javascript:void(0))

Show Solution (javascript:void(0))

---

**Question #2**

Modify the program you wrote in exercise #1 so that readNumber() and writeAnswer() live in a separate file called "io.cpp". Use a forward declaration to access them from main().

If you're having problems, make sure "io.cpp" is properly added to your project so it gets compiled.

Show Solution (javascript:void(0))

**Question #3**

Modify the program you wrote in #2 so that it uses a header file (named io.h) to access the functions instead of using forward declarations directly in your code (.cpp) files. Make sure your header file uses header guards.

Show Solution (javascript:void(0))

If you compile your program and get errors like one of these:

```
unresolved external symbol "int __cdecl readNumber(void)" (?readNumber@@YAHXZ)
undefined reference to `readNumber()'
```

Then you probably forgot to include `io.cpp` in your project, so the definitions for `readNumber()` (and `writeAnswer()`) aren't being compiled into your project.


**Next lesson**
3.1 Syntax and semantic errors

Back to table of contents

**Previous lesson**
2.13 How to design your first programs