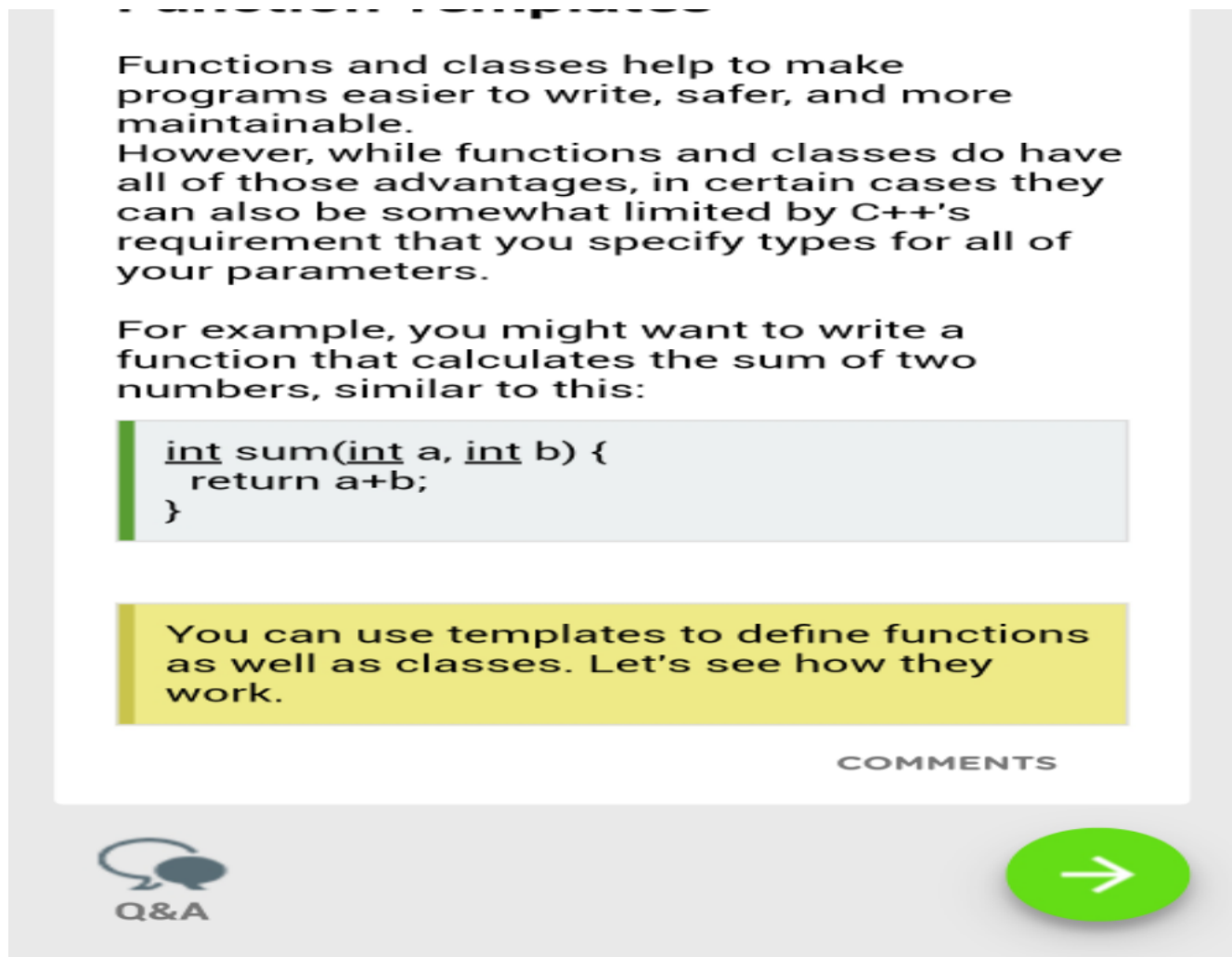


DAILY ASSESSMENT FORMAT

Date:	26-06-2020	Name:	Jagadeesha Hegde
Course:	sololearn	USN:	4AL17EC036
Topic:	C++ programming	Semester & Section:	6th A-sec
Github Repository:	Jagadeesha-036		

FORENOON SESSION DETAILS

Image of session



The image is a screenshot of a presentation slide. At the top, there is a title bar that is partially cut off. The main content area has a light gray background. It contains two paragraphs of text. The first paragraph discusses the advantages and limitations of functions and classes in C++. The second paragraph provides an example of a function that calculates the sum of two numbers. Below the text, there is a code block with a green border containing C++ code for a function named 'sum'. Underneath the code block, there is a yellow box with text explaining that templates can be used to define functions and classes. At the bottom right of the slide, there is a 'COMMENTS' button. At the bottom left, there is a 'Q&A' icon. At the bottom right, there is a large green circular button with a white right-pointing arrow.

Functions and classes help to make programs easier to write, safer, and more maintainable. However, while functions and classes do have all of those advantages, in certain cases they can also be somewhat limited by C++'s requirement that you specify types for all of your parameters.

For example, you might want to write a function that calculates the sum of two numbers, similar to this:

```
int sum(int a, int b) {  
    return a+b;  
}
```

You can use templates to define functions as well as classes. Let's see how they work.

COMMENTS

Q&A

C++ is general-purpose object-oriented programming (OOP) language developed by Bjarne Stroustrup.

Originally, C++ was called "C with classes," as it had all the properties of the C language with the addition of user-defined data types called "classes." It was renamed C++ in 1983.

C++ is considered an intermediate-level language, as it includes both high and low-level language features.

Some key benefits of C++ are outlined below:

Self memory management :

Using pointers, C++ allows self-memory management, that enhances the execution speed of a program. But it is necessary to explicitly free up the reserved space later on.

The char 'x' takes 1 byte and int 'num' takes 4 bytes in (hypothetical) Memory Addresses.

Object-oriented support C++ can be coded in C style or object-oriented style. In certain scenarios, it can be coded either way - making C++ a good example of a hybrid language.

High performance:

Since C++ allows to manipulate the processor on a lower level, it is quite faster than advanced level languages like Python or C#.

Other essential concepts of C++ include:

1. Polymorphism
2. Virtual and friend Functions
3. Templates
4. Namespaces
5. Pointers

DAILY ASSESSMENT FORMAT

Date:	26-06-2020	Name:	Jagadeesha Hegde
Course:	sololearn	USN:	4AL17EC036
Topic:	C++ programming	Semester & Section:	6th A-sec
Github Repository:	Jagadeesha-036		

AFTERNOON SESSION DETAILS

The C Preprocessor :

The C preprocessor is a macro processor that is used automatically by the C compiler to transform your program before actual compilation. It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs.

The C preprocessor provides four separate facilities that you can use as you see fit:

- Inclusion of header files. These are files of declarations that can be substituted into your program.
- Macro expansion. You can define macros, which are abbreviations for arbitrary fragments of C code, and then the C preprocessor will replace the macros with their definitions throughout the program.
- Conditional compilation. Using special preprocessing directives, you can include or exclude parts of the program according to various conditions.
- Line control. If you use a program to combine or rearrange source files into an intermediate file which is then compiled, you can use line control to inform the compiler of where each source line originally came from.

C preprocessors vary in some details. This manual discusses the GNU C preprocessor, the C Compatible Compiler Preprocessor. The GNU C preprocessor provides a superset of the features of ANSI Standard C. ANSI Standard C requires the rejection of many harmless constructs commonly used by today's C programs. Such incompatibility would be inconvenient for users, so the GNU C preprocessor is configured to accept these constructs by default. Strictly speaking, to get ANSI Standard C, you must use the options ``-trigraphs'`, ``-undef'` and ``-pedantic'`, but in practice the consequences of having strict ANSI Standard C make it undesirable to do this. The C preprocessor is designed for C-like languages; you may run into problems if you apply it to other kinds of languages, because it assumes that it is dealing with C. For example, the C preprocessor sometimes outputs extra white space to avoid inadvertent C token concatenation, and this may cause problems with other languages.

