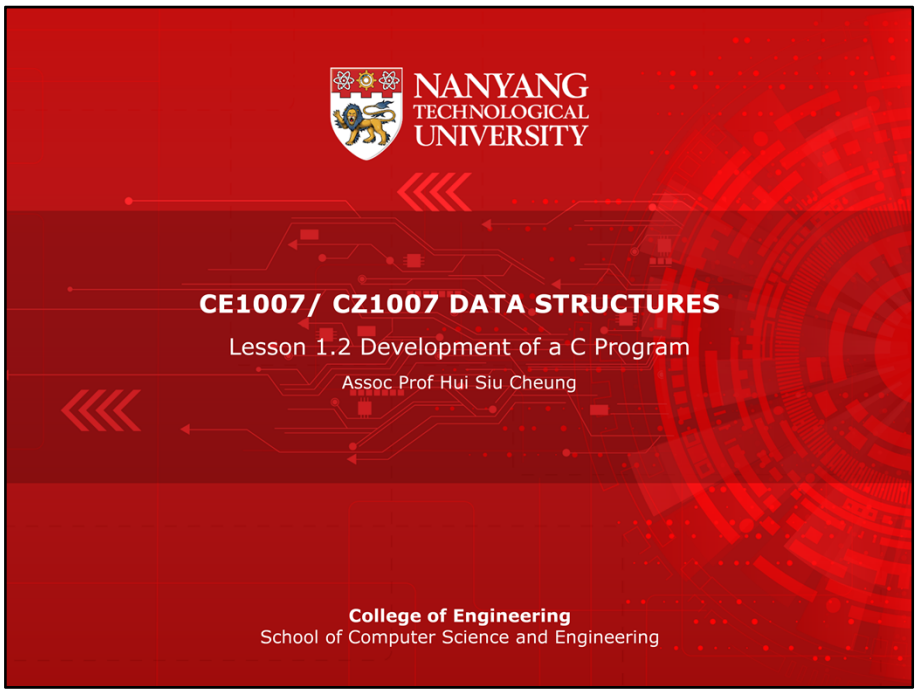
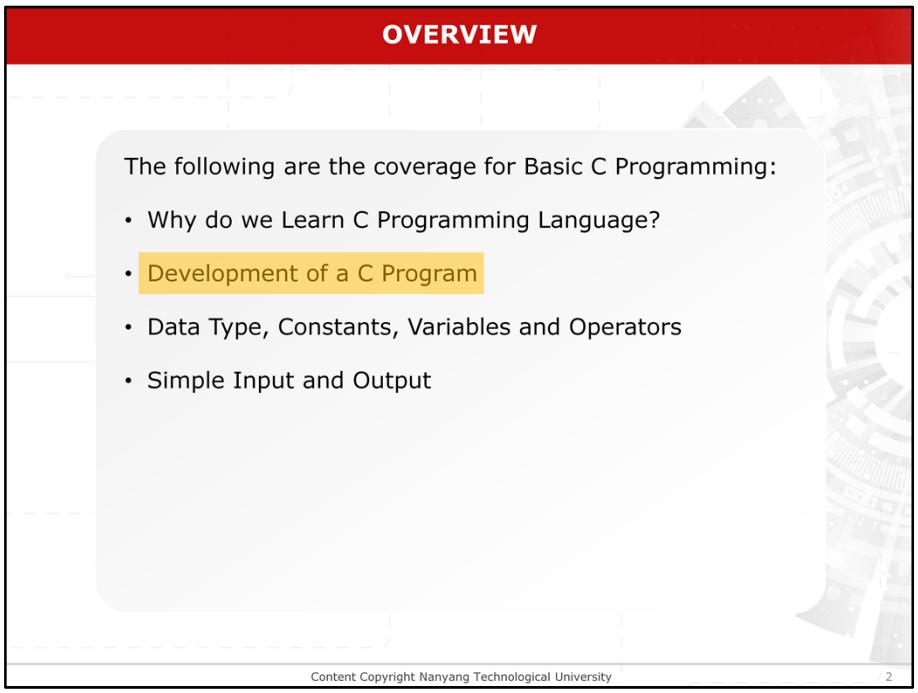


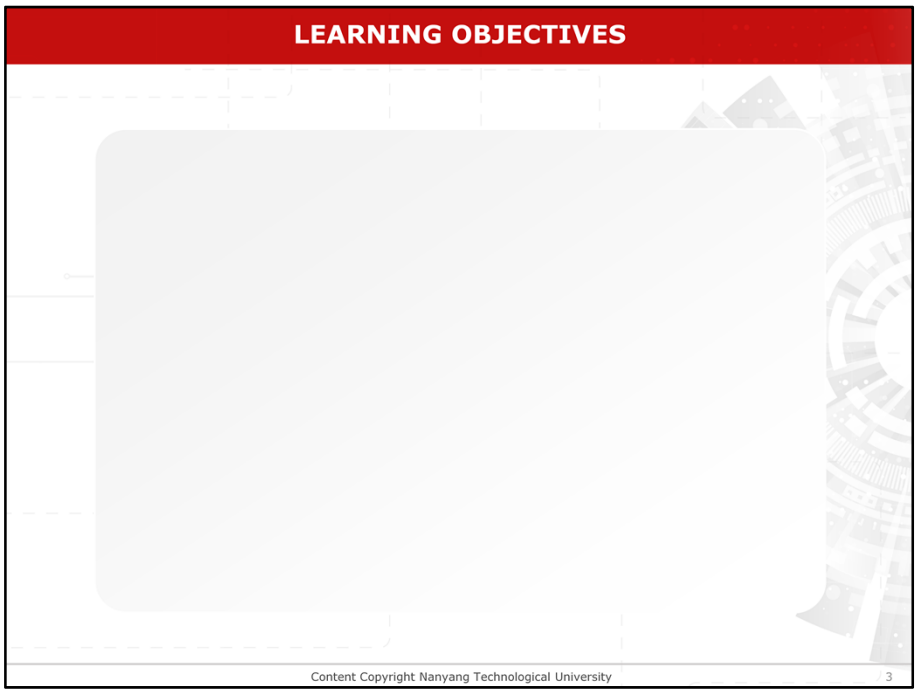
This lesson is on Basic C Programming



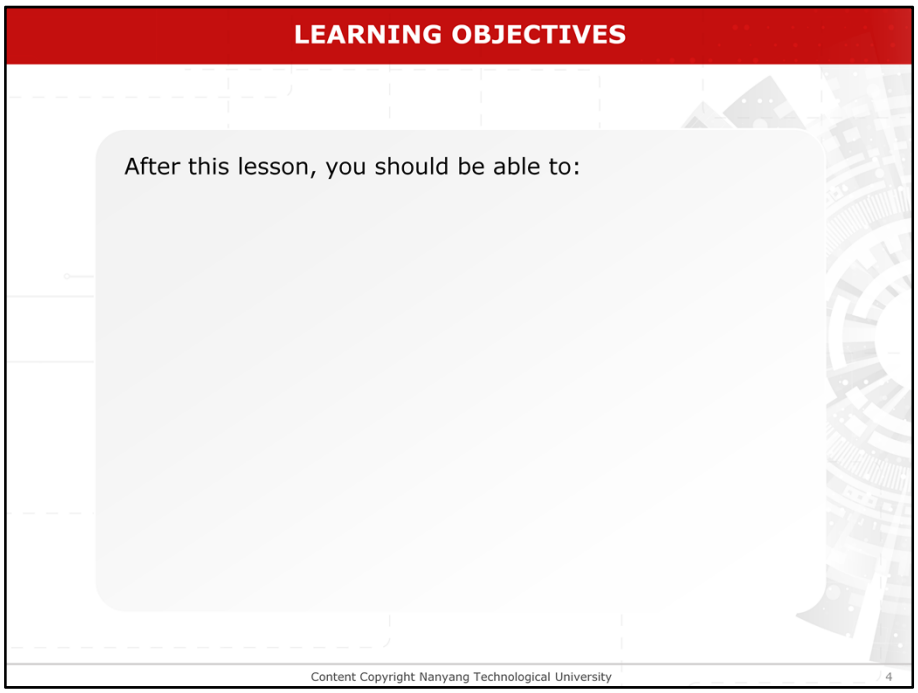
Basic C Programming

There are 4 main sections to cover for Basic C Programming as shown.

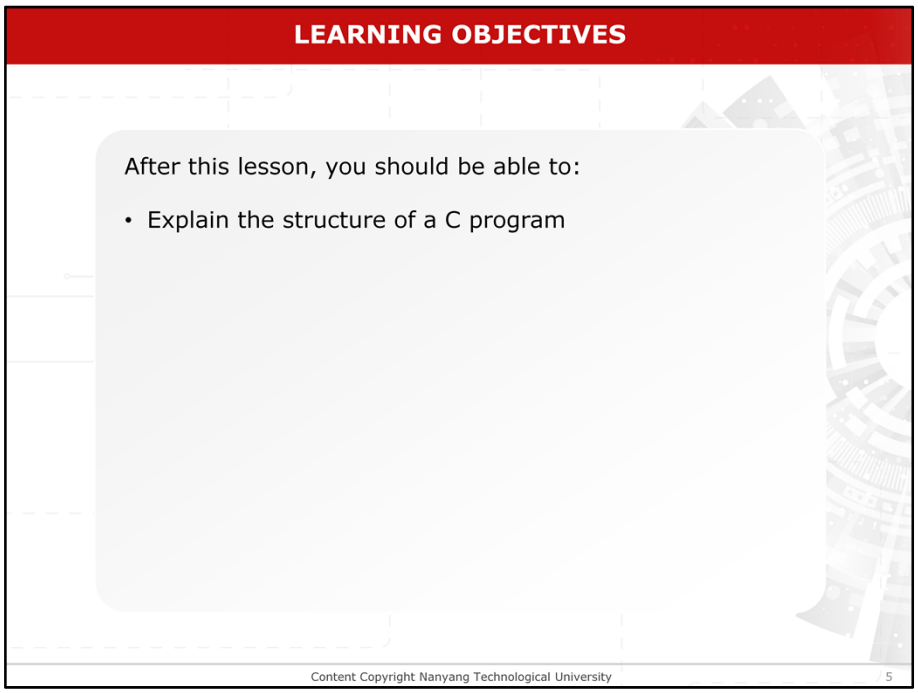
This video lesson focuses on: Development of a C program


LEARNING OBJECTIVES
Content Copyright Nanyang Technological University
3

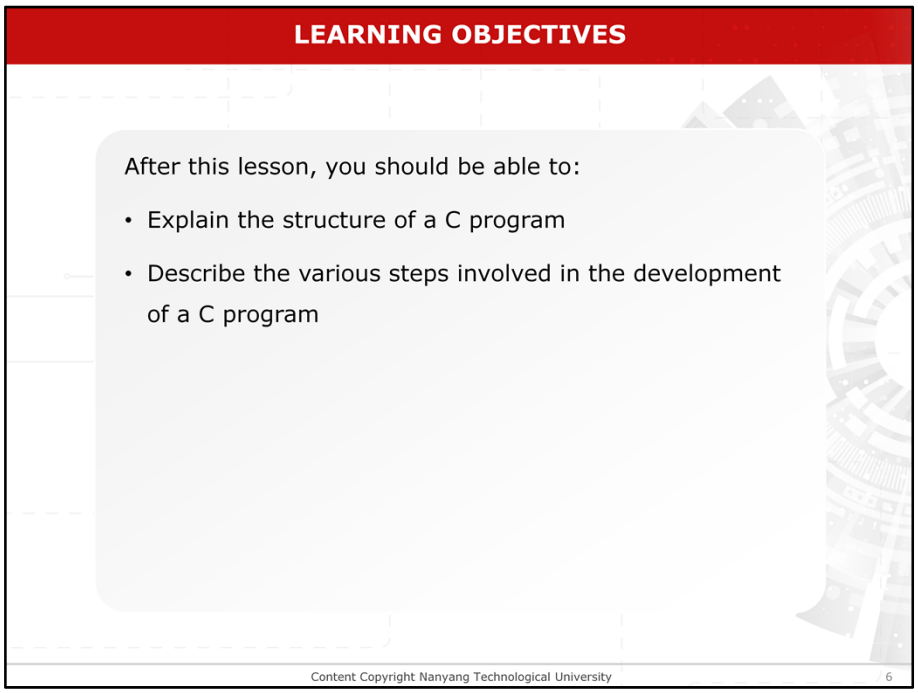
[Pause 1 sec]



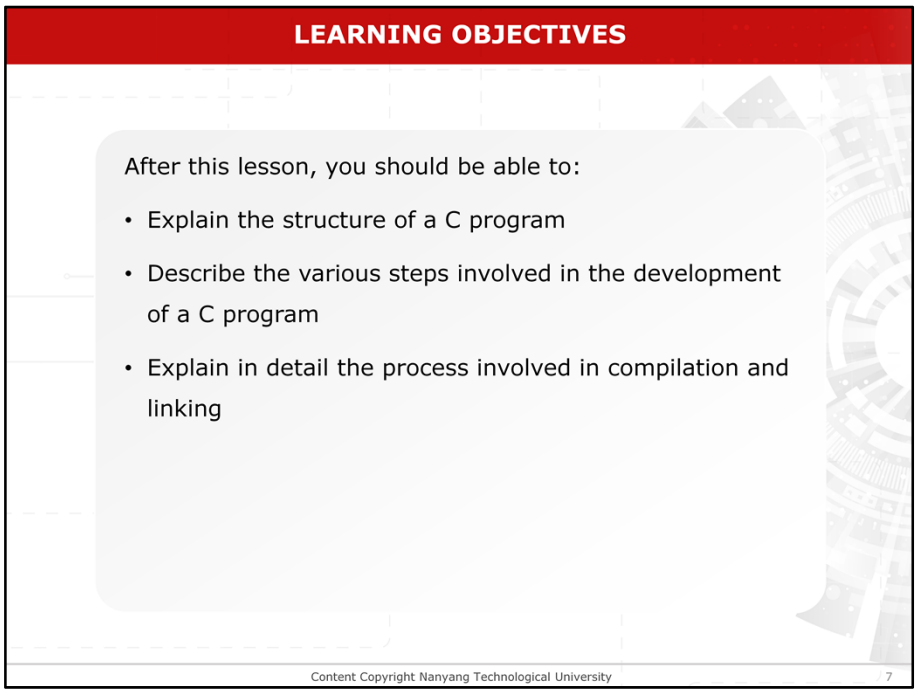
After this lesson, you should be able to
Pause: 1 sec



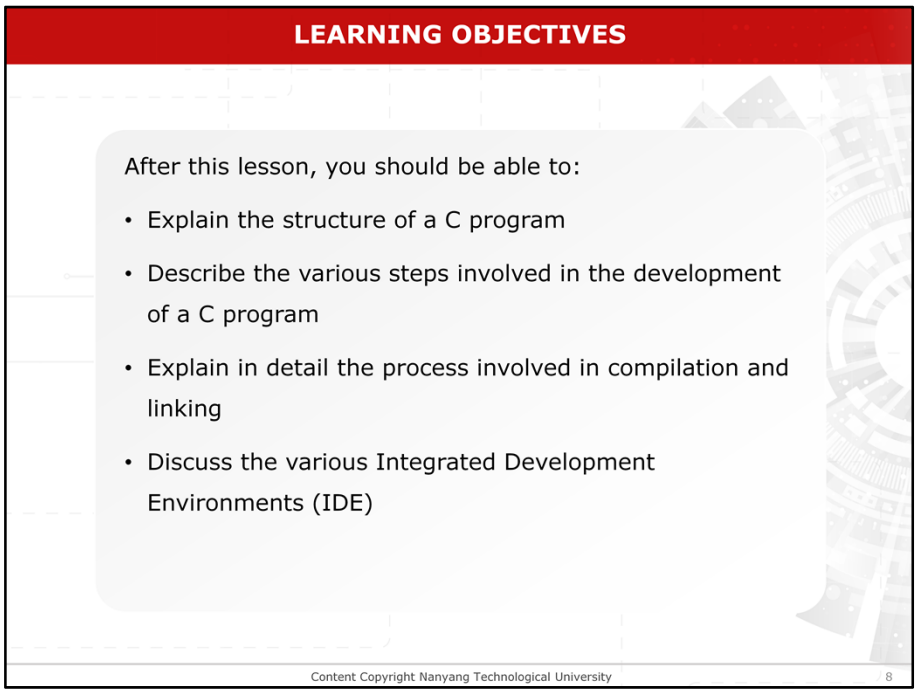
Explain the structure of a C program
[Pause: 1 sec]



Describe the various steps involved in development of a C program
[Pause 1 sec]

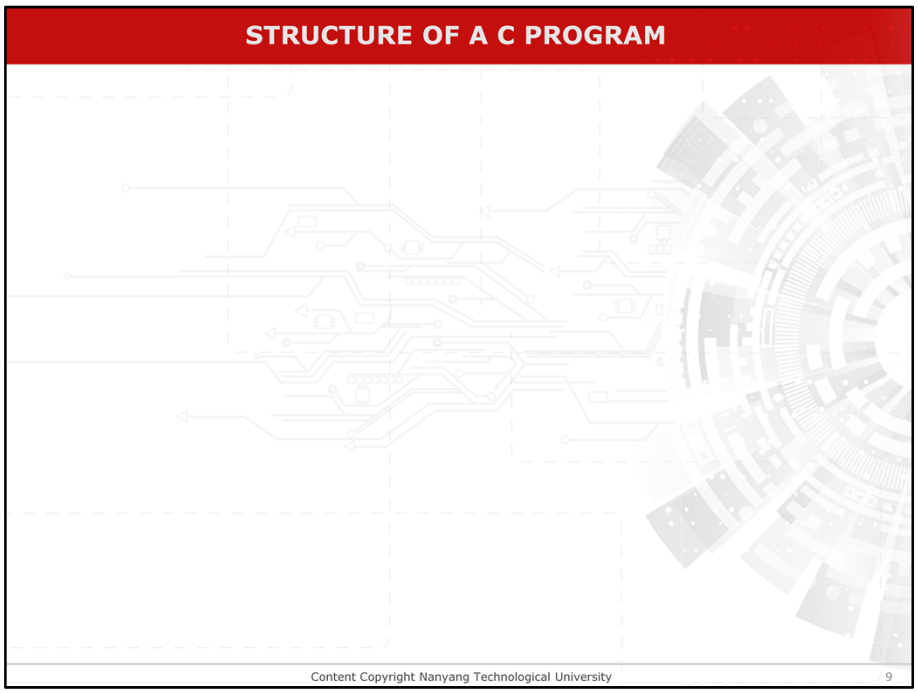


Explain in detail the process involved in compilation and linking
[Pause: 1 sec]

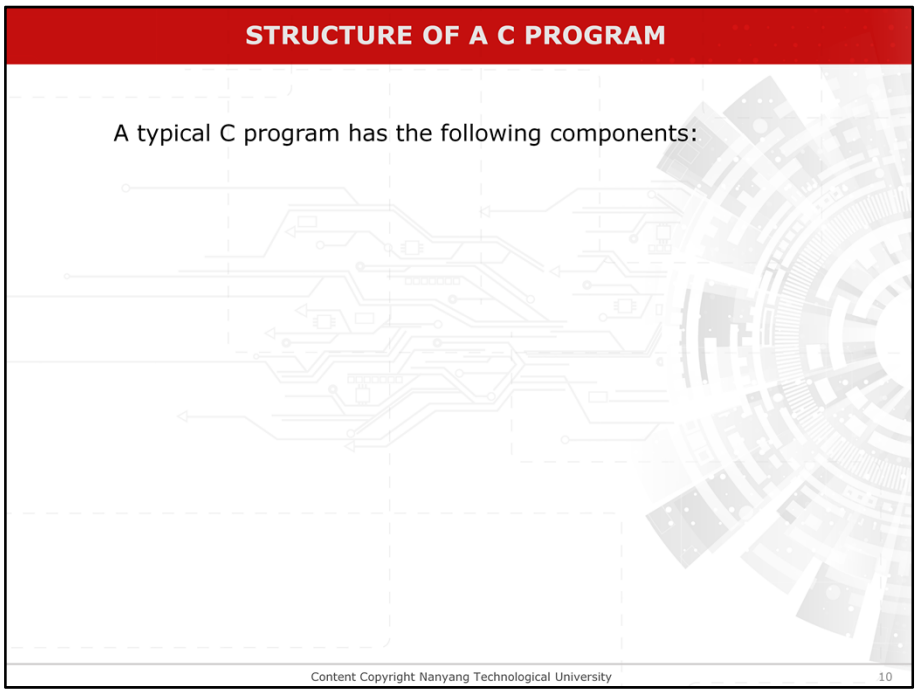


Discuss the various Integrated Development Environments (IDE)

[Pause 1 sec]

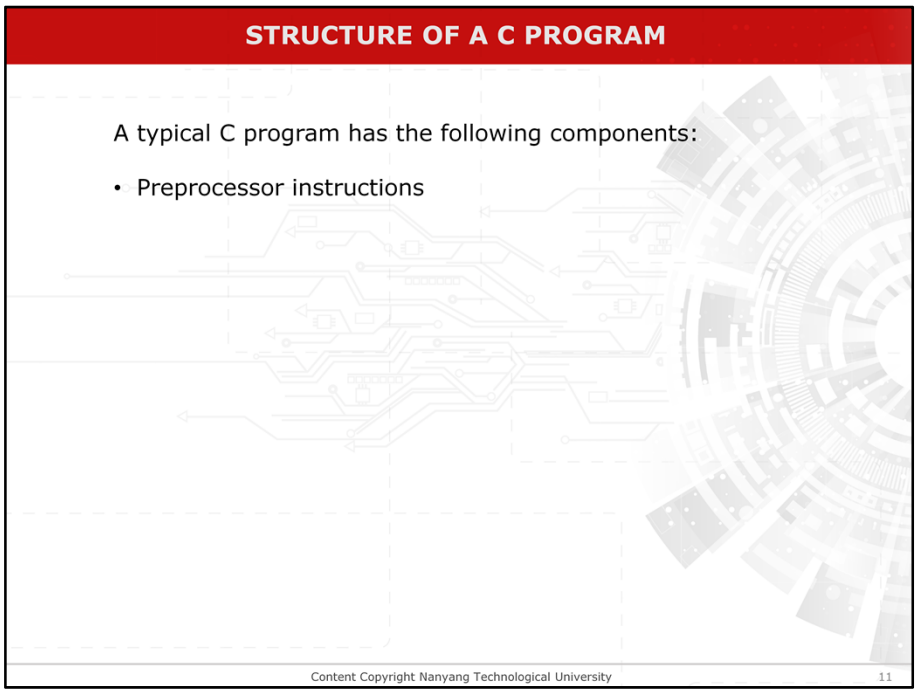


Structure of a C Program

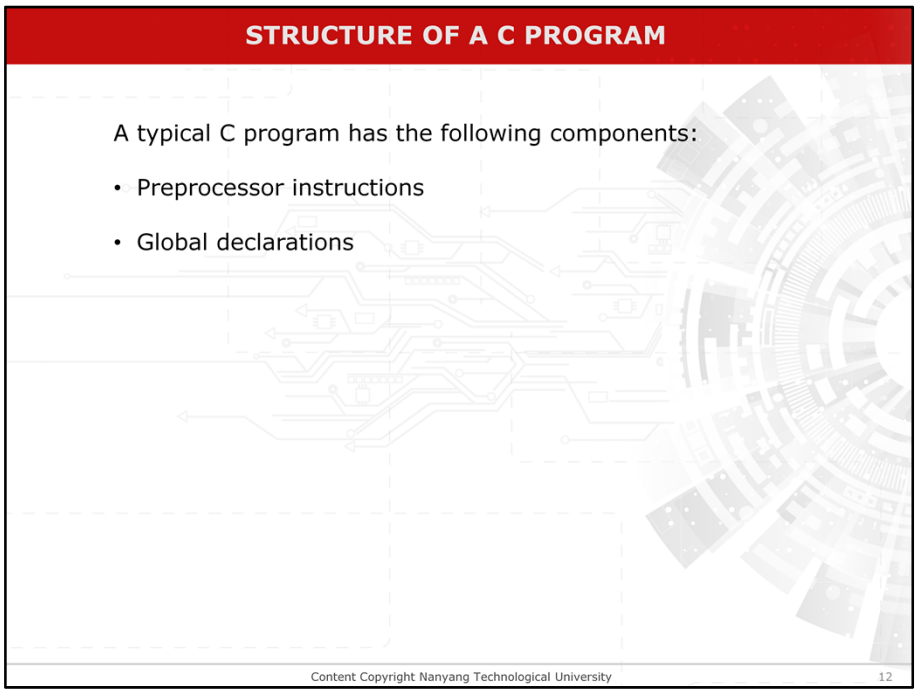


Structure of a C Program

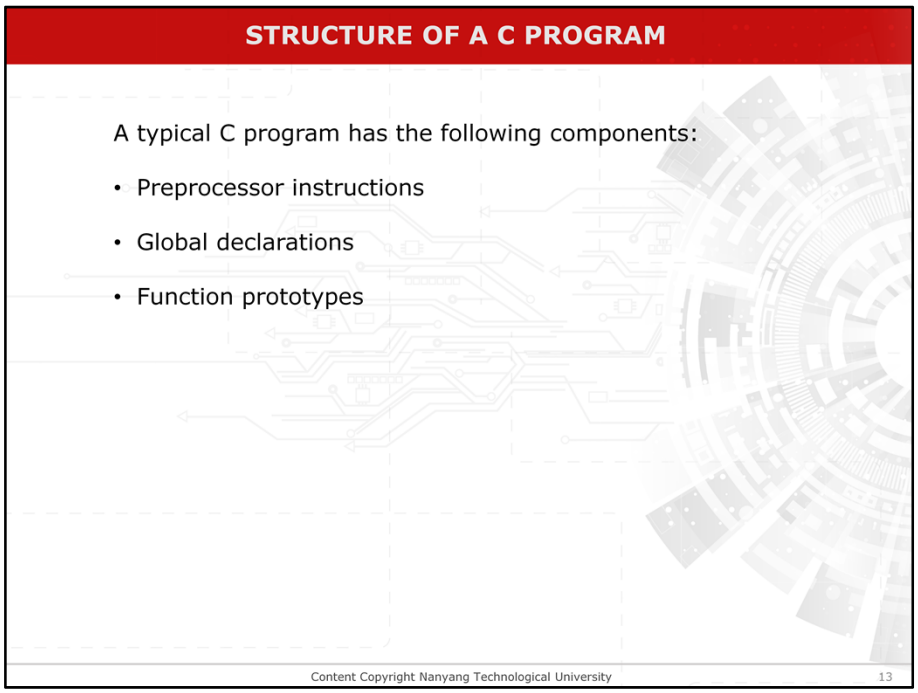
A typical C program has the following components



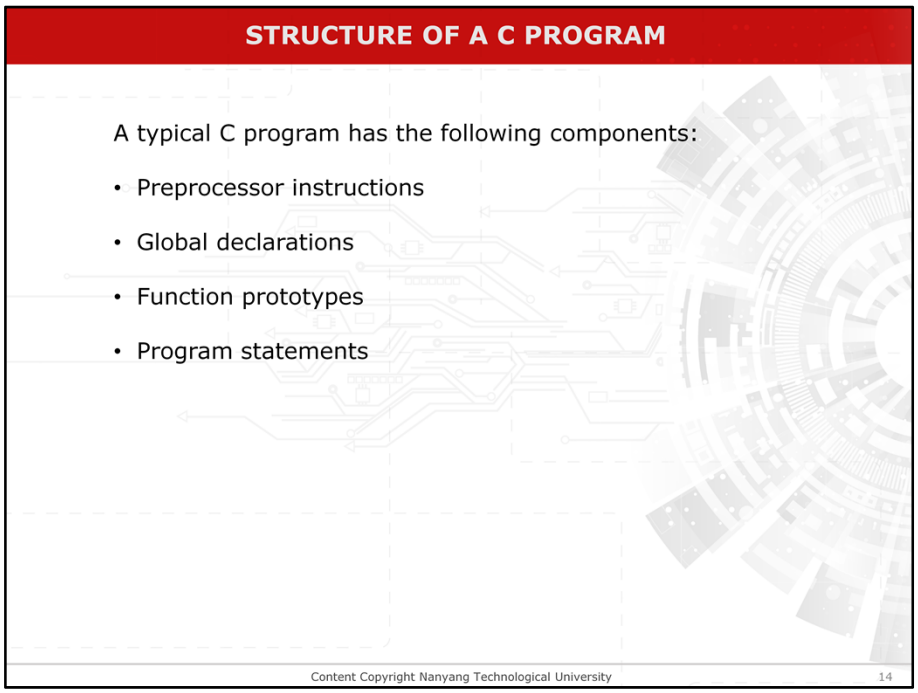
preprocessor instructions



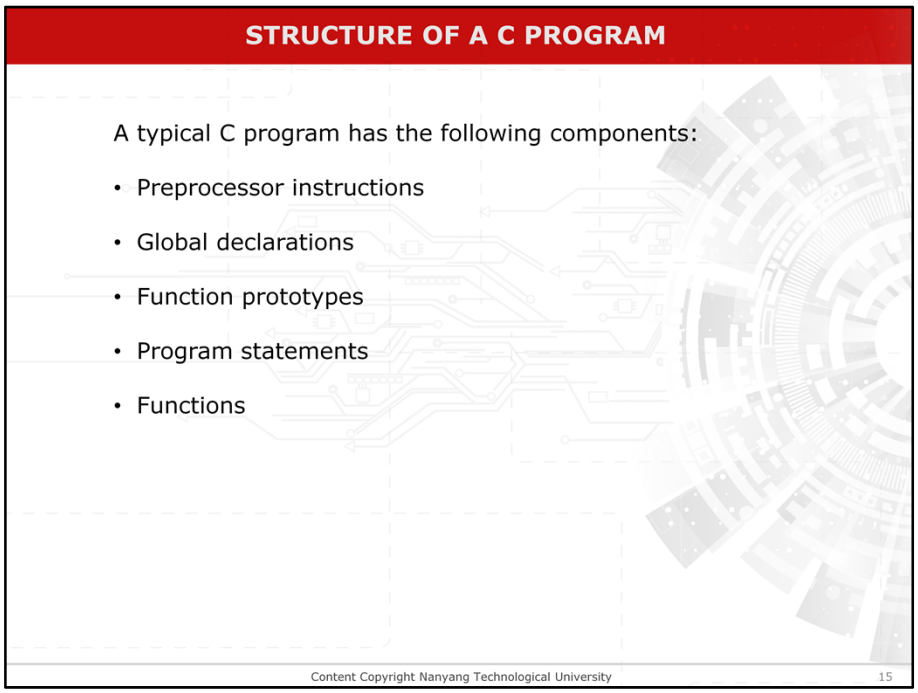
global declarations



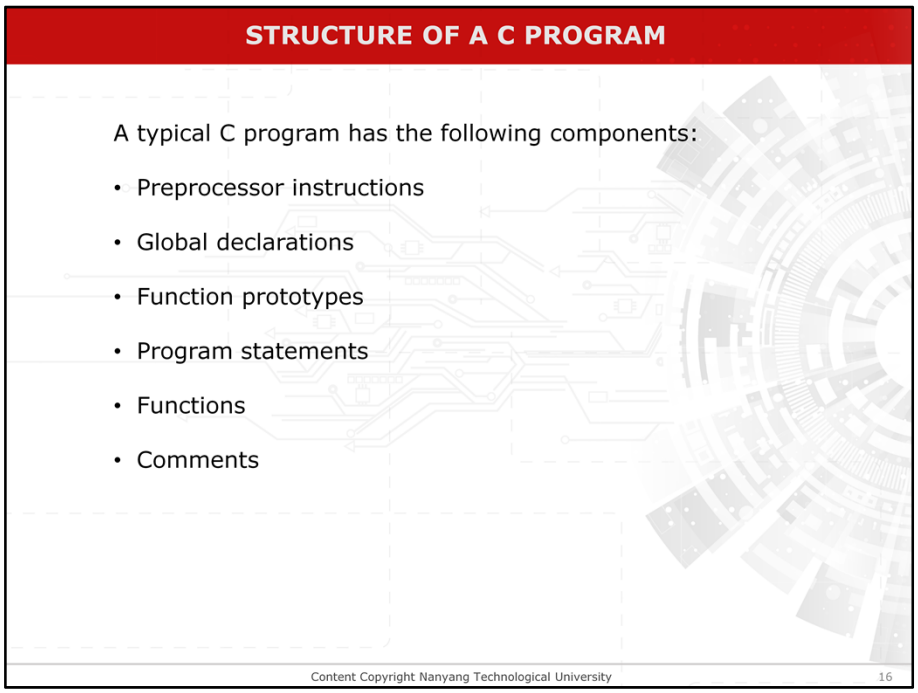
function prototypes



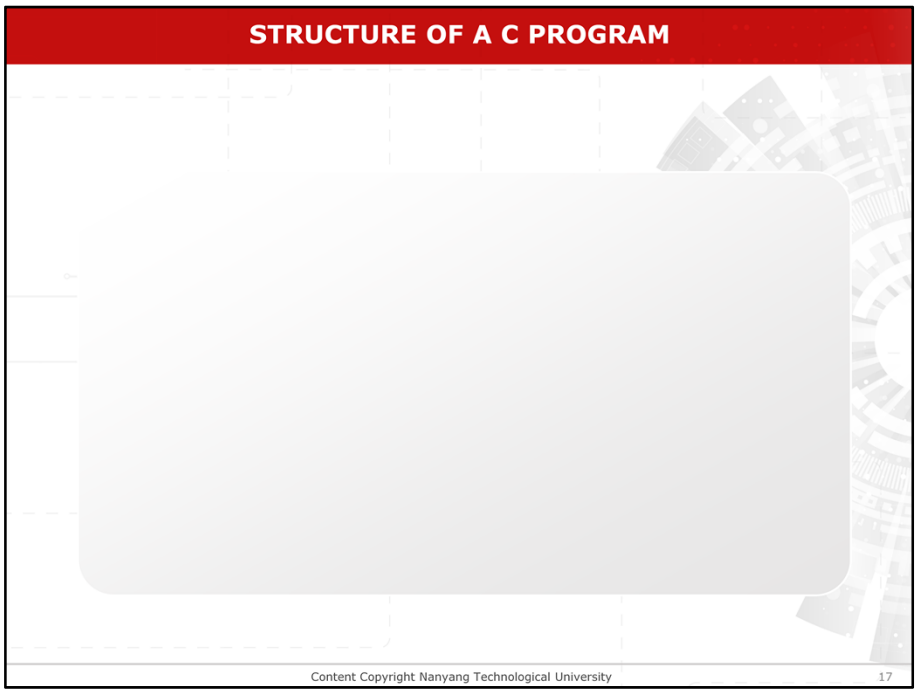
program statements


STRUCTURE OF A C PROGRAM
A typical C program has the following components:
• Preprocessor instructions
• Global declarations
• Function prototypes
• Program statements
• Functions
Content Copyright Nanyang Technological University
15

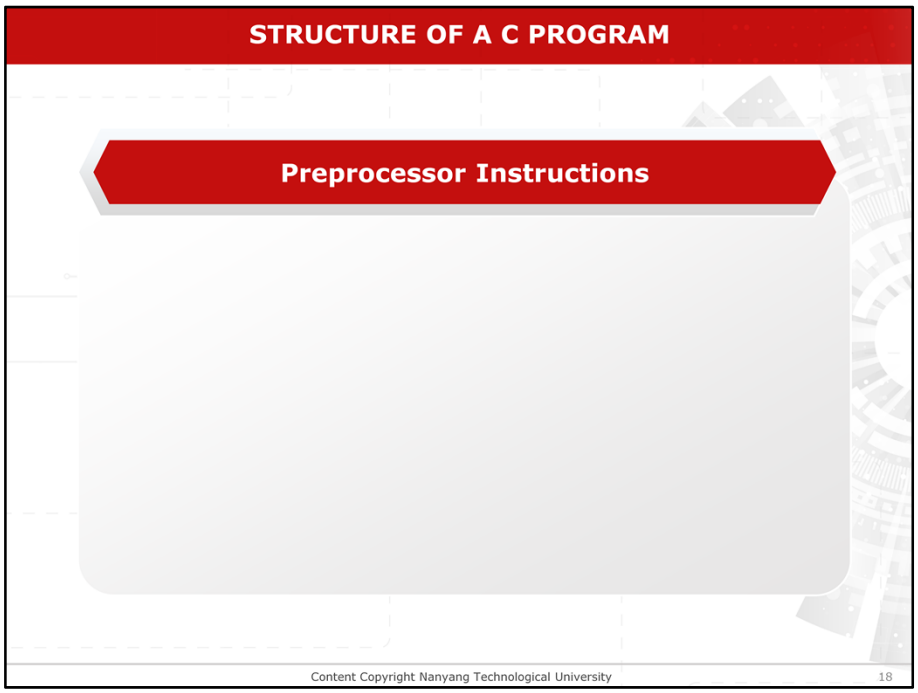
functions



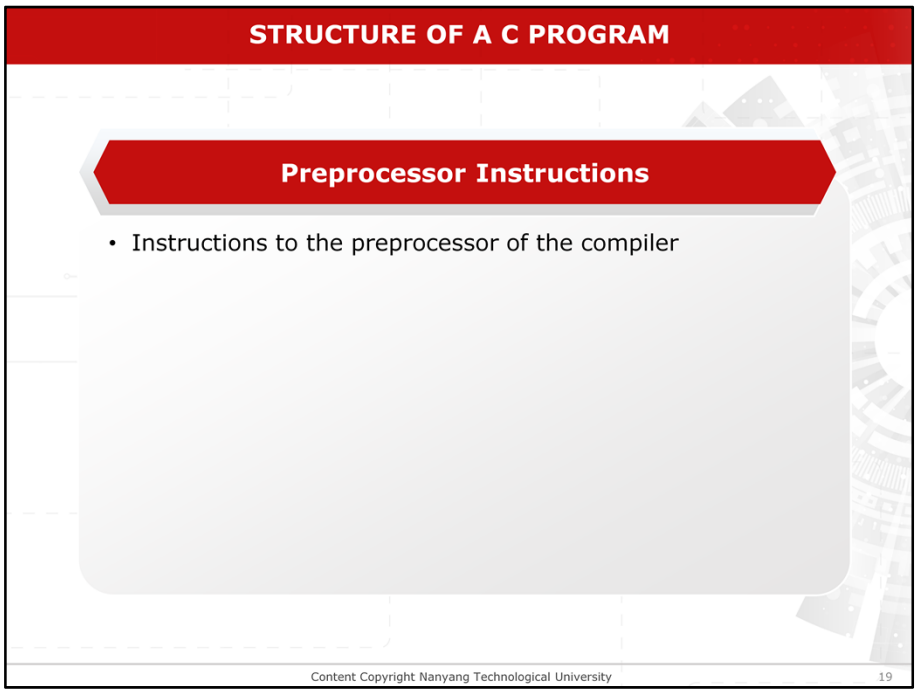
and comments.



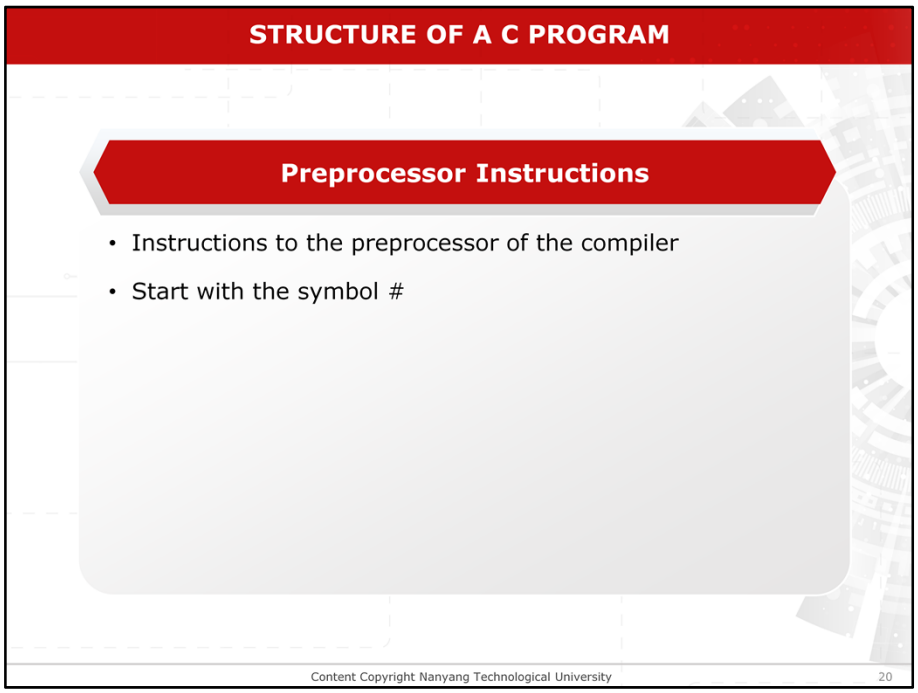
Structure of a C program
[Pause: 1sec]



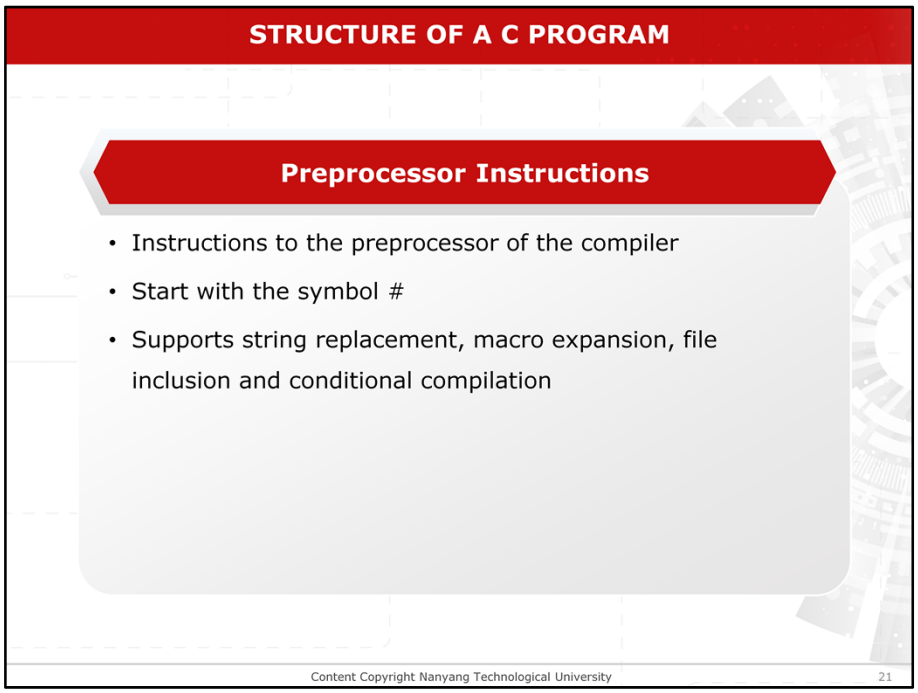
Preprocessor Instructions
[Pause: 1 sec]



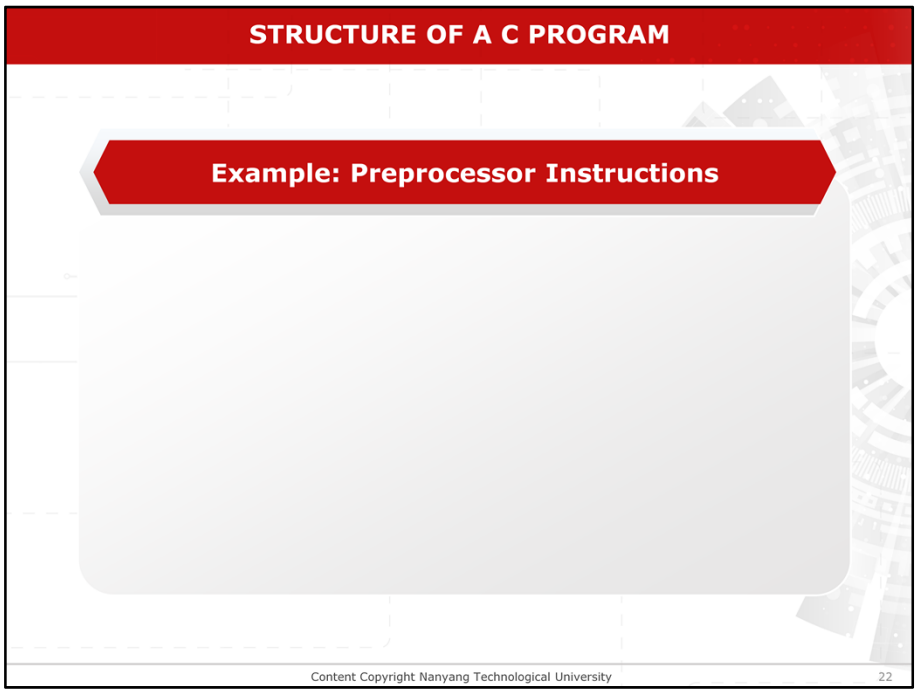
Preprocessor instructions refer to the instructions to the preprocessor of the compiler.



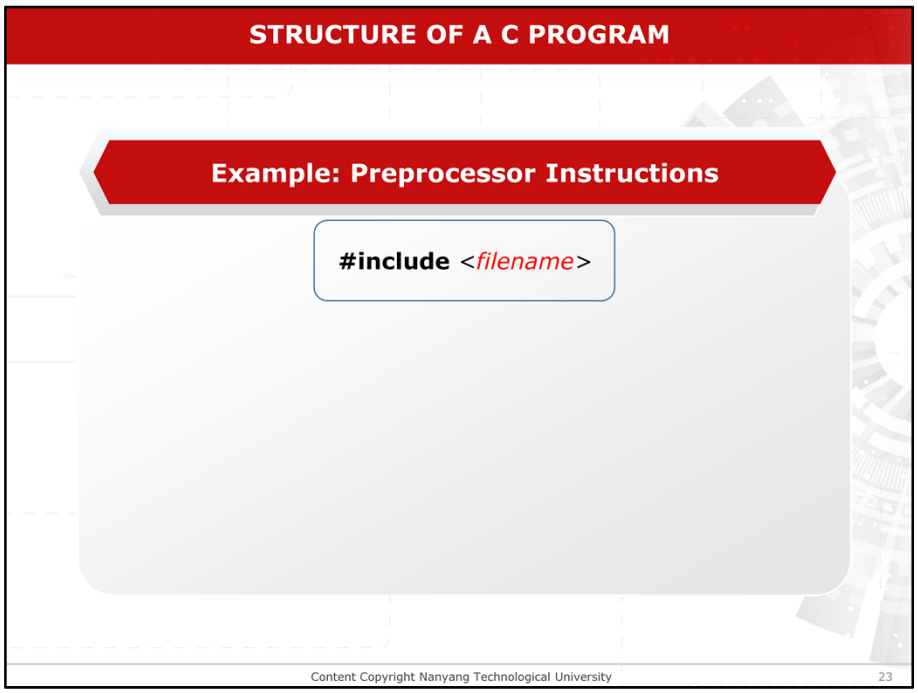
. All preprocessor instructions start with the symbol **#**.



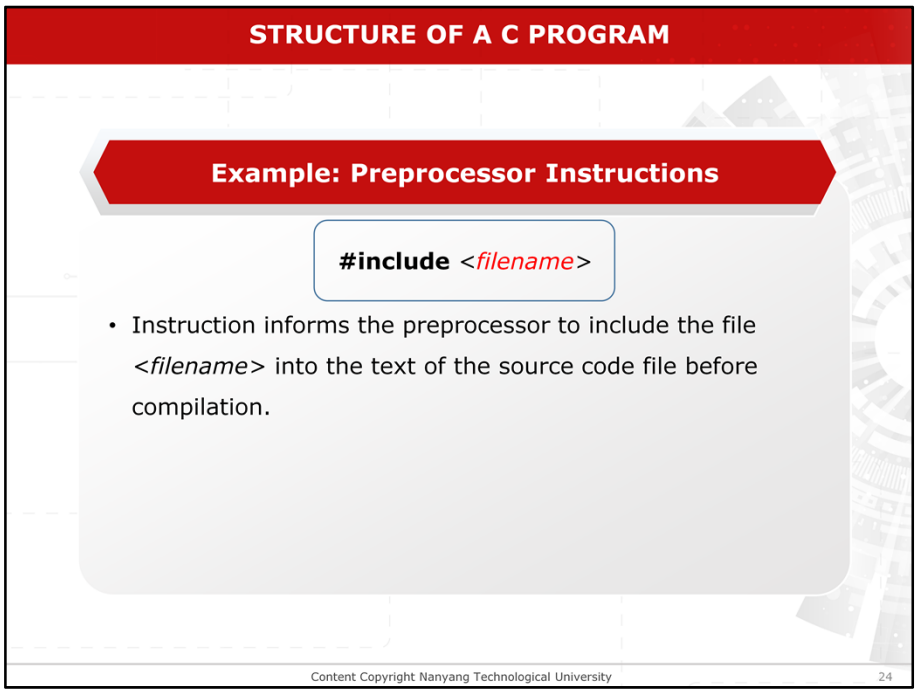
The C preprocessor supports string replacement, macro expansion, file inclusion and conditional compilation.



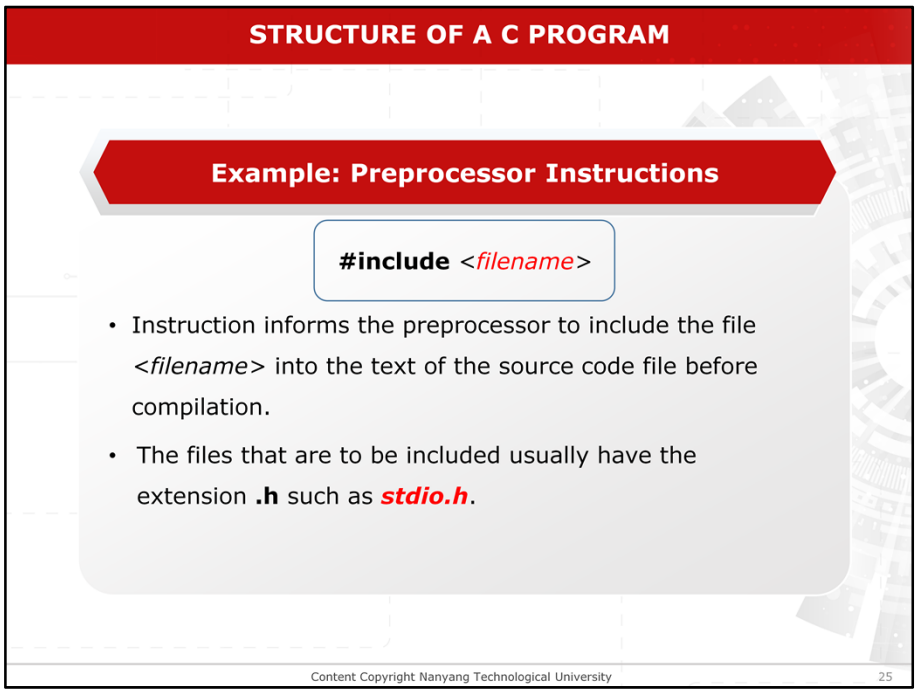
For example
[Pause: 1sec]



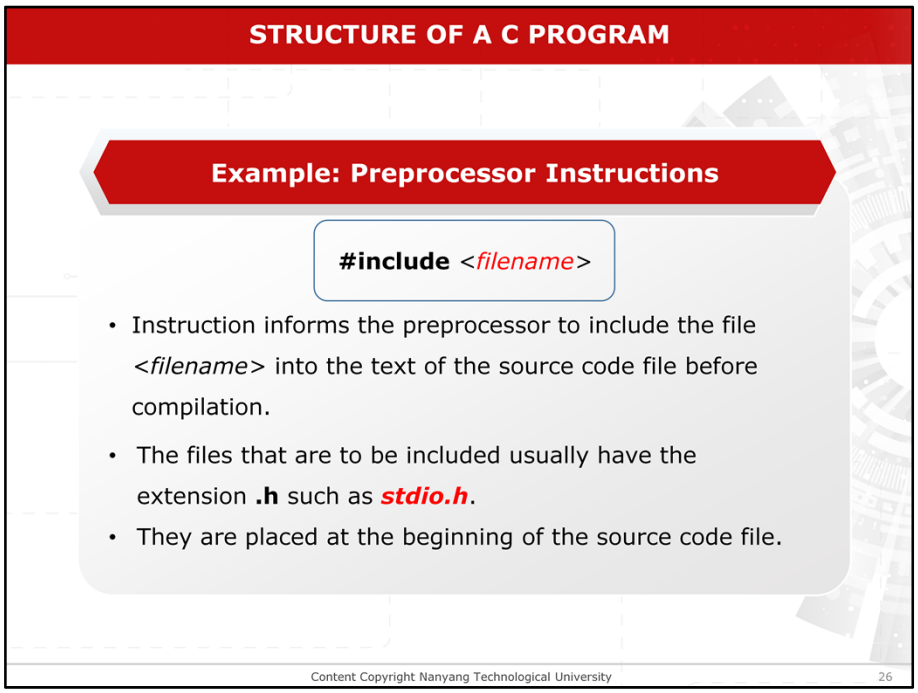
the **#include** *<filename>*



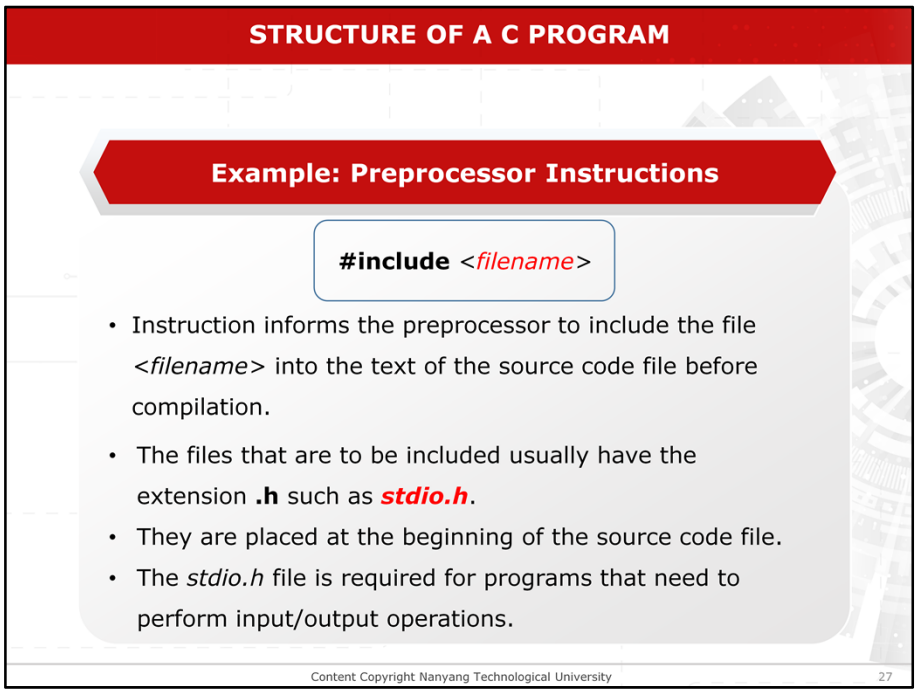
informs the preprocessor to include the file *<filename>* into the text of the source code file before compilation.



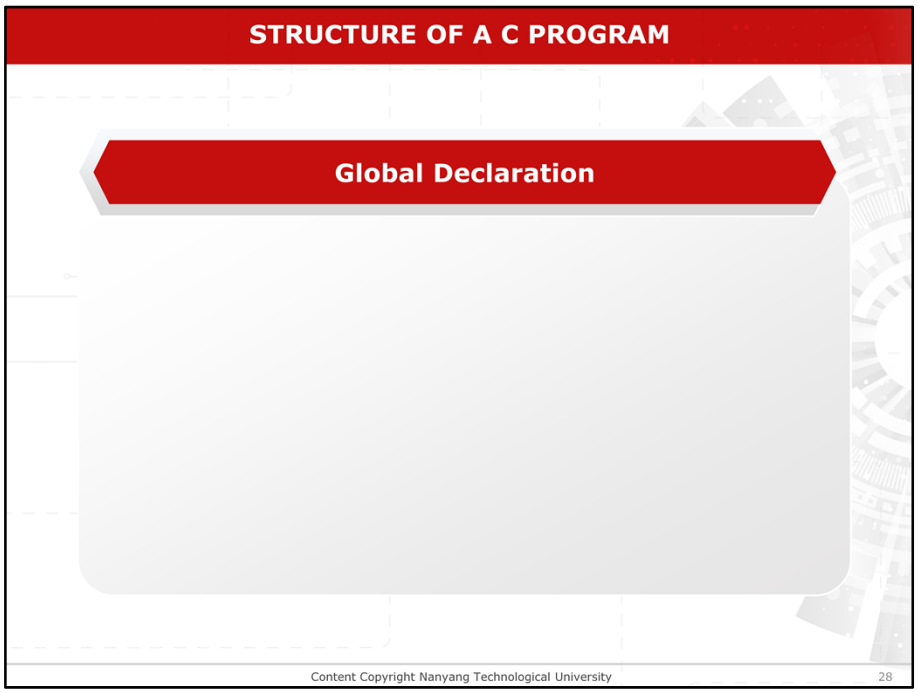
The files that are to be included usually have the extension (dot h) such as *s t d i o dot h*.



They are placed at the beginning of the source code file.

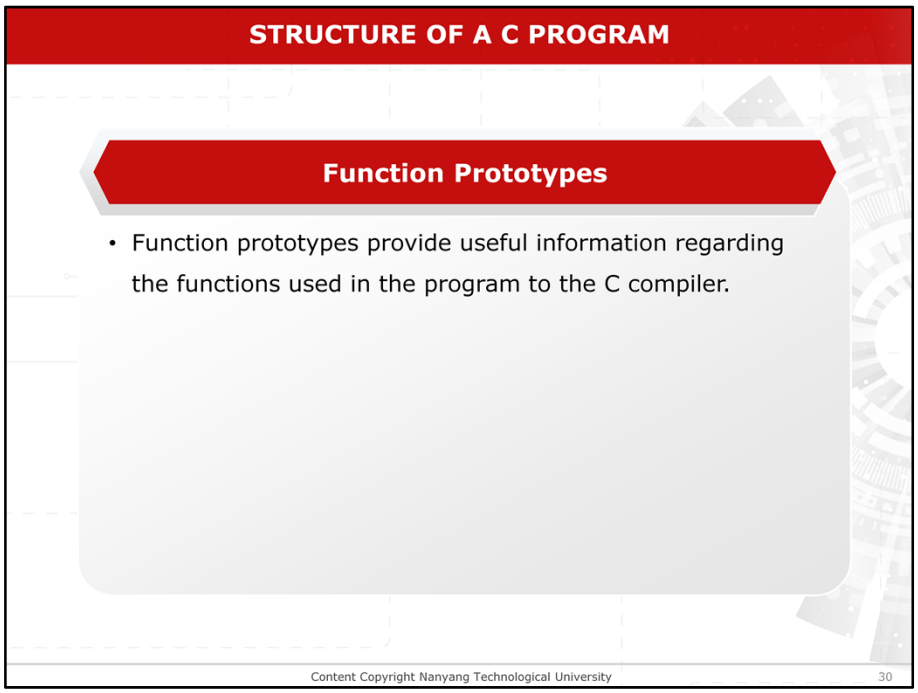


The *stdio.h* file is required for programs that need to perform input/output operations.

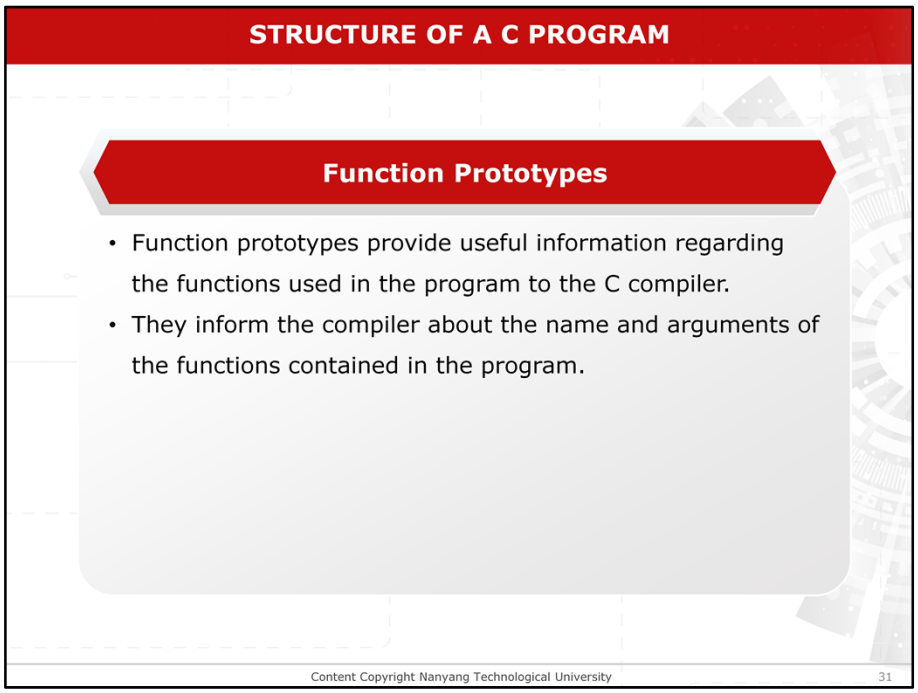


Global declaration
[Pause: 1 sec]

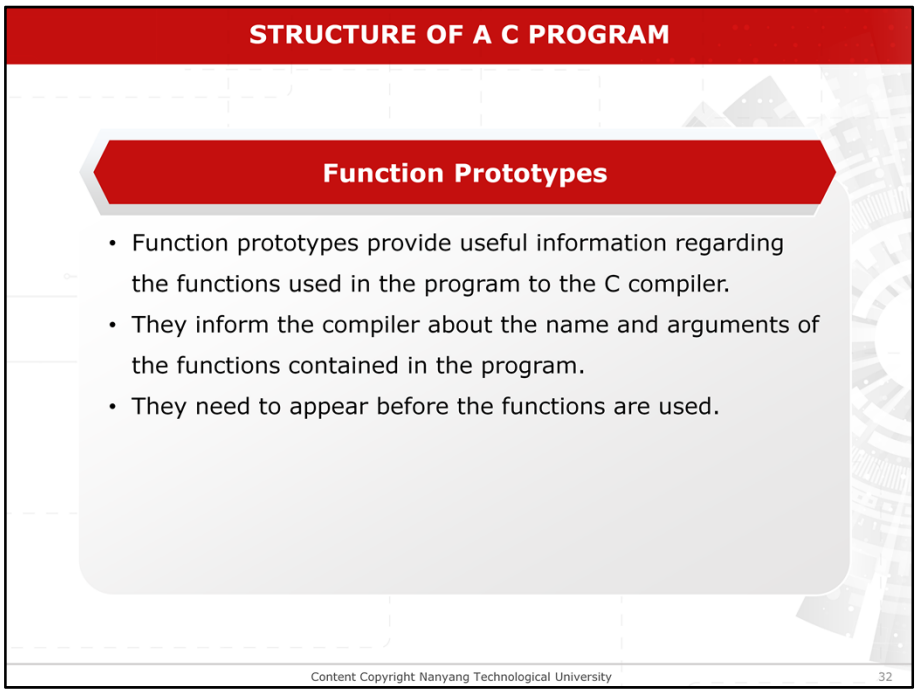
Declares global variables that are accessible anywhere within a C program
[Pause: 1 sec]



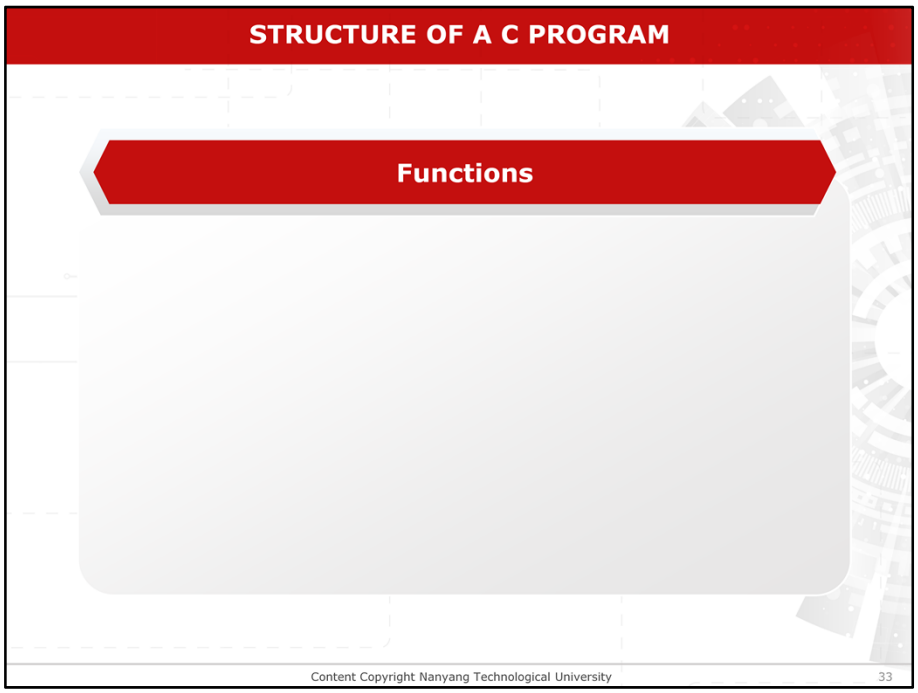
Function prototypes provide useful information regarding the functions used in the program to the C compiler.
[Pause: 1sec]



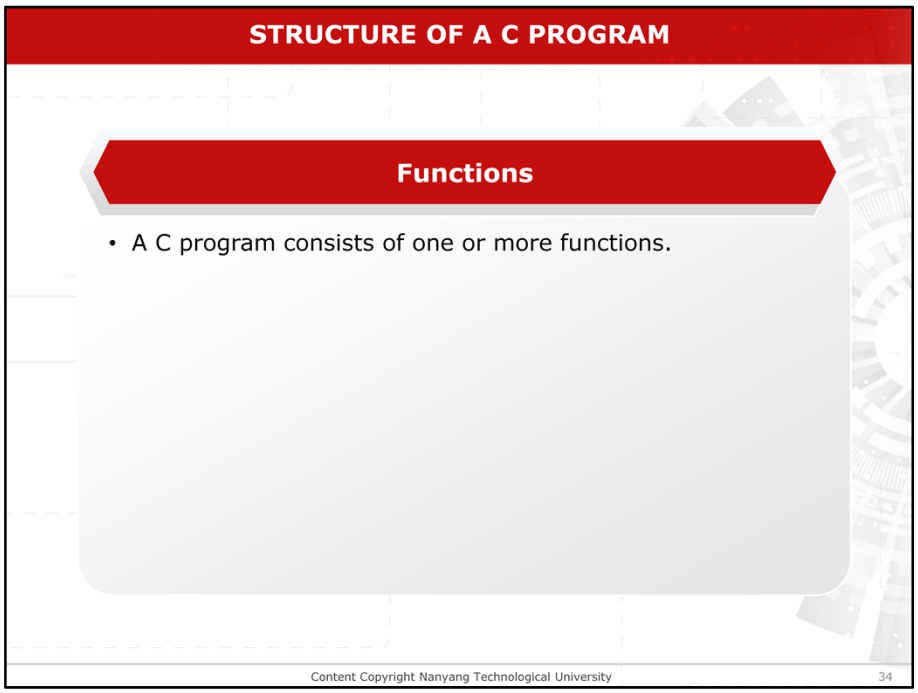
They inform the compiler about the name and arguments of the functions contained in the program.
[Pause: 1sec]



They need to appear before the functions are used.
[Pause: 1sec]

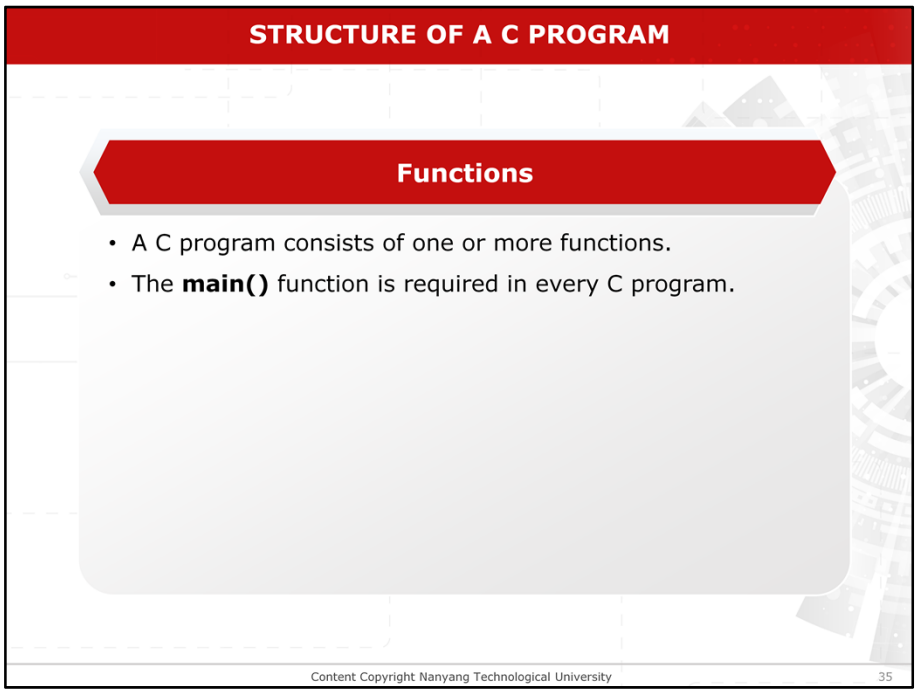


They need to appear before the functions are used.
[Pause: 1sec]



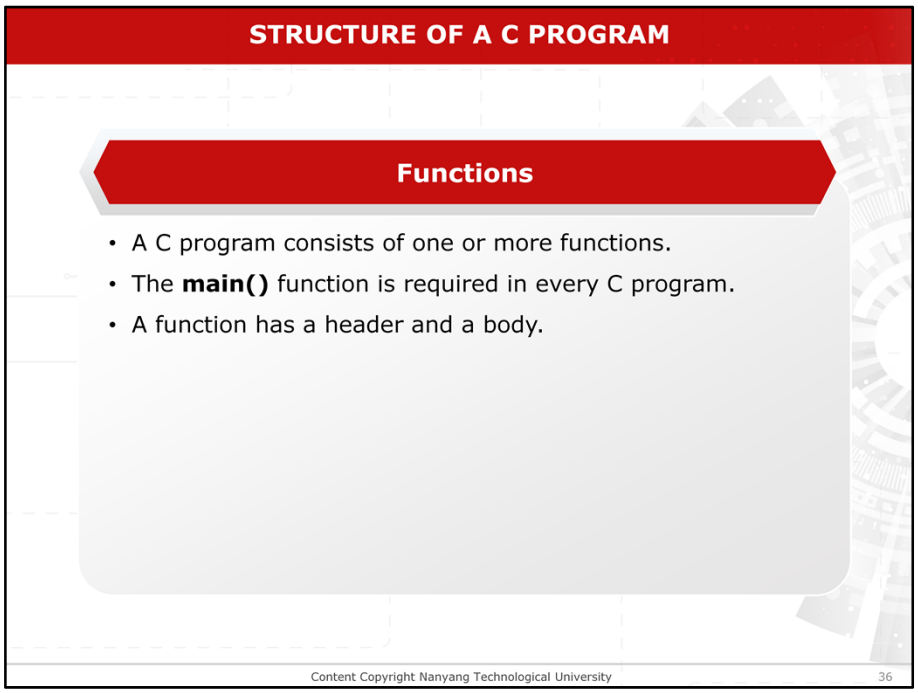
A C program consists of one or more functions.

[Pause: 1sec]



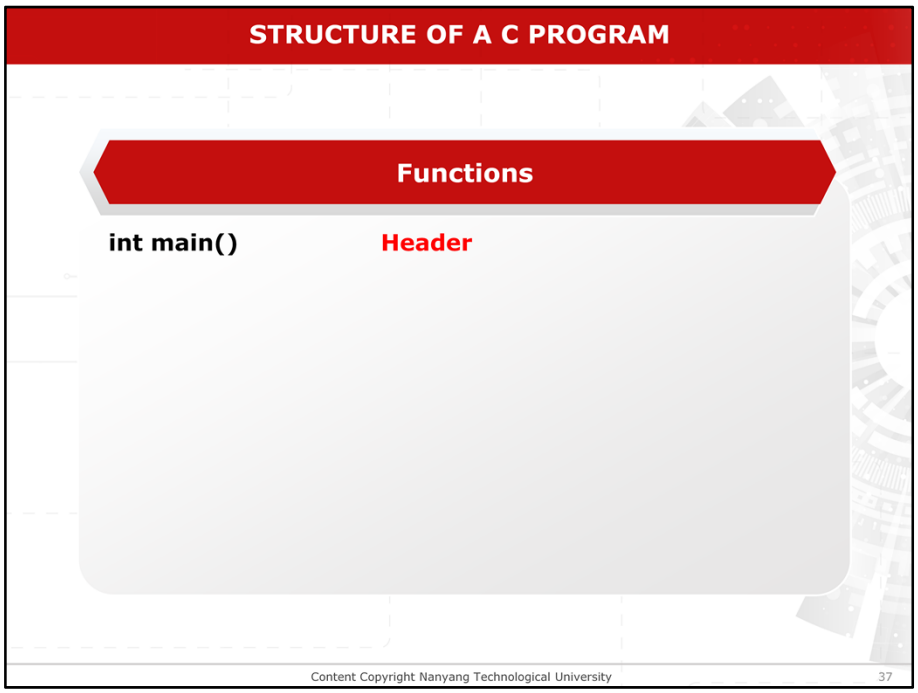
The **main()** function is required in every C program.

[Pause: 1sec]

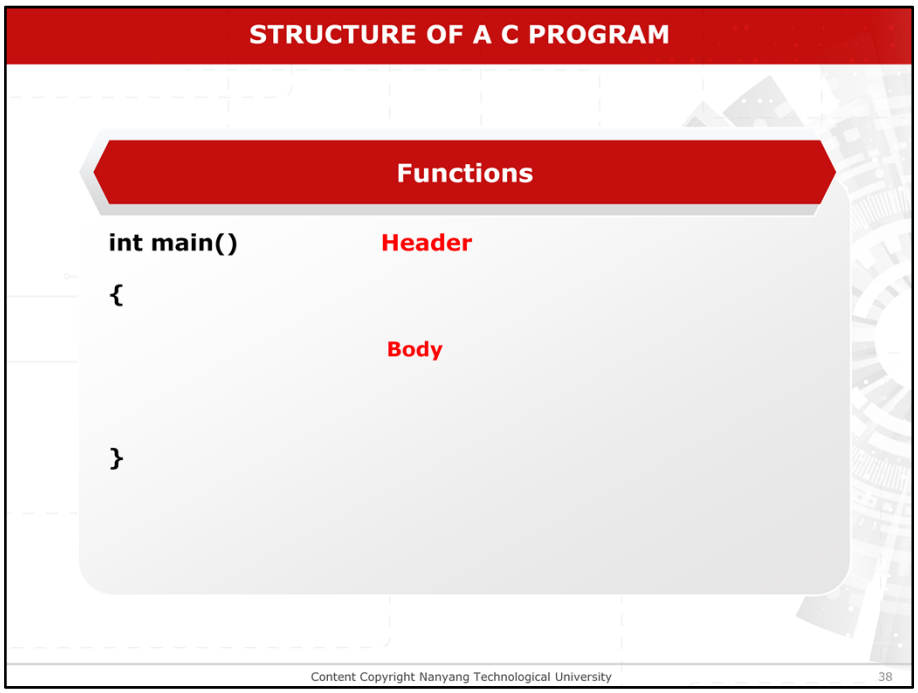


A function has a header and a body.

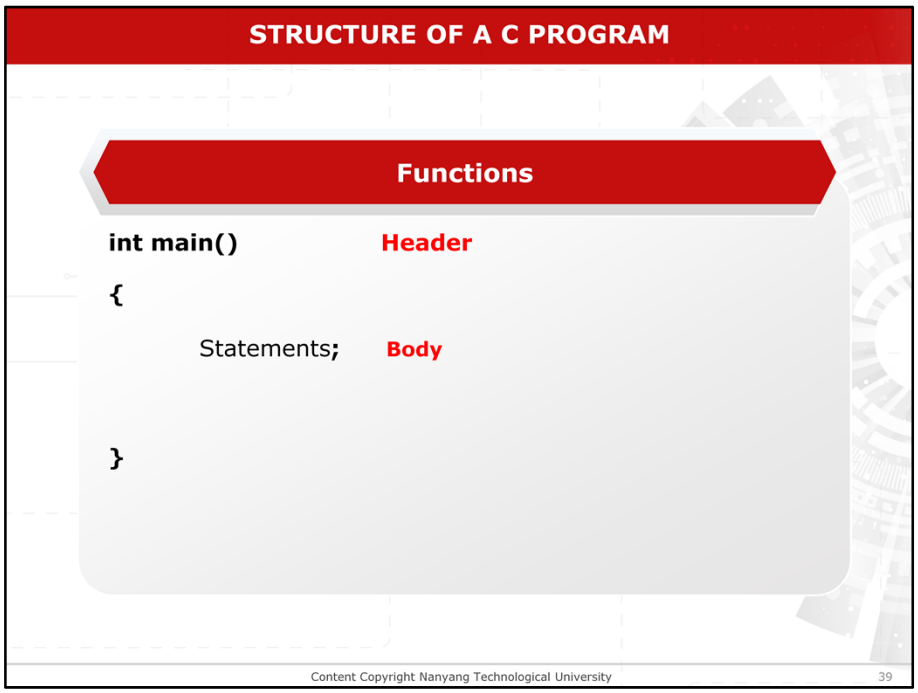
[Pause: 1sec]



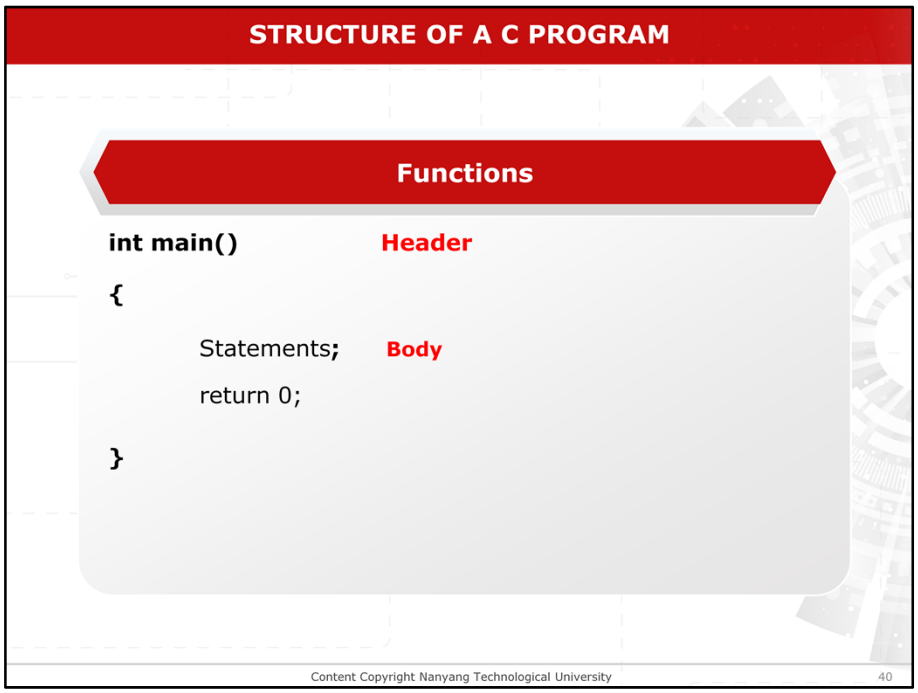
The header is int main ( ).The function header defines the name of the function, and the inputs expected by the function.
Pause: 1 sec



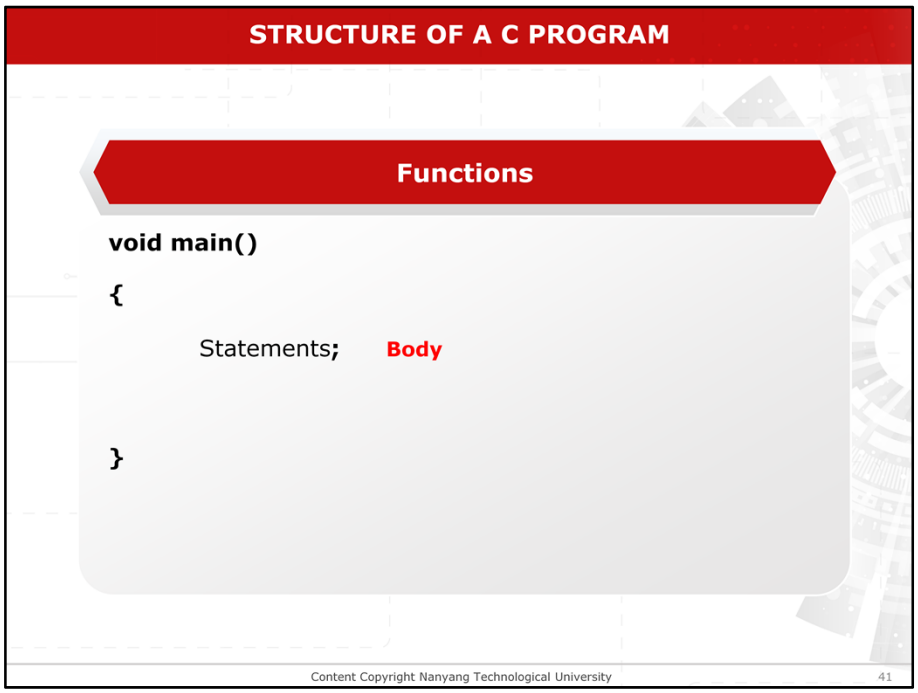
The body of the int main() function is enclosed by the braces { }.
Pause: 1 sec



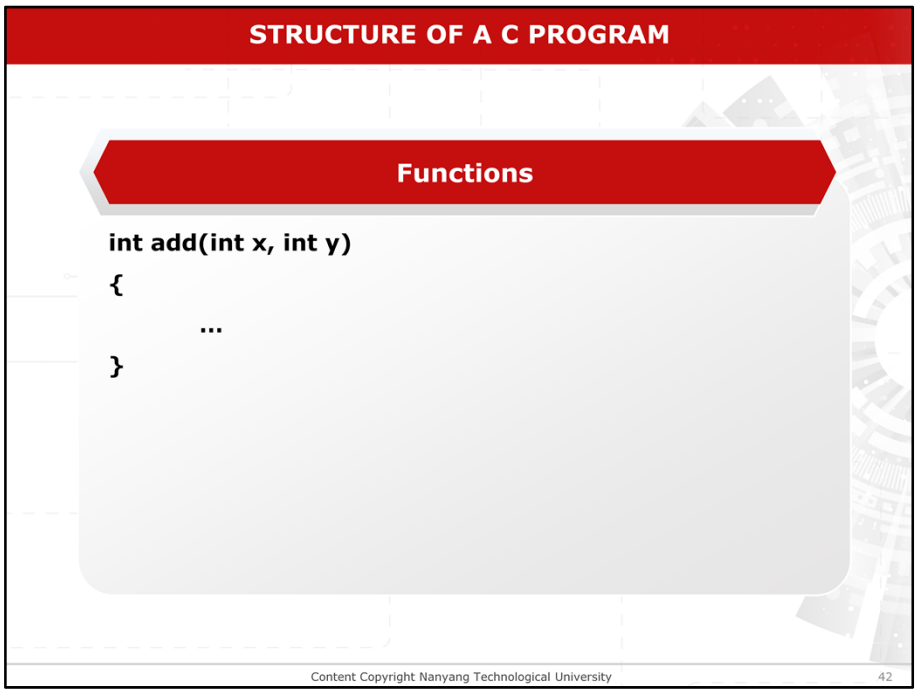
The function body contains statements that are written to perform a specific task.
The body will be executed when the function is called.
Program execution starts from the beginning of the body.
Pause: 1 sec



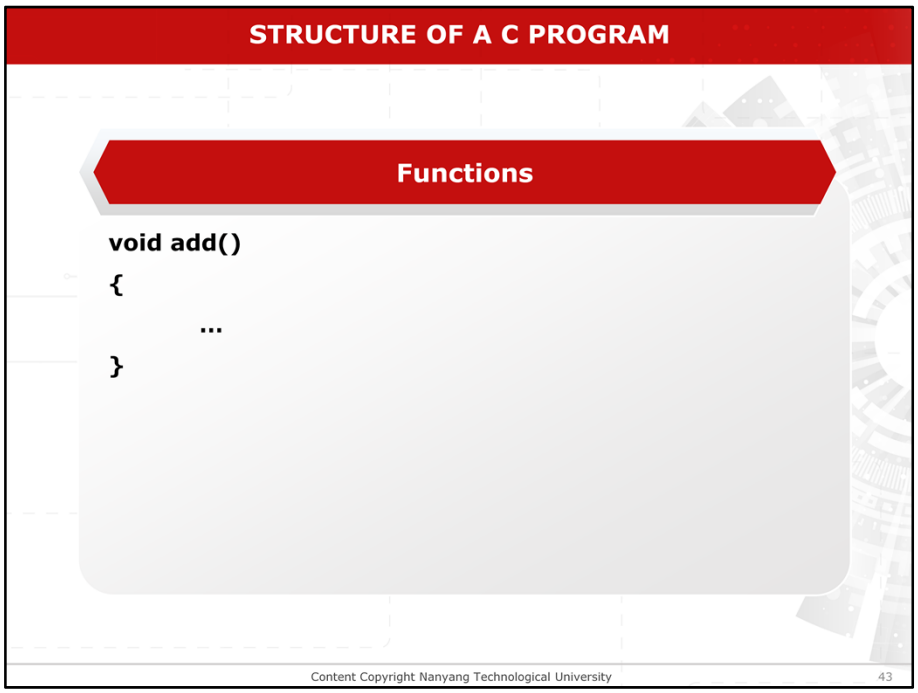
The statement **return 0** can be the last statement of the int **main()** function depending on whether the function returns any values.
Pause: 1 sec



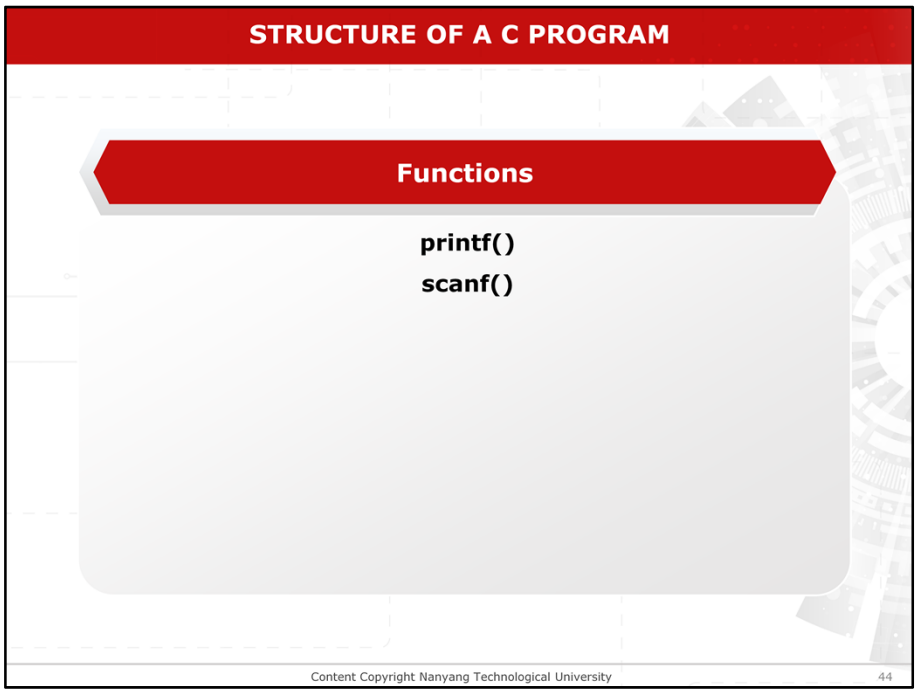
The statement **void main()** does not require a **return** statement in the **main()** body.



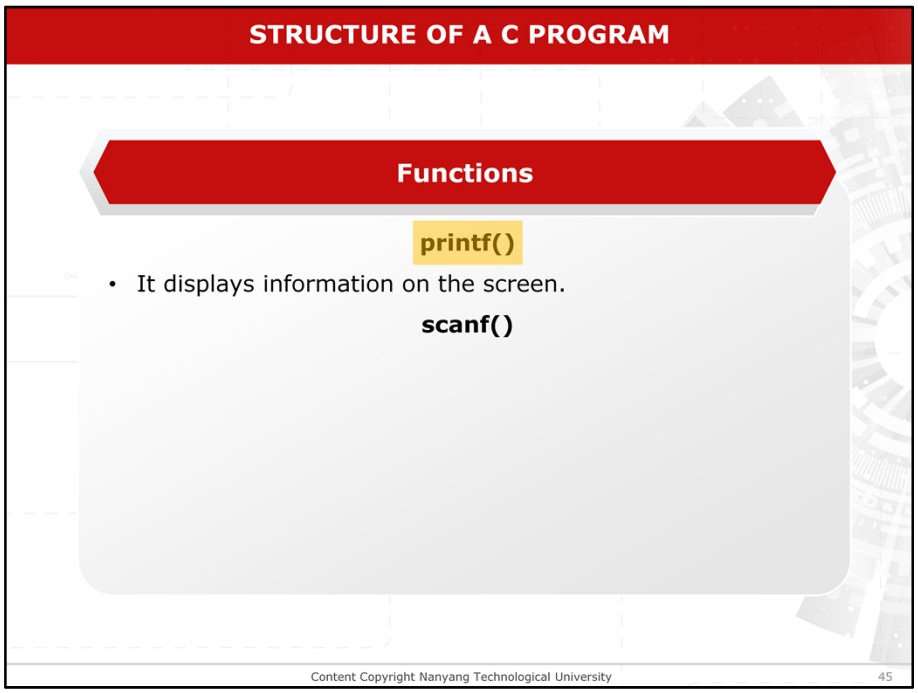
A function may or may not have arguments, and it may or may not have a return value. This function has two arguments and requires a return value.



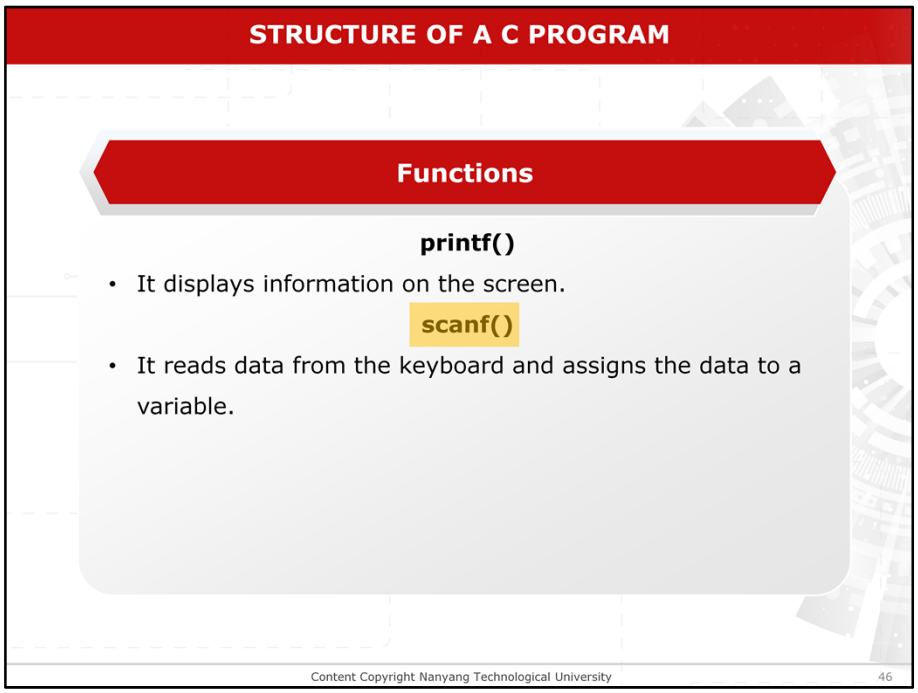
while the function **void add() { … }** has no argument and does not require a return value.



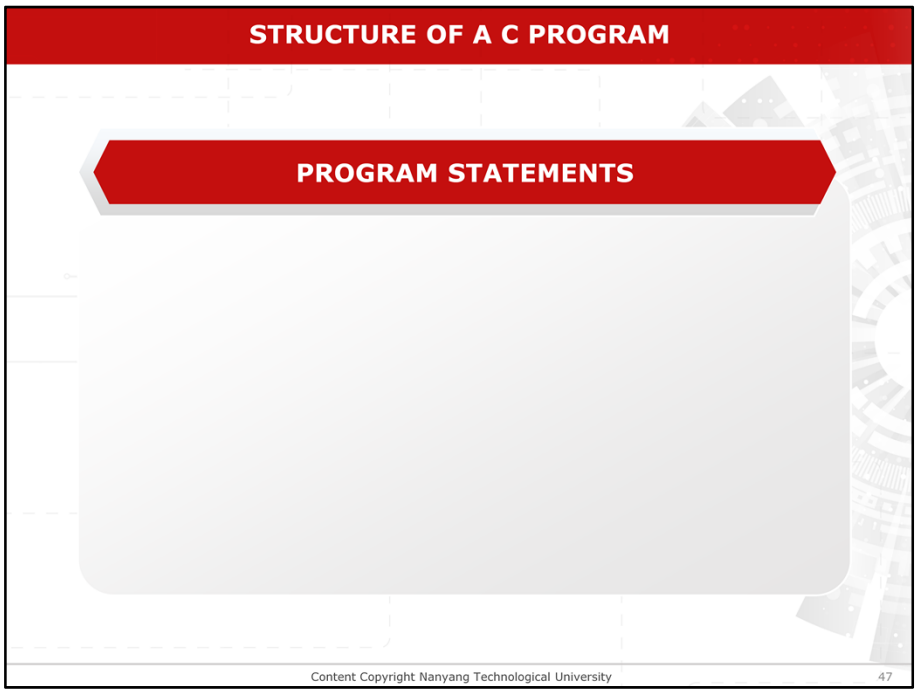
The **printf()** and **scanf()** statements are the most commonly used library functions for input/output operations.



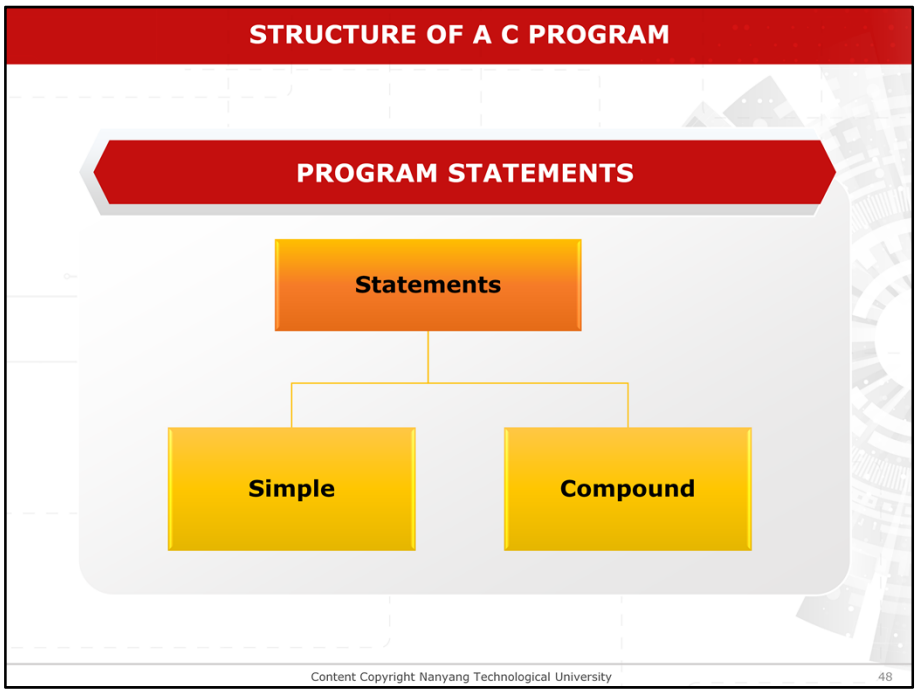
The **printf()** statement displays information on the screen,



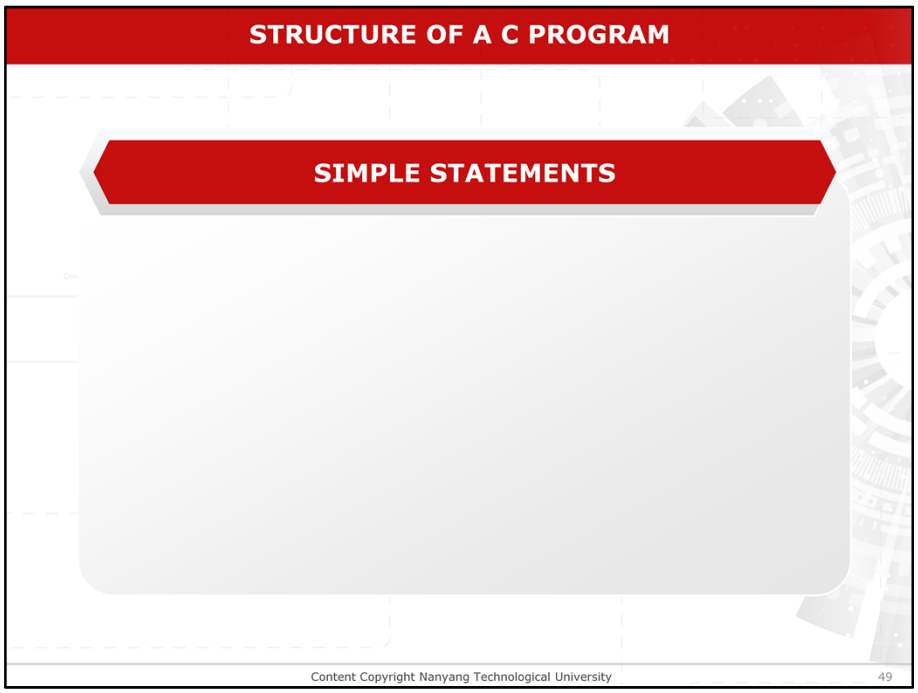
while the **scanf()** statement reads data from the keyboard and assigns the data to a variable.



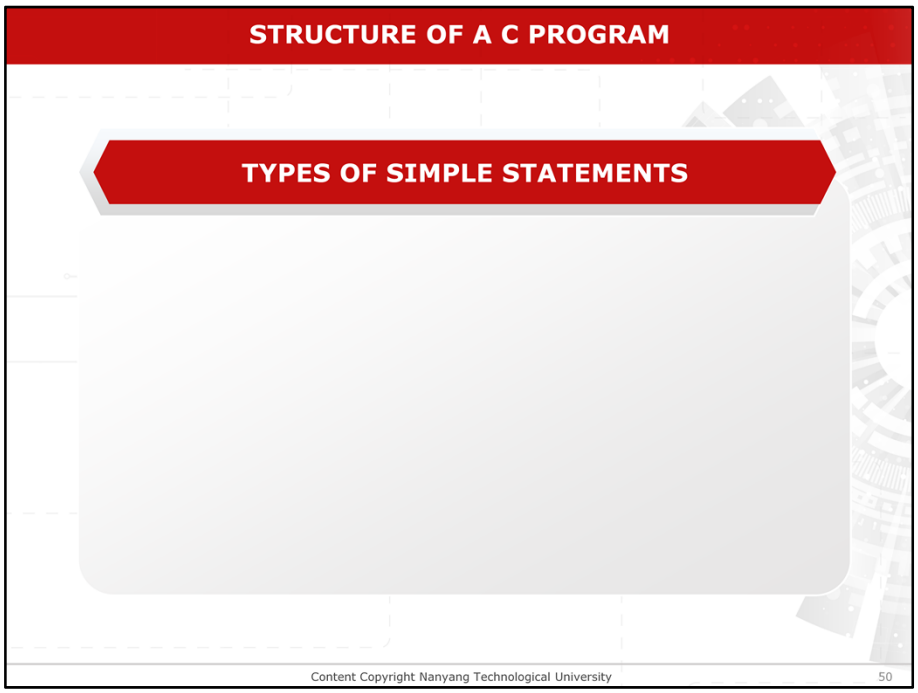
A statement is a command to the computer.



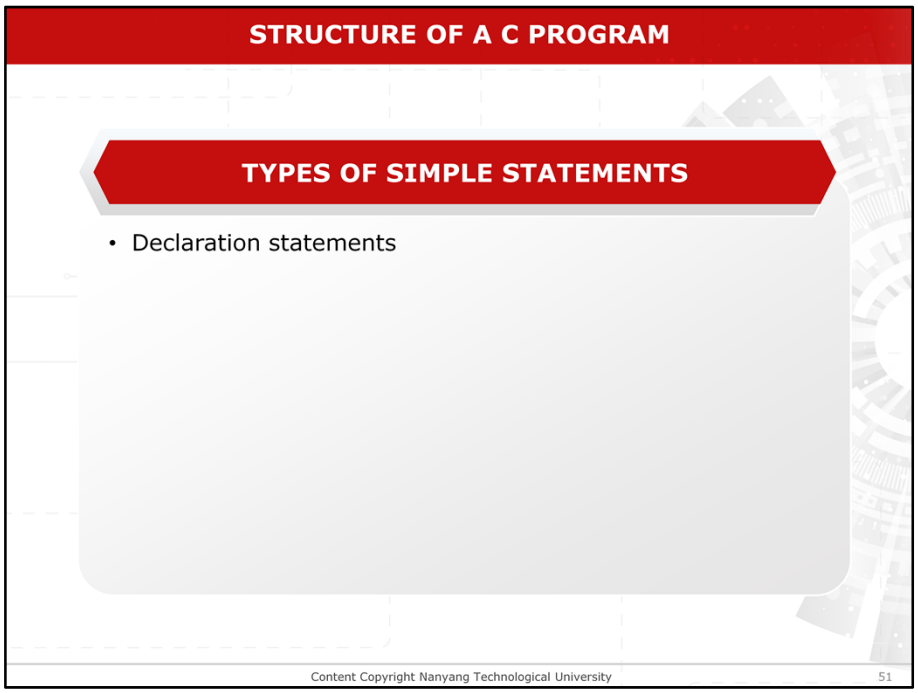
There are two types of statements: simple or compound statements.



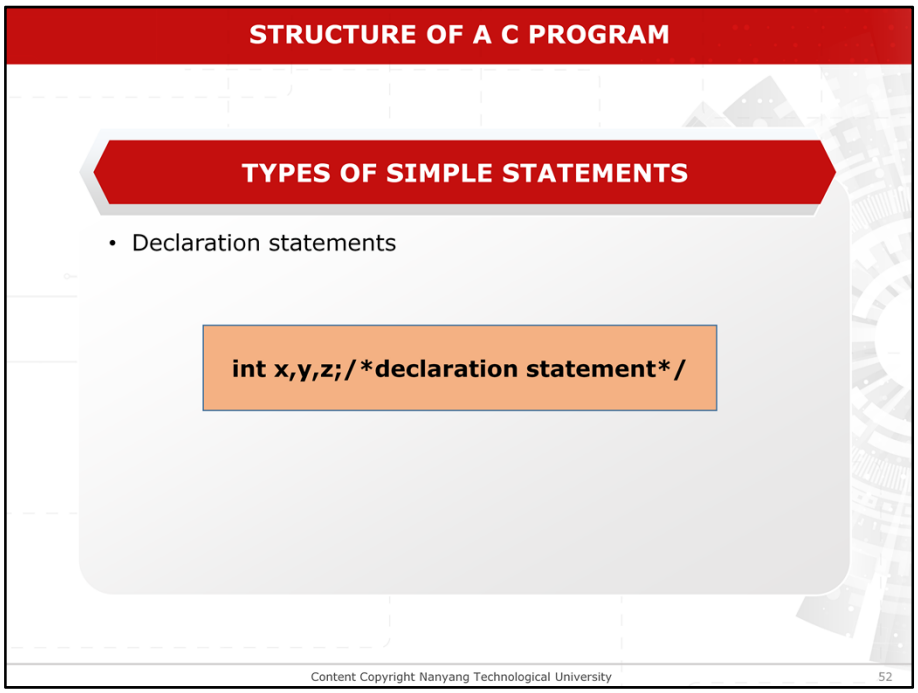
A simple statement is a statement terminated by a semicolon.



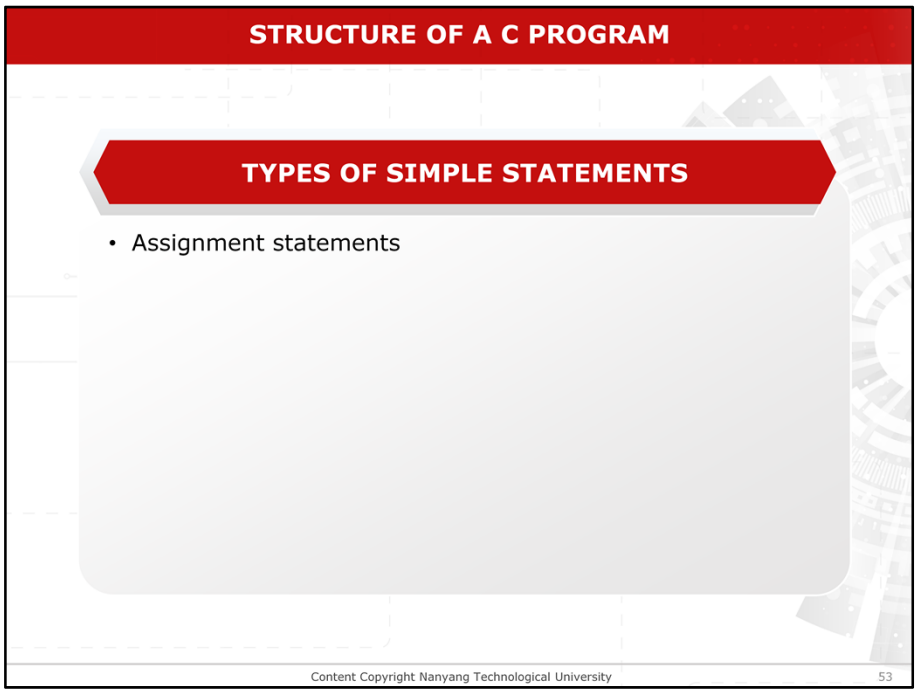
There are four types of simple statements:



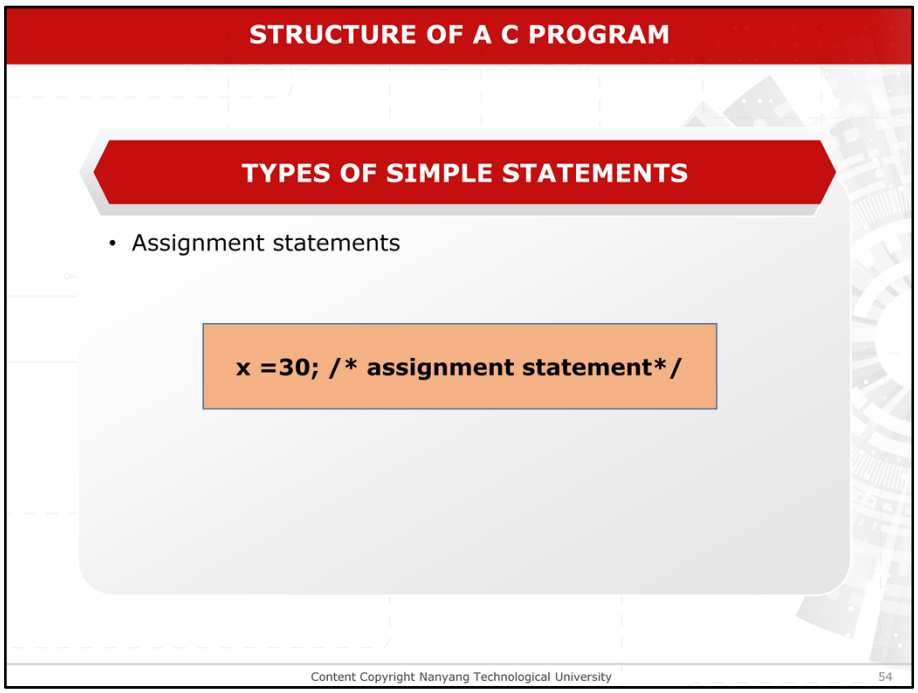
Declaration statement



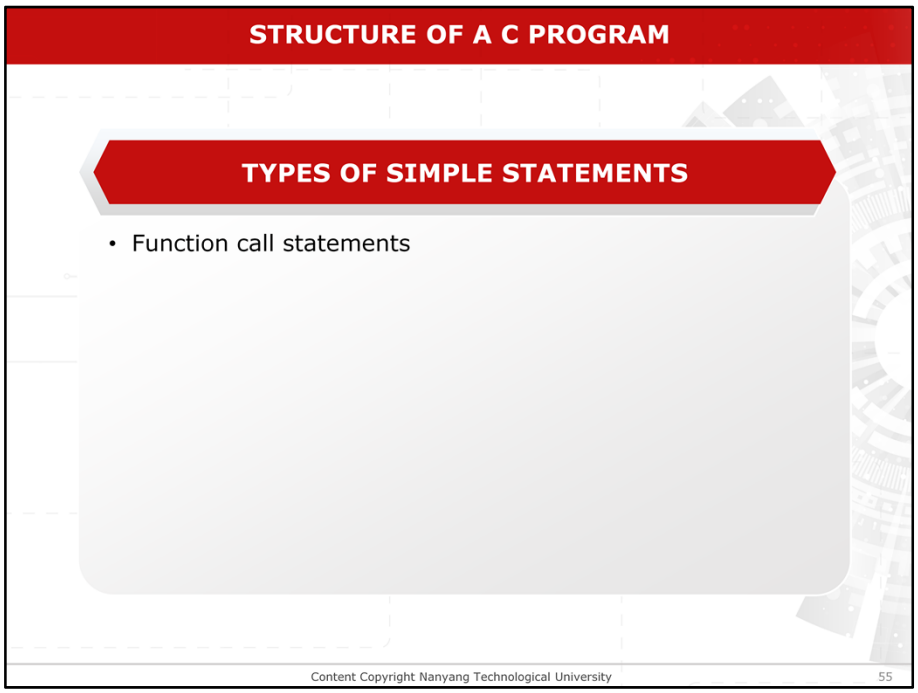
This statement declares x, y, z to be integers.



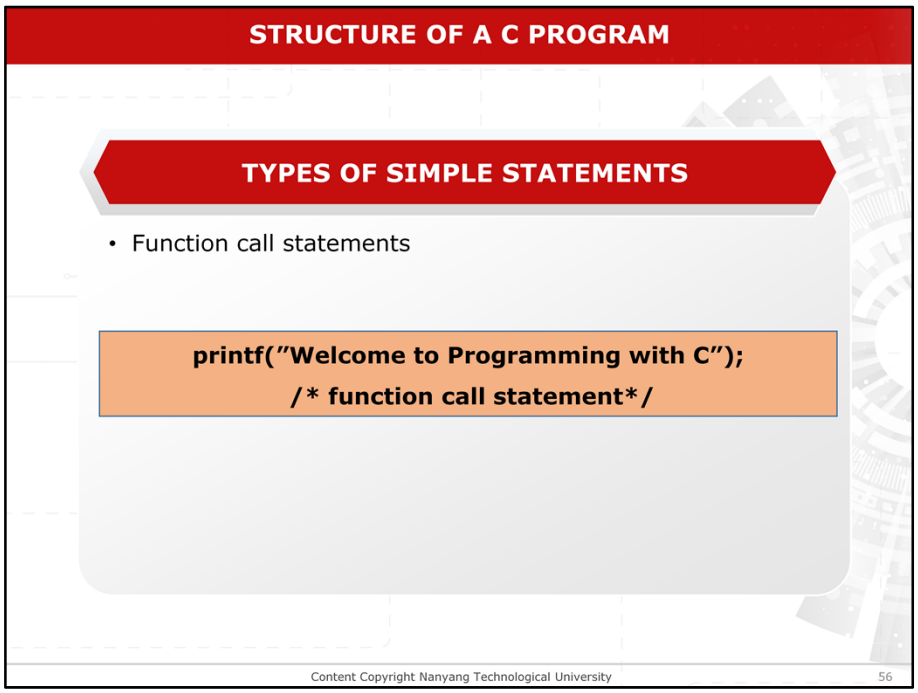
Assignment statements



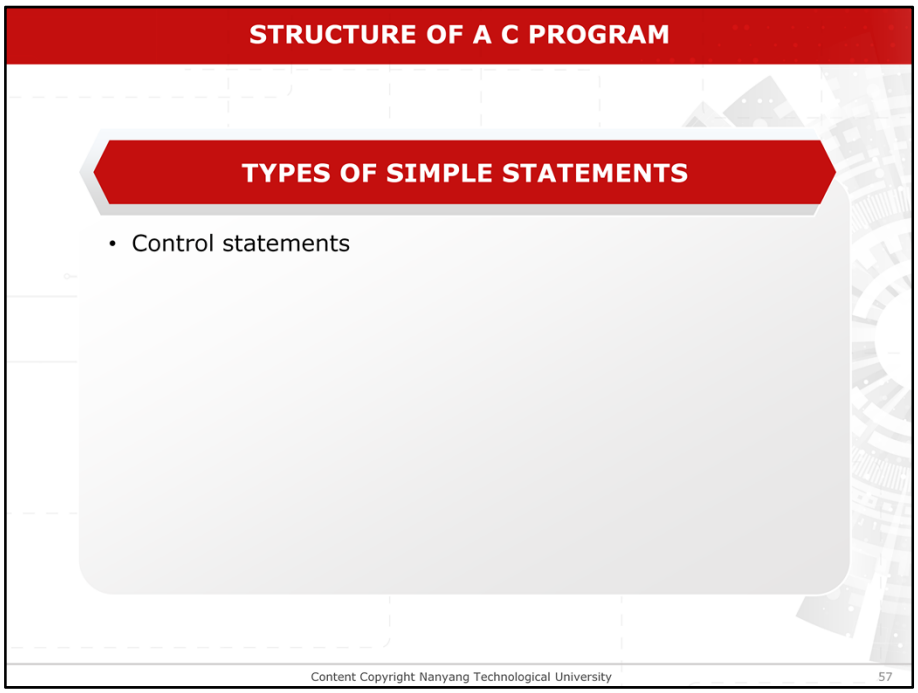
A value of 30 is assigned to x.



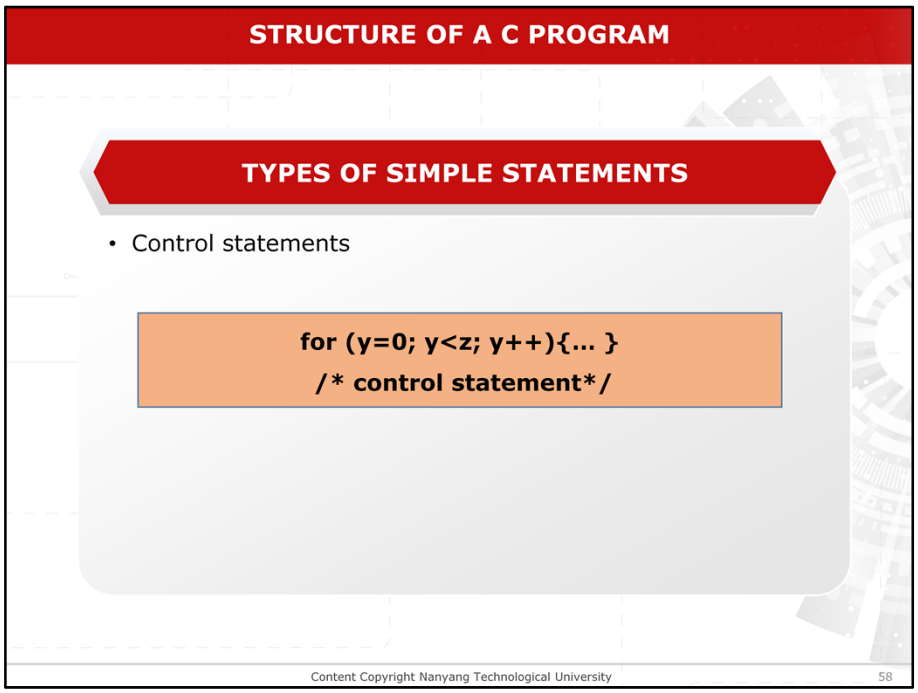
Function call statements



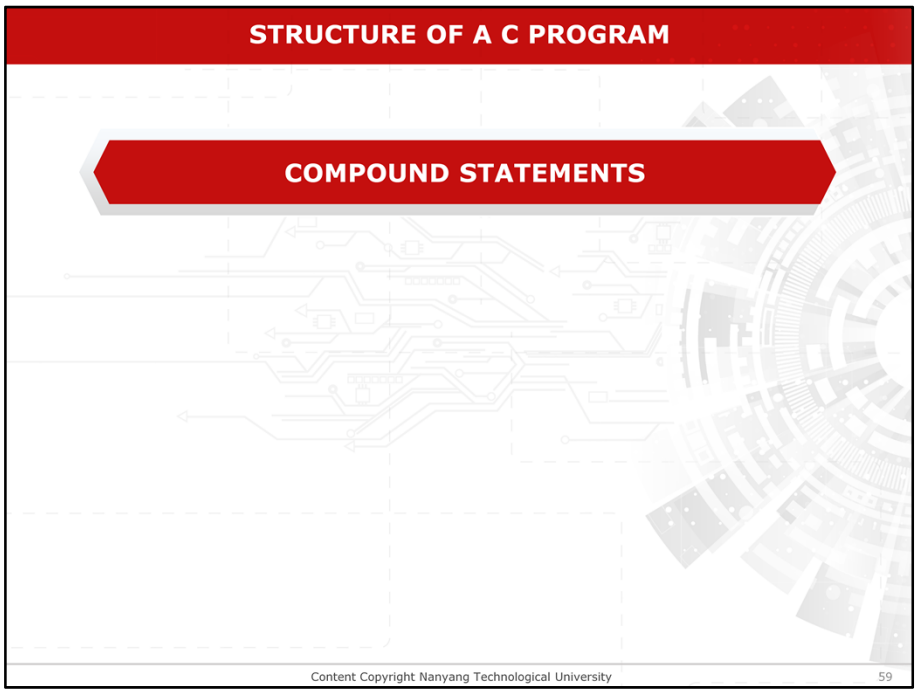
This is an example of a function call statement.



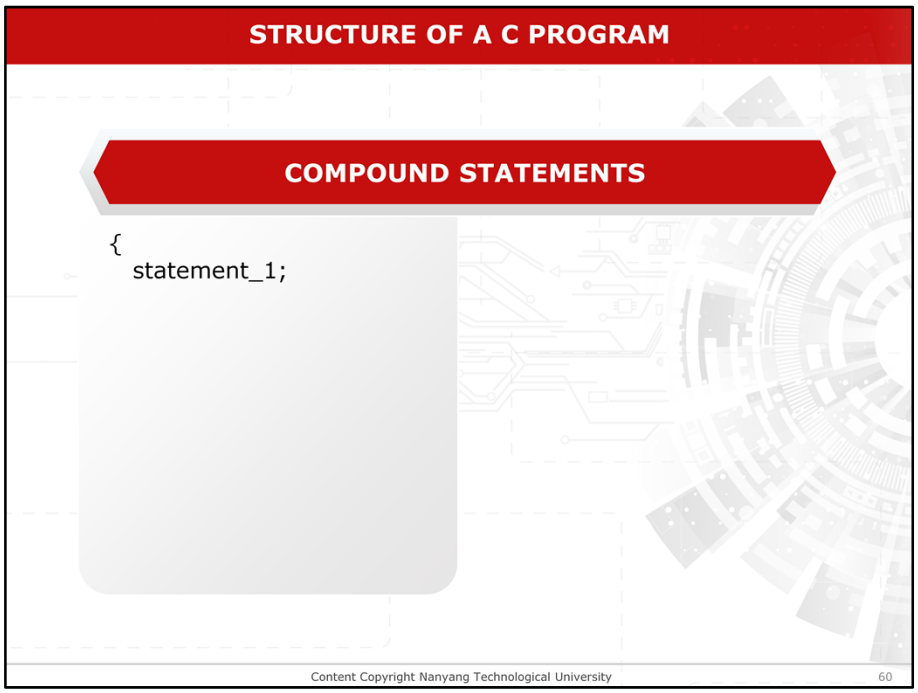
Control statements



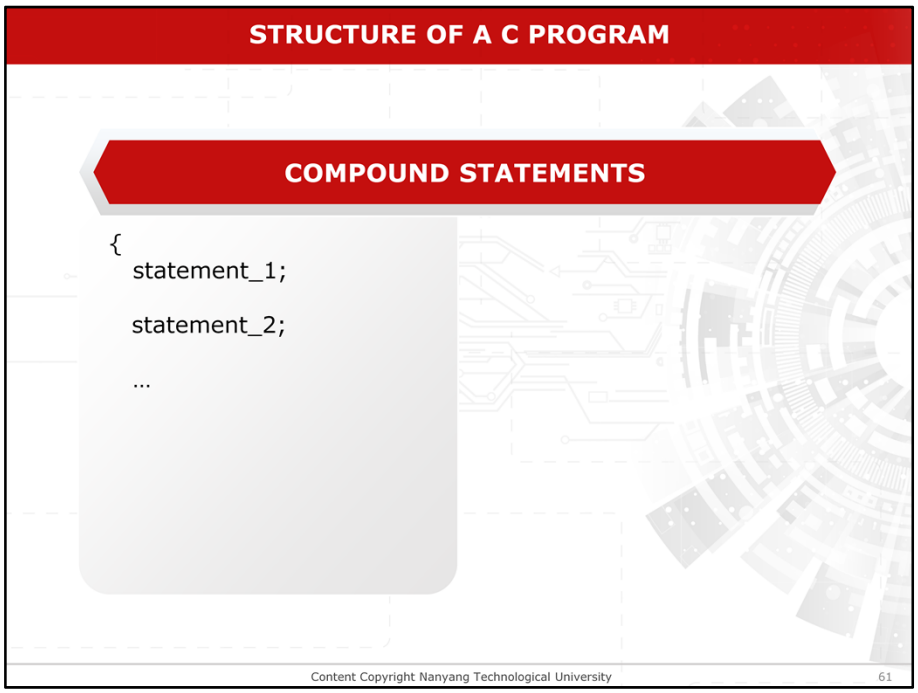
This is an example of control statement.



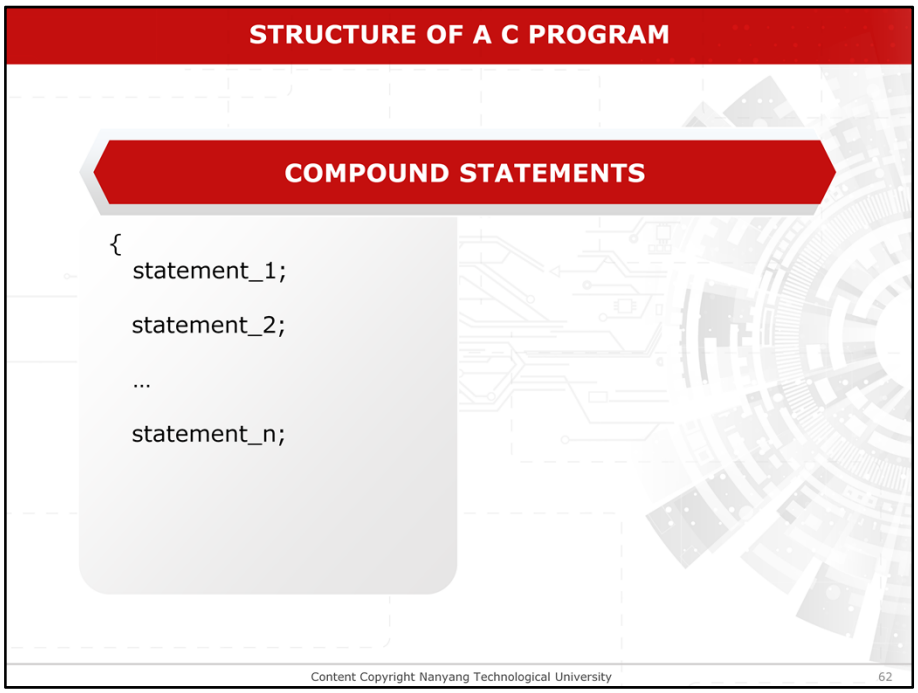
Compound statement



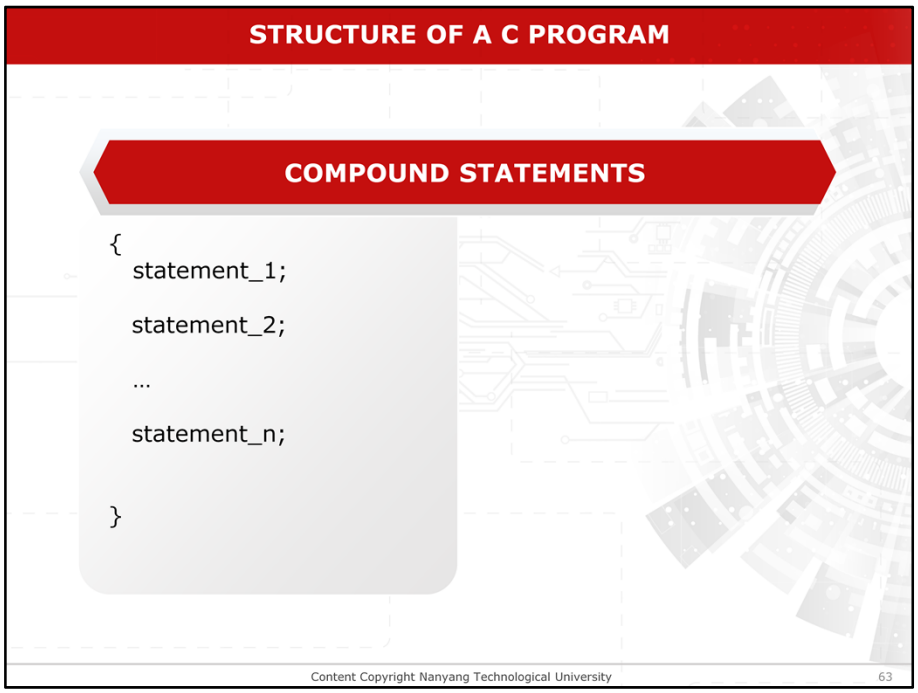
A compound statement is a sequence of one or more statements enclosed in braces



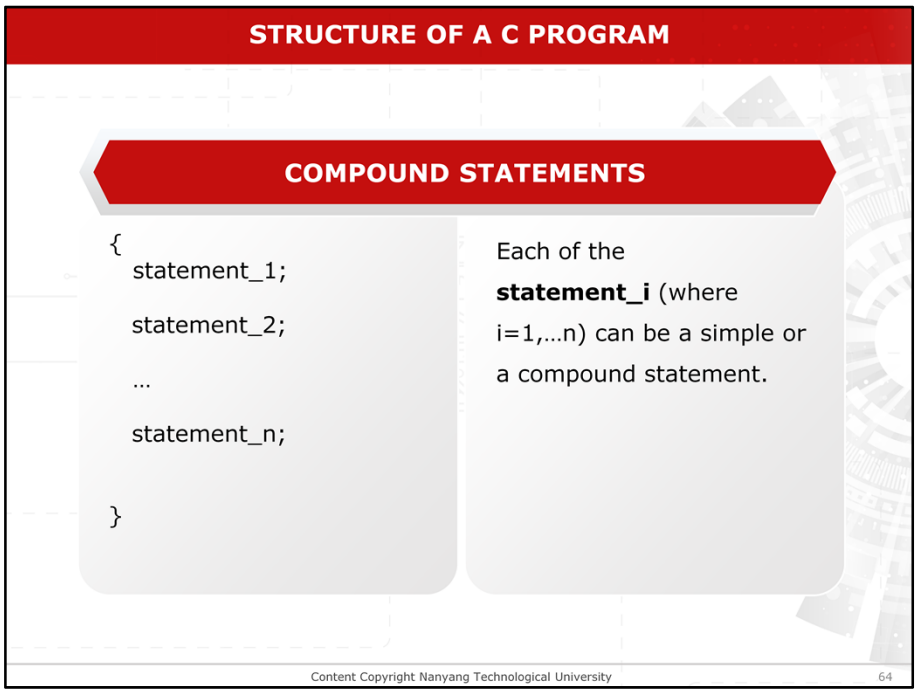
A compound statement is a sequence of one or more statements enclosed in braces



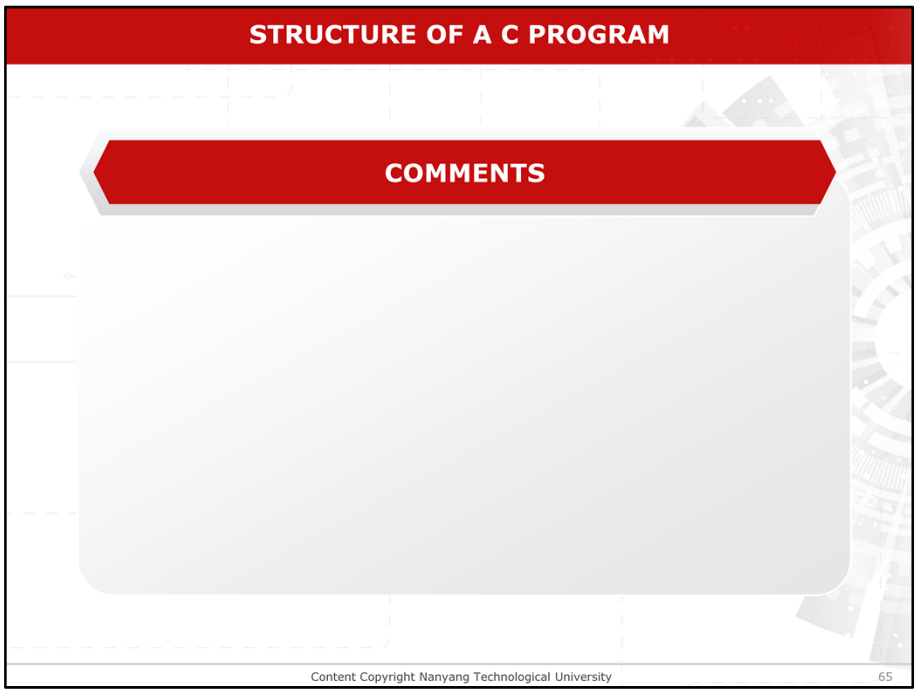
A compound statement is a sequence of one or more statements enclosed in braces

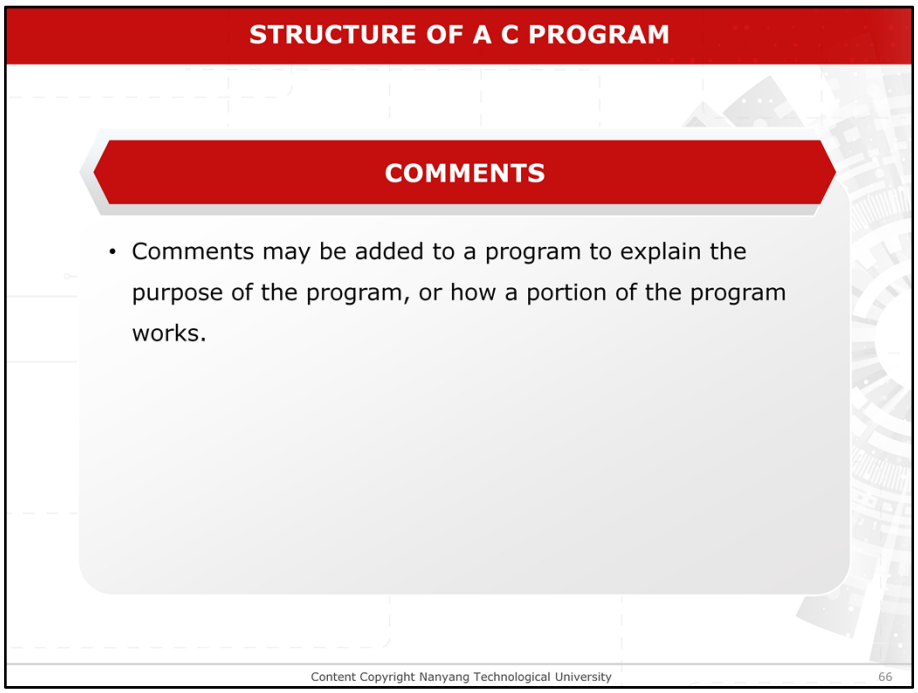


A compound statement is a sequence of one or more statements enclosed in braces

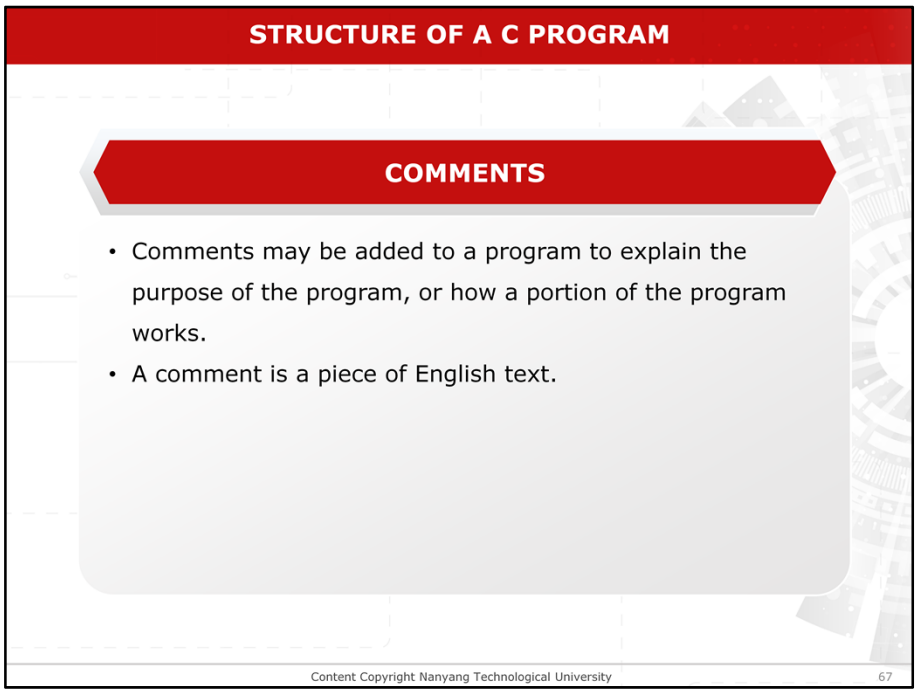


Each of the statement can be a simple or a compound statement.

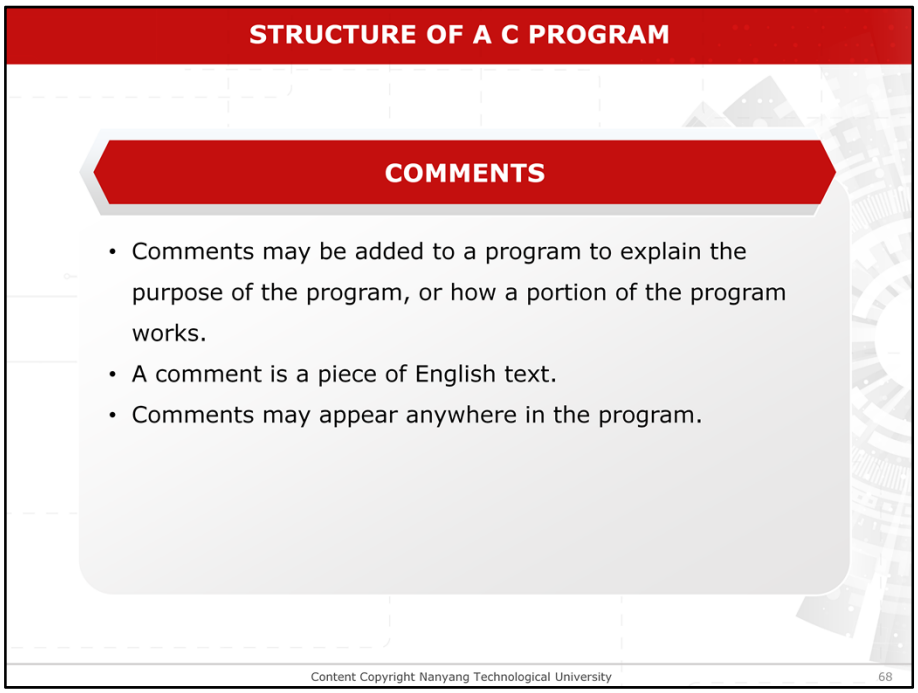

STRUCTURE OF A C PROGRAM
COMMENTS
Content Copyright Nanyang Technological University
65



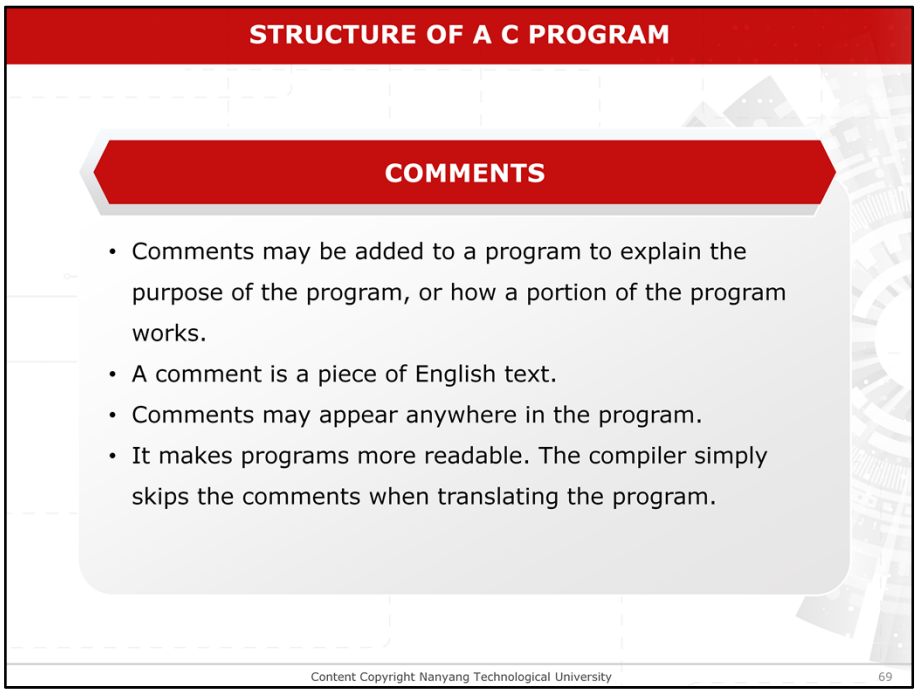
Comments may be added to a program to explain the purpose of the program, or how a portion of the program works.



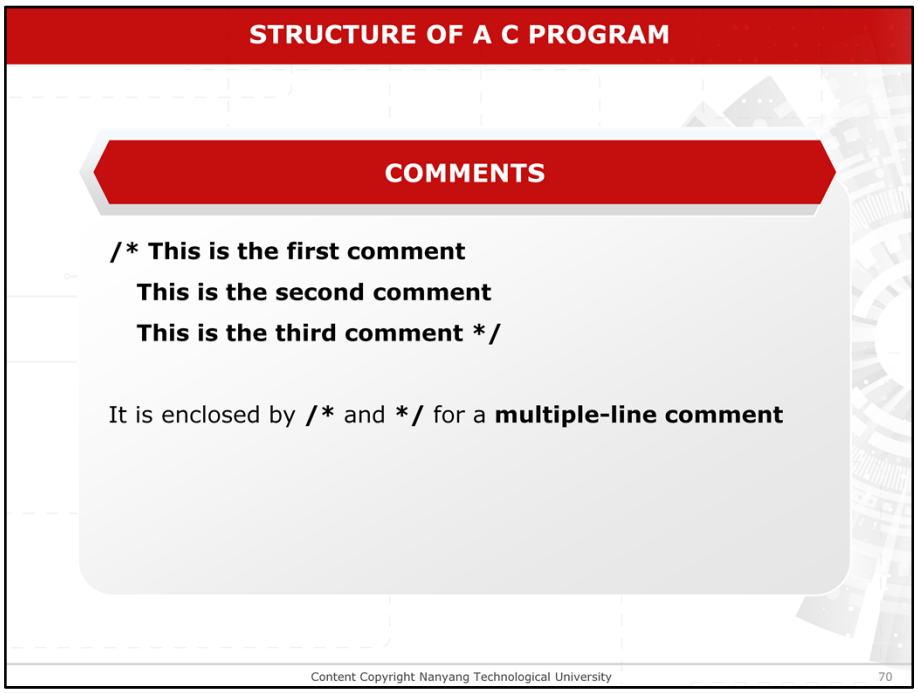
A comment is a piece of English text.



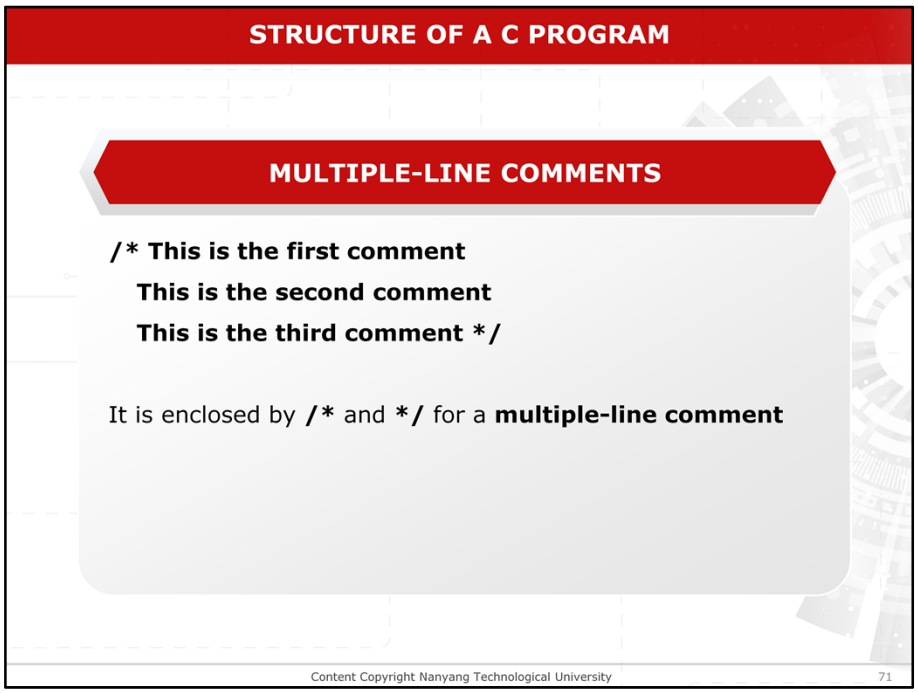
Comments may appear anywhere in the program.



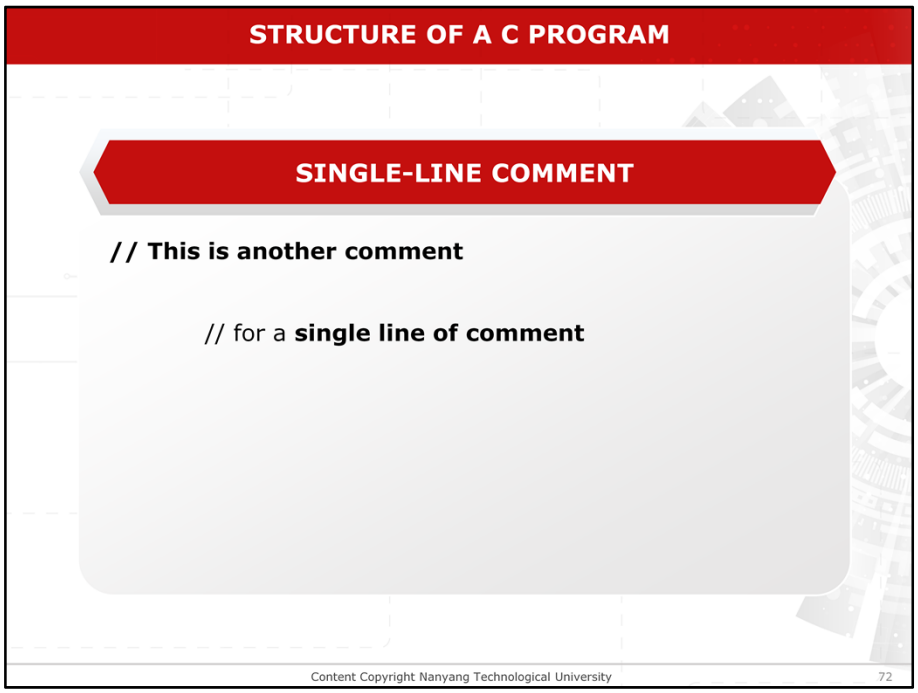
It makes programs more readable. The compiler simply skips the comments when translating the program.



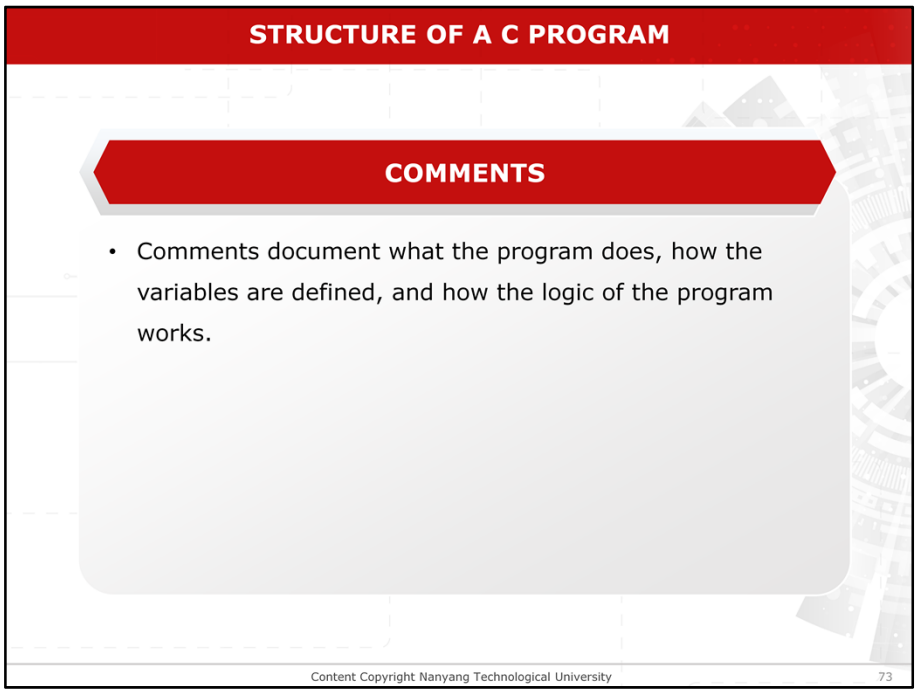
It is enclosed by **/*** and ***/** for a **multiple-line comment**



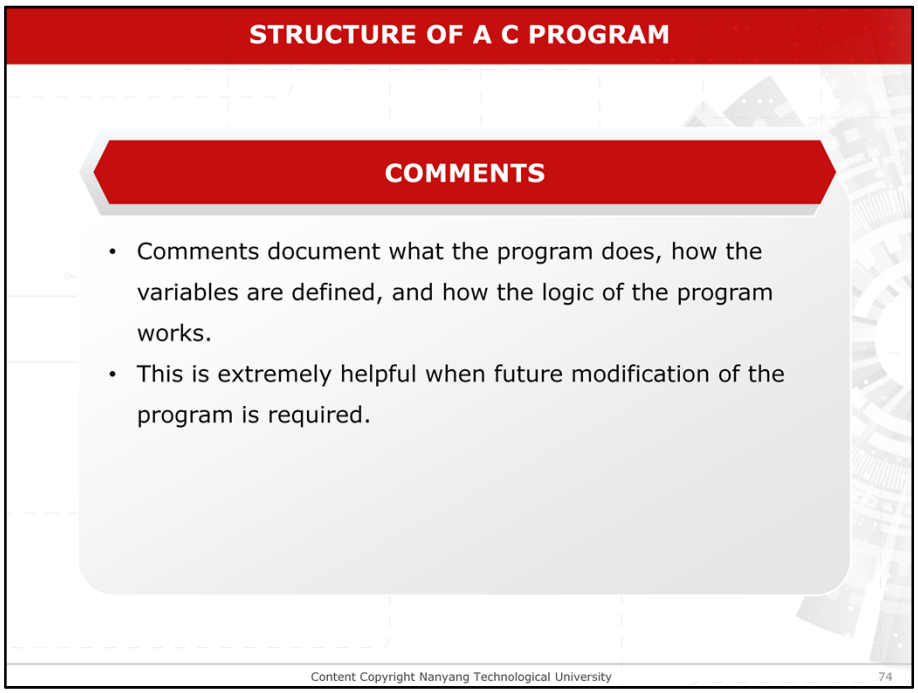
It is enclosed by **/*** and ***/** for a **multiple-line comment**



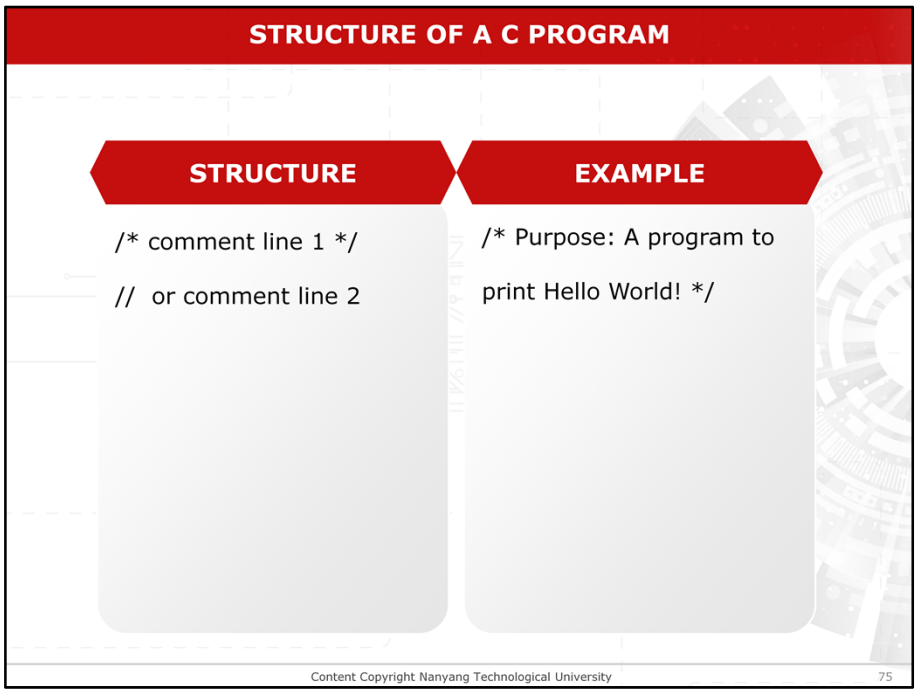
Single line comment is written with double slash in the beginning of the comment.



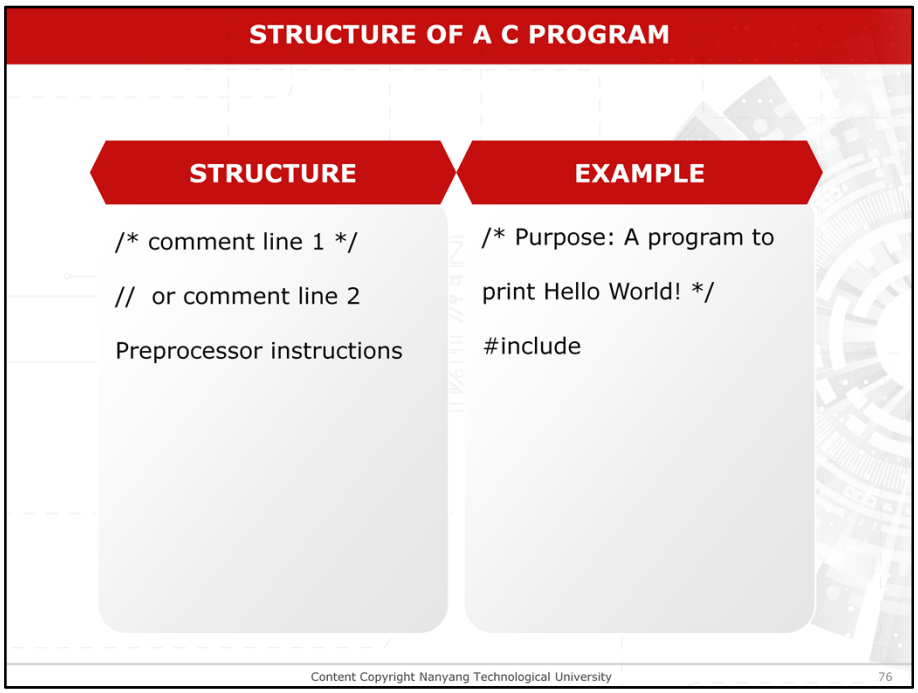
It is important to have comments in the programs. Comments document what the program does, how the variables are defined, and how the logic of the program works.



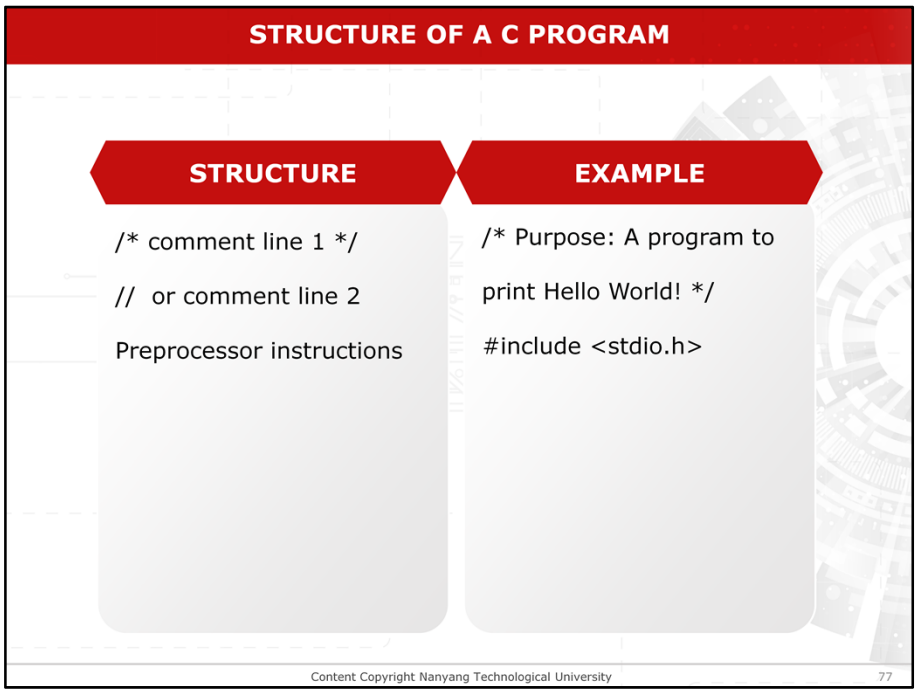
This is extremely helpful when future modification of the program is required. Therefore, it is necessary to develop the habit of using comments to document the programs including the use of comments for programming structures and operations.



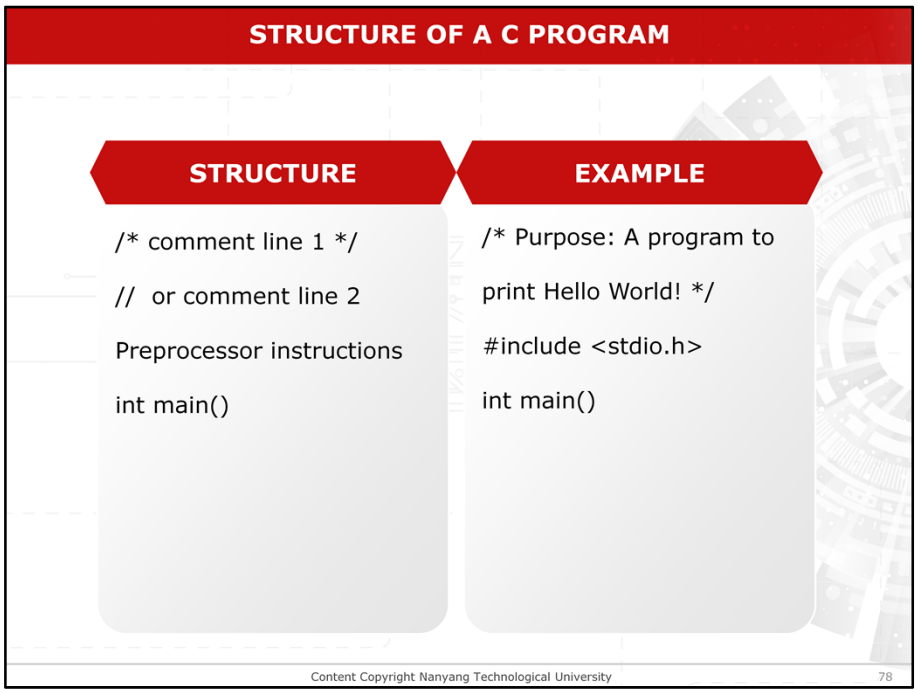
In the example program, the first two lines are the comments which state the purpose of the program.



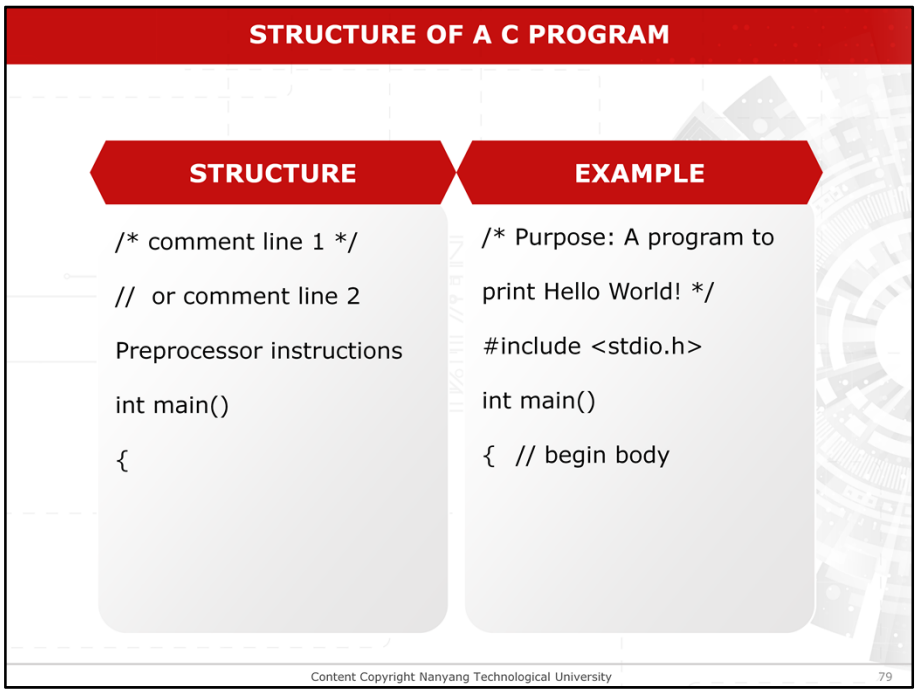
The **#include** preprocessor directive instructs the C compiler to add the contents of the file *stdio.h* into the program during compilation.



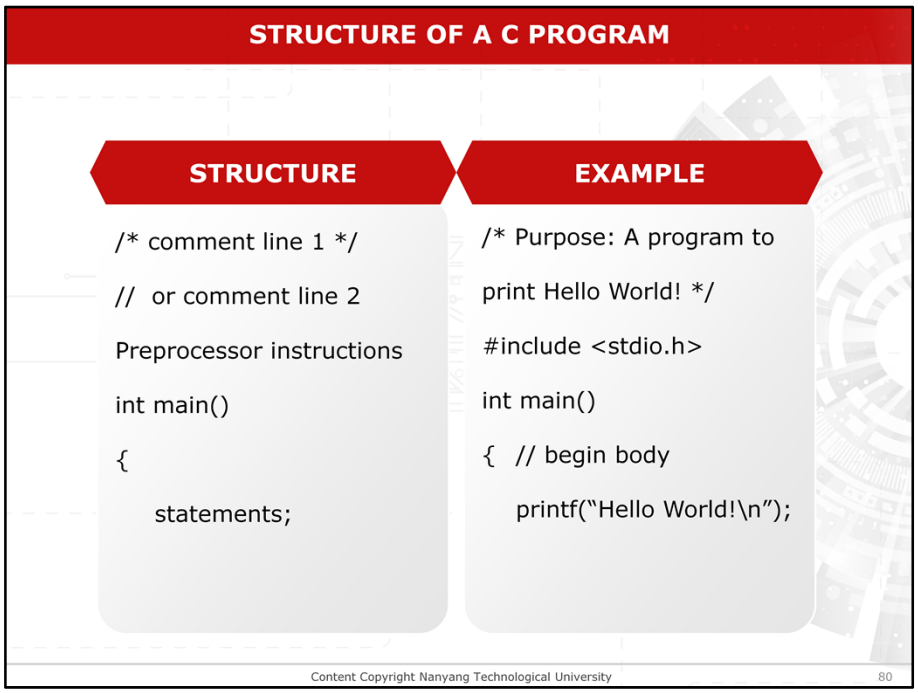
The *stdio.h* file is part of the standard library. It supports input and output operations that the program requires.



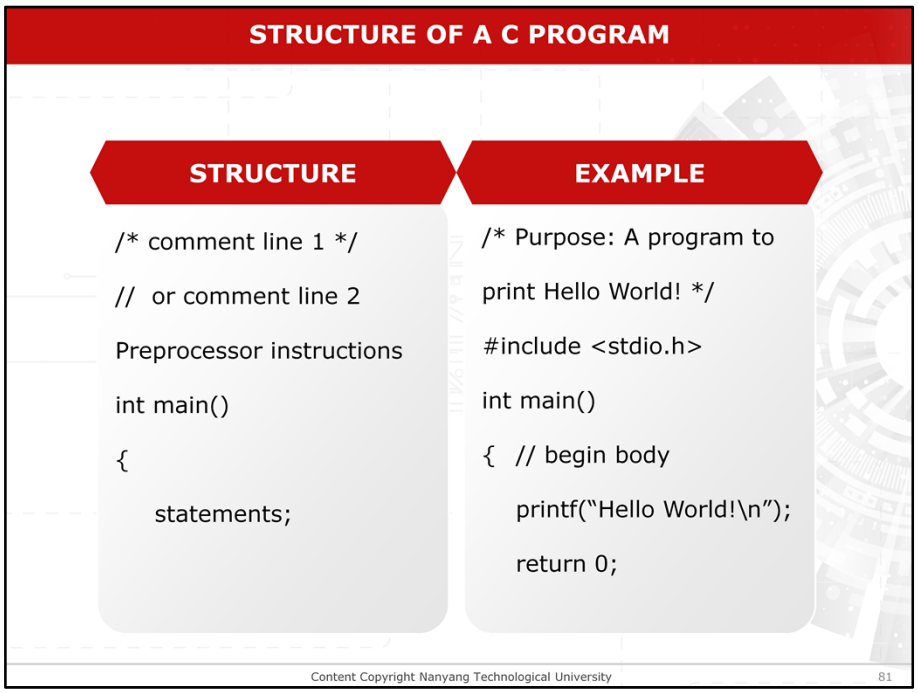
In the **main()** function, it requires to return an integer value, so the keyword **int** is used to inform the C compiler.



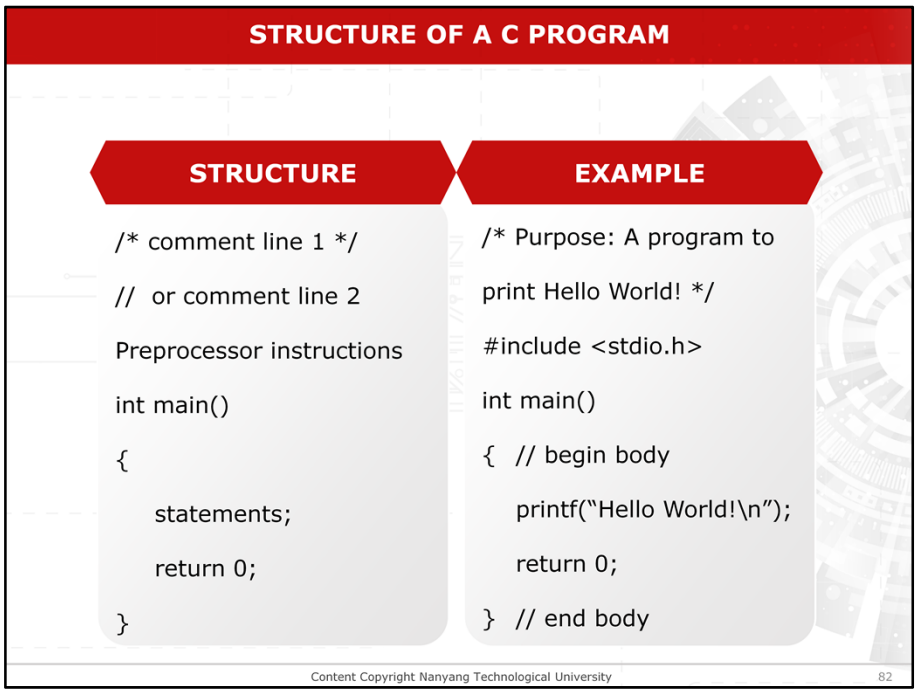
The braces **{}** are used to enclose the **main()** function body.



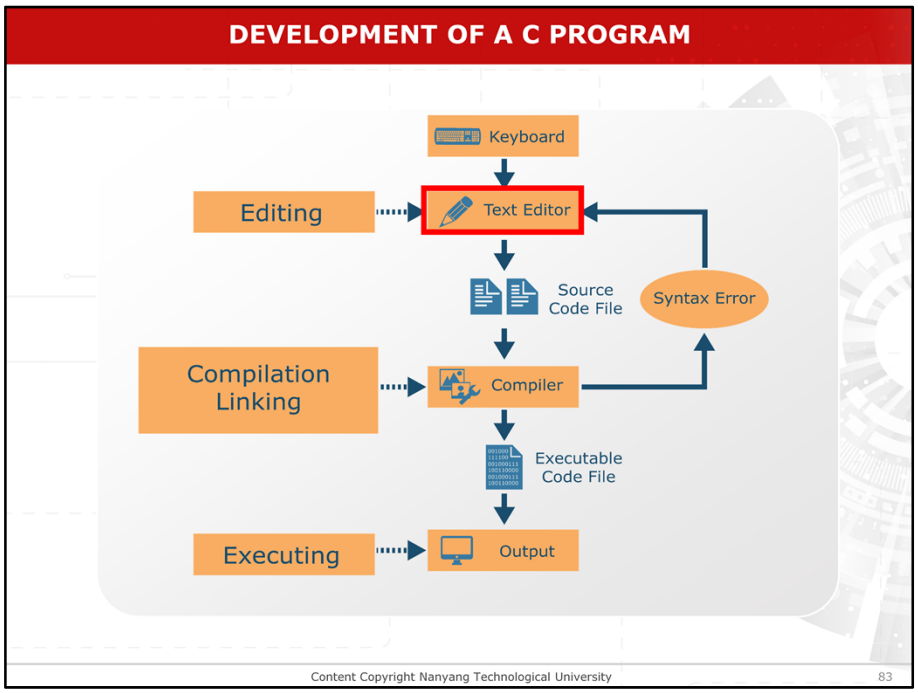
The **printf()** function is the library output function call to print a character string on the screen.



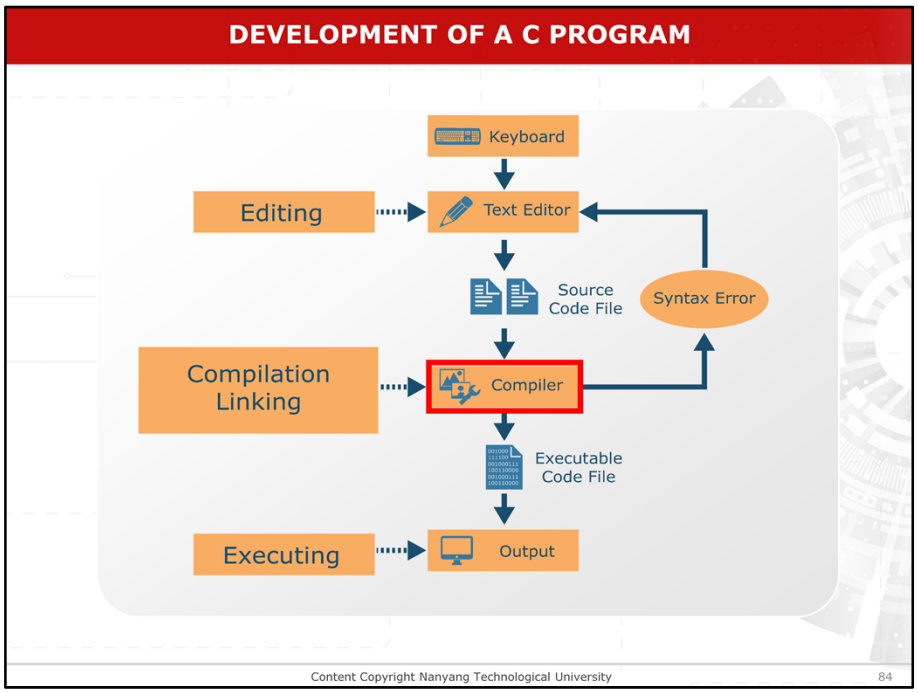
Finally, the statement **return 0** returns the control back to the system.



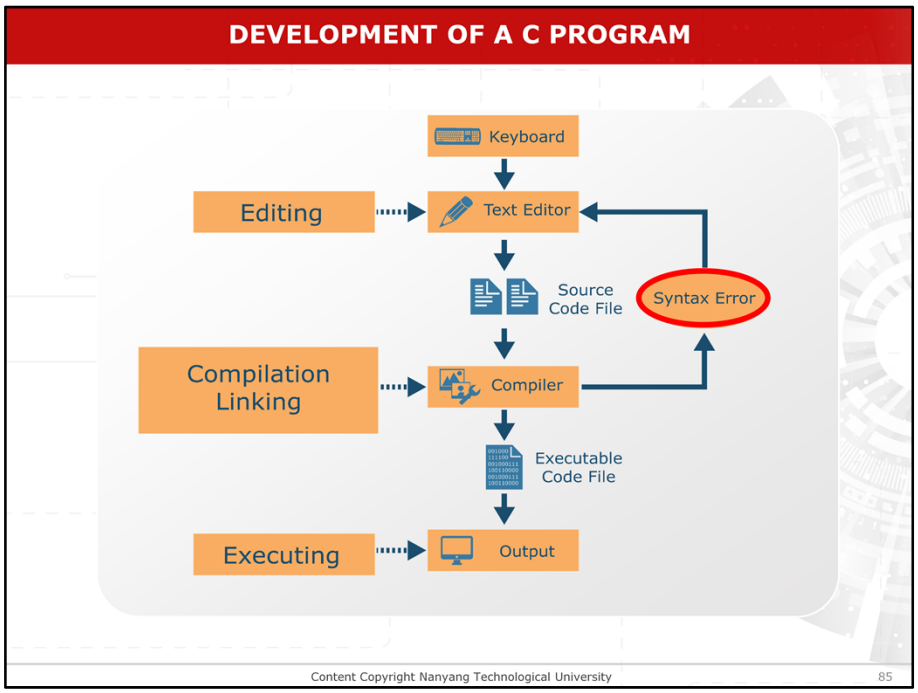
Finally, the last line (i.e. **// end body**) is a comment.



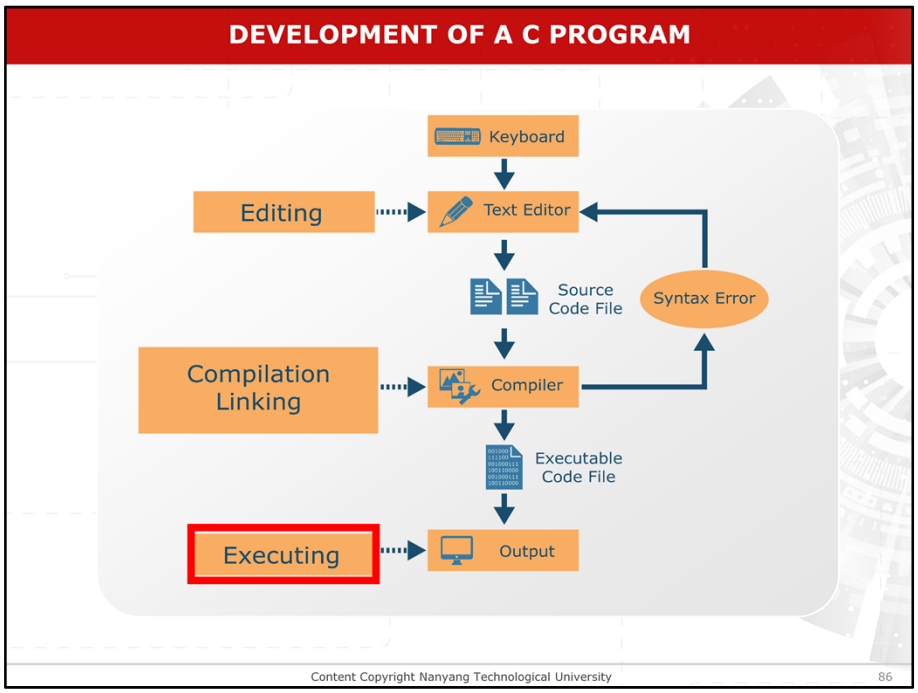
To develop a C program, a text editor is used to create a file containing the source code in C. Most compilers come with editors that can be used to enter and edit source code.



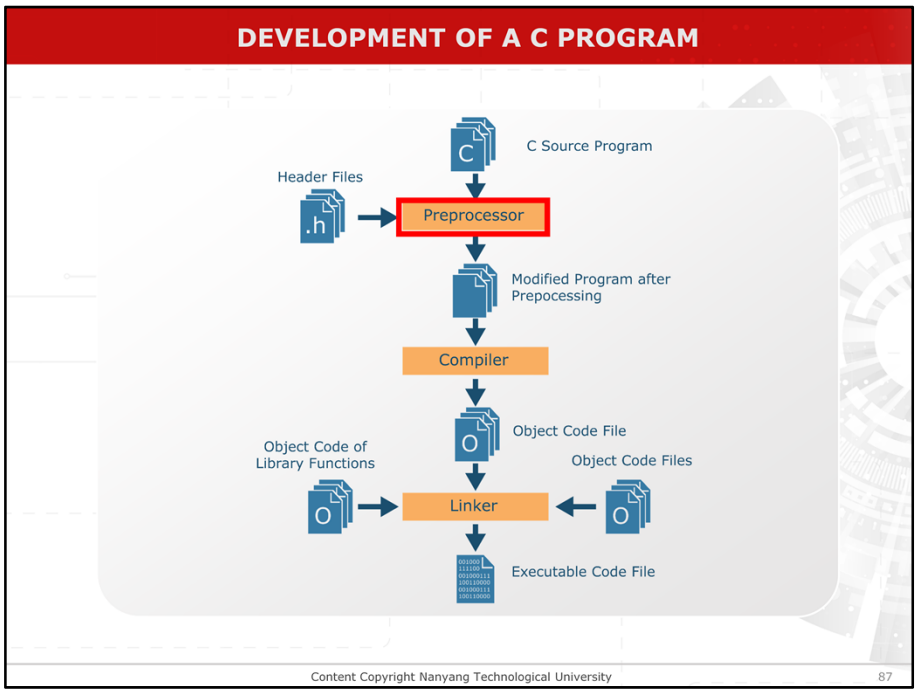
Then, the source code needs to be processed by a *compiler* to generate an object file.



If syntax errors are occurred during compilation, we will need to rectify the errors, and compile the source code again until no further errors are occurred. The *linker* is then used to link all the object files to create an executable file.

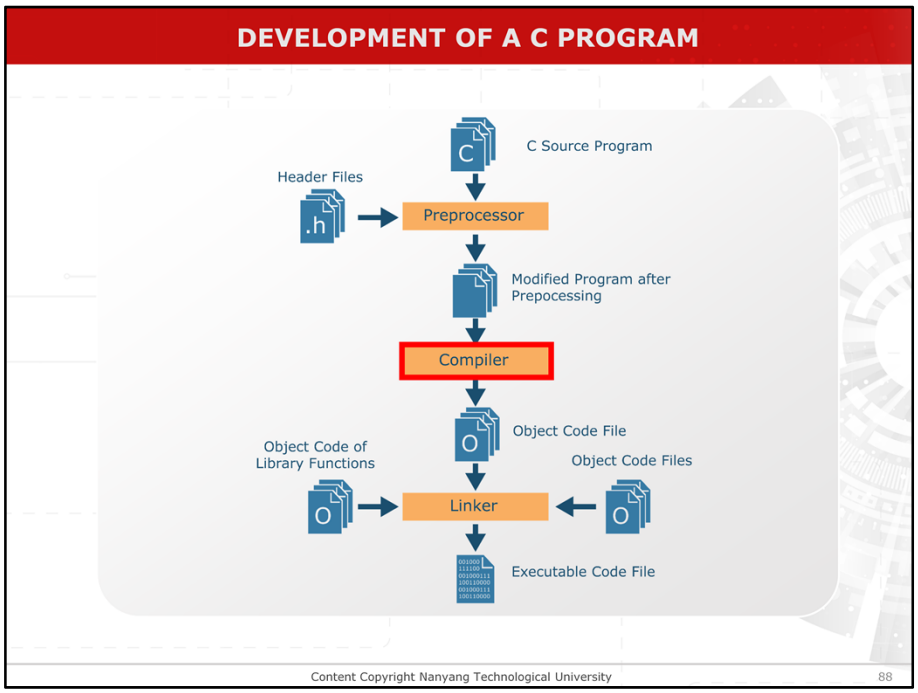


Finally, the executable file can be run and tested.

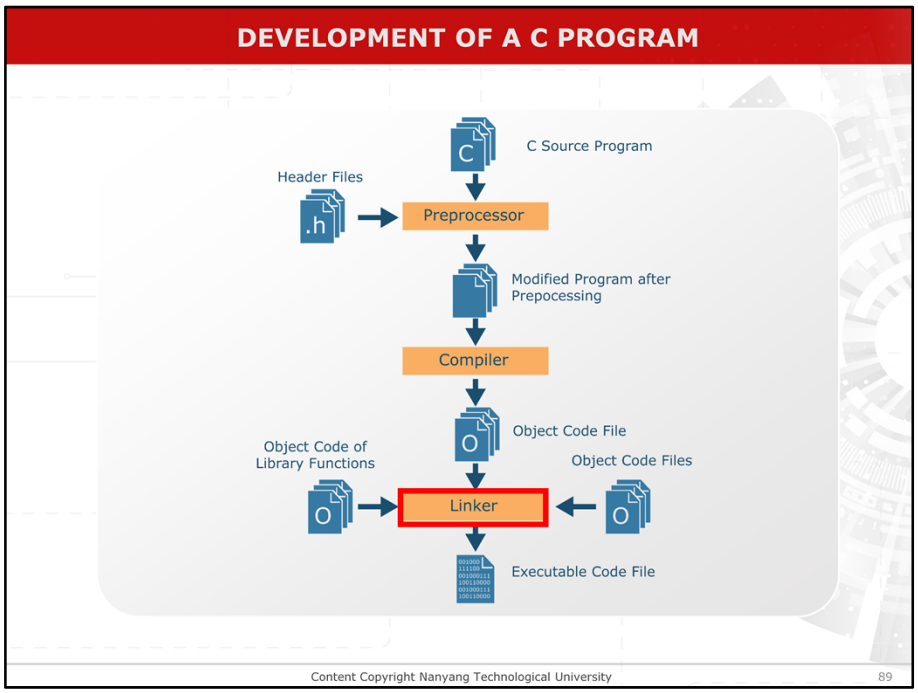


After writing the C program and typing it into the editor, the program needs to be processed by

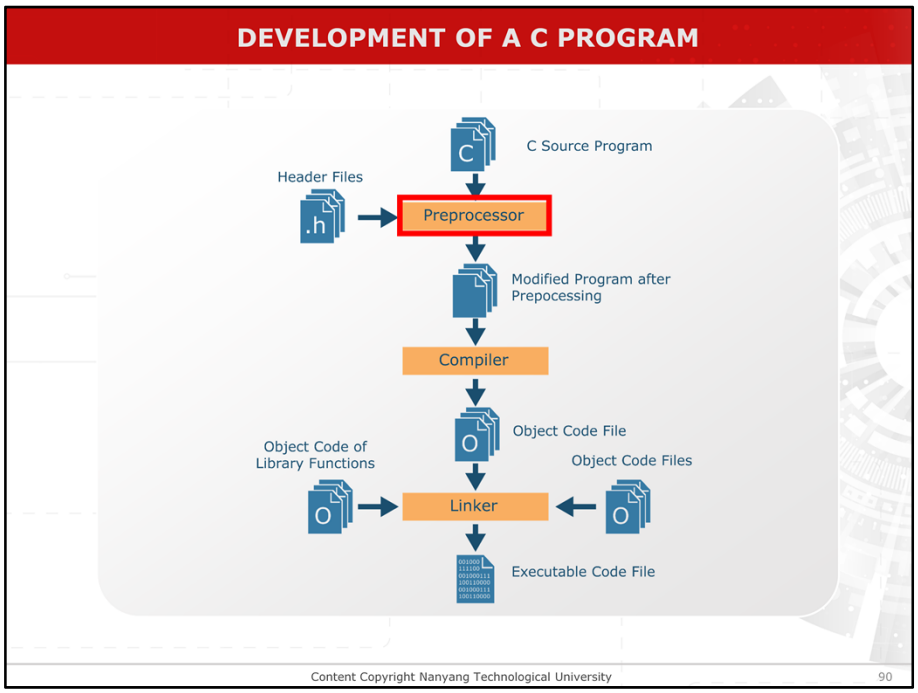
(1)preprocessor


DEVELOPMENT OF A C PROGRAM
C Source Program
Header Files
.h
Preprocessor
Modified Program after Prepocessing
Compiler
Object Code File
Object Code of Library Functions
Object Code Files
Linker
Executable Code File
Content Copyright Nanyang Technological University
88

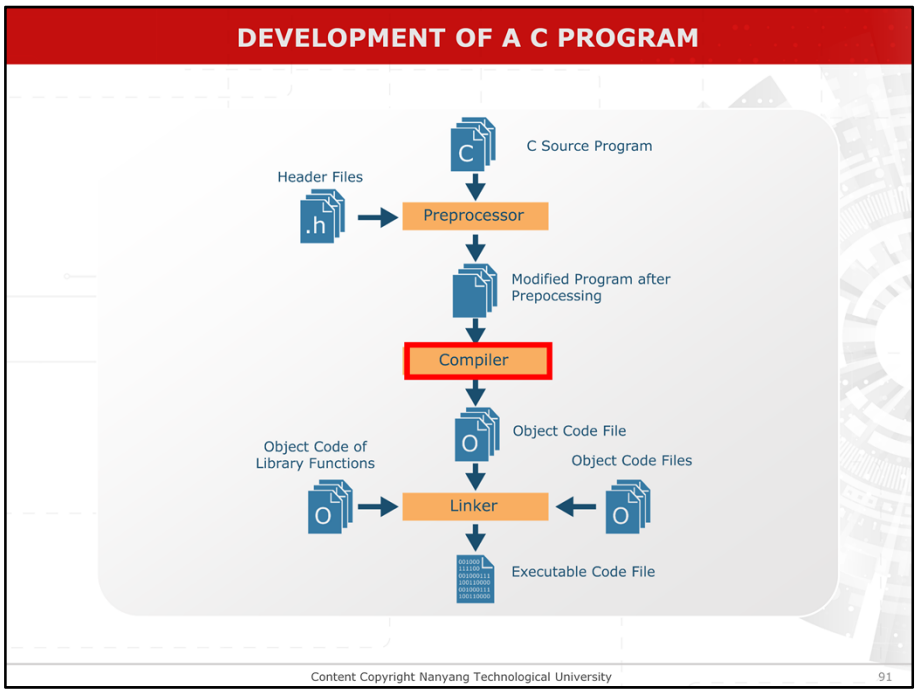
compiler



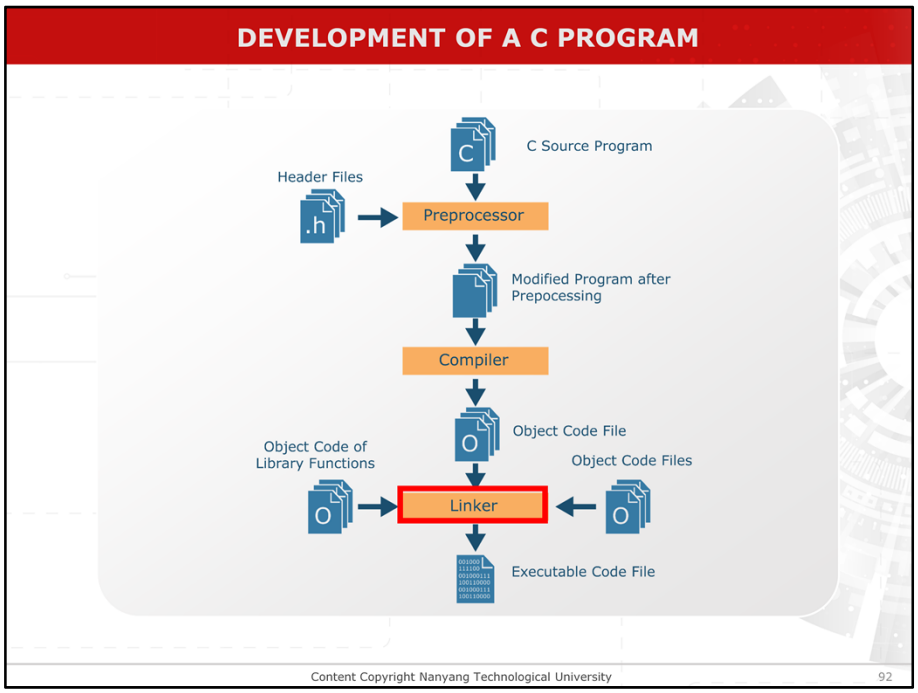
linker before you can execute the program.



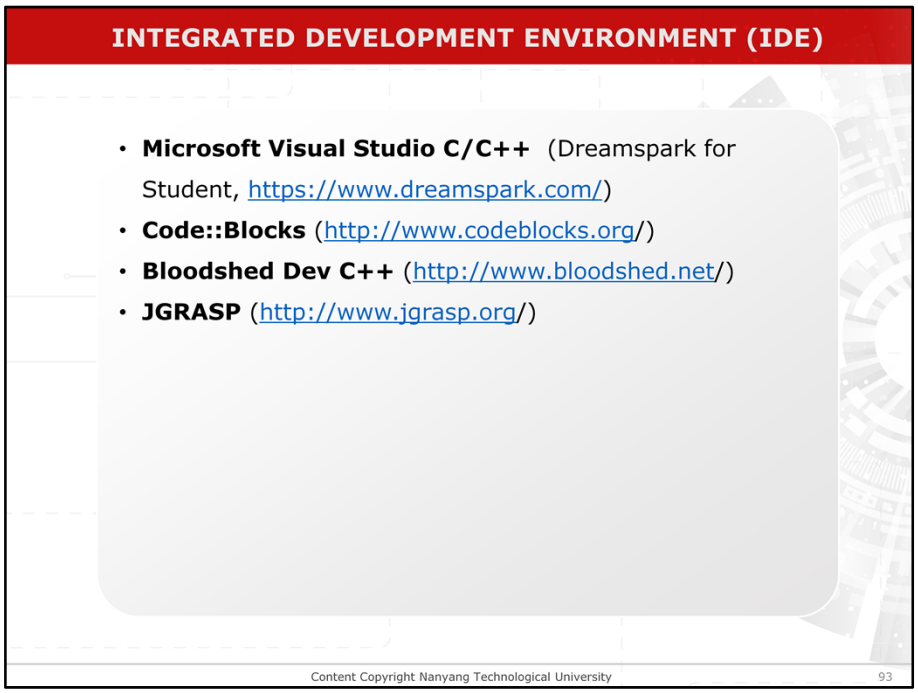
The preprocessor processes the preprocessing statements in the source code file. Preprocessing statements can support string replacement, macro expansion, file inclusion and conditional compilation. An example of preprocessing statement is **#include** *<filename>,* which informs the preprocessor to include the file *<filename>* into the text of the source code file before compilation.



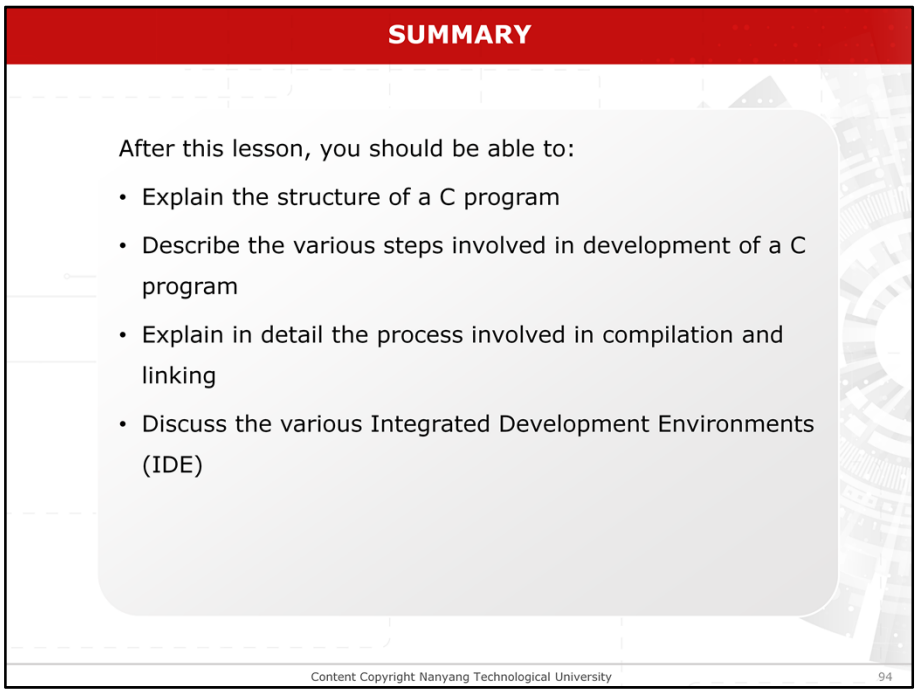
The compiler checks for syntax errors in the source files. Error messages will be given if errors are found. The locations of the errors are also given in the error messages. All the errors need to be corrected. If no more errors are detected, the compiler then translates the source code into machine language instructions, which are called object code. A file is created to store the object code. The object file has the same name as the source file, but with a different extension (**.obj** or **.o**). The extension is used to indicate that the file contains object code.



The linker combines the object code with other object files to produce an executable program. The linker allows us to develop programs that can be placed in several files. Functions that are frequently used in other programs are placed in separate files and compiled separately. This can save time to re-compile the functions every time they are used. In addition only files containing updated source code need to be re-compiled before linking. The linker also provides a convenient way to support the use of standard libraries. Most C programs use standard library functions to perform certain specific tasks such as input/ output operations and mathematical functions. During linking, the linker combines the object code with the object code from the standard library provided by the compiler to generate the executable program.



Programs can be developed and executed on different environments using an Integrated Development Environment or a console window. The most popular Integrated Development Environments for developing C programs are Microsoft Visual Studio C++, Code Blocks, Bloodshed Dev-C++ and JGrasp.



In summary, after viewing this video lesson, your should able to do the points listed.