

This lesson is on Basic C Programming

OVERVIEW

The following are the coverage for Basic C Programming:

- Why do we Learn C Programming Language?
- **Development of a C Program**
- Data Type, Constants, Variables and Operators
- Simple Input and Output

Content Copyright Nanyang Technological University

2

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]

LEARNING OBJECTIVES

After this lesson, you should be able to:

Content Copyright Nanyang Technological University 4

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

LEARNING OBJECTIVES

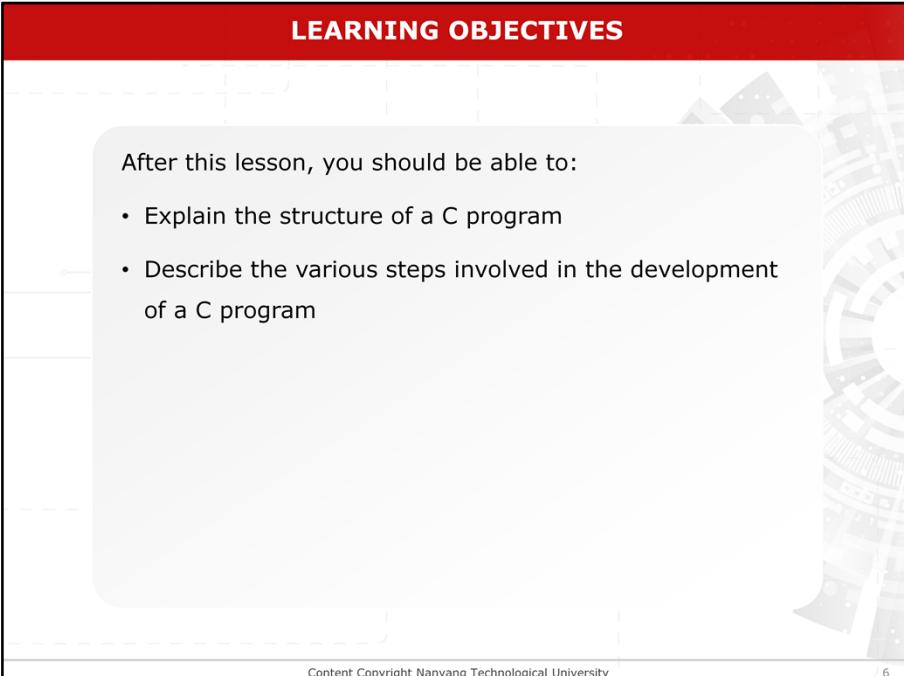
After this lesson, you should be able to:

- Explain the structure of a C program

Content Copyright Nanyang Technological University 5

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

LEARNING OBJECTIVES



After this lesson, you should be able to:

- Explain the structure of a C program
- Describe the various steps involved in the development of a C program

Content Copyright Nanyang Technological University 6

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

LEARNING OBJECTIVES

After this lesson, you should be able to:

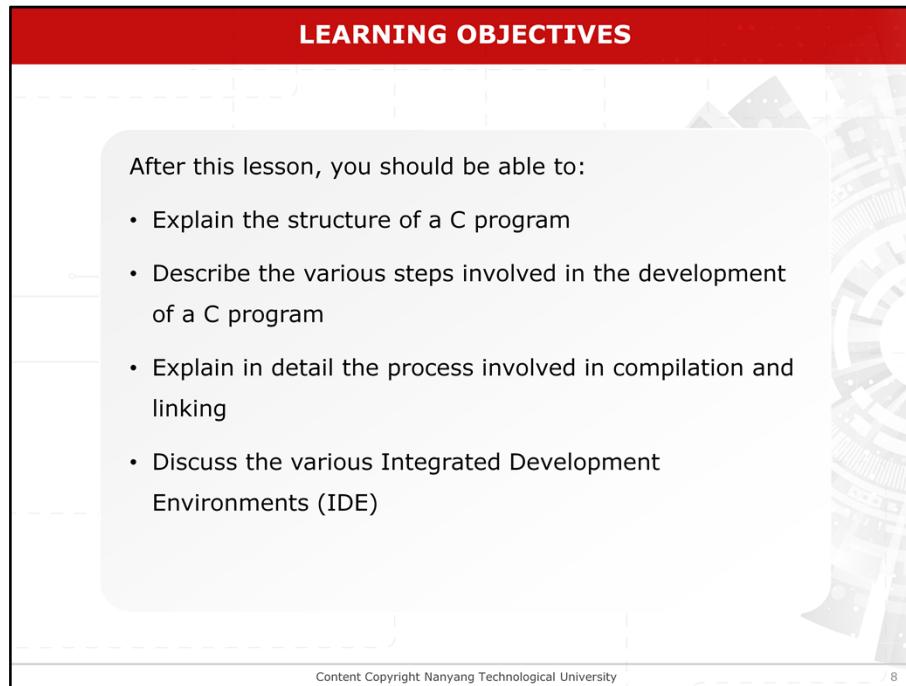
- Explain the structure of a C program
- Describe the various steps involved in the development of a C program
- Explain in detail the process involved in compilation and linking

Content Copyright Nanyang Technological University

7

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

LEARNING OBJECTIVES



After this lesson, you should be able to:

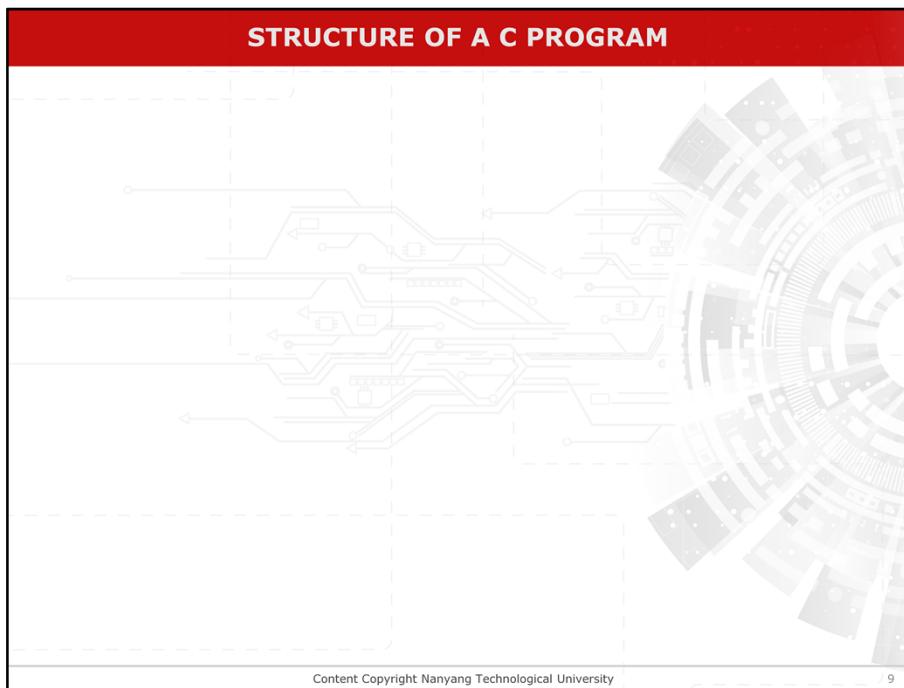
- Explain the structure of a C program
- Describe the various steps involved in the development of a C program
- Explain in detail the process involved in compilation and linking
- Discuss the various Integrated Development Environments (IDE)

Content Copyright Nanyang Technological University

8

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:

Content Copyright Nanyang Technological University

10

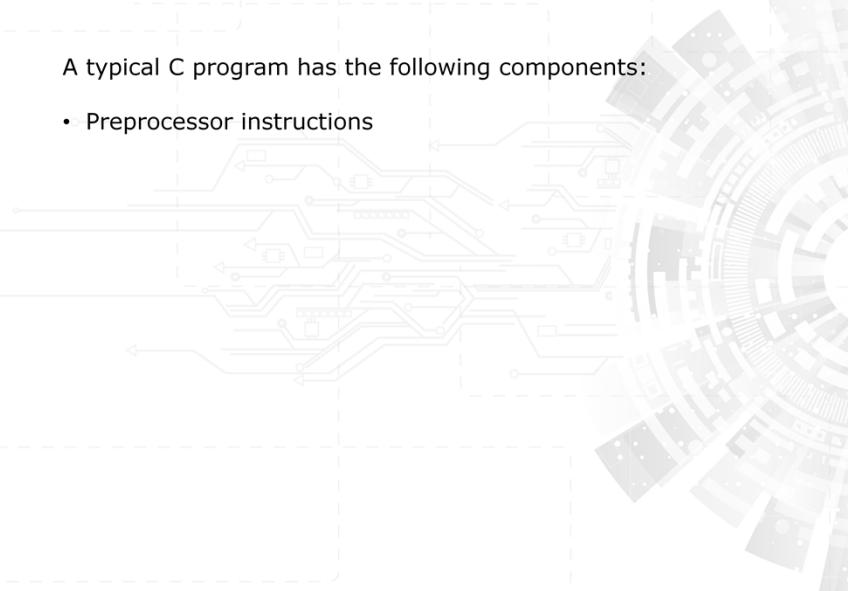
Structure of a C Program

A typical C program has the following components

STRUCTURE OF A C PROGRAM

A typical C program has the following components:

- Preprocessor instructions



Content Copyright Nanyang Technological University 11

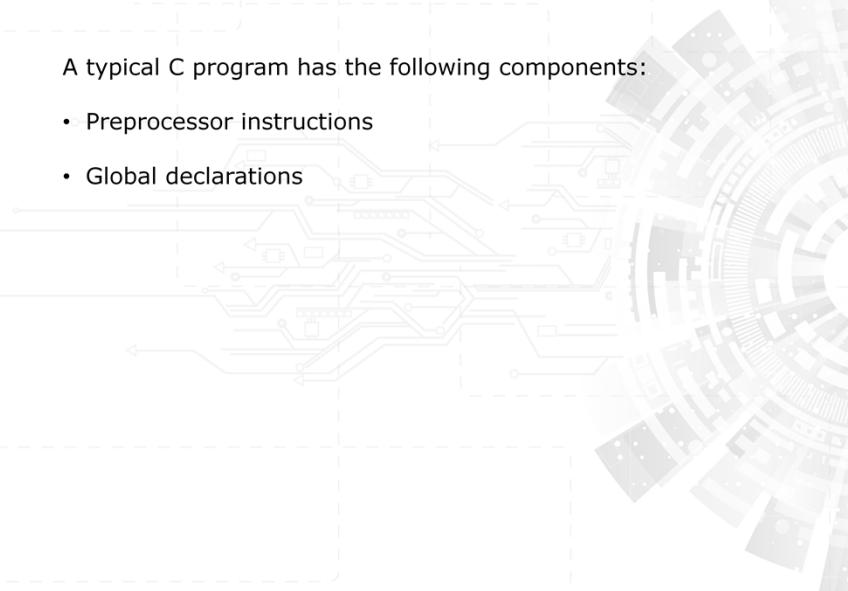
preprocessor instructions

STRUCTURE OF A C PROGRAM

A typical C program has the following components:

- Preprocessor instructions
- Global declarations

Content Copyright Nanyang Technological University 12



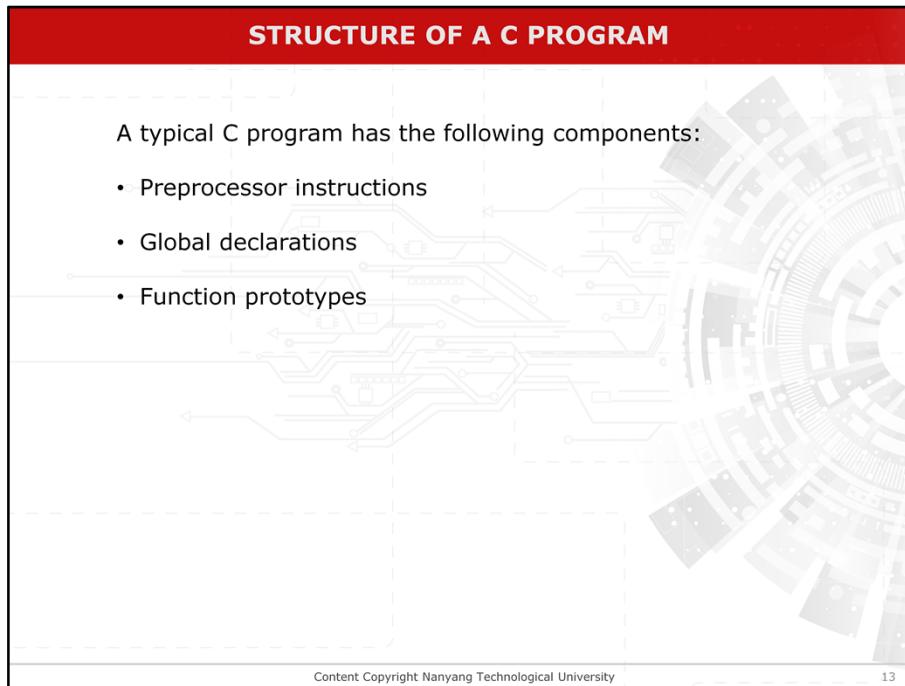
global declarations

STRUCTURE OF A C PROGRAM

A typical C program has the following components:

- Preprocessor instructions
- Global declarations
- Function prototypes

Content Copyright Nanyang Technological University 13



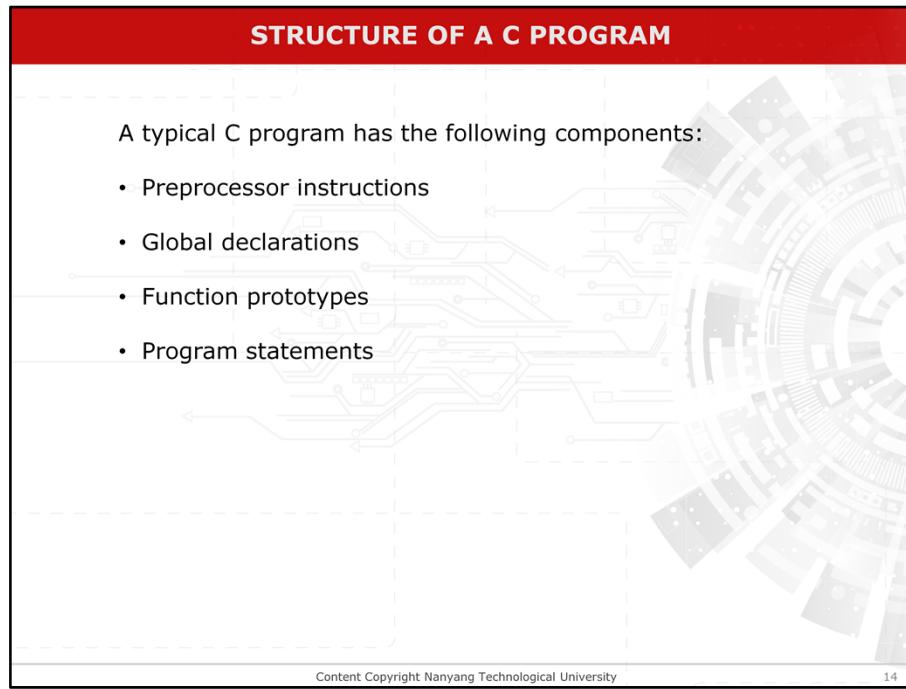
function prototypes

STRUCTURE OF A C PROGRAM

A typical C program has the following components:

- Preprocessor instructions
- Global declarations
- Function prototypes
- Program statements

Content Copyright Nanyang Technological University 14



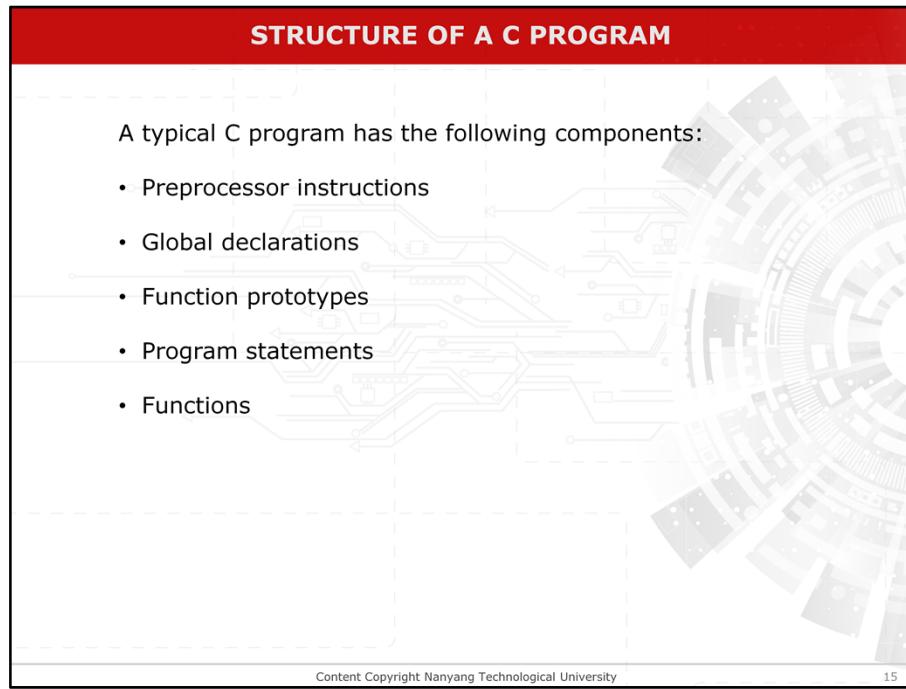
program statements

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

A faint background image of a complex circuit board or microchip design, showing various electronic components and interconnects.

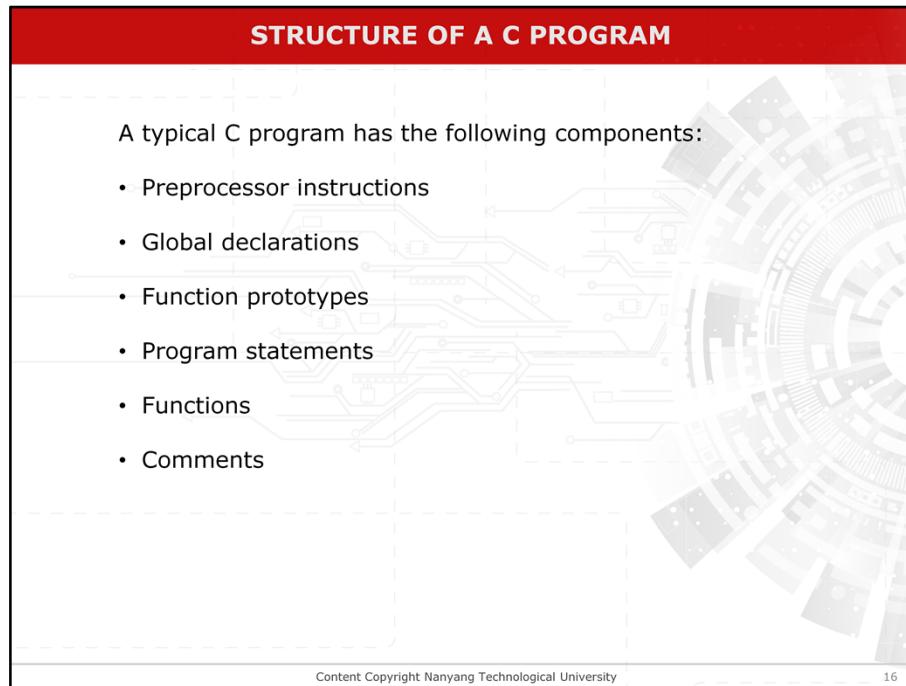
functions

STRUCTURE OF A C PROGRAM

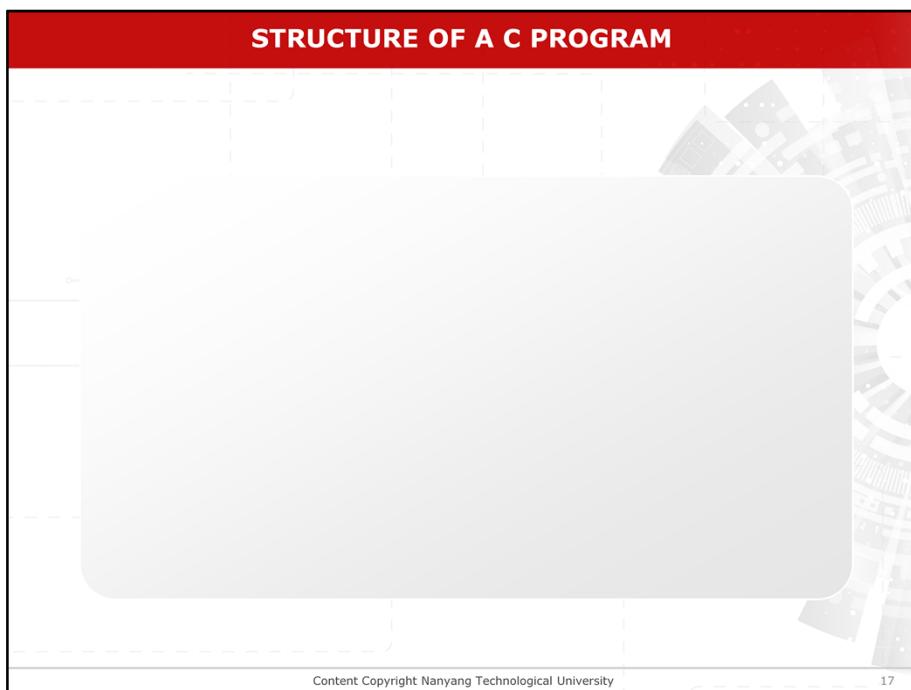
A typical C program has the following components:

- Preprocessor instructions
- Global declarations
- Function prototypes
- Program statements
- Functions
- Comments

Content Copyright Nanyang Technological University 16



and comments.



Structure of a C program
[Pause: 1sec]

STRUCTURE OF A C PROGRAM

Preprocessor Instructions

Content Copyright Nanyang Technological University

18

Preprocessor Instructions
[Pause: 1 sec]

STRUCTURE OF A C PROGRAM

Preprocessor Instructions

- Instructions to the preprocessor of the compiler

Content Copyright Nanyang Technological University

19

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

STRUCTURE OF A C PROGRAM

Preprocessor Instructions

- Instructions to the preprocessor of the compiler
- Start with the symbol #

Content Copyright Nanyang Technological University

20

. All preprocessor instructions start with the symbol #.

STRUCTURE OF A C PROGRAM

Preprocessor Instructions

- Instructions to the preprocessor of the compiler
- Start with the symbol #
- Supports string replacement, macro expansion, file inclusion and conditional compilation

Content Copyright Nanyang Technological University

21

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

STRUCTURE OF A C PROGRAM

Example: Preprocessor Instructions

Content Copyright Nanyang Technological University

22

For example
[Pause: 1sec]

STRUCTURE OF A C PROGRAM

Example: Preprocessor Instructions

```
#include <filename>
```

Content Copyright Nanyang Technological University

23

the **#include <filename>**

STRUCTURE OF A C PROGRAM

Example: Preprocessor Instructions

`#include <filename>`

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

Content Copyright Nanyang Technological University

24

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

STRUCTURE OF A C PROGRAM

Example: Preprocessor Instructions

`#include <filename>`

- Instruction 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 `.h` such as `stdio.h`.

Content Copyright Nanyang Technological University

25

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

STRUCTURE OF A C PROGRAM

Example: Preprocessor Instructions

`#include <filename>`

- Instruction 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 `.h` such as `stdio.h`.
- They are placed at the beginning of the source code file.

Content Copyright Nanyang Technological University

26

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

STRUCTURE OF A C PROGRAM

Example: Preprocessor Instructions

`#include <filename>`

- Instruction 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 `.h` such as `stdio.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.

Content Copyright Nanyang Technological University

27

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

STRUCTURE OF A C PROGRAM

Global Declaration

Content Copyright Nanyang Technological University

28

Global declaration

[Pause: 1 sec]

STRUCTURE OF A C PROGRAM

Global Declaration

- Declares global variables that are accessible anywhere within a C program.

Content Copyright Nanyang Technological University

29

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

STRUCTURE OF A C PROGRAM

Function Prototypes

- Function prototypes provide useful information regarding the functions used in the program to the C compiler.

Content Copyright Nanyang Technological University

30

Function prototypes provide useful information regarding the functions used in the program to the C compiler.

[Pause: 1sec]

STRUCTURE OF A C PROGRAM

Function Prototypes

- Function prototypes provide useful information regarding the functions used in the program to the C compiler.
- They inform the compiler about the name and arguments of the functions contained in the program.

Content Copyright Nanyang Technological University

31

They inform the compiler about the name and arguments of the functions contained in the program.

[Pause: 1sec]

STRUCTURE OF A C PROGRAM

Function Prototypes

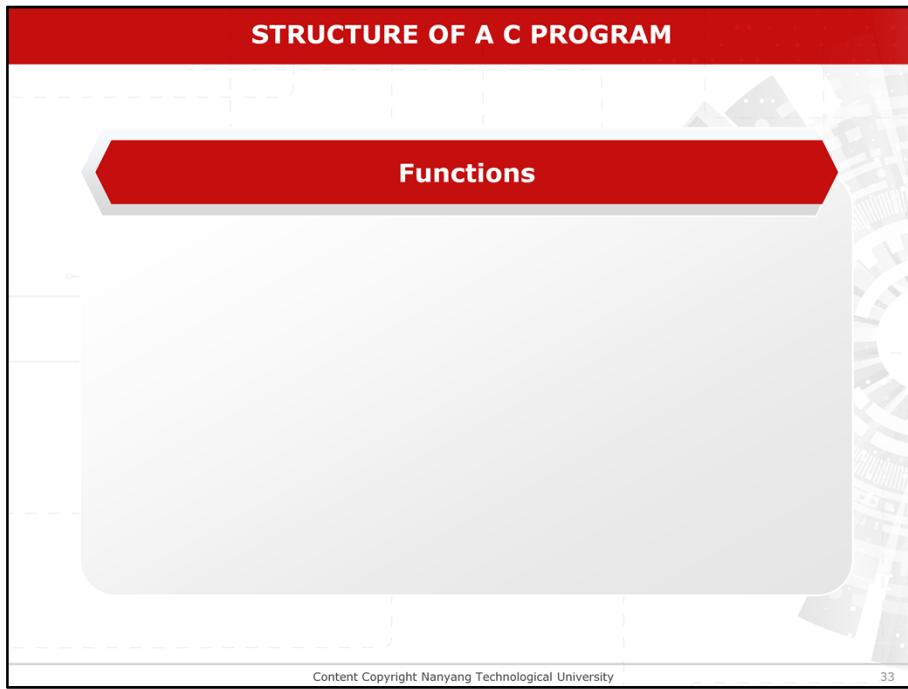
- Function prototypes provide useful information regarding the functions used in the program to the C compiler.
- They inform the compiler about the name and arguments of the functions contained in the program.
- They need to appear before the functions are used.

Content Copyright Nanyang Technological University

32

They need to appear before the functions are used.

[Pause: 1sec]



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

STRUCTURE OF A C PROGRAM

Functions

- A C program consists of one or more functions.

Content Copyright Nanyang Technological University

34

A C program consists of one or more functions.

[Pause: 1sec]

STRUCTURE OF A C PROGRAM

Functions

- A C program consists of one or more functions.
- The **main()** function is required in every C program.

Content Copyright Nanyang Technological University

35

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

[Pause: 1sec]

STRUCTURE OF A C PROGRAM

Functions

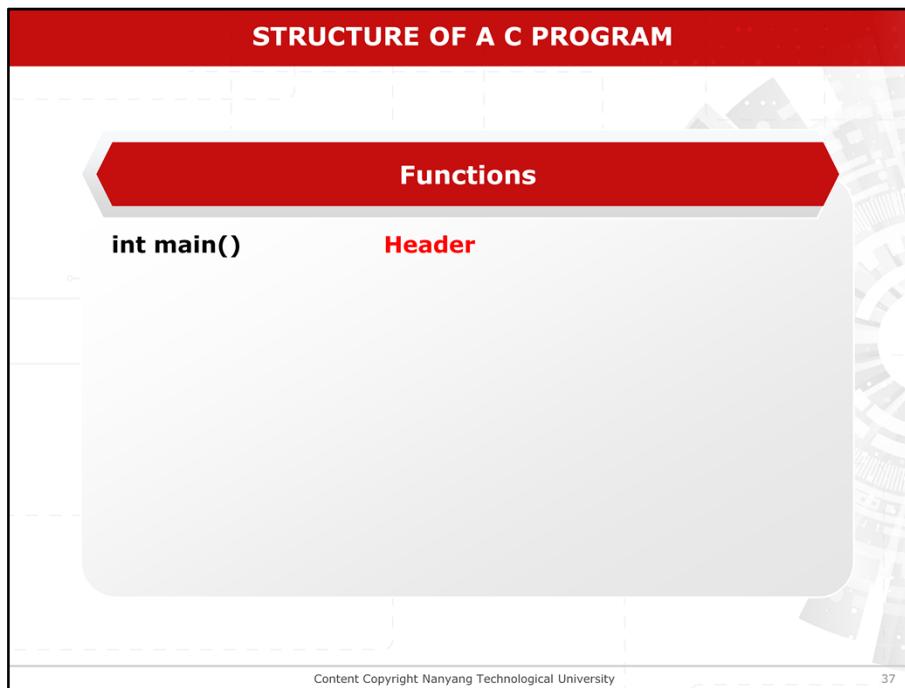
- A C program consists of one or more functions.
- The **main()** function is required in every C program.
- A function has a header and a body.

Content Copyright Nanyang Technological University

36

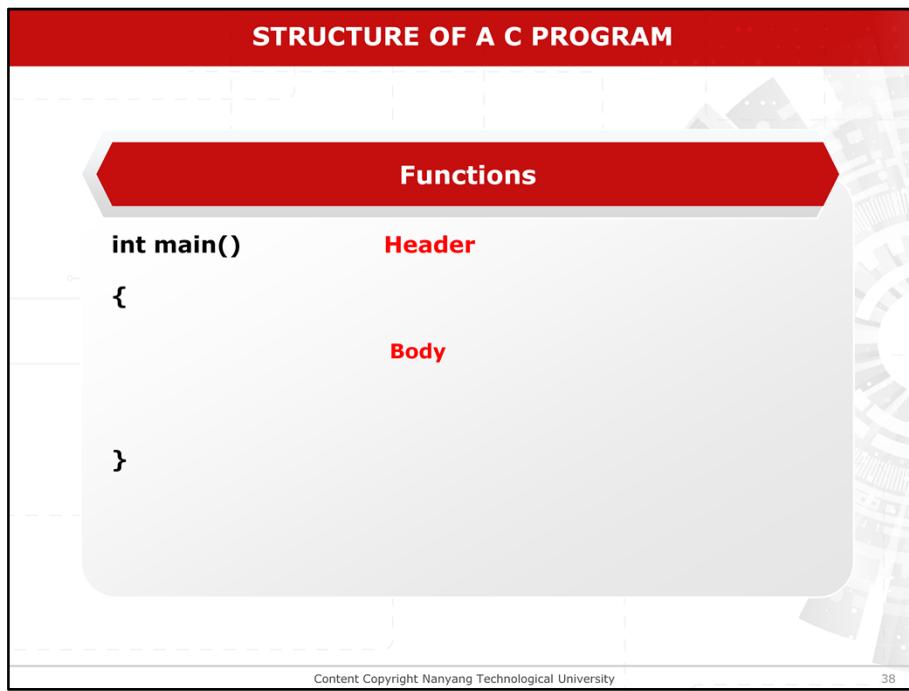
A function has a header and a body.

[Pause: 1sec]



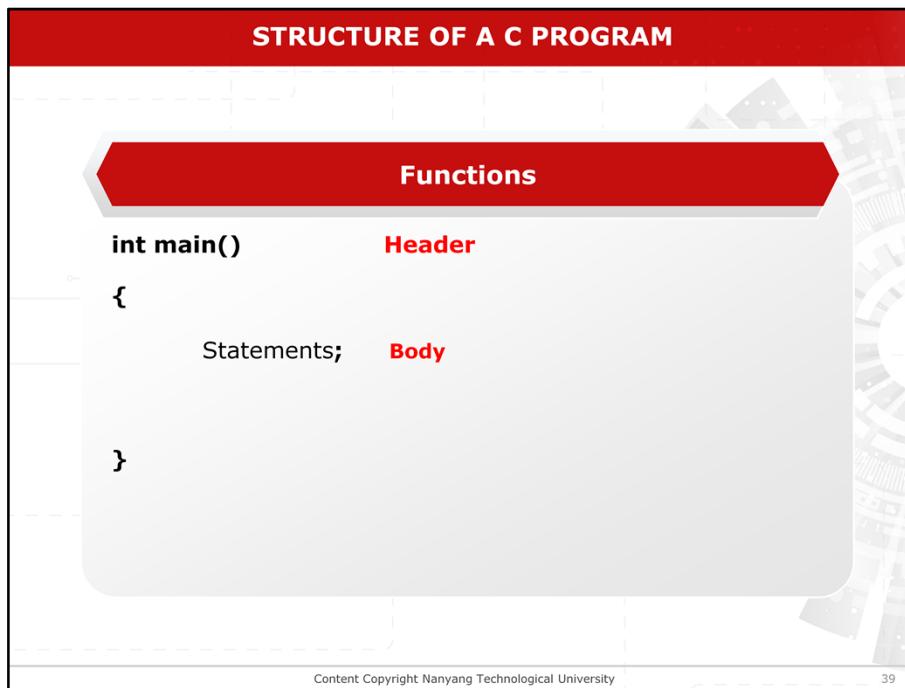
The header is int main ().The function header defines the name of the function, and the inputs expected by the function.

Pause: 1 sec

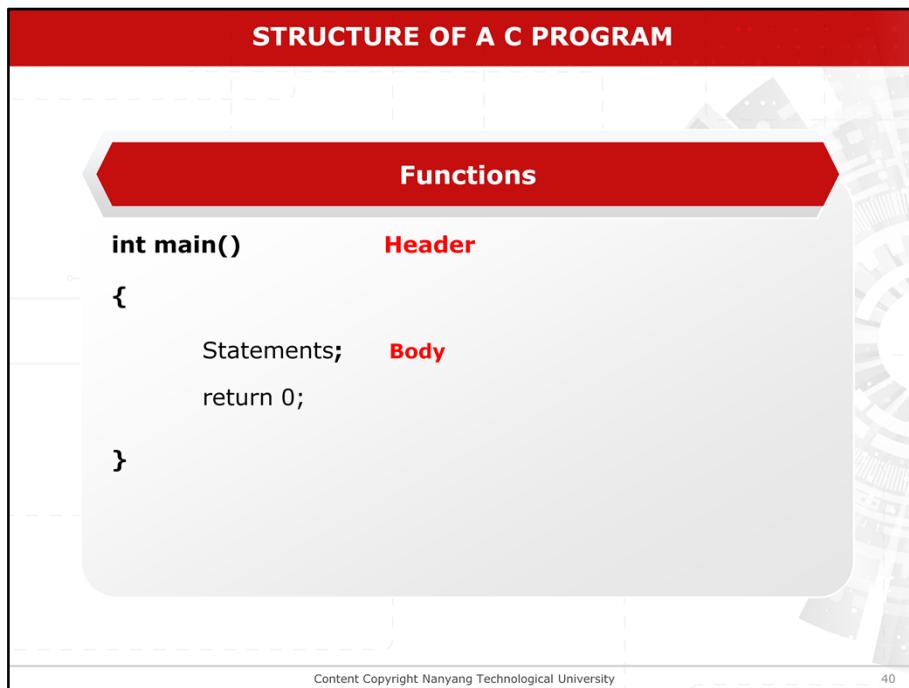


The body of the int main() function is enclosed by the braces { }.

Pause: 1 sec

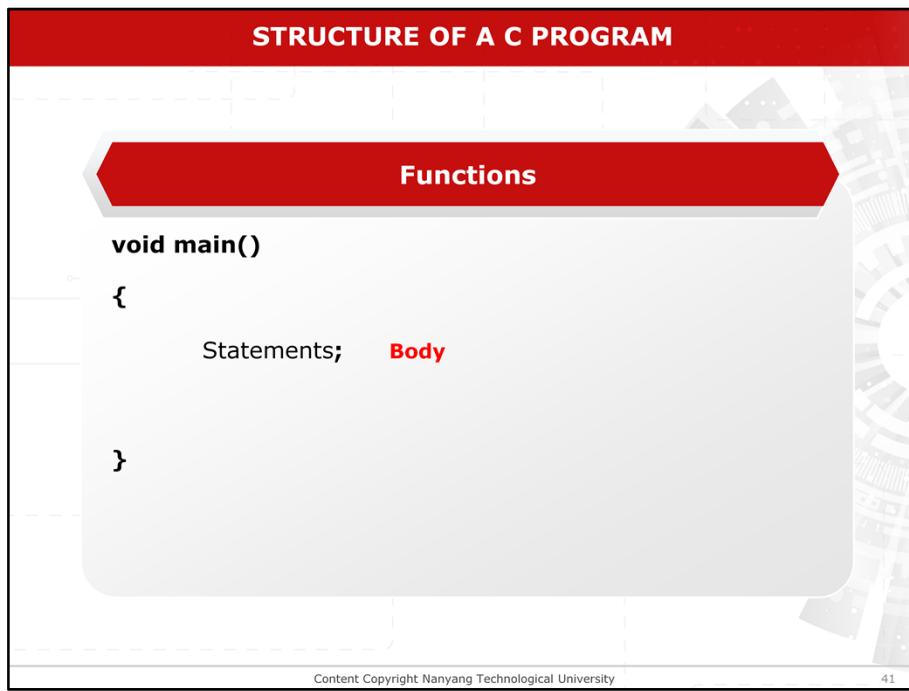


The function body contains statements that are written to perform a specific task.
The body will be executed when the function is called.
Program execution starts from the beginning of the body.
Pause: 1 sec



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.

STRUCTURE OF A C PROGRAM

Functions

```
int add(int x, int y)
{
    ...
}
```

Content Copyright Nanyang Technological University

42

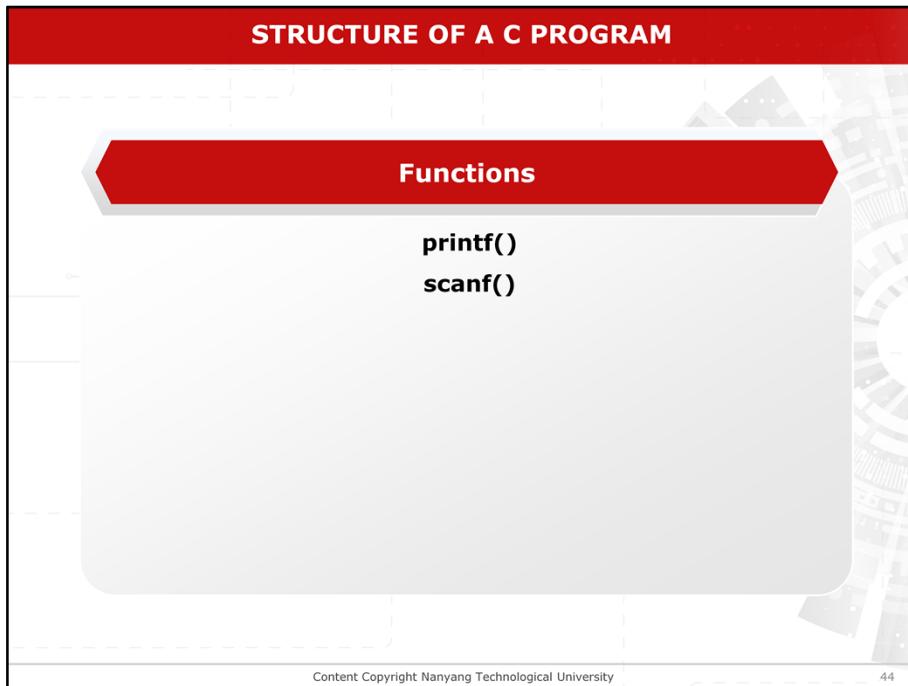
A function may or may not have arguments, and it may or may not have a return value. This function has two arguments and requires a return value.

The diagram illustrates the structure of a C program. At the top, a red bar contains the text "STRUCTURE OF A C PROGRAM". Below this, a large white area represents a code editor window. In the center of this window, a red arrow-shaped callout points to the word "Functions". Inside the callout, the following C code is shown:

```
void add()
{
    ...
}
```

At the bottom left of the slide, the text "Content Copyright Nanyang Technological University" is visible. At the bottom right, the number "43" indicates the slide number.

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.

STRUCTURE OF A C PROGRAM

Functions

printf()

- It displays information on the screen.

scanf()

Content Copyright Nanyang Technological University

45

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

STRUCTURE OF A C PROGRAM

Functions

printf()

- It displays information on the screen.

scanf()

- It reads data from the keyboard and assigns the data to a variable.

Content Copyright Nanyang Technological University

46

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

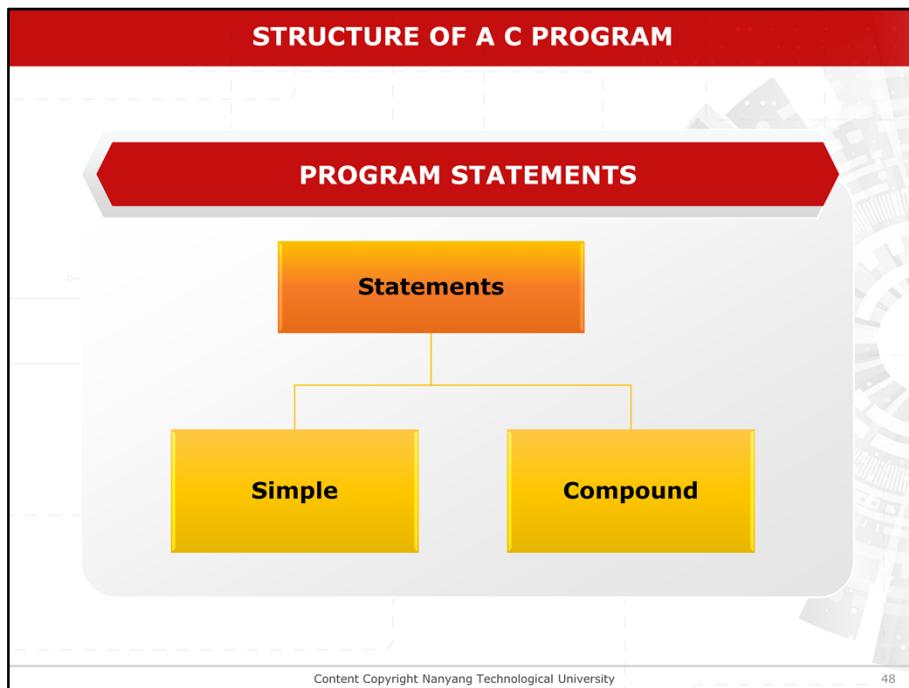
STRUCTURE OF A C PROGRAM

PROGRAM STATEMENTS

Content Copyright Nanyang Technological University

47

A statement is a command to the computer.



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

STRUCTURE OF A C PROGRAM

SIMPLE STATEMENTS

Content Copyright Nanyang Technological University

49

A simple statement is a statement terminated by a semicolon.

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

Content Copyright Nanyang Technological University

50

There are four types of simple statements:

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Declaration statements

Content Copyright Nanyang Technological University

51

Declaration statement

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Declaration statements

```
int x,y,z; /*declaration statement*/
```

Content Copyright Nanyang Technological University

52

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

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Assignment statements

Content Copyright Nanyang Technological University

53

Assignment statements

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Assignment statements

```
x =30; /* assignment statement */
```

Content Copyright Nanyang Technological University

54

A value of 30 is assigned to x.

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Function call statements

Content Copyright Nanyang Technological University

55

Function call statements

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Function call statements

```
printf("Welcome to Programming with C");
/* function call statement */
```

Content Copyright Nanyang Technological University

56

This is an example of a function call statement.

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Control statements

Content Copyright Nanyang Technological University

57

Control statements

STRUCTURE OF A C PROGRAM

TYPES OF SIMPLE STATEMENTS

- Control statements

```
for (y=0; y<z; y++){... }  
/* control statement */
```

Content Copyright Nanyang Technological University

58

This is an example of control statement.

STRUCTURE OF A C PROGRAM

COMPOUND STATEMENTS

Content Copyright Nanyang Technological University

59

Compound statement

STRUCTURE OF A C PROGRAM

COMPOUND STATEMENTS

```
{  
    statement_1;
```

Content Copyright Nanyang Technological University

60

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

STRUCTURE OF A C PROGRAM

COMPOUND STATEMENTS

```
{  
    statement_1;  
    statement_2;  
    ...
```

Content Copyright Nanyang Technological University

61

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

STRUCTURE OF A C PROGRAM

COMPOUND STATEMENTS

```
{  
    statement_1;  
    statement_2;  
    ...  
    statement_n;
```

Content Copyright Nanyang Technological University

62

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

STRUCTURE OF A C PROGRAM

COMPOUND STATEMENTS

```
{  
    statement_1;  
    statement_2;  
    ...  
    statement_n;  
}
```

Content Copyright Nanyang Technological University

63

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

STRUCTURE OF A C PROGRAM

COMPOUND STATEMENTS

```
{  
    statement_1;  
    statement_2;  
    ...  
    statement_n;  
}
```

Each of the
statement_i (where
 $i=1,...n$) can be a simple or
a compound statement.

Content Copyright Nanyang Technological University

64

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

STRUCTURE OF A C PROGRAM

COMMENTS

Content Copyright Nanyang Technological University

65

STRUCTURE OF A C PROGRAM

COMMENTS

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

Content Copyright Nanyang Technological University

66

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

STRUCTURE OF A C PROGRAM

COMMENTS

- 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.

Content Copyright Nanyang Technological University

67

A comment is a piece of English text.

STRUCTURE OF A C PROGRAM

COMMENTS

- 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.

Content Copyright Nanyang Technological University

68

Comments may appear anywhere in the program.

STRUCTURE OF A C PROGRAM

COMMENTS

- 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.

Content Copyright Nanyang Technological University

69

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

STRUCTURE OF A C PROGRAM

COMMENTS

```
/* This is the first comment  
This is the second comment  
This is the third comment */
```

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

Content Copyright Nanyang Technological University

70

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

STRUCTURE OF A C PROGRAM

MULTIPLE-LINE COMMENTS

```
/* This is the first comment  
This is the second comment  
This is the third comment */
```

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

Content Copyright Nanyang Technological University

71

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

STRUCTURE OF A C PROGRAM

SINGLE-LINE COMMENT

// This is another comment

// for a **single line of comment**

Content Copyright Nanyang Technological University

72

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

STRUCTURE OF A C PROGRAM

COMMENTS

- Comments document what the program does, how the variables are defined, and how the logic of the program works.

Content Copyright Nanyang Technological University

73

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.

STRUCTURE OF A C PROGRAM

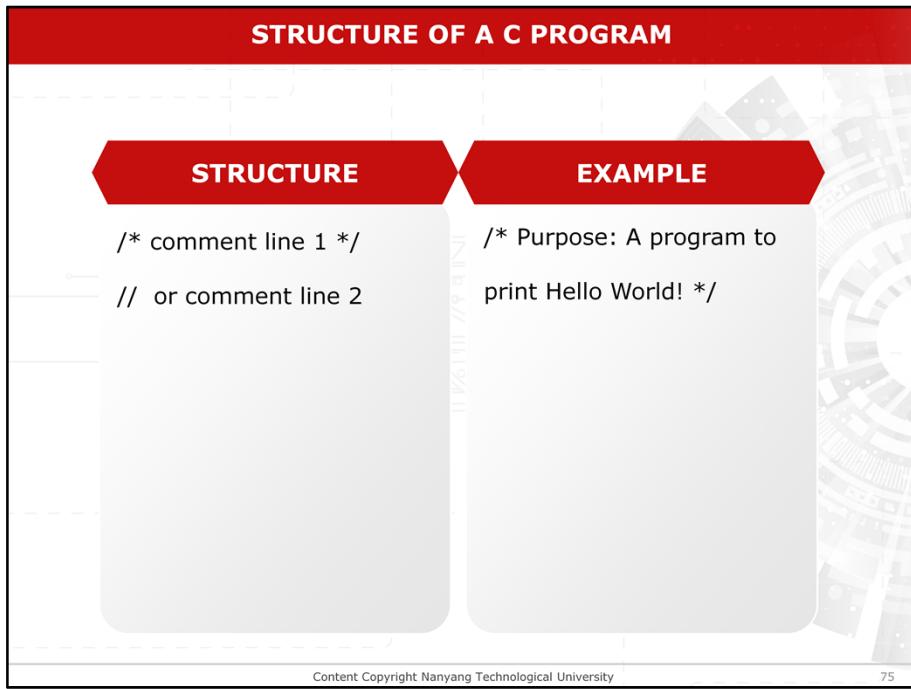
COMMENTS

- 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.

Content Copyright Nanyang Technological University

74

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.

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions</pre>	<pre>/* Purpose: A program to print Hello World! */ #include</pre>

Content Copyright Nanyang Technological University

76

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

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions</pre>	<pre>/* Purpose: A program to print Hello World! */ #include <stdio.h></pre>

Content Copyright Nanyang Technological University

77

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

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions int main()</pre>	<pre>/* Purpose: A program to print Hello World! */ #include <stdio.h> int main()</pre>

Content Copyright Nanyang Technological University

78

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

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions int main() {</pre>	<pre>/* Purpose: A program to print Hello World! */ #include <stdio.h> int main() { // begin body</pre>

Content Copyright Nanyang Technological University

79

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

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions int main() { statements;</pre>	<pre>/* Purpose: A program to print Hello World! */ #include <stdio.h> int main() { // begin body printf("Hello World!\n");</pre>

Content Copyright Nanyang Technological University

80

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

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions int main() { statements;</pre>	<pre>/* Purpose: A program to print Hello World! */ #include <stdio.h> int main() { // begin body printf("Hello World!\n"); return 0;</pre>

Content Copyright Nanyang Technological University

81

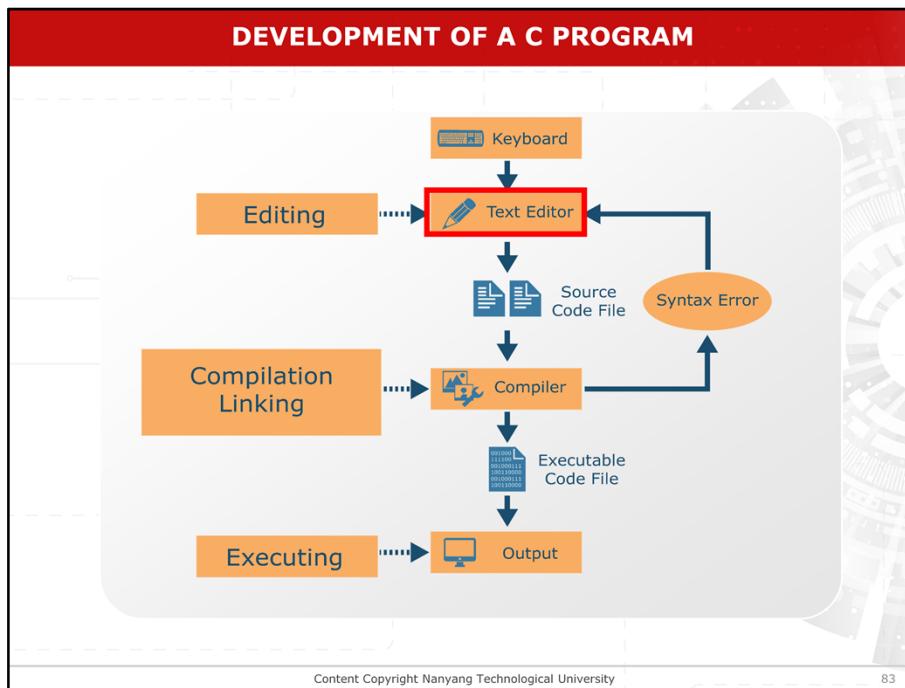
Finally, the statement **return 0** returns the control back to the system.

STRUCTURE OF A C PROGRAM	
STRUCTURE	EXAMPLE
<pre>/* comment line 1 */ // or comment line 2 Preprocessor instructions int main() { statements; return 0; }</pre>	<pre>/* Purpose: A program to print Hello World! */ #include <stdio.h> int main() { // begin body printf("Hello World!\n"); return 0; } // end body</pre>

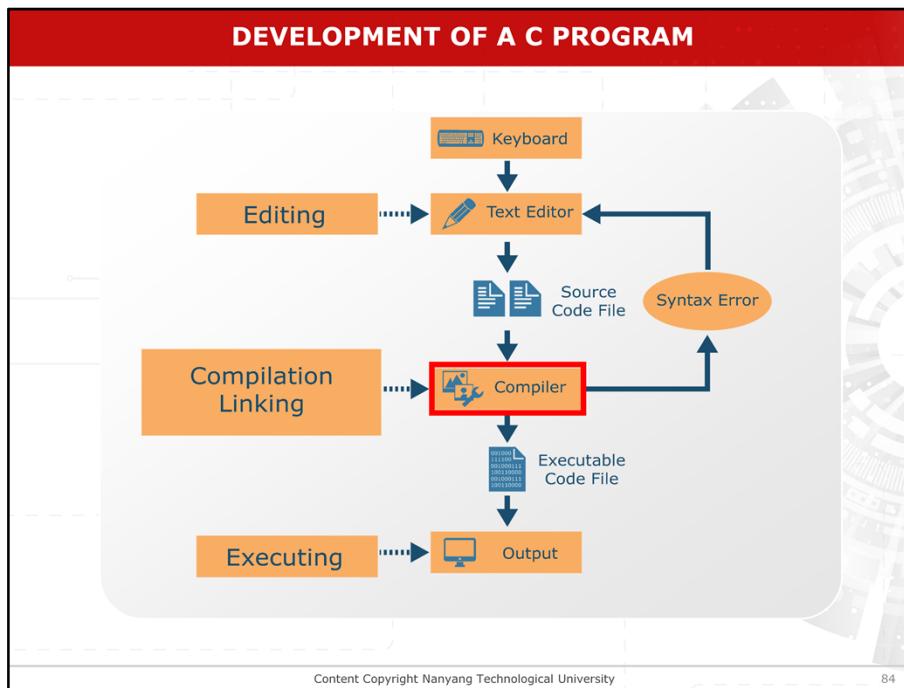
Content Copyright Nanyang Technological University

82

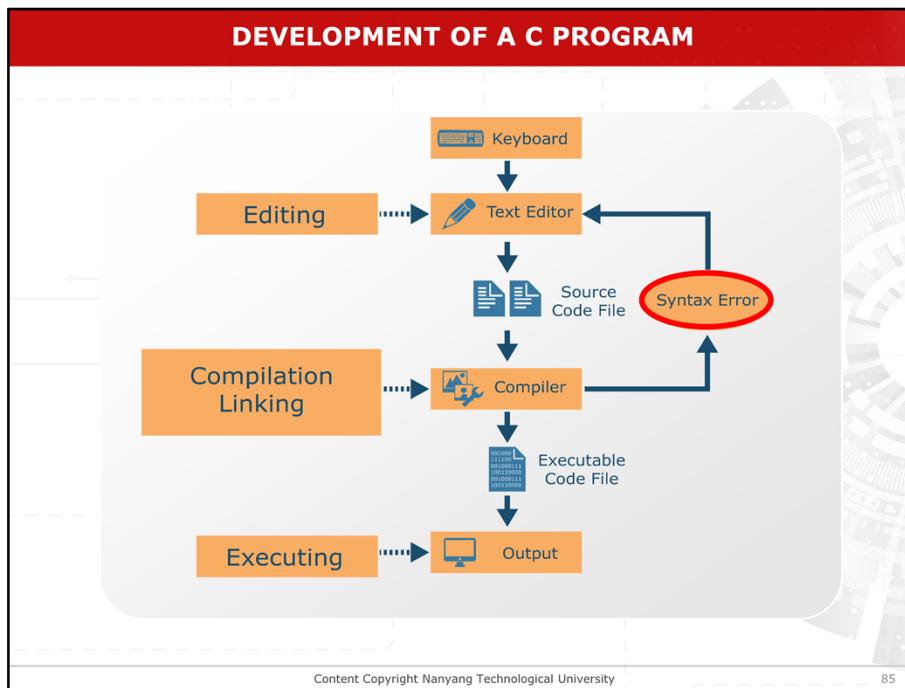
Finally, the last line (i.e. **// end body**) is a comment.



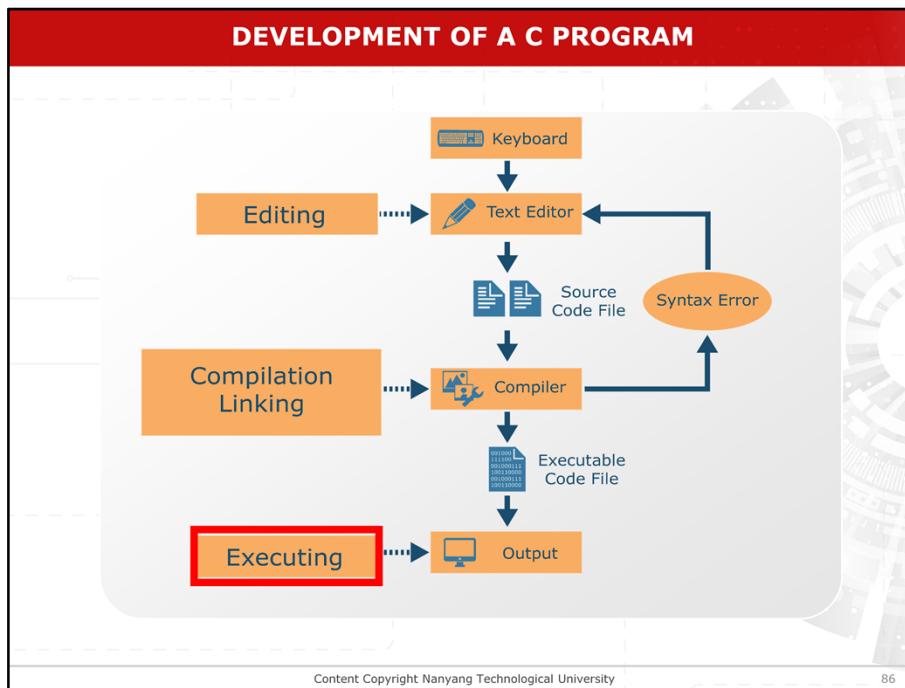
To develop a C program, a text editor is used to create a file containing the source code in C. Most compilers come with editors that can be used to enter and edit source code.



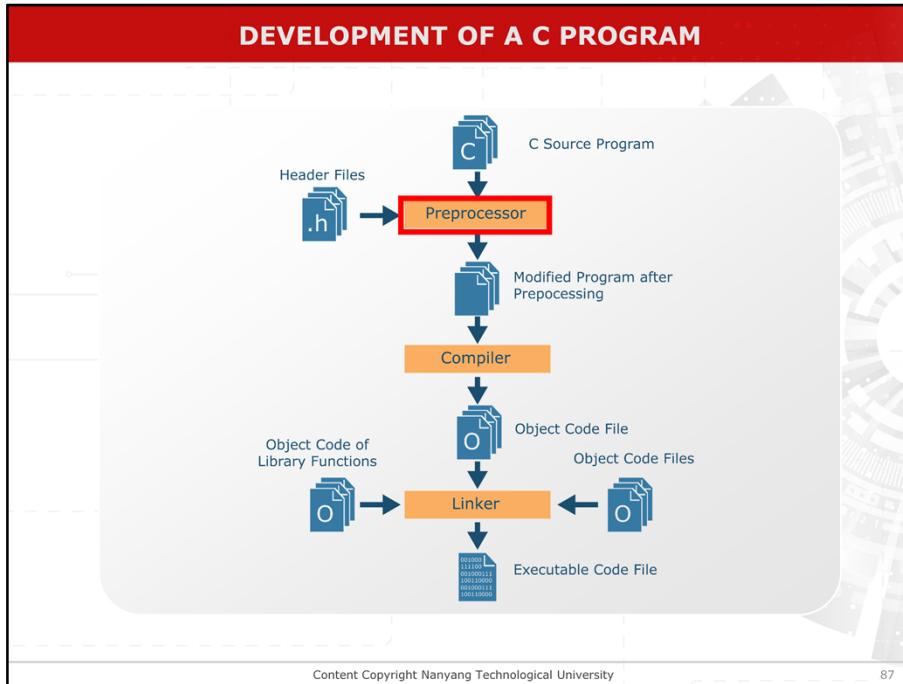
Then, the source code needs to be processed by a *compiler* to generate an object file.



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.

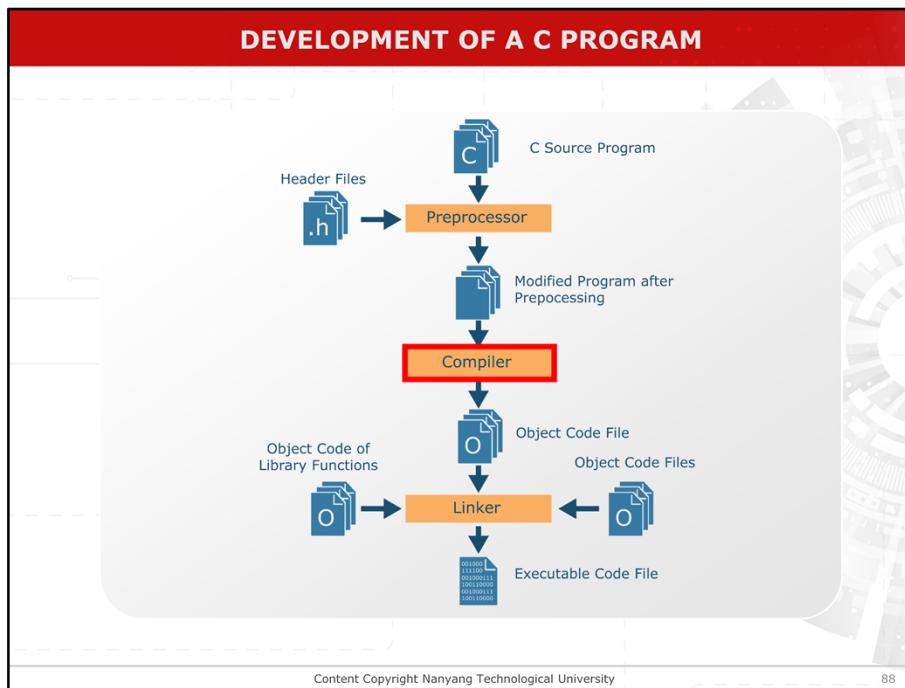


Finally, the executable file can be run and tested.

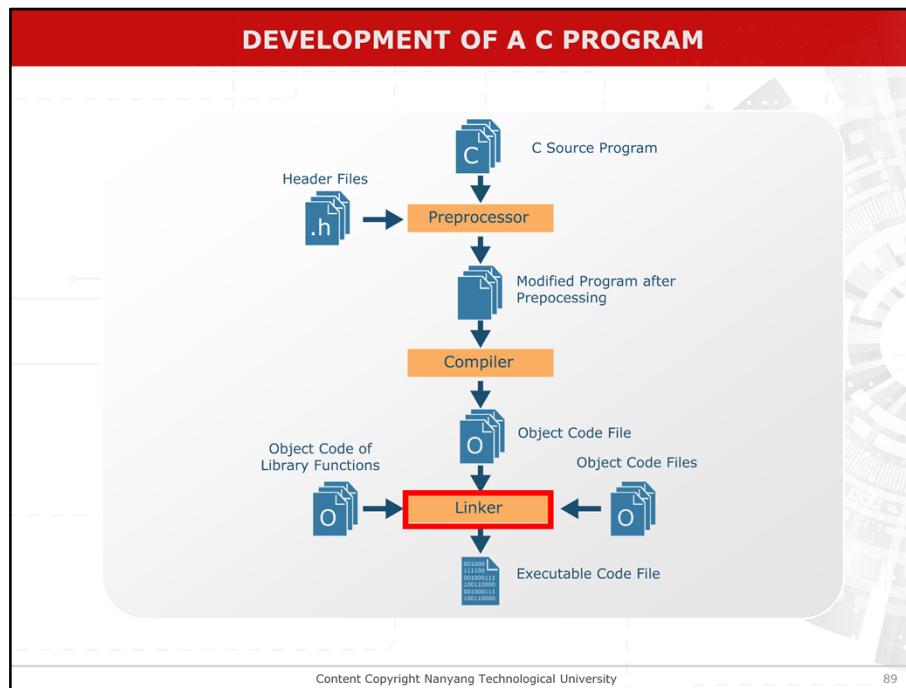


After writing the C program and typing it into the editor, the program needs to be processed by

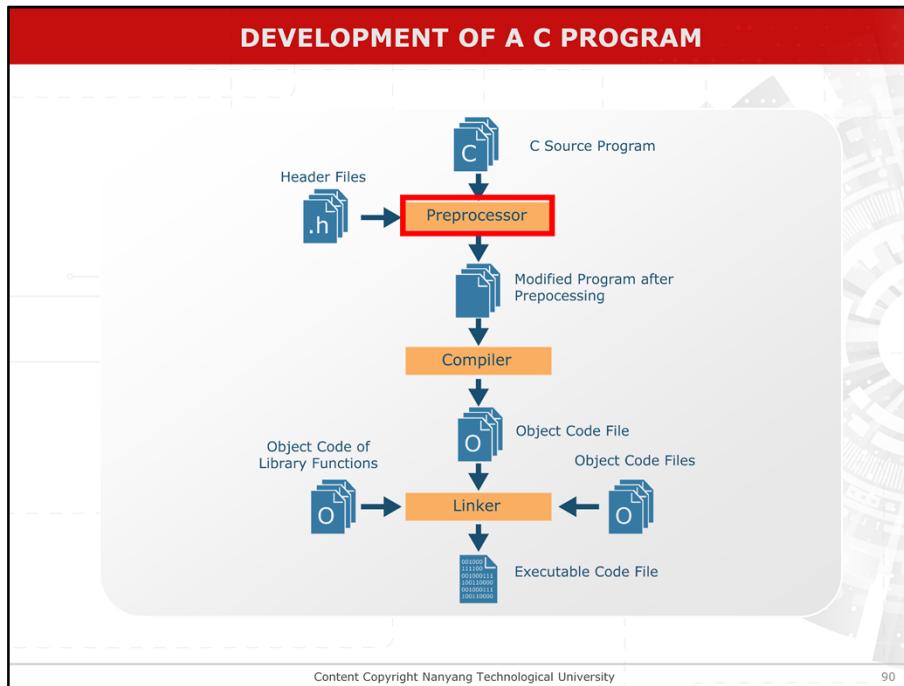
(1)preprocessor



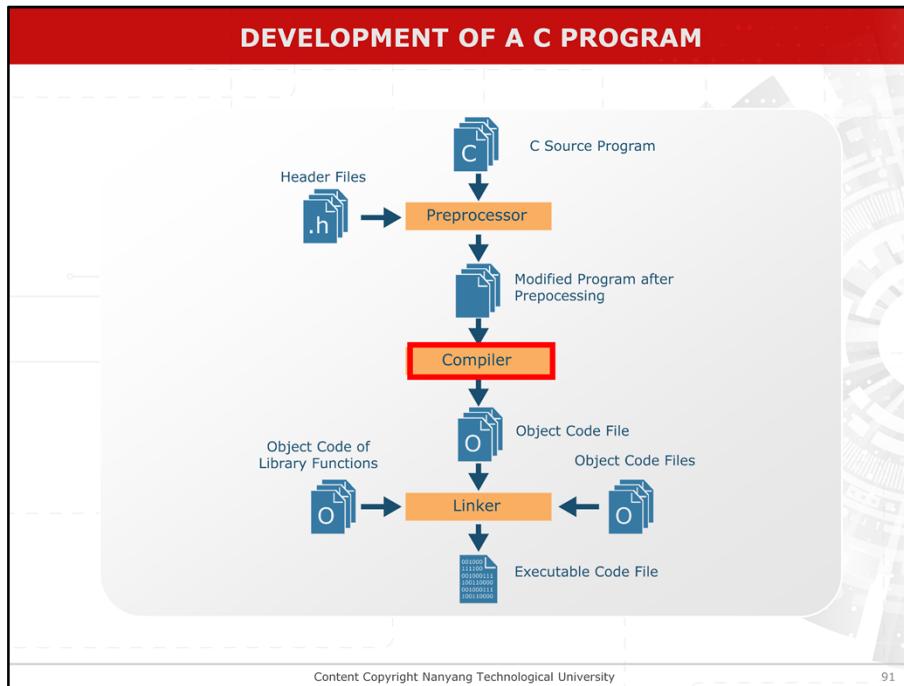
compiler



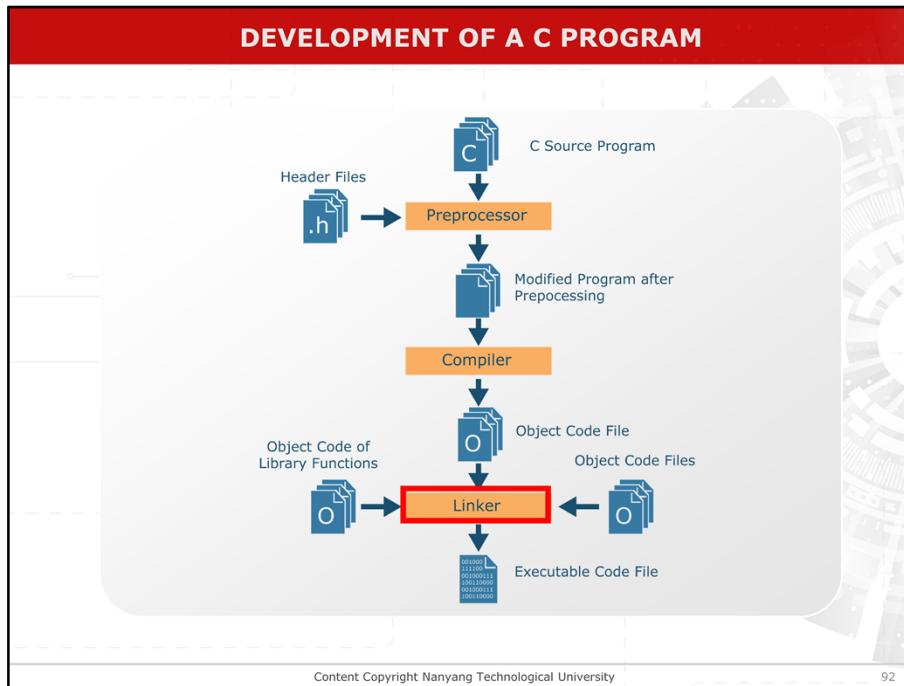
linker before you can execute the program.



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.

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

- **Microsoft Visual Studio C/C++** (Dreamspark for Student, <https://www.dreamspark.com/>)
- **Code::Blocks** (<http://www.codeblocks.org/>)
- **Bloodshed Dev C++** (<http://www.bloodshed.net/>)
- **JGRASP** (<http://www.jgrasp.org/>)

Content Copyright Nanyang Technological University 93

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.

SUMMARY

After this lesson, you should be able to:

- Explain the structure of a C program
- Describe the various steps involved in development of a C program
- Explain in detail the process involved in compilation and linking
- Discuss the various Integrated Development Environments (IDE)

Content Copyright Nanyang Technological University 94

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