Section 7: Diving In

▼ 7.1 Project Template

- **Reuse `.vscode` Folder**:
  - Copy the `.vscode` folder from the project template. This allows you to avoid reconfiguring VS Code settings each time you create a new project.
- **Explore VS Code Settings**:
  - **Themes**: Customize the editor's appearance to your preference.
  - **Settings**: Adjust various preferences such as font size, indentation, and other editor behaviors.
  - **Inbuilt Terminal**: Learn how to use and configure the integrated terminal for a more streamlined workflow.

▼ 7.2 Your First C++ Program

- Create a new folder and copy-paste the configuration files.
- Open VS Code through Developer PowerShell/CMD.

```
#include <iostream>  // Include the iostream library for input and output operations

int main() {
    // Output "Number 1" followed by a newline character to move to the next line
    std::cout << "Number 1" << std::endl;

    // Output "Number 2" followed by a newline character
    std::cout << "Number 2" << "\n";

    // Output "Number 3" followed by a newline character
    std::cout << "Number 3" << std::endl;

    // Return 0 to indicate that the program ended successfully
    return 0;
}
```

▼ 7.3 Comments

- Comments are lines of text that are not executed by the compiler/interpreter.
- They are generally used to explain the code and make it easier to understand.
- Nesting of block comments is not allowed.

Summary

1. `//` Comments out a single line.
2. `/* ..... */` Blocks comments out a block of text.
3. Block comments can't be nested.
4. Use comments to document your code; don't overdo it, though.

```
// 7.3 Comments
#include <iostream>
```

```
int main(){

    // What are comments?
    // Comments are

    //One line comment

    /*
        Multi
        Line
        Comment

    /*
        Nesting isn't allowed
    */
    */
}
```

▼ 7.4 Errors

Errors: Mistakes that occur during the development of a program or software application.

There are 3 types:

1. **Compile Time Errors**: The program won't compile.
2. **Runtime Errors**: The compilation will be successful, but an error appears when you try to run the program.
3. **Warnings**: Indicate potential issues, such as dividing by zero.

Note: Runtime errors are generally worse than compile-time errors because they produce a binary executable file that does not perform as intended.

```
#include <iostream>

int main() {
    // 1. Compile Time Error: Skipping a semicolon will result in a compile-time error.
    // These errors can be seen in the Problems tab.
    // You won't get a binary executable file if there is a compile-time error.
    // std::cout << "Hello World!" << std::endl  // Missing semicolon

    // 2. Runtime Error and Warning
    // These can be seen in the Problems tab.
    // Code starts here
    std::cout << "Hello World!" << std::endl;
    int value = 7 / 0;  // Division by zero causes runtime error and warning
    std::cout << "Value is: " << value << std::endl;
    // Code ends here

    // You will get a binary file in this case with a WARNING like this:
    /*
    C:\Users\harvy\Desktop\cpp_kagwaya\SEC7_Diving_in\7_4_Errors\main.cpp:13:18: warning: div
   13 |     int value = 7 / 0;
    */
    // Although you can execute the binary file, the intended output will not be obtained.
    // In this case, it will not print the "value".

    return 0;
}
```

▼ 7.5 Statements

- **Definition**: A statement is a basic unit of computation in a C++ program.
- **Program Structure**: Every C++ program is a collection of statements organized in a specific way to achieve a particular goal.
- **Termination**: Statements end with a semicolon (`;`).
- **Execution Order**: Statements are executed in order from top to bottom when the program runs.
- **Termination**: Execution continues until a statement causes the program to terminate or triggers another sequence of statements.

```
#include <iostream>

//function 1
int addNumbers(int first_param, int second_param){
    int sum = first_param + second_param;
    return sum;
}

//Exercise
int mulNumbers(int num1,int num2){
    return num1*num2;
}

int main(int argc, char **argv){
    //statement 1
    int firstNumber {12};


    //statement 2
    int secondNumber {7};

    //statement 3 and so on
    std::cout<<"The firstNumber is :"<<firstNumber<<std::endl;
    std::cout<<"The secondNumber is :"<<secondNumber<<std::endl;

    int sum = firstNumber + secondNumber;
    std::cout << "The sum of the two numbers is : " << sum << std::endl;


    //calling the function addNumbers and storing the result in sum
    sum = addNumbers(firstNumber,secondNumber);
    std::cout << "The sum of the two numbers is : " << sum << std::endl;

    sum = addNumbers(50,100);
    std::cout << "The sum of the two numbers is : " << sum << std::endl;

    //Using functions without storing it any variable
    std::cout << "The sum2 of the two numbers is : " << addNumbers(24,4) << std::endl;


    //Exercise
    std::cout<<"The multiplication of "<<firstNumber<<" and "<<secondNumber<<" is ";
    std::cout<<mulNumbers(firstNumber,secondNumber)<<std::endl;

    return 0;
}
```

> **Note**:
>
> - **Function Definition**: In simple terms, a function is a block of code that performs a specific task. Functions must be defined before they are used in a program.
> - **Benefit**: Functions allow for code reuse, meaning you can call the same block of code from different parts of your program without having to rewrite it

▼ 7.6 Data Input and Output

```
std::cout << "Hello world" << std::endl;
```

- `<<` indicates that data is being sent to the `std::cout` stream.

Streams and Their Purposes:

- `std::cout`: Used for printing data to the console (terminal).
- `std::cin`: Used for reading input from the console (terminal).
- `std::cerr`: Used for printing error messages to the console (terminal).
- `std::clog`: Used for printing log messages to the console (terminal).

▼ `getline` Function

`getline` is used to read an entire line of text from an input stream into a string variable. This is particularly useful when you want to read user input or data that includes spaces.

Usage in Simple Words:

- **Purpose**: Reads a line of text from the input (e.g., from the console) and stores it in a string variable.

```
#include <iostream>
#include <string>

int main() {
    std::string name;
    std::cout << "Enter your name: ";
    std::getline(std::cin, name); // Reads the entire line of input into 'name'
    std::cout << "Hello, " << name << "!" << std::endl;
    return 0;
}
```

▼ Code :

```
#include <iostream>
#include <string>

// function demonstrating chaining std::cin
void gatherDetails(){
    int age;
    std::string name;

    std::cout<< "Please type in your Last name and age, seperated by spaces : "<<std::endl
    std::cin>>name>>age; // Input name and age
    std::cout<<"Hello "<<name<<"! You are "<<age<<"years old."<<std::endl;
}

// int main(){
//     // READING DATA
//     int age;
```

```
//     std::string name;
//     std::cout<<"Please type in your Last name : "<<std::endl;
//     std::cin>>name;

//     std::cout<<"Please type in your age : "<<std::endl;
//     std::cin>>age;
//     std::cout<<"Hello "<<name<<"! You are "<<age<<" years old "<<std::endl;

//     std::cout<<"Calling gatherData function to demonstrate input chaining"<<std::endl;
//     gatherDetails();

//     std::cout<<"#################"<<std::endl;
//     // ############################################################
//     // PRINTING DATA
//     std::cout<<"Hello World!"<<std::endl;

//     int age2 {21};
//     std::cout<<"The age is : "<<age2<<std::endl;

//     // Error
//     std::cerr<<  "std::cerr output : Something went wrong"<<std::endl;

//     // Log message
//     std::clog << "std::clog output : This is a log message"<<std::endl;
// }

int main(){
    std::cout<<"Getline demonstration "<<std::endl;
    std::string full_name;
    int age;

    std::cout<<"Please type in your full name here: "<<std::endl;
    std::getline(std::cin,full_name);
    std::cin>>age;

    std::cout<<"Hello "<<full_name<<"!, You are "<<age<<" years old"<<std::endl;
    return 0;
}
```

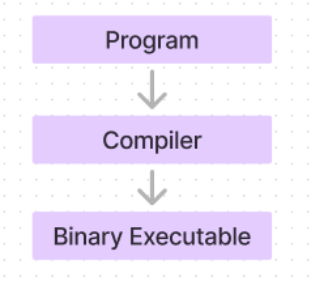
▼ 7.7 Exercise

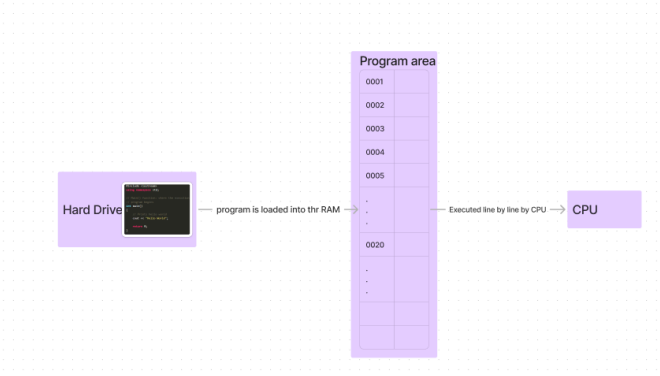
Your task is to print the message "I am the phoenix king!" on the terminal using std::cout. You don't need to put in a new line character using std::endl, just the message.

```
#include <iostream>
// #include "exercise.h"

void say_something(){

    //Don't modify anything above this line
    //Your code will go below this line
    std::cout<<"I am the phoenix king!";
    //Your code will go above this line
    //Don't modify anything below this line
}

int main(){
    say_something();
```

```
    return 0;
}
```

▼ 7.8 C++ Program execution model and memory model





▼ 7.9 C++ core language Vs Standard Library Vs STL

Core Features:

- Foundational elements including basic data types such as `int` and `float`.

Standard Library:

- Highly specialized components for essential functionalities.
- Examples include `cout` and `cin` for input and output operations.

Standard Template Library (STL):

- A crucial part of the C++ standard library.
- Comprises a collection of template classes designed for efficient data handling.
- Includes special types like containers, algorithms, and iterators to facilitate easier and optimized code management.