Basic Syntax in C++

Program Structure

C++ programs run line by line and generally follow the same program structure:

- `#include` statements at the beginning of the program, which allow access to library functionalities.
- `main()` function, which is run when the program is executed.
- `return 0` at the end of the `main()` function, which indicates that the program ran without issues.

```
#include <iostream>
using namespace std;

int main() {
  cout << "Hello, world!";
  return 0;
}
```

Comments

Comments are notes left by the programmer that explain complex code. Comments do not affect the performance of a program because they are ignored by the compiler.
In C++, there are two types of comments:

- Single-line: begin with `//`.
- Multi-line: begin with `/*` and end with `*/`.

As a rule of thumb, comments should always **go above** the code they are commenting on.

```
// I am a single-line comment

/*
I am a
multi-line
comment
*/
```

Input and Output

Input and output make C++ programs more interactive.

- `#include <iostream>` must be placed at the beginning of the program to access input and output.
- `std::cout` is the "character output" and it is used together with `<<` to print to the terminal.
- `std::cin` is the "character input" and it is used together with `>>` to read user input.
- `std::endl` or `\n` can be used to insert a new line.

```
#include <iostream>

int main() {
  int age;
  std::cout << "How old are you? ";
  std::cin >> age;
  std::cout << "You are " << age << "
years old.";

  return 0;
}
```

Variables

Variables are used to store and retrieve data. When declaring a variable, it must be given a data type and a name.
Multiple variables **of the same type** can be declared in a single statement using a comma-separated list.
Variables can be declared with the `const` keyword, which prevents their value from being changed later.

```
int number = 100;

char letter;
letter = 'c';

const int pi = 3.14;
```

References and Pointers

C++ provides two powerful features for memory manipulation:

- References: aliases to existing variables
- Pointers: store memory address as its value

Reference variables are created using the `&` symbol.
`&` is also used to access the memory address of a variable.
Pointer variables are created using the `*` symbol. `*` is also used to obtain the value pointed to by a pointer variable.

```
int year = 2021;

int &ref = year;

int *ptr = &year;

std::cout << &year << "\n";

std::cout << *ptr << "\n";
```

User Input

`std::cin`, which stands for "character input", reads user input from the keyboard.
Here, the user can enter a number, press enter, and that number will get stored in `tip`.

```
int tip = 0;

std::cout << "Enter amount: ";
std::cin >> tip;
```

Variables

A variable refers to a storage location in the computer's memory that one can set aside to save, retrieve, and manipulate data.

```
// Declare a variable
int score;

// Initialize a variable
score = 0;
```

Arithmetic Operators

C++ supports different types of arithmetic operators that can perform common mathematical operations:

- `+` addition
- `-` subtraction
- `*` multiplication
- `/` division
- `%` modulo (yields the remainder)

```
int x = 0;

x = 4 + 2;  // x is now 6
x = 4 - 2;  // x is now 2
x = 4 * 2;  // x is now 8
x = 4 / 2;  // x is now 2
x = 4 % 2;  // x is now 0
```

`double` Type

`double` is a type for storing floating point (decimal) numbers. Double variables typically require 8 bytes of memory space.

```
double price = 8.99;
double pi = 3.14159;
```

Chaining the Output

`std::cout` can output multiple values by chaining them using the output operator `<<`.
Here, the output would be `I'm 28`.

```
int age = 28;

std::cout << "I'm " << age << ".\n";
```

Print Share