5.1 Basic syntax and structure of a C++

A C++ program begins with preprocessor directives, an example of which is including header files. Preprocessor directives provide instructions to the compiler and tell it what additional files to include in the compilation process.

Following the preprocessor directives are declarations, which include variables, constants, and user-defined functions.

The main function is the entry point of any C++ program, and contains all of the program's executable code. Within the main function are more definitions, which are additional declarations of data types, variables, constants, and user-defined functions. Finally, the program is concluded with the return 0 statement, indicating success.

Developing a C++ program requires careful attention to the order in which the preprocessor directives, declarations, main function, and definitions are written. Only by understanding the basic structure of a C++ program can a programmer write effective, efficient, and bug-free code.

Here is an example of a basic C++ program that blinks LED on a 13-th pin of an Arduino Uno controller and can be written in Arguing IDE:

Program 1: Native C++ program for ATmega328.

```
#include <avr/io.h>
#include <util/delay.h>
int time_ms = 1000;     // Variable declaration
void setup();           // Function declaration
void loop();

int main()
{
    setup();            // Function call
    while (true){       // Main LOOP
        loop();
    }
    return 0;
}
void setup() {          // Function definition
    PDDB |= (1<<PINB5);
}
void loop(){
    PORTB |= (1<<PINB5);
    _delay_ms(time_ms);
    PORTB &= !(1<<PINB5);
    _delay_ms(time_ms);
}
```

Programming an Arduino Uno board in native C++ is much more difficult than in Arduino IDE. Arduino IDE makes it easier for users to write and debug code without having to know the details of the underlying hardware. In addition, the IDE provides many additional functions which simplify the usage of additional peripherals and actuators such as serial communication, LCDs, servo motors, step motors… This is especially true and important for beginners.

5.1.1 Tasks:

1. <++>
2. <++>
3. <++>

5.1.2 Questions:

1. <++>
2. <++>
3. <++>

5.1.3 Summary:

5.1.3.1 <++>

5.1.4 Issues:

5.1.4.1 Not including a semicolon at the end of each statement: Every statement in C++ must end with a semicolon. If a semicolon is omitted, the code will not compile correctly.

5.1.4.2 Not properly formatting the code: Properly indenting and spacing code is important in C++ to make the code easier to read. Not formatting the code correctly can lead to syntactical errors.

5.1.4.3 Not using correct capitalization: C++ is a case sensitive language and therefore proper capitalization is important. If the wrong capitalization is used, it can lead to syntax errors.