# CSE 102: Data Structure and Algorithm
# Lab Sheet -1

## C++ Resources

- The www.cplusplus.com web site has a lot of resources. In particular there are a number of good tutorials:
  http://www.cplusplus.com/doc/tutorial
- The www.learncpp.com is also a great website with a tutorial for learning C++:
  http://www.learncpp.com
- The tutorialspoint.com web site is pretty thorough and aimed at a beginner with some programming experience:
  http://www.tutorialspoint.com/cplusplus/index.htm
- The following GeeksForGeeks resources are also very helpful:
  C++ Programming Language - GeeksforGeeks

Note: Please pick up one resource and try to follow it end to end.

## Introduction to C++ Programming Language

**C++** is a general-purpose programming language that was developed as an enhancement of the C language to include object-oriented paradigm. It is an imperative and a **compiled** language.

1. C++ is a high-level, general-purpose programming language designed for system and application programming. It was developed by Bjarne Stroustrup at Bell Labs in 1983 as an extension of the C programming language. C++ is an object-oriented, multi-paradigm language that supports procedural, functional, and generic programming styles.
2. One of the key features of C++ is its ability to support low-level, system-level programming, making it suitable for developing operating systems, device drivers, and other system software. At the same time, C++ also provides a rich set of libraries and features for high-level application programming, making it a popular choice for developing desktop applications, video games, and other complex applications.
3. C++ has a large, active community of developers and users, and a wealth of resources and tools available for learning and using the language. Some of the key features of C++ include:

4. Object-Oriented Programming: C++ supports object-oriented programming, allowing developers to create classes and objects and to define methods and properties for these objects.
5. Templates: C++ templates allow developers to write generic code that can work with any data type, making it easier to write reusable and flexible code.
6. Standard Template Library (STL): The STL provides a wide range of containers and algorithms for working with data, making it easier to write efficient and effective code.
7. Exception Handling: C++ provides robust exception handling capabilities, making it easier to write code that can handle errors and unexpected situations.

Overall, C++ is a powerful and versatile programming language that is widely used for a range of applications and is well-suited for both low-level system programming and high-level application development.

**Here is a simple C++ code examples to help you understand the language:**

**1.Hello World:**

```
#include <iostream>

int main() {
   std::cout << "Hello, World!" << std::endl;
   return 0;
}
```

## C++ Basic Syntax

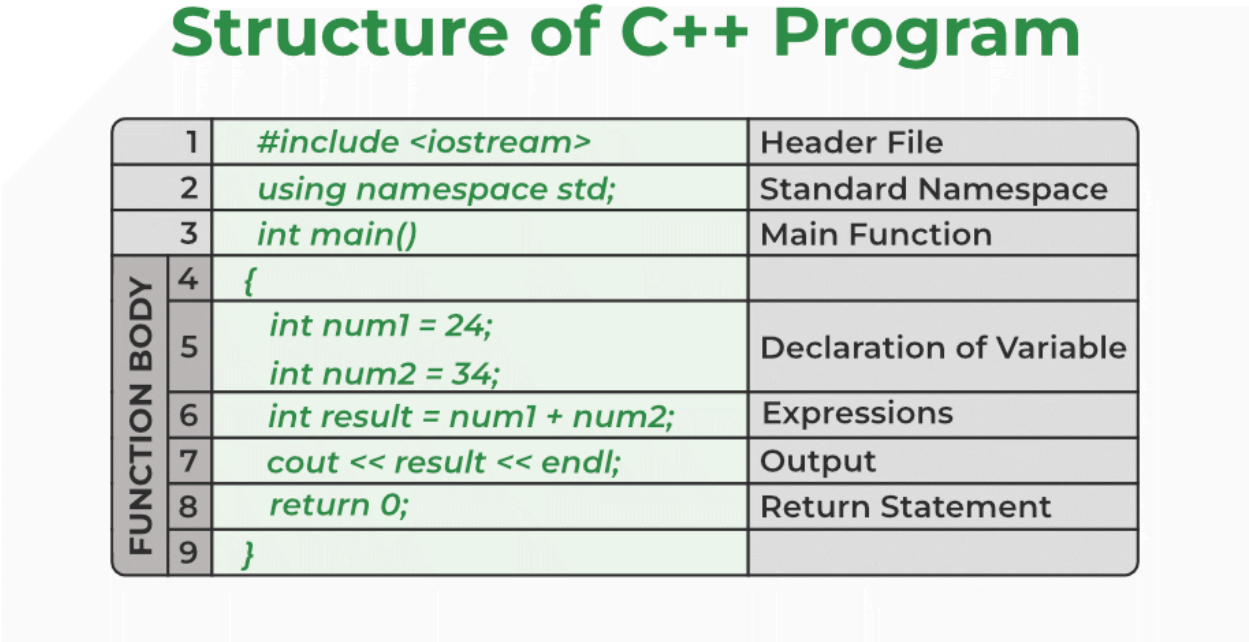
**What is Syntax?**

Syntax refers to the rules and regulations for writing statements in a programming language. They can also be viewed as the grammatical rules defining the structure of a programming language.

The C++ language also has its syntax for the functionalities it provides. Different statements have different syntax specifying their usage but C++ programs also have basic syntax rules that are followed throughout all the programs.

Basic Syntax of a C++ Program

We can learn about basic C++ Syntax using the following program.




The image above shows the basic C++ program that contains header files, main function, namespace declaration, etc.

C++ programming language supports both procedural-oriented and object-oriented programming. The above example is based on the procedural-oriented programming paradigm. So let's take another example to discuss **Object Oriented Programming in C++.** The concepts of object oriented programming are outside the scope of this course.

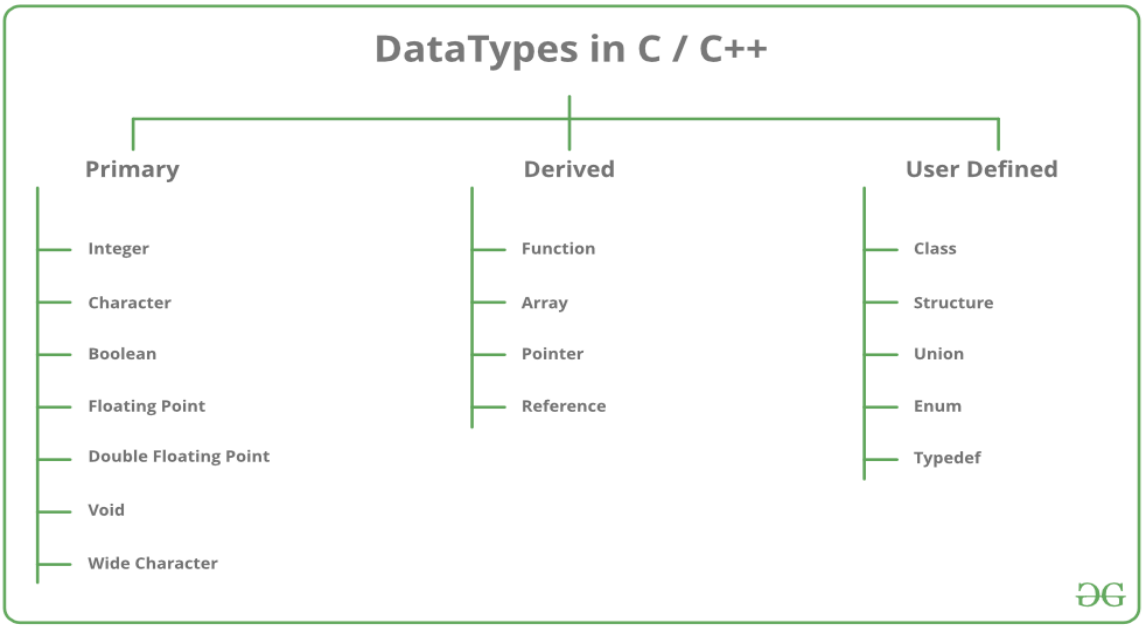
## Structure of Object Oriented Program

| | | | |
|---|---|---|---|
| | 1 | `#include <iostream>` | Header File |
| | 2 | `using namespace std;` | Standard Namespace |
| | 3 | `class Calculate` | Class Declaration |
| | 3 | `{` | |
| CLASS BODY | 4 | `public:` | Access Modifiers |
| | 5 | `int num1 = 50;`<br>`int num2 = 30;` | Data Members |
| | 6 | `int addition() {`<br>`int result = num1 + num2;`<br>`cout << result << endl;`<br>`}` | Member Function |
| | 7 | `};` | |
| | 8 | `int main() {` | |
| | 9 | `Calculate add;` | Object Declaration |
| | 10 | `add.addition();` | Member Function Call |
| | 11 | `return 0;`<br>`}` | Return Statement |

## C++ Data Types

C++ supports the following data types:

1. Primary or Built-in or Fundamental data type
2. Derived data types
3. User-defined data types



**1. Primitive Data Types:** These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char, float, bool, etc. Primitive data types available in C++ are:

- Integer
- Character
- Boolean
- Floating Point
- Double Floating Point
- Valueless or Void
- Wide Character

**2. Derived Data Types:** Derived data types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:

- Function
- Array
- Pointer
- Reference

**3. Abstract or User-Defined Data Types:** Abstract or User-Defined data types are defined by the user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:

- Class
- Structure
- Union
- Enumeration
- Typedef defined Datatype

**Primitive Data Types**

- **Integer:** The keyword used for integer data types is int. Integers typically require 4 bytes of memory space and range from -2147483648 to 2147483647.

- **Character:** Character data type is used for storing characters. The keyword used for the character data type is char. Characters typically require 1 byte of memory space and range from -128 to 127 or 0 to 255.
- **Boolean:** Boolean data type is used for storing Boolean or logical values. A Boolean variable can store either *true* or *false*. The keyword used for the Boolean data type is bool.
- **Floating Point:** Floating Point data type is used for storing single-precision floating-point values or decimal values. The keyword used for the floating-point data type is float. Float variables typically require 4 bytes of memory space.
- **Double Floating Point:** Double Floating Point data type is used for storing double-precision floating-point values or decimal values. The keyword used for the double floating-point data type is double. Double variables typically require 8 bytes of memory space.
- **void:** Void means without any value. void data type represents a valueless entity. A void data type is used for those function which does not return a value.
- **Wide Character:** Wide character data type is also a character data type but this data type has a size greater than the normal 8-bit data type. Represented by wchar_t. It is generally 2 or 4 bytes long.
- **sizeof() operator:** sizeof() operator is used to find the number of bytes occupied by a variable/data type in computer memory.

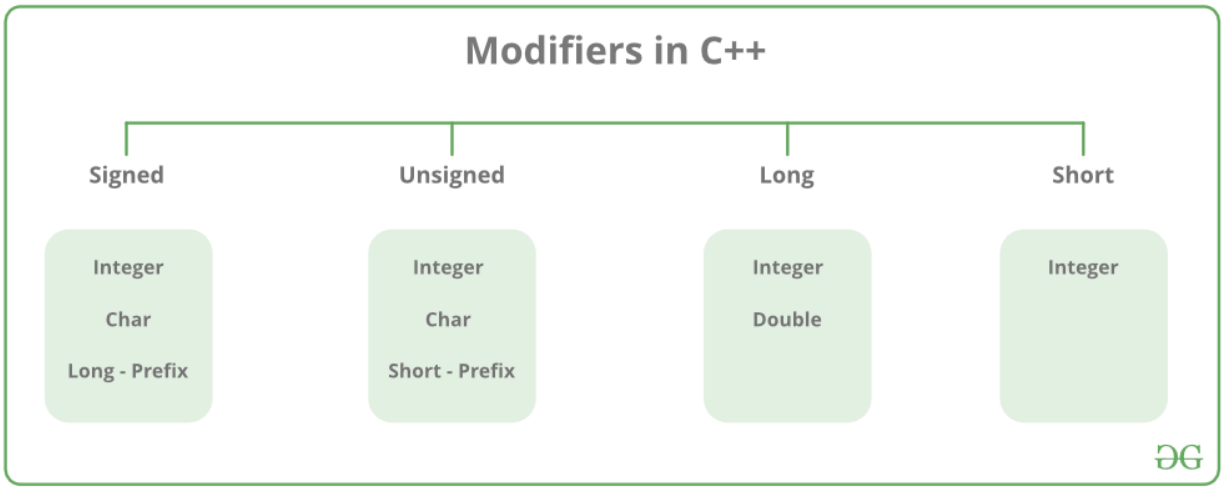
**Example:**

```
// C++ Program to Demonstrate the correct size
// of various data types on your computer.
#include <iostream>
using namespace std;

int main()
{
   cout << "Size of char : " << sizeof(char) << endl;
   cout << "Size of int : " << sizeof(int) << endl;

   cout << "Size of long : " << sizeof(long) << endl;
   cout << "Size of float : " << sizeof(float) << endl;
```

```
    cout << "Size of double : " << sizeof(double) << endl;

    return 0;
}
```

**Datatype Modifiers**

As the name suggests, datatype modifiers are used with built-in data types to modify the length of data that a particular data type can hold.




Data type modifiers available in C++ are:

- Signed
- Unsigned
- Short
- Long

The below table summarizes the modified size and range of built-in datatypes when combined with the type modifiers:

| Data Type | Size (in bytes) | Range |
|---|---|---|

| | | |
|---|---|---|
| short int | 2 | -32,768 to 32,767 |
| unsigned short int | 2 | 0 to 65,535 |
| unsigned int | 4 | 0 to 4,294,967,295 |
| int | 4 | -2,147,483,648 to 2,147,483,647 |
| long int | 4 | -2,147,483,648 to 2,147,483,647 |
| unsigned long int | 4 | 0 to 4,294,967,295 |

| | | |
|---|---|---|
| long long int | 8 | -(2^63) to (2^63)-1 |
| unsigned long long int | 8 | 0 to 18,446,744,073,709,551,615 |
| signed char | 1 | -128 to 127 |
| unsigned char | 1 | 0 to 255 |
| float | 4 | -3.4×10^38 to 3.4×10^38 |
| double | 8 | -1.7×10^308 to1.7×10^308 |

| | | |
|---|---|---|
| long double | 12 | -1.1×10^4932 to1.1×10^4932 |
| wchar_t | 2 or 4 | 1 wide character |

**Macro Constants**

| Name | Expresses |
|---|---|
| **CHAR_MIN** | **The minimum value for an object of type char** |
| **CHAR_MAX** | **Maximum value for an object of type char** |
| **SCHAR_MIN** | **The minimum value for an object of type Signed char** |

| | |
|---|---|
| **SCHAR_MAX** | **Maximum value for an object of type Signed char** |
| **UCHAR_MAX** | **Maximum value for an object of type Unsigned char** |
| **CHAR_BIT** | **Number of bits in a char object** |
| **MB_LEN_MAX** | **Maximum number of bytes in a multi-byte character** |
| **SHRT_MIN** | **The minimum value for an object of type short int** |
| **SHRT_MAX** | **Maximum value for an object of type short int** |

| | |
|---|---|
| **USHRT_MAX** | **Maximum value for an object of type Unsigned short int** |
| **INT_MIN** | **The minimum value for an object of type int** |
| **INT_MAX** | **Maximum value for an object of type int** |
| **UINT_MAX** | **Maximum value for an object of type Unsigned int** |
| **LONG_MIN** | **The minimum value for an object of type long int** |
| **LONG_MAX** | **Maximum value for an object of type long int** |

| | |
|---|---|
| ULONG_MAX | Maximum value for an object of type Unsigned long int |
| LLONG_MIN | The minimum value for an object of type long long int |
| LLONG_MAX | Maximum value for an object of type long long int |
| ULLONG_MAX | Maximum value for an object of type Unsigned long long int |

**Example 1:**

```
// C++ program to Demonstrate the sizes of data types
#include <iostream>
#include <limits.h>
using namespace std;

int main()
{
   cout << "Size of char : " << sizeof(char) << " byte"
        << endl;

   cout << "char minimum value: " << CHAR_MIN << endl;

   cout << "char maximum value: " << CHAR_MAX << endl;
```

```
    cout << "Size of int : " << sizeof(int) << " bytes"
         << endl;

    cout << "Size of short int : " << sizeof(short int)
         << " bytes" << endl;

    cout << "Size of long int : " << sizeof(long int)
         << " bytes" << endl;

    cout << "Size of signed long int : "
         << sizeof(signed long int) << " bytes" << endl;

    cout << "Size of unsigned long int : "
         << sizeof(unsigned long int) << " bytes" << endl;

    cout << "Size of float : " << sizeof(float) << " bytes"
         << endl;

    cout << "Size of double : " << sizeof(double)
         << " bytes" << endl;

    cout << "Size of wchar_t : " << sizeof(wchar_t)
         << " bytes" << endl;

    return 0;
}
```

**Example 2:**

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

int main() {
  // Integer data types
  int a = 10;
  short b = 20;
  long c = 30;
  long long d = 40;
  cout << "Integer data types: " << endl;
  cout << "int: " << a << endl;
  cout << "short: " << b << endl;
```

```
    cout << "long: " << c << endl;
    cout << "long long: " << d << endl;

    // Floating-point data types
    float e = 3.14f;
    double f = 3.141592;
    long double g = 3.14159265358979L;
    cout << "Floating-point data types: " << endl;
    cout << "float: " << e << endl;
    cout << "double: " << f << endl;
    cout << "long double: " << g << endl;

    // Character data types
    char h = 'a';
    wchar_t i = L'b';
    char16_t j = u'c';
    char32_t k = U'd';
    cout << "Character data types: " << endl;
    cout << "char: " << h << endl;
    wcout << "wchar_t: " << i << endl;
    cout << "char16_t: " << j << endl;
    cout << "char32_t: " << k << endl;

    // Boolean data type
    bool l = true;
    bool m = false;
    cout << "Boolean data type: " << endl;
    cout << "true: " << l << endl;
    cout << "false: " << m << endl;

    // String data type
    string n = "Hello, world!";
    cout << "String data type: " << endl;
    cout << n << endl;

    return 0;
}
```

## Operators in C++

An operator is a symbol that operates on a value to perform specific mathematical or logical computations. They form the foundation of any programming language. In C++, we have built-in operators to provide the required functionality.

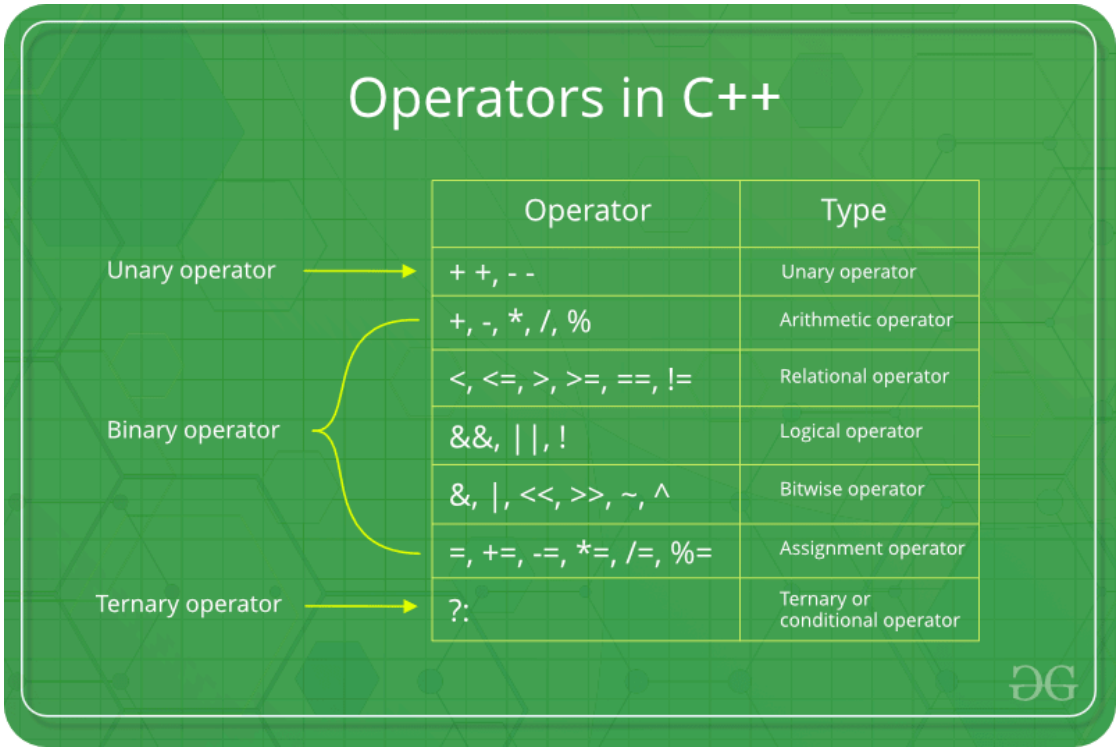
An operator operates the operands. For example,

int c = a + b;

Here, '+' is the addition operator. 'a' and 'b' are the operands that are being 'added'.

Operators in C++ can be classified into 6 types:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Ternary or Conditional Operators



## Operator Precedence Chart

| Precedence | Operator | Description | Associativity |
|---|---|---|---|
| 1. | () | Parentheses (function call) | left-to-right |
| | [] | Brackets (array subscript) | |
| | . | Member selection via object name | |
| | -> | Member selection via a pointer | |

| | | | |
|---|---|---|---|
| | ++/– | Postfix increment/decrement | |
| 2. | ++/– | Prefix increment/decrement | right-to-left |
| | +/- | Unary plus/minus | |
| | !~ | Logical negation/bitwise complement | |
| | (type) | Cast (convert value to temporary value of type) | |
| | * | Dereference | |

| | | | |
|---|---|---|---|
| | & | Address (of operand) | |
| | sizeof | Determine size in bytes on this implementation | |
| 3. | *,/,% | Multiplication/division/modulus | left-to-right |
| 4. | +/- | Addition/subtraction | left-to-right |
| 5. | << , >> | Bitwise shift left, Bitwise shift right | left-to-right |

| | | | |
|---|---|---|---|
| 6. | < , <= | Relational less than/less than or equal to | left-to-right |
| | > , >= | Relational greater than/greater than or equal to | left-to-right |
| 7. | == , != | Relational is equal to/is not equal to | left-to-right |
| 8. | & | Bitwise AND | left-to-right |
| 9. | ^ | Bitwise exclusive OR | left-to-right |
| 10. | \| | Bitwise inclusive OR | left-to-right |

| | | | |
|---|---|---|---|
| 11. | && | Logical AND | left-to-right |
| 12. | \|\| | Logical OR | left-to-right |
| 13. | ?: | Ternary conditional | right-to-left |
| 14. | = | Assignment | right-to-left |
| | += , -= | Addition/subtraction assignment | |
| | *= , /= | Multiplication/division assignment | |

| | | | |
|---|---|---|---|
| | %= , &= | Modulus/bitwise AND assignment | |
| | ^= , \|= | Bitwise exclusive/inclusive OR assignment | |
| | <>= | Bitwise shift left/right assignment | |
| 15. | , | expression separator | left-to-right |

# Exercises (1X10 = 10)

Write C++ programs to perform the following tasks.

Q1.

Input two numbers and work out their sum, average and sum of the squares of the numbers.

Q2.

Input and output your name, address and age to an appropriate structure.

Q3.

Write a program that works out the largest and smallest values from a set of 10 inputted numbers.

Q4.

Write a program to read a "float" representing a number of degrees Celsius, and print as a "float" the equivalent temperature in degrees Fahrenheit. Print your results in a form such as

100.0 degrees Celsius converts to 212.0 degrees Fahrenheit.

Q5.

Write a program to print several lines (such as your name and address). You may use either several cout instructions, each with a newline character in it, or one cout with several newlines in the string.

Q6.

Write a program to read a positive integer at least equal to 3, and print out all possible permutations of three positive integers less or equal to than this value.

Q7.

Write a program to read a number of units of length (a float) and print out the area of a circle of that radius. Assume that the value of pi is 3.14159 (an appropriate declaration will be given you by ceilidh - select setup).

Your output should take the form: The area of a circle of radius ... units is .... units.

Q8.

Given as input a floating (real) number of centimeters, print out the equivalent number of feet (integer) and inches (floating, 1 decimal), with the inches given to an accuracy of one decimal place.

Assume 2.54 centimeters per inch, and 12 inches per foot.

If the input value is 333.3, the output format should be:

333.3 centimeters is 10 feet 11.2 inches.

Q9.

Given as input an integer number of seconds, print as output the equivalent time in hours, minutes and seconds. Recommended output format is something like

7322 seconds is equivalent to 2 hours 2 minutes 2 seconds.

Q10.

Write a program to read two integers with the following significance.

The first integer value represents a time of day on a 24 hour clock, so that 1245 represents quarter to one mid-day, for example.

The second integer represents a time duration in a similar way, so that 345 represents three hours and 45 minutes.

This duration is to be added to the first time, and the result printed out in the same notation, in this case 1630 which is the time 3 hours and 45 minutes after 12.45.

Typical output might be Start time is 1415. Duration is 50. End time is 1505.

Start time is 2300. Duration is 200. End time is 100.