Computer Programming Fundamentals

Handouts No. 4

1. C++ Program Structure

A C++ program is structured in a specific and particular manner. In C++, a program is divided into the following three sections:

- Standard Libraries Section
- Main Function Section
- Function Body Section

For example, let's look at the implementation of the **Hello World** program:

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Hello World!";
  return 0;
}
```

Standard libraries section

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

#include is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
The file <iostream>, which is a standard file that should come with the C++ compiler, is short for input-output streams. This command contains code for displaying and getting an input from the user.

namespace is a prefix that is applied to all the names in a certain set. iostream file defines one name used in this program – cout. This code is saying: Use the cout tool from the std toolbox.

Main function section

```
int main()
{

}
```

The starting point of all C++ programs is the main function. This function is called by the operating system when your program is executed by the computer.

{ signifies the start of a block of code, and **}** signifies the end.

Function body section

```
cout << "Hello World" << endl;
return 0;
```

The name **cout** is short for **character output** and displays whatever is between the << brackets.
Symbols such as << can also behave like functions and are used with the keyword **cout**.

The **return** keyword tells the program to return a value to the function **int main**. After the **return** statement, execution control returns to the operating system component that launched this program. Execution of the code terminates here.

2. Variables

2.1. Declaring a Variable

Variables are used for storing data during program execution. C++ has different data types depending on what data you need to store. Some simple types are:

int	Integer numbers.
double	Floating-point (also called real numbers).
char	Characters (as Integer)

Before a variable can be used, it first has to be declared. To declare a variable, you start with the data type you want the variable to hold followed by an identifier, which is the name of the variable.

```
int myInt;
```

2.2.Naming Variables' Rules

The identifier can consist of letters, numbers, and underscores, but:

- It cannot start with a number.
- It also cannot contain spaces or special characters.
- It must not be a reserved keyword. (see reserved words list below)

below are examples of incorrect identifier names:

```
int 32Int;
int my Int;
int Int@32;
int int;
```

Note that C++ is a case sensitive programming language, so uppercase and lowercase letters have different meanings.

alignas (since C++11)	atomic_noexcept (TM	case
alignof (since C++11)	TS)	catch
and	auto (1)	char
and_eq	bitand	char8_t (since C++20)
asm	bitor	char16_t (since C++11)
atomic_cancel (TM TS)	bool	char32_t (since C++11)
atomic_commit (TM TS)	break	class (1)

compl	if	static_assert (since C++11)
concept (since C++20)	inline (1)	static_cast
const	int	struct (1)
consteval (since C++20)	long	switch
constexpr (since C++11)	mutable (1)	synchronized (TM TS)
constinit (since C++20)	namespace	template
const_cast	new	this
continue	noexcept (since C++11)	thread_local (since C++11)
co_await (since C++20)	not	throw
co_return (since C++20)	not_eq	true
co_yield (since C++20)	nullptr (since C++11)	try
decltype (since C++11)	operator	typedef
default (1)	or	typeid
delete (1)	or_eq	typename
do	private	union
double	protected	unsigned
dynamic_cast	public	using (1)
else	reflexpr (reflection TS)	virtual
enum	register (2)	void
explicit	reinterpret_cast	volatile
export (1) (3)	requires (since C++20)	wchar_t
extern (1)	return	while
false	short	xor
float	signed	xor_eq
for	sizeof (1)	
friend	static	
goto		

Table 1: C++ Reserved keywords

2.3. Assignment Operator

To assign a value to a declared variable the equal sign is used, which is known as the assignment operator (**=**). This is called assigning or initializing the variable.

```
myInt = 50;
```

The declaration and assignment can be combined into a single statement. When a variable is assigned a value it then becomes defined.

```
int myInt = 50;
```

If you need to create more than one variable of the same type, there is a shorthand way of doing it using the comma operator (**,**).

```
int x = 1, y = 2, z;
```

Once a variable has been defined (declared and assigned), you can use it by simply referencing the variable's name: for example, to copy the value to another variable.

```
int a = x;
```

2.4. Display Variable Value

In addition to strings the predefined object **cout** can be used to print values and variables to the standard output stream (screen) as seen in the following example.

Example 2:

```
#include <iostream>
using namespace std;
int main()
{
int x = 8;
cout<<"x is "<<x;
return 0;
}
```

2.5. Summation Operation

Two variables values can be summed together using the addition operator (+). The value of the addition operation can be saved in a variable by using the assignment operator (=).

Example 3A:

```
#include <iostream>
using namespace std;
int main() {
int num1 = 8;
int num2 = 10;
cout<<"summation of the two numbers are "<<num1+num2;
return 0;
}
```

Example 3B:

```
#include <iostream>
using namespace std;
int main()
{
int num1 = 8;
int num2 = 10;
int sum;
sum = num1 + num2;
cout<<"summation of the two numbers are "<<sum;
return 0;
}
```

Example 3C:

```
#include <iostream>
using namespace std;
int main()
{
int num1 = 8 ;
int num2 = 10 ;
int sum;
sum = num1 + num2 ;
cout<<"The summation of"<<num1<<" and "<<num2<<" are "<<sum;
return 0;
}
```

2.6. Floating-Points Numbers

The floating-point types can store real numbers. the keyword (**double**) is used to declare a variable that can store real numbers.

```
double num ;
```

a number with a floating point can be assigned to the double variable as follows:

```
num=1.25 ;
```

in order to display the value that is already stored in a double variable, we write the following line:

```
cout<<"The floating-point number is " <<num;
```

Example 4: Write a C++ program to find the summation of two real numbers, which are 0.25 and 2.6

```
#include <iostream>
using namespace std;
int main()
{
double sum;
sum = 0.25 + 2.6 ;
cout<<"The summation is "<< sum ;
return 0;
}
```

2.7.Constants

Variables are named variables because their value are changing during the execution of the program. However, in C++, you can define a constant storage as well by placing the keyword (const) in front of the variable declaration line. Value must be assigned to a constant on the declaration line. If no value has been assigned to a constant it will take the default value.

Data Type	Default Value
int	0
double	0.000000
char	null

Table 3: Default values

Example 5A:

```
#include <iostream>
using namespace std;
int main()
{
int a = 4;
const int b =6;
a=12;
b = 10;
cout<<"Constant number is "<< b;
return 0;
}
```

Output:

```
error: assignment of read-only variable ‘b’
        b = 10;
          ^
```

Example 5B:

```
#include <iostream>
using namespace std;
int main()
{
const int b =6;
cout<<"Constant number is "<< b;
return 0;
}
```

Example 5C:

```
#include <iostream>
using namespace std;
int main()
{
const int b;
cout<<"Constant number is "<< b;
return 0;
}
```

2.8. Receiving Input

The cin object is used to accept input from the standard input device i.e. keyboard. It is defined in the iostream header file. see the following example:

Example 6: Write a C++ program that calculates the area of a circle.

Circle Area = $r^2\pi$ where r is radius and π is 3.14

```
#include<iostream>
using namespace std;
int main()
{
const double pi = 3.14 ;
double r, area ;
cout<<"Enter the Circle Radius : " ;
cin>>r;
area = r*r*pi ;
cout<<"Area = " << area;
return 0;
}
```

2.9. Characters

A character set is a set of alphabets, letters and some special characters that are valid in C++ language.

2.9.1. Alphabets

C++ accepts both lowercase and uppercase alphabets as variables and functions.

Uppercase: A B C .... X Y Z

Lowercase: a b c .... x y z

2.9.2. Digits

0 1 2 3 4 5 6 7 8 9

2.9.3. Special Characters

,	<	>	.	_
(	)	;	$	:
%	[	]	#	?
'	&	{	}	"
^	!	*	/	|
-	\	~	+	

2.9.4. Using char Data Type

The char type is commonly used to represent ASCII characters. Such character constants are enclosed in single quotes and can be stored in a variable of char type.

```
char c = ‘A’;
```

To print the stored character, we write the following code:

```
cout<<"Character stored in c is "<< c;
```

To print the ASCII code of the stored character, we use + unary operator as in the following code:

```
cout<<"Character ASCII is "<< +c);
```

To read a char form the user:

```
char a;
cin>>a;
```

Example 7: Write a C++ Program that read a character from user at the runtime and display it ASCII code

```
#include<iostream>
using namespace std;
int main()
{
char c;
cout<<"Enter single character: ";
cin>>c;
cout<<"ASCII code of "<< c <<" is "<< +c;
return 0;
}
```

2.10. Literals

Literals are data used for representing fixed values. They can be used directly in the code. For example: 1, 2.5, 'c' etc.

2.10.1. Integer Literals

An integer is a numeric literal(associated with numbers) without any fractional or exponential part. There are three types of integer literals in C++ programming:

- decimal (base 10)
- octal (base 8)
- hexadecimal (base 16)

For example:

Decimal: 0, -9, 22 etc
Octal: 021, 077, 033 etc
Hexadecimal: 0x7f, 0x2a, 0x521 etc
In C++ programming, octal starts with a 0, and hexadecimal starts with a 0x.

2.10.2. Floating-Point Literals

floating-point literal is a numeric literal that has either a fractional form or an exponent form. For example:

-2.0
0.0000234
-0.22E-5
0.4E4

Note: E-5 = 10^{-5}

2.10.3. Character Literals

A character literal is created by enclosing a single character inside single quotation marks.
For example: 'a', 'm', 'F', '2', '}' etc.

2.10.4. String Literals

A string literal is a sequence of characters enclosed in double-quote marks. For example:

```
"good"
""
"      "
"x"
"Earth is round"
```

2.11. Comments

Comments are used to insert notes into the source code. They have no effect on the end program and are meant only to enhance the readability of the code, both for you and for other developers. A multiline comment is delimited by /* and */.

```
/* multi-line
comment */
```

The single-line comment, which starts with // and extends to the end of the line.

```
// single-line comment
```

Keep in mind that whitespace characters – such as comments, spaces, and tabs – are generally ignored by the compiler. This allows you a lot of freedom in how to format your code.

3. Operators (Part1)

A numerical operator is a symbol that makes the program perform a specific mathematical or logical manipulation. The numerical operators in C++ can be grouped into five types: arithmetic, assignment, comparison, logical, and bitwise operators.

3.1. Arithmetic Operators

There are four basic arithmetic operators, as well as the modulus operator (%), which is used to obtain the division remainder.

```
int x = 5 + 2;          // 7 - addition
x = 5 - 2;              // 3 - subtraction
x = 5 * 2;              // 10 - multiplication
x = 5 / 2;              // 2 - division
x = 5 % 2;              // 1 - modulus (division remainder)
```

Exercises

1. Write C++ program that calculates the circumference of the circle. Read the radius value from the user at the runtime as a floating-point number.

Where circle circumference = 2 r π and π = 3.14

2. Write C++ program that calculates the area and the circumference of a rectangle. Read the required input from the user at the runtime as an integer number.

Where area = width x height and circumference = 2 (width + height)

3. Write C++ program that calculates the area and the circumference of a square. Read the required input from the user at the runtime.

Where area = edge^2 and circumference = 4 edge

4. Write C++ program that calculates the volume and the surface area of a cylinder. Read the required input from the user at the runtime as a floating-point number.

Where volume = $r^2 \pi h$ and surface area = $2 r \pi h + 2 r^2 \pi$

Where r is radius, h is height and, $\pi = 3.14$

5. Write C++ program that converts the input Celsius degree to Fahrenheit degree. Read the input at the runtime.

   Where F = (C x 1.8) +32

6. Write C++ program that converts the input Fahrenheit degree to Celsius degree. Read the input at the runtime.

   Where C = (F - 32) / 1.8

7. Write C++ program that converts the input Kelvin degree to Celsius degree. Read the input at the runtime.

   Where K = C + 273

8. Write C++ program to find the quotient and reminder for dividing two integer numbers.