

Chapter 1
Review of C++ Programming



Basics of C++	
Character set	Fundamental unit of C++ language. Classified into letters, digits, special characters and white spaces.
Tokens	Building blocks of C++ programs. Classified into keywords, identifiers, literals, punctuators and operators.
Keywords	Reserved words that convey specific meaning to the language compiler.
Identifiers	User-defined words to identify memory locations, statements, etc. Identifiers include variables, labels, function names, etc.
Literals (Constants)	Tokens that do not change their value during the program run. Classified into integer constants, floating point constants, character constants and string constants.
Operators	Symbols that represents some operations. They consist of arithmetic, relational and logical operators. There are some special operators named <i>get from</i> (>>), <i>put to</i> (<<) and assignment (=). Another category of operators include increment (++), decrement (--) and arithmetic assignment (+=, -=, *=, /=, %=) operators.
Punctuators	Special characters like comma (,), semi colon (;), etc. used for the perfection of syntax of various constructs of the language.
Data types	These are means to identify the type of data and associated operations. Data types are classified into fundamental and user-defined data types. Fundamental data types include <code>int</code> , <code>char</code> , <code>float</code> , <code>double</code> and <code>void</code> .
Control statements	Two types – Selection statements (if, switch) Looping statements (while, for, do – while) while and for are entry controlled loops do – while is exit controlled loop
Looping statements	There will be four components – initialization expression, test expression, update expression, loop-body. In the case of entry-controlled loop, body will be executed only after evaluation the test expression (condition). But, in the case of the exit-controlled loop, condition will be checked only after executing the loop-body.

switch V/s if – else if statement

Switch	if – else if
<ul style="list-style-type: none"> • Only equality conditions are checked. • Program control goes outside the block only if break is used after each case. • default case is for an action where all the conditions fail. 	<ul style="list-style-type: none"> • Any relational expression can be used for conditions. • No need of break to take the control outside after executing a block. • else is used for an action where all the conditions fail.



Chapter 2 Arrays

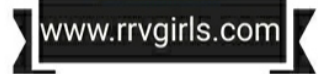
An **array** is a collection of elements of the same type placed in contiguous memory locations. Arrays are used to store a set of values of the same type under a single variable name. Each element in an array can be accessed using its position in the list called index number or subscript.

The syntax for declaring an array in C++ is as follows.

```
data_type array_name[size];
```

Eg: `int num[10];`

This statement declares an array named `num` that can store 10 integer numbers.



Accessing array elements

Elements of an array are easily accessed using a **for** loop. For example, the elements in the above array can be displayed using the following loop:

```
for (i=0; i<5; i++)
    cout<<score[i];
```

String handling using arrays

A character array can be used to store a string, since it is a sequence of characters. The array `char my_name[10];` can store a string of 9 characters. One location will be used to store '\0' (null character) as string terminator.

A string can be input using the statement:

```
cin >> my_name;
```

This statement can store a string without any white space (that is, only one word). If we want store strings containing white spaces (strings having more than one word) we can use `gets()` function, `getline()` function or `get()` function.

`cin.getline(my_name,10);` can accept a string containing white spaces. The functions `getline()` and `get()` are stream functions to input string and `gets()` is a console function.

Similarly to display string data we can use `puts()` function and `write()` function.



Chapter 3 Functions

Predefined functions

Type	Functions	Operation
(cstdio / stdio.h)	<code>getchar()</code>	To input a character
	<code>putchar()</code>	To display a character
Stream functions (iostream)	<code>get()</code>	To input a character To input a string of maximum 10 characters
	<code>getline()</code>	To input a string of maximum 10 characters
	<code>put()</code>	To display a character
	<code>write()</code>	To display a string of maximum 10 characters

Web Technology

Static web page V/s Dynamic web page

Static web page	Dynamic web page
The content and layout of a web page is fixed.	The content and layout may change during run time.
Static web pages never use databases.	Database is used to generate dynamic content through queries.

Client side scripting V/s Server side scripting

Client side scripting	Server side scripting
Script is copied to the client browser	Script remains in the web server
Script is executed in the client browser	Script is executed in the web server and the web page produced is returned to the client browser

Client side scripting languages: JavaScript, VB Script

Server side scripting languages: PHP, JSP, ASP, Pearl

Structure of HTML code

```
<HTML>
  <HEAD>
    <TITLE>    </TITLE>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```

HTML Tags

Tags	Use	Attributes	Values and Purpose
<HTML>	To start an HTML document		
<HEAD>	To specify the head section of an HTML document.		
<TITLE>	This tag pair contains the text to be displayed in the title bar of browser.		
<BODY>	Defines the body section of the web page.	Bgcolor	Colour for the background of a web page.
		Background	Image as the background of a web page.
		Text	Colour of the text in the web page.

Tags	Use	Attributes	Values and Purpose
<H1> <H6>	To provide different levels of headings.	Align	"left", "right" and "center" are the values.

	To break the current line of text and continues in the next line. No attributes.		
<P>	To create a paragraph leaving a blank line.		
<HR>	To draw a horizontal line across the width of the browser window		

