55 COMMON MISTAKES
You often do in C++ programming
(Who says programming is difficult?)

Authors:

Rashidah Mokhtar, Safura Adeela Sukiman, Azrina Suhaimi,
Mohd Lezam Lehat & Rohaizan Ramlan

email:
rohaizan@uthm.edu.my & pt@uthm.edu.my

Abstract:

This book aims to expose learners to common mistakes done while writing C++ programming language codes. This book covers the fundamentals of C++ programming language which relate to basic C++ concepts such as basic elements of C++, selection, looping, function and array and it is associated with structured programming.

This book is compiled based on the authors' experience who had taught C++ programming for more than 7 years. It is the common syntax errors that are often done by students. This is the reason why this book is produced in the first place. This book also includes tips on how to avoid the mistakes by showing the correct syntax.

The book is made up of 14 sections, each showing various errors and its correction. For each error there is a solution on the correct syntax. This book also employs simple English; this makes it one of the must-have guide book for beginners. Hopefully, this book will prove beneficial to all learners especially to beginner of C++ language.

Keywords: C++ Programming, Syntax Errors, Beginner Guide, Structured Programming & Coding Tips

FIRST EDITION

55 COMMON MISTAKES

You Often Do in C++ Programming

(Who says programming is difficult?)

Rashidah Mokhtar
Safura Adeela Sukiman
Azrina Suhaimi
Mohd Lezam Lehat
Rohaizan Ramlan

Penerbit UTHM

First Published 2022

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data

Rashidah Mokhtar
55 COMMON MISTAKES : You Often Do in C++ Programming
(WHO says programming is diffficult?) / Rashidah Mokhtar,
Safura Adeela Sukiman,
Azrina Suhaimi, Mohd Lezam Lehat, Rohaizan Ramlan. – FIRST EDITION.
ISBN 978-967-0061-42-9
1. C++ (Computer program language).
2. Computer programming.
3. Government publications--Malaysia.
I. Safura Adeela Sukiman. II. Azrina Suhaimi.
III. Mohd. Lezam Lehat. IV. Rohaizan Ramlan.
V. Title.
005.133

Published by:
Penerbit UTHM
Universiti Tun Hussein Onn Malaysia
86400 Parit Raja,
Batu Pahat, Johor
Tel: 07-453 7051
Fax: 07-453 6145

Website: http://penerbit.uthm.edu.my
E-mail: pt@uthm.edu.my
http://e-bookstore.uthm.edu.my

Penerbit UTHM is a member of
Majlis Penerbitan Ilmiah Malaysia
(MAPIM)

Contents

Preface

Most people say learning programming language is difficult; while others claim otherwise. We just need to know the appropriate learning technique and style that can help accelerate the learning process. Nowadays, there are many learning programming techniques; problem-based learning, game-based learning, pair programming, learning through example and learning through error. This book has adopted learning through error approach, where is suitable for beginners.

This book aims to expose learners to common mistakes done while writing C++ programming language codes. This book covers the fundamentals of C++ programming language which relate to basic C++ concepts such as basic elements of C++, selection, looping, function and array, and it is associated with structured programming.

This book is compiled based on the authors' experience who had taught C++ programming classes. It is consisting of the common syntax errors, semantic and logic errors that are often done by students. This book also includes tips on how to avoid the mistakes by showing the correct syntax.

The book is made up of 14 sections, each showing various errors and its correction. For each error there is a solution on the correct syntax. This book also employs simple English; this makes it one of the must-have guide book for beginners.

It is hope that this book will prove beneficial to all learners especially to beginners of C++ language.

Rashidah Mokhtar (PhD)
Safura Adeela Sukiman
Azrina Suhaimi
Mohd Lezam Lehat
Rohaizan Ramlan

Acknowledgement

Firstly, we would like to express our gratitude to the Almighty, who gives us the brilliant idea and strength to write this book from scratch until publication. Secondly, of course to all authors in this book who are really cooperative. Without a good teamwork and patience, we will not be able to complete this book successfully. So thank you to all authors who have contributed ideas, share experiences and give the best in writing this book. The authors would also like to express utmost appreciation to those who have contributed directly or indirectly in helping to complete this book. Last but not least, we would like to thank our families for their love, support and patience for the past eleven months in publishing "55 Common Mistakes You Often Do in C++ Programming".

VARIABLE AND IDENTIFIER CONCEPT

SECTION 1

This section would go through some of the most frequent errors made by using variables concept in the C++ programming language.

What is variable?

A variable is used to store information about data that is used to construct a computer program. It stores a value based on the data type specified during declaration.

How to create a variable?

In order to construct a valid variable, certain rules must be followed. The rules state that it must begin with:-

- a letter or underscores
- there must be no space between two words
- reserve words or special symbols cannot be used.

What is an identifier?

In contrast, an identifier is simply a name given to an entity in a program. For instance, a variable's name, a function's name, and so on. Identifiers are used to describe a certain entity in a program, as the name implies. Variables are only one kind of identifier; there are also array names, class names, structure names, and so on.

HOW TO USE VARIABLE IN A COMPUTER PROGRAM?

Before you can use a variable in a computer program, you must first declare it. The following is the general declaration syntax:

- General syntax:

  dataType variableName;

- Example:

```
int your_age;
float price;
char studName[100];
```

MATHEMATICAL EXPRESSION

SECTION 2

This section would go through some of the most frequent errors made by using mathematical expression in the C++ programming language.

What is mathematical expression?

A mathematical expression is a set of operands or variables with arithmetic operators that produces a mathematical result.

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus / Remainder

ORDER OF PRECEDENCE

highest

Operator	Description
()	Parentheses
++ , --	Suffix / postfix
+ , -	Unary plus minus
* , / , %	Multiplication, division, remainder
+ , -	Addition, subtraction,

lowest

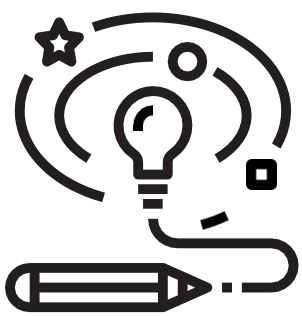
RESERVED WORDS

SECTION 3

This section would go through some of the most frequent errors made by using reserved words in the C++ programming language.

What are reserved words?

A reserved word also known as a keyword, is a word that has a specific meaning in programming. In a program, reserved words cannot be used as identifiers.



SYNTAX

The letters of a reserved word are usually all lowercase and has the look of a regular word such as **return, int , double, void, class, break , case, default, do, if , else, namespace, new, switch, true, false, this, throw, while** etc.

char VERSUS string

SECTION 4

This section would go through some of the most frequent errors made by using char and string data type in the C++ programming language.

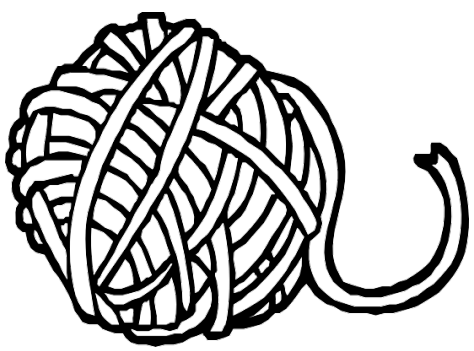
What is char?

A char is a datatype that represent a character in a form of letter, number, space, punctuation mark, or symbol that can be typed on a computer.

What is string?

A string is a data type used in programming, made up of a sequence of zero or more characters. The length of a string is the number of characters in it.

Every character in a string has a position in the string that is relative to it. The first character's position is 0, the second character's position is 1, and so on.



SYNTAX - char

Each character must be enclosed with a pair of single quotation marks when using char data type and a single character to be placed between the quotation marks.

```
Examples : 'A', 'a', '0',
'*', '+', '$', '&', ' '
```

SYNTAX - string

Strings in C++ are enclosed in double quotation marks.

```
Examples : " ", "c++
programming",      "123",
"total (students)", etc
```

SELECTION CONTROL STRUCTURE

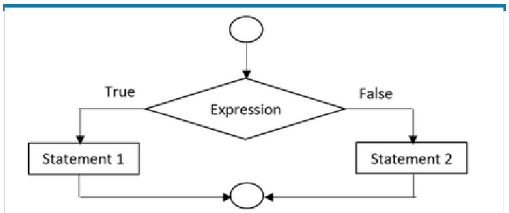
SECTION 5

This section would go through some of the most frequent errors made by using selection control structure in the C++ programming language.

What is selection control structure?

Selection control is a statement that performs different processes based on whether a Boolean condition is true or false. It provides alternatives to sequential program execution and are used to alter the sequential flow of execution.

Basic Flow



Relational Operators

Operator	Description
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>==	Greater than or equal to

Logical Operators

Operator	Description
!	not
&&	and
||	or

SYNTAX

a) One-way selection:

```
if (expression)
     statements;
```

b) Two-way selection:

```
if (expression)
     statement1;
else
     statement2;
```

c) Multiple selections:

```
if (expression)
     statement1;
else if (expression2)
     statement2;
.
.
else
     statement3;
```

COMPOUND BLOCK

SECTION 6

This section would go through some of the most frequent errors linked to compound blocks in the C++ programming language.

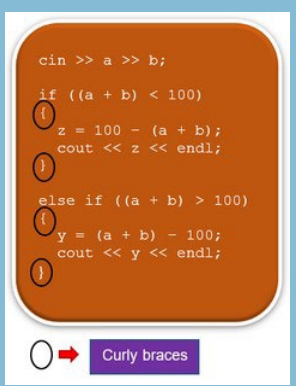
What is a compound block?

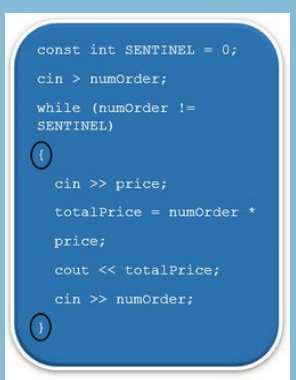
A compound block can be described as a group of statements consisting of more than one line of codes that is treated by the compiler as if it were a single statement. By definition, some books refer to this term as a compound statement. To implement this compound block, the group of statements must be attached with curly braces ({…}). No semicolon (;) is required after placing the curly braces.

When to use a compound block?

Compound blocks can be executed anywhere in the program, but they are typically very much related to selection control structures such as `if...else` and `switch...case`, as well as repetition control structures (`while`, `do…while` and `for` loops). In the presence of a compound block, all statements belonging to a particular block are displayed when the condition is evaluated to be True (1).

HOW TO USE COMPOUND BLOCK(S) IN A COMPUTER PROGRAM?





switch STATEMENT

SECTION 7

This section would go through some of the most frequent errors made by using switch statements in the C++ programming language.

What is switch Statement?

A switch statement is an alternate to an if-else statement. Nevertheless, the switch expression is limited to integral expression. An integral expression produces the integer value as an output. This expression normally is used with data stored in int, char and bool types.

switch Syntax

```
switch (expression)
{
case constant_1:
      statements
      break;
case constant_2:
      statements
      break;
.
.
.
case constant_n:
      statements_n
      break;
default:
      statements
}
```

- An expression is an integral expression that is going to be compared with the case constants. Examples:
  - switch (option) → This expression contains variable option that may be declared as an int type.
  - switch (gender) → This expression contains variable gender that may be declared as char type.
  - switch (number % 2 == 1) → This expression evaluates to true or false. True or false are bool values.
- Constants are the specific values that may not be altered during the program execution. Examples:
  - case 1: → The value 1 is an int value.
  - case 'F': → The value 'F' is a char value.
  - case true: → The value true is a bool value.
- When the expression and the case constant are matched, the statements following the case execute until it reaches the break statement. The statements may consist of one or more statements that is separated by a semicolon (;). However, there is no need to make them a compound statement.
- After the break statement is executed, the program control goes to the line following the switch structure.
- The default statements execute when the expression does not match any of the constants.
- break and default are optional. They may be omitted and the results are different.

switch EXAMPLE

```
int main()
{
    char gender;
    cout << "Enter M for Male Gender and F for Female Gender : ";
    cin >> gender;

    switch (gender)
    {
    case 'm':
    case 'M':
        cout << "Your Malaysian Identity Card should end with an odd number.";
        break;
    case 'f':
    case 'F':
        cout << "Your Malaysian Identity Card should end with an even number.";
        break;
    default:
        cout << "There is no gender other than Male and Female." << endl;
    }
    return 0;
}
```

Sample Input/Output 1

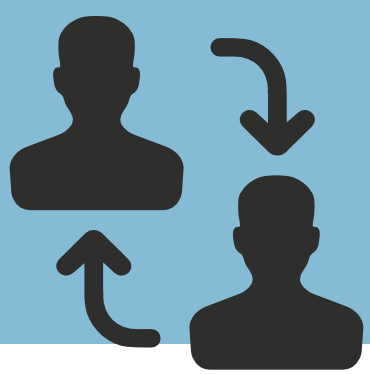
```
Enter M for Male Gender and F for Female Gender : M
Your Malaysian Identity Card should end with an odd
number.
```

Sample Input/Output 2

```
Enter M for Male Gender and F for Female Gender : F
Your Malaysian Identity Card should end with an even
number.
```

Sample Input/Output 3

```
Enter M for Male Gender and F for Female Gender : a
There is no gender other than Male and Female.
```



REPETITION CONTROL STRUCTURE

SECTION 8

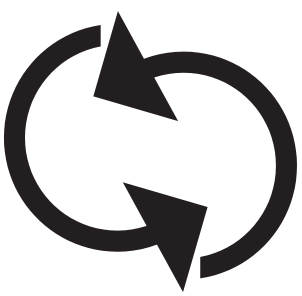
This section would go through some of the most frequent errors associated with repetition control structures in the C++ programming language.

What is a repetition control structure?

Repetition or iteration (or looping) is one of the useful features in C++. Most of the time, programmers want to execute certain blocks of code more than once, and with repetition, it is possible for this situation to occur. Repetition allows a group of codes to be executed repeatedly, until the condition is evaluated to become False (0). Infinite repetition, where the repetition will not stop is also possible. However, infinity repetition is not recommended because it illustrates unsatisfactory level of coding ethics.

What are the types of repetition control structures?

Controlled Loop	Counter	Sentinel	Flag
while • *The best choice when number of iterations cannot be determined in advance, and it could be zero.*	✓	✓	✓
for • *The best choice when number of iterations can be determined in advance.*	✓		
do…while • *The best choice when number of iterations can not be determined in advance, but iterate at least once.* • *E.g. Offer a repeating menu to the user.*	✓	✓	✓



HOW TO USE REPETITION CONTROL STRUCTURE(S) IN A COMPUTER PROGRAM?

Let us take a few scenarios when a programmer writes a computer program:

·Scenario 1 – if the programmer knows exactly how many times a block of code needs to be repeated, then the counter-controlled `while`, `for` and `do…while` looping will fit this scenario.

·Scenario 2 – on the other hand, if the programmer has absolutely no idea how many times a block of code is to be run repeatedly, then sentinel-controlled `while` looping is the right answer.

·Scenario 3 – whenever the programmer wants to use a `bool` variable, with the value of True or False to control repetition, then the flag-controlled `while` looping is the most accurate to execute.

·Scenario 4 – if the programmer wants the repetition to occur at least once, `do…while` looping is the best since the loop condition is evaluated after the body of the loop is executed. This type of looping is also known as posttest looping.

INPUT AND OUTPUT STREAM

SECTION 9

This section would go through some of the most frequent errors relevant to misuse of input and output streams.

What is an input and output stream?

Input occurs when the flow direction of the byte is from the input device i.e., keyboard to the main memory. Meanwhile, the output is the opposite. Output occurs when the direction of the byte flow is from the main memory to the output device i.e., monitor. Therefore, because writing code involves obtaining input from the user using a keyboard, and displaying the output of the program on the screen, C++ statements require special operators to indicate the flow of bytes.

The input stream is indicated by the insertion operator `<<`, while the extraction operator `>>` is an indication for the output stream. To successfully run a C++ program with input and output streams, the `<iostream>` header must be written above the program.

The insertion operator is usually paired with the `cin` command, while the `cout` is paired with the `>>` extraction operator. Programmers are allowed to get more than one input at a time by writing code using the following syntax:

```
cin >> variable1 >> variable2 >> variable3 >> ... ;
```

Programmers are also allowed to display more than one output at a time using the following program syntax:

```
cout << value1 << value2 << value3 << ... ;
```

However, programmers are not allowed to combine both insertion and extraction operators on one line as below:

```
cin >> variable1 >> variable2 << value2 << value2 ;
```

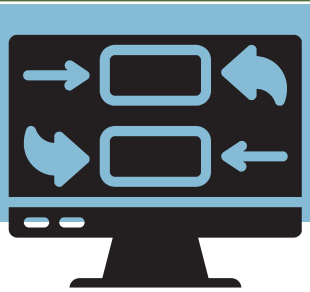
HOW TO CORRECTLY USE INPUT AND OUTPUT STREAMS IN A COMPUTER PROGRAM?

```
cout << "Insert your first and last name: ";
cin >> firstName >> lastName;
```

- You are also allowed to get inputs one by one as below:

```
cout << "Please insert your first name :" ;
cin >> firstName ;

cout << "\nPlease insert your last name :" ;
cin >> lastName ;
```



FUNCTION

SECTION 10

This section would go through some of the most frequent errors made by using function in the C++ programming language.

What is a function in C++?

A C++ function is a block of C++ statements to perform a specific task. Generally, there are two types of function:

1. Predefined Function
- There are various operations of predefined functions organized in the C++ Standard Library.
- To use this function, you must know the operation to be executed, the header file of the function, the name of the function, the number of parameters, the data type of each parameter, and whether that function returns a value or not.

2. User-defined Function
- Functions are defined by programmers.
- To define the function, programmers have to determine whether the function returns the value or not. A function that returns a value is called a value-returning function. A function that does not return a value is called a void-function.

Value-Returning vs. void Function

	Value-Returning Function	void Function
	• Function returns a single value when it is called.	• Function does not return a value when it is called.
Function Definition	• The place where the code of the specific task is written.	
	• Function header should start with the data type of the returned value. E.g. `int`, `double`, `char`, `bool`. • It has to use the keyword `return` to return the value to where it is called.	• Function header should start with keyword `void`. • No keyword `return` but may use the keyword to exit early from the function.
Function Call	• The code that calls the function to be executed.	
	• Normally, it is written in an output statement or assignment statement.	• It is written in a stand-alone statement.
Function Prototype	• It is not compulsory if the function definition appears before the function call. • It is actually a function header that terminates with a semicolon (;). • It is to declare the function to the compiler.	

HOW GENERALLY USER-DEFINED FUNCTIONS ARE WRITTEN IN A PROGRAM?

If you want to write a complicated program with various tasks to perform, C++ allows you to divide the program into a sub program. Each sub program represents a specific task. In C++, this sub program is written in a function and technically, the specific task is written in a function definition. To perform the task, you need to write the function call at the line it is needed to be executed. The function call can be written either in the main() function or in another function definition.

```
//Function Prototype of Task 1
//Function Prototype of Task 2
//Function Prototype of Task N

int main()
{
   //variable declarations
   //executable codes
   .
   //Function Call of Task 1
   //Function Call of Task 2
   //Function Call of Task N
   .
   .
   return 0;
}

//Function Definition of Task 1
//Function Definition of Task 2
//Function Definition of Task N
```

OPERATOR '=' VERSUS '=='

SECTION 11

This section would go through some of the most frequent errors made by using assignment operator and equality operator in the C++ programming language.

What is an assignment operator?

An assignment operator (=) is used to assign values to the variable. It is used in an assignment statement. The assignment statement has the syntax as shown below:
variable = expression;

The expression can be a constant, variable or mathematical expression.

Examples:

```
int num1, num2, num3;
num2 = 4 * 3;
num1 = num2;
num3 = 99;
```

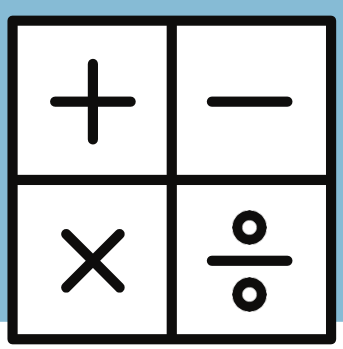
What is an equality operator?

An equality operator (==) is a relational operator. It is a binary operator that requires two operands to be compared. If both operands are equal, the expression returns true and otherwise, it returns false.

Examples:

```
char myChar, yourChar; myChar = 'a';
yourChar = 'b';
cout << (myChar == 'b'); //display 0
cout << (yourChar == 'b'); //display 1
```

ASSIGNMENT AND EQUALITY OPERATORS COULD NOT BE USED INTERCHANGEABLY AS IT WOULD CAUSE LOGIC ERRORS TO OCCUR.



MISUSE OF SEMICOLON (;)

SECTION 12

This section would go through some of the most frequent errors related to misapplication of semicolon in the C++ programming language.

What is a semicolon (;)?

A semicolon is a special symbol in C++ programming language. This symbol is taken from the English's punctuation marks and it is needed to end C++ statements. Examples of statements that need semicolon are variable declarations, input statements, output statements, and even formulas.

WHEN NOT TO USE A SEMICOLON (;)?

UNNECESSARY SEMICOLON	

After curly braces	`if (x <= 9)`
	`{`
	`  result = getResult (x);`
	`  cout << result << endl;`
	`};` Unnecessary Semicolon

`if` statements	`if (x <= 9);` Unnecessary Semicolon
	`{`
	`  result = getResult (x);`
	`  cout << result << endl;`
	`}`

`for` statements	`cin >> input;`
	`for (int i = 0; i < input; i = i + 2);` Unnecessary Semicolon
	`  cout < i < "\t";`

`while` statements	`int a = 15;`
	`while (a < 50);` Unnecessary Semicolon
	`{`
	`  cout << a << "\t";`
	`  a = a + 3;`
	`}`

`else` statements	`if (score <= 50.0)`
	`  cout << "Congratulations!!";`
	`else;` Unnecessary Semicolon
	`  cout << "Shame on you!!";`

CONDITION STATEMENTS

SECTION 13

This section would go through some of the most frequent errors connected to expression construction in selection control structures.

What is a condition statement?

In most cases, novice programmers will discover the syntax of selection control structures. Since this structure consists of one-way selection, two-way selection and nested if…else structures, then the syntax can be found to have differences. But, one thing in common is the (expression) after the if keyword. The following depicts it in more detail:

```
if (expression)
    statement
```

Syntax for one-way selection

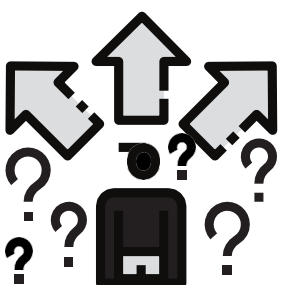
```
if (expression)
    statement1
else
    statement2
```

Syntax for two-way selection

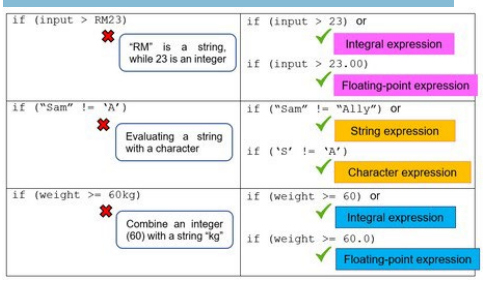
```
if (expression1)
    statement1
else
    if (expression2)
        statement2
else
    if (expression3)
        statement3
else
    if (nth expression)
        nth statement
```

Syntax for nested if…else structure

- An expression produces a value to be evaluated either True or False
- An expression can be in a form of integral, floating-point, character or string in selection control structures
- Programmers are NOT ALLOWED to combine different data types in an expression



HOW TO APPLY CONDITION STATEMENT(S) IN A COMPUTER PROGRAM?



SINGLE DIMENSIONAL ARRAY

SECTION 14

This section would go through some of the most frequent errors made by using one dimensional array in the C++ programming language.

What is One Dimensional Array?

A simple variable store only one value at a time. Let's say you want to store values of temperatures for seven days. If you are not familiar with the arrays, you may store the temperatures in seven declared variables of the same data type. This is not the practical way to do it because operation like finding the highest temperature is quite troublesome to be performed if the temperatures are not stored in an array. The array allows you to store more than one value of the same type in one declared variable. If a single dimensional array is declared, it stores the values in a list form. This means the values stored in the memories are adjacent to each other.

How to declare One Dimensional Array?

General syntax:

```
data_type array_name[array_size];
```

data_type	E.g. int, char, bool, float, double
array_name	An array name is actually an identifier. Thus, it should follow the rule of identifier to name the array.
array_size	An array size should evaluate to a positive integer. It shows the number of values stored in the array.

This example shows how to store values of temperatures for seven days in a variable named, `temperature`.

```
double temperature[7];
```

temperature [0]	temperature [1]	temperature [2]	temperature [3]	temperature [4]	temperature [5]	temperature [6]

The above declaration instructs the computer to provide memories to store the values of temperature. Each memory has the index number to show the position of the values in the array. The index number is a continuing number. It starts at 0 and ends at `array_size - 1`.

However, the concept of a single dimensional array is slightly different if you want to store a sequence of characters in the array. This is because you need to consider a null terminated value (`'\0'`) to be appended at the end of the characters. For example, if you want to store the value "Covid19", the size of the array is 8 because of the null terminated value.

```
char myChar[8] = "Covid19";
```

myChar [0]	myChar [1]	myChar [2]	myChar [3]	myChar [4]	myChar [5]	myChar [6]	myChar [7]
'C'	'o'	'v'	'i'	'd'	'1'	'9'	'\0'

HOW TO ACCESS ARRAY COMPONENTS?

General syntax:

```
array_name[index]
```

index	The index should be in the boundary of index >=0 and index <= array_size - 1

Examples:

To assign values to the specific index:

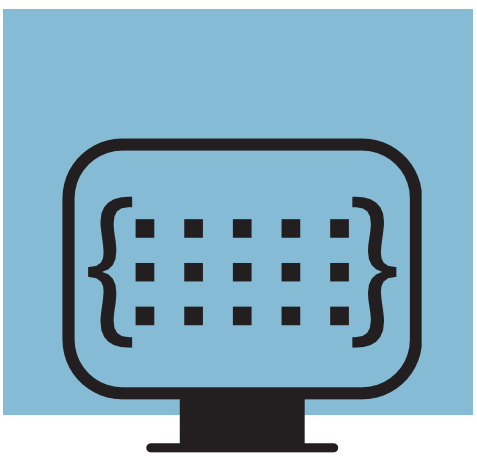
```
temperature[3] = 39.5;
```

To get input:

```
cin >> temperature[0]; //get input for the specific index
for (int i=0; i<7; i++)
   cin >> temperature[i]; //get input for all the indices
```

To display output:

```
cout << temperature[4]; //specific index
for (int i=0; i<7; i++)
   cout << temperature[i]; //display output for all the indices
```



Index

Author's Biography

Rashidah Mokhtar is a senior lecturer at Universiti Teknologi MARA Johor Branch Segamat Campus. She had more than 14 years of experience teaching programming subject using C++ language, web development application using HTML, JavaScript, PhP and ASP. She was graduated in Computer Science and got her Master of Science in Information Technology (Management) from Universiti Teknologi Malaysia (UTM). Rashidah also had Professional Technologist certification carried Ts title. Besides that, she also accomplished her degree for Doctor of Philosophy in Information Systems. Rashidah's vision is to provide an excellent educational solution, provide a real and beneficial values to the learners.

Safura Adeela Sukiman is also a senior lecturer at Universiti Teknologi MARA Johor Branch Segamat Campus with more than 12 years of teaching experience in programming and Information Systems. She also has a few years of working experience in industry as an associate researcher for various research development and maintenance projects. She obtained a degree majoring in Software Engineering from Multimedia University and a master in IT Management from Universiti Teknologi Malaysia (UTM). With her interests in the field of e-education, she continues to contribute towards the delivery of more innovative and effective learning.

Azrina Suhaimi has been teaching in UiTM Johor for more than 10 years. Among the subjects she teaches are Programming Languages C++ and JAVA, Data Structures and Fundamentals of Computer Problem Solving. She holds a master's degree and a bachelor's degree in Computer Science from Universiti Teknologi Malaysia (UTM). Azrina is very passionate about programming. Using her expertise in programming, she has contributed to finding system solutions in several projects such as Malaysian tax calculation.

Mohd Lezam Lehat holds a master's degree in IT Management and a bachelor's degree in Computer from Universiti Teknologi Malaysia (UTM). He has been actively involved in blended learning environments and has been in education world for more than 10 years. He has been a lecturer at Unisel before and thus, he has extensive knowledge particularly in multimedia programming, computer architecture and system development. Now, he is with UiTM Cawangan Johor under Faculty of Computer and Mathematical Sciences.

Rohaizan Ramlan is a Senior Lecturer at Universiti Tun Hussein Onn Malaysia (UTHM) with 19 years of teaching experiences. She holds a master's degree in IT Management (Manufacturing) and a bachelor's degree in Computer from Universiti Teknologi Malaysia (UTM).

Bibliography

D.S. Malik, C++ Programming: From Problem Analysis to Program Design, Course Technology, Eighth Edition, CENGAGE Learning, 2017.

Liang, Y. D., Introduction to Programming with C++, 3rd Edition, Pearson Higher Education, 2013.