Lesson Outcomes

At the end of this chapter, student should be able to:

- ✓ Understand basic program components
- ✓ Define and use the identifier, variable, constant and statement
- ✓ Understand the standard data type (**`int`**, **`float`**, **`double`**, **`char`**, **`char[ ]`**, **`const`**)
- ✓ Use the input and output statement, formatted output
- ✓ Understand the use of mathematical predefined function (`sqrt()`, `abs()`, `pow()` etc.)
- ✓ Use the predefined function of `cin.getline()`, `gets()`, `strcpy()`
- ✓ Use the arithmetic operator (`+`, `-`, `*`, `/`, `%`)
- ✓ Understand the assignment statement (=)
- ✓ Write simple programs

i. Basic Program Components

A simple C++ program has the following form:

```
//line comments
/*block
  comments
*/

#include <iostream.h>      //preprocessor directive

int main()    //main function
{      //body of program
       variable declaration section;
       statements;
return 0;
}
```

Example: A program that converts miles into kilometers

The following shows the problem definition, algorithm design and the algorithm implementation to carry out the task of converting miles into kilometers.

Problem definition

Output: kilo

Input: miles

Formula: kilo = miles * 1.609

Algorithm design

```
1. Get input
       Read miles
2. Calculate kilo
       kilo = miles * 1.609
3. Display output
       Print "Miles in kilometer is " kilo
```

Algorithm implementation

```
/* Date: 12 Jun 2006
Comments: My first program


This program converts miles into kilometer
*/

#include <iostream.h> /*preprocessor
                        directive*/

int main() //function body
{
       //variable declarations
       float miles;
       float km = 0.0;

       //input prompt
       cout << "Please enter miles: ";
       cin >> miles;

       //calculate kilo
       km = miles * 1.609;

       //display output
       cout << "\n Miles in kilometers is "
              << km << endl;
return 0;
}
```

Explanation:

i. *Lines 1-6*

- These are introductory comments. The comments are not executed by the compiler and they are used only to allow the readability of the program.

ii. *Line 8*

- The `#include` directive is a special instruction for the C++ compiler that tells the compiler to include the contents of another file, in this case the `iostream` file. Files that are included in the programs are called **include files** or **header** files.
- In the C++, the `iostream` file contains the instruction (source code) needed to handle input and output operation, such as inputting data form keyboard and outputting information on the computer screen. C++ programs typically include at least one directive, and most include many directives.

iii. *Line 10*

- The word `main`, which must be typed in lowercase letters, is the name of function. A function is simply a block of code that performs a task. The entire line of code, `void main()` is referred to as function header, because it marks the beginning of a function.

iv. *Line 11*

- The beginning brace `{` marks the beginning of the code block that comprises the function. Everything between the opening and closing set of braces (`{ }`) belongs to this `main` function and is referred to as the function body.

v. *Line 12, 16, 20, 23*

- These are the comments that tell the purpose of the program's section.

vi. *Line 13, 14*

- These instructions declare or reserve two variables, which are simply memory locations that the program uses while it is running. The keyword `float` tells the compiler that the variable or the memory location can store a number with a decimal place.
- The instruction to declare a variable is considered a statement in C++, therefore it ends with a semicolon(`;`).

vii. Line 17

- The `cout` object and the insertion operation (<<) indicate that the message in the enclosed quotation marks (" ") will be sent to the console (screen).

viii. Line 18

- The `cin` object and the extraction operator (>>) will accepts a value from the keyboard and store the value into the memory location `miles`.

ix. Line 21

- An instruction to multiply `miles` by 1.609 and stores the result into the memory location `kilo`.

x. Line 24-25

- Display the message "`Miles in kilometers is `" and the value stored in the variable `kilo` to the screen. This statement end with '`;`' can be written in two lines.

xi. Line 26

- **`return`** `0` – this keyword indicate that program terminate in normal condition. **`return`** `0` must be written if the `main()` function has data type integer (**`int`**).

xii. Line 27

- Closing brace `}` marks the end of the program body.

Most program like the one shown above include the following components:

- Identifiers: variables and constant
- Operators

ii. **Identifier**

- Identifier is simply references to memory location which can hold data and used to represent variables, constants and name of functions (sub-modules) in computer programs.
- Rules for naming identifiers are:
  a. ***The 1st letter must be alphabet or underscore.***

Example of valid identifiers:	Example of invalid identifiers
`form1   _2nd_class   side_3`	`3number   8tvi   1_to_0`

  b. ***Can be a combination of alphabet letter, digit and underscore.***

Example of valid identifiers:	Example of invalid identifiers
`MilesToKilo kilo_to_miles`	`total$   km/hour   comm%`

c. ***Cannot used C++ reserved word, for example:***

Table 1: Some of the C++ reserved word

int	**char**	**long**
float	**double**	**default**
case	**for**	**void**
break	**const**	**else**
if	**private**	**do**
else	**goto**	**switch**
while	**public**	**class**

d. ***No blanks or white space characters.***

Example of valid identifiers:	Example of invalid identifiers
`form1 _2nd_class side_3`	`number two first number`

e. ***Case sensitive (the lowercase and uppercase letters are treated as different characters).***

Example: `firstNumber`, `firstnumber`, `FIRSTNUMBER`, `FirstNumber` and `Firstnumber` are treated as different identifiers.

1. *Variable*

- Variable is an identifiers that refers to memory location, which can hold values. Variable can only store one value at one time. Thus the content of variable may change during the program execution. Before it is used, it must be declared with its data type.
- Syntax: `data_type variable_name`
- Example: `int sum, float price, char name`
- Once a variable is declared, it contains none useful value (garbage). To make it useful, the variable must be given a value, either by the assignment statement or input from the user, before it is used. The value serves as the initial value to the variable.
- Example of variable initialization

```
num=10;
```

set the initial value of 10 into variable `num` using assignment operator (=)

```
cin >> number;
```

set the initial value into the variable `number` using input statement called extraction (>>)

- Single declaration:

  Example:
  ```
  int num1;
  int num2;
  int sum;
  ```

- Multiple declarations – variables having the same data type can be grouped and declared using the single declaration statement. For example, single declaration as shown above can be declared as a single statement.

  Example: `int num1, num2, sum;`

  Example of initialization: `int num1 = 10, num2 = 20, sum = 0;`

- The following example gives all the variables the value 0,

  Example: `score1 = score2 = score3 = 0;`

2. Constant

- An Identifier whose value does not change throughout program execution.
- Allow you to give name to a value that is used several times in a program.
- Usually constant name is capital to distinguish from other variables.
- 2 ways of declaring constant:
  a. Using constant keywords: `const float PI=3.142;`
  b. Using preprocessor directives: `#define PI 3.142`

Example:

Using constant keywords

```
#include <iostream.h>
#include <math.h>

int main()
{
      const float PI = 3.142;
      float radius = 7;
      float area = 0;
      area = PI * pow(radius,2);
      cout << "Area = " << area;
return 0;
}
```

Using preprocessor directives

```
#include <iostream.h>
#include <math.h>
#define  PI  3.142

int main()
{
      float radius = 7;
      float area = 0;
      area = PI * pow(radius,2);
      cout << "Area = " << area;
return 0;
}
```

iii. ***Escape sequences***

- The combination of backslash (\) and specific characters are called *escape* sequences. Escape sequences change the meaning of the character that follows it.
- Those escape sequences must be within quotation marks (" ").
- Table 2 illustrates some escape sequences.

Table 2: Escape sequences

Code	Meaning
`\a`	Audio bell
`\b`	Backspace
`\f`	Form feed
`\n`	Newline
`\r`	Carriage return
`\t`	Horizontal tab
`\\`	Backslash character
`\'`	Single quote character
`\"`	Double quote character
`\0`	NULL ASCII 0

iv. ***Basic data types***

Data type	C++ keyword	Syntax	Bit Width	Example value	Range
integer	**int**	**int** no1, no2;	32	-1, 34000	-32768 to 32767
long integer	**long**	**long** amount;	64	10000000	-4294967296 to 4294967296
short integer	**short**	**short** totalweight;	16	127	-128 to 127
unsigned integer	**unsigned**	**unsigned** sum;	32	200	0 to 65535
character	**char**	**char** name; **char** name[n]; //n=length	8	'r' "roslan"	0 to 255
floating point	**float**	**float** amount;	32	20.00	Approximately 6 digits of precision
double floating point	**double**	**double** totalweight;	64	100.85321	Approximately 12 digits of precision
Boolean	**bool**	**bool** found	N/A	true/false	true/false
void	**void**	**void** main()	N/A	N/A	valueless

v. ***Arithmetic***

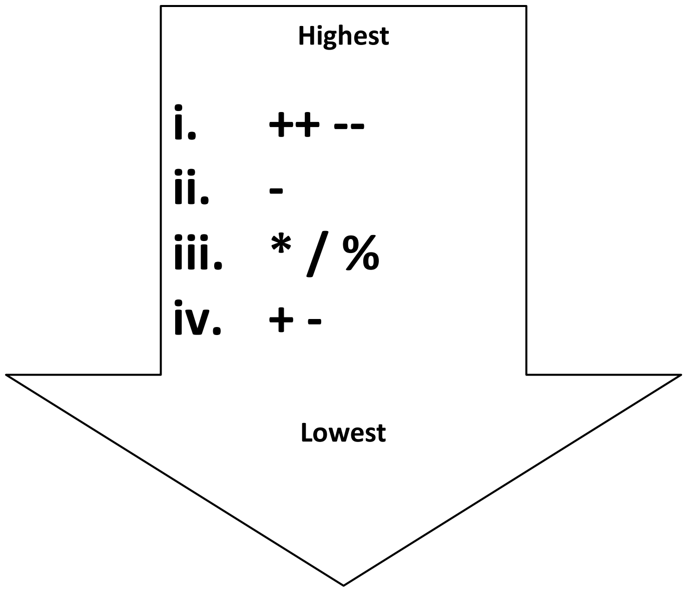
1. *Arithmetic operators*

- The seven arithmetical operations supported by the C++:

i.	**+**	addition
ii.	**-**	subtraction
iii.	*****	multiplication
iv.	**/**	division
v.	**%**	modulus division
vi.	**--**	decrement
vii.	**++**	increment

2. *Arithmetic precedence*

- Operators on the same precedence level are evaluated by the compiler from left to right.
- Of course parentheses may be used to alter the order of evaluation. Parentheses are treated by C++ in the same way that they are by virtually all other computer languages: They are force an operation, or set of operation to have a higher precedence level.



References:
Malik, D.S., (2010). *C++ Programming, From Problem Analysis to Program Design*. Course Technology, Canada.
Gary, R.B., (2010). *C++ For Engineers & Scientists: 3rd Edition.* Course Technology, Canada.

3. Arithmetic expression

- Formally, arithmetic expression is constructed by using arithmetic operators and numbers.
- The numbers appearing in the expression are called operands.
- Operators that have only one operand are called **unary operators**.
- Operators that have two operands are called **binary operators**.
- Example:

```
-5
8 - 7
3 - 4
2 + 3 * 5
5.6 + 6.2 * 3
x + 2 * 5 + / y
```

4. Example of Arithmetic operator and arithmetic expression

Operator	Action	Expression	Value (a=7, b=2)
`-`	subtraction	`a - b`	5
`+`	addition	`a + b`	9
`*`	multiplication	`a * b`	14
`/`	division	`a / b`	3 (remainder is discarded)
`%`	modulus division	`a % b`	1 (produces remainder)
`--`	decrement by 1	`a--`	6 (subtracts 1 from variable a)
`++`	increment by 1	`a++`	8 (adds 1 to variable a)

vi. *Assignment*

1. Assignment Operator

- C++ has several assignment operators. The most commonly used assignment operator is =.
- Assignment operations using = has the general form:

```
identifier = expression
```

- Where identifier generally represent variable, and expression represent a constant, a variable or a more complex expression.

2. Assignment Statement

An assignment statement is used to place a value into another variable coded directly within the program. In other words, the variable gets the value initially assigned to it in the program (hard-coded) and not from the value keyed in by the user.

- Example:

```
quantity = 20;
price = 5.50;
amount =  basic_pay + overtime_hours * overtime_rate;
deductions = socso + insurance_premium + car_loan;
net_pay = gross_pay -  deduction;
current_counter = current_counter ++;
```

- Another example of writing statement:

`counter++` or also can be written as `counter = counter + 1`

`sum = sum + num` or also can be written as `sum += num`

More examples:

Task	Usage

To assign a value for a numeric data type	**`int`** `number;`
	`number = 50;`

To assign a value for char data type	**`char`** `kod;`
	`kod = 'X';`

To assign a value to array of characters	**`char`** `title[19];`
	`strcpy(title, "Programming is fun!");`

To assign a value from an expression	`total = no1 + no2 + no3;`
	`total = 2 + 4 + 6;`

- ***Some introduction to Mathematical predefine functions***

  Note: Header file used is `math.h` *(include* `<math.h>`*)*

Some of the most commonly used functions are given below.

Name	**Description**
`pow(x, y)`	Return x raised to the power of y
`sqrt(x)`	Square root of x
`abs(x)`	Absolute value |x|

References:
Malik, D.S., (2010). *C++ Programming, From Problem Analysis to Program Design*. Course Technology, Canada.
Gary, R.B., (2010). *C++ For Engineers & Scientists: 3rd Edition.* Course Technology, Canada.

Example

The following program accepts a number, gets the square root of the number and multiplies the number by using the `sqrt()` and `pow()`.

```
#include <iostream.h>
#include <math.h>

int main()
{
      double number;
      cout << "Enter a number: ";
      cin >> number;
      cout << "The square root of " << number << " is " << sqrt(number) <<
            endl;
      cout << number << " raised to the power of 2 is " << pow(number,2) <<
            endl;
return 0;
}
```

Output

```
Enter a number: 9
The square root of 9 is 3
9 raised to the power of 2 is 81
```

vii. ***Input/output statement***

- Input/output operation is fundamental to any programming language.
- In C++ `cin` and `cout` are used to input data and to output the data.

  `cin >>` is used to get data from keyboard

  `cout <<` is used to send output to the screen.
- Preprocessor directive `#include <iostream.h>` must be used in order to handle the input (`cin>>`) and output (`cout <<`) statement.
- The other type of input is `gets(variableName)`, and `cin.getline(variableName, length)`. The `gets(variableName)` should be handle by preprocessor directive `#include <stdio.h>`. While `cin.getline(variableName, length)` should be handle by preprocessor directive `#include <string.h>`.

Example of input statement:

Task	Usage / Sample Input Data
To input numeric data type	`int num1;` `float num2;` `cin >> num1 >> num2;      //input: 10 5.5`
To input 2 character data types	`char letter, symbol;` `cin >> letter >> symbol;  //input: A #`
To input sequence of characters (string)	`char customerName[80];` `gets(customerName);`
	`char customerName[80];` `cin.getline(customerName, 80);`

Example of output statement:

Task	Usage / Sample Output Data
To display labels `Enter 2 number:`	`cout << "Enter 2 numbers: ";`
To display results from an arithmetic expression.	`cout << 15 * 2;` `cout << sum /count;`
To display labels `Total is:` and a value stored in variable total. Complete output is `Total is:10`	`int total = 10;` `cout << "Total is: " << total;`
To display formatted output and predefined symbol, ***end***	`cout << "\n\n";` `cout << endl;`

- There are three ways of getting input data
    1. Get input data from keyboard

Example:

```
#include <iostream.h>

int main()
{     float salary = 0.00;    //initialize value of salary to 0.00
      float deduction = 0.00; //initialize value of deduction to 0.00

      cout << "Enter Your Salary ";
      cin >> salary;          //get input from keyboard

      deduction =  0.08 * salary;   //calculate deduction

      cout << "Your Salary is " << salary << "And Your EPF Deduction is "
<< deduction;

return 0;
}
```

2. Get input data from fix data (assignment or initialization)

Example:

```
#include <iostream.h>

int main()
{     const float salary = 1000.00;
                        //initialize value of salary to 1000.00
      const float deduction = 0.00 ;
                        //initialize value of deduction to 0.00

      deduction =  0.08 * salary;   //calculate deduction

      cout << "Your Salary is " << salary << "And Your EPF Deduction is " <<
                  deduction;

return 0;
}
```

3. Get input data from external file

   Used `#include <fstream.h>` header for file input and output:

   a. `ifstream` to read data from input file:

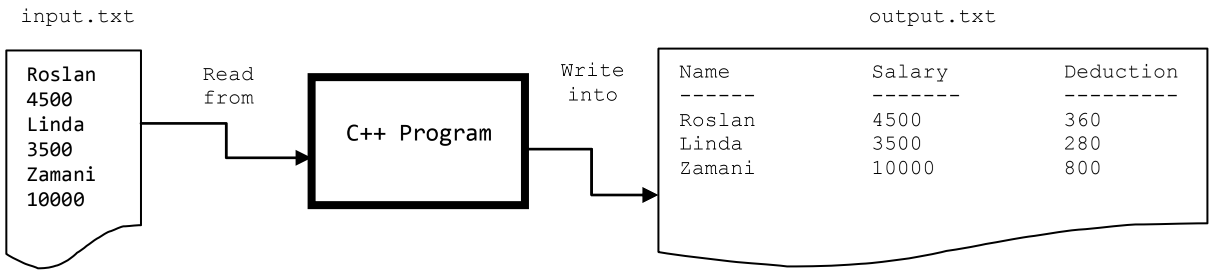
      i.e. `object_name("input_file_name")`

   b. `ofstream` to write data into output file:

      i.e. `object_name("output_file_name")`

Example:

```
#include <fstream.h>     //header file for file input and output
#define N 3              // define constant N with value 2

int main()
{
      //declare variables
      char name[20];
      float salary = 0.00, deduction = 0.00;

      //declare constant
      const float TAX = 0.08;

      ifstream read("input.txt");   // read from the file input.txt
      ofstream write("output.txt"); //to send write to output.txt

      //write output to output.txt
      display << "\n\nName \t\t Salary \t Deduction ";
      display << "\n------ \t\t -------\t --------- ";
      for(int i=0; i<N; i++)
      {
            //read 2 data items from input.txt
            read>>name>>salary;
            deduction = salary * TAX;
            write <<"\n"<<name<<" \t\t "<<salary<<" \t\t "<<deduction;
      }

return 0;
}
```



```
Name           Salary         Deduction
------         -------        ---------
Roslan         4500           360
Linda          3500           280
Zamani         10000          800
```

viii. ***Formatting Numeric Output***

- It is important that the numerical results are presented attractively and formatted correctly. In C++, there are a number of stream manipulators that can be used commonly for that purpose, as illustrated in the following table.

Manipulators	Action
`setw(n)`	Set the field width to n
`setprecision(n)`	Set the decimal point precision to n places
`setiosflags(ios::fixed)`	Display the number in conventional fixed-point decimal notation
`setiosflags(ios::showpoint)`	Display a decimal point of that number

- To used stream manipulators, the `iomanip.h` header file must be included as part of the program. This is accomplished by the preprocessor command `#include <iomanip.h>`.

Example of usage:

```
#include <iostream>
#include <iomanip.h>

int main()
{
      cout << setw(8) << 15 << endl;
      cout << setw(8) << 256 <<endl;
      cout << setw(8) << setprecision(2)<< 88.5 << endl;
      cout << setw(8) << setiosflags(ios::fixed) << setprecision(2) <<
      26.583 << endl;
      cout << setw(8) << 72.9 << endl;

return 0;
}
```

- Sample output on screen (represented by column width)

```
1      2      3      4      5      6      7      8
---------------------------------------------------------------------------
                                         1      5
                                  2      5      6
                     8      8     .      5      0
                     5      6     .      5      8
                     7      2     .      9      0
```

ix. ***Character input with member function: cin.getline()***

- `cin.getline()` is a predefined function to read the entire line of input from user as a string including blank space and newline character.
- To use string predefined functions, `string.h` header file must be included in the program.
- The syntax take the form

  `cin.getline(variable, length);`
- For example,

  `cin.getline (text, 80);`

  indicates that text is a variable used to store a string of characters, and 80 is the maximum length the variable `text` can store.

Example of `cin.getline()` usage:

```
#include <iostream.h>

int main()
{     char text[80];
      cout << "Enter a line of text: ";
      cin.getline(text,80);
      cout << "\n You have entered: " << text << endl;
return 0;
}
```

Output screen:

```
Enter a line of text: Have a nice day!
You have entered: Have a nice day!
```

x. ***String library function: `strcpy()`***

- `strcpy()` is another string predefined function used to copy a string into another string variable.
- The syntax takes the form `strcpy(x, y)`, where the contents of string `y` is copied into string `x`.

Example:

The following program will copy "`Hi there`" into the string variable greeting.

```
#include <iostream.h>
#include <string.h>

int main()
{
      char greeting[80];
      strcpy(greeting, "Hi there");
      cout << greeting;

return 0;
}
```

```
//OUTPUT

Hi there
```

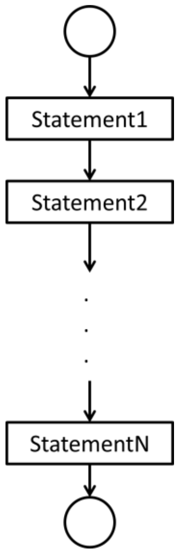
xi. ***C++ block structure (program control structure)***

- Program control structure controls the flow of execution of program statement. C++ provides structures that will allow an instruction or a block of instruction to be executed, repeated or skipped.
- There are four control structures:
  i. Sequential structure
  ii. Selection structure (by making a choice, which called a branch).
  iii. Repetition structure (by executing a statement over and over using a structure called a loop or by calling a function).
  iv. Modular structure (calling a function).

1. Sequence structure

- The sequential structure has one entry point and one exit point as shown below.
- No choices are made and no repetition.
- Statements are executed in sequence, one after another without leaving out any single statement.

Example:

Sequential Structure

Example of sequential programming

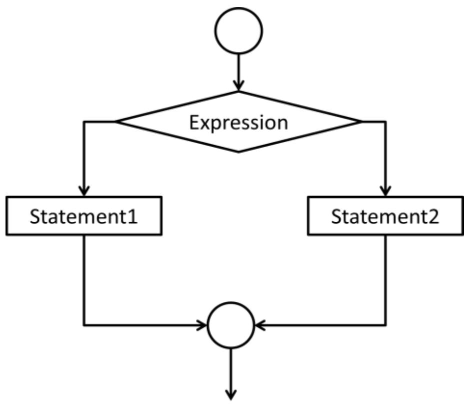
```
#include <iostream.h>

int main()
{
      float payRate = 8.5;
      float hoursWorked = 25.0;
      float wages;
      wages = hoursWorked * payRate;
      cout << "\n Wages = RM " << wages << endl;

return 0;
}
```

2. Selection structure

- The selection structure is used to allow choices to be made.
- The program executes particular statements depending on some condition(s).
- C++ uses the `if-else` and `switch` statement for making decision.

Example:

Selection Structure



Example of selection programming

```
#include <iostream.h>

int main()
{
      char code;
      cout << "\n Enter code (M/F) ";
      cin >> code;

      if (code == 'm' || code == 'M')
            cout << "\n Male" << endl;
      else if (code == 'f' || code == 'F')
            cout <<"\n Female" << endl;
      else
            cout << "\n Invalid code";

return 0;
}
```

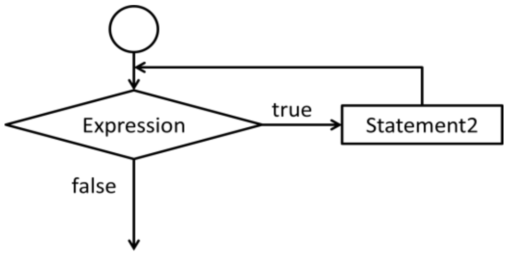
References:
Malik, D.S., (2010). *C++ Programming, From Problem Analysis to Program Design*. Course Technology, Canada.
Gary, R.B., (2010). *C++ For Engineers & Scientists: 3rd Edition.* Course Technology, Canada.

3. Repetition structure

- Statements are executed repeatedly while certain condition remains true.
- In the C++, `while`, `do-while` and `for` are the statements commonly used within the repetition structure.

Example:

Repetition Structure



Example of repetition programming

```
#include <iostream.h>

int main()
{
      int num;

      cout << "\n Enter number :  ";
      cin >> num;

      while (num>0)
      {
            cout << "\n Number is positive number";
            cout << "\n Enter number :  ";
            cin >> num;
      }
      cout << "\n End of program";

return 0;
}
```

TUTORIAL

1. List TWO (2) rules for naming an identifier?
2. Determine which of the following are valid identifiers?

Identifiers	Answer
a. `r2d2`	
b. `H20`	
c. `secondCousinOnceRemove`	
d. `the_Legend-City_of_Malaysia`	
e. `_TIME_`	
f. `_12345`	
g. `x(3)`	
h. `cost_in_$`	

3. State the difference between identifier and keyword.
4. Which of the following is reserved word in C++?

Identifiers	Answer
a. `integer`	
b. `long`	
c. `float`	
d. `single`	
e. `double`	
f. `char`	

5. What is the difference between variable and constant?
6. Write a constant declaration for each of the following:
   a. A constant that contains the number of minutes in an hour.
   b. A constant that holds the value of PI.
7. Give the most appropriate data type for each of the following values:

Identifiers	Answer
a. `23`	
b. `‘c’`	
c. `8.52`	
d. `9537`	
e. `20125.12345`	
f. `True`	

8. Create a variable name for each of the following and provide the appropriate data type:

Descriptions	Variable Name	Data Type
a. Speed of an automobile		
b. Shipping rate per ringgit		
c. Highest score in exam		
d. Initial 'm' for male or 'f' for female		
e. Amount of students in a class		
f. Pass or Fail		

9. For the following short scenario, list the variables that you need to solve the problem. In your list, include the variable name, its data type and the declaration statement that you would use.

   Your friend asked you to write a program that will calculate the area of rectangle and the total price of a tile. The program will receive a length and width, in feet of a rectangle, and the price of a square foot of tile.

Variables	Data Type	Declaration

PROBLEM SOLVING AND PROGRAMMING ACTIVITIES

1. Write a C++ statement for each of the following:
   a. How can you test that an integer number is a multiple of 2?
   b. How can you test that an integer number is a multiple of 3?
   c. How can you test that an integer number is an even number?
   d. How can you test that an integer number is an odd number?
   e. How can you test that an integer number is divisible by 5 and not divisible by 7?
2. Covert each of the following mathematical formulae to C++ expression:
   a. $3x$
   b. $6x + 2y$
   c. $\frac{x+y}{6}$
   d. $\frac{\sqrt{b^2-4ac}}{2}$
   e. $A = \pi r^2$
3. If x = 6; and y = 7; then after the statement x = y; what is the value of x and y?
4. Complete the table below by filling in the value of each variable after each line is executed.

C++ Program	Value of x	Value of y
`#include <iostream.h>`		
`int main()`		
`{`		
`int x, y;`		
`x = 10;`		
`y = 3 * x;`		
`x = y + x;`		
`cout << x + 3;`		
`y = y + 2;`		
`cout <<  endl << y;`		
`return 0;`		
`}`		

References:
Malik, D.S., (2010). *C++ Programming, From Problem Analysis to Program Design*. Course Technology, Canada.
Gary, R.B., (2010). *C++ For Engineers & Scientists: 3rd Edition.* Course Technology, Canada.

5. Write a statement for each step given below. Declare all variables used by the given names.

a. The following steps are to calculate a company's profit:

Steps	Statement in C++
Assign the value `85347` to the variable `revenue`.	
Assign the value `53210` to the variable `cost`.	
Assign the difference between the variables `revenue` and `cost` to the variable `profit`.	
Display the value of the variable `profit`.	

b. The following steps are to calculate the price of an item after a 15% reduction.

Steps	Statement in C++
Assign the value `20.25` to the variable `price`.	
Assign the value `15` to the variable `discPercent`.	
Assign the value `discPercent` divided by 100 times `price` to the variable `discount`.	
Decrease `price` by `discount`.	
Display the value of `price`.	

c. The following steps are to compute the price of drinks.

Steps	Statement in C++
Assign "`coffee`" to the variable `drinks`.	
Assign `2.50` to the variable `price`.	
Assign `4` to the variable `cupsOfCoffe`.	
Assign the value `price` times `cupsOfCoffe` to the variable `totalPrice`.	
Display the phrase "`You've ordered 4 cups of coffee and cost you RM10.00`".	

References:
Malik, D.S., (2010). *C++ Programming, From Problem Analysis to Program Design*. Course Technology, Canada.
Gary, R.B., (2010). *C++ For Engineers & Scientists: 3rd Edition.* Course Technology, Canada.

6. Write another C++ statement equivalent to the statements below:

   a. `x = x + 3`

   b. `x = x + x`

   c. `y = y – x`

   d. `w = w / 5`

   e. `z = z * y`

7. Evaluate the numeric integer expression below where `a = 2`, `b = 3`, and `c = 4`.

Statements	Answer
i. (a * b) + c	
ii. a + b * c	
iii. (c - a) % b	
iv. (a * a + b * b + c * c) / 2	
v. a++ + --c	
vi. ++a – b * c	

8. Write another C++ statement that is equivalent to each of the following statement below

Statements	Answer
i. y /= 9 / z	
ii. ++number	
iii. number++	
iv. w *= w * w	
v. sum = sum + num	
vi. x += 5 – w	
vii. y /= 2	

9. The numbers of calories burnt per hour by cycling, jogging, and swimming are 200, 475 and 275 respectively. A person loses 1 pound of weight for each 3500 calories burnt. Write a program that allows the user input the number of hours spent at each activity and then calculates the number of pounds worked off.

10. Suka-Suki Employer wants a program that will allow the company's clerk to enter an employee's name and the number of hours the employee works every month, (the number of hours worked will always be an integer). The program will display the name, numbers of weeks (assume a 40-hour week), days (assume an eight-hour day), and hours worked. For example, if the employee enters the number 70, the program will display employee's name, then 1 week, 3 days, and 6 hours.
    a. List the output, input and process.
    b. Draw the flowchart.
    c. Write a program using C++ programming language.
11. Design an algorithm for a program to convert temperature in degrees Fahrenheit to degree Celsius. The equation for this conversion is: `Celsius = 5.0 / 9.0 (Fahrenheit -32.0)`. Then write a program using the C++ programming language based on the flowchart.
12. Design an algorithm for a program to calculate the sum of the arbitrary numbers from 1 to 100. The formula for calculating this sum is:
    ```
    sum = (n / 2) (2 * a + (n - 1) d)
    ```
    where `n` is number of terms to be added, `a` is the first number and `d` is the difference between each number. Afterwards, translate the algorithm into C++ program.
13. Shipping Industries needs a program that allows its shipping staff to enter an item's name, the quantity of the item in inventory, and how many units of the item can be packed in a box for shipping. The program should display the item's name, the number of full boxes that can be packed from the quantity on hand, and the quantity items left over.
    a. List the output, input and process for this task.
    b. Draw the flowchart
    c. Write a C++ program

14. Write a program to create a customer's bill for a wholesale company. The company sells five computer gadgets which are CD, keyboard, printer, pendrive, and speaker in one package. The unit prices are RM60.00 per pack, RM130.00, RM850.00, RM65.00, and RM72.50 respectively. The program must read from the keyboard the quantity of each piece of equipment purchased. It then calculates the cost of each item, the subtotal, and the total cost after 3.25% sales tax.

    The input data consist of a set of integers representing the quantities of each items sold. Example of the input is shown below:

```
      ++++++++++++++++++++++++++++++++++
      +     SUKA-SUKI WHOLESALE MART    +
      ++++++++++++++++++++++++++++++++++

How many pack of CDs were sold   :     5
How many keyboards were sold     :     120
How many printers were sold      :     35
How many pendrives were sold     :     236
How many speakers were sold      :     83
```

    The format for the output is as below:

```
      ++++++++++++++++++++++++++++++++++
      +     SUKA-SUKI WHOLESALE MART    +
      ++++++++++++++++++++++++++++++++++

QTY   DETAILS          UNIT PRICE            TOTAL PRICE
---   -------          ----------            -----------
XX    CD                     60.00               XXXX.XX
XX    KEYBOARD              130.00               XXXX.XX
XX    PRINTER               850.00               XXXX.XX
XX    PENDRIVE               65.00               XXXX.XX
XX    SPEAKER                72.50               XXXX.XX
                                             -----------
                       SUBTOTAL                 XXXXX.XX
                       TAX                       XXXX.XX
                       TOTAL                    XXXXX.XX
```

    Use **CONSTANT** for the unit price and the tax rate.

References:
Malik, D.S., (2010). *C++ Programming, From Problem Analysis to Program Design*. Course Technology, Canada.
Gary, R.B., (2010). *C++ For Engineers & Scientists: 3rd Edition.* Course Technology, Canada.