Chapter-8

DATA TYPES

➢ Introduction
• To understand any programming languages we need to first understand the elementary concepts which form the building block of that program.
• The basic building blocks include the variables, data types etc.
• C++ provides a set of data types to handle the data that is used by the program.

➢ Variable:
• A variable is an object or element and it is allowed change during the execution of the program.
• Variable represents the name of the memory location.

✓ Declaration of a variable:
• The syntax for declaring a variable is:
  datatype variable_name;
• The variable_name is an identifier. These variables are used to denote constants, arrays, function, structures, classes and files.
• The variables are named storage location whose values can be manipulated during program run.
• Examples:
  Some valid variables are:
  reg_no, marks, name, student1, dob;
  Some invalid variables are:
  Double - keyword cannot be name of the variable.
  Total marks - empty spaces are not allowed between variables names
  2student - variable name should be begin with an alphabet
  ?result - variable should begin with alphabet or underscore only.

✓ Initializing a variable:
• The syntax to initialize a variable is:
  data_type variable_name = value;
• Example: Let b be a variable declared of the type int. then
  int b = 100;
• There are two values associated with a variable known as lvaue and rvalue. It means, for example, let p be a variable declared of the type int. then
int p = 100;

- Here, name of the variable is p
  - values assigned to variable is 100 i.e. rvalue
  - memory address location is 2000 i.e. lvalue

- Lvalue is the location value. It holds the memory address location at which the data value is stored.
- Rvalue is the data value. It holds the value assigned to the variable by the programmer i.e. Rvalue of p = 100.

- C++ compiler allows us to declare a variable at run time. This is dynamic initialization. They can be initialized anywhere in the program before they are used.
- The access modifier ‘const’ prefixed along with the data type for a variable does not allow the value to be changed at all throughout the program.
  ```
  cont int a = 100;
  ```
- The keyword const becomes an access modifier for the variables.
  - A value can be assigned to lvalue only in an expression.
  - Value can be assigned to a variable using the assignment operator ‘=’.
  - The expression to the left of an assignment operator should always be an lvalue (memory location) because that memory location should be available to store the rvalue.
  - Constant identifiers can appear to the right of the assignment operator only since are not lvalues.

> Data Types:

- Data Types can be defined as the set of values which can be stored in a variable along with the operations that can be performed on those values.
- The main aim of C++ program is to manipulate data.
- C++ defines several types of data and each type has unique characteristics.
- C++ data types can be classified as:
  1. The fundamental data type(built-in data)
  2. Derived Data type
  3. User-defined data type
- The simple or fundamental data types are the primary data types which are not composed of any other data types.
- The simple data types/fundamental data types include int, char, float, double and void.
The int type:

- The int type is used to store integers.
- Integers are whole numbers without any fractional parts.
- This includes numbers such as 1, 45, and -9 are integers.
- 5.2 is not an integer because it contains a decimal point.
- The integer can be positive or negative values and the ranges of numbers we can store are from -32786 to 32767.
- An integer is allocated 2 bytes (16 bits) of memory space.
- The possible operations include addition, subtraction, multiplication, division, remainder etc.
- The general form of an integer declaration is:
  
  ```
  int variable_name;
  ```

- Example: `int a, b=5;`

The char type:

- It is a character data type to store any character from the basic character set.
- Characters are enclosed in single quotation marks ('). 'A', 'a', 'b', '9', '+' etc. are character constants.
- When a variable of type char is declared, the compiler converts the character to its equivalent ASCII code.
- A character is allocated only 1 byte (8 bits) of memory space.
- A character is represented in a program by the keyboard char.
- The general form of a character declaration is:
  
  ```
  char variable_list;
  ```

- Example: `char alpha='a';`
The float type:
- This represents the number with fractional part i.e. real numbers.
- The float type is used to store real numbers.
- Number such as 1.8, 4.5, 12e-5 and -9.66 are all floating point numbers.
- It can also be both positive and negative. The range of numbers we can store from -34e-38 to 3.4e38.
- Float is allocated 4 bytes (32 bits) of memory space.
- The general form of a float declaration is:
  ```
  float variable_name;
  ```
- Example: `float a=5.5;`

The double type:
- The double and float are very similar. The float type allows you to store single precision floating point numbers, while the double keyword allows you to store double precision floating point numbers.
- Its size is typically 8 bytes of memory space.
- The range of numbers we can store are from -1.7e308 to 1.7e308.
- The general form of a double declaration is:
  ```
  double variable_list;
  ```
- Example: `double a = 5.5e-7;` //a is equivalent to 5.5x10^-7

The void type:
- The void data type has no values and no operations.
- In other words both the set of values and set of operations are empty.
- Example: `void main( )`
- In this declaration the main function does not return any value.

The bool type:
- The bool type has logical value `true` or `false`. The identifier `true` has the value 1, and the identifier `false` has the value 0.
- The general form of a bool declaration is:
  ```
  bool variable_name;
  ```
- Example: `bool legal_age=true;`
- The statement `legal_age= (age>=21);` assigns the value true if age is greater than or equal to 21 or else it returns the value false.
### Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Size in bytes</th>
<th>Range</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>2</td>
<td>-32768 to 32767</td>
<td>8, 100, -39</td>
</tr>
<tr>
<td>char</td>
<td>1</td>
<td>-128 to 127</td>
<td>‘d’, ‘6’, ‘#’</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
<td>3.4 x 10^-38 to 3.4 x 10^38 -1</td>
<td>45.345, 0.134, 3.142</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
<td>1.7 x 10^-308 to 1.7 x 10^308 -1</td>
<td>3.1415678888888888888</td>
</tr>
</tbody>
</table>

#### Derived data types
- These data types are constructed using simple or fundamental data types.
- This includes arrays, functions, pointers and references.

#### User defined data types
- These data types are also constructed using simple or fundamental data types.
- Some user defined data types include structure, union, class and enumerated.

#### Enumerated data type:
- An enumeration is a user defined type consisting of a set of named constants called enumerators.
- `enum` is a keyword that assigns values 0, 1, 2…… automatically.
- This helps in providing an alternative means for creating symbolic constants.
- The syntax for `enum` is as follows:
  ```
  enum [tag] { enum – list} ;
  ```
  //for definition for enumerated type
  ```
  enumtaddeclarator;
  ```
  //for declaration of variable type tag
- Example 1:
  ```
  enum choice { very_bad, bad, satisfactory, good, very_good};
  choice mychoice;
  ```
- Example 2:
  ```
  enum MyEnumType { ALPHA, BETA, GAMMA };
  ```
  Here, ALPHA takes the value 0, BETA takes the value of 1, GAMMA takes the value of 2.
- Example 3:
  ```
  enum footsize { small = 5, medium = 7, large = 10};
  ```

#### C++ enum type conversion rules:
- These rules apply to C++ only.
- There is an implicit conversion from any enum type to int. Suppose this type exists.
  ```
  enum MyEnumType { ALPHA, BETA, GAMMA };
  ```
- Then the following lines are legal
int i = BETA; //give i value of 1
int j = 3 + GAMMA //give j a value of 5

- On the otherhand, C++ does not support an implicit conversion form int to an enum type. This type conversion is always illegal.

    MyEnumType x = 2; //should not be allowed by compiler
    MyEnumType y = 123; //should not be allowed by compiler

- Note that it does not matter whether the int matches one of the constants of the num type.

**CHAPTER 8 – DATA TYPES BLUE PRINT**

<table>
<thead>
<tr>
<th>VSA (1 marks)</th>
<th>SA (2 marks)</th>
<th>LA (3 Marks)</th>
<th>Essay (5 Marks)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>01 Question</td>
<td></td>
<td></td>
<td>02 Marks</td>
</tr>
</tbody>
</table>