CHAPTER 7
DATA HANDLING

CONCEPT OF DATA TYPES

- ✓ Data can be of many types e.g. character, string, integer, real etc.

- ✓ C++ like any other language provides ways and facilities to handle different types of data by providing data types.

DATA TYPES (def.)

- **DATA TYPES** are means to identify the type of data and associated operations of handling it.

C++ DATA TYPES

- ✓ C++ data types are of two types:
- ✓ Fundamental types
- ✓ Derived types

FUDAMENTAL DATA TYPES

There are 5 fundamental data types:

1. **int**
2. **float**
3. **double**
4. **void**
5. **char**

DERIVED DATA TYPES

Derived data types constructed from fundamental data types are:

1. Array
2. Functions
3. Pointers
4. References
5. Constants
6. Classes
7. Structures
8. Unions
9. enumerations

FUNDAMENTAL DATA TYPES

Fundamental (atomic) data types are those that are not composed of other data types.

INT

- **Int data type (for integers).** Integers are whole numbers (without any fractional part).
- Ex:123,-789 etc.

/* PROGRAM THAT ILLUSTRATES INT DATA TYPE*/

```
#include<iostream.h>
void main()
{
int n=45;
cout<<"the integer value is "<<n;
getch();
}
```

NOTE: n is an integer number which holds 2 bytes of memory. the value ranges from -32768 to +32767.

CHAR

- Char data type (for characters). Characters can store any member of the C++ implementation's basic character set.

CHAR

- Char type is often said to be an integer type.
- Because they are represented by associated number code (ascii code).
- ASCII code stores numbers codes for 256 known characters.

PROGRAM

```
#include<iostream.h>
void main()
{
char c='a';
cout<<"the char value is "<<c;
getch();
}
```

Note:c is a character type data defined by char keyword which holds 1 byte of memory and it stores a single character.

PROGRAM

- The above program initialises a character variable *ch* with character ‘a’. The second statement of *main()* initialisses integer variable *num* with the associated number code of *ch.* (*ch* is having ‘a’ and associated ASCII code of ‘A’ is 65).

FLOAT

Float data type (for floating-point numbers). A number having fractional part is known as floating point number.

/* Program that illustrates float data type*/

```
#include<iostream.h>
void main()
{
float n=45.123;
cout<<"the float value is "<<n;
getch();
}
```

NOTE: n is a floating-point number which holds 4 bytes of memory. the value ranges from -3.4×10^{-38} to $3.4 \times 10^{38} - 1$ and digits of precision is 7

MORE ABOUT FLOAT DATA TYPE

- Floating point numbers have two *advantages* over integers.

1. They can represent values between the integers.
2. They can represent a greater range of values.

MORE ABOUT FLOAT DATA TYPE

- But they one disadvantage also.

Floating-point operations are usually slower than integer operations.

DOUBLE

Double data type (for double precision floating-point numbers). The data type double is also used for handling floating-point numbers. But it is treated as a distinct data type because it occupies twice as much memory as type float, and stores floating-point with much larger range and precision (significant numbers after decimal point). It stands for double precision floating-point.it is used when type float is too small or insufficiently precise.

DOUBLE

- Type double is slower than float.
- You must use the smallest data type which is sufficient.

NOTE: double type number which holds 8 bytes of memory. the value ranges from 17×10^{-308} to $3.4 \times 10^{308} - 1$ and digits of precision is 15

VOID

Void data type(for empty set of values and non-returning functions). The void type specifies an empty set of values. It is used as the return type for functions that do not return a value. No object of type void may be declared.

DATA TYPE MODIFIERS

A modifier is used to alter the meaning of base type to fit various situations more precisely.

MODIFIERS

List of modifiers

1. signed
2. unsigned
3. long
4. short

INTEGER TYPE MODIFIERS

By using different numbers of bytes to store values, c++ offers three types of integers:

1. Short
2. Long
3. int

INTEGER TYPE MODIFIERS

Type	Size(bytes)	Range
short	2	−32768 to 32767
long	4	−2,147,483,648 to 2,147,483,647
int	2	−32768 to 32767

CHARACTER TYPE MODIFIERS

- ✓ The *char* type is really another integer type (as inside memory it actually holds numbers i.e. equivalent codes of characters/symbols).
- ✓ The *char* type can also be *signed* or *unsigned.*
- ✓ Unlike int, char is neither signed or unsigned by default.
- ✓ The unsigned char represents the range 0 to 255 and signed represents the range −128 to 127.

FLOATING-POINT TYPE MODIFIERS

- C++ has three floating-point types:

1. float
2. double
3. long double

DERIVED DATA TYPES ARRAYS

- Arrays refer to a named list of a finite number n of similar data elements.
- Arrays can be one dimensional, two dimensional, or multi dimensional.

Further discussion in a different chapter.

FUNCTIONS

- A function is named part of program that can be invoked from other parts of program as often needed.

POINTERS

- A pointer is a variable that holds a memory address.
- The address is usually the location of another variable in memory.
- If one variable contains the address of another variable, it is said to *point* to the second.

REFERENCE

- A *reference* is an alternative name for an object. It provides an *alias* for a previously defined variable.

Syntax

type &ref–var =var–name;

TAKE A NOTE

NOTE

There can be no references to references, no arrays of references, and no pointers to references.

CONSTANTS

- The keyword *const* can be added to the to the variable to make an object a constant rather than a variable.
- Thus, the value of *named constant* cannot be altered during the program run.

SYNTAX

- The general form of constant declaration is given below:
- const type name = value;
- Where const is the keyword that must be used for declaring a constant, type is any valid c++ data type, name is the name of the constant and value is the constant value of the data type.
- For ex: const int upperage =50;

Declares a constant named as upperage of type interger that holds value 50.

Be careful!!!!

- A constant must be initialized at the time of declaration.
- If you give only const in place of const int, it means the same.

NOTE the const modifier on it's own is equivalent to const int.

USER DEFINED DERIVED DATA TYPES

- There are some derived dat types that are defined by the user. These are :

1. Class
2. structure
3. union
4. enumeration

CLASS

- A class represents a group of similar objects.
- A class bears the same relationship to an object that a type does to a variable.
- **Note :—** The class describes all the properties of a data type, and an object is an entity created according that description.

STRUCTURE

A structure is collection of variables (of different data types) referenced under one name, providing a convenient means of keeping related information togrther.

For ex:-

Insert a program

DIFFERNCES

- A structure is different from an array in the sense that an array represents an aggregate of elements of same type whereas a structure represents an aggregate of elements of (nearly) arbitrary types.
- A structure is different from a class in the sense that a class can represent data elements as well as their associated functions whereas a structure can represent only data elements, not their associated functions.

UNION

- A union is a memory location that is shared by two or more different variables, generally of different types at different times.
- The keyword *union* is used for declaring and crating a union.

ENUMERATION

- An alternative method for naming integer constants is often more convenient than *const.*
- This can be achieved by creating enumeration using keyword *enum.*
- *Ex:*

WHY SO MANY DATA TYPES?

- The reason for providing so many data types is to allow programmer to take advantage of hardware characteristics. Machines are significantly different in their memory requirements, memory access times (time taken to read memory), and computation speeds.

VARIABLES

- Variables represent named storage locations, whose value can be manipulated during program run.
- There are *two* values associated with a symbolic variable :

1. Its data value, stored at some location in memory. This is sometimes referred to a s a variable's *rvalue* (pronounced *"are-value"*).
2. Its location value ;i.e., the address in memory at which its data value is stored. This is sometimes referred to as a variable's *lvalue* (pronounced as *"el-value").*

NOTE

- Remember that constant values and constant identifiers (declared with const keyword) are not lvalues and can only appear to the right side of an assignment operator.

DECLARATION OF A VARIABLE

- Any variable in c++ takes the following:-
- *Type* name
- Where type is valid c++ data type
- Name is any name you give to the variable(an identifier).

SYNTAX

- Syntax of variable in c++ :--

type var-name = value;

Here type is a keyword

Var-name is an identifier

=is an assignment operator

Value is the value you assign to the variable.

INITIALIZATION OF A VARIABLE

- A variable with a declared first value is said to be an *initialised variable.*
- *C+= supports two forms* of variable initialization at the time of variable definition:
- Int val =1001;
- And
- Int val (1001);

NOTE

- Please note that an small l (small L) or L suffix on an integer means the integer is a type long constant, a u or U suffix indicates an unsigned int constant and ul or lu indicates a type unsigned long constant.
- A variable can be declared anywhere in the c++ program but before its first reference.

DYNAMIC INITIALIZATION

- One additional feature of c++ is that it permits initialization of variable at run time.this is referred to as *dynamic initialization.*
- Ex:--
- float avg;
- avg=sum/count;
- The above two statement can be combined into one as follows :--
- float avg=sum/count;

THE ACCESS MODIFIER CONST

- If you use modifiers const or constants before a variable's definition, it modifies its access type i.e., the access of the const variable is read only; it can only be read and written on to.
- For ex:—
- Int val=10;
- The above statement initiates a variable vak which holds 10.
- Now
- Const int val=10;
- The difference is that the const will never change throughout the program run whereas simply int val will change as required .

FORMATTING OUTPUT

- C++ offers various i/o manipulators ;two of which are **setw()** and **setprecision()**.
- In order to use these manipulator you must include header file **iomanip.h**.

SETW() MANIPULATOR

- The setw manipulator sets the width of the field assigned for the output.

✓ How to use it?

cout<<setw(6) <<"R" ;

The above statement will produce following output :--

_ _ _ _ _R (each underscore represents a blank.)

NOTE

- The **setw()** manipulator **does not stick** from one cout statement to the next.

SETPRECISION() MANIPULATOR

- The **setprecision()** manipulator sets the total number of digits to be displayed when floating point numbers are printed.
- For ex:—
- Cout<<setprecision(5)<<123.456 ;
- Will print the following output to the screen (notice the rounding)

123.46

SETPRECISION ()contd…

- ✓ The setprecision() manipulator can also be used to **set the number of decimal places to be displayed.**
- ✓ But In Order To Implement It you need to set an ios flag.
- ✓ The flag is set with the following statement :

```
cout.setf(ios::fixed) ;
```

- ✓ Now setprecision will set the number of decimal places.

For ex:--

```
cout<<setprecision(5)<<12.3456789;
```

Output will be

12.34567

ADDITIONAL IOS FLAGS

- ✓ These are the format options. Other format options can be one of the following:
- ✓ Left ---left -justify the following
- ✓ Right---right-justify the following
- ✓ Showpoint---display decimal point and trailing zeroes for all floating point numbers, even if the decimal places are not needed
- ✓ Uppercase---display the "e" in E-notation as "E" rather than "e"
- ✓ Showpos---display a leading plus sign before positive values
- ✓ Scientific---display floating-point numbers in scientific ("E")notation
- ✓ Fixed---display floating-point numbers in normal notation – no trailing zeroes and no scientific notation

NOTE

- You can remove these options by replacing setf used with cout, recall cout.setf) with unsetf.
- Please note that the manipulator setprecision does sticks that is, the same format is forwarded for next cout statements. (it is ***sticky).s***

THANK YOU