Introduction to C Language

- The C character set, identifiers and keywords, data types, constants, symbolic constants, variables and declarations, operators, expressions, statements, Library Functions.

C Language

- The programming language C is a general purpose computer language.
- It is a structured, high-level and machine independent language.
- It was originally created for the specific purpose of writing Operating System software.

C Language

- The development in C has seen many evolutionary processes.
- A lot of new features have been added to make it more useful and powerful.
- C was evolved from ALGOL, BCPL and B by ***Dennis Ritchie*** at the *Bell Laboratories* in 1972.

C Language

- The language became more popular after the publication of the book ‘The C Programming Language’ by **Brain Kerningham and Dennis Ritchie** in 1978.
- To assure that the C language remains standard, in 1983,
- *American National Standards Institute* (ANSI) appointed a technical committee to define a standard for C, the committee approved a version of C in December 1989 which is now known as ***ANSI C***.
- It was then approved by the *International Standards Organization* (ISO) in 1990.

FUNDAMENTALS OF C LANGUAGE

- The basic elements of C language can be classified as follows:
    - Identifiers
    - Keywords
    - Constants
    - C Character set

Identifiers

- Identifiers are names that are given to various program elements, such as variables, functions and arrays.
- Identifiers consist of letters and digits in any order except that the first character must be a letter.

Identifiers

- To construct an identifier you must obey the following points :
- Only alphabet, digit and underscores are permitted
- An identifier can’t start with a digit.
- Identifiers are case sensitive, i.e. uppercase and lower case letters are distinct.
- Maximum length of an identifier is 32 characters.

Keywords (Reserved words)

- Reserved words are the essential part of a language definition.
- The meaning of these words has already been explained to the C compiler.
- So you can’t use these reserved words as a variable names.
- Since these reserved words have some special meaning in C, therefore these words are often known as “keyword”.

Keywords (Reserved words)

auto *double* *if* *static*
break *else* *int* *struct*
case *enum* *long* *switch*
char *extern* *near*
typedef
const *far* *register* *union*
continue *float* *return* *unsigned*
default *for* *short* *void*
do *goto* *signed* *while*

Constants

- A ***constant*** is a container to store value. But you can't change the value of that container (constant) during the execution of the program.
- Thus, the value of a constant remains constant through the complete program.

Constants

- There are two broad categories of constant in C, ***literal constant*** and ***symbolic constant***.
- A literal constant is just a value. For example, 10 is a literal constant. It does not have a name, just a literal value.
- Depending of the type of data, literal constant is of different type.
- They include *integer*, *character* and *floating point* constant.
- Integer constant can again be subdivided into *decimal* (base-10), *octal* (base-8), and *hexadecimal* (base-16) integer constant.

Constants

- One important variation of character constant is *string constant*.
- Character constant is always enclosed with single quotation mark, whereas a string constant is always enclosed with a double quotation mark.
- An octal integer constant always starts with 0 and a hexadecimal constant with 0x.

Constants

EXAMPLE	TYPES
153	Decimal integer constant
015	Octal integer constant
0xA1	Hexadecimal integer constant
153.371	Floating point constant
‘a’	Character constant
‘1’	Character constant
“a”	String constant
“153”	String Constant

SYMBOLIC CONSTANTS IN C

- A *symbolic constant* is a name that substitutes for a sequence of characters.
- a symbolic constant allows a name to appear in place of a numeric constant, character constant or a string.
- When a program is compiled, each occurrence of a symbolic constant is replaced by its corresponding character sequence.

SYMBOLIC CONSTANTS IN C

- Symbolic constants are usually defined at the beginning of a program.
- Symbolic constants are defined using *#define* as given below:

  #define<symbolic constant name> <value>

  #define PI 3.14

  #define TAXRATE 0.55

  #define TRUE 1

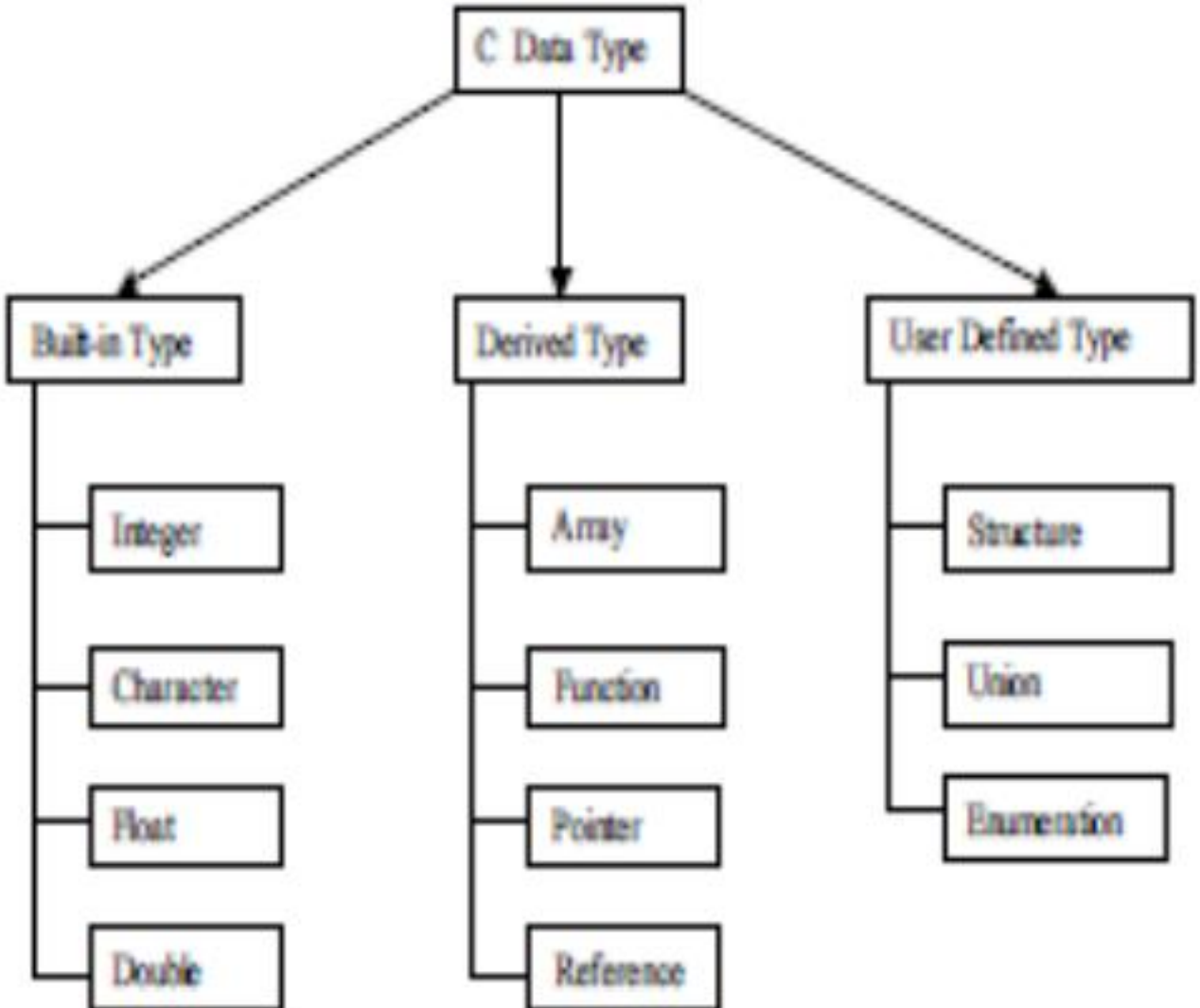
  #define FALSE 0

Character Set

The only characters required by the C Programming Language are

- Alphabets A – Z, a - z
- Digits 0 - 9
- Special symbols # & | ! ? _ ~ ^ { } [ ] ( ) < >, space . , : ; ‘ $ “ + - / * = %

BASIC DATA TYPES IN C



Format specifier or *Conversion specifier*

- The “%” symbol along with a (special) character in known as ***format specifier*** or ***conversion specifier***.
- It indicates the data type to be printed or scanned and how that data type is converted to the character that appears on the screen.

Format specifier or *Conversion specifier*

Format Specifier	**Usual VariableType -**	**Display as**
%c	char single	character
%d	int signed	integer
%f %lf	float or double	signed decimal
%e %le	float or double	exponential format
%o	int	unsigned octal value
%u	int	unsigned integer
%x	int	unsigned hex value
%ld	int	long decimal integer
%s	array of char	sequence of

VARIABLES AND THEIR DECLARATIONS

- A variable is an identifier to store value.
- You can resemble a variable with a container which takes different values at different time during the execution of the program.
- Thus, the value of the variable may change within the program.
- A variable name can be chosen by the programmer in a meaningful way that reflects what it represents in the program.
- The naming convention of variable follows the rule of constructing identifiers.

VARIABLES AND THEIR DECLARATIONS

```
int A ;
A = 153 ;
```

- The first statement says that A is a container, where we can store only integer type variable.
- This type of statement is known as declaration statement (A declares that A can store only integer type of variable).

The general form of declaration of a variable is

data_type variable1, variable2, ... , variableN;

VARIABLES AND THEIR DECLARATIONS

By declaring a variable we tell 3 things to the compiler :

- What the variable name is.
- What type of data the variable will hold.
- and the scope of the variable.

You can store values to a variable in two ways

- By using assignment statement.
- By using a read statement.

OPERATORS

- ***Operators*** are special symbols which instruct the compiler to perform certain mathematical or logical manipulations.
- Operators are used in programs to manipulate data and variables.
- The data items that operators act upon are called ***operands***.
- Operators are used with operands to build expressions. Some operators require two operands, while other act upon only one operand.

EXPRESSIONS

- C Expressions are based on algebra expressions - they are very similar to what we learn in Algebra, but they are not exactly the same.
- An expression is a combination of variables, constants and operators written according to the syntax of C language.
- In C every expression evaluates to a value i.e., every expression results in some value of a certain type that can be assigned to a variable.

EXPRESSIONS

- ***Expressions*** in C are syntactically valid combinations of operators and operands that compute to a value determined by the priority and associativity of the operators.
- Some examples of expressions:

```
float a, b, c x, y, z;
a = 9;          // a variable
b = 12; c=3;
x = a – b / 3 + c * 2 – 1;
y = a – b / (3 + c) * (2 – 1);
z = a – ( b / (3 + c) * 2) – 1;
```

EXPRESSIONS

```
printf (“x = %fn”,x);
printf (“y = %fn”,y);
printf (“z = %fn”,z);
```

Output

x = 10.00

y = 7.00

z = 4.00

Rules for evaluation of expression

- First parenthesized sub expression left to right are evaluated.
- If parenthesis are nested, the evaluation begins with the inner most sub expression.
- The precedence rule is applied in determining the order of application of operators in evaluating sub expressions.

Rules for evaluation of expression

- The associability rule is applied when two or more operators of the same precedence level appear in the sub expression.
- Arithmetic expressions are evaluated from left to right using the rules of precedence.
- When Parenthesis are used, the expressions within parenthesis assume highest priority.