INTRODUCTION TO C

INFORMATION SYSTEM and IT

- Information System has six parts :

1. People
2. Procedure
3. Software (or Program)
4. Hardware
5. Data
6. Connectivity

- Information technology (IT) is the use of computers to create, process, store, retrieve, and exchange all kinds of data and information.

PROGRAMMING LANGUAGES

- A program or software means step-by-step instructions that tell computer how to work.
- The purpose of software is to convert ***data*** into ***information***.
- A programming language is required to create :

1. Software Systems
   - Operating systems
   - Utilities (Antivirus, Uninstall, Backup, File Compression, etc)
   - Device Drivers (allows communication between device and OS)
2. Applications
   - Browsers, Word Processor, Spreadsheets, Database, Multimedia software, Presentation software, etc.

TYPES OF PROGRAMMING LANGUAGES

- LOW LEVEL Programming Languages
  - Machine Language
  - Assembly Language
- MIDDLE LEVEL Programming Languages
- HIGH LEVEL Programming Languages

HIGH LEVEL PROGRAMMING LANGUAGES
Procedural – C, FORTRAN, BASIC, COBOL, PASCAL
Scripting – JavaScript, VBScript, PHP, Python
Functional – LISP
Object Oriented – C++, Java, C#, VB.NET
Object Based – VB6

‘C’ HISTORY

- Initially developed by Denis Ritchie using UNIX system at Bell Research Lab in 1972.
- A modified standard was defined by Brian Kernighan and Denis Ritchie in 1978.
- ANSI C is standard for C defined by American National Standards Institute (ANSI).

‘C’ Language Features

- Middle Level Language
- Efficient
- Robust
- Portable
- Easy to Use

Embedded C

- Extension to C language.
- Used to develop micro-controller based applications.
- C is a compiled language so the machine directly translates the code.

C PROGRAM STRUCTURE

DOCUMENTATION SECTION
LINK SECTION
DEFINITION SECTION
GLOBAL DATA DECLARATIONS
THE main() function **{** **Declaration Part** **Executable Part** **}**
USER DEFINED FUNCTION DEFINITIONS

A Simple Program

- Simple C program to Display “Hello World”

```
#include<stdio.h>
int main()
{
    printf(“Hello World”);
    return 0;
}
```

C TOKENS

- The C Compiler breaks a program into the smallest possible units and proceeds to the various stages of the compilation, which is called a token.
- Each word and punctuation mark in a C program is a token.
- Separated by white spaces, horizontal tab, vertical tab or newline.
- C tokens are the smallest building block or smallest unit of a C program.

TYPES OF TOKENS

1. **IDENTIFIERS**
   - Name given to variable, function, array, etc.
2. **KEYWORDS**
3. **CONSTANTS or LITERALS**
   - Value assigned to memory location
4. **OPERATORS**
5. **SPECIAL SYMBOLS**
   - Characters other than operator like (,),<,>,{,},[,],semi-colon, #, etc.
6. **STRING LITERAL**
   - Series of characters in double quote(“ “)

IDENTIFIERS

- User Defined Words.
- Case Sensitive.
- Should Be alphanumeric starting with character like letters [A-Z, a-z] or underscore.
- Must not contain white space or a special character.
- Identifiers are names given to variables, functions, array, structure, union, etc.
- Examples - Rate_of_interest, rate1, a123, _num, num_1, _num_1, PI, Sum, add_num.

KEYWORDS

- Certain words are reserved and are called keywords.
- Built in Words whose meanings are already defined.
- Cannot be used as an identifier name.

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

CONSTANTS

- Constant means value assigned to a memory location or the data stored.

- **Integer Constants** (Decimal, Octal, Hexadecimal)
  - 123, -3100, 0170, 0x2A, 5678U, 7865432L, 120US, -100s, 0x34ADL, 543987654UL, 0X56A

- **Real or Floating-point Constants**
  - 0.246, 975.64, -.54, 5. , 0.23178e3 ($0.23178*10^3$), 0.000045e-5.

- **Character Constants**
  - Examples - ‘a’, ‘#’, ‘(’, ‘2’
  - Backslash Character Constants (Escape Sequences)

DATATYPES

- The C language has a rich set of data types to handle different kind of data entered by programmer.
- A data type informs compiler about
  - Memory required to store input or data.
  - Type of input or data
  - Range of data
- Data types are used extensively for declaring variables and functions of different types.
- Format Specifiers define the output format for display or to read from user.

MODIFIERS

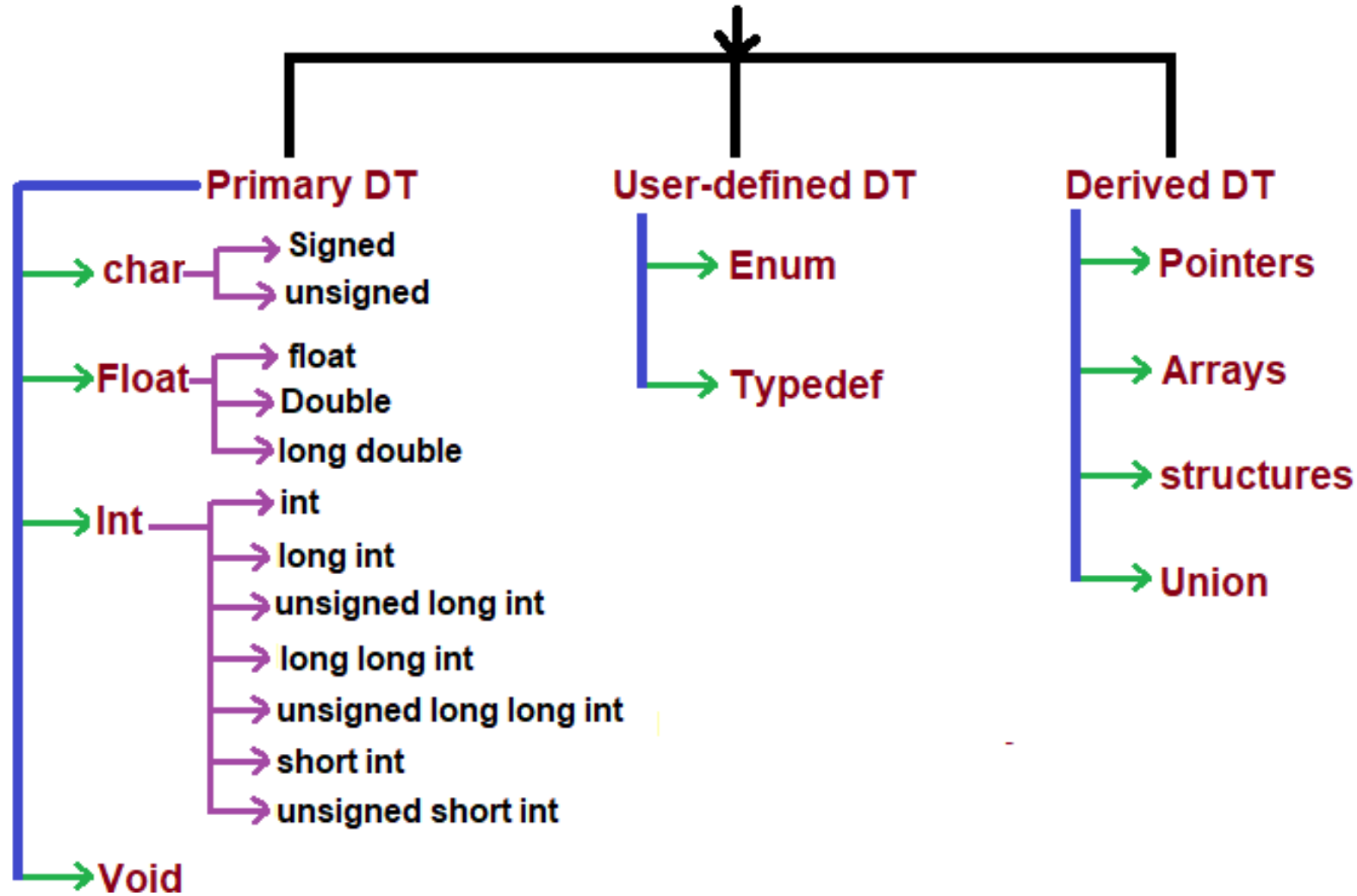
- Modifiers, Qualifiers and Storage Class are prefixed or postfixed after datatype.

<table>
<tr><th colspan="2">MODIFIERS</th><th>QUALIFIERS</th><th>STORAGE CLASS</th></tr>
<tr><td rowspan="2">SIZE MODIFIER</td><td rowspan="2">SIGN MODIFIER</td><td rowspan="2">const</td><td>auto</td></tr>
<tr><td>static</td></tr>
<tr><td rowspan="2">short
long
long long</td><td rowspan="2">signed
unsigned</td><td rowspan="2">volatile</td><td>extern</td></tr>
<tr><td>register</td></tr>
</table>

- ***short*** , ***long*** and ***long long*** cannot be applied to ***float*** and ***char.***
- ***signed*** and ***unsigned*** cannot be applied to ***float***, ***double*** and ***long double***.

DT - Data type

Data Types in C



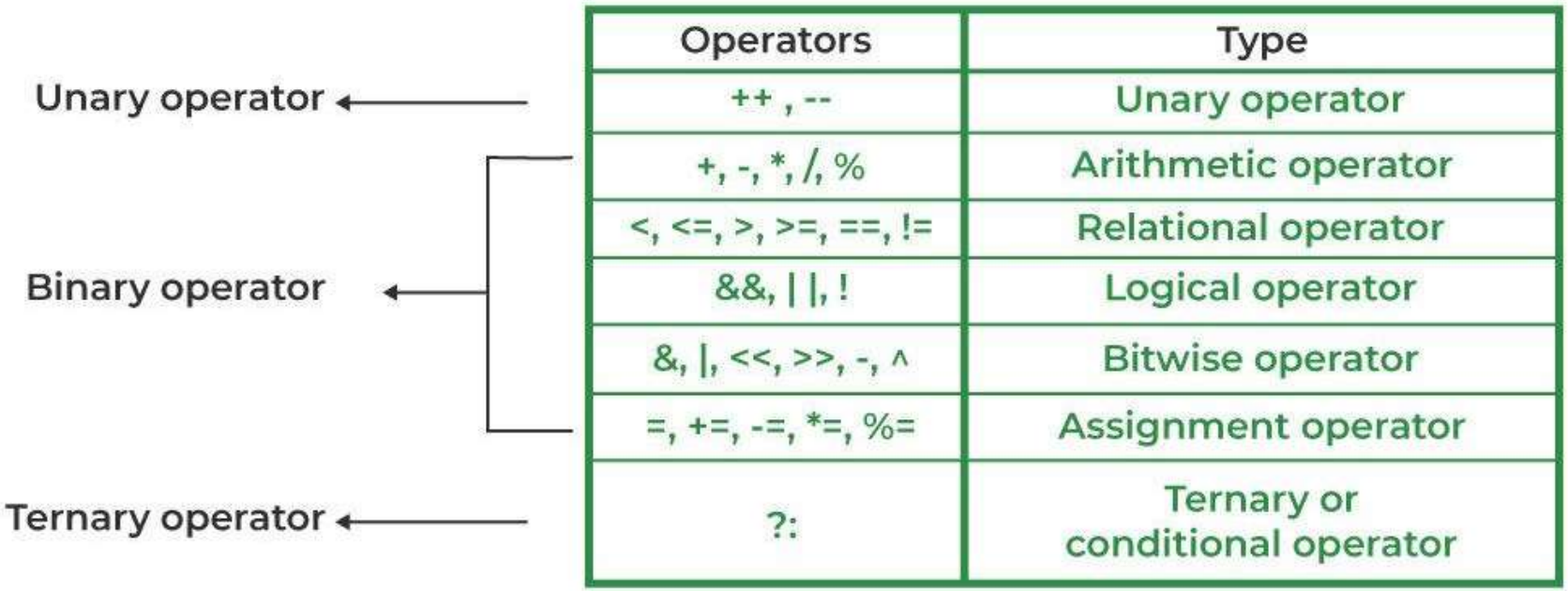
Data Type	Memory (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38 (6 decimal places)	%f
double	8	2.3E-308 to 1.7E+308 (15 decimal places)	%lf
long double	12	3.4E-4932 to 1.1E+4932 (19 decimal places)	%Lf

Data Types with Format Specifiers

DATA TYPE	'C' KEYWORD	EXAMPLE	NO. OF BYTES	FORMAT SPECIFIER
Character	***char***	Storing 'y' or'n', name of person	1	***%c***
Integer	***int***	Roll number, serial number	4	***%d***
Single-precision	***float***	Temperature, salary	4	***%f***
Double-precision	***double***	Store numbers with more precision	8	***%lf***
NO DATA	***void***	Used in functions and pointers	--	**--**
Character Array (STRING)	***char***	Series of characters	Depends on size of array	***%s***

OPERATORS

- Operators Operate on operands and returns a value



Operators	Type
++ , --	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, | |, !	Logical operator
&, |, <<, >>, -, ^	Bitwise operator
=, +=, -=, *=, %=	Assignment operator
?:	Ternary or conditional operator

Precedence, Associativity and Arity

- The order of evaluation of operators is **precedence**.
- The order of operations in same precedence group is called **associativity**.
- The number of operands that are required for an operator to operate is called **arity** of the operator.

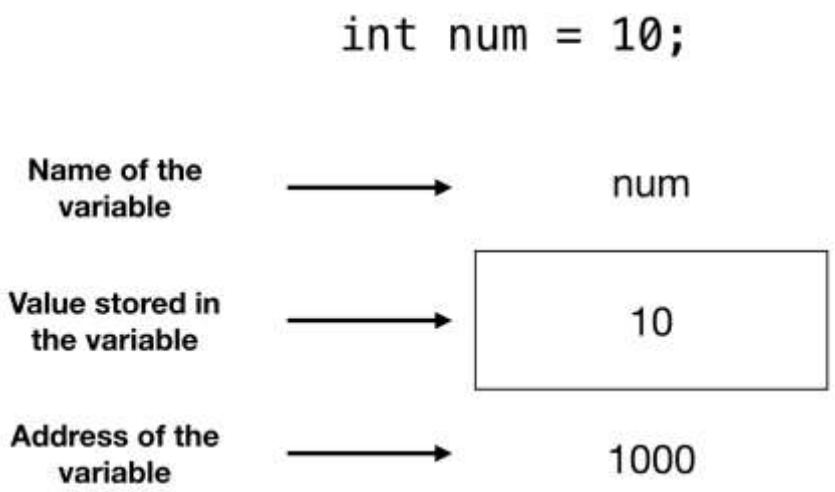
OPERATOR	TYPE	ASSOCIAVITY
( ) [ ] . ->		left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
|	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
||	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operator	right-to-left
,	Comma	left-to-right

ASSIGNMENT(=) OPERATOR

- The symbol ’=‘, known as the assignment operator, requires two operands. Value on
- R.H.S. is assigned to L.H.S.
- It is used to assign an initial value (of the declared type) to the variable.
- Once a variable is declared, you can assign and re-assign a value to a variable, using the assignment operator "=".

VARIABLES

- Variable is a name given to a memory location.
- The length of variable name can be upto 8 characters.
- Hold values of different types.
- Variables must be declared and initialized or defined before using it.



DECLARING VARIABLES

- The assignment operator(**=**), is used to assign a value to the variable.
- **int sum;** // Terminate the statement with a semi-colon.
- **int number1, number2;** // separated by a comma.
- **int height = 20;** // Assigned an initial value.

- Once a variable is declared, you can assign and re-assign a value to a variable, via the assignment operator "=".
- **int num;** // Declare a variable named "num" of the type "int"
- **num = 99;** // Assign an integer value of 99 to variable "num"
- **num = 88;** // Re-assign a value of 88 to "num"
- **num = num + 1;** // Evaluate "num + 1", and assign the result back to "num"

C EXPRESSION

- An expression in C is defined as 2 or more operands are connected by one operator.
- An operand can be a variable, an array element or any constant.
- There are 4 types of expressions:

1. Arithmetic expressions
2. Relational expressions
3. Logical expressions
4. Conditional expressions

- The result of this expression operation produces a specific value.

C INSTRUCTIONS

- A program is nothing but set of instructions.
- TYPES OF INSTRUCTIONS:

1. DECLARATION INSTRUCTION – Used to declare type of variable
2. ARITHMETIC INSTRUCTION – Used to perform arithmetic operations on constants and variables.
3. CONTROL INSTRUCTION – Used to control the sequence of execution of various statements.

Console I/O Functions

- Built-in functions for user interaction.

1. **Formatted I/O functions**

- ***int printf("control string", arg1, arg2,…,argn);***
  - It returns the number of characters printed.
- ***int scanf("control string", arg1, arg2,…,argn);***
  - It returns the number of inputs successfully matched and assigned.
- ***sprintf()***
- ***sscanf()***

2. **Unformatted I/O functions**

- ***getchar(), putchar(), gets(), puts().***

A Simple C Program

Program to display addition of two integers

```
#include<stdio.h>
int main()
{
    int num1=10, num2=20, res;
    res = num1 + num2;
    printf("Addition is %d", res);
    return 0;
}
```