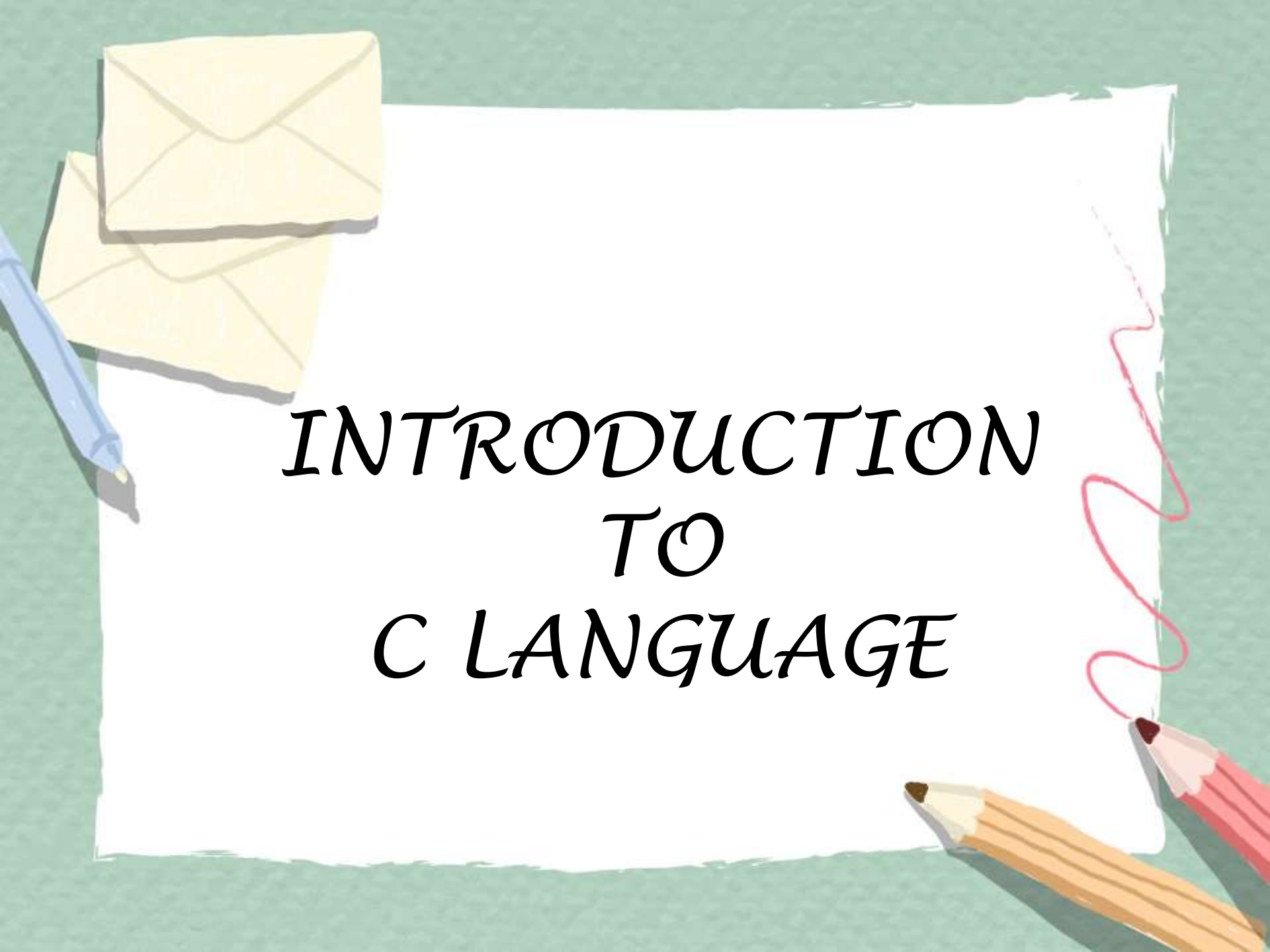


PROGRAMMING PRINCIPLE & ALGORITHM

UNIT – 1

Prepared by : Sonakshi Dhiman



INTRODUCTION TO C LANGUAGE

HISTORY OF C LANGUAGE

The C programming language is a widely-used and influential language in the world of computer programming. It is structure oriented.

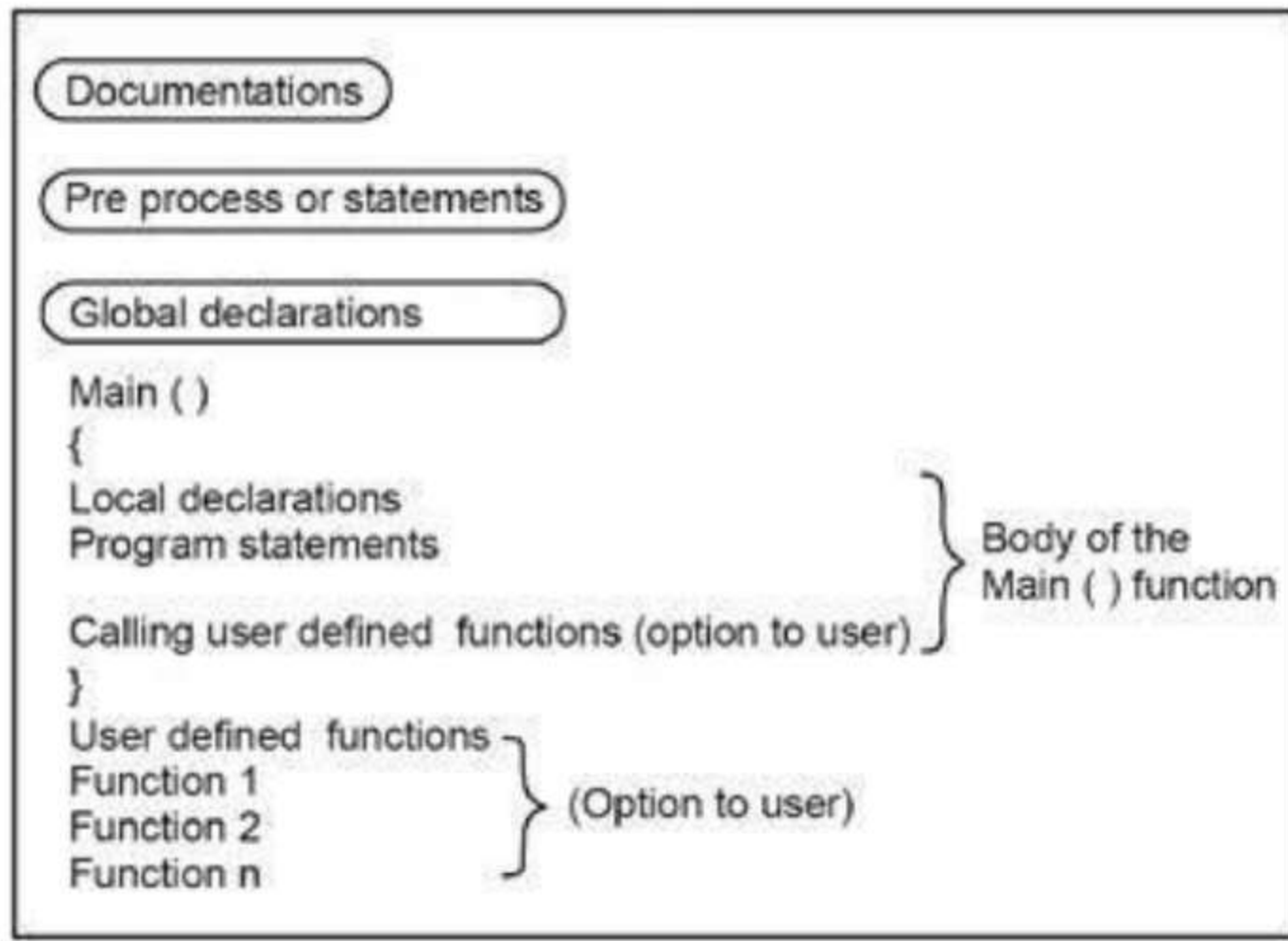
Initially C was developed to be used in UNIX operating system, but nowadays it's compatibility has grown a lot ranging from windows, Mac, Linux, Ubuntu and others. Many of the important working principle of C comes from BCPL, developed by Martin Richards, although B,BCLP are typeless languages C provides a number of different data types.

C Programming History

- In 1960, ALGOL was developed by the international group.
- In 1966, **Martin Richards** developed BCPL (Basic Combined Programming Language) before developing C.
- Later, **Ken Thomson** further developed BCPL by developing the first letter of BCPL i.e. he developed the B language.
- Ken Thomson developed BASIC type of UNIX Operating System, while the B language was in motion.
- After all this, by further developing the B language, Dennis Ritchie developed the C language in the Bell laboratory.
- After developing the C language, Dennis Ritchie and Ken Thompson together developed the UNIX Operating System.
- In 1978, **Kerni Ghan** and **Ritchie** both did further improvement in C language and they brought the C programming language in front of the whole world.
- In 1989, **ANSI C** developed by the **ANSI Committee**.
- In 1999, The new version of C was developed, In which have new features like Data Types- char, float, int, double was used.
- Today's we are using the new version of C is **C11**, which is developed by **Standard Committee**.

STRUCTURE OF A C PROGRAM

The structure of a C program is a protocol (rules) to the programmer, which he has to follow while writing a C program. The general basic structure of C program is shown in the figure below.



Based on this structure, we can sketch a C program.

Example:

```
/* This program accepts a number & displays it to the user*/  
  
#include <stdio.h>  
  
void main(void)  
{ int number;  
 printf( "Please enter a number: " );  
 scanf( "%d", &number );  
 printf( "You entered %d", number );  
 return 0;}
```

Stepwise explanation:

#include

- The part of the compiler which actually gets your program from the source file is called the preprocessor.
 - *#include <stdio.h>*
- *#include* is a pre-processor directive. It is not really part of our program, but instead it is an instruction to the compiler to make it do something. It tells the C compiler to include the contents of a file (in this case the system file called `stdio.h`).
- The compiler knows it is a system file, and therefore must be looked for in a special place, by the fact that the filename is enclosed in `<>` characters

<stdio.h>

- *stdio.h* is the name of the standard library definition file for all STanDard Input and Output functions.
- Your program will almost certainly want to send information to the screen and read things from the keyboard, and *stdio.h* is the name of the file in which the functions that we want to use are defined.
- The function we want to use is called *printf*. The actual code of *printf* will be tied in later by the linker.
- The ".h" portion of the filename is the language extension, which denotes an include file.

void

- This literally means that this means nothing. In this case, it is referring to the function whose name follows.
- Void tells to C compiler that a given entity has no meaning, and produces no error.

main

- In this particular example, the only function in the program is called *main*.
- A C program is typically made up of large number of functions. Each of these is given a name by the programmer and they refer to each other as the program runs.
- C regards the name *main* as a special case and will run this function first i.e. the program execution starts from *main*.

(void)

- This is a pair of brackets enclosing the keyword *void*.
- It tells the compiler that the function *main* has no parameters.
- A parameter to a function gives the function something to work on.

{ (Brace)

- This is a brace (or curly bracket). As the name implies, braces come in packs of two - for every open brace there must be a matching close one.
- Braces allow us to group pieces of program together, often called a block.
- A block can contain the declaration of variable used within it, followed by a sequence of program statements.
- In this case the braces enclose the working parts of the function main.

; (semicolon)

- The semicolon marks the end of the list of variable names, and also the end of that declaration statement.
- All statements in C programs are separated by ";" (semicolon) characters.
- The ";" character is actually very important. It tells the compiler where a given statement ends.
- If the compiler does not find one of these characters where it expects to see one, then it will produce an error.

scanf

- In other programming languages, the printing and reading functions are a part of the language.
- In C this is not the case; instead they are defined as standard functions which are part of the language specification, but are not a part of the language itself.
- The standard input/output library contains a number of functions for formatted data transfer; the two we are going to use are `scanf` (scan formatted) and `printf` (print formatted).

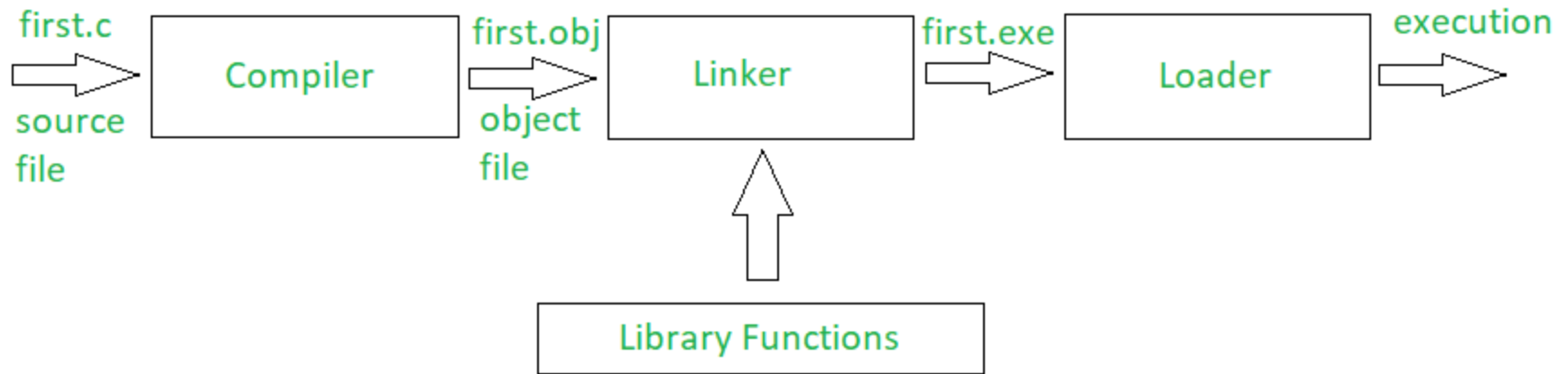
printf

- The `printf` function is the opposite of `scanf`.
- It takes text and values from within the program and sends it out onto the screen.
- Just like `scanf`, it is common to all versions of C and just like `scanf`, it is described in the system file `stdio.h`.
- The first parameter to a `printf` is the format string, which contains text, value descriptions and formatting instructions.

FILES USED IN A C PROGRAM

- **Source File-** This file contains the source code of the program. The file extension of any c file is **.c**. The file contains C source code that defines the *main* function & maybe other functions.
- **Header File-** A header file is a file with extension **.h** which contains the C function declarations and macro definitions and to be shared between several source files.
- **Object File-** An object file is a file containing object code, with an extension **.o**, meaning relocatable format machine code that is usually not directly executable. Object files are produced by an assembler, compiler, or other language translator, and used as input to the linker, which in turn typically generates an executable or library by combining parts of object files.
- **Executable File-** The binary executable file is generated by the linker. The linker links the various object files to produce a binary file that can be directly executed.

COMPILATION & EXECUTION OF A C PROGRAM



FUNCTIONS AS BUILDING BLOCKS

C Programming is known as **procedure oriented programming language**.

It contains different procedures or functions which accomplish some specified task.

Any C program must contain a function named **main()**, which actually acts as an entry point or it is the first program to be executed. Functions can share and exchange some data. Functions should be made to achieve one specific task.

Functions in C are the basic building blocks of a C program. A function is a set of statements enclosed within curly brackets (`{}`) that take inputs, do the computation, and provide the resultant output. You can call a function multiple times, thereby allowing reusability and modularity in C programming. It means that instead of writing the same code again and again for different arguments, you can simply enclose the code and make it a function and then call it multiple times by merely passing the various arguments.

An illustration featuring a green textured background. In the top left, two yellow envelopes are stacked, with a blue pen lying next to them. In the bottom right, two pencils (one orange, one red) are shown, with a red squiggly line drawn near the red pencil. A white rectangular area with a torn edge is in the center, containing the text.

LANGUAGE FUNDAMENTALS

C Character Set

As every language contains a set of characters used to construct words, statements, etc., C language also has a set of characters which include **alphabets**, **digits**, and **special symbols**. C language supports a total of 256 characters.

Every C program contains statements. These statements are constructed using words and these words are constructed using characters from C character set. C language character set contains the following set of characters...

- 1.Alphabets
- 2.Digits
- 3.Special Symbols

Alphabets

C language supports all the alphabets from the English language. Lower and upper case letters together support 52 alphabets.

lower case letters - a to z

UPPER CASE LETTERS - A to Z

Digits

C language supports 10 digits which are used to construct numerical values in C language.

Digits - **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**

Special Symbols

C language supports a rich set of special symbols that include symbols to perform mathematical operations, to check conditions, white spaces, backspaces, and other special symbols.

Special Symbols - **~ @ # \$ % ^ & * () _ - + = { } [] ; : ' " / ? . > , < \ | tab newline space NULL bell backspace vertical tab etc.,**

C Tokens

Every C program is a collection of instructions and every instruction is a collection of some individual units. Every smallest individual unit of a C program is called token. Every instruction in a C program is a collection of tokens. Tokens are used to construct C programs and they are said to be the basic building blocks of a C program.

In a C program tokens may contain the following...

- 1.Keywords
- 2.Identifiers
- 3.Operators
- 4.Special Symbols
- 5.Constants
- 6.Strings

C Keywords

As every language has words to construct statements, C programming also has words with a specific meaning which are used to construct C program instructions. In the C programming language, keywords are special words with predefined meaning. Keywords are also known as reserved words in C programming language.

In the C programming language, there are **32 keywords**. All the 32 keywords have their meaning which is already known to the compiler.

Whenever C compiler come across a keyword, automatically it understands its meaning.

Properties of Keywords

- 1.All the keywords in C programming language are defined as lowercase letters so they must be used only in lowercase letters
- 2.Every keyword has a specific meaning, users can not change that meaning.
- 3.Keywords can not be used as user-defined names like variable, functions, arrays, pointers, etc...
- 4.Every keyword in C programming language represents something or specifies some kind of action to be performed by the compiler.

32 Keywords in C Programming Language with their Meaning

S.No	Keyword	Meaning
1	auto	Used to represent automatic storage class
2	break	Unconditional control statement used to terminate swith & looping statements
3	case	Used to represent a case (option) in switch statement
4	char	Used to represent character data type
5	const	Used to define a constant
6	continue	Unconditional control statement used to pass the control to the begining of looping statements
7	default	Used to represent a default case (option) in switch statement
8	do	Used to define do block in do-while statement
9	double	Used to present double datatype
10	else	Used to define FALSE block of if statement
11	enum	Used to define enumerated datatypes
12	extern	Used to represent external storage class
13	float	Used to represent floating point datatype
14	for	Used to define a looping statement
15	goto	Used to represent unconditional control statement
16	if	Used to define a conditional control statement

17	int	Used to represent integer datatype
18	long	It is a type modifier that alters the basic datatype
19	register	Used to represent register storage class
20	return	Used to terminate a function execution
21	short	It is a type modifier that alters the basic datatype
22	signed	It is a type modifier that alters the basic datatype
23	sizeof	It is an operator that gives size of the memory of a variable
24	static	Used to create static variables - constants
25	struct	Used to create structures - Userdefined datatypes
26	switch	Used to define switch - case statement
27	typedef	Used to specify temporary name for the datatypes
28	union	Used to create union for grouping different types under a name
29	unsigned	It is a type modifier that alters the basic datatype
30	void	Used to indicate nothing - return value, parameter of a function
31	volatile	Used to creating volatile objects
32	while	Used to define a looping statement

- All the keywords are in lowercase letters
- Keywords can't be used as userdefined name like variable name, function name, lable, etc...
- Keywords are also called as Reserved Words

C Identifiers

In C programming language, programmers can specify their name to a variable, array, pointer, function, etc... An identifier is a collection of characters which acts as the name of variable, function, array, pointer, structure, etc... In other words, an identifier can be defined as the user-defined name to identify an entity uniquely in the c programming language that name may be of the variable name, function name, array name, pointer name, structure name or a label.

The identifier is a user-defined name of an entity to identify it uniquely during the program execution

Example

```
int marks;  
char studentName[30];
```

Here, **marks** and **studentName** are identifiers.

Rules for Creating Identifiers

1. An identifier can contain **letters** (UPPERCASE and lowercase), **numerics** & **underscore** symbol only.
2. An identifier should not start with a numerical value. It can start with a letter or an underscore.
3. We should not use any special symbols in between the identifier even whitespace. However, the only underscore symbol is allowed.
4. Keywords should not be used as identifiers.
5. There is no limit for the length of an identifier. However, the compiler considers the first 31 characters only.
6. An identifier must be unique in its scope.

Rules for Creating Identifiers for better programming

The following are the commonly used rules for creating identifiers for better programming...

- 1.The identifier must be meaningful to describe the entity.
- 2.Since starting with an underscore may create conflict with system names, so we avoid starting an identifier with an underscore.
- 3.We start every identifier with a lowercase letter. If an identifier contains more than one word then the first word starts with a lowercase letter and second word onwards first letter is used as an UPPERCASE letter. We can also use an underscore to separate multiple words in an identifier.

C Variables

Variables in a c programming language are the named memory locations where the user can store different values of the same datatype during the program execution. That means a variable is a name given to a memory location in which we can store different values of the same data type. In other words, a variable can be defined as a storage container to hold values of the same datatype during the program execution. The formal definition of a data type is as follows...

Variable is a name given to a memory location where we can store different values of the same datatype during the program execution.

A variable name may contain letters, digits and underscore symbol. The following are the rules to specify a variable name...

1. Variable name should not start with a digit.
2. Keywords should not be used as variable names.
3. A variable name should not contain any special symbols except underscore(_).
4. A variable name can be of any length but compiler considers only the first 31 characters of the variable name.

Declaration of Variable

Declaration of a variable tells the compiler to allocate the required amount of memory with the specified variable name and allows only specified datatype values into that memory location. In C programming language, the declaration can be performed either before the function as global variables or inside any block or function. But it must be at the beginning of block or function.

Declaration Syntax :

datatype variable name;

Example :

int number;

The above declaration tells to the compiler that allocates **2 bytes** of memory with the name **number** and allows only integer values into that memory location.

C Constants

In C programming language, a constant is similar to the variable but the constant hold only one value during the program execution. That means, once a value is assigned to the constant, that value can't be changed during the program execution. Once the value is assigned to the constant, it is fixed throughout the program. A constant can be defined as follows...

A constant is a named memory location which holds only one value throughout the program execution.

In C programming language, a constant can be of any data type like integer, floating-point, character, string and double, etc.,

Integer constants

An integer constant can be a decimal integer or octal integer or hexadecimal integer. A decimal integer value is specified as direct integer value whereas octal integer value is prefixed with 'o' and hexadecimal value is prefixed with 'OX'.

An integer constant can also be unsigned type of integer constant or long type of integer constant. Unsigned integer constant value is suffixed with 'u' and long integer constant value is suffixed with 'l' whereas unsigned long integer constant value is suffixed with 'ul'.

Example

125 -----> Decimal Integer Constant

O76 -----> Octal Integer Constant

OX3A -----> Hexa Decimal Integer Constant

50u -----> Unsigned Integer Constant

30l -----> Long Integer Constant

100ul -----> Unsigned Long Integer Constant

Floating Point constants

A floating-point constant must contain both integer and decimal parts. Some times it may also contain the exponent part. When a floating-point constant is represented in exponent form, the value must be suffixed with 'e' or 'E'.

Example

The floating-point value **3.14** is represented as **3E-14** in exponent form.

Character Constants

A character constant is a symbol enclosed in single quotation. A character constant has a maximum length of one character.

Example

'A' , '2' , '+'

In the C programming language, there are some predefined character constants called escape sequences. Every escape sequence has its own special functionality and every escape sequence is prefixed with '\' symbol. These escape sequences are used in output function called 'printf()'.

String Constants

A string constant is a collection of characters, digits, special symbols and escape sequences that are enclosed in double quotations.

We define string constant in a single line as follows...

"This is a smart class"

C data types

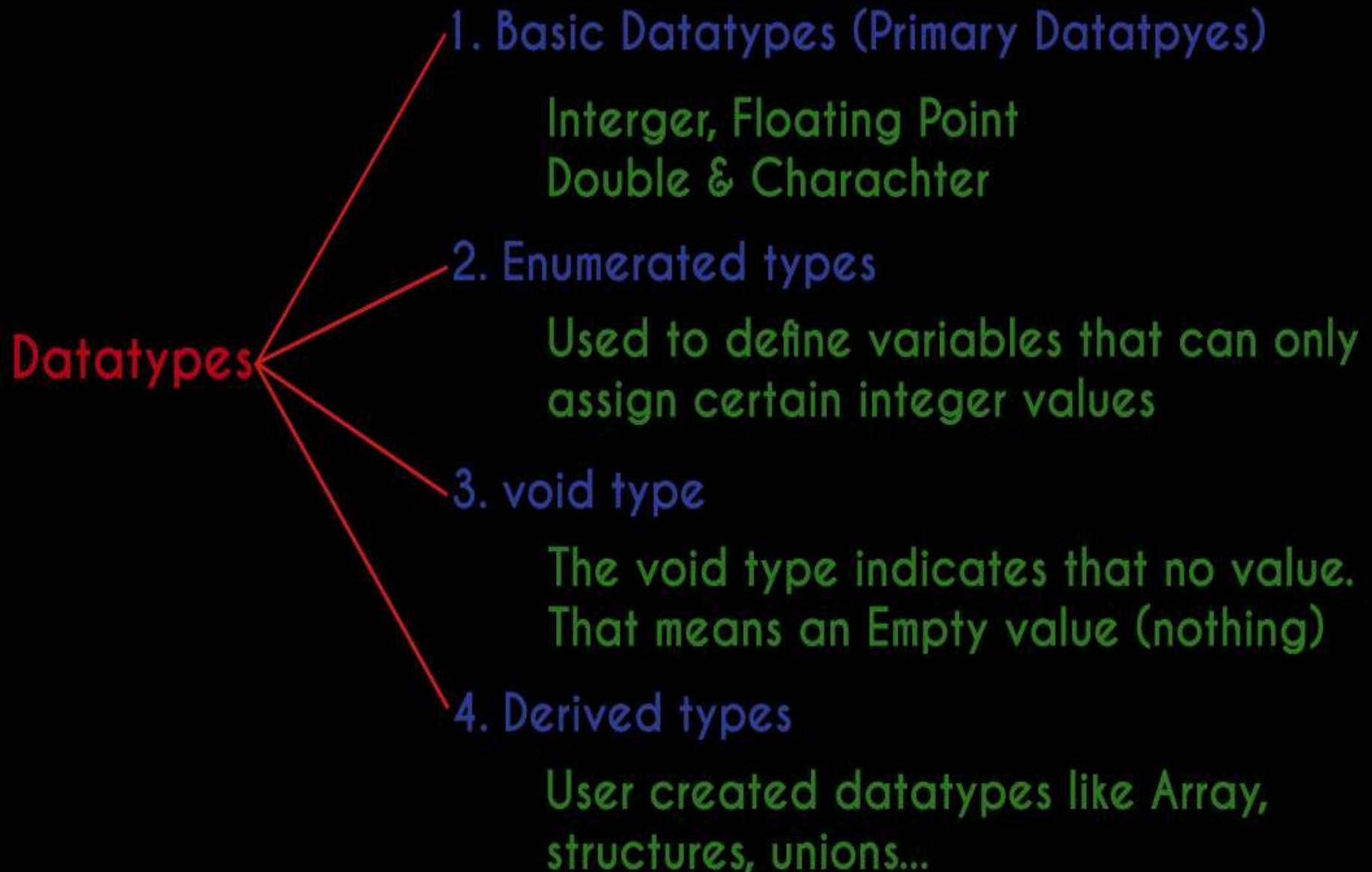
Data used in c program is classified into different types based on its properties. In the C programming language, a data type can be defined as a set of values with similar characteristics. All the values in a data type have the same properties.

Data types in the c programming language are used to specify what kind of value can be stored in a variable. The memory size and type of the value of a variable are determined by the variable data type. In a c program, each variable or constant or array must have a data type and this data type specifies how much memory is to be allocated and what type of values are to be stored in that variable or constant or array. The formal definition of a data type is as follows...

The Data type is a set of value with predefined characteristics. data types are used to declare variable, constants, arrays, pointers, and functions.

In the c programming language, data types are classified as follows...

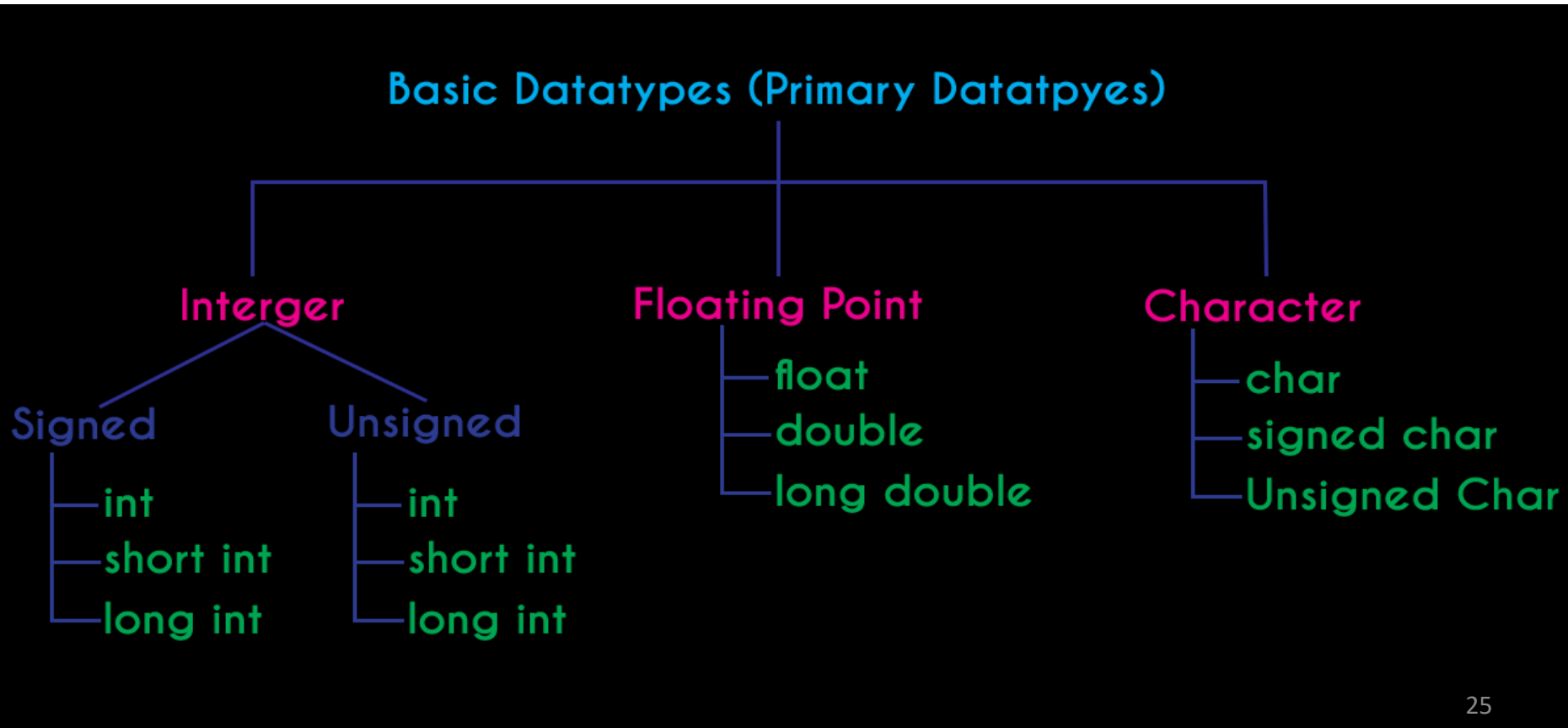
- 1.Primary data types (Basic data types OR Predefined data types)
- 2.Derived data types (Secondary data types OR User-defined data types)
- 3.Enumeration data types
- 4.Void data type



Primary data types

All the primary data types are already defined in the system. Primary data types are also called as Built-In data types. The following are the primary data types in C programming language...

- 1.Integer data type
- 2.Floating Point data type
- 3.Double data type
- 4.Character data type



Integer Data type

The integer data type is a set of whole numbers. Every integer value does not have the decimal value. We use the keyword "**int**" to represent integer data type in c. We use the keyword `int` to declare the variables and to specify the return type of a function. The following table provides complete details about the integer data type.

Type	Size (Bytes)	Range	Specifier
int (signed short int)	2	-32768 to +32767	%d
short int (signed short int)	2	-32768 to +32767	%d
long int (signed long int)	4	-2,147,483,648 to +2,147,483,647	%d
unsigned int (unsigned short int)	2	0 to 65535	%u
unsigned long int	4	0 to 4,294,967,295	%u

Floating Point data types

Floating-point data types are a set of numbers with the decimal value. Every floating-point value must contain the decimal value. The floating-point data type has two variants...

- float
- double

We use the keyword "**float**" to represent floating-point data type and "**double**" to represent double data type in c. Both float and double are similar but they differ in the number of decimal places. The float value contains 6 decimal places whereas double value contains 15 or 19 decimal places. The following table provides complete details about floating-point data types.

Type	Size (Bytes)	Range	Specifier
float	4	1.2E - 38 to 3.4E + 38	%f
double	8	2.3E-308 to 1.7E+308	%ld
long double	10	3.4E-4932 to 1.1E+4932	%ld

Character data type

The character data type is a set of characters encloseded in single quotations. The following table provides complete details about the character data type.

Type	Size (Bytes)	Range	Specifier
char (signed char)	1	-128 to +127	%c
unsigned char	1	0 to 255	%c

The following table provides complete information about all the data types in c programming language...

	Integer	Floating Point	Double	Character
What is it?	Numbers without decimal value	Numbers with decimal value	Numbers with decimal value	Any symbol enclosed in single quotation
Keyword	int	float	double	char
Memory Size	2 or 4 Bytes	4 Bytes	8 or 10 Bytes	1 Byte
Range	-32768 to +32767 (or) 0 to 65535 (Incase of 2 bytes only)	1.2E - 38 to 3.4E + 38	2.3E-308 to 1.7E+308	-128 to + 127 (or) 0 to 255
Type Specifier	%d or %i or %u	%f	%ld	%c or %s
Type Modifier	short, long signed, unsigned	No modifiers	long	signed, unsigned
Type Qualifier	const, volatile	const, volatile	const, volatil	const, volatile

void data type

The void data type means nothing or no value. Generally, the void is used to specify a function which does not return any value. We also use the void data type to specify empty parameters of a function.

Enumerated data type

An enumerated data type is a user-defined data type that consists of integer constants and each integer constant is given a name. The keyword "**enum**" is used to define the enumerated data type.

Derived data types

Derived data types are user-defined data types. The derived data types are also called as user-defined data types or secondary data types. In the c programming language, the derived data types are created using the following concepts...

- Arrays
- Structures
- Unions
- Enumeration

Concept of comments in computer programming

Comments are text notes added to the program to provide explanatory information about the source code. They are used in a programming language to document the program and remind programmers of what tricky things they just did with the code and also helps the later generation for understanding and maintenance of code. The compiler considers these as non-executable statements. Different programming language uses a different way of including the comments in the source code.

Multi line Comment

```
/* this is a comment in C.
```

```
This comment syntax is guaranteed to  
work on every compiler */
```

Single line Comment

```
// This is also a comment in C
```

```
// used for one line comment
```

UNIT - 1

COMPLETE