What is Programme ?

A computer program is a set of instructions for a computer to perform a specific task.

C Language

- C is a general-purpose language which has been closely associated with the UNIX operating system for which it was developed - since the system and most of the programs that run it are written in C.
- Many of the important ideas of C stem from the language BCPL, developed by Martin Richards.
- The influence of BCPL on C proceeded indirectly through the language B, which was written by Ken Thompson in 1970 at Bell Labs, for the first UNIX system on a DEC PDP-7. BCPL and B are "type less" languages whereas C provides a variety of data types.
- In 1972 Dennis Ritchie at Bell Labs writes C and in 1978 the publication of
- The C Programming Language by Kernighan & Ritchie caused a revolution in the computing world.
- In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or "ANSI C", was completed late 1988.

Structure of C Programme

```
#include <stdio.h>


void main()
{
   /* my first program in C */
   printf("Hello, World! \n");



}
```

Explanation

- The first line of the program #include <stdio.h> is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
- The next line void main() is the main function where the program execution begins.
- The next line /*...*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.
- The next line printf(...) is another function available in C which causes the message "Hello, World!" to be displayed on the screen.

Documentation section

Link section

Definition section

Global declaration section

main () Function section

{

Declaration part
Executable part

}

Subprogram section

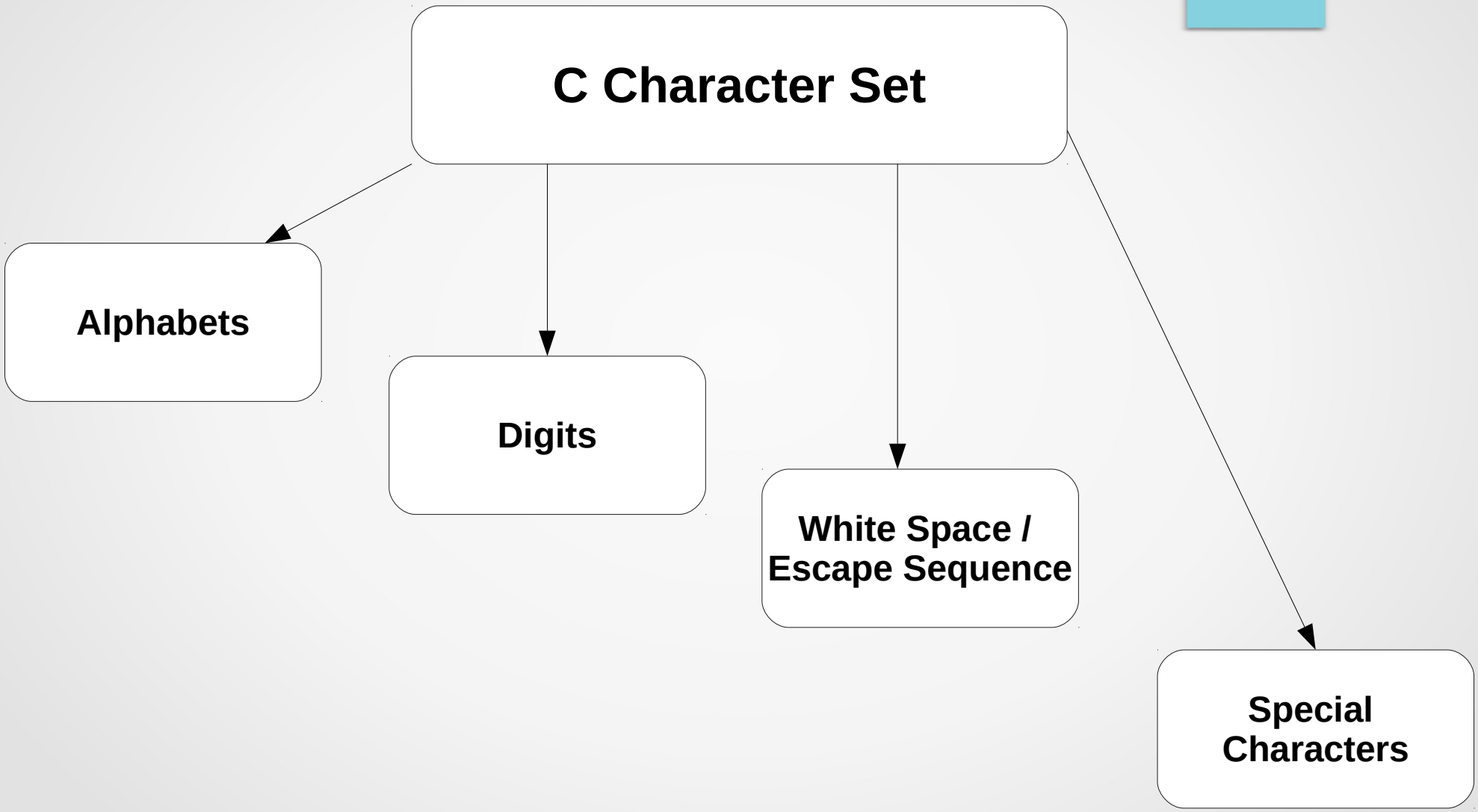
Function 1
Function 2
……
……
Function n

(User defined functions)

Compile and Execute C Program

- Open a text editor and add the above-mentioned code.
- Save the file as hello.c
- Open a command prompt and go to the directory where you have saved the file.
- Type gcc hello.c and press enter to compile your code.
- If there are no errors in your code, the command prompt will take you to the next line and would generate a.out executable file.
- Now, type ./a.out to execute your program.
- You will see the output "Hello World" printed on the screen.

UNIT -2 C Character Set



Character Set

1. Letters : a b c ..... z, A B C ..... Z
2. Digits : 0 1 2 3 4 5 6 7 8 9
3. White Spaces : Blank Space, Form Feed, Horizontal Tab, New Line, Vertical Tab
4. Special Charecters :

,	Comma	&	Ampersand
.	Period	^	Caret
;	Semicolon	*	Asterisk
:	Colon	-	Minus Sign
?	Question Mark	+	Plus Sign
'	Aphostrophe	<	Opening Angle (Less than sign)
"	Quotation Marks	>	Closing Angle (Greater than sign)
!	Exclaimation Mark	(	Left Parenthesis
|	Vertical Bar	)	Right Parenthesis
/	Slash	[	Left Bracket
\	Backslash	]	Right Bracket
~	Tilde	{	Left Brace
-	Underscore	}	Right Bracket
$	Dollar Sign	#	.Number Sign
%	Percentage Sign		

C Character Set

Sr.No	White Space & Eascape Sequences	Symbol
1.	Back Space	\b
2.	Horizontal Tab	\t
3.	Vertical Tab	\v
4.	New Line	\n
5.	Form Feed	\f
6.	Back Slash	\\

C Character Set

Sr.No	White Space & Eascape Sequences	Symbol
7.	Alert Bell	\a
8.	Carriage Return	\r
9.	Question Mark	\?
10.	Single Quote	\’
11.	Double Quote	\”
12.	Octal Number	\0
13	Hexadecimal Number	\xhh

C Character Set

Symbol	Symbol	Symbol	Symbol
`	&	/	( )
.	^	\	[ ]
;	*	~	{ }
“	+	_	%
!	<	$	#
|	>	?	= @

Comments

- Two types

  1. Single Line Comment

     // Statemments

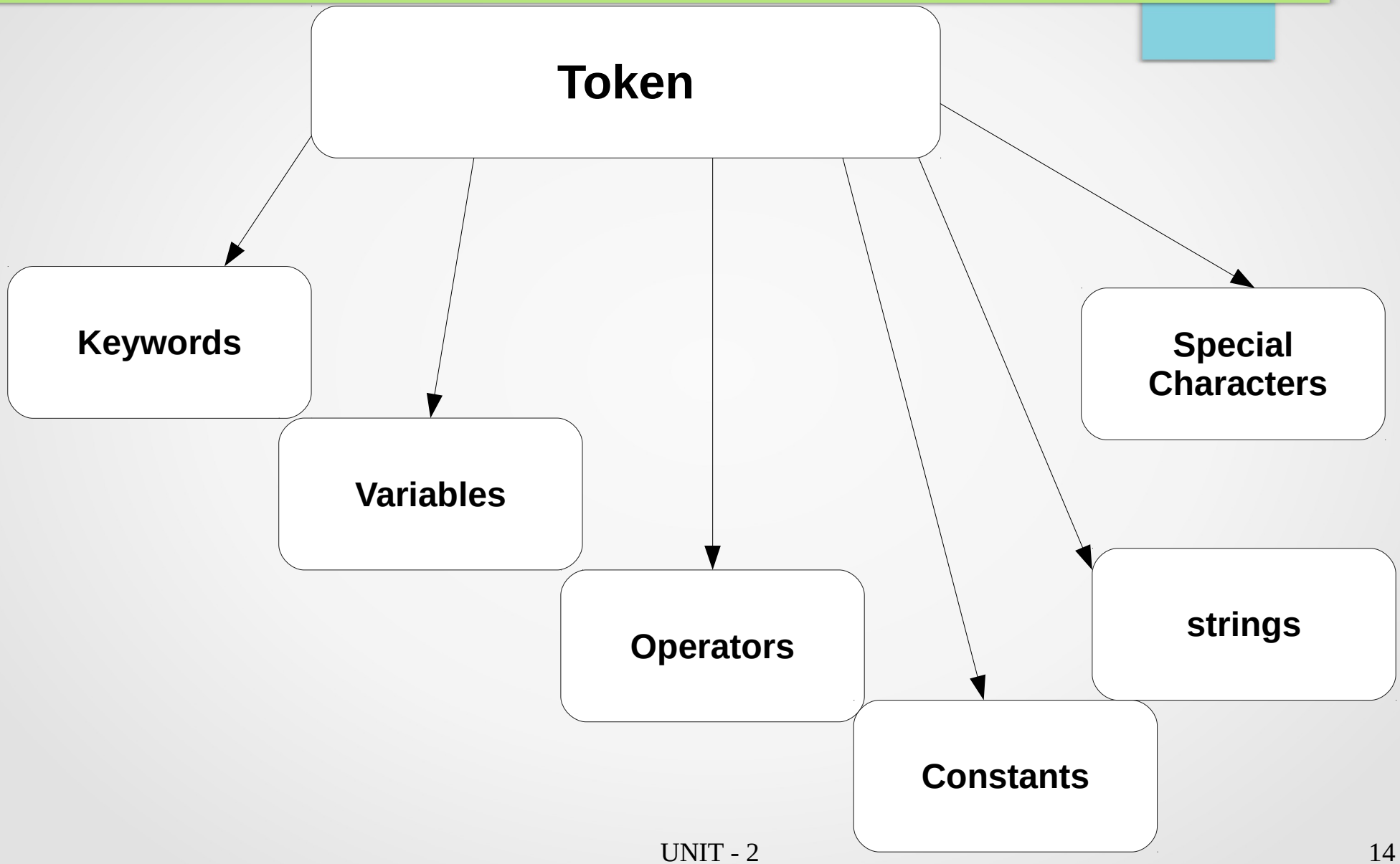
  2. Multi Line Comment

     /* Statements

     State....

     State.... */

Tokens


Token
Keywords
Variables
Operators
Constants
strings
Special Characters

Tokens

- The smallest individual unit in a c program is known as a token or a lexical unit.

- Keywords
- Identifiers
- Constants
- Strings
- Special Symbols
- Operators

Keywords

- C keywords are the words that convey a special meaning to the c compiler.
- The keywords cannot be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed.

keywords

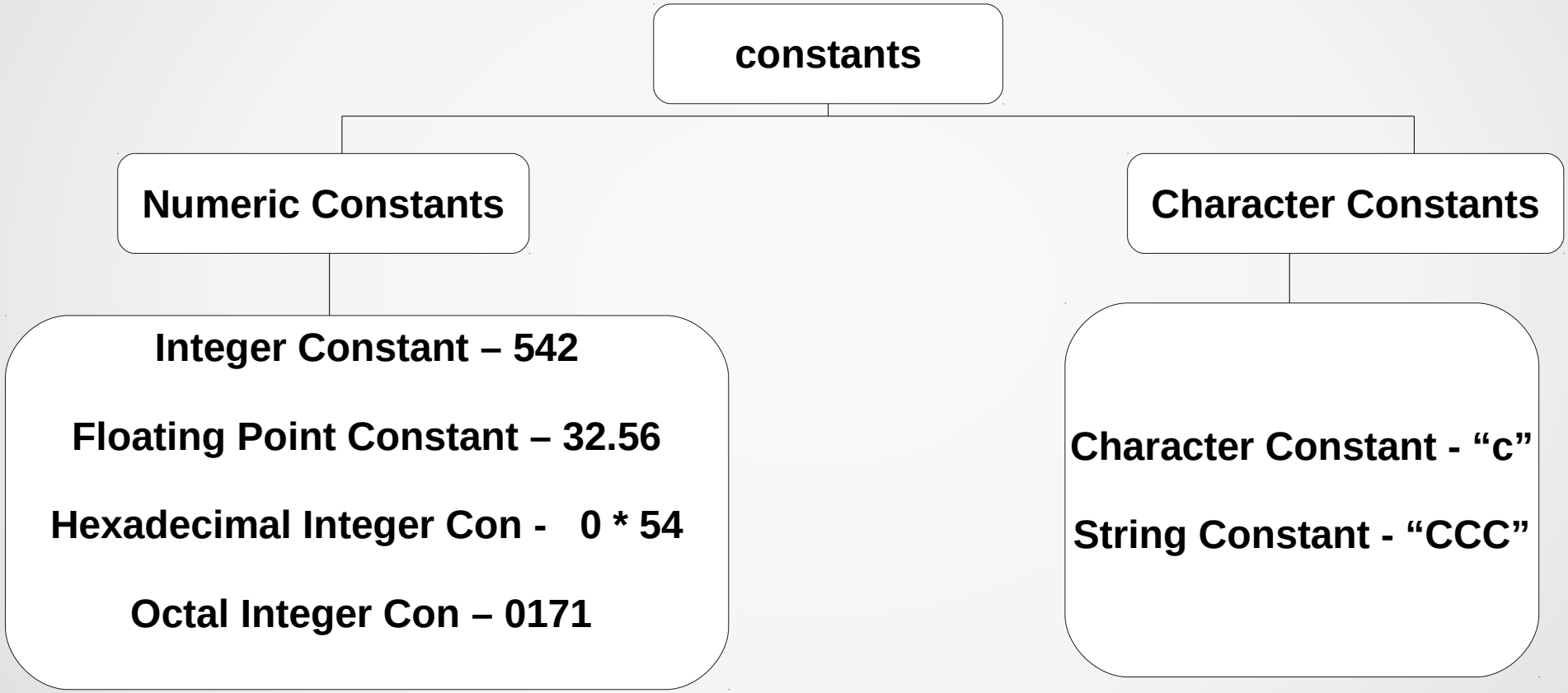
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Identifires

- Identifiers are used as the general terminology for the names of variables, functions and arrays.
- These are user defined names consisting of arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character.
- Eg: - a variable
  - Function
  - Structure
  - Array
  - Union etc....

Constants

- The constants in C, which do not change during the execution of a program.



Variables

- Variable are the basic objects manipulated in a program.
- A variable is used to store value.
- It store single value at a time.
  - Int x = 10;
  - Int a = 20;

Rules....

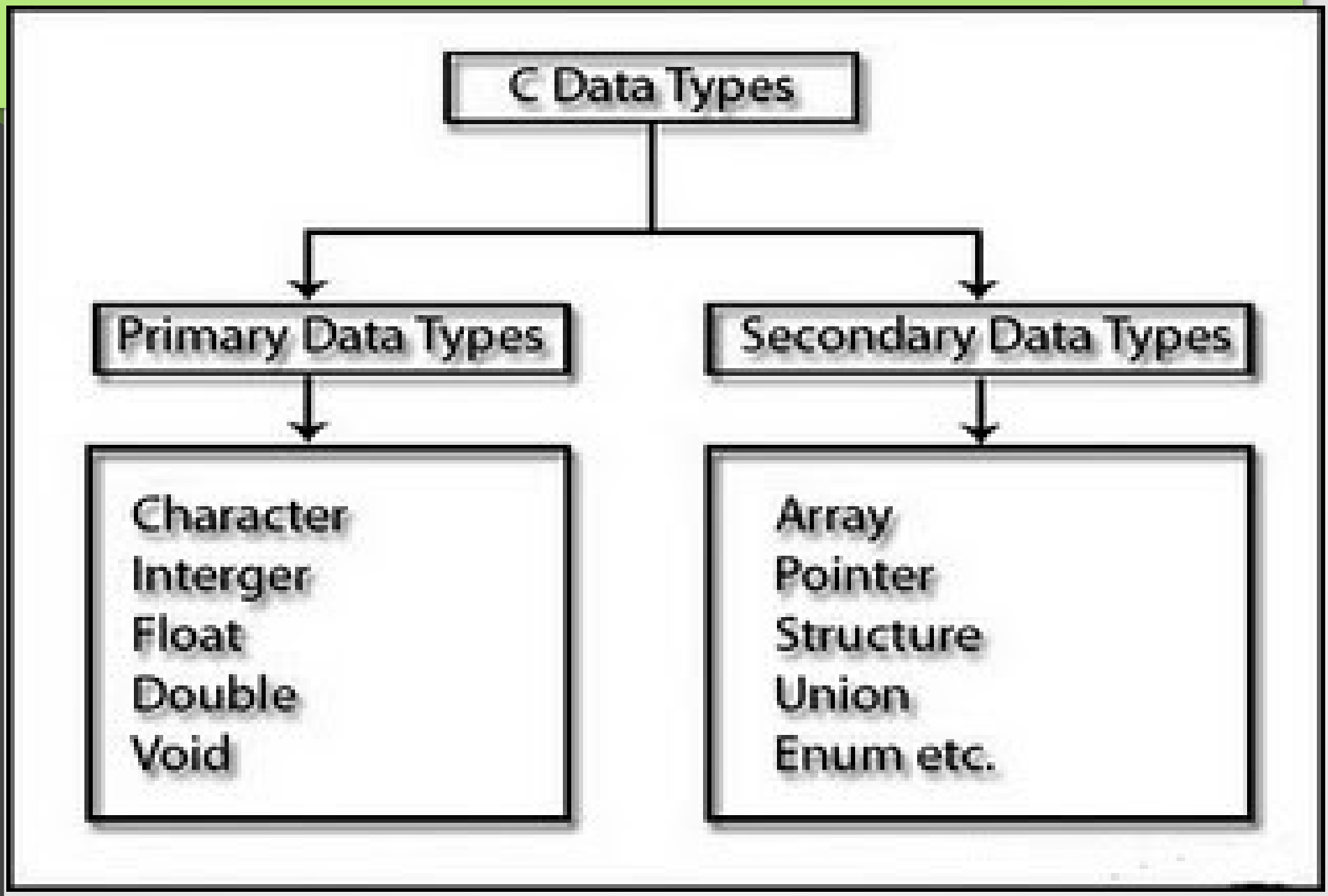
- They must begin with a letter or underscore(_).
- They must consist of only letters, digits, or underscore. No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- It should be up to 31 characters long as only first 31 characters are significant.

C Constants

- C constants refers to the data items that do not change their value during the program execution.
- Integer Constants
- Float Constant

Data Types

- Data types in c refer to an extensive system used for declaring variables or functions of different types.
- The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.


C Data Types
Primary Data Types
Secondary Data Types
Character
Interger
Float
Double
Void
Array
Pointer
Structure
Union
Enum etc.

Data type	Size(bytes)	Range	Format string
Char	1	128 to 127	%c
Unsigned char	1	0 to 255	%c
Short or int	2	-32,768 to 32,767	%i or %d
Unsigned int	2	0 to 65535	%u
Long	4	-2147483648 to 2147483647	%ld
Unsigned long	4	0 to 4294967295	%lu
Float	4	3.4 e-38 to 3.4 e+38	%f or %g
Double	8	1.7 e-308 to 1.7 e+308	%lf
Long Double	10	3.4 e-4932 to 1.1 e+4932	%lf

Datatype	Size	Format string
CHARACTER (char)	1 BYTE	%c
INTEGER (int)	2 BYTE	%d
FLOAT (float)	4 BYTE	%f
DOUBLE (double)	8 BYTE	%LF

Variable Declaration

- A variable is nothing but a name given to a storage area that our programs can manipulate.
- A variable definition tells the compiler where and how much storage to create for the variable
- **Type** **Description**

  char Typically a single octet(one byte). This is an integer type.

  int The most natural size of integer for the machine.

  float A single-precision floating point value.

  double A double-precision floating point value.

  void Represents the absence of type.

Syntax

- Datatype variablename;

  Ex:

- int i, j, k;
- char c, ch;
- float f, salary;
- double d;

Variables can be initialized

- Datatype variablename=value;

  Ex:

  int d = 3, f = 5; // declaration of d and f.
- int d = 3, f = 5; // definition and initializing d and f.
- char x = 'x'; // the variable x has the value 'x'.

Type Conversion

- Type casting is a way to convert a variable from one data type to another data type.
- For example, if you want to store a 'long' value into a simple integer then you can type cast 'long' to 'int'.
- Implicit and Explicit
- You can convert the values from one type to another explicitly using the cast operator as follows
- (type_name) expression

Ex

```
main()
 {
int sum = 17, count = 5;
double mean;

 mean = (double) sum / count;

printf("Value of mean : %f\n", mean );

}
```

Defination :

- Operator : Through which we perform some particular operation.
- Operand : On which we have to perform the operation.
- Expression : The formula that involves the operand and operator.

UNIT -2 Operators

- In order to perform different kinds of operations, C uses different operators.
- C use four classes of operators
  1) Arithmetic
  2) Relational
  3) Logical
  4) Bitwise

UNIT -2 Operators

- Types of Operators

Type of Operator	Symbolic Representation
Arithmetic Operators	+, -, *, /, %
Relation Operators	>, <, ==, <=, >=, !=
Logical Operators	&&, ||, !
Increment and Decrement Operators	++, ---

UNIT -2 Operators

- Types of Operators

Type of Operator	Symbolic Representation
Assignment Operators	=, *=, /=, %=, +=, -=, &=, ^=, |=, <<=, >>=,
Bitwise Operators	&, |, ^, >>, <<, ~
Comma Operator	,
Coditional Operator	? :

UNIT -2 Operator Precedence

- X = 5 * 4 + 8 / 2;

- X = (8 / (2 * (2 * 2)));

UNIT -2 Operator Precedence

OPERATORS	ASSOCIATIVITY	PRIORITY
(), [], ->, ->, .,	Left to right	1st
+, -, ++, --, !, ~, *, &, Sizeof, Type	Right to left	2nd
*, /, %,	Left to right	3rd
+, -	Left to right	4th

UNIT -2 Operator Precedence

OPERATORS	ASSOCIATIVITY	PRIORITY
<<, >>	Left to Right	5th
<, <=, >, >=	Left to Right	6th
==, !=	Left to Right	7th
&	Left to Right	8th

UNIT -2 Operator Precedence

OPERATORS	ASSOCIATIVITY	PRIORITY
^	Left to Right	9th
|	Left to Right	10th
&&	Left to Right	11th
||	Left to Right	12th

UNIT -2 Operator Precedence

OPERATORS	ASSOCIATIVITY	PRIORITY
? :	Right to Left	13th
=, *=, /=, %=, +=, -=, &=, ^=, |=, <<=, >>=,	Right to Left	14th
,	Right to Left	15th

UNIT -2 Arithmetic Operator

- Two types of Arithmetic Operator
  - Binary Operator
    - +, -, *, /, %
    - C = a + b;
    - D = a – b;
  - Uniary Operator
    - ++
    - --
    - &
    - Sizeof

binary_opr.c
unary_opr.c
unary_opr1.c

UNIT -2 Relational Operators

- Relational Operators
  - Less than <
  - Greater than >
  - Less than or equal to <=
  - Greater than or equal to >=
  - Equal to ==
  - Not Equal to !=

Relat_opr.c

UNIT -2 Logical Operator

Operator	Meaning of Operato	Example
&&	Logial AND. True only if all operands are true	If c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression ((c == 5) || (d > 5)) equals to 1.
!	! Logical NOT. True only if the operand is 0	If c = 5 then, expression ! (c == 5) equals to 0.

logica_opr1.c

UNIT -2 Bitwise Operator

- During computation, mathematical operations like: addition, subtraction, addition and division are converted to bit-level which makes processing faster and saves power.
- Bitwise operators are used in C programming to perform bit-level operations.

Operator	Meaning of Operato
&	Bitwise AND
|	Bitwise OR
~	One’s Complement
^	Bitwise XOR
>>	Right Shift
<<	Left Shift

UNIT -2 Bitwise Rightshift Operator

- A = 8;

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- A >>2; or a =a>>2;

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- A = 2;

bit_opr2.c

UNIT -2 Bitwise leftshift Operator

- A = 2;

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- A <<=3; or a =a<<2;

0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- A = 16;

bit_opr2.c

UNIT -2 Bitwise & Operator

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- A = 2; B = 16
- C = A & B;
- C = 0

bit_opr3.c

UNIT -2 Bitwise | Operator

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

- A = 2; B = 16
- C = A | B;
- C = 18

bit_opr3.c

UNIT -2 Bitwise ^ (XOR) Operator

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

- A = 2; B = 16
- C = A ^ B;
- C = 18

bit_opr3.c

UNIT -2 Coditional Operator

- Syntax

  conditionalExpression ? expression1 : expression2

  - If conditionalExpression is true, expression1 is evaluated.
  - If conditionalExpression is false, expression2 is evaluated.

con_opr1.c

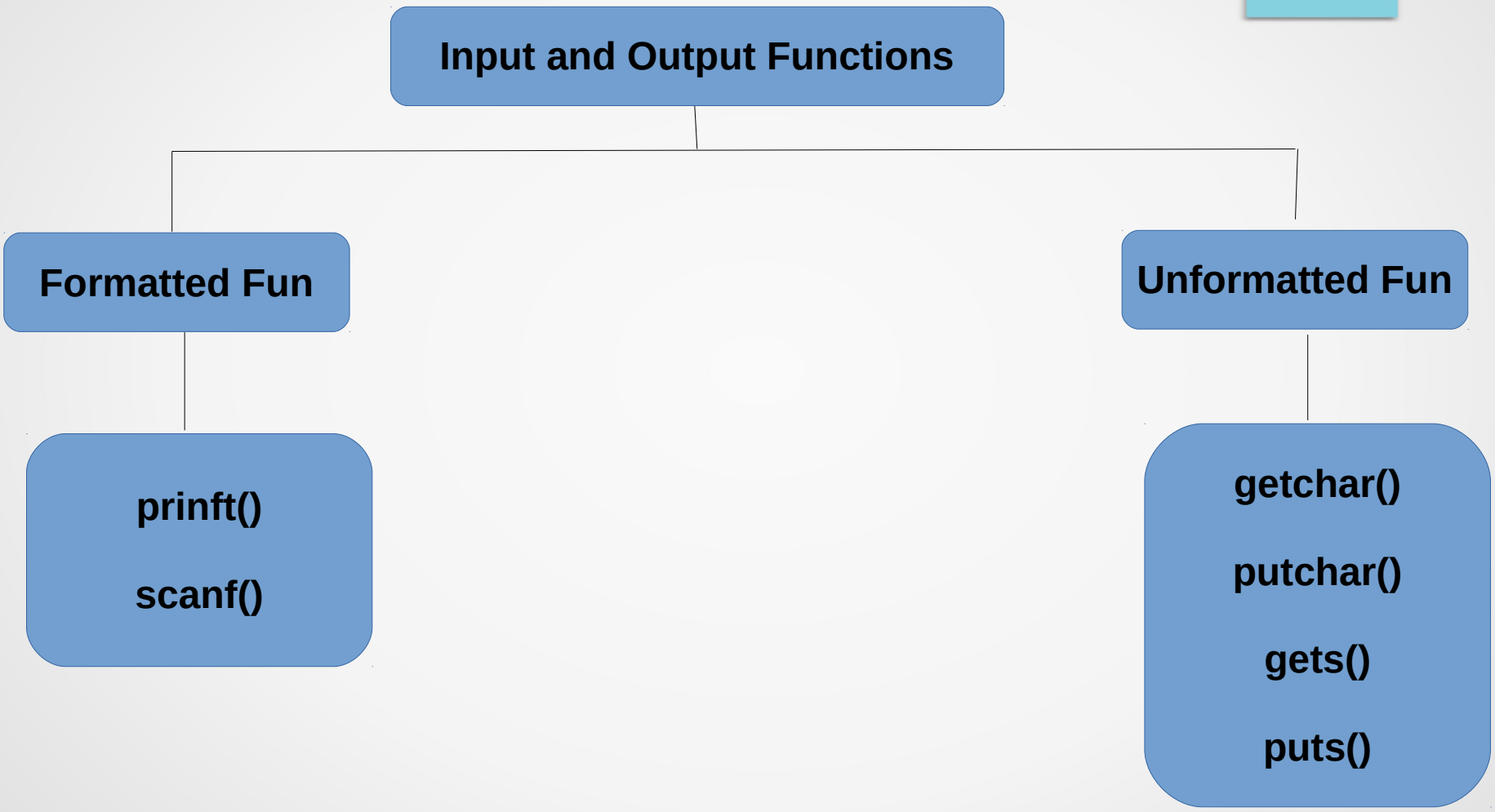
UNIT -2 Header File

Header File	Functions	Function Example
Stdio.h	Input,output and file operation function	*Printf(), scanf(),*
Conio.h	Console input and output functions	*Clrscr(), getche()*
Alloc.h	Memory allocation related function	*Malloc(), realloc()*
Graphics.h	All graphic related functions	*Circle(), bar3d()*

UNIT -2 Header File

Header File	Functions	Function Example
Math.h	Matematical functions	*Abs(), sqrt()*
String.h	String manipulation functions	*Strcpy(), strcat()*
Bios.h	BIOS accessing functions	*biosdisk()*
Assert.h	Contains macros and definitions	*Void assert(int)*
Ctype.h	Contains prototype of functions which test characters	*isalpha(char)*
Time.h	Contains date and time related functions	*asctime()*

UNIT -2 Input and Output Functions



UNIT -2 Input and Output Functions

- Printf() function

Format	Meaning
%wd	Format for integer output
%w.cf	Format for float numbers
%w.cs	Format for string oupput

print_fun1.c

UNIT -2 Input and Output Functions

- scanf() function
  - Syntax:
    - *scanf(“control string”,address of variable1, add of var2,.....);*
    - *Scanf(“%d”,&a);*

scanf_fun1.c
scanf_fun2.c

UNIT -2 Input and Output Functions

- getchar()
  - This function reads a character type data from standard input.
- putchar()
  - This function prints one character on the screen at a time, read by the standard input

getchar_fun.c

UNIT -2 Input and Output Functions

- gets()
  - This function is used for accepting any string through stdin(keyboard) until enter key is pressed.
- puts()
  - This function is used for printing any string on stdout(monitor) device.

gets_fun.c