Introduction to C-programming

Session Objectives

- To Learn the elements of C program
- To Learn about variable declarations and data types
- To about Operators and Expressions
- To know about input output operations

Session Topics

- Introduction to C programming
- History of C
- Operators and Expressions
- Data Input and Output

Introduction to C

•C is a general purpose computing programming language.

•C was invented and was first implemented by ***Dennis Ritchie*** with the Unix Operating System in 1972.

•C is often called a ***middle level*** computer language.

•C is a ***Structured Language***.

Dennis Ritchie

History of the C language

- ALGOL 60 (1960): the first programming language with block structures, control constructs, and recursion possibilities.
- BCPL (Basic Combined Programming Language) (1967), developed by Martin Richards at Cambridge which built a foundation for many C elements.
- B (1970), developed by Ken Thompson at the Bell Laboratories for the first UNIX system.
- BCPL and B are type less languages whereas C offers a variety of data types.
- C (1972), written by Dennis Ritchie at Bell Labs for the implementation of UNIX. With the publication of “The C Programming Language” in 1978 by Kernighan and Ritchie evolved into the standard for C.JJ

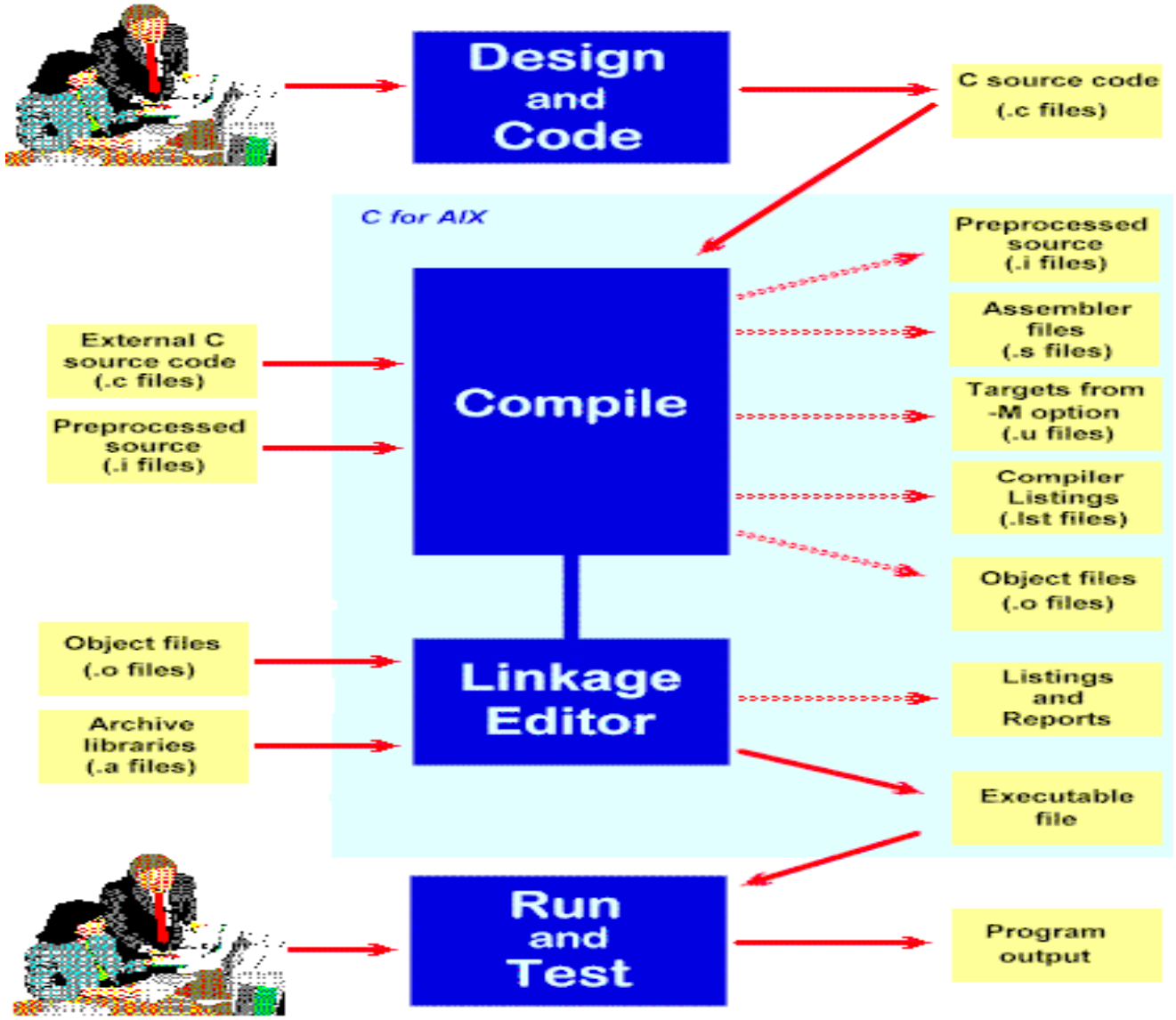
Usage of C

- C's primary use is for ***system programming***, including implementing ***operating systems*** and ***embedded system*** applications.
- C has also been widely used to implement ***end-user*** applications, although as applications became larger much of that development shifted to other, higher-level languages.
- One consequence of C's wide acceptance and efficiency is that the ***compilers***, libraries, and interpreters of other higher-level languages are often implemented in C.
- You will be able to read and write code for a large number of platforms – even ***microcontrollers*****.**

Characteristics of C

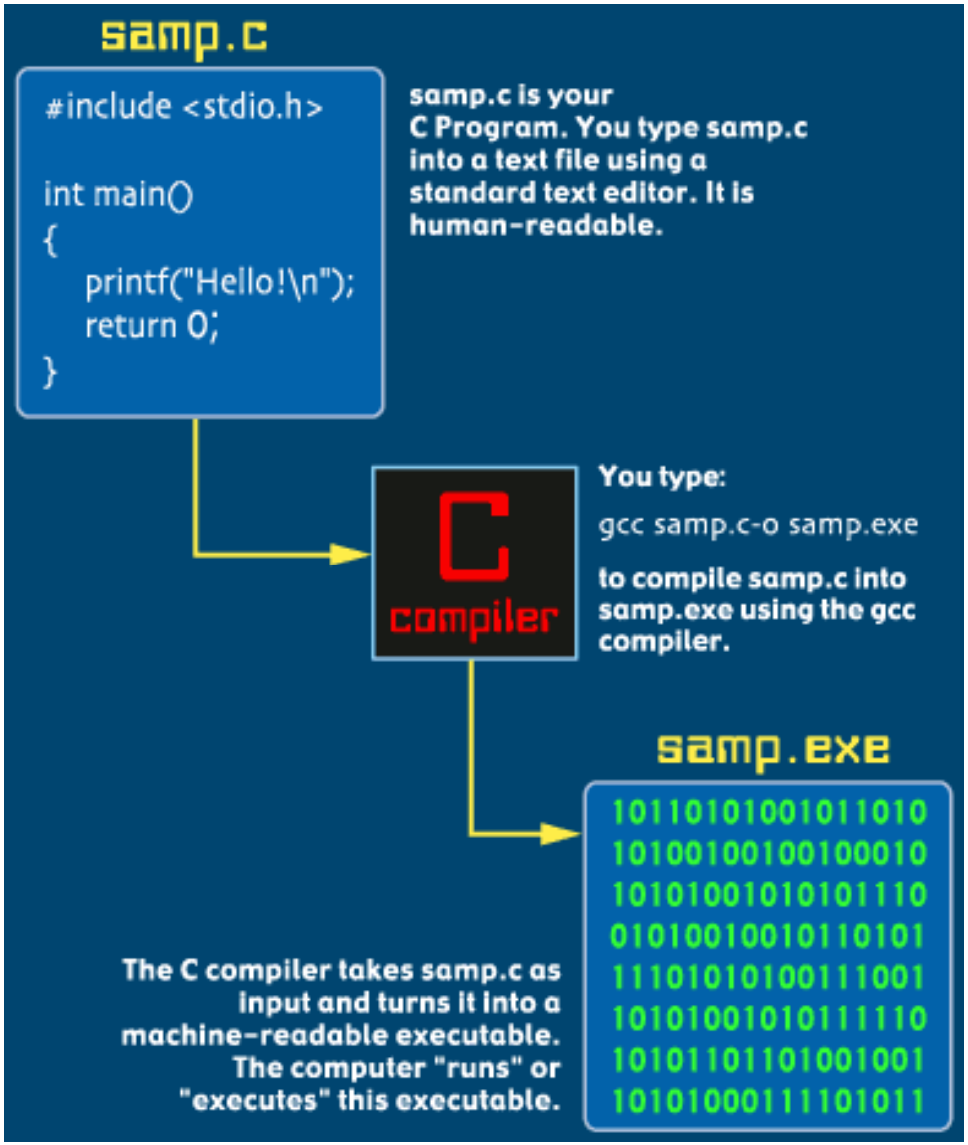
- Portability
  - Portability means it is easy to adapt software written for one type of computer or operating system to another type.
- Structured programming language
  - It make use of subroutines by making use of temporary variables.
- Control the memory efficiently
  - It makes the concept of pointers.
- Various application
  - Wide usage in all upcoming fields.

4 steps of development of C- program



Compilers

- **Commercial Compilers:**
  - Microsoft Visual C++
  - Borland C++ Builder
- **Freeware Compilers:**
  - Borland C++ 5.5 Compiler, also called Turbo C
  - Dev-C++, free IDE (Integrated Development Environment) and compiler for C and C++
  - DJGPP, a DOS based Compiler for C, C++ and Pascal
  - LCC-Win32, free compiler for Windows
  - GCC, the most famous compiler



The Simplest C Program

- Let's start with the simplest possible C program and use it both to understand the basics of C and the C compilation process.
- Type the following program into a standard text editor. Then save the program to a file named samp.c. If you leave off .c, you will probably get an error when you compile it, so make sure you remember the .c.

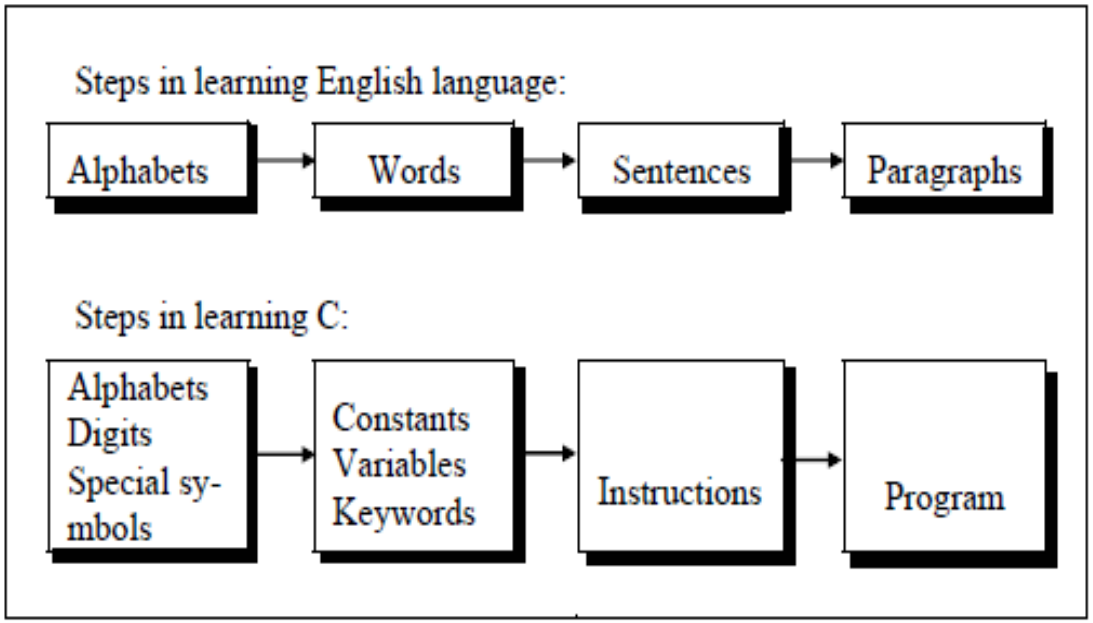
```
 #include <stdio.h>
int main()
{
  printf(“First program!\n");
  return 0;
}
```

Compilation using ***gcc***

1. Use the command mkdir to create a new directory.
2. Use the text editor vi followed by the name of the file.Ex: vi sample.c
3. Use the insert mode to type the text.Use the Insert key or ‘I’.
4. Save the file using :wq. The file is now saved.
5. Compile using gcc followed by the name of the file. Ex: gcc sample.c
6. View the output using a.out file. Type ./a.out.

Language

- Means of Communication
- Chinese – Chinese, English – Chinese, Nepali-Chinese



General Structure of a C Program

- **Opening comment block**
  - Should always contain author's name, name of the file, and purpose of the program
- **Preprocessor directives**
  - include all header files for needed libraries
  - define any useful constants
- **Function prototypes**
  - Must be declared before the function is called
- **Main function**
  - Program execution begins at the start of main() and ends when it reaches the end of main()
  - Any additional functions called **main** are ignored.
- **Other function definitions**

Internal Structure of a C Program

A C source program is a collection of one or more directives, declarations, and statements contained in one or more source files**.**

- **Statements**: Specify the action to be performed.
- **Directives**: Instruct the preprocessor to act on the text of the program.
- **Declarations**: Establish names and define linkage characteristics such as scope, data type, and linkage.
- **Definitions**: Are declarations that allocate storage for data objects or define a body for functions. An object definition allocates storage and may optionally initialize the object.

General Programming Rules

- All C Statements are free-form
  - Can begin and end on any line and in any column
- C statements are always terminated with a semicolon “;”.
- Blank lines are ignored
- White space (blanks, tabs and new lines) must separate keywords from other things
- Comments –

  All text enclosed within “/* ----- */”

  Text on the same line following “//”

  Examples:

```
// This is a comment
/* So is
        this. */
```

C Tokens

- Character Set :
  - Letters : A..Z, a...z and Digits : 0...9
  - Special characters : , . : ; ? ’ “ ! | / \ ~ -_ $ % # & ^ * + - < > ( ) { } [ ]
  - White space : blank space, horizontal tab, vertical tab,carriage return, new line, form feed.
- C tokens : individual units.
  - Keyword – float, while, for, int,….
  - Identifier – main( ) , amount, sum, …
  - Constants – -13.5, 500, …
  - Strings – “ABC”, “MCA”, …
  - Operators – + - * % …
  - Special Symbols – [ ] { }…

C Tokens

- There are several rules that you must follow when naming constants and variables:

Names...	Example
CANNOT start with a number	2i
CAN contain a number elsewhere	h2o
CANNOT contain any arithmetic operators...	r*s+t
CANNOT contain any other punctuation marks...	#@x%£!!a
CAN contain or begin with an underscore	_height_
CANNOT be a C keyword	struct
CANNOT contain a space	im stupid
CAN be of mixed cases	XSquared

- All variables must be declared before using them in a program and Variable declarations can also be used to initialize the variable. (not good practice)
- We can declare variables in a single statement with the list of variables separated by commas. Eg. int lower, upper, step;

Keywords used in C

- Fixed meaning, cannot be changed
- Basic building block
- Lowercase

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

Identifiers, Constants & Variables

Identifiers

- Refers to names of variables, functions,…
- User defined names.
- Uppercase and lowercase.

Constants

- Fixed values.
- Does not changes during execution of program.
- Numeric constant – Integer (decimal, octal,hexadecimal) and Real
- Character constant :
  - Single character constant
  - String constant
  - Backslash character constant

Variables

- Data name used to store data value.
- May take different values at different times during execution.
- Chosen by the programmer.
- Letters, digits and '_' are allowed.

Data types

- Primary data types
- User defined data type
- Derived data type (array, function, structure,..)
- Empty data set

Type	Size (bits)	Range
char or signed char	8	-128 to 127
unsigned char	8	0 to 256
int or signed int	16	-32768 to 32767
unsigned int	16	0 to 65535
short int or signed short int	8	-128 to 127
unsigned short int	8	0 to 255
long int or signed long int	32	-2147483648 to 1247483647
unsigned long int	32	0 to 4294967295
float	32	3.4 E –38 to 3.4 E + 38
double	64	1.7 E –308 to 1.7 E + 308
long double	80	3.4 E –4832 to 1.1 E +4932

Variable Declarations in C

- All variables must be declared before using them in a program.
- Variable declarations can also be used to initialize the variable.
- We can declare variables in a single statement with the list of variables separated by commas.
  - int lower, upper, step;
  - float fahr, celsius;
  - int i=0;
  - char backslash = '\\'

Characters Representation

- Characters are represented at the machine level as an integer
- Computers use ASCII (American Standard Code for Information Interchange) code
- The values used as code range from 0 to 255
- A-Z (upper case) are represented by 65-90
- a-z (lower case) are represented by 97-122
- Constants can be defined using the *#define* construct (symbolic constant)
  - #define LOWER 0
  - #define UPPER 300
  - #define STEP 20

Test your basic knowledge

- C is ??? Programming language
  High level, mid-level, Low-level
- C was invented by ??? In 1972
  Madonna, Dennis Taylor, Guy Ritchie, **Dennis Ritchie**, Dennis the Menace
- Computer understands C.
  TRUE or **FALSE**
- Which language does not have syntax similar to that of C?
  C++, JavaScipt, Perl, **Visual Basic**, Java
- C is not case sensative.
  TRUE or **FALSE**
- Which is a valid C commnet?
  /*/*…*/*/, **/*…*/**, <!… >, ‘…., //….
- Which is a correct sequence?
  coding linking compiling, **coding compiling linking,** linking compiling coding, linking coding compiling, compiling coding linking

Test your basic knowledge

- Which is a valid Hello World program?

```
#include<stdio.c>
int main(){
printf("Hello
World");
return 0; }
```

```
#include<stdio.h>
int main(){
print("Hello
World");
return 0; }
```

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

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

Operators in C

Arithmetic operators

- + Addition
- - Subtraction
- * Multiplication
- / Division
- % Modulus (remainder)

Comparison Operators

- > Greater than
- >= Greater than or equal to
- < Less than
- <= Less than or equal to
- == Identically equal to
- != Not identically equal

All arithmetic operators have higher precedence than the comparison operators.

The top four have higher precedence (priority of evaluation order) than the "equal to" and "not equal to" operators.

Boolean Values in C

- There is no separate Boolean data types in C
- Any variable can be treated as a Boolean value
- Rules of evaluation
  - A numerical value of 0 (zero) represents ***FALSE***.
  - A numerical value other than zero represents ***TRUE***.
- When a Boolean expression is evaluated, it returns:
  - 0 for *FALSE*
  - 1 for *TRUE*

Logical Operators

- NOT reverses the truth value of its operand:

Expression	Return Value
!1	0
!0	1

- AND returns 1 if both operands return non zero values Expression Return Value

1 && 1	1
1 && 0	0
0 && 1	0
0 && 0	0

- OR only returns 0 if both operands return zero:

Expression	Return Value
1 || 1	1
1 || 0	1
0 || 1	1
0 || 0	0

NOT has higher precedence than AND, which has higher precedence than OR.

Bitwise Logical operator

-- & bit wise *AND:* 9 & 23 =01001 & 10111=00001

– | bit wise inclusive *OR* 9|23=01001|10111=11111

– ^ bit wise exclusive *OR* 9 ^ 23=01001^ 10111=11110

– << left shift - add zeros to the right bit, a << b returns $a*2^b$, 4 << $2=4*2^2 = 16$

– >> right shift - takes away bit on the right a >> b = $a/ 2^b$ 8 >> 2 = $8 / 2^2 =2$

– ~ one's complement-

9 is 0000 0000 0000 1001

~9= 1111 1111 1111 0110 in unsigned int is 65536 and in signed short int is –10 (convert 1111 1111 1111 0110 using 2's compliment notation)

Remember that in signed short int mode, ~x is -1 - x !

Bitwise Logical operator

- **The Conditional Operator** - It's possible to say: "If this condition is true, do this.... , otherwise, do this .... " all in one line. This can be achieves using the **CONDITIONAL OPERATOR**, which is represented by is represented by **? :**
  - It may look a bit odd at first, but it takes 3 expressions as operands, like this: a ? b : c; a should be a condition you want to test for. b is an expression that is evaluated if a returns a non zero value (in other words, if a is true), else expression c is evaluated.
  - For example: x<1 ? printf("x<1") : printf("x>=1");
  - Note the 3 separate operands are expressions - the whole line is a statement. It's a common mistake to put a semi colon in the middle expression like this:

    x<1 ? printf("x<1"); : printf("x>=1"); - this will generate an error when you compile!
- Increment and decrement
  - ++n pre-increment n++ post increment
  - --n pre-decrement n-- post decrement

Operators in C

- Assignment operations
  - True for the following operators

    + - * / % << >> & ^ |
  - Expression of the form: e1 op = e2 is equivalent to e1 = e1 op e2
  - Examples
    - j += 4; is the same as j = j + 4;
    - x *= y+1; is equivalent to x = x*(y+1)
- What will the value of x be after executing the following code?

```
a = 5; b = 6
if (a = b)     x = 1;
else           x=2;
```

Rules of Associativity

Operator	Associativity
( ) [ ] ->	Left to right
! ~ ++ -- - (type) * & sizeof()	Right to left
* / %	Left to right
+ -	Left to right
<< >>	Left to right
< <= > >=	Left to right
== !=	Left to right
&	Left to right
|	Left to right
^	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= += -= /= %= &= |= ^=	Right to left
,	Left to right

Format Specifiers

Format Specifier	Meaning
%c	Single character
%d	Signed decimal integer
%e	Floating-point number, e notation
%E	Floating-point number, E notation
%f	Floating-point number, decimal notation
%g	Causes %f or %e to be used, whichever is shorter
%G	Causes %f or %E to be used, whichever is shorter
%i	Signed decimal integer
%o	Unsigned octal integer
%p	Pointer
%s	Character string
%u	Unsigned decimal integer
%x	Unsigned hex integer using digits 0-f
%X	Unsigned hex integer using digits 0-F
%%	Print a percent sign

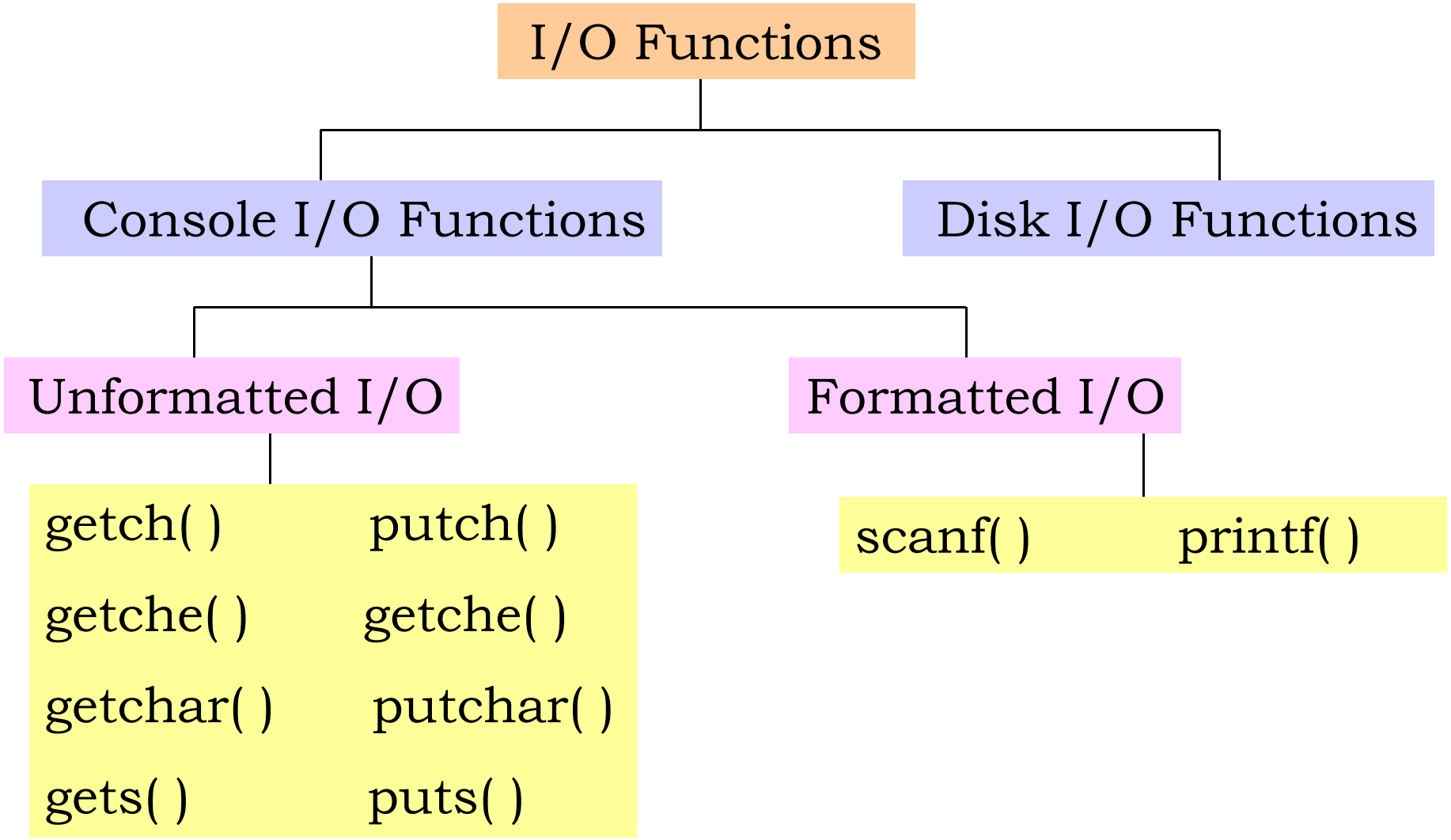
contd…Format Specifiers

Format Specifier	Meaning
%i	Signed decimal integer
%o	Unsigned octal integer
%p	Pointer
%s	Character string
%u	Unsigned decimal integer
%x	Unsigned hex integer using digits 0-f
%X	Unsigned hex integer using digits 0-F
%%	Print a percent sign

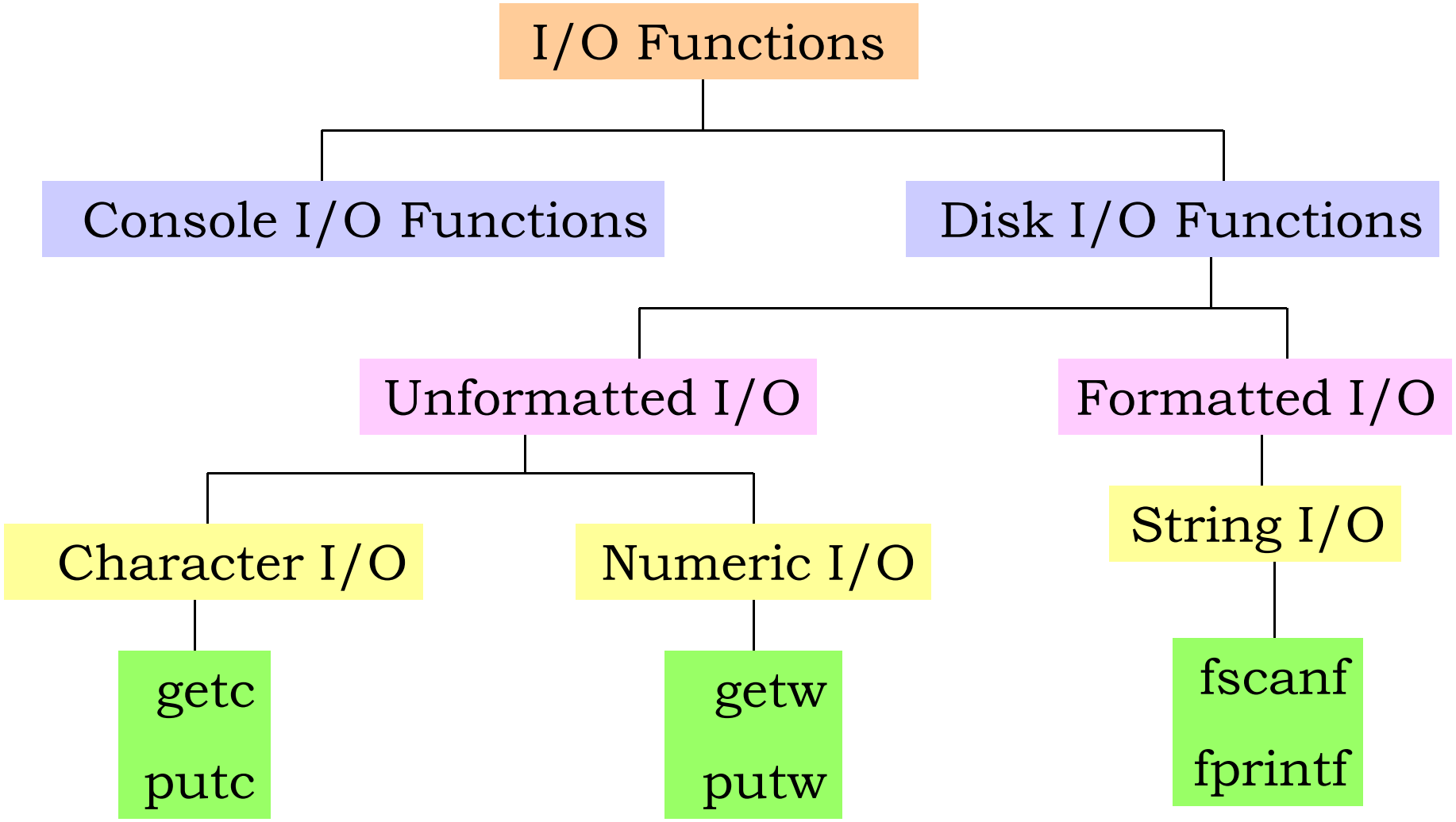
Mathematical functions

- C compiler support basic math functions like cos, sin, sqrt,..
  - #include<math.h>
- Functions-
  - log(x) , acos(x), asin(x), atan(x), cos(x), sin(x), log10(x), pow(x,y), sqrt(x), csoh(x), sinh(x), tann(x), ceil(x), floor(x), exp(x), fabs(x) , fmad(x,y)
- Redundant parenthesis do not cause errors or slow down the execution of an expression
- Use parenthesis to clarify the exact order of evaluation

Console Input/Output Functions



Disk Input/Output Functions



Formatted Input Function

scanf( )

- int scanf(char *format, args....) -- reads from stdin and puts input in address of variables specified in argument list.

- Returns, number of charecters read.
- The address of variable or a pointer to one is required by scanf.
  Example: scanf(``%d",&i);
- We can just give the name of an array or string to scanf since this corresponds to the start address of the array/string.

Example:

```
char string[80];
scanf(``%s",string);
```

Formatted Output Function

printf()

- The printf function is defined as follows:
  int printf(char *format, arg list ...) --
  prints to stdout the list of arguments according specified format string.
  Returns number of characters printed.
- The format string has 2 types of object:
- *ordinary characters* -- these are copied to output.
- *conversion specifications* -- denoted by % and listed in Table.

Formatted I/O Functions

- Between % and format char we can put:
- - (minus sign)
  - -- left justify.
- integer number
  - -- field width.
- -- m = field width, d = precision of number of digits after decimal point or number of chars from a string.
  So:

  ```
  printf("%-2.3f n",17.23478);
  ```

  The output on the screen is:

  ```
  17.235
  ```

  and:

  ```
  printf("VAT=17.5%% ");
  ```

  ...outputs:

  ```
  VAT=17.5%
  ```

Program illustrating Formatted I/O Functions

```
#include <stdio.h>

int main()
{
  int day;
  int month;
  int year;
  char name[30];

  printf("Enter your name:\n">
  scanf("%s", name);

  /* skipping spaces */
  printf("Hi %s. Enter birthdate as: dd mm yyyy\n", name);
  scanf("%d %d %d", &day, &month, &year);

  /* alternative */
  printf("Hi %s.  Enter birthdate as: dd-mm-yyyy\n", name);
  scanf("%d-%d-%d", &day, &month, &year);

  return 0;
}
```

Note: no ampersand for strings

Conversion specifier

Literal characters

Unformatted Console I/O Functions

getch( ) and getche( )

- This function will read a single character the instant it is typed without waiting for the Enter key to be hit.
- These functions return the character that has been more recently typed.
- The ‘e’ in ***getche()*** function means it echoes(displays) the character that has been typed.
- The ***getch()*** just return the character that has been typed without echoing it on the screen.

Unformatted Console I/O Functions

getchar( ) and putchar( )

- The ***getchar( )*** accepts a single character the instant it has been typed.
- The ***getchar( )*** echoes the character that has been typed.
- It requires the Enter key to be typed following the character that has been typed.
- The functions ***putch()*** and ***putchar()*** print the character on the screen.

Program illustrating Console Input Functions

```
main()
{       char ch;
        printf("Hello\n");
        getch();
        printf("Type any character\n");
        ch=getche();
        printf("Type any character\n");
        ch=getchar();
}
```

Output:

Hello

Type any character A

Type any character B

Program illustrating Console Output Functions

```
main()
{       char ch='D';
        putch(ch);
        putchar(ch);
        putch('S');
        putchar('S');
}
```

Output:

DDSS

Unformatted Console I/O Functions

gets( ) and puts( )

- The function ***gets()*** receives a string from the keyboard.
- It is terminated when an Enter key is hit.
- The spaces and tabs are acceptable as part of the input string.
- The gets() function gets a newline '\n' terminated string of charecters and replaces the '\n' with '\0'.
- The ***puts()*** function outputs a string to the screen.

Program illustrating Console I/O Functions

```
main()
{
        char arr[25] ;
        puts("Enter name\n");
        gets(arr);
        puts("Welcome\n");
        puts(arr);
}
```

Output:

Enter name

Embedded Systems

Welcome

Embedded Systems

Summary

- C is a general purpose computing programming language which was invented and was first implemented by ***Dennis Ritchie***
- The main characteristics of C language are portability, structured programming, memory efficiency etc.,
- Data types supported by C language are integer, float, double, character etc.,
- Both formatted and unformatted input output statements are supported by C

Thank You!