LOVELY PROFESSIONAL UNIVERSITY

CSE101
Computer Programming

Lecture #0

Course Details

- **L T P**

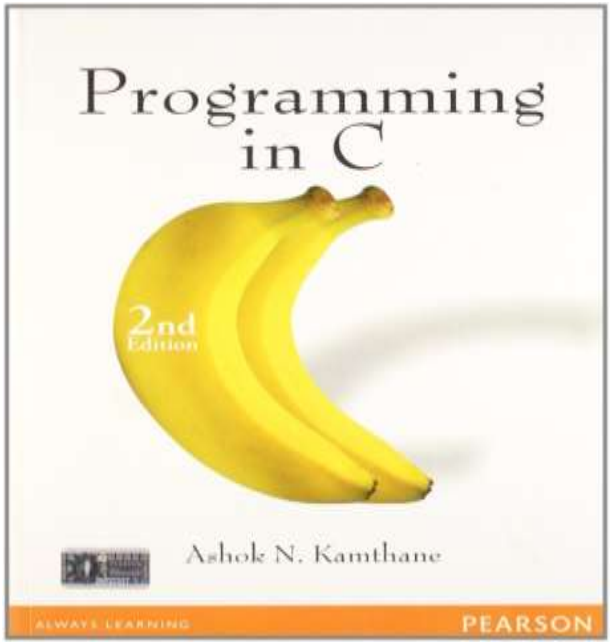
  2 0 2

- **Text Book**
  - "PROGRAMMING IN C"

  by

  ASHOK N. KAMTHANE

  PEARSON, 2nd Edition



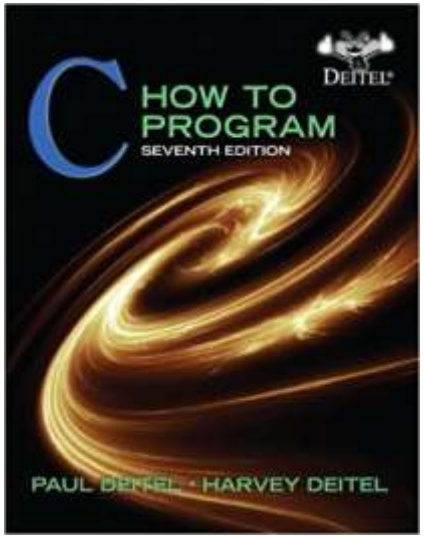
Reference Books

- "C HOW TO PROGRAM" by

PAUL DEITEL AND

HARVEY DEITEL

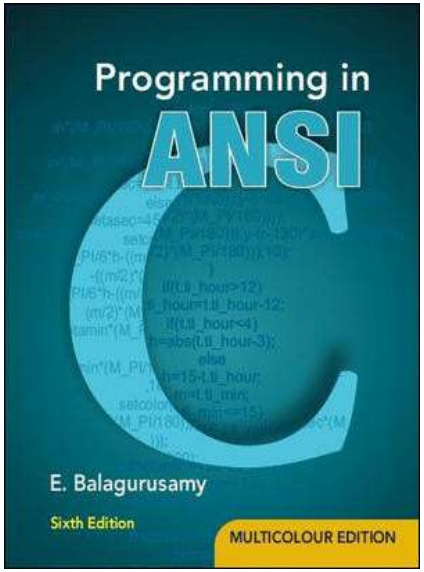
PHI(Prentice Hall India)



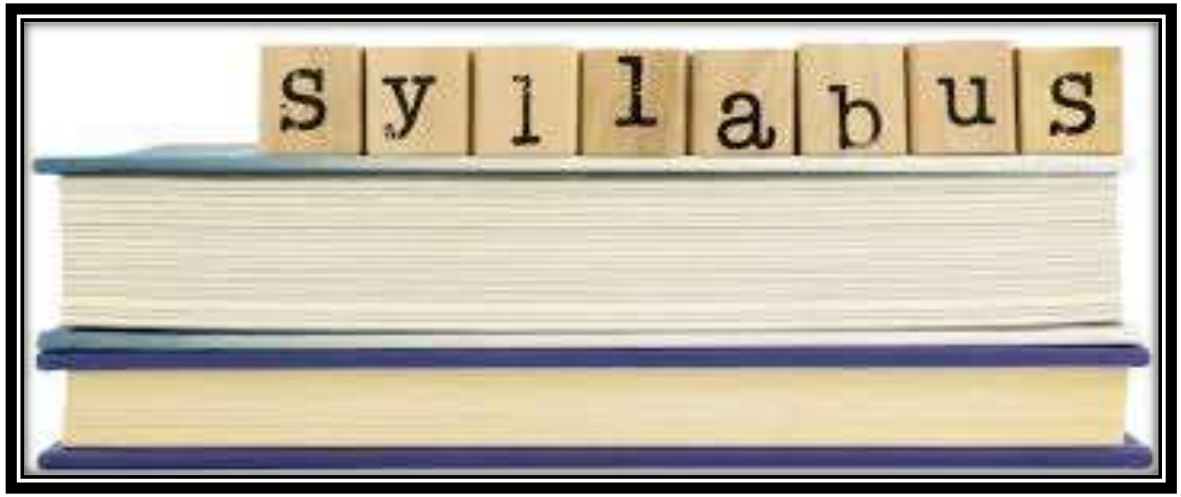
- "PROGRAMMING IN ANSI C"

By E. BALAGURUSAMY

McGraw Hill Education



Syllabus and Instruction Plan(IP)



Course Assessment Model

- **Marks break up**

Attendance	5
CA	25
MTT (MCQ pattern)	20
ETT (MCQ pattern)	50
Total	**100**

Academic Task

Component	Week
1. Mini Project	3rd
2. Code based test 1 (offline)	5th
3. Code based test 2 (online)	12th

- Mini project (compulsory)
- From CBT1(MCQ) and CBT2 (online coding test) best of one will be considered.
- CBT1 before midterm and CBT2 after midterm.
- Total Weeks: 14(7 Before MTE and 7 After MTE).

Programming Practice

Each week Two questions will be given

One will be done in class and one will be given as homework.

NOTE: Students have to submit at least 75% Questions otherwise they will be DEBARRED from the examinations.

Mode of Conduct

- **BYOD(Bring your own device)**
- **Note: Laptops are mandatory for program implementation.**

WHY C??????????????

- If we have number of powerful programming languages available with us then why c??????

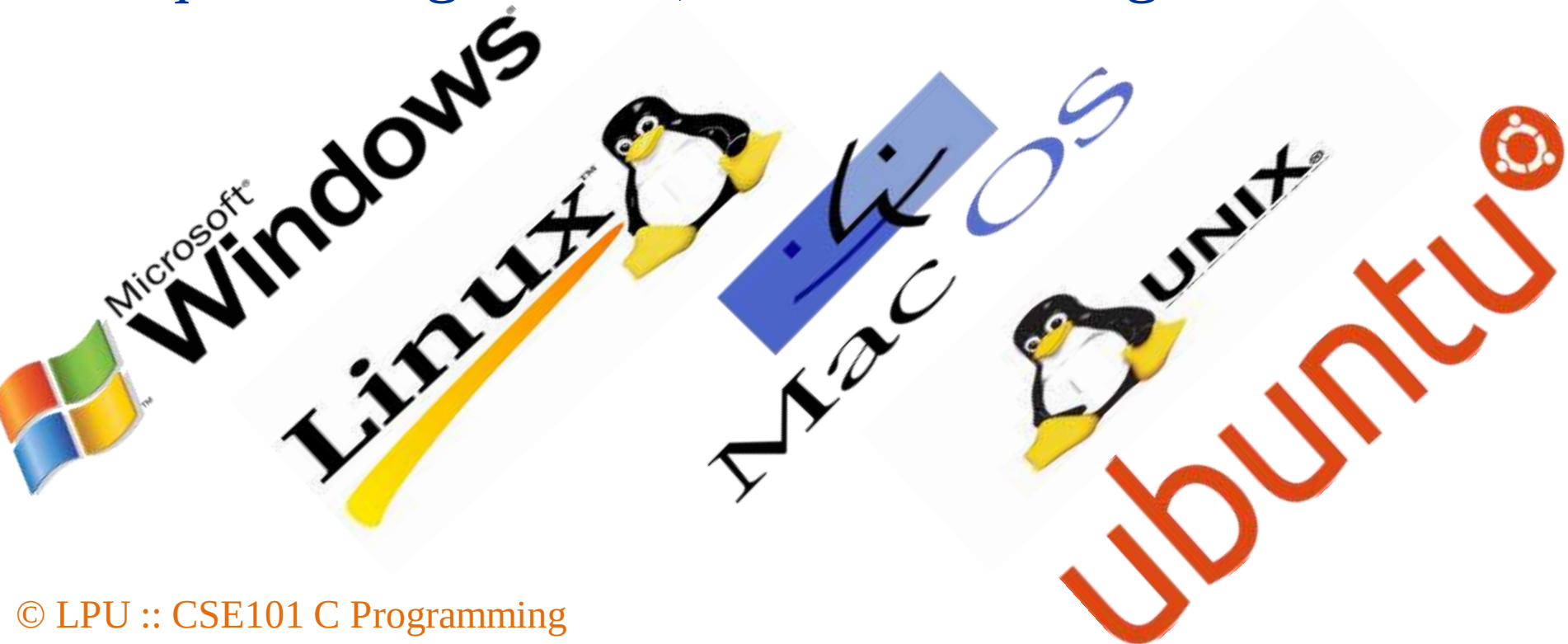
The hitch.....

Some burning questions in mind.....

- C is a very old language. Why are we still studying this language??????
- Now, we have very powerful languages with us then, why c??
- There is no scope of this language in industry

Lets take you close to the reality

- All of us use Computers or Laptops for different purposes.
- Could you tell me which system software is most required to get our system in working mode??????



Let's Explore more

- Could you tell me which programming language is used in writing all these operating system??????

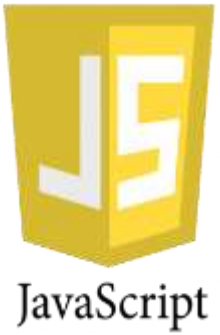
- Latest version of Microsoft Windows i.e. Windows10 is still being written in C Language

Windows 10

C PROGRAMMING

Contd.....

- Device drivers are also written in C language.
- All these modern programming languages are influenced by C language



- Compilers for Python and PHP language are also written in C language

Contd.....

- Embedded systems are also developed with the help of C language

Contd.....

- Git 
- Microsoft Excel 
- Oracle Database 
- MySql 
- Linux 
- Unix 
- Android 
- Google 

MNCs

Top rated Companies which has a dearth of C programmers

Here are the Answers of Questions

- C is very a old language still, why do we study C language??

- Now, we have very powerful languages with us then why c??

- There is no scope of this language in industry

Scope for Experts always Exists

History of C

- Guys Can you make a sentence with the word '***Necessity***'

  "Necessity is the mother of invention"

- Dennis Ritchie and Ken Thompson were working on developing a new operating system i.e UNIX
- But the programming language they were using was not providing them the portability feature
- So Dennis Ritchie developed new language i.e C

History continued...

Summary -		
1	B Language Developed By	Ken Thompson
2	Operating System Developed in C	UNIX
3	Developed at	AT & T Bell Laboratory
4	Creator of Traditional C	Dennis Ritchie
5	Year	**1972**

Why “C” name was given???

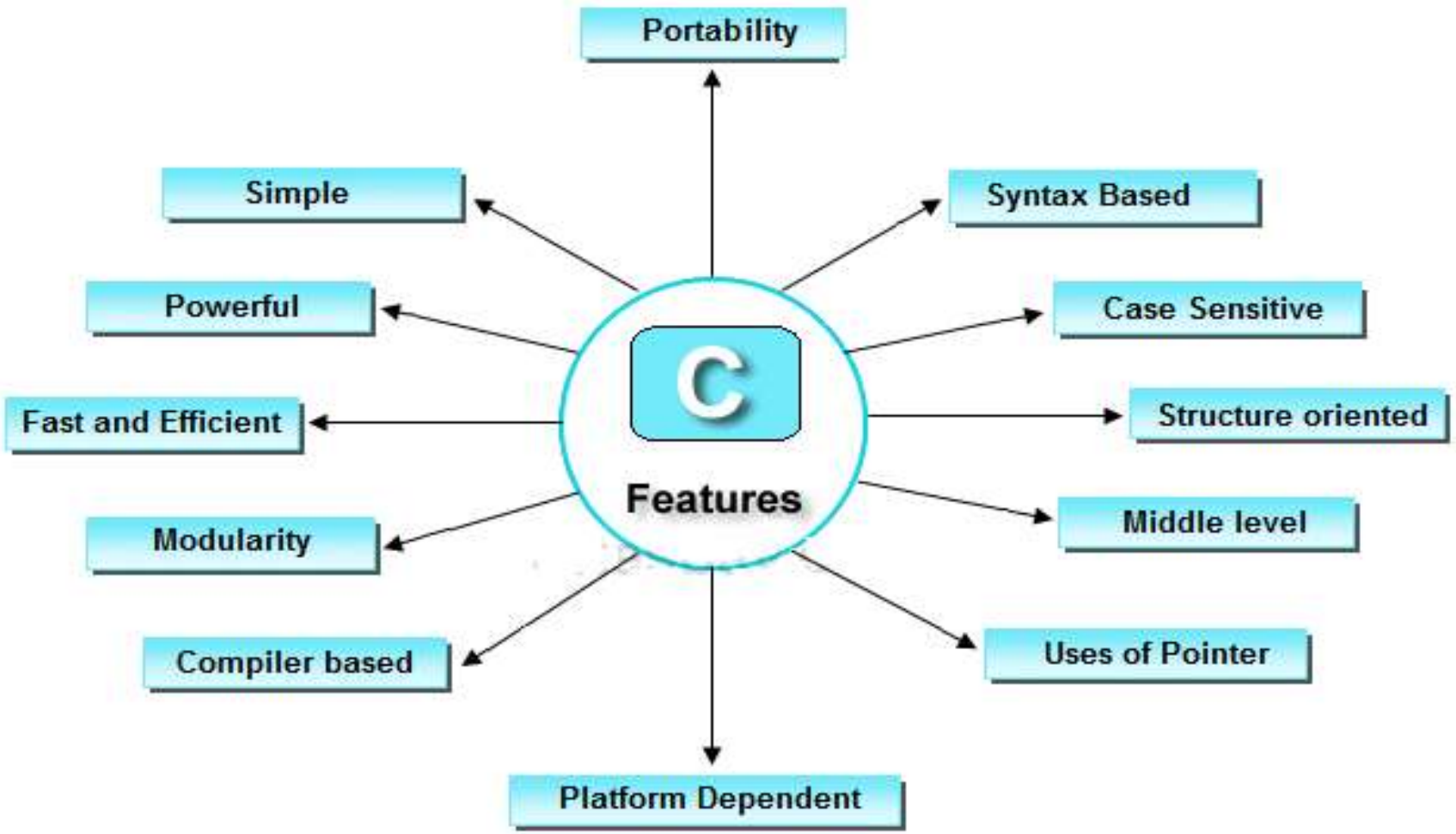
- Many of C’s principles and ideas were derived from the earlier language B. (Ken Thompson was the developer of B Language.)
- BCPL and CPL are the earlier ancestors of B Language
- CPL is Combined Programming Language. In 1967, BCPL Language (Basic CPL) was created as a scaled down version of CPL
- As many of the **features were derived from “B” Language thats why it was named as “C”**.
- After 7-8 years C++ came into existence which was first example of object oriented programming .

Evolution of C...

Language Developers

Algol	• International Group
BCPL	• Martin Richards
B	• Ken Thomson
Traditional C	• Dennis Ritchie
K&R C	• kernighan & Ritchie
ANSI C	• ANSI Commitee
ANSI/ISO C	• ISO Commitee
C99	• Standerd Commitee

Features of C Language



Applications of C

Various Real World Applications of C

Course Contents

Before MTE

- ✓ Data Types & Operators
- ✓ Control Structures
- ✓ User Defined Functions
- ✓ Storage Classes

After MTE

- ✓ Arrays and Strings
- ✓ Pointers
- ✓ Dynamic Memory Allocation
- ✓ File Input/Output
- ✓ Derived Data Types- Structures and Union

Who is father of C Language?

A. Bjarne Stroustrup
B. James A. Gosling
C. Dennis Ritchie
D. Dr. E.F. Codd

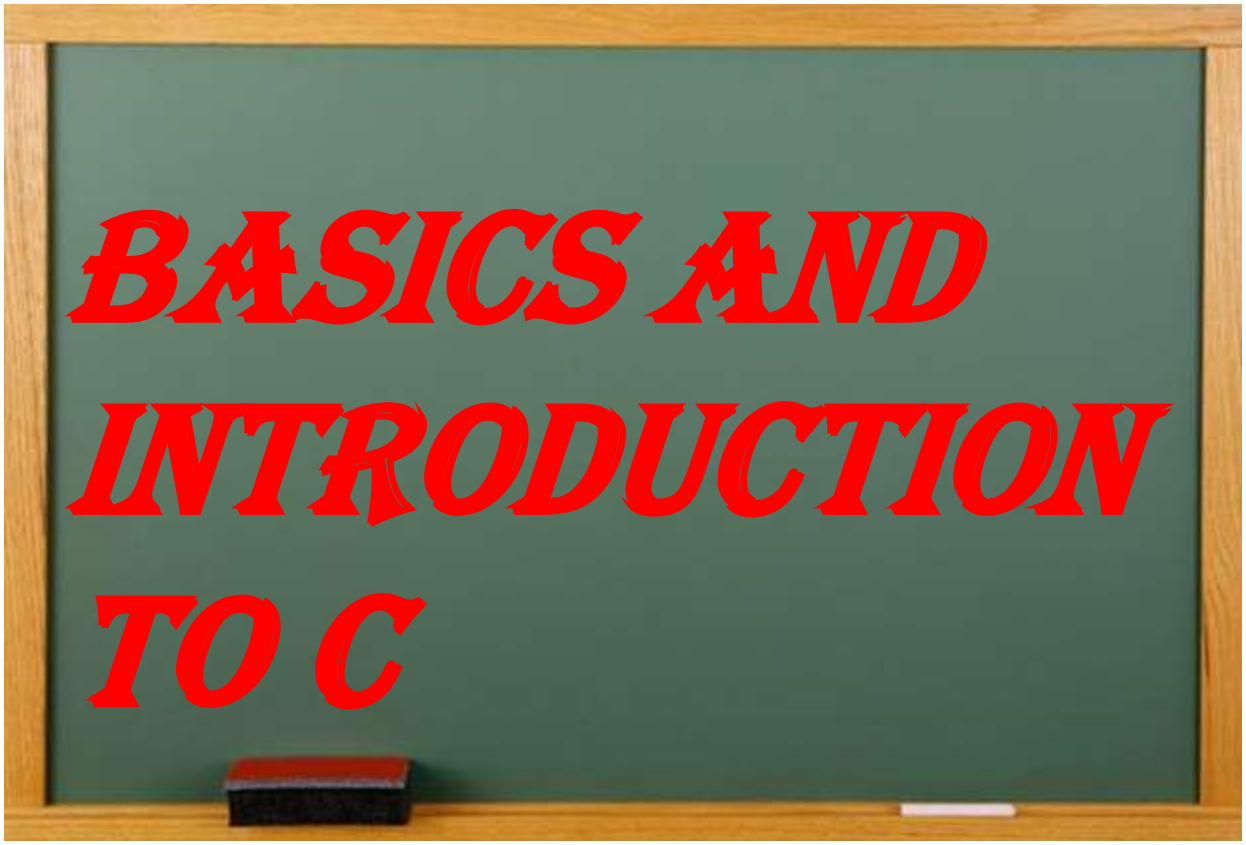
Which of the following statement is correct about the C language?

1. The C language is a binary language with some extra features.
2. The C language is a high-level language with some low features.
3. The C language is a mid-level language with some high features.
4. The C language is a low-level language.

C Language developed at _____?

A. AT & T's Bell Laboratories of USA in 1972
B. AT & T's Bell Laboratories of USA in 1970
C. Sun Microsystems in 1973
D. Cambridge University in 1972


The Next Class


BASICS AND
INTRODUCTION
TO C

CSE101-Lec#1

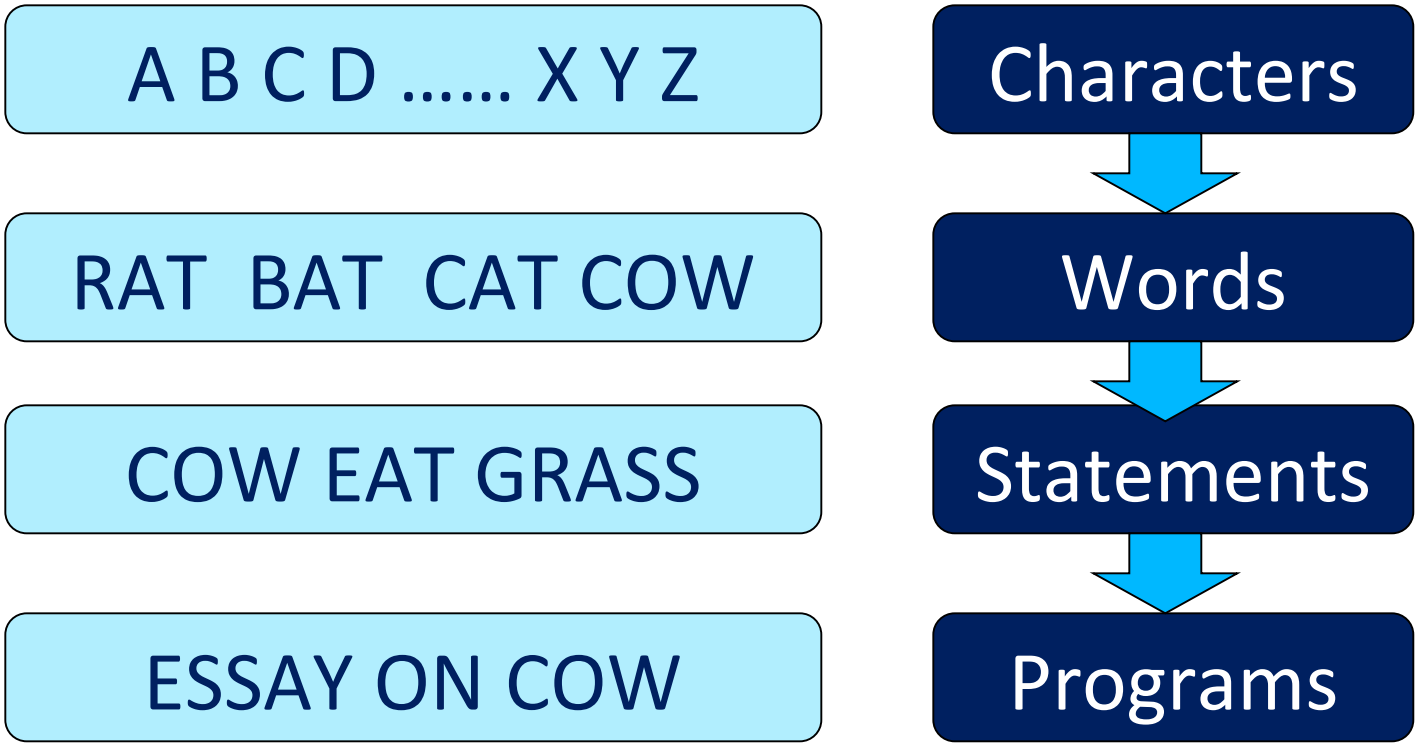
Character Set
Identifiers and Keywords
Data types

OUTLINE

- In this lecture we will cover
  - Character set
  - Identifiers
  - Keyword
  - Data types

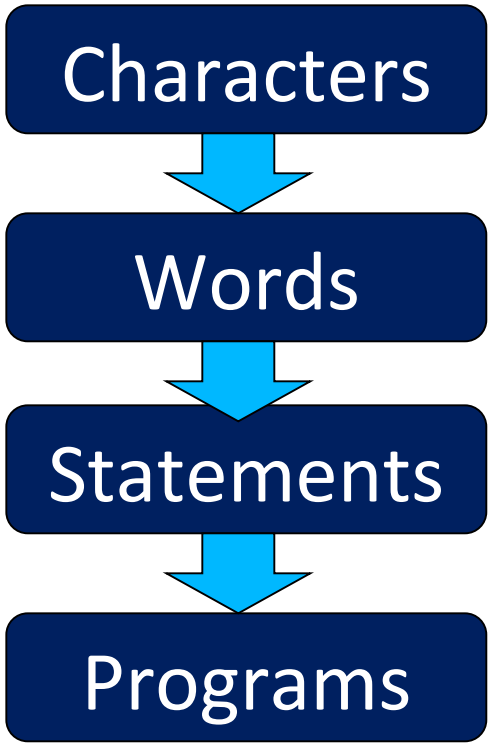
Language: its influence in our life

- Let us look to what we are doing since our childhood, how did we learnt ENGLISH- A recap



Introduction to C

- Like every language C programming language requires basic building blocks to communicate with the computer.
- So we require
  - Character set
  - Words(keywords and identifiers)
  - Statement (instructions)
  - Program



Character Set

- The character set of C represents alphabet, digit or any symbol used to represent information.

Types	**Character Set**
Uppercase Alphabets	A, B, C, … Y, Z
Lowercase Alphabets	a, b, c, … y, z
Digits	0, 1, 2, 3, … 9
Special Symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /
White spaces	Single space, tab, new line.

Meaningfulness

- Let us look to some words
- **saslc, enp, keib, rac, llab**

- Rearrange
- **Class, pen, bike, car, ball**

- This is the influence of adding meaning by logical and sensible grouping in mode of communication through language

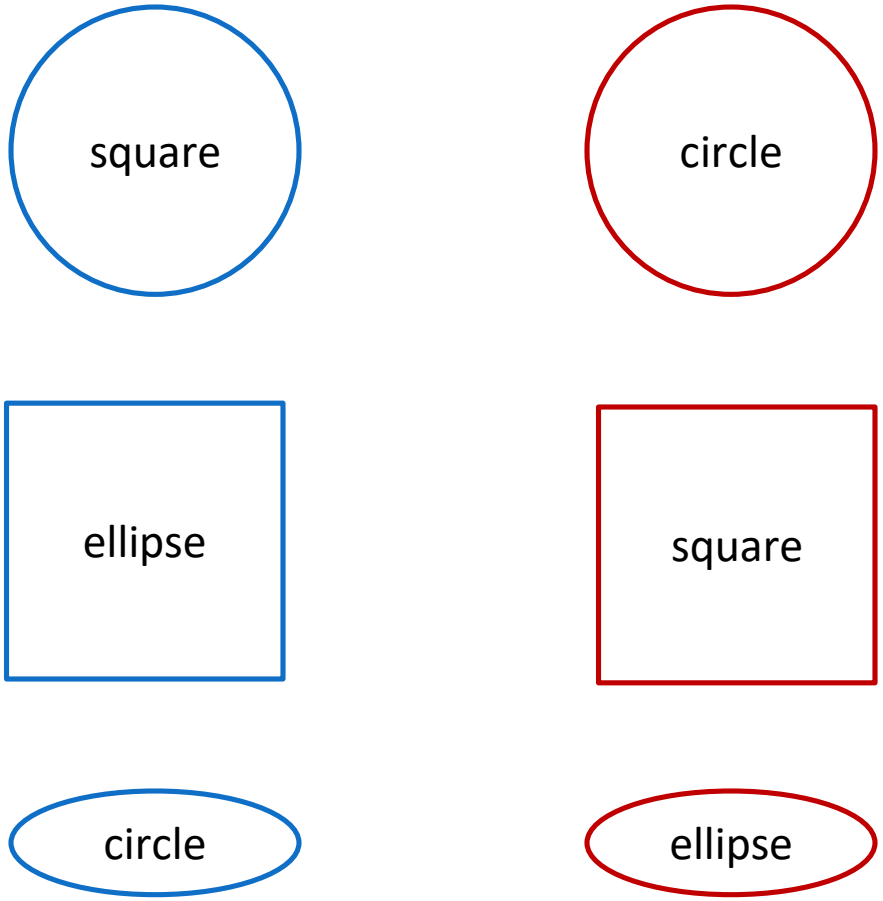
Token

- Every single element in a C Program is Token

Token

- Smallest unit in a program/statement.
- It makes the compiler understand what is written in the program.
- Example: main, printf , name,), etc.
- Tokens are broadly classified as:
  - Identifiers
  - Keywords
  - Constants
  - Variables
  - Strings
  - Operators
  - Special character

Lets Identify the following:



Identifiers

- So to identify things we have some name given to them .
- Identifiers are the fundamental building blocks of a program
- Used to give **names** to variables, functions, constant, and user defined data.
- They are user-defined names and consist of a sequence of letters and digits

Rules for naming an Identifier

1. An identifier name is any combination of 1 to 31 alphabets, digits or underscores.

2. The first character in the identifier name must be an alphabet or underscore.

3. No blanks or special symbol other than an underscore can be used in an identifier name.

4. Keywords are not allowed to be used as identifiers.

Some Identifiers

Tool_spanner;
tool_spanner;

both are different

FORMULA1;

engine_1;

Wrong identifiers name

1_engine;

break;

@car-roof;

C Keywords

- Keywords are the reserved words whose meaning has already been explained to the C compiler.
- We cannot use these keywords as variables.
- Each keyword is meant to perform a specific function in a C program.
- There are 32 keywords in C language.
- All keywords are written in lowercase only

Eg: The **name** of person can never be **home**, **eat**, **sleep**, **run**, etc because these words have some predefined meaning to perform some task.

List of C 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

Which of the following is a valid identifier name?

A. 1abcpqr
B. break
C. abc_pqr
D. ^abc

Which of the following is not a valid identifier?

a) __a3

b) __3a

c) __A3

d) None of the mentioned

Which of the following is not a valid identifier?

a) _a3;

b) a_3;

c) 3_a;

d) _3a;

Data Types

- Data type means the type of value a variable will have.
- It also defines memory space for a particular variable in computer.
- The type of value of variable can be alphabets or numbers.
- The numbers can be further divided as the integer or rational number.

- Lets see a mathematics problem:

My-Car

1. If the radius of car wheel is 15inch then what will the distance traveled after one rotation of that wheel?

Sol: Given-

radius = 15 — 15 — Integer(int in C)

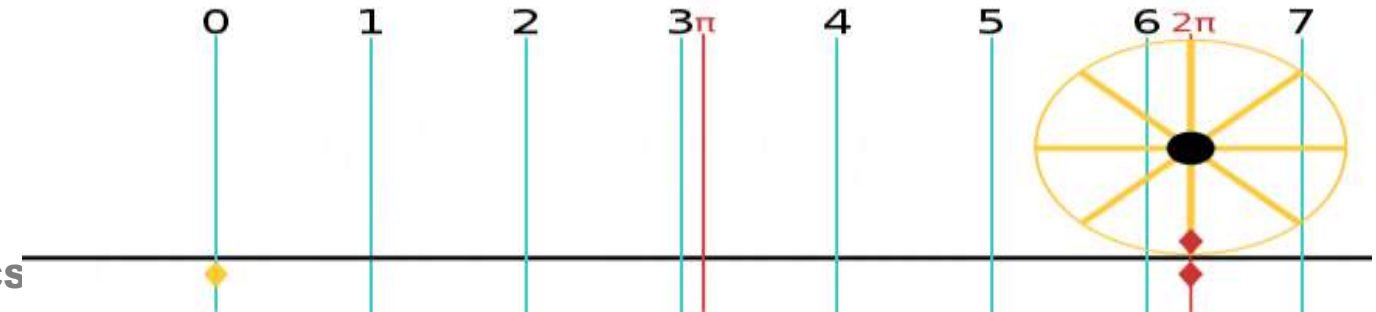
dist_travelled = ?

So, Circumference of circle = 2 * pi * r

dist_travelled = 2 * 3.14 * radius — 3.14 — Real (float in C)

dist_travelled = 6.28 * 15

dist_travelled = 94.2 Ans. — 94.2 — Real (float in C)



My-Grades

2. Five students have appeared for Mathematics exam and their respective marks are

   84,34,97,58,64

consider the rank bands and their respective grades as

80 to 100 – A
60 to 79 – B
40 to 59 – C
less than 40 – D

So find the grade for each students?

Sol: Given-

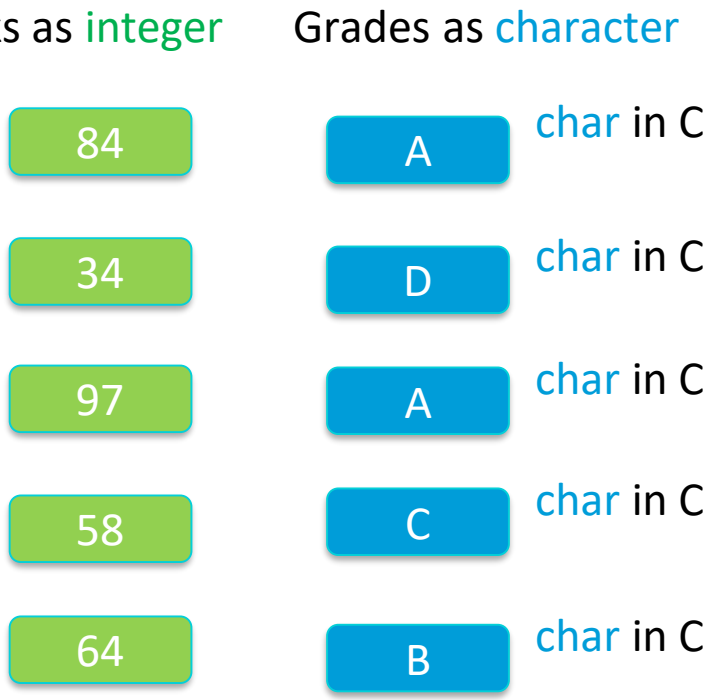
M1=84, G1=?

M2=34, G2=?

M3=97, G3=?

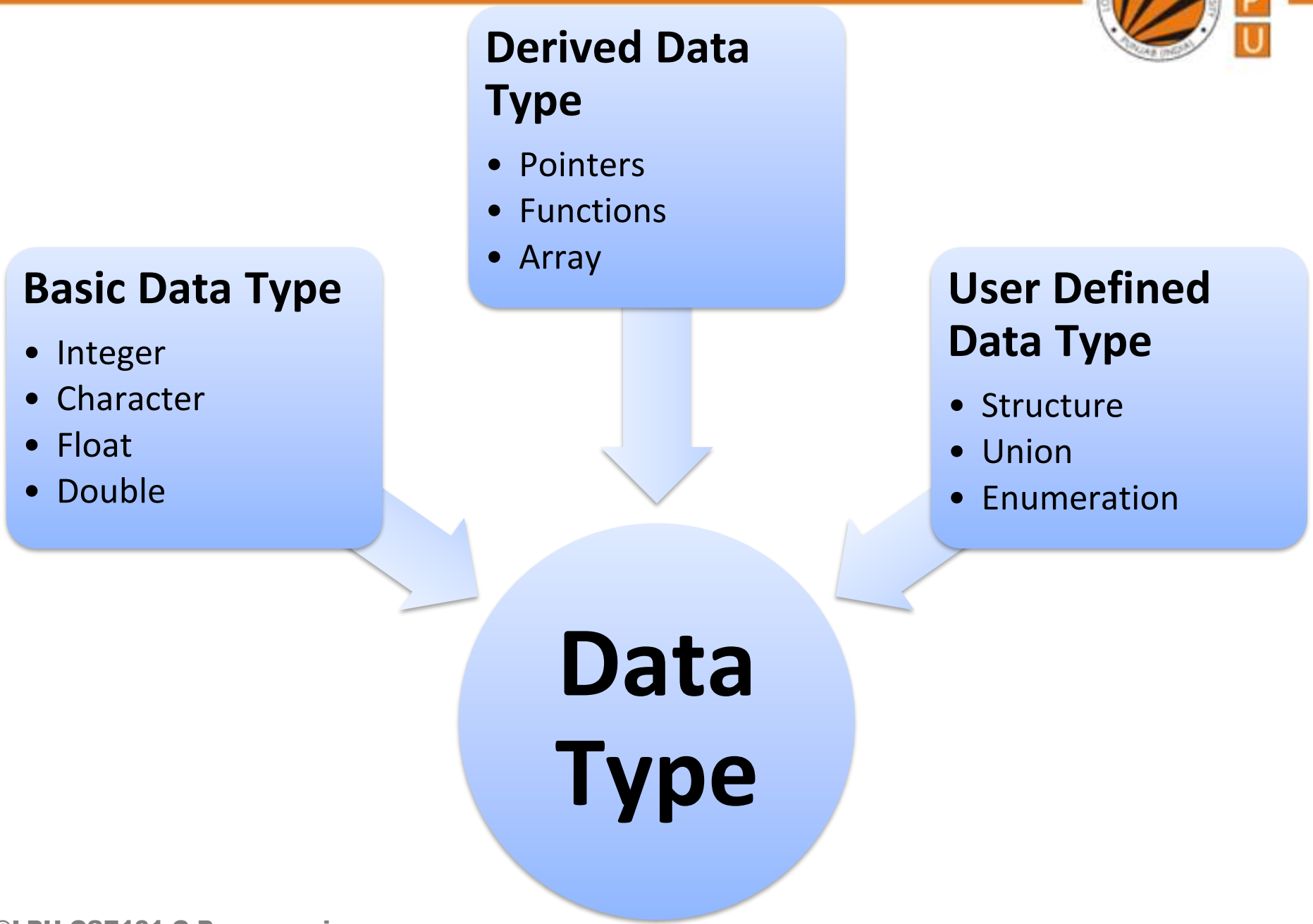
M4=58, G4=?

M5=64, G5=?



Classification of Data Types

- In C data type is broadly classified as:
  - Basic data types
  - Derived data types
  - User defined data types


Derived Data Type
• Pointers
• Functions
• Array
Basic Data Type
• Integer
• Character
• Float
• Double
User Defined Data Type
• Structure
• Union
• Enumeration
Data Type

List of Data Types

(Size of the data type depends upon the compiler also, following sizes may vary also, as per different compilers)

Type	Size (bytes)	Minimal range
char	1	-128 to 127
unsigned char	1	0 to 255
int	2 or 4	-32768 to 32767
unsigned int	2 or 4	0 to 65535
short int	2	-32768 to 32767
unsigned short int	2	0 to 65535
long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
float	4	3.4e-38 to 3.4e+38 with 6 digits of precision
double	8	1.7e-308 to 1.7e+308 with 15 digits of precision
long double	10 or 12 or 16	3.4e-4932 to 1.1e+4932 with 20 digits of precision

Integer

- It is used to store positive and negative counting numbers, as well as zero.

{...,-2,-1,0,1,2,...}

- The numbers written in green box of My-Car problem are the integers.

- The **type modifiers** for the integer data type are: signed, unsigned, short, long .
- Signed types represent positive and negative numbers.
- Unsigned represent zero and positive numbers only.
- Long and short represent the range of integer number

Short Integer

- Occupies 2 bytes in memory.
- Format specifier is %d or %i.
- Range is -32768 to 32767.
- By default int variable is short signed int.

```
int cost=100;
short int si;
```

Long Integer

- Occupies 4 bytes in memory.
- Format specifier is %ld.
- Range is -2147483648 to 2147483647

```
long radius=123456;
long int value;
```

Signed Integer

- Occupies 2 bytes in memory
- Format specifier is %d or %i
- There are also long signed integers having range from -2147483648 to 2147483647
- Example:

```
int firstvalue=10;
long int WaterLevel;
```

Unsigned Integer

- Occupies 2 bytes in memory
- Format specifier is %u.
- There are also long unsigned int with range 0 to 4294967295
- Example:

```
unsigned long count=567898;
unsigned short int page;
```

Float

- Floating point numbers are real numbers that, unlike integers, may contain fractional parts of numbers, like 1.446, -112.972, 3.267e+27.
- It is used to store real numbers with single precision i.e. a precision of 6 digits after decimal point.

- Format specifier is **%f.**
- The **type modifier** for float are float, double and long double.

- The rational number written in red box of My-Car problem are the float numbers.

3.14 94.2

Type	Float	Double	Long double
Storage Size	4 bytes	8 bytes	10 bytes/or 16 bytes
Value range	3.4e-38 to 3.4e+38	1.7e-308 to 1.7e+308	3.4e-4932 to 1.1e+4932
Precision	6 decimal places	15 decimal places	20 decimal places
Example	pi=3.141592	3.141592741012573	3.141592653589793238 46

Character

- It stores a single character of data belonging to the C character set.

- The alphabets written in blue box of My-Grades problem are the character.

A	D	A	B	C

- It occupies 1 byte of memory.
- Format specifier is **%c**.
- The range is 0 to 255 for unsigned char.
- The range is -128 to 127 for signed char.
- Each char type has an equivalent integer interpretation, ASCII value, so that a char is really a special kind of short integer.

  char choice='y';

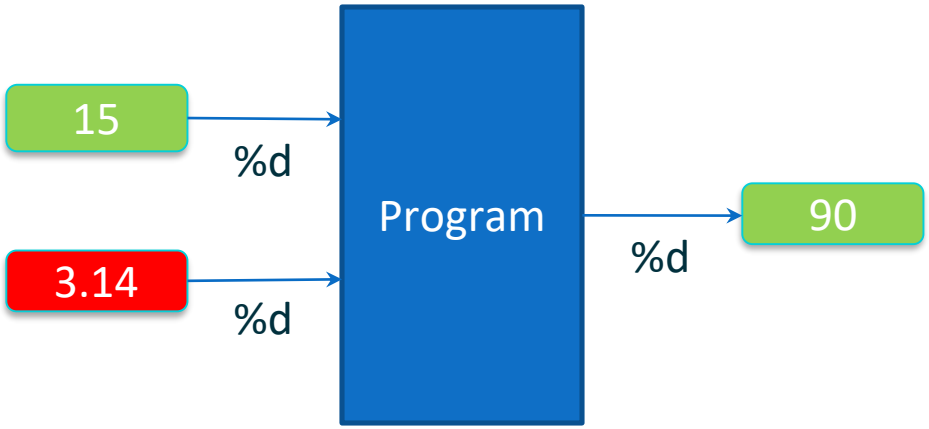
Format Specifier

- Specifies the format according to which the value will be printed on screen in C.

Example:

- %d or %i : signed integer
- %ld: long integer
- %u : unsigned integer
- %c : single character
- %f or %g : float
- %lf: double
- %Lf: long double
- %s : string

- Remember car example?



My-Car

1. If the radius of car wheel is 15inch then what will the distance traveled after one rotation of that wheel?

Sol: Given-

radius = 15 inch — 15 Integer(int in C)

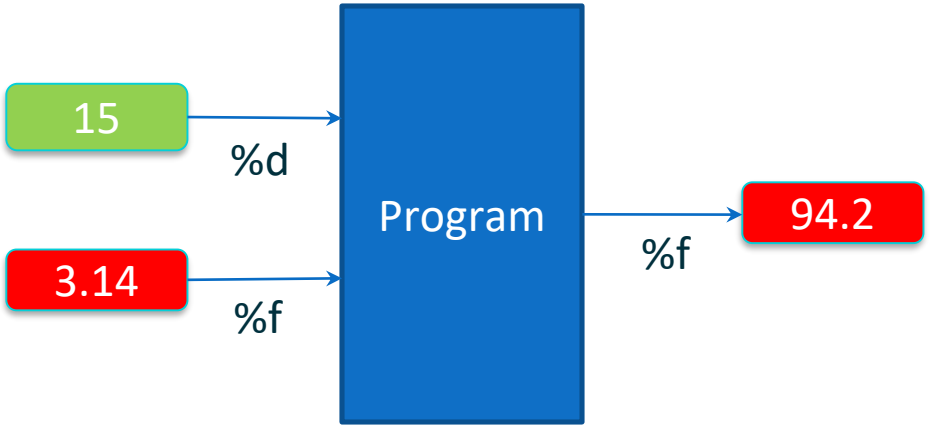
dist_travelled = ?

So, Circumference of circle = 2 * pi * radius

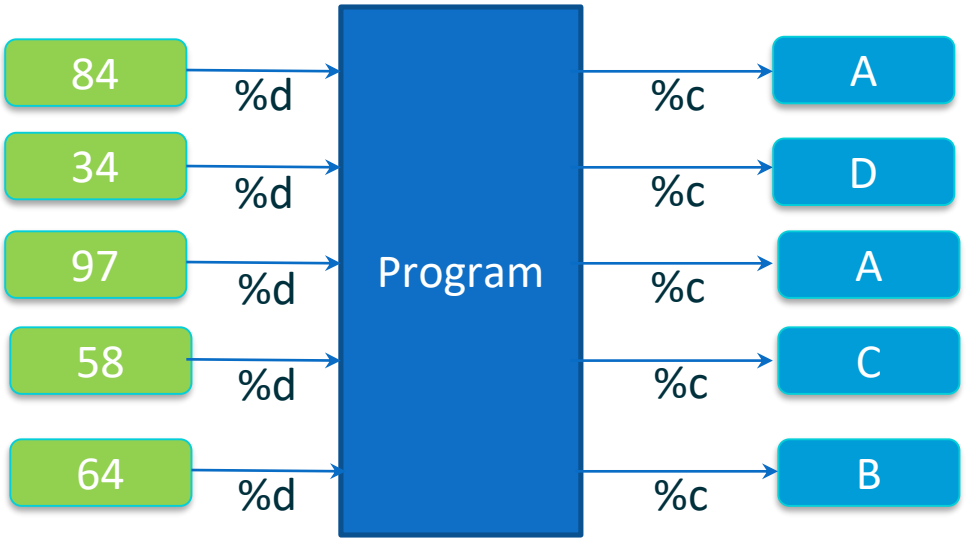
dist_travelled = 2 * 3.14 * radius — 3.14 Real (float in C)

dist_travelled = 6.28 * 15

dist_travelled = 94.2 inch Ans. — 94.2 Real (float in C)



- Grade example:



2. Five students have appeared for Mathematics exam and their respective marks are

   84,34,97,58,64

consider the rank bands and their respective grades as

80 to 100 – A
60 to 79 – B
40 to 59 – C
less than 40 – D

So find the grade for each students?

Q1

Which of the following is not a basic data type in C language?
a) float
b) int
c) real
d) char

Q2

The format identifier ‘%i’ is also used for _____ data type.
a) char
b) int
c) float
d) double

Q3

In a C program, following variables are defined:

float x = 2.17;

double y = 2.17;

long double z = 2.17;

Which of the following is correct way for printing these variables via printf.

A. printf("%f %lf %Lf",x,y,z);

B. printf(“%f %f %f”,x,y,z);

C. printf("%f %ff %fff",x,y,z);

D. printf("%f %lf %llf",x,y,z);

Q4

What will be the output of the following C code?

```
#include  <stdio.h>
int main()
{
   signed char chr;
   chr = 128;
   printf("%d\n", chr);
   return 0;
}
```

a) 128

b) -128

c) Depends on the compiler

d) None of the mentioned

Q5

Which is correct with respect to the size of the data types?

a) char > int > float

b) int > char > float

c) char < int < double

d) double > char > int

Next Lecture: Constants
Variables
Expressions

CSE101-Lec#2

- Constant
- Variable
- Expression

Outline

- In this lecture we will study
  - Constant
  - Variable
  - Expression

Tokens

- We have seen that Tokens are broadly classified as:
  - Identifiers
  - Keywords
  - **Constants**
  - **Variables**
  - Strings
  - Operators
  - Special character

Constants

$=$ Lion $\neq$

$=$ Spanner $\neq$

Constants

- The entity which do not change throughout the execution are called constants.
- Types of constants:
  - Integer constant
  - Character constant
  - Floating point constants
  - String constants

Name of person remains same through out the life, example: Amit, Shubnam, etc.

- **Integer Constants**
  - When the constant contains only digits without any decimal part

Example : 5;
-987;

- **Floating Constant**
  - Constants that contains number with decimal points

Example : 3.14;
309.89

- **Character constants**
  - Constants enclosed in single quotes(‘ ’).
  - It can be any letter from character set.

Example : ‘\n’, ‘\t’ or ‘a’

- **String Constants**
  - Set of zero or more characters enclosed in double quotes (eg: “ ”)
  - It is represented as sequence of characters within double quotes.

Example : “This is C programming”

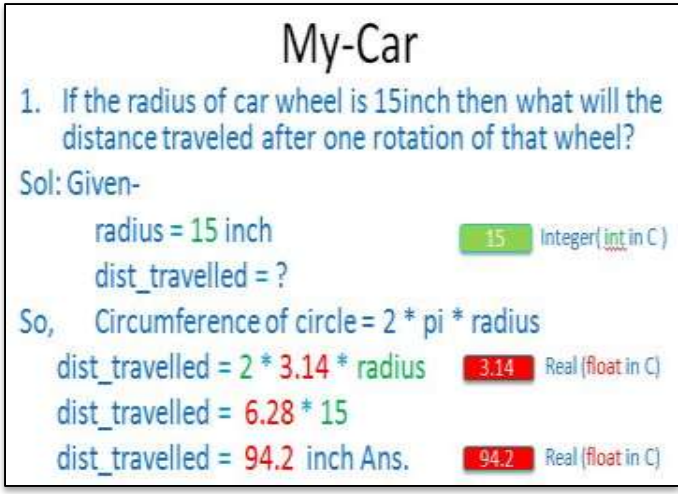
My-Car

In My-Car problem the constant value is 3.14 which is the value of pi and always same.

- pi = 3.14

Therefore:

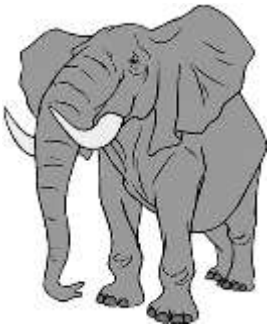
dist_travelled = 2 * pi * radius.



➢ pi is a floating point constant.

Variables

Animal =



Tool =

Variables

- Variable is an entity which may change.
- Variable is used to hold result and reserve memory for the data.

Syntax

```
datatype variable_name;
```

The naming of variable is done by following the same rules of identifier naming.

Eg. What is your **hobby**?
The answer could be **reading**, **dancing**, **drawing**, etc.
So the answer to such questions may change during the life time of the person

Rules for naming a Variable

1. An variable name is any combination of 1 to 31 alphabets, digits or underscores.

2. The first character in the variable name must be an alphabet or underscore.

3. No blanks or special symbol other than an underscore can be used in an variable name.

4. Keywords are not allowed to be used as variables.

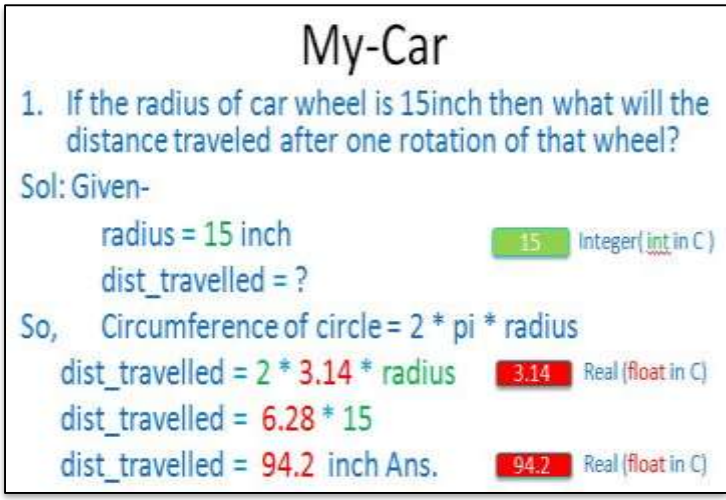
Variables

In My-Car problem the variable was

- radius and dist_travelled

It can also be named as

- radius_wheel or r1 or car_wheel_radius
- Distance or d1 or dist_by_1rotation



Variables

Let us build some variables:

For speed of car we need to know

- Distance traveled
- Time taken to travel the distance

$$s1 = d1 \div t1$$

Variables to be declared as

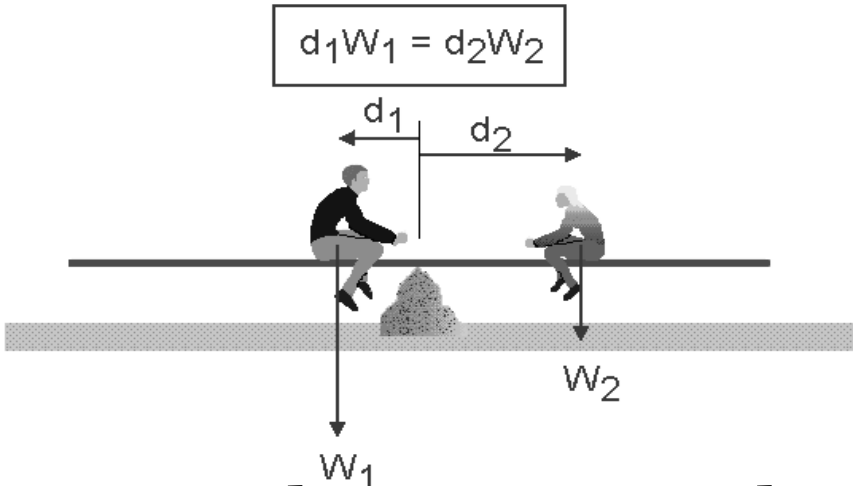
- Speed, **s1**, speed_of_car
- Distance, **d1**, dist
- Time, **t1**, time_of_travel

See-Saw

- A bit more complex situation see-saw

Variables to be declared as

- Weight_boy, **w1**, wb
- Distance_boy, **d1**, db
- Weight_girl, **w2**, wg
- Distance_girl, **d2**, dg



$$w1 \times d1 = w2 \times d2$$

$$wb \times db = wg \times dg$$

- It is to be assessed that at what distance 50Kg girl should sit in order to balance a boy of 70Kg sitting 2m away from the center ‘o’

Variable Initialization

- Assigning some value to the variable at time of creation of variable is known as variable initialization.

Syntax

```
datatype variable_name =  value;
```

Example: int radius= 15;
float pi = 3.14;
char grade = 'A';

Expressions

- Expressions are the statements or the instruction given to computer to perform some operation.
- Every expression results in some value that can be stored in a variable.
- Following are few example of expressions in program:
  - Expression to calculate speed of a car.
    - Speed=distance/time
  - Expression to find similarity of two things.
    - c=value1>value2

- Expressions in C are basically **operators** acting on **operands**.
- An **operand** is an entity on which operation is to be performed.

Example: addition of two numbers, 5+8, these numbers will be operands.

- An **operator** specifies the operation to be applied on operands.

Example: The addition, subtraction, etc will be operators

- Expressions are made of one or more operands.
- Statements like :

  a = b + c,

  ++z

  300 > (8 * k)

Types of Expressions

- The type of expression depend upon the type of operator used in the expression.
- It can be:
  - Arithmetic operators.

    3 + 6 = 9

    4 * 2 = 8
  - Relational or logical operators.

    height_boy>=height_girl
  - Increment and decrement operator.

    count=count++

CSE101-Lec#3

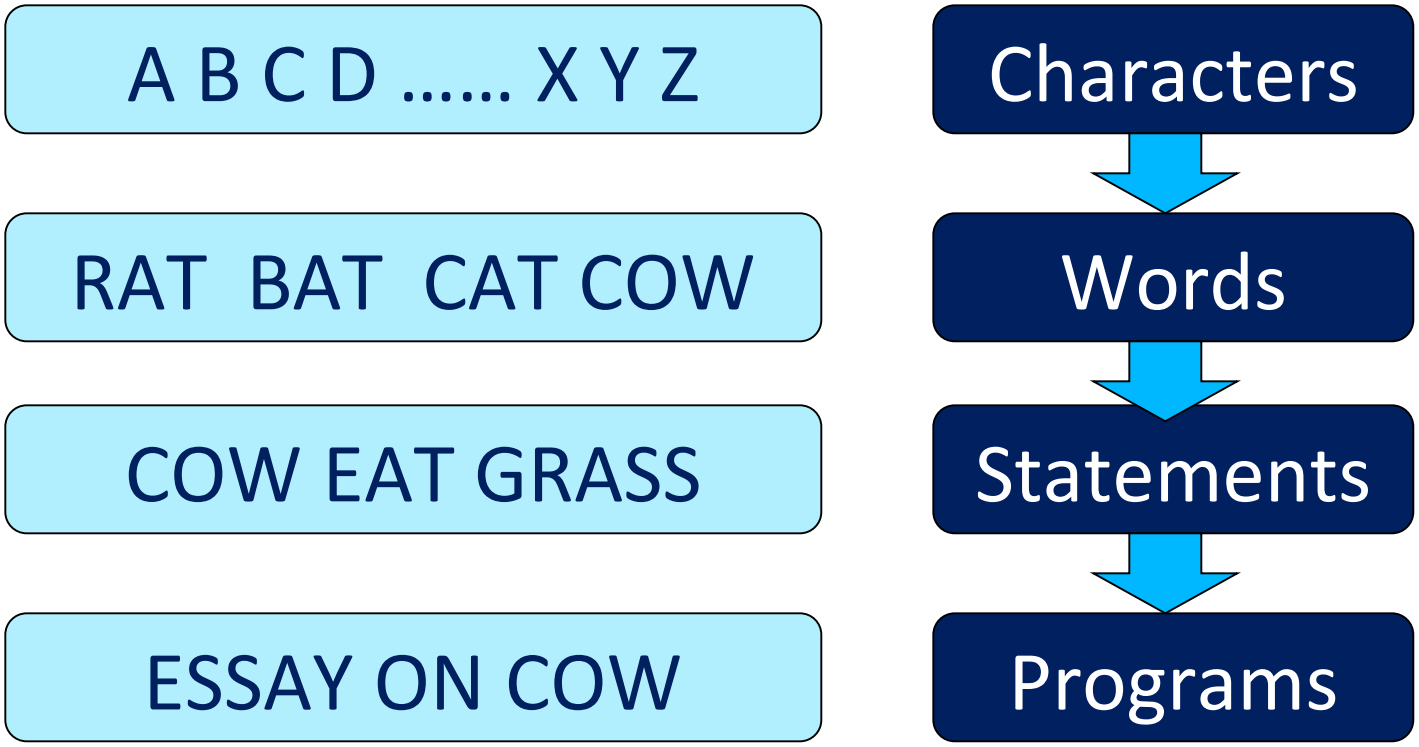
Character Set
Identifiers and Keywords
Data types

OUTLINE

- In this lecture we will cover
  - Character set
  - Identifiers
  - Keyword
  - Data types

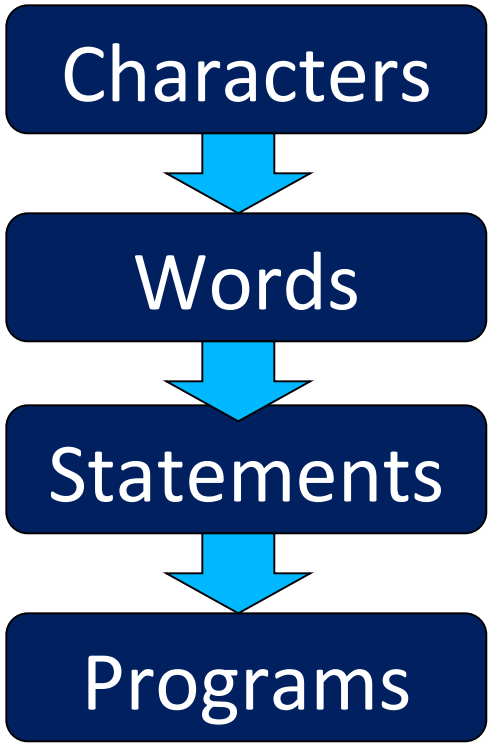
Language: its influence in our life

- Let us look to what we are doing since our childhood, how did we learnt ENGLISH- A recap



Introduction to C

- Like every language C programming language requires basic building blocks to communicate with the computer.
- So we require
  - Character set
  - Words(keywords and identifiers)
  - Statement (instructions)
  - Program



Character Set

- The character set of C represents alphabet, digit or any symbol used to represent information.

Types	**Character Set**
Uppercase Alphabets	A, B, C, … Y, Z
Lowercase Alphabets	a, b, c, … y, z
Digits	0, 1, 2, 3, … 9
Special Symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /
White spaces	Single space, tab, new line.

Meaningfulness

- Let us look to some words
- **saslc, enp, keib, rac, llab**

- Rearrange
- **Class, pen, bike, car, ball**

- This is the influence of adding meaning by logical and sensible grouping in mode of communication through language

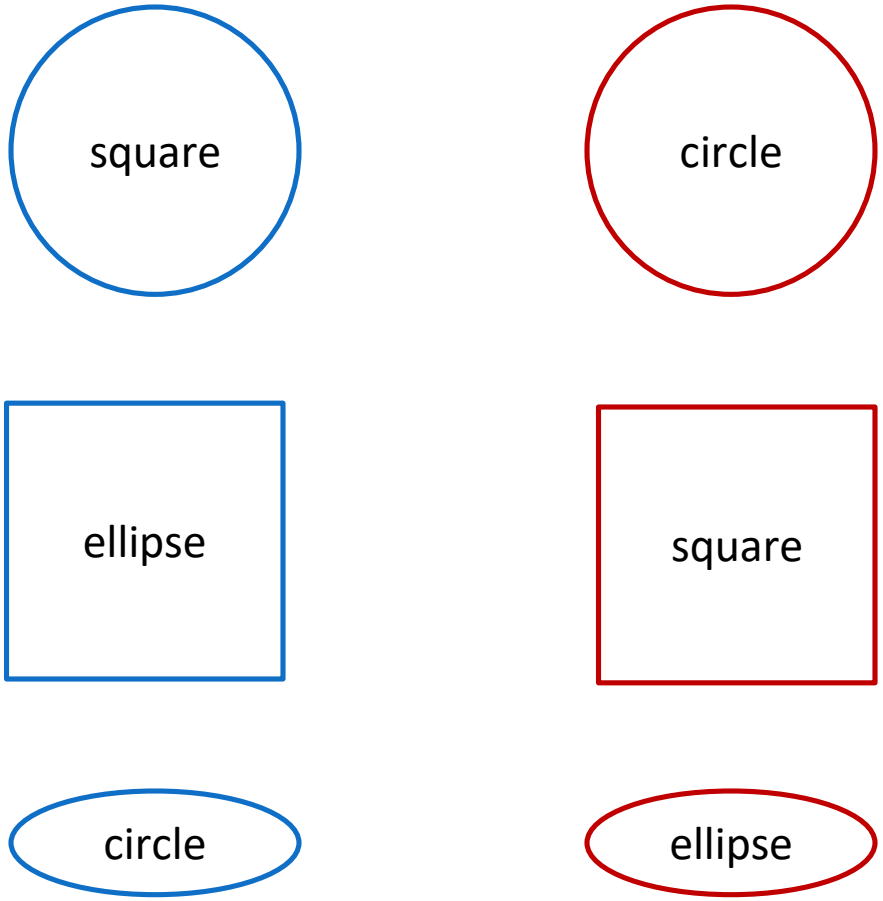
Token

- Every single element in a C Program is Token

Token

- Smallest unit in a program/statement.
- It makes the compiler understand what is written in the program.
- Example: main, printf , name,), etc.
- Tokens are broadly classified as:
  - Identifiers
  - Keywords
  - Constants
  - Variables
  - Strings
  - Operators
  - Special character

Lets Identify the following:



Identifiers

- So to identify things we have some name given to them .
- Identifiers are the fundamental building blocks of a program
- Used to give **names** to variables, functions, constant, and user defined data.
- They are user-defined names and consist of a sequence of letters and digits

Rules for naming an Identifier

1. An identifier name is any combination of 1 to 31 alphabets, digits or underscores.

2. The first character in the identifier name must be an alphabet or underscore.

3. No blanks or special symbol other than an underscore can be used in an identifier name.

4. Keywords are not allowed to be used as identifiers.

Some Identifiers

Tool_spanner;
tool_spanner;

both are different

FORMULA1;

engine_1;

Wrong identifiers name

1_engine;

break;

@car-roof;

C Keywords

- Keywords are the reserved words whose meaning has already been explained to the C compiler.
- We cannot use these keywords as variables.
- Each keyword is meant to perform a specific function in a C program.
- There are 32 keywords in C language.
- All keywords are written in lowercase only

Eg: The **name** of person can never be **home**, **eat**, **sleep**, **run**, etc because these words have some predefined meaning to perform some task.

List of C 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

Which of the following is a valid identifier name?

A. 1abcpqr
B. break
C. abc_pqr
D. ^abc

Which of the following is not a valid identifier?

a) __a3

b) __3a

c) __A3

d) None of the mentioned

Which of the following is not a valid identifier?

a) _a3;

b) a_3;

c) 3_a;

d) _3a;

Data Types

- Data type means the type of value a variable will have.
- It also defines memory space for a particular variable in computer.
- The type of value of variable can be alphabets or numbers.
- The numbers can be further divided as the integer or rational number.

- Lets see a mathematics problem:

My-Car

1. If the radius of car wheel is 15inch then what will the distance traveled after one rotation of that wheel?

Sol: Given-

radius = 15 — 15 — Integer(int in C)

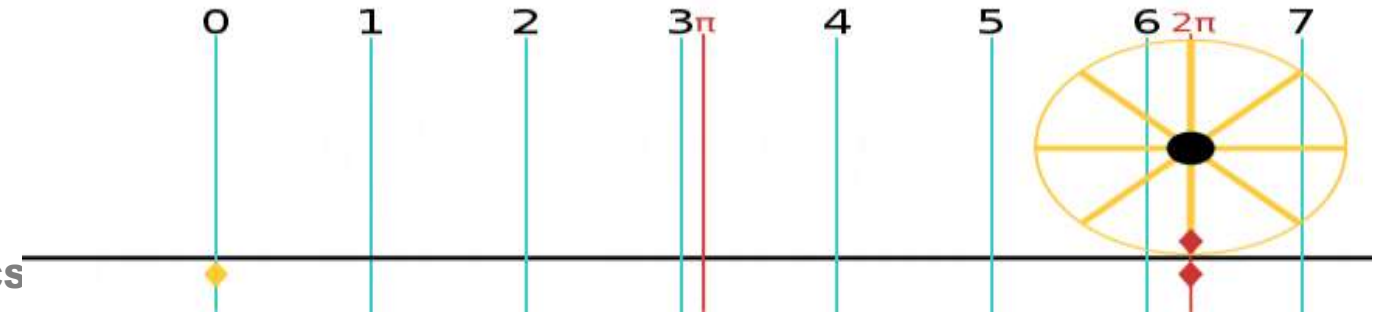
dist_travelled = ?

So, Circumference of circle = 2 * pi * r

dist_travelled = 2 * 3.14 * radius — 3.14 — Real (float in C)

dist_travelled = 6.28 * 15

dist_travelled = 94.2 Ans. — 94.2 — Real (float in C)



My-Grades

2. Five students have appeared for Mathematics exam and their respective marks are

   84,34,97,58,64

consider the rank bands and their respective grades as

80 to 100 – A
60 to 79 – B
40 to 59 – C
less than 40 – D

So find the grade for each students?

Sol: Given-

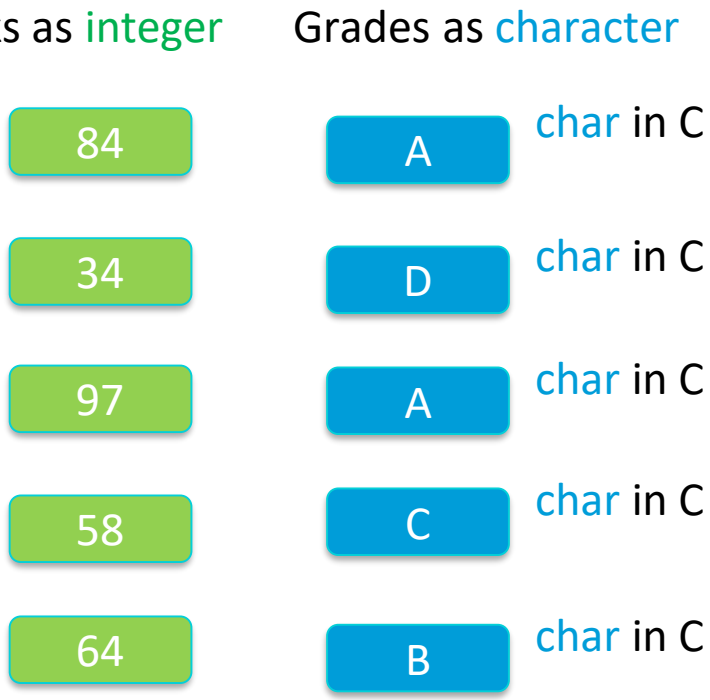
M1=84, G1=?

M2=34, G2=?

M3=97, G3=?

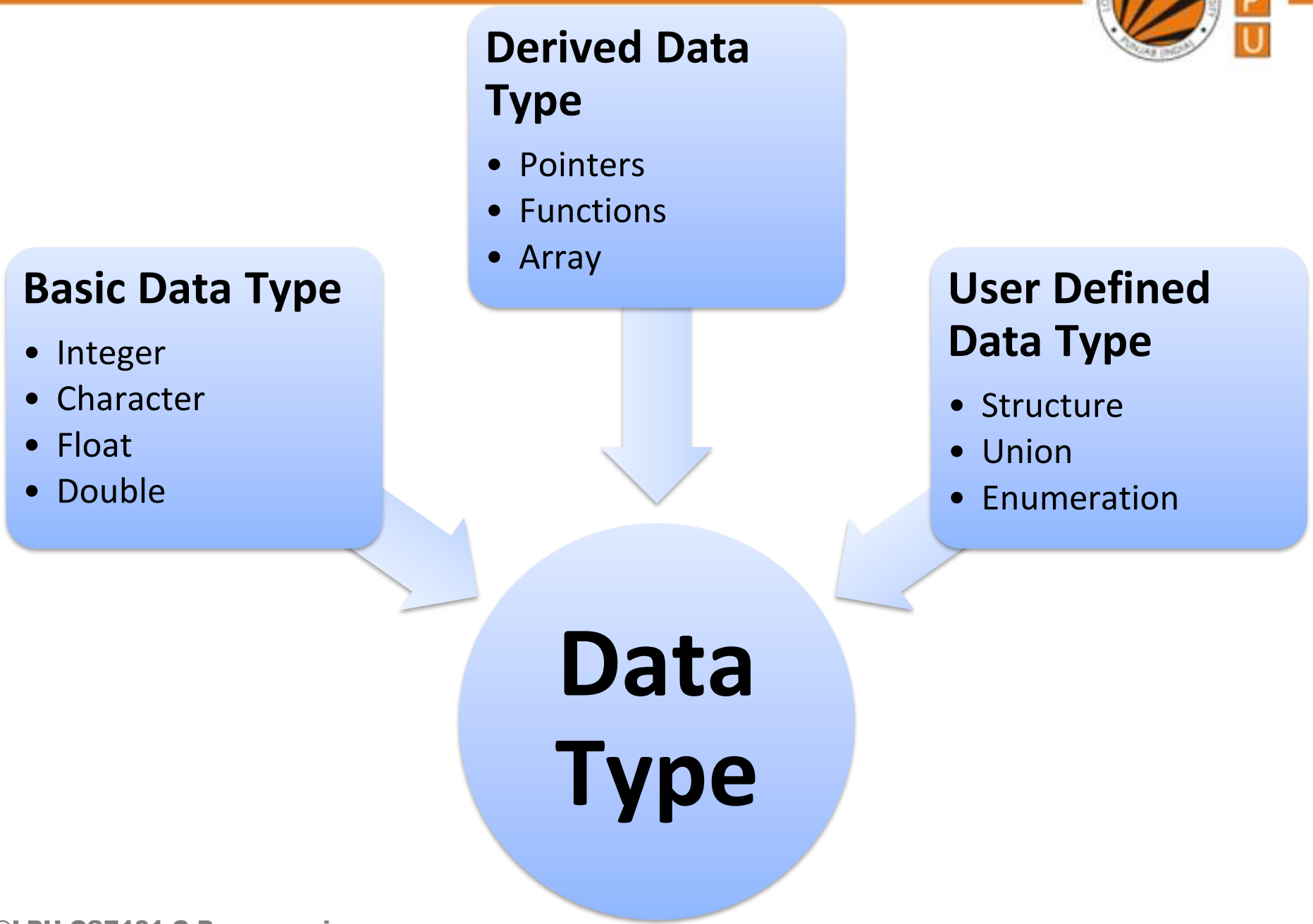
M4=58, G4=?

M5=64, G5=?



Classification of Data Types

- In C data type is broadly classified as:
  - Basic data types
  - Derived data types
  - User defined data types


Derived Data Type
• Pointers
• Functions
• Array
Basic Data Type
• Integer
• Character
• Float
• Double
User Defined Data Type
• Structure
• Union
• Enumeration
Data Type

List of Data Types

(Size of the data type depends upon the compiler also, following sizes may vary also, as per different compilers)

Type	Size (bytes)	Minimal range
char	1	-128 to 127
unsigned char	1	0 to 255
int	2 or 4	-32768 to 32767
unsigned int	2 or 4	0 to 65535
short int	2	-32768 to 32767
unsigned short int	2	0 to 65535
long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
float	4	3.4e-38 to 3.4e+38 with 6 digits of precision
double	8	1.7e-308 to 1.7e+308 with 15 digits of precision
long double	10 or 12 or 16	3.4e-4932 to 1.1e+4932 with 20 digits of precision

Integer

- It is used to store positive and negative counting numbers, as well as zero.

{...,-2,-1,0,1,2,...}

- The numbers written in green box of My-Car problem are the integers.

- The **type modifiers** for the integer data type are: signed, unsigned, short, long .
- Signed types represent positive and negative numbers.
- Unsigned represent zero and positive numbers only.
- Long and short represent the range of integer number

Short Integer

- Occupies 2 bytes in memory.
- Format specifier is %d or %i.
- Range is -32768 to 32767.
- By default int variable is short signed int.

```
int cost=100;
short int si;
```

Long Integer

- Occupies 4 bytes in memory.
- Format specifier is %ld.
- Range is -2147483648 to 2147483647

```
long radius=123456;
long int value;
```

Signed Integer

- Occupies 2 bytes in memory
- Format specifier is %d or %i
- There are also long signed integers having range from -2147483648 to 2147483647
- Example:

```
int firstvalue=10;
long int WaterLevel;
```

Unsigned Integer

- Occupies 2 bytes in memory
- Format specifier is %u.
- There are also long unsigned int with range 0 to 4294967295
- Example:

```
unsigned long count=567898;
unsigned short int page;
```

Float

- Floating point numbers are real numbers that, unlike integers, may contain fractional parts of numbers, like 1.446, -112.972, 3.267e+27.
- It is used to store real numbers with single precision i.e. a precision of 6 digits after decimal point.

- Format specifier is **%f.**
- The **type modifier** for float are float, double and long double.

- The rational number written in red box of My-Car problem are the float numbers.

3.14 94.2

Type	Float	Double	Long double
Storage Size	4 bytes	8 bytes	10 bytes/or 16 bytes
Value range	3.4e-38 to 3.4e+38	1.7e-308 to 1.7e+308	3.4e-4932 to 1.1e+4932
Precision	6 decimal places	15 decimal places	20 decimal places
Example	pi=3.141592	3.141592741012573	3.141592653589793238 46

Character

- It stores a single character of data belonging to the C character set.

- The alphabets written in blue box of My-Grades problem are the character.

A	D	A	B	C

- It occupies 1 byte of memory.
- Format specifier is **%c**.
- The range is 0 to 255 for unsigned char.
- The range is -128 to 127 for signed char.
- Each char type has an equivalent integer interpretation, ASCII value, so that a char is really a special kind of short integer.

  char choice='y';

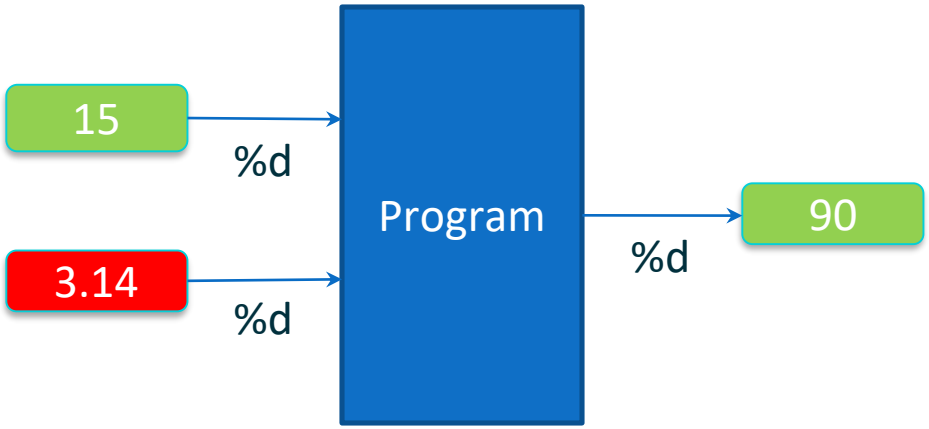
Format Specifier

- Specifies the format according to which the value will be printed on screen in C.

Example:

- %d or %i : signed integer
- %ld: long integer
- %u : unsigned integer
- %c : single character
- %f or %g : float
- %lf: double
- %Lf: long double
- %s : string

- Remember car example?



My-Car

1. If the radius of car wheel is 15inch then what will the distance traveled after one rotation of that wheel?

Sol: Given-

radius = 15 inch — 15 Integer(int in C)

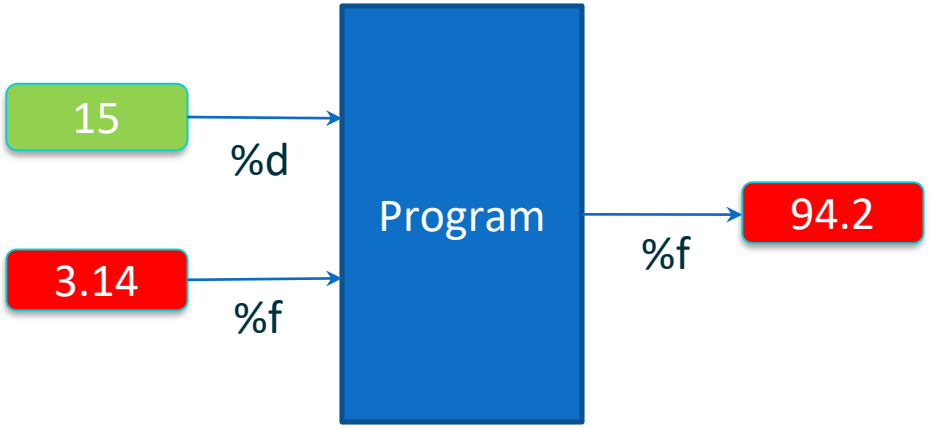
dist_travelled = ?

So, Circumference of circle = 2 * pi * radius

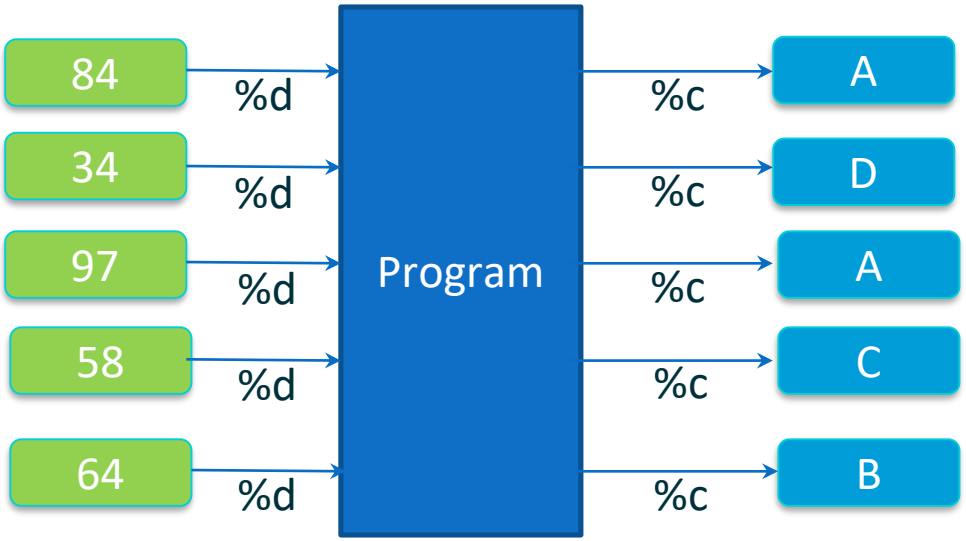
dist_travelled = 2 * 3.14 * radius — 3.14 Real (float in C)

dist_travelled = 6.28 * 15

dist_travelled = 94.2 inch Ans. — 94.2 Real (float in C)



- Grade example:



2. Five students have appeared for Mathematics exam and their respective marks are

   84,34,97,58,64

consider the rank bands and their respective grades as

80 to 100 – A
60 to 79 – B
40 to 59 – C
less than 40 – D

So find the grade for each students?

Q1

Which of the following is not a basic data type in C language?
a) float
b) int
c) real
d) char

Q2

The format identifier ‘%i’ is also used for _____ data type.
a) char
b) int
c) float
d) double

Q3

In a C program, following variables are defined:

float x = 2.17;

double y = 2.17;

long double z = 2.17;

Which of the following is correct way for printing these variables via printf.

A. printf("%f %lf %Lf",x,y,z);

B. printf(“%f %f %f”,x,y,z);

C. printf("%f %ff %fff",x,y,z);

D. printf("%f %lf %llf",x,y,z);

Q4

What will be the output of the following C code?

```
#include  <stdio.h>
int main()
{
   signed char chr;
   chr = 128;
   printf("%d\n", chr);
   return 0;
}
```

a) 128

b) -128

c) Depends on the compiler

d) None of the mentioned

Q5

Which is correct with respect to the size of the data types?

a) char > int > float

b) int > char > float

c) char < int < double

d) double > char > int

Next Lecture: Constants
Variables
Expressions

Operator	Description	Associativity
( ) [ ] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

CSE101-Lec#2-First Part

- Operators

- In this lecture we will study
  - Operators
  - Types of Operators

Operators

- Operator is the symbol which performs some operations on the operands.

5+5=10

+ and = are the operator and
5 and 10 are operands

Types of Operators

- **Types of operators are:**
  1. Arithmetic operator
  2. Unary operator
  3. Relational operator
  4. Logical operator
  5. Assignment operator
  6. Conditional operator
  7. Bitwise operator
  8. Special operator

Description of Operators

➢ Arithmetic Operators

These are binary operators i.e. expression requires two operands

Operator	Description	Example (a=4 and b=2)
+	Addition of two operands	a + b = 6
-	Subtraction of two operands	a – b = 2
*	Multiplication of two operands	a * b = 8
/	Division of two operands	a / b = 2
%	Modulus gives the remainder after division of two operands	a % b = 0

Arithmetic Operators

If the radius of car wheel is 15inch then what will the diameter and calculate distance traveled after one rotation of that wheel?

Arithmetic Operators

Sol:

r = 15

diameter = r + r = 2 * r = 2 * 15 = 30

dist_travelled = pi * d

dist_travelled = pi * diameter

= 3.14 * 30 = 94.2

Arithmetic Operators

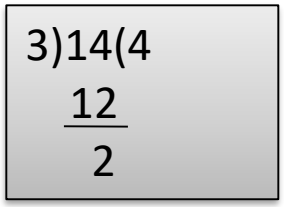
To get the remainder of the integer value.

Eg:

14 mod 3 = 2

17 mod 2 = 1

190 mod 3 = 1



Q:Suppose we have to distribute 10 chocolates among 3 students equally then after equal distribution how many chocolates will be left?

Sol: 10 mod 3 = 1

So 1 chocolate will be left as all 3 students will have 3 chocolates each.

Q1

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = -3;
    int k = i % 2;
    printf("%d\n", k);
    return 0;
}
```

A. Compile time error

B. -1

C. 1

D. None of these

Q2

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 3;
    int l = i / -2;
    int k = i % -2;
    printf("%d %d\n", l, k);
    return 0;
}
```

A. Compile time error

B. -1 1

C. 1 -1

D. None of these

Q3

What will be the final value of x in the following C code?

```
#include <stdio.h>
int main()
{
    int x = 5 * 9 / 3 + 9;
    printf("%d",x);
    return 0;
}
```

A. 3.75

B. Depends on compiler

C. 24

D. 3

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 5.3 % 2;
    printf("Value of x is %d", x);
    return 0;
}
```

A. Value of x is 2.3

B. Value of x is 1

C. Value of x is 0.3

D. Compile time error

Q5

What will be the output of the following C code?

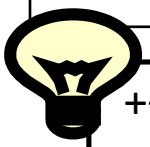
```
#include <stdio.h>
int main()
{
    int a = 10;
    double b = 5.6;
    int c;
    c = a + b;
    printf("%d", c);
    return 0;
}
```

A. 15
B. 16
C. 15.6
D. 10

➢Unary Operator

These operator requires only one operand.

Operator	Description	Example(count=1)
+	unary plus is used to show positive value	+count; value is 1
-	unary minus negates the value of operand	-count; value is -1
++	Increment operator is used to increase the operand value by 1	++count; value is 2 count++; value is 2
--	Decrement operator is used to decrease the operand value by 1	--count; value is 1 count--; value is 1



++count increments count by 1 and then uses its value as the value of the expression. This is known a **prefix operator**.
count++ uses count as the value of the expression and then increments count by 1. This is known as **postfix operator**.

Difference between Prefix and Postfix

- Unary Prefix increment/ decrement performs the operation first, and then the value is assigned/ or used

Example:

Consider x=2, then

y = ++x; is equivalent to writing

//x = x + 1;

//y = x;

So eventually x will be incremented by 1, i.e x will become 3, and then the value 3 will be assigned to y

- Unary Postfix increment/ decrement will assign/ or use the value first and then the operation is performed

Example:

Consider x=2, then

y = x++; is equivalent to writing

//y = x;

//x=x+1;

Here y will take value 2, and then the value of x will be increment by 1, and x becomes 3

Difference between Prefix and Postfix

- Example:

```
#include<stdio.h>
int main() {
  int x = 3, y, z;
  y = x++;
  z = ++x;
  printf("\n%d,%d,%d",x,y,z);
  return 0;
}
```

Output:

5, 3, 5

Explanation:

- Initialize x to 3
- Assign y the value we get by evaluating the expression x++, i.e, the value of x before increment then increment x.
- Increment x then assign z the value we get by evaluating the expression ++x, i.e, value of x after the increment.
- Print these values

Q1

What will be the output of following code?

```
#include <stdio.h>
    int main()
    {
        int a=1,b=1,c;
        c = a++ + b;
        printf("%d,%d,%d", a,b,c);
        return 0;
    }
```

A. 2,1,1

B. 1,2,1

C. 2,1,2

D. 1,1,2

Q2

What will be the output of following code?

```
#include <stdio.h>
   int main()
   {
      int d, a = 1, b = 2;
      d =  a++ + ++b;
      printf("%d %d %d", d, a, b);
      return 0;
   }
```

A. 4 2 2
B. 3 1 2
C. 4 2 3
D. 3 2 3

Q3

What will be the output of following code?

```
#include <stdio.h>
   int main()
   {
      int i = 0;
      int x = i++;
      int y = ++i;
      printf("%d % d\n", x, y);
      return 0;
   }
```

A. 0, 2
B. 0, 1
C. 1, 2
D. 1, 1

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 4, y, z;
    y = --x;
    z = x--;
    printf("%d%d%d", x,  y, z);
    return 0;
}
```

A. 3 2 3

B. 2 3 3

C. 3 2 2

D. 2 3 4

Relational Operator

It compares two operands depending upon the their relation. Expression generates zero(false) or nonzero(true) value.

Operator	Description	Example (a=10 and b=20)
<	less than, checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) value is 1(true)
<=	less than or equal to, checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) value is 1 (true).
>	greater than, checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) value is 0 (false).
>=	greater than or equal to, checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) value is 0 (false).
==	equality ,checks if the value of two operands is equal or not, if yes then condition becomes true.	(a == b) value is 0 (false).
!=	inequality, checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a != b) value is 1 (true).

Relational Operator

Q: Age of Sam is 20 and age of Tom is 19.

Verify the relationship between their age.

Sol: age of Sam = S1 = 20

age of Tom = T1 = 19

S1 < T1 = 0 (false)

S1 > T1 = 1 (true)

So, Sam is elder than Tom.

S1 == T1 = 0 (false)

Q1

What will be the output of following code?

```
#include <stdio.h>
    int main()
    {
        int a=1,b=2,c;
        c=a>b;
        printf("\n%d",c);
        return 0;
    }
```

A. 0

B. 1

C. 2

D. None of these

Q2

What will be the output of following code?

```
#include <stdio.h>
   int main()
   {
      int a=1,b=2;
      printf("\n%d",a!=b);
      return 0;
   }
```

A. 0

B. 1

C. 2

D. None of these

Q3

What will be the final value of d in the following C code?

```
#include <stdio.h>
int main()
{
    int a = 10, b = 5, c = 5;
    int d;
    d = b + c == a;
    printf("%d", d);
    return 0;
}
```

A. Syntax error

B. 1

C. 5

D. 10

Logical Operator

It checks the logical relationship between two expressions and the result is zero(false) or nonzero(true).

Operator	Description	Example
&&	Logical AND operator. If both the operands are true then condition becomes true.	(5>3 && 5<10) value is 1 (true).
||	Logical OR Operator. If any of the two operands is true then condition becomes true.	(5>3 || 5<2) value is 1 (true).
!	Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(8==8) value is 0 (false).

Logical Operator

Grade system :

```
If (Marks >=90 || marks == 100)
        students performance is excellent.
If (Marks <= 40 && attendance < 75)
        student is detained.
```

Q1

```
//What will be the output of following code?
   #include <stdio.h>
   int main()
   {
      int a = 10, b = 0,c;
      c=a&&b;
      printf("%d",c);
   }
```

A. 0

B. 1

C. -1

D. None of these

Q2

What will be the output of following code?

```
#include <stdio.h>
int main()
{
    int a = 10, b = 0,c=2,d;
    d=a&&b||c-2;
    printf("%d",d);
}
```

A. 0

B. 1

C. -1

D. None of these

Q3

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 1, y = 0, z = 5;
    int a = x && y || z++;
    printf("%d", z);
    return 0;
}
```

A. 6
B. 5
C. 0
D. None of these

Q4

What will be the output of following code?

```
#include <stdio.h>
int main()
{
    int x = 1, y = 0, z = 5;
    int a = x && y && z++;
    printf("%d", z);
    return 0;
}
```

A. 6
B. 5
C. 0
D. None of these

➢Assignment Operator

They are used to assign the result of an expression on right side to a variable on left side.

Operator	Description	Example(a=4 and b=2)
+=	a=a+b	a+=b; a=a+b = 6
-=	a=a-b	a-=b; a=a-b = 2
*=	a=a*b	a*=b; a=a*b = 8
/=	a=a/b	a/=b; a=a/b = 2
%=	a=a%b	a%=b; a=a%b = 0
<<=	a=a<<b	a=00000100 << 2 = 00010000
>>=	a=a>>b	a=00000100 >> 2 = 00000001
&=	a=a&b	(a=0100, b=0010) a&=b; a=a&b = 0000
|=	a=a|b	(a=0100, b=0010) a|=b; a=a|b =0110
^=	a=a^b	(a=0100, b=0010) a^=b; a=a^b = 0110

Assignment Operator

- To increase the cost of item soap by 50rs.

  Cost_soap = Cost_soap + 50;

  or Cost_soap += 50;

- To double the quantity of water in a bowl.

  Water_inBowl *= 2;

✓ Therefore assignment operator are used to store the changed value of the variable in the same variable.

➢Conditional Operator

Conditional operator contains condition followed by two statements. If the condition is true the first statement is executed otherwise the second statement.

It is also called as **ternary operator** because it requires three operands.

Operator	Description	Example
?:	conditional expression, Condition? Expression1: Expression2	(a>b)? “a is greater”: “b is greater”

Conditional Operator

- Eligibility to cast vote

  (age>=18)? “can cast vote”: “cannot cast vote”;

- In C

  (age>=18)? printf(“can cast vote”) : printf(“cannot cast vote”);

➢Bitwise Operator

A bitwise operator works on each bit of data.

Logical Table				
a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Operator	Description	Example(a=1 and b=0)
&	bitwise AND	a & b = 0
|	bitwise OR	a| b = 1
^	bitwise XOR	a ^ b = 1
~	bitwise one's complement	~a = 0, ~b=1
<<	bitwise left shift, indicates the bits are to be shifted to the left.	1101 << 1 = 1010
>>	bitwise right shift, indicates the bits are to be shifted to the right.	1101 >> 1 = 0110

Explanation

- The & (bitwise AND) in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
- The | (bitwise OR) in C takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
- The ^ (bitwise XOR) in C takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
- The << (left shift) in C takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
- The >> (right shift) in C takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
- The ~ (bitwise NOT) in C takes one number and inverts all bits of it

Program Example

```
#include <stdio.h>
int main()
{
    int a = 2, b = 4;
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a & b);
    printf("a|b = %d\n", a | b);
    printf("a^b = %d\n", a ^ b);
    printf("~a = %d\n", a = ~a);
    printf("b<<1 = %d\n", b << 1);
    printf("b>>1 = %d\n", b >> 1);
    return 0;
}
```

- Output:

```
a = 2, b = 4
a&b = 0
a|b = 6
a^b = 6
~a = -3
b<<1 = 8
b>>1 = 2
```

Bitwise operators (Explanation)

Consider
```
int a=2, b=4;
```

Bitwise AND:

a & b

As a and b values are in decimal number system, convert them into binary.

a=2 → 2 | 2 | 0 (1 → 1) ↑ 00000010 (2 in Binary)

b=4 → 2 | 4 | 0 / 2 | 2 | 0 (1 → 1) ↑ 00000100 (4 in Binary)

Truth Table of Bitwise AND:

x	Y	x & Y
0	0	0
0	1	0
1	0	0
1	1	1

Now, a & b (apply truth table logic)

a = 2 → 00000010
(AND) &
b = 4 → 00000100
Result → 00000000 → convert to decimal again
$2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0$

$2^7 \times 0 + 2^6 \times 0 + 2^5 \times 0 + 2^4 \times 0 + 2^3 \times 0 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 0 = 0$

So final output is 0.

Bitwise OR:

a | b

Truth Table of Bitwise OR:

x	Y	x | Y
0	0	0
0	1	1
1	0	1
1	1	1

a = 2 → 00000010
b = 4 (OR) | 00000100
00000110
$2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0$

$2^7 \times 0 + 2^6 \times 0 + 2^5 \times 0 + 2^4 \times 0 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 0$.

$\Rightarrow 4 + 2 = 6$ → final output

3) Bitwise XOR (^)

a = 2 → (00000010)
b = 4 → (00000100)
[Result] → 00000110

Truth table:

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	0

When both the inputs are opposite of each other, then XOR operation gives true, otherwise false

Result → 00000110
↳ Convert back to decimal : (6) output

4) Bitwise One's Complement (~) (or Bitwise NOT) Negation.

a = ~a

Formula is : $-(n+1)$

So, if n = 2
then ~n ⇒ $-(n+1)$
⇒ $-(2+1)$
⇒ −3 output

So, if a = 2
then ~a = −3

5) Shift operators

Bitwise left shift:

b = 4 → 00000100

b << 1 → By one bit
↓
Left shift

00000100 << 1

⇒ 000001000 → output

Convert to decimal.

0000 1000
$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

⇒ $2^3 \times 1$
= 8
Final output.

6) Bitwise right shift

b = 4 → 00000100

b >> 1

00000100 >> 1

000000100
↓
Convert to decimal

$2^1 \times 1$ = 2 → output

Q1

What will be the output of following code?

```
#include<stdio.h>
int main()
{
int a=10,b=5;
printf("%d",a&b);
return 0;
}
```

A. 10
B. 5
C. 0
D. 1

Q2

What will be the output of following code?

```
#include<stdio.h>
int main()
{
int a=7,b=5;
printf("%d",a|b);
return 0;
}
```

A. 7

B. 5

C. 12

D. 0

Q3

What will be the output of following code?

```
#include<stdio.h>
int main()
{
int a=8,b=3;
printf("%d",a^b);
return 0;
}
```

A. 8
B. 3
C. 1
D. 11

Q4

What will be the output of following code?

```
#include<stdio.h>
int main()
{
int a=10;
printf("%d",~a);
return 0;
}
```

A. 11

B. -11

C. 9

D. -9

➢Some Special Operators

Operator	Description	Example
,	comma operator, can be used to link the related expressions together	int a, b, x;
sizeof ()	sizeof operator to find the size of an object.	int a; sizeof(a)=2
type	Cast operator, to change the data type of the variable	float x= 12.5; int a; a = (int) x; value of a is 12.

Explanation

COMMA OPERATOR

- The comma operator in C takes two operands. It works by evaluating the first and discarding its value, and then evaluates the second and returns the value as the result of the expression.
- Comma separated operands when chained together are evaluated in left-to-right sequence with the right-most value yielding the result of the expression.
- Among all the operators, the comma operator has the lowest precedence. For example,

```
int a=2, b=3, x=0;
x = (++a, b+=a);
```

Now, the value of x = 6.

sizeof operator

sizeof is a unary operator used to calculate the sizes of data types.

It can be applied to all data types.

The operator returns the size of the variable, data type or expression in bytes.

'sizeof' operator is used to determine the amount of memory space that the variable/expression/data type will take. For example,

sizeof(char) returns 1, that is the size of a character data type

- Comma operator can be used like:

  for(i=0 , j=1 ; i>10 ; i++ , j++)

- To know space occupied by variable in computer memory we use *sizeof()* operator.

  char choice;

  int char_sz = sizeof(choice); // 1 because char is 1byte

- If we are adding float number and integer number and we require output in float then integer number is converted to float using *type cast* operator.

  int num1;

  float num2, sum;

  sum= (float) num1 + num2;

Uses of printf

```
#include<stdio.h>
main()
{
printf("Create a new line\n");
printf("Print a double quotes (\") within a string\n");
printf("Print a single quotes (\') within a string\n");
printf("Print a Backslash\\ within a string\n");
printf("Using\tTab within a string\n");
}
```

Lecture-5(Part-2)Operators

Precedence and Associativity of Operators

Precedence of Operators

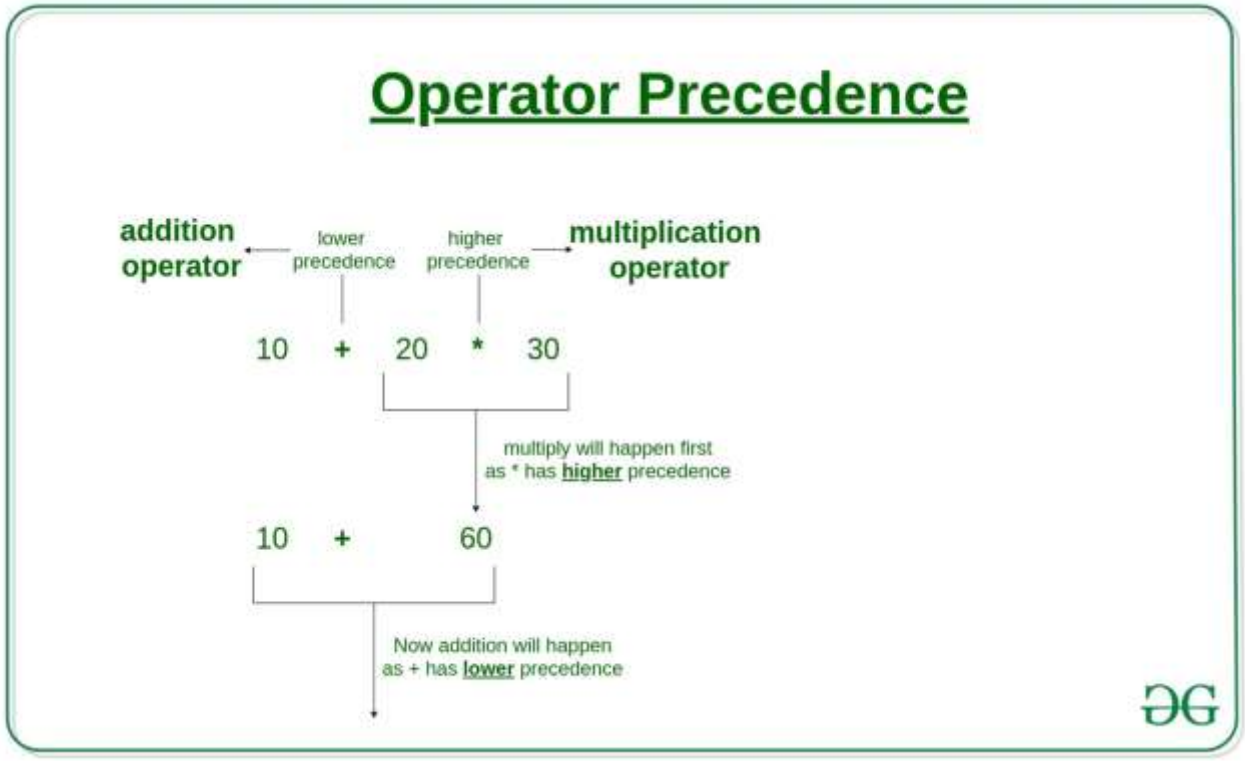
- The precedence of operators determine a rank for the operators. The higher an operator's precedence or priority,

Example: So how the expression a * b + c will be interpreted?

(a * b) + c or a * (b + c),

here the first interpretation is the one that is used because the multiplication operator has higher precedence than addition.

Precedence-Example



Associativity of Operators

- Associativity tell us the order in which several operators with equal precedence are computed or processed in two directions, either from left to right or vice-versa.

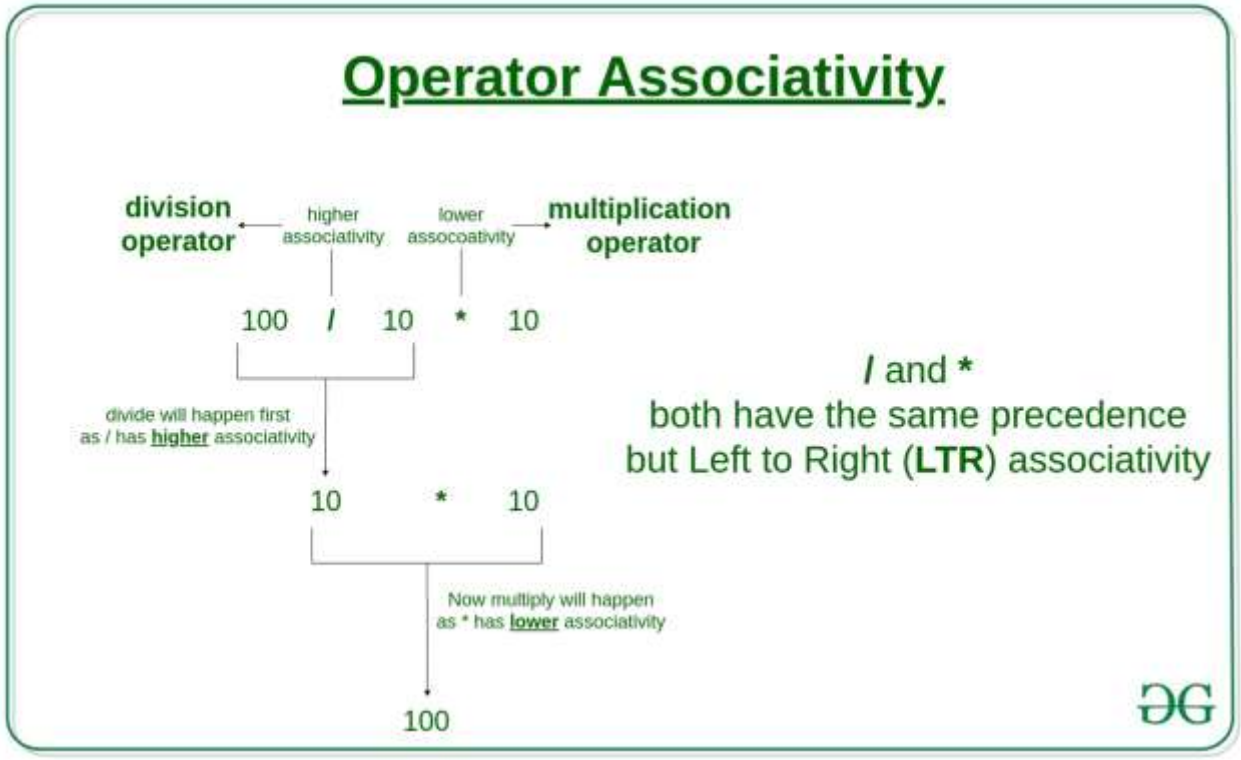
Example: In the expression

a * b / c,

since multiplication and division have the same precedence we must use the associativity to determine the grouping. These operators are left associative which means they are grouped left to right as if the expression was

(a * b) / c.

Associativity-Example



Operator	Description	Associativity
( ) [ ] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

Example Expressions Using the Precedence Chart

```
1. x = 3 * 4 + 5 * 6
     = 12 + 5 * 6
     = 12 + 30
     = 42
2. x = 3 * (4 + 5) * 6
     = 3 * 9 * 6
     = 27 * 6
     = 162
3. x = 3 * 4 % 5 / 2
     = 12 % 5 / 2
     = 2 / 2
     = 1
4. x = 3 * (4 % 5) / 2
     = 3 * 4 / 2
     = 12 / 2
     = 6
5. x = 3 * 4 % (5 / 2)
     = 3 * 4 % 2
     = 12 % 2
     = 0
6. x = 3 * ((4 % 5) / 2)
     = 3 * (4 / 2)
     = 3 * 2
     = 6
```

```
10. a && b
    = 0
11. a < b && c < b
    = 1
12. b + c || ! a
    = ( b + c) || (!a)
    = 0 ||1
    = 1
13. x * 5 && 5 || ( b / c)
    = ((x * 5) && 5) || (b / c)
    = (12.5 && 5) || (1/-1)
    = 1
14. a <= 10 && x >= 1 && b
    = ((a <= 10) && (x >= 1)) && b
    = (1 && 1) && 1
    = 1
15. !x || !c || b + c
    = ((!x) || (!c)) || (b + c)
    = (0 || 0) || 0
    = 0
```

Consider

int a=0,b=1,c=-1

float x=2.5,y=0.0

Practice questions(Q1)

```
#include <stdio.h>
 int main()
  {
     double b = 5 % 3 & 4 + 5 * 6;
     printf("%lf", b);
     return 0;
  }
```

A. 2
B. 30
C. 2.000000
D. 5

(Q2)

```
#include <stdio.h>
   int main()
   {
      double b = 3 && 5 & 4 % 3;
      printf("%lf", b);
      return 0;
   }
```

A. 3.000000
B. 4.000000
C. 5.000000
D. 1.000000

(Q3)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    double b = 5 & 3 && 4 || 5 | 6;
    printf("%lf", b);
    return 0;
}
```

A. 1.000000

B. 0.000000

C. 7.000000

D. 2.000000

(Q4)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int b = 5 + 7 * 4 - 9 * (3, 2);
    printf("%d", b);
}
```

A. 6

B. 15

C. 13

D. 21

(Q5)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int b = 4 * 6 + 3 * 4 < 3 ? 4 : 3;
    printf("%d\n", b);
}
```

A. 3

B. 33

C. 34

D. 4

(Q6)

Which of the following operators has an associativity from Right to Left?

A. <=

B. <<

C. ==

D. +=

(Q7)

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int a = -1, b = 4, c = 1, d;
    d = ++a && ++b || ++c;
    printf("%d, %d, %d, %d\n", a, b, c, d);
    return 0;
}
```

A. 0, 4, 2, 1
B. 0, 5, 2, 1
C. -1, 4, 1, 1
D. 0, 5, 1, 0