Page No.
Date

INTRODUCTION TO COMPUTERS

Q:- What is Computer?

Ans:- A Computer is an electronic machine that perform tasks or computation according to set of instructions called program.
The first fully electronic computer machine introduced in the 1940s were huge machines that required a team of peoples to operate. Compared to those early machine today's computers are amazing - they are thousands of time faster and can fit on your desk, in your lap or even in your pocket.

A Computer is generally capable for :-

(i) Communicating with other Computers
(ii) Storing data & instructions in its memory.
(iii) Performing arithmetic and logical operation acc. to the specified functions with very high speed and great accuracy.
(iv) Producing the results in many forms.

HARDWARE:- Hardware refers to that part of the Computers which you can see and touch. The most important part of computer hardware is tiny rectangular chip inside your Computer is called microprocessor. It's the brain of computer - the part that translate instruction and perform Calculation. Hardware items such as keyboard, CPU, monitor, mouse, printer and other components are often called hardware devices.

SOFTWARE:- It refers to instructions and programs

Page No.
Date / 20

that tell the hardware what to do. The most important example of software is operating system (OS) that manages all the resources such as memories, input/output devices etc. of your computer system and provides an interface using which the user can interact with other computers to perform various tasks. Two well-known operating systems are windows and Linux.

DATA & INFORMATION

The terms data and information have been used to mean the same thing but actually information is more than data just.

Data:- The term data simply refers to a value or set of values. These values may represent some observations from an experiment such as marks obtained by a student in examination.

PROGRAMS IN C

1) Program to print 'Hello world'.

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

2) Addition of two numbers :-

```
#include <stdio.h>
int main()
{
  int a, b;
  printf(" Enter number a and b");
  scanf("%d %d", &a, &b);
  printf("the sum is %d\n", a+b);
  return 0;
}
```

* **OPERATORS :-** An operator is a symbol used to perform operations in given programming language.

TYPES :- (i) Arithmetic
(ii) Relational (iii) Logical
(iv) Bitwise (v) Assignment

I) Arithmetic

Addition	—	+
Substraction	—	-
Multiplication	—	*
Division	—	/
Modulus	—	%

II Relational

is equal to	—	==
is not equal to	—	!=
Greater than	—	>
lesser than	—	<

Greater than or equal to – >=
Lesser than or equal to – <=

III LOGICAL

Logical AND Operator – &&
(if both the operands are non-zero, then the condition is true).
(A&&B) is false.

Logical OR operator – ||
(if any of these two operands is non-zero, then condition become true). (A||B) is true.

Logical NOT operator – !
It is used to reverse the logical state of its operand. If condition is true, then Logical NOT operator will make it false.
!(A&&B) is true.

IV BITWISE OPERATORS

a	b	a&b	a|b (OR)	a^b (XOR)
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Page No.
Date / 20

ARITHMETIC PROGRAM

```
#include <stdio.h>
int main()
{
    int a, b;
      a = 34;
      b = 6;
  printf("a+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(
   return 0;
}
```

For Float :-

```
#include <stdio.h>
int main()
 {
    int a;
    float b;
      a = 34 ,
        b = 6.34;
    printf("a+b = %f\n", a+b);
     return 0;
}
```

OTHERS BITWISE OPERATORS

Binary one complement operator - ~
Binary left shift operator - <<
Binary Right shift operator - >>

Page No.
Date / 20

V ASSIGNMENT OPERATORS

= Simple assignment operator. Assign values from right side operand to left side operand.

+= Add AND assignment operator. It adds the right operand to the left operand and assign the result to left operand

-= Substract AND assignment operator. It substracts the right operand from left operand and result assign to left operand.

*= Multiple AND assignment operator It multiplies the right operand with left and result assign to left operand.

/= Divide AND assignment operator. It divide the left operand with right operand and result is assigned to the left.

VI MISCELLANEOUS OPERATORS

Size of()	Return the Size of variable
&	Return the address of variable.
*	Pointer to a variable
?:	Conditional Expression

OPERATOR PRECEDENCE IN C.

Page No.
Date / 20

Multiplicative ↓ * / %
Additive ↓ + -
Shift ↓ << >>
Relational ↓ <<= >>=
Equality == !=

Left to Right

	Category	Operator	Associativity
1)	Postfix	() [] -> . ++ --	Left to Right
2)	Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to Left
3)	Multiplicative	* / %	Left to Right
4)	Additive	+ -	"
5)	Shift	<< >>	"
6)	Relational	<<= >>=	"
7)	Equality	== !=	"
8)	Bitwise AND	&	"
9)	Bitwise XOR	^	"
10)	Bitwise OR	|	"
11)	Logical AND	&&	"
12)	Logical OR	||	"
13)	Conditional	? :	Right to Left
14)	Assignment	= += -= *= /= %=	"
15)	Comma	,	Left to Right

Assignment

=>> , <<= , &= ^= |= .

Page No.
Date / 20

1. Add two numbers

```
# include <stdio.h>
void main()
{
   int a=7 ,b=9, c;
       c=a+b;
printf (" sum of is two n:%.d", c);
    return 0;
}
```

Output

Sum is 16

2. Add two Nos.

```
# include <stdio.h>
  void main()
{
  int a,b,c;
printf(" enter a and b");
scanf("%.d %.d", &a, &b);
  c= a+b;
  printf ("sum is %.d", c);
}
```

Output

enter a and b
5 8
Sum is 13.

3. Difference / Multiplication / Division

```
#include <stdio.h>
void main()
{
    int a, b, c;
    printf("enter a and b");
    scanf("%d %d", &a, &b);
    printf("difference is %d\n", a-b);
    printf("multiplication is %d\n", a*b);
    printf("modulus is %d\n", a/b);
    printf("remainder is %d\n", a%b);
}
```

Output:- enter a and b;
7 8
difference is -1
multiplication is 56
modulus is 0
remainder is 7.

FORMAT SPECIFIERS AND Escape Sequence in C

1) format specifier is way to tell the compiler what type of data is in a variable during taking input displaying output to the user.
2) printf("The is a good boy %.2f", var); will print var with 2 decimal points in a '2' character space.
3) Let to our IDE & learn more about the format specifier

do while loop

Date / 20

```
do {
// code to be executed
} while (condition)
```

(if condition fail then loop se bhr).

Ex
```
int i=0;
do {
    i=i+1;
    printf ("%d", i);
} while (i<10);
```

(do while loop atleast once).

Example :-
```
#include <stdio.h>
int main()
{
    int num, index=0;
    printf("Enter a number\n");
    scanf("%d", &num);
    do
    {
        printf("%d", index);
        index = index+1;
    } while (index < num);
    return 0;
}
```

While loop :-

```
int i=0
while (i<30) {
printf ("%d", i);
    i= i+1;
}
```

```
#include <stdio.h>
int main()
{
    int i=0
    while(i<54)
    {
        printf("%d\n", i);
        i= i+1;
    }
    return 0;
}.
```

For loop :- (i) It is used to iterate the statements or a part of a program several times.

(ii) It is used to traverse the data structures like the arrays and linked lists.

(iii) It has a little diff. syntax than while and do while loop.

Basic syntax :-

The syntax of for loop in C language :-

```
for ( Expression 1 ; Expression 2; Expression 3) {
    // code to be run
}.
```

ex

```
for ( i=0 , i<5, i++) {
    printf ("%d", i);
}
```

(Phele increment krega (i++), then Condition Check krega the execution

Page No.
Date / / 20

properties of Expression 1 (i=0)

(i) The expression represents the Initialization of the loop.
(ii) We can initialize more than one variable in Expression 1.
(iii) Expression 1 is optional.

Properties of Expression 2. (i<5)

i) It is a conditional expression. It checks for a specific condition to be satisfied. If it is not then loop is terminated
(ii) It can have more than one condition However the loop will iterate until the last condition become false. The condition will be treated as statements
(iii) It is optional
(iv) Expression 2 can perform the task of expression 1 & 3. That is, we can initialize the variable as well as update the loop variable in expression 2 itself.
(v) We can pass zero or non-zero value in expression 2. However in C, any non-zero value is true and zero is false by default

Properties of Expression 3. (i++)

(i) It is used to update the loop variable
(ii) We can update more than one variable at same ~~point~~ time.
(iii) Expression 3 is optional.

Break and Continue Statements in C.

Page No.
Date

Break statement:

(i) used to bring the program control out of the loop
(ii) The break statement is used inside loops or switch statements.
(iii) Break statements can be used with (i) loops (ii) switch case expressions

Continue Statements

(i) used to bring the program control to the next iteration of the loop.
(ii) The continue statement skips some code inside the loop and continue with the next iteration.
(iii) It is mainly used for a condition so that we can skip some lines of code for a particular condition.

Functions in C:-

Q:- What is a function?
Ans (i) functions are used to divide a large C program into smaller pieces.
(ii) A function can be called multiple times to provide reusability and modularity to the C program.
(iii) Also called procedure or Subroutine.

Basic syntax of C function:-

```
return_type function_name (data_type parameter1,
                           data_type parameter2, ...) {
    // code to be executed
}
```

Advantages of C functions

(i) We can avoid rewritting same logic through function.
(ii) We can divide work among programmers using functions
(iii) We can easily debug a program using function

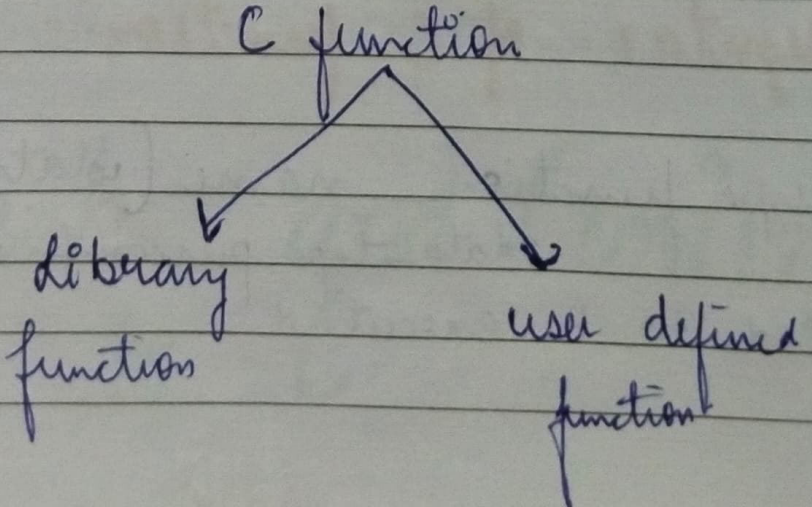
Declaration, Definition and Call

(i) A function is declared to tell a compiler about its existence.
(ii) A function is defined to get some task done
(iii) A function is called in order to be used.

Types of functions :-

Library functions :- functions included in C header files.

User defined function :- function created by C programmer to reduce Complexity of a program.



Function Code Examples

1) without arguments & without return type
2) " " & with " "
3) with " & without " "
4) with " & with " "

Recursion in C

1) Recursion is a process when a function calls a copy of itself to work on a smaller problem.
2) Any function which calls itself is called recursive function.
3) This makes the life of programmer easy by dividing a given problem into easier problems.
4) A termination condition is imposed on such functions to stop them executing copies of themselves forever.
5) Any problem that can be solved recursively can also be solve iteratively (otherway) mean

Why Recursions?

Ans: 1) Any problem that can be solved recursively, can also be solved iteratively.
2) However, some problems are best suited to be solved using recursion.
3) For example, tower of Hanoi, fibonacci series, factorial finding etc.

Factorial Calculations

i) The case at which the function doesn't recur is called the base case.
ii) The instances where the function keeps calling itself to perform a subtask is called recursive case.

Q:- What is an Array?

1) An array is the collection of data items of same type.
2) Items are stored at contiguous memory locations
3) It can also store the collection of derived data type, such as pointers, structure etc.
4) A one-dimensional array is like a list.
5) A two-dimesonal array is like a table (matrix)
6) The c-language places no limits on the no. of dimensions in an array.
7) Some text refer to one-dimensional arrays as vectors, two dimensional arrays as matrices and use the general term arrays when the no. of dimensions is unspecified or unimportant.

Q:- Why do we need arrays?

1) Code that use arrays is sometimes more organized and readable
2) If you were to store the marks in a test of 56 students 56 variable will make program look cluttered and messy.
3) Solution to this is arrays!
4) We can create arrays of integer and store the consecutive marks corresponding to the roll number in array

Page No.
Date / / 20

Advantage of Arrays :-

1) It is used to represent multiple data items of same type by using only single name.
2) Accessing an item in a given array is very fast!
3) 2 Dimensional arrays makes it easy in mathematical application as it is used to represent a matrix.

Properties of Arrays :-

1) Data in an array is stored in contiguous memory locations.
2) Each element of an array is of same size.
3) An element of the array with given index can be accessed very quickly by using its address which can be calculated using the base address and the index.

Syntax for declaring and initializing an array

→ Data_type name [size];
→ Data_type name [size] = { x, y, z, ... }; // Size not required in this case
→ Data_type name [rows] [columns]; // for 2-D arrays
→ We can also initialize the array one by one by accessing it using its index!

name [0] = 0;

→ **Fibonacci Series :-** first element = 0
Second element = 1

0, 1, 1, 2, 3, 5, 8, 13

Next, Third element = add first + Second element

0+1=1, 1+1=2

POINTERS

Q :- What is a pointer ?

Ans:- 1) Variable which stores the address of another variable.

2) Can be type int, char, array, function or any other pointer.

3) Size depends on the architecture. Example 2 bytes for 32 bit.

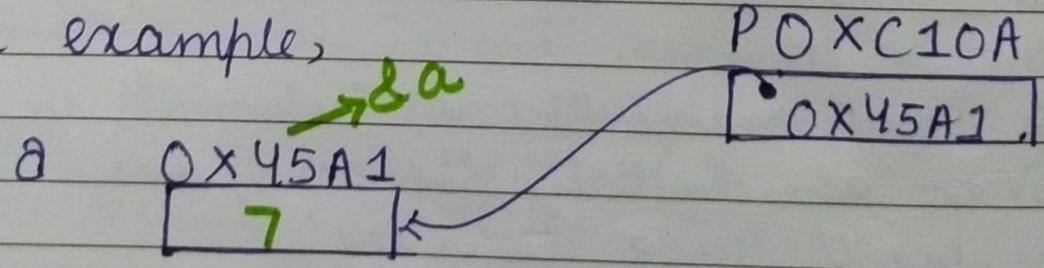
4) Pointer in C programming language can be declared using (*) called asterisk symbol

→ **'&' and '*' Operators in Pointers**

Ans:- • The address of operator '&' returns the address of variable.

• * is the dereference operator (also called indirection operator) used to get the value at a given address.

for example,



int a = 7;

&a → address of a.

int *pa = &a

Page No.
Date / 20

Program :-

```
#include <stdio.h>
int main()
{
printf("Lets learn about pointers\n");
int a=76;
int *ptr a = &a;
printf("The address of pointer to a is %p\n",&ptr a);
printf("The address of a is %p\n", ptr a);&a);
printf("The address of a is %p\n", ptr a);
printf("The value of a is %d\n", *ptr a);
printf("The value of a is %d\n", a);
return 0;
}
```

output

→ The address of pointers to a is 0061FED8
The address of a is 0061FED8
The address of a is 0061FED8
The value of a is 76
The value of a is 76.

printf format specifiers

%c	character
%d	decimal (integer) number (base 10)
%e	exponential floating-point number
%f	floating point number
%i	integer (base 10)
%o	octal number (base 8)
%s	a string of characters
%x	number in hexadecimal (base 16)

Null Pointers

1) A pointer that is not assigned any value but NULL is known as the null pointer.
2) In C programming, a null pointer is a pointer that does not point to any object a function.
3) we can use it to initialize a pointer variable when that pointer variable isn't assigned any valid memory address yet.
4) `int * ptr = NULL;`

Use of Pointer :-

1) Dynamic memory allocation
2) Arrays, functions and structures.
3) Return multiple values from a function.
4) Pointer reduces the code and improves the performance.

Pointer Arithmetic

There are four arithmetic operators that Can be used on pointers :-

→ ++ → +
→ -- → -

Arrays & Pointers

→ arr[i] == * (arr + i)
→ if arr is a pointer to arr[0] then arr + i is a pointer to arr[i].

Page No.
Date / 20

Functions

→ Call Actual and formal Parameters

Ans. When a function is called the values (expression) that are passed in the call are called the arguments or acutal parameters.

→ Formal parameters are local variables which are assigned values from the arguments when the function is called.

→ Types of function in C programming

Ans In C programming language, we can call a function in two different ways, based on how we specify the arguments, and these two ways:-

1) Call by value (value remain same)
2) Call by Reference. (value change bcoz address hai).

→ Call by value

1. When we call a function by value, it means that we are passing the values of the arguments which are copied into the formal parameter of the function.
2. Which means that the original values remain unchanged and only the parameters inside the function changes.

→ Call by Reference (address copy hota hai)

1) It is the method of passing arguments to a C function copies the address of the arguments into the formal parameter.
2) Addresses of the actual arguments are copied and then assigned to the corresponding formal arguments.

→ Example of Call by Reference

Page No.
Date / 20

Structures:-

Q:- What is structure in C?

Ans 1. Structures are user-defined data types in C.
2. Using structures allows us to combine data of different types together.
3. It is used to create the complex data which contains diverse information.
4. They are very similar to arrays but structure can store data of any type, which is practically more useful.

Syntax to defining a structure:-

```
struct [structure_name]
{
    // data type var 1
    // data type var 2
    // data type var 3 ...
};
    [structures_____variables];
```

Declaring a structure:-

(with structure defination)

```
# include <stdio.h>
struct Employee
{
    int id;
    Char name[40];
    float marks;
};
struct Employee e1, e2;
```

Page No.
Date / 20

```
int main()
{
  return 0;
}
```

OR (without defination)

```
#include <stdio.h>
struct Employee
{
  int id;
  char name[40];
  float marks;
} e1, e2;
int main()
{
  struct Employee tt;
  return 0;
}
```

[for ex
e1 → structure name
e2 → member name]

→ Accessing structure members :-

1) Arrays elements are accessed using the subscript variable.
2) In a similar fashion, structure members are accessed using dot [.] operator.
3) (.) is called as "structure member operator".
4) To access the member of the structure, we use this operator in between "structure name" and "member name".