CBCS SCHEME

USN | | | | | | | | | | |

BPOPS103/203

First/Second Semester B.E./B.Tech. Degree Examination, June/July 2023
Principles of Programming Using C

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Define Computer. Describe the characteristics of computer in detail.	10	L2	CO1
	b.	Explain various Input devices.	10	L2	CO1
		OR			
Q.2	a.	Explain the following programming paradigms. i) Procedural Programming ii) Structured Programming iii) Object Oriented Programming.	10	L2	CO2
	b.	Explain printf( ) and scanf( ) functions with their syntax. Give the illustrative examples.	10	L2	CO2
		Module – 2			
Q.3	a.	Explain any five types of operators in C language with the illustrative examples.	10	L2	CO2
	b.	Write a C program to find the roots of quadratic equation by accepting the coefficients. Print appropriate messages.	10	L3	CO2
		OR			
Q.4	a.	What are iterative statements? Explain three types of iterative statements with their syntax.	10	L2	CO2
	b.	Write a program to print the following pattern. 1 1 2 1 2 3 1 2 3 4	10	L3	CO2
		Module – 3			
Q.5	a.	Explain the syntax of function declaration and function definition with example.	06	L2	CO2, CO5
	b.	Write a C program to swap two numbers using call by reference method.	06	L3	CO2, CO5
	c.	Describe different types of storage classes with examples.	08	L2	CO2
		OR			
Q.6	a.	What is an array? Explain how arrays are declared and initialized with example.	08	L2	CO3
	b.	Write a C program to transpose a 3×3 matrix.	08	L3	CO3
	c.	List applications of arrays.	04	L3	CO3

		Module – 4			
Q.7	a.	Write a C program to convert characters of a string into upper case without using built-in function.	10	L3	CO3
	b.	Discuss the working of the following string manipulation functions with suitable examples. i) strcmp ii) strlen iii) strcpy iv) strcat v) strcmp	10	L2	CO3
		OR			
Q.8	a.	Define Pointer. Explain the declaration of a pointer variable with an example.	05	L2	CO2, CO4
	b.	Mention the applications of pointers.	05	L2	CO4
	c.	Develop a C program to compute the sum, mean and standard deviation of all elements of an array using pointers.	10	L3	CO3, CO4
		Module – 5			
Q.9	a.	What is structure? Explain the declaration of a structure with an example.	06	L2	CO4
	b.	Differentiate between Structures and Unions.	06	L3	CO4
	c.	Develop a C program to read and display the information of all the students in the class.	08	L3	CO4
		OR			
Q.10	a.	Define Enumerated datatype. Explain the declaration and access of enumerated datatypes with a code in C.	06	L2	CO2
	b.	Explain the process of opening a file in C.	06	L2	CO2
	c.	Write a C program to demonstrate fwrite( ) function.	08	L3	CO2

* * * * *

C ISC
CDAC
Study Spring

Principles of Programming Using C - June/July 2023

Module - 1

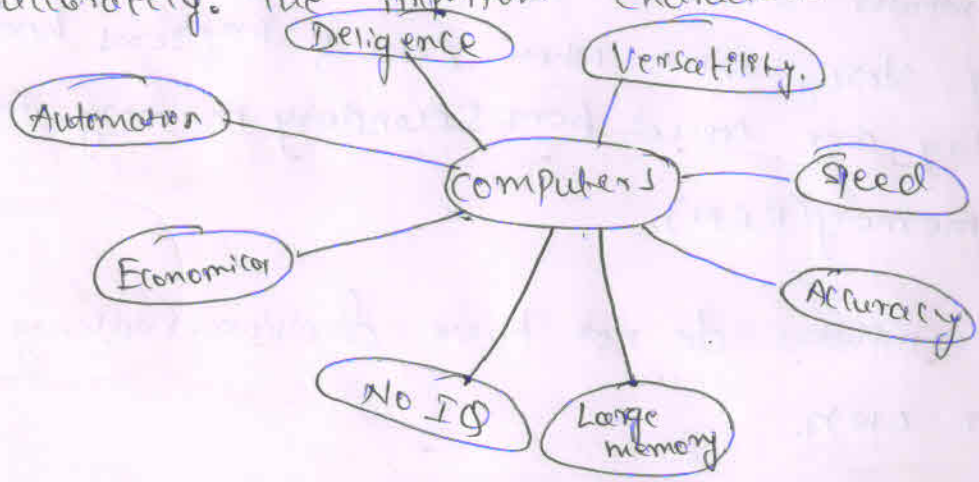
Q 1)

a) Define computer. Describe characteristics of computer in det

→ A computer is an electronic device that is designed to accept data, perform the required mathematical & logical operations at high speed and output the result.

Characteristics of Computers

Computers are basically meant to solve problems quickly & accurately. The important characteristics of a computer are:



① Speed: Computers can perform millions of operations per second. the speed is usually given in nano seconds

② Accuracy: A computer is a very fast, reliable & robo electronic device. It always gives accurate result provided the correct data & set of instructions are input to it. If input is wrong, then output will be erroneou

Automation - Computers are automatable device that ca perform a task without any user intervention.

Deligence - Computers never get tired of a repetitive tas It can continually work for hours without creating errors.

Versatile - Computers can perform multiple tasks of differe nature at the same time.

Memory: The computer have two kinds of memory
1) primary memory 2) secondary memory.
The computer stores large amount of data & programs i secondary storage space. When data & programs have to be used, they are copied from secondary memory into the primary memory (RAM).

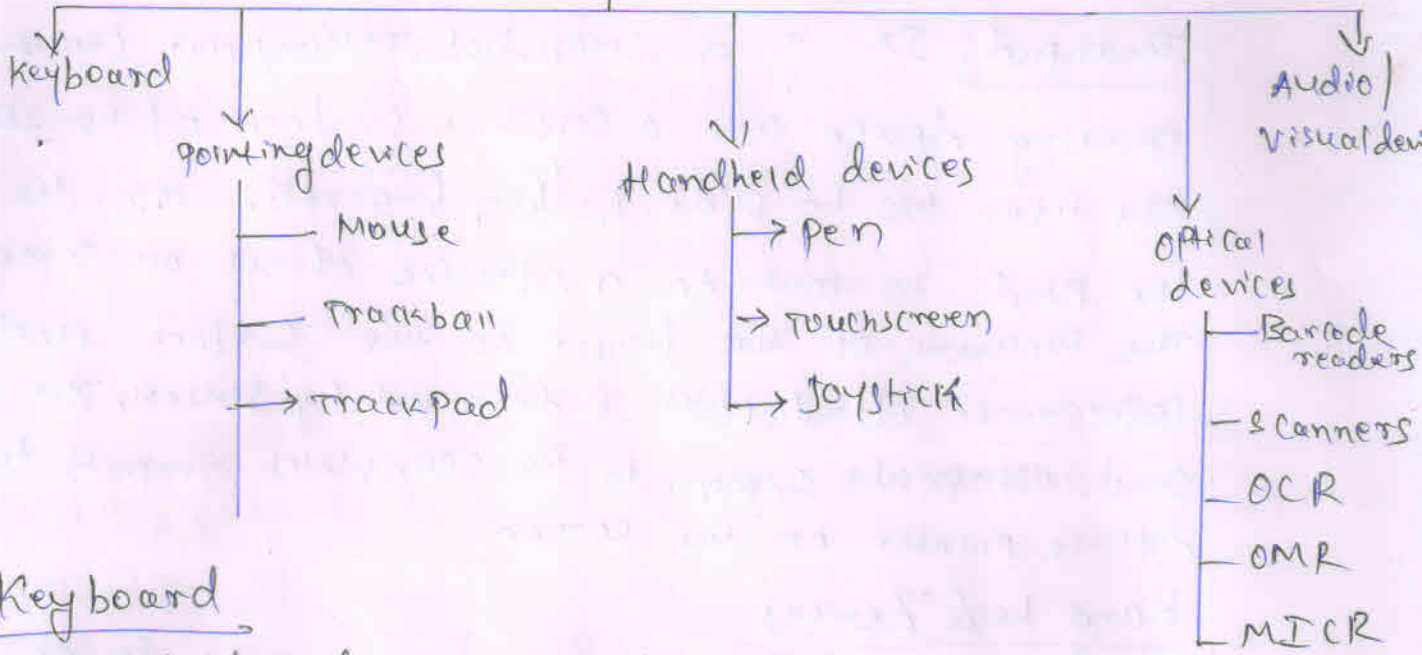
No IQ: Computers do not have decision-making abilities on their own.

Economical: Today, Computers are considered as short-ter investments for achieving long-term gains. Using computer also reduces manpower requirements and leads to an elegant & efficient way of performing various tasks. Hence Computers save time, energy & money

Q2) Explain various input devices

→ An input device is used to feed data & instructions into a computer.

Input devices



① Keyboard

- The keyboards have 80-110 keys
- Using keyboard, the user can type a document, use keystroke shortcuts, access menus, play games, etc.
- The keyboard includes following keys:-
  - → Typing keys
  - → Numeric keys
  - → Function keys
  - → Control keys

② Pointing Devices

A pointing device enables users to easily control the movement of pointer to select items on a display screen, to draw graphics, etc.

i) **Mouse:-** It is the key input device used in GUI. The mouse has two buttons with a scroll wheel.
Operations performed on mouse are:- point, click, drag, scroll.

• Trackball: It is a pointing device used to control the position of the cursor on the screen. The working of trackball is identical to that of mouse.

Touchpad: It is a small, flat, rectangular stationary pointing device with a sensitive surface of 1.5-2 inches. The user has to slide his/her fingertips a/c the surface of the pad to point to a specific object on screen. The pressure of the finger on the surface leads to a capacitance effect, which is detected by sensor. The sensors send appropriate signals to the CPU, which interprets them & displays pointer on the screen.

Hand held devices

• Stylus (Pen): It is a pen shaped input device used to enter information/write on the touchscreen of a handheld device.

• Touchscreen: It is a display screen that can identify the occurence and position of a touch inside the display region.

• Joystick: It is a cursor control device widely used in games & Computer Aided Design.

Optical Devices

• Barcode Reader: It is used to capture & read information stored in barcode.

• Image Scanner: A scanner is a device that captures images, printed text, & handwriting from different sources such as photographic prints, posters & magazines & convert them into digital images for editing & display on computers.

- Optical Character Recognition (OCR) Device: It is the process of converting printed materials into text or word processing files that can be easily edited and stored.

- Optical Mark Recognition (OMR) Device: It is the process of electronically extracting data from marked fields such as checkboxes & fill-in fields on printed forms. The optical mark reader is fed with OMR sheet that has pen/pencil marks in pre-defined positions to indicate each selected response.

- Magnetic Ink Character Reader (MICR): It is used to verify the legitimacy of paper documents, especially bank cheques. It consists of magnetic ink printed characters that can be recognized by high-speed magnetic recognition device.

Audio/Visual Input Devices

- Audio devices: These are used to capture/create sound.
  e.g. Microphones & CD players.

- Video Input devices: These are used to capture video from the outside world into the computer.
  e.g. Webcamera, Digital camera.

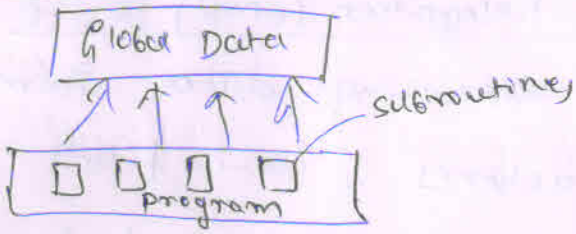
Q 2

(a) Explain the following programming Paradigms.

i) Procedural programming ii) Structured programming

iii) Object oriented programming.

→ i) Procedural programming :-



* A program is divided into subroutines that can access global data.

* Each subroutine performs well defined task.

* Programming Languages – FORTRAN & COBOL.

Advantages

→ Goal is to write correct programs

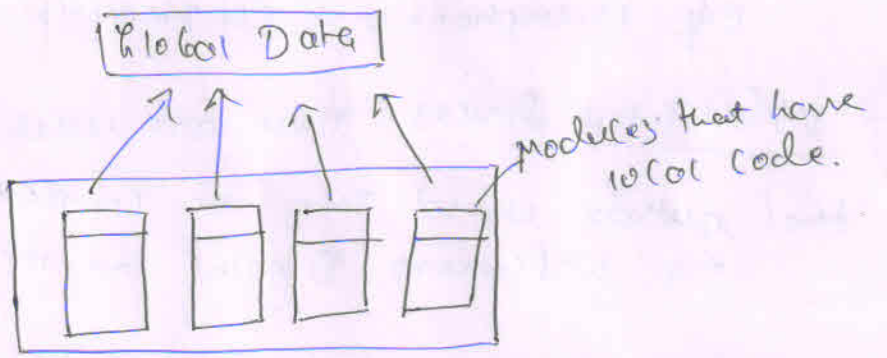
→ programs are easier to write compared to monolithic

Disadvantages

→ No reusability

→ Needs more time & Effort to write programs.

ii) Structured Programming :



* It is also called Modular programming.

* It employs top-down approach in which overall program structure is broken down into separate modules.

* Modules are coded separately & once module is written &

tested individually, it is then integrated with other modules to form the overall program structure.

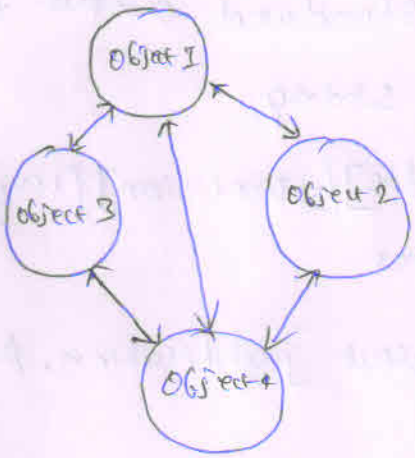
Advantages:

- Goal is to write correct programs that are easy to understand & change
- Each module performs specific task & has its own local data.

Disadvantages:

- Not data-centered
- Main focus is on functions.

ii) Object-oriented Programming (OOP)



* OOP treats data as a critical element in the program. development & restricts its flow freely around the system.

* In OOP Paradigm all the relevant data and tasks are grouped together in entities known as objects

Features of OOP

① Programs are data centered.
② Programs are divided in terms of objects.
③ Follows bottom-up approach for problem solving.

Q 2(b) Explain printf() and Scanf() functions with their Syntax. Give illustrative examples.

→ printf():

* It is used to display information required by user & also prints the values of the variables.

* Syntax:-

```
printf("control string", variable-list);
```

Control string - Contains the text that has to be written on to standard output device.

- Prototype of control string.

%[flags][width][.precision][length modifier] type specifies.

flag - Specifies output justification. Accepted values are -, +, #, 0.

width - Specifies minimum number of positions in output

precision - specifies maximum number of characters to print. Accepted values (d, i, o, u, x, X)

Length Modifier - Accepted values (h, l, L).

Type specifier - Define the type & interpretation of value corresponding to argument.
Accepted values - c, d, b, E, e, f, g. etc

example

```
printf("Result : %d %x %x %#x", 234, 234, 234, 239);
```

output: Result: 234 EA 0xEA

```
char str[] = "Good Morning"
printf("%.7s", str);
```

output: Good Mo.

→ Scanf():

* It is used to read formatted data from the keyboard

*the scanf() takes a text stream from keyboard, extracts & formats data from the stream according to control strings & then stores the data in specified program variables.

Syntax:-

Scanf("control string", arg1, arg2, arg3, ... argn);

The control string specifies the type & format of the data that has to be obtained from keyboard & stored in memory locations pointed by arg1, arg2, ...

Prototype of control string

%[*][width][modifier]type.

* → suppresses assignment of input field.

Width → Specifies maximum number of characters to read

Modifier → It can be h, l, or L for data pointed by corresponding additional arguments.

Type → Specifies type of data that has to be read.

Example

① Scanf("%2d %5d", &num1, &num2);

// the first ~~digit~~ number can have 2 integers while second ~~digit~~ number can have maximum of 5 digits.

② Char str[10];
Scanf("%s", str);

Module-2

Q3

(a) Explain any five types of operators in C language with the illustrative examples.

→ An operator is a symbol that specifies the mathematical, logical or relational operation to be performed.

Types of operators

1. Arithmetic operators
2. Relational operators
3. Equality Operators
4. Logical operators
5. unary operators
6. Conditional operators
7. Bitwise operators
8. Assignment operators
9. comma operator
10. Size of operator.

① Arithmetic operators

Arithmetic operators can be applied to any integer or floating-point number.

e.g int a=9, b=3, result;

Operation	Operator	Syntax	Command	Result
Multiply	*	a * b	result = a * b	27
Divide	/	a / b	result = a / b	3
Addition	+	a+b	result = a+b	12
Subtraction	-	a - b	result = a-b	6
Modulus	%	a % b	result = a % b	0

② Relational operators

* Compares two values.
* Return true/false value.

Operator	Meaning	Example
<	Less than	3<5 gives 1
>	greater than	7>9 gives 0
<=	less than or equal to	100 <= 100 gives 1
>=	greater than or equal to	50 >= 100 gives 0

③ Conditional Operator (? :)

It is similar to if-else statement that can be used within expressions

Syntax:-

exp1 ? exp2 : exp3;

exp1 is evaluated first. If it is true then exp 2 is evaluated & becomes result of the expression, otherwise exp3 is evaluated & becomes result of expression.

Example

```
int a=5, b=3, c=7, small;
small = (a<b ? (a<c ? a : c) : (b<c ? b : c));
```

④ Assignment operator

* It is responsible for assigning values to variables.
* The assignment operator has left to right associativity

e.g.

```
int x=2, y=3, sum=0;
sum = x+y // sum = 5
```

② a=b=c=10;

(8) Comma operator

- Takes 2 operands.
- It works by evaluating the first & discarding its value & then evaluates the second & returns the value as the result of expression.
- It has lowest precedence.

e.g. int a=2, b=3, x=0;
x = (++a, b+=a);
// x=6;

Q3 (b) Write a C program to find roots of quadratic equation by accepting the coefficients. Print appropriate messages.

→ Program

```
#include<stdio.h>
#include<math.h>
Void main()
{
  int a, b, c;
  float D, deno, root1, root2;
  printf("Enter the value of a,b, & c\n");
  scanf("%d %d %d", &a, &b, &c);
  D = (b*b) - (4*a*c);
  deno = 2*a;
  if (D>0)
  {
    printf("Real Roots\n");
    root1 = (-b + sqrt(D)) / deno;
    root2 = (-b - sqrt(D)) / deno;
```

```
printf("Root 1 = %.f \t Root 2 = %.f", root1, root2);
}
else if (D==0)
{
printf("Equal roots");
root1 = -b/deno;
printf("Root1 = %.f \t, Root 2 = %.f \t", root1, root2);
}
else
printf("Roots are imaginary");
}
```

Output:

Enter values of a, b, & C: 3 4 5

Roots are imaginary.

Q4

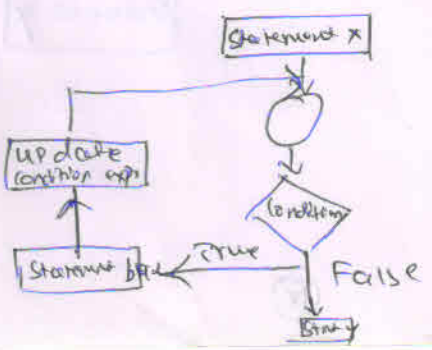
(a) What are iterative statements? Explain 3 types of iterative Statements with their syntax.

→ The three types of iterative statements are:-

① While Loop ② do-While Loop ③ for loop.

① While Loop:

Provides Mechanism to repeat one or more statements while a particular condition is True.



The condition is tested before any Statements in Statement block is executed. If condition is true, only then the statements will be executed otherwise, control will jump to statement y, which is outside the while Loop block.

Syntax:-

```
Statement x;
While (condition)
{
  Statement block;
}
Statement y;
```

E.g.

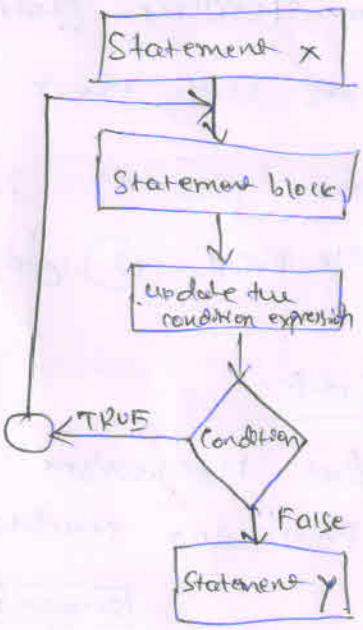
```
#include<stdio.h>
int main()
{
  int i=1;
  While (i<=10)
  {printf("%d", i);
    i=i+1;
  }
return 0;
}
```

ii) do-while loop

- The test condition is evaluated at the end of the loop.
- The body of the loop is executed atleast once.

Syntax:-

```
Statement x;
do
{
Statement block;
}while (condition);
Statement y;
```



E.g.

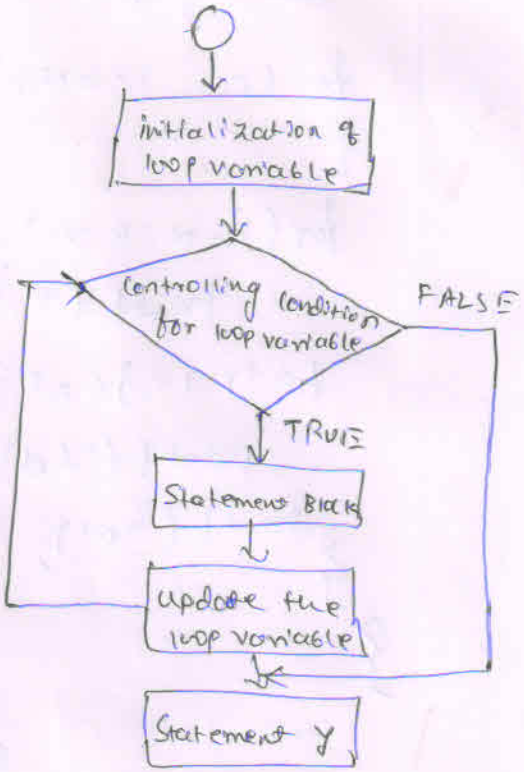
```
#include<stdio.h>
int main()
{ int i=1;
do
  {printf("%d", i);
    i=i+1;
  } while (i<=10);
}
```

For Loop:

- Provides Mechanism to repeat a task until a particular condition is true.

- Syntax:

```
for (initialization; condition; increment/decrement/update)
{
  Statement block;
}
Statement Y;
```



In syntax, initialization of loop variable allows the programmer to give it a value.

The condition specifies, expression is true the loop should continue itself.

with every iteration, the loop variable must be updated.

e.g.

```
#include<stdio.h>
int main()
{
  int i,n;
  printf("Enter the value of n");
  scanf("%d", &n);
 for(i=1; i<=n; i++)
  printf("%d", i);
}
```

Q. Write a program to print the following pattern.

```
      1
    1 2
  1 2 3
1 2 3 4
```

```
#include<stdio.h>
#define N 4
int main()
{
int i,j,k;
for(i=1; i<=N; i++)
{
for(k=N; k>=i; k--)
    printf(" ");
for(j=1; j<=i; j++)
    printf("%d",j);
printf("\n");
}
}
```

Module – 3

Q5 (a) Explain the syntax of function declaration and function definition with example.

Function declaration

Syntax:

return_data-type function_name(data_type variable_1, data_type variable2, ---)

function_name ⇒ valid name for function.

return_data-type ⇒ specifies the data type of the value that will be returned to the calling function as a result of the processing performed by the called function.

data type_variable 1, data-type_variable 2 ... ⇒ list of variables of specified data types. These variables are passed from the calling function to the called function.

e.g.

```
int func(int, char, int);
```

①

OR

```
int func(int num, char ch, int n);
```

② void print();

Function definition

When a function is ~~defined~~, space is allocated for that function in the memory.

A function definition comprises of 2 parts:-

① Function header ② Function body.

Syntax:

```
return_data_type function_name(data_type variables, data_type variables)
{
  ........
  Statements
  ........
  return(variables);
}
```

return_data_type function_name(data_type variables, data_type variables) ⟹ it's known as function header

The program statements written within { } is function body.

```
int swap(int x, int y) // Function header.
{
  int temp;
  temp = x;          // Function body
  x = y;
  y = temp;
}
```

Q5 (b) write a c-program to swap two numbers using call by reference method.

```
#include<stdio.h>
Void swap(int *, int *);
int main()
{
   int a=1, b=2, c=3, d=4;
   swap(&c, &d);
   printf(" In main c=%d, &d=%d ", c, d);
   swap( &a, &b);
   Printf (" In main a=%d, &b=%d", a, b);
}
Void swap (int *c, int *d)
{
   int temp;
   temp = *c;
   *c = *d;
   *d = temp;
}
```

output:

In main c=4, &d=3

In main a=2 & b=1

Q5 (c) Describe different types of Storage classes with examples.

→ Storage class defines the scope & lifetime of variables & (or functions declared within C program.

C supports 4 Storage classes:

1) automatic 2) register 3) external 4) Static

Syntax:

<Storage_class> <data type> <variable name>

① Auto Storage Class

- It is used to explicitly declare a variable with automatic storage
- It is default storage class for variables declared inside a block.

e.g.
```
auto int x.
```
x is an integer that has automatic storage.

② register Storage class

- When a variable is declared using register as its storage class, it is stored in a CPU register instead of RAM.
- e.g.
```
register int x;
```
- Register variables are used when quick access to the variable is needed.

③ extern Storage class

- It is used to give a reference of a global variable that is visible to all the program files.
- External variables may be declared outside any function in a source code file as any other variable is declared.

eg.
```
extern int x;
```

④ Static Storage Class

- It is the default storage for all global variables.
- ~~Memory~~ Static variables have a lifetime over the entire program

Eg
```
static int x = 10;
```

Q 6 a.

a. What is an array? Explain how arrays are declared and initialized with example.

→ An array is a collection of similar data elements. These data elements are of same data type.

Declaration of Arrays

- An array must be declared before being used.
- Syntax:

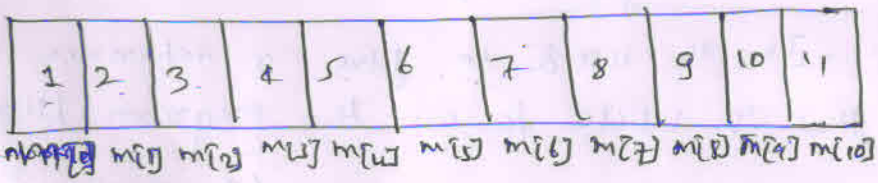
  type name[size]

  type ⇒ Data type - int, float, char, double.

  name ⇒ Any valid identifier

  size ⇒ Maximum number of values that the array can hold.

- e.g. int marks[10]; // Array of 10 elements.
  int m[11]; // Array of 11 elements



Initializing Arrays

There are 3 methods for initializing the arrays

① Initialize the elements during declarations

② Input values for the elements from the keyboard

③ Assign values to individual elements.

① Initializing Arrays during declarations

Elements of the array can be initialized at the time of declaration. When an array is initialized, the value for every element is provided.

Syntax:

type array_name[size] = {list of values};

e.g.

int marks[5] = {90, 82, 78, 95, 88};

- Also the programmer may omit the size during declaration

e.g. int marks[] = {98, 97, 90};

② Inputting values from keyboard

- An array can be filled by inputting values from the keyboard.
- In this method, a while/do-while/for loop is executed to input the value for each element of the array.

e.g.
```
int i, marks[10];
for(i=0; i<10; i++)
    scanf("%d", &marks[i]);
```

③ Assigning values to individual elements

- This is done using assignment operator.

e.g. marks[3] = 100.

Here 100 is assigned to 4th element of array.

Q.6 (b) Write a C program to transpose a 3x3 matrix

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i,j, mat[3][3], transpose_mat[3][3];
Printf("Enter elements of the matrix \n");
For (i=0; i<3; i++)
  for (j=0; j<3; j++)
    scanf("%d", &mat[i][j]);
```

```
printf("The elements of the matrix are\n");
    for (i=0; i<3; i++) { printf("\n");
        for(j=0; j<3; j++)
            printf("\t %d", mat[i][j]);
        }
    for(i=0; i<3; i++)
        {
        for(j=0; j<3; j++)
        transpose_mat[i][j] = mat[j][i];
        }
    printf("The elements of transposed matrix are\n");
    for (i=0; i<3; i++)
        {
        printf("\n");
        for(j=0; j<3; j++)
        printf("%d ", transpose_mat[i][j]);
        }
}
```

Output

```
Enter the elements of matrix
1 2 3 4 5 6 7 8 9
The elements of matrix are:
1  2  3
4  5  6
7  8  9
The elements of transposed matrix are:
1  4  7
2  5  8
3  6  9
```

Q6 (c) Applications of arrays

- Arrays are widely used to implement mathematical Vectors, matrices.
- Many databases include 1-D arrays whose elements are records.
- Arrays are used to implement other data structures like stacks, queues, heaps & Hash tables.
- Arrays can be used in sorting.

Q7 Module - 4

(a) Write a C program to convert characters of a string into upper case without using built in functions.

→
```
#include <stdio.h>
#include <conio.h>
int main()
{
char str[100], upper[100];
int i=0, j=0;
printf ("Enter the string \n");
gets(str);
while (str[i] != '\0')
{
if (str[i] >= 'a' && str[i] <= 'z')
        upper[j] = str[i] - 32;

else
        upper[j] = str[i];
    i++; j++;
}
upper = '\0';
printf ("The string converted into upper case is: \n");
puts(upper);
}
```

output:

Enter the string: hello
The string converted into upper case is: HELLO

Q 7

(b) Discuss the working of the following string manipulation functions with suitable examples.

i) Strcmp ii) Strlen iii) Strcpy iv) Strcat v) Strcmp.

-> i) Strcmp:

- This function compares the string pointed to by str1 to the string pointed to by str2

Syntax:-

int Strcmp (const char *str1, const char *str2);

e.g.
```
#include <stdio.h>
#include <string.h> ----
char str1[10] = "Hello";
char str2[10] = "HEY";
if (strcmp(str1, str2) == 0)
    printf("strings are equal");
else
    printf("strings are not equal");
```

O/P:

strings are not equal

ii) Strlen

This function calculates the length of the string 'str' up to but not including the null character.

Syntax:

size_t strlen (const char *str);

e.g.
```
#include <string.h>
---
char str[] = "Hello"
printf("Length of str is : %d", strlen(str));
---
```

Output
Length of Str is: 5

iii) Strcpy
- This function copies the string pointed to by str2 to Str1 including the null character of str2. It returns the argument Str1.

- Syntax:
  Char *Strcpy(Char *Str1, const Char *Str2);

- e.g.
```
#include<string.h>
  - - -
Char Str1[10], Str2[10] = "Hello";
Strcpy(Str1, Str2);
Printf("Str1: %s", Str1);
  - - -
```
- Output
  Hello.

iv) Strcat
- This function appends the string pointed to by 'str2' to the end of the string pointed to by 'Str1'. The terminating null character of 'Str1' is overwritten.

- Syntax
  Char *Strcat(Char *Str1, const Char *Str2)

- e.g.
```
#include<string.h>
  - - -
Char Str1[50] = "programming";
Char Str2[50] = "In C";
Strcat (Str1, Str2)
Printf ("Str1: %s", Str1);
  - - -
```
Output
Str1: programming in C

Q8

(a) Define pointer. Explain the declaration of a pointer variable with an example.

-> A pointer is a variable that contains the memory locations of another variable.

Syntax of declaring pointer variable.

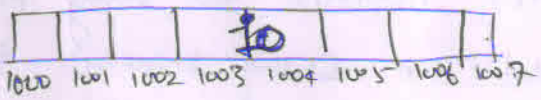
```
data_type *ptr_name;
```

e.g.
```
int *pnum;      // integer pointer
Char *pch;      // Character pointer
float *pfnum;   // floating point pointer.
```

// Consider the below snippet.

```
int x=10;
int *ptr
ptr = &x;
```

'ptr' is a pointer variable of integer type. 'ptr' is assigned the address of x.



Since x is integer, it will be allocated 2 bytes. Now ptr = 1003, starting address of 'x' in memory.

// program

```
#include<stdio.h>
int main()
{
 int num, *pnum;
 pnum = &num;
printf("Enter the number");
Scanf("%d", &num)
printf("the entered number is\n");
printf("%d", *pnum);
printf("the address of number in memory is %p", &num);
}
```

Output:

Enter the number: 10

The number that was entered is: 10

The address of number in memory is: FFDC

Q 8

(b) Mention the applications of pointers

→ ① To pass information back & forth between a function & its reference point.

② Enable programmers to return multiple data items from a function via function arguments

③ To pass arrays & strings as function arguments

④ For dynamic memory allocation of a variable.

⑤ To create complex data structures like Linked lists, Trees,

These are some of the applications, where pointers play important role in computation.

Q 8 (c) Develop a C program to ~~read &~~ compute the sum, mean & standard deviation of all elements of an array using pointers.

→

```
#include<stdio.h>
#include<math.h>
main()
{
    int i, num;
    float mean=0.0, var=0.0, Sd=0.0, arr[100], sum=0.0;
    float *fptr;
    Printf ("Enter the number of values");
    Scanf ("%d", &num);
    fptr = arr;
Printf ("Enter %d values", num);
```

```
for (i=0; i<num; ++i)
{
  scanf ("%f", fptr+i);
    sum = sum + *(fptr+i);
}
mean = sum/num;
for(i=0; i<num; ++i)
{
    var = var + (fptr[i] - mean)*(fptr[i]-mean);
}
variance.
    var = var/num;
    sd = sqrt(var);
Printf("Sum = %f, Mean = %f, Standard deviation = %f", sum, mean, sd);
}
```

Output:

```
Enter no. of values: 3
Enter values: 1 2 3
Sum = 6.000000
Mean = 2.000000
std dev = 0.8164
```

Module-5

Q 9

a. What is structure? explain the declaration of a structure with an example.

→ A structure is a user defined data type that can store related information together.

A structure is declared using "struct" keyword followed by name of the structure.

Syntax:-

```
struct <name>
{
  data_type var1;
  data_type var2;
  ....
};
```

e.g.

```
Struct student
{
  int r_no;
  Char name[20];
  Char course[20];
  Float fees;
}
```

Memory is allocated for structure when we declare a variable of structure.

e.g.

```
Struct Student stud1;
```

Structures can also be declared as:

```
Struct Student
{
  int r_no;
  Char name[20];
  char course[20];
  Float fees;
} stud1, stud2;
```

Q 9 (b) Differentiate between structures & Unions.

	Structure	Union
①	It is defined with 'Struct' keyword	It is defined with 'Union' keyword

②	Syntax:
	`Struct <name>`
	`{`
	`data_type var1;`
	`data_type var2;`
	`- - -`
	`} var1, var2, - -;`
		Syntax:-
		`Union <name>`
		`{`
		`data_type var1;`
		`data_type var2;`
		`- - -`
		`} var1, var2 - -;`

③	User can initialise multiple members at a time	User can initialise only 1st member ~~at a time~~.

	Structure	Union
④	User can access individual members at a given time	User can access only one member at a given time.
⑤	Size of structure is greater than or equal to sum of sizes of its members	Size of union is equal to the size of largest member.
⑥	Each member has a unique storage area in memory	Memory allocated is shared by individual members of union.

Q 9 (c) Develop a C program to read & display the information of all the students in the class.

→
```
#include<stdio.h>
int main()
{
  struct student
  {
    int roll_no;
    char name[20];
    int fees;
    char DOB[20];
  };
  struct student stud[50];
  int n,i;
  printf("Enter the number of students:");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    printf("Enter roll no\n");
    scanf("%d",&stud[i].roll_no);
```

```
printf ("Enter the name\n");
gets (stud[i].name);
printf ("Enter the fees\n");
scanf ("%d", &stud[i].fees);
printf ("Enter the DOB:");
gets (stud[i].DOB);
for (i=0; i<n; i++)
{
 printf ("Roll No. = %d", stud[i].rollno);
 printf ("Name = %s", stud[i].name);
 printf ("Fees = %d", stud[i].fees);
 printf ("DOB = %s", stud[i].DOB);
}
```

Output:

```
Enter the number of students: 2
  Enter the roll number: 1
  Enter the name: AbC
  Enter the fees: 1234
  Enter the DOB: 1 9 91
  Enter roll number: 2
  Enter the name: XYZ
  Enter the Rees: 5678
  Enter the DOB: 27 6 91

  Roll No. = 1
  Name = AbC
    Fees = 1234
    DOB = 1 9 91

  Roll NO = 2
  Name = XYZ
   Fees = 5678
   DOB = 27 6 91
```

Q 10

(a) Define Enumerated data type. Explain the declaration & access of enumerated datatypes with a code in C.

→ The enumerated data type is a user-defined type based on the standard integer type.

An enumeration consists of a set of named integer constants.

Syntax:

enum <name> { identifier$_1$, identifier$_2$, ---- identifier$_n$ };

Example

```
#include <stdio.h>
int main()
{
enum color{Red=2, Blue, Black=5, Green=7,
           Yellow, Purple, White=15};

Printf("Red = %d", Red);
Printf("Blue = %d", Blue);
Printf("Black = %d", Black);
printf("Green = %d", Green);
Printf("Yellow = %d", Yellow);
Printf("Purple = %d", Purple);
Printf("White = %d", White);
}
```

Output

```
Red = 2
Blue = 3
Black = 5
Green = 7
Yellow = 8
Purple = 9
White = 15
```

Q 10 (b) Explain the process of opening a file in C.

→ - A file must be opened before data can be read from it or written to it.

- fopen() - is used to open a file

Syntax:

```
FILE *fopen(const char *name, const char *mode);
```

e.g.

```
FILE *fp
fp= fopen ("student.txt", "r");
if (fp==NULL)
 {
   printf ("File error");
   exit(1);
 }
```

- Here, the file whose pathname is the string pointed by 'name' is opened in 'mode' specified.
- Upon successful execution, fopen() returns pointer-to-structure otherwise NULL.
- 'Mode' conveys to C the type of processing that will be done with the file.

The different modes are:-

'r' → open a text file for reading

'w' → open a text file for writing.

'a' → Append to a text file.

Q 10 (c) Write a C program to demonstrate fputc() function.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
int main()
{
 FILE *fp;
 size_t count;
 char str[] = "Good Morning";
 fp = fopen("welcome.txt", "w");
  if (fp==NULL)
  {
   printf("The could not be opened");
   exit(1);
  }
 count = fwrite(str, 1, strlen(str), fp);
 printf("%d bytes were written to the file", count);
 fclose(fp);
}
```

Output :

13 bytes were written to the file.

Prepared by
Prof. Pranesh Kulkarni -
Prof. Ravindra Patil -

HOD 09/12/19
Computer Science & Engineering
KLS Vishwanathrao Deshpande
Institute of Technology, Haliyal

Dean, Academics
KLS VDIT, HALIYAL