



<b>FOCP-1</b>
Lab Manual

**Department of Computer Science and  
Engineering  
The NorthCap University, Gurugram**

# **FOCP-1 Lab Manual**

## **CSL 106**

### **Faculty Incharges**

Ms. Gurjapna Anand  
Dr. Vandana  
Dr. Ruchika  
Ms. Neetu  
Ms. Priya  
Ms. Ruchika  
Ms. Jyoti Yadav



Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2022-23

*Published by:*

**School of Engineering and Technology**

**Department of Computer Science & Engineering**

## **The NorthCap University Gurugram**

- **Laboratory Manual is for Internal Circulation only**

### **© Copyright Reserved**

*No part of this Practical Record Book may be*

*reproduced, used, stored without prior permission of The NorthCap University*

Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

~~Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.~~

## **PREFACE**

**Fundamentals of Computer Programming -I Lab Manual is designed to meet the course and program requirements of the NCU curriculum for B.Tech I year students of all branches. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.**

**The Lab manual is written on the basis of “teach yourself pattern” and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The laboratory exercises will include familiarization with**

**Number System; Experiments on C Programming covering concepts like conditional statements, iteration, functions, pointers, arrays, strings, structures and file management. Finally, the students would require applying all of the stated C programming concepts by experimenting and building projects on their own. Groups of two shall be created for the students to work on the assigned project. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.**

**The lab manual is a part of the curriculum for the TheNorthCap University, Gurugram. Teacher's copy of the experimental results and answer for the questions are available as sample guidelines.**

**We hope that lab manual would be useful to students of CSE, ECE, ME and Civil branches and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.**

**Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.**

**Authors**

**The NorthCap University**

**Gurugram, India**

<b>CONTENTS</b>		
<b>S.No.</b>	<b>Details</b>	<b>Page No.</b>
	<b>Syllabus</b>	
<b>1</b>	<b>Introduction</b>	
<b>2</b>	<b>Lab Requirement</b>	
<b>3</b>	<b>General Instructions</b>	
<b>4</b>	<b>List of Experiments</b>	
<b>5</b>	<b>List of Flip Assignment</b>	
<b>6</b>	<b>List of Projects</b>	
<b>7</b>	<b>Rubrics</b>	
<b>8</b>	<b>Annexure 1 (Format of Lab Report)</b>	
<b>9</b>	<b>Annexure 2 (Format of Lab Certificate)</b>	

## SYLLABUS

<b>1. Department:CSE</b>	<b>Department of CSE</b>		
<b>2. Course Name:</b>  Fundamentals of computer Programming-1	<b>3. Course Code</b>	<b>4. L- P</b>	<b>5. Credits</b>
	<b>Code: CSL106</b>	2-4	4
<b>6. Type of Course (Check one):</b>	Programme Core <input type="checkbox"/> Programme Elective <input type="checkbox"/> Open Elective <input type="checkbox"/>		
<b>7. Frequency of offering (check one):</b> Odd <input checked="" type="checkbox"/> Even <input type="checkbox"/> Either Sem. <input type="checkbox"/> Every Sem. <input type="checkbox"/>			
<b>8. Brief Syllabus:</b>  This course is foundational for all streams of engineering, be it Mechanical, Civil, Electronics or Computer Science. It is an elementary course for all students coming in first semester so that they get introduced with the concepts of computer working & programming language. This course focuses on intrinsic concepts of programming language that helps the students to mutate from one language to another in future. It provides the sagacity of procedural programming approach applied in C programming language. In this course, students will be able to fathom all the pivotal concepts of C language. They will be able to write the code of a program by developing logic with progression to writing pseudo codes, designing flowcharts and finally developing management projects.			
<b>9. Total lecture and Practical Hours for this course</b> <b>100 Hours</b> The class size is maximum 30 learners.			

## 10. Course Outcomes (COs)

Possible usefulness of this course after its completion i.e. how this course will be practically useful to him once it is completed

<b>CO 1</b>	Understand the working of a computer system with its all peripherals, and master the conversion from one base of number system to another base
<b>CO 2</b>	Analyze the problem and apply logic to design solutions to problems
<b>CO 3</b>	Develop the logic for a given problem and then progress to writing code and relate the programming concepts to algorithms and comprehend the basic programming constructs used in C language and write structured programs.
<b>CO 4</b>	Address the concepts of arrays, pointers, and different storage classes in C
<b>CO 5</b>	Apply concepts of pre-processor directives and files to develop management system and using functions of graphics to design different shapes and interface.

## 11. UNIT WISE DETAILS No. of Units: -5

### Unit I: Introduction to Computers and Number systems

**Hours: 12**

Introduction to Computer Fundamentals: Parts of computer, Generations, Types, Applications, I/O devices, Number Systems with their importance and exercises, Computer Networks, basic architecture and topologies, Types of Programming languages and current technologies used in Real scenarios.

### Unit II: Hands-on Pseudocodes, Flowcharts, Algorithms.

**Hours: 18**

Pseudocodes, and Flowcharts with exercises, Practice on Pseudocodes, and moving to algorithms, Introduction to Programming, Programming Languages, Assembler, Compiler, Interpreter, Linker, and Loader.

### Unit III: Introduction to C, Loops in C

**Hours: 15**

Introduction to C, data types, variables, operators, Implementing expressions, introduction to loops, control flow, sequential, selection (if-else, switch) iterative control statements (for), Do, while, Jumping statements,



<b>Unit IV: Functions, Pointers, Arrays, Strings in C</b> Function definition and declaration in C, Call by value and call by reference, Recursion with implementation. Pointers in C, Arrays and Strings.	<b>Hours: 21</b>
<b>Unit V: Structures, Storage classes, File management in C</b> Structures in C, Storage classes with their importance, Using pointers in Structures with implementation, File Management in C, File operations	<b>Hours: 16</b>
<b>12. Guided Project (No. of Hours): 9</b>  <b>Unguided Project (No. of Hours): 9</b>	
<b>13. Brief Description of Self-learning component by students (through books/resource material etc.): Topics:</b> Translators (Compiler, Interpreter, Assembler, Linker and Loader) and current technology in real scenarios (case study).	
<b>14. Books Recommended:</b>  <b>Text Books:</b> <ul style="list-style-type: none"> <li>• Yashwant Karnetkar, "Let Us C", BPB Publications, 16th Edition</li> <li>• Byron S. Gottfried, "Programming with C", McGraw-Hill, 4th Edition</li> </ul> <b>Reference Books:</b> <ul style="list-style-type: none"> <li>• J.B. Dixit, "Fundamentals of Computers &amp; Programming in C", Laxmi Publications, 2nd Edition</li> <li>• Yashwant Karnetkar, "Test your C Skills", BPB Publications, 5th Edition</li> </ul> <b>Reference websites:</b> <ul style="list-style-type: none"> <li><a href="https://www.tutorialspoint.com/cprogramming/">https://www.tutorialspoint.com/cprogramming/</a></li> <li><a href="https://nptel.ac.in/courses/106104128/">https://nptel.ac.in/courses/106104128/</a></li> </ul>	

## 1. INTRODUCTION

That ‘learning is a continuous process’ cannot be over emphasized. It is a more of learning practical concepts with little theoretical knowledge. Thus practical makes an integral part of a learning process.

The purpose of conducting experiments can be stated as follows:

- To familiarize the students with the basic concepts of C, programming skill development and the take home assignments mainly implementation-oriented which have to be coded in C language. The lab sessions will be based on implementing different concepts on a topics discussed in class.
- Observing basic structure and characteristics of the C programming language.
- Reporting and analyzing the complexities.
- Hands on experience on the programs

## **2. LAB REQUIREMENTS**

<b>S.No.</b>	<b>Requirements</b>	<b>Details</b>
<b>1</b>	<b>Software Requirements</b>	DevC++
<b>2</b>	<b>Operating System</b>	Windows Operating System
<b>3</b>	<b>Hardware Requirements</b>	Windows and Linux: Intel 64/32 or AMD Athlon 64/32, or AMD Opteron processor 2 GB RAM 80 GB hard disk space
<b>4</b>	<b>Required Bandwidth</b>	NA

## **3. GENERAL INSTRUCTIONS**

### **3.1 General discipline in the lab**

- Students must turn up in time and contact concerned faculty.
- Students will not be allowed to enter late in the lab.
- Students will not leave the class until the period is over.
- Students should come prepared for their experiment by revising the concepts taught in class.
- Programs implemented with their output should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
- Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
- Violating the above code of conduct may attract disciplinary action.
- Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.

### **3.2 Attendance**

- Attendance in the class is compulsory.
- Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
- On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the practicals. No attendance will be granted for such case.

### **3.3 Preparation and Performance**

- Students should come to the lab thoroughly prepared on the practicals they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.
- Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
- Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.

- Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
- Refer **Annexure 1** for Lab Report Format

#### **4. LIST OF EXPERIMENTS**

<b>S.No.</b>	<b>Experiments</b>
1	Number Systems
2	Operators and Expressions
3	Loops- If-else, for loop
4	While and do-while loop, switch-case
5	Functions
6	Arrays
7	Strings
8	Pointers
9	Structures
10	Files

#### **5. LIST OF PROJECTS**

1. Library Management System
2. Bank Management System
3. Inventory Management System
4. Flight Ticket booking system
5. Bus Reservation system
6. Garments Management System
7. College Record management system

8. Telephone Directory record management system
9. Blood Bank Management system
10. School Fee Management System
11. School Bus Transport System
12. QuizGame

## 6. RUBRICS

Marks Distribution				
S. No	TYPE OF COURSE	PARTICULAR	ALLOTTE D RANGE OF MARKS	PASS CRITERIA
1	Theory+Practical Course (L-T-P/L-T-0/L-0-P/L-0-0)	End Term Project	25%	Must Secure 30% Marks Out of Combined Marks of Minor Test Plus Major Test with Overall 40% Marks in Total.
		Minor Test	15%	
		Major Test	30%	
		Class Test/ Assignment	15%	
		Online Test	5%	
		Class Participation Evaluation Through Class Tests/Practice/Assignments/Presentation/Quiz	10%	

**Annexure 1**

**FOCP-1  
(CSL 106)**

Lab Practical Report



Faculty name Ms. Jyoti Yadav

Student name.: Saarmeen

Roll No.: 22CSU119

Semester: 1st

Group: CS-5

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2022-23

## INDEX

S. No	Experiment	Page No.	Date of Experiment	Date of Submission	Marks	CO Covered	Signature
1.	Practical -1	15	5-9-22				
2.	Practical -2	18	12-9-22				
3.	Practical -3	26	19-9-22				
4.	Practical -4	35	26-9-22				
5.	Practical -5	48	3-10-22				
6.	Practical -6	53	10-10-22				
7.	Practical -7	63	9-12-22				


### **Practical No. 1**

Student Name and Roll Number: Saarmeen 22csu119

Semester /Section: 1st CSE-C

Link to Code:

Date:

Faculty Signature:

Remarks:

### **Objective**

To familiarize the students about Number Systems

### **Program Outcome**

- The students will be able to represent numbers in various numeration systems , convert a numeral from one base to another base and perform arithmetic operations in bases other than ten



### Background Study:

A number system is defined as a system of writing for expressing numbers. It is the mathematical notation for representing numbers of a given set by using digits in a consistent manner. It provides a unique representation of every number and represents the arithmetic and algebraic structure of the figures. It also allows us to operate arithmetic operations like addition, subtraction, and division.

There are various types of the number system in mathematics. The four most common number system types are:

System	Base	Symbols
Decimal	10	0, 1, ... 9
Binary	2	0, 1
Octal	8	0, 1, ... 7
Hexa-decimal	16	0, 1, ... 9, A, B, ... F

### Suggested Question Bank

- What is the decimal equivalent of the number  $3A_{16}$ ?  
 $3A$   
 $3 \ 10$   
 $3 \cdot 10^1 + 10 \cdot 10^0$   
 $30 + 10$   
 $40$
- What is the 8 bit unsigned binary result of  $56_{10} - 31_{10}$

56-31

25

33

3. What is the result of adding  $7_{10}$  and  $-4_{10}$  using 8 bit signed binary notation?

7 -4

0111 1100

0011

4. Which of the following 4 bit Excess 3 numbers is equivalent to  $5_{10}$ ?

510+333

843

1000 0100 0011

100001000011

5. Consider the equation  $(123)_5 = (x8)_y$  with x and y as unknown. The number of possible solutions is \_\_\_\_\_ .

$$1*5^2 + 2*5^1 + 3*5^0 = x*y^1 + 8*y^0$$

$$38 = xy + 8$$

$$xy = 30$$

No of possible solutions is 3

6. The range of integers that can be represented by an n bit 2's complement number system is:

$$-2^{(n-1)} \text{ to } 2^{(n-1)} - 1$$

7. Convert binary 11111110010 to hexadecimal.

1111 1111 0010

15 15 2

F F 2

FF2

8. The representation of octal number  $(532.2)_8$  in decimal is \_\_\_\_\_

$$5*10^2 + 3*10^1 + 2*10^0$$

346

346.25

9. The decimal equivalent of the octal number  $(645)_8$  is \_\_\_\_\_

$$6*8^2 + 4*8^1 + 5*8^0$$

384+32+5

- 421
10. The quantity of double word is \_\_\_\_\_  
32 bits
11. Octal to binary conversion:  $(24)_8 = ?$   
2 4  
010 100  
010100
12. Convert binary to octal:  $(110110001010)_2 = ?$   
110 110 001 010  
6 6 1 2  
6612
13. The octal number  $(651.124)_8$  is equivalent to (\_\_\_\_\_) <sub>10</sub>.  
 $6 \cdot 8^2 + 5 \cdot 8^1 + 1 \cdot 8^0$   
425  
425.0128203125
14. Convert the hexadecimal number  $(1E2)_{16}$  to decimal:  
1 E 2  
1 14 2  
 $1 \cdot 16^2 + 14 \cdot 16 + 2$   
 $256 + 224 + 2$   
498
15. Let  $r$  denote number system radix. The only value(s) of  $r$  that satisfy the equation  
 $\sqrt{121_r} = 11_r$  is/are  
 $(121)^{1/2} = 11$   
 $(1 \cdot r^2 + 2 \cdot r^1 + 1 \cdot r^0)^{1/2} = 1 \cdot r^1 + 1 \cdot r^0$   
 $(r^2 + 2r + 1)^{1/2} = 1 + r$   
 $((1+r)^2)^{1/2} = 1 + r$   
 $1 + r = 1 + r$   
Any value greater than 2

### Practical No. 2

Student Name and Roll Number: Saarmeen 22CSU119

Semester /Section: 1st CSE-C

Link to Code:

Date:

Faculty Signature:

Remarks:

## Objective

To familiarize the students with different operators in C and how to use them to solve expressions.

## Program Outcome

Through this lecture, students will learn the syntax of C programming and understand the control flow of C program

## Background Study:

An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition.

C has a wide range of operators to perform various operations.

### *C Arithmetic Operators*

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning of Operator
+	Addition or unary plus
-	subtraction or unary minus
*	Multiplication

/	Division
%	remainder after division (modulo division)

### *C Increment and Decrement Operators*

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

### *C Relational Operators*

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1

<=	Less than or equal to	5 <= 3 is evaluated to 0
----	-----------------------	--------------------------

### *C Logical Operators*

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

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

### **List of Programs**

1. Write a program that reads a floating-point number and then displays the right-most digit of the integral part of the number

*Test case:*

Enter two operands: 32.5

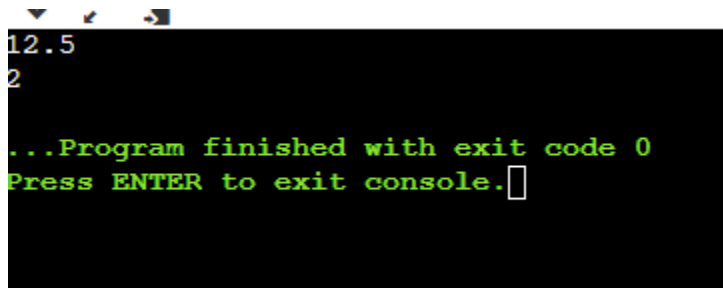
O/P: 2

*Code:*

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

```
int main()
{
    float num;
    int x;
    scanf("%f",&num);
    x=(int)num;
    printf("%d",x%10);
    return 0;
}
```

*Output (with Screenshots):*



```
12.5
2
...Program finished with exit code 0
Press ENTER to exit console.
```

2. Given an integer number, write a program that displays the number as follows:

First line: all digits

Second line: all except first digit

Third line: all except first two digits

.....

Last line: The last digit

## Program Outcome

The number 5678 will be displayed as:

5 6 7 8

6 7 8

7 8

8

## Code

```
#include<stdio.h>

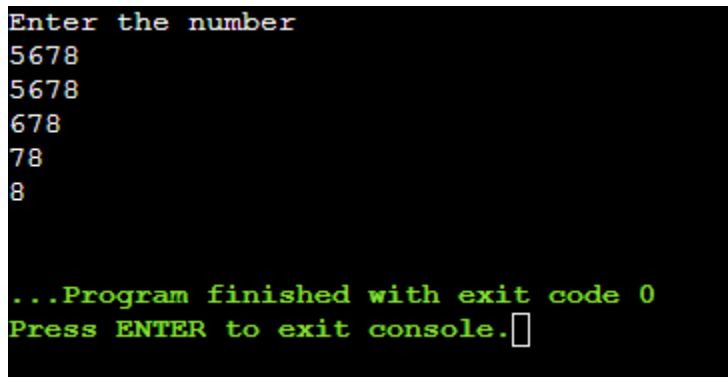
#include<math.h>

int main()
{
    int x,y=0,z,a,b;
    printf("Enter the number\n");
    scanf("%d",&x);
    a=x;
    for(x;x!=0;x=x/10)
        y++;
    for(z=y;z>=1;z--)
    {
        b=a%(int)(pow(10,z));
        printf("%d\n",b);
```



```
}  
  
return 0;  
  
}
```

### Output: (with Screenshots)



```
Enter the number  
5678  
5678  
678  
78  
8  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

### Suggested Question Bank

1. A C program contains the following declarations:

```
int i,j;  
long ix ;  
short s;  
float x;  
double dx;  
char c;
```

Determine the data type of each of the following expressions.

- |                       |                 |
|-----------------------|-----------------|
| (a) $i + c$           | (f) $i * x + j$ |
| (b) $x + c$           | (g) $s + c$     |
| (c) $dx + x$          |                 |
| (d) $((int) dx) + ix$ |                 |
| (e) $i + x$           |                 |

(a)  $i + c$  - integer

(b)  $x + c$  - float

(c)dx+x - float or double

(d)((int)dx)+ix - long int

(e)i+x - float

(f)ix+j - long float or double

(g)s+c - integer

2. What is the output of the following?

```
#include<stdio.h>
int main()
{
    unsigned char a = 5, b = 9;
    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;
}
```

a = 5, b = 9

a&b = 1

a|b = 13

a^b = 12

~a = 250

b<<1 = 18

b>>1 = 4

## LIST OF FLIP EXPERIMENTS

1. Write a program in C to print the alphabet E, using dollar (\$)

Definition of Done: The output contains the height of 7 characters and width of 6 and 5 characters.

```
#include <stdio.h>
int main()
{
    char x;
    printf("Enter the character\n");
    scanf("%c",&x);
    if(x=='$')
        printf("EEEEEE\nE\nE\nEEEEEE\nE\nE\nEEEEEE");
    else
        return 0;
}
```

2. Write a program in C to convert 1000 meter per hour to miles per hour.

```
#include <stdio.h>

int main()
{
    float x,y;

    printf("Enter the value in metres per hour\n");

    scanf("%f",&x);

    y=x/1600;

    printf("The value in miles per hour is %f",y);

    return 0;
}
```

3. Write a program in C to input the seconds and output the number of hours, minutes and seconds.

Definition of Done:

- The program asks the user to input the total seconds.
- The programs outputs in the format as:

Hours:

Minutes:

Seconds:

- For example, if there are 9945 seconds, then:

Hours: 2

Minutes:45

Seconds:45

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

```
int main()
```

```
{
```

```
int x,y,z,a,b,c;
```

```
printf("Enter the number of seconds\n");
```

```
scanf("%d",&x);
```

```
y=x/3600;
```

```
z=x%3600;
```

```
a=z/60;
```

```
c=z%60;
```

```
printf("Hours:%d\nMinutes:%d\nSeconds:%d\n",y,a,c);
```

```
return 0;
```

```
}
```

### Practical No: 3

Student Name and Roll Number: Saarmeen 22CSU119

Semester /Section: 1st CSE-C

Link to Code:

Date:

Faculty Signature:

Remarks:

▪

## **Objective**

To familiarize the students with if-else loop.

## **Program Outcome**

The students will learn the concept of looping in C. They will be able to understand the different types of statements encountered in C.

## **Background Study**

The syntax of the if statement in C programming is:

if (test expression)

{

    // statements to be executed if the test expression is true

}

How if statement works?

The if statement evaluates the test expression inside the parenthesis ().

1. If the test expression is evaluated to true, statements inside the body of if are executed.
2. If the test expression is evaluated to false, statements inside the body of if are not executed.

## **LIST OF PROGRAMS**

1. While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses.

```
#include <stdio.h>
int main()
{
    int x;
    printf("Enter the amount\n");
    scanf("%d",&x);
    if(x>=1000){
        x=x-x/10;
    }else{}
    printf("Your total is %d",x);
    return 0;
}
```

2. The marks obtained by a student in 5 different subjects are input through the keyboard. The student gets a division as per the following rules: Percentage above or equal to 60 - First division Percentage between 50 and 59 - Second division Percentage between 40 and 49 - Third division Percentage less than 40 - Fail Write a program to calculate the division obtained by the student.

```
#include <stdio.h>

int main()
{
    int marks;

    printf("Enter your marks\n");

    scanf("%d",&marks);

    if(marks>=60 && marks<=100)

        printf("You scored First Division.");
}
```

```
else if(marks>=50 && marks<=59)

    printf("You scored Second Division.");

else if(marks>=40 && marks<=49)

    printf("You scored Third Division.");

else if(marks<0)

    printf("You Failed.");

else

return 0;

}
```

3. Write a program to check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees.

```
#include <stdio.h>

int main()

{

    int angle1,angle2,angle3;

    printf("Enter your angles\n");

    scanf("%d%d%d",&angle1,&angle2,&angle3);

    if(angle1+angle2+angle3 == 180)

        printf("A triangle can be created.");

    else

        printf("A triangle cannot be created.");

}
```

```
return 0;  
  
}
```

4. Write a program in C to read the age and display whether the candidate is eligible to vote or not.

Definition of Done:

- The program should ask the user to enter an integer. If it is floating, ask the user to enter appropriate number
- The program should use if-else statement

```
#include <stdio.h>  
  
int main()  
{  
  
    int age;  
  
    printf("Enter your age\n");  
  
    scanf("%d",&age);  
  
    if(age>=18)  
        printf("You are eligible to vote.");  
  
    else  
        printf("You are not eligible to vote.");  
  
    return 0;  
  
}
```

### Suggested Question Bank

1. Write a program in C to find the roots of a quadratic equation.

Definition of Done:



- Firstly, the program shall display the conditions of no solution, real, and imaginary roots.
- Then it shall ask the user to enter the values of A,B,C.
- Then, it shall output as:
- “The equation ..... contains ..... roots which are .....” or “The equation..... has no solution”.

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

```
int main()
```

```
{
```

```
    int a,b,c,d;
```

```
    printf("Enter A,B and C\n");
```

```
    scanf("%d%d%d",&a,&b,&c);
```

```
    d=b*b - 4*a*c;
```

```
    printf("D=%d\n",d);
```

```
    if(d>0){
```

```
        printf("There are 2 real distinct roots.\n");
```

```
        printf("The equation %dx^2+%dx+%d=0 contains distinct roots which are real",a,b,c);
```

```
    }else if(d==0){
```

```
        printf("There are 2 real equal roots.\n");
```

```
        printf("The equation %dx^2+%dx+%d=0 contains equal roots which are real",a,b,c);
```

```
    }else{
```

```
        printf("There are 2 imaginary distinct roots.\n");
```

```
        printf("The equation %dx^2+%dx+%d=0 contains distinct roots which are imaginary",a,b,c);
```

```
    }
```

```
return 0;  
  
}
```

2. Write a program to salary of 10 employees of a company and also find and print the max and min salary.

```
#include<stdio.h>  
  
#include<math.h>  
  
int main()  
{  
  
    int a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,ab,bc,cd,ef,gh,ij;  
  
    printf("Enter the salaries of the employees\n");  
  
    scanf("%d%d%d%d%d%d%d%d%d%d",&a,&b,&c,&d,&e,&f,&g,&h,&i,&j);  
  
    k=fmax(a,b);  
  
    l=fmax(c,d);  
  
    m=fmax(e,f);  
  
    n=fmax(g,h);  
  
    o=fmax(i,j);  
  
    p=fmin(a,b);  
  
    q=fmin(c,d);  
  
    r=fmin(e,f);  
  
    s=fmin(g,h);  
  
    t=fmin(i,j);  
  
    u=fmax(k,l);
```

```
v=fmax(m,n);  
  
w=fmax(t,o);  
  
x=fmax(u,v);  
  
y=fmax(w,x);  
  
ab=fmin(p,q);  
  
cd=fmin(r,s);  
  
ef=fmin(t,ab);  
  
gh=fmin(cd,ef);  
  
printf("The maximum is %d\n",y);  
  
printf("The minimum is %d",gh);  
  
return 0;  
  
}
```

## **LIST OF FLIP EXPERIMENTS**

1. What is the usage of nested if statements?

Nested if statement is a decision-making statement that is used when we have to make decisions based on certain conditions. It allows you to test multiple criteria and increases the number of possible outcomes.

2. What will be the output of the C program?

```
#include<stdio.h>  
int main()  
{  
    int i = 5, j = 6, k = 7;  
    if(i > j == k)  
        printf("%d %d %d", i++, ++j, --k);  
    else
```

```
        printf("%d %d %d", i, j, k);  
    return 0;  
}
```

5 6 7

3. What will be the output of the C program?

```
#include<stdio.h>  
int main()  
{  
    int i = 2;  
    if(i == (1, 2))  
        printf("Hai");  
    else  
        printf("No Hai");  
    return 0;  
}
```

- a) Compilation error
- b) Runtime error
- c) Hai
- d) No Hai
- c) Hai

4. What will be the output of the C program?

```
#include<stdio.h>  
int main()  
{  
    if(sizeof(0))  
        printf("Hai");  
    else  
        printf("Bye");  
    return 0;  
}
```

- a) Hai
- b) Bye
- c) Compilation Error
- d) None
- a) Hai

### **Practical No: 4**

Student Name and Roll Number: Saarmeen 22CSU119

Semester /Section:1st CSE-C

Link to Code:

Date:

Faculty Signature:

Remarks:

▪

### **Objective**

To familiarize the students with switch-case statements.

### **Program Outcome**

The students will learn the concept of looping in C. They will be able to understand where and how switch case statements are implemented.

### **Background Study**

The switch statement allows us to execute one code block among many alternatives.

You can do the same thing with the if...else..if ladder. However, the syntax of the switch statement is much easier to read and write.

---

Syntax of switch...case

switch (expression)

{

case constant1:

    // statements

    break;

case constant2:

```
// statements  
break;  
.  
.  
.  
default:  
    // default statements  
}
```

#### *How does the switch statement work?*

The expression is evaluated once and compared with the values of each case label.

- If there is a match, the corresponding statements after the matching label are executed. For example, if the value of the expression is equal to constant2, statements after case constant2: are executed until break is encountered.
- If there is no match, the default statements are executed.
- If we do not use break, all statements after the matching label are executed.
- The default clause inside the switch statement is optional.

#### *Rules for switch-case*

- 1) The expression provided in the switch should result in a constant value otherwise it would not be valid.

Valid expressions for switch:

// Constant expressions allowed

switch(1+2+23)

switch(1\*2+3%4)

// Variable expression are allowed provided

// they are assigned with fixed values

switch(a\*b+c\*d)

switch(a+b+c)

- 2) Duplicate case values are not allowed.
- 3) The default statement is optional. Even if the switch case statement do not have a default statement, it would run without any problem.
- 4) The break statement is used inside the switch to terminate a statement sequence. When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- 5) The break statement is optional. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.
- 6) Nesting of switch statements are allowed, which means you can have switch statements inside another switch. However nested switch statements should be avoided as it makes program more complex and less readable.

## **LIST OF PROGRAMS**

1. Write a C Program with an algorithm and flowchart to make a simple calculator performing +, -, \*, / using switch-case. The program takes two integer operands and one operator from the user, performs the operation and then prints the result.

### **Code**

```
#include<stdio.h>

int main(){
    int x,y;
    char z;

    printf("Enter the operator(+, -, *, /)\n");
    scanf("%c",&z);
    printf("Enter the two numbers.\n");
```

```
scanf("%d%d",&x,&y);
switch(z){
    case '+' :
        printf("%d+%d=%d",x,y,x+y);
        break;
    case '-' :
        printf("%d-%d=%d",x,y,x-y);
        break;
    case '*' :
        printf("%d*%d=%d",x,y,x*y);
        break;
    case '/' :
        printf("%d/%d=%d",x,y,x/y);
        break;
    default:
        printf("operator not correct\n");
}
return 0;
}
```

### Algorithm

Step 1: Start

Step 2: Input the operator

Step 3: Read the operator



Step 4: Input two operands

Step 5: Read the operands

Step 6: If operator is '+' go to step 6.1 otherwise go to step 7

Step 6.1: Add first operand to second operand

Step 6.2: Display the sum

Step 6.3: Go to step 11

Step 7: If operator is '-' go to step 7.1 otherwise go to step 8

Step 7.1: Subtract first operand from second operand

Step 7.2: Display the difference

Step 7.3: Go to step 11

Step 8: If operator is '\*' go to step 8.1 otherwise go to step 9

Step 8.1: Multiply the first operand to second operand

Step 8.2: Display the product

Step 8.3: Go to step 11

Step 9: If operator is '/' go to step 9.1 otherwise go to step 10

Step 9.1: Divide first operand from second operand

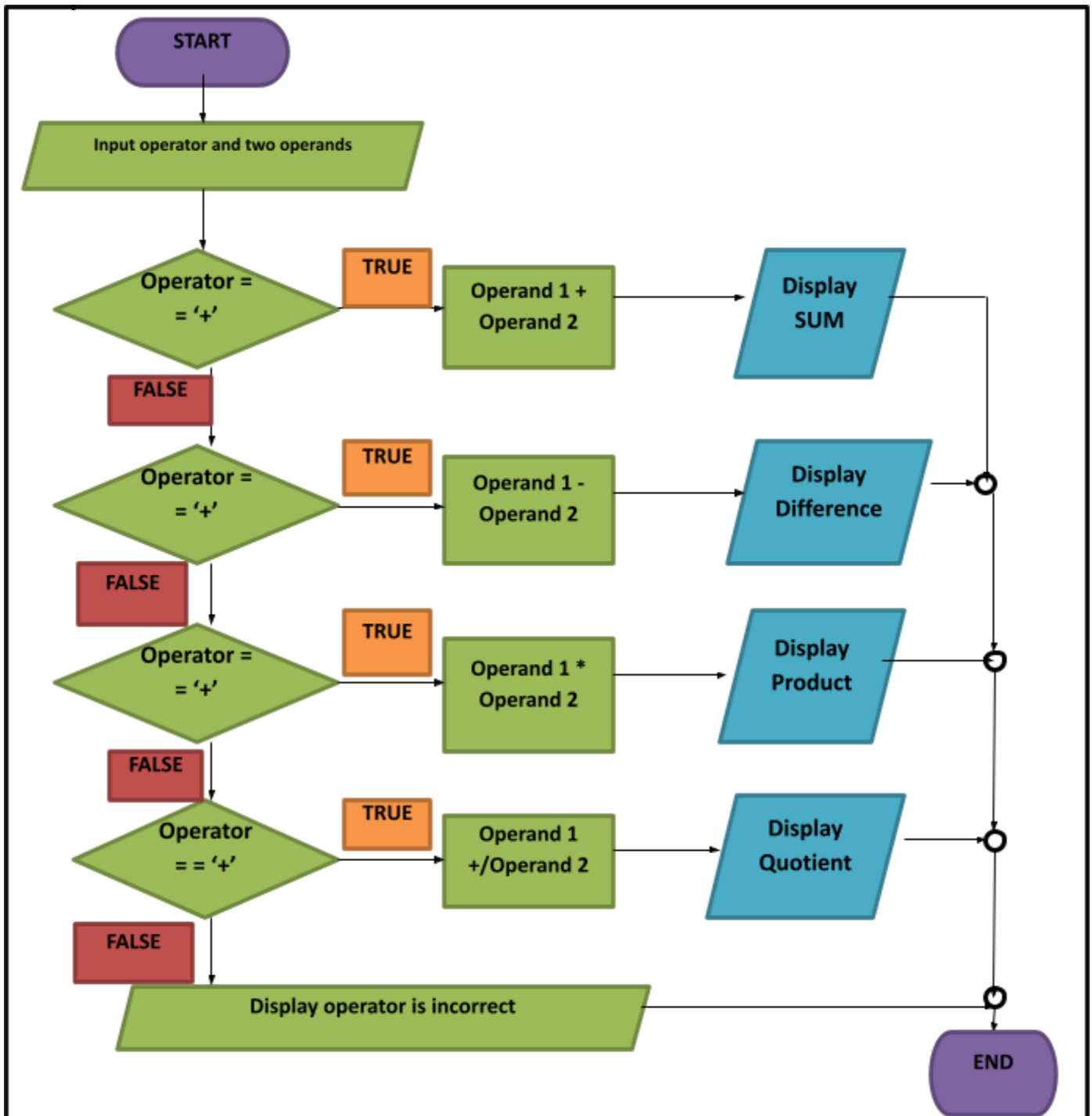
Step 9.2: Display the quotient

Step 9.3: Go to step 11

Step 10: Display operator is incorrect

Step 11: End

**Flowchart**



2. A character is entered through keyboard. Write a C program and its algorithm to determine whether the character entered is a capital letter, a small case letter, a digit or a special symbol using switch case. The following table shows the range of ASCII values for various characters.

Characters	ASCII values
A – Z	65 – 90
a – z	97 – 122
0 – 9	48 – 57
Special symbols	0 – 47, 58 – 64, 91 – 96, 123 – 127

### **Algorithm:**

Step 1: Start

Step 2:Input a character

Step 3:Convert the character to an integer value

Step 4:Check whether the value of integer is between 65 and 90 or not. If true go to step 4.1 otherwise go to step 5

Step 4.1:Display the character is a capital letter.

Step 4.2:Go to step 8

Step 5: Check whether the value of integer is between 97 and 122 or not. If true go to step 5.1 otherwise go to step 6

Step 5.1:Display the character is a small letter.

Step 5.2:Go to step 8

Step 6: Check whether the value of integer is between 48 and 57 or not. If true go to step 6.1 otherwise go to step 7

Step 6.1:Display the character is a digit.

Step 6.2:Go to step 8

Step 7: Check whether the value of integer is between 0 and 47 or 58 and 64 or 91 and 96 or 123 and 127 or not. If true go to step 7.1 otherwise go to step 8

Step 7.1: Display the character is a special symbol.

Step 7.2: Go to step 8

Step 8: End

### **Code**

```
#include <stdio.h>

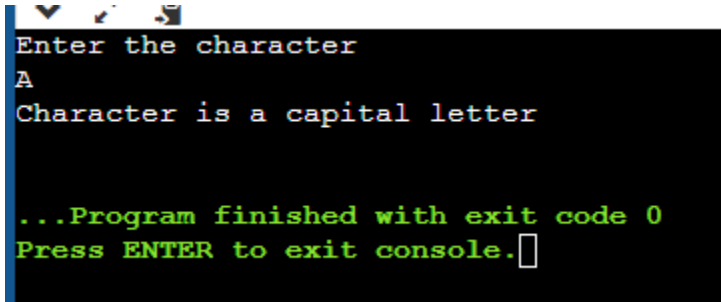
int main()
{
    char a;

    printf("Enter the character\n");
    scanf("%c",&a);
    switch(a){
        case 'A'...'Z':
            printf("Character is a capital letter\n");
            break;
        case 'a'...'z':
            printf("Character is a lower letter\n");
            break;
        case '0'...'9':
            printf("Character is a Digit \n");
            break;
```

default:

```
printf("Character is a Special character\n");  
}  
return 0;  
}
```

### Output: (with Screenshots)



```
Enter the character  
A  
Character is a capital letter  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

3. Write a C program using switch case to display grade of students based on the total marks obtained by the student in five subjects

Here is the range of Grades:

Marks  $\geq 90$  : Grade A

Marks  $\geq 70$  &&  $< 90$  : Grade B

Marks  $\geq 50$  &&  $< 70$  : Grade C

Marks  $< 50$  : Grade D

Marks  $< 50$  : Fail

*Program Outcome*

Enter marks of 5 subjects

97 89 78 87 68

Grade : B

### **Code**

```
#include <stdio.h>

int main()
{
    int marks;

    printf("Enter your marks\n");
    scanf("%d",&marks);
    if(marks>=90 && marks<=100)
        printf("You scored A Grade.");
    else if(marks>=70 && marks<=89)
        printf("You scored B Grade.");
    else if(marks>=50 && marks<=69)
        printf("You scored C Grade.");
    else if(marks<50)
        printf("You Failed.");
    else
        return 0;
}
```

### **Output: (with Screenshots)**

```
Enter your marks
77
You scored B Grade.

...Program finished with exit code 0
Press ENTER to exit console.
```

### Suggested Question Bank

1. Why do we use of break keyword?

To terminate a loop or during switch statements to differentiate between loops

2. Can we assign float value to a selection variable in switch?

No you can not assign value of a selection variable in switch as float or double

3. In what ways does a switch statement differ from if statement?

During an if statement, the compiler goes through all the loops from beginning whichever case might be suitable, but in a switch statement all cases are pre recorded in the memory, if you choose any case your compiler will directly run that case rather than going through the whole loop.

4. Which of the following statement is correct for switch controlling expression?

- a) Only int can be used in “switch” control expression
- b) Both int and char can be used in “switch” control expression.
- c) All types i.e. int, char and float can be used in “switch” control expression
- d) “switch” control expression can be empty as well

b) both int and char can be used in “switch” control expression.

### LIST OF FLIP EXPERIMENTS

1. What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

```
#include <stdio.h>
void main()
{
    double ch;
    printf("enter a value between 1 to 2:");
    scanf("%lf", &ch);
    switch (ch)
    {
        case 1:
```

```
        printf("1");  
        break;  
    case 2:  
        printf("2");  
        break;  
    }  
}
```

Switch value not an integer

2) What will be the output of the following C code? (Assuming that we have entered the value 1 in the standard input)

```
#include <stdio.h>  
void main()  
{  
    int ch;  
    printf("enter a value between 1 to 2:");  
    scanf("%d", &ch);  
    switch (ch)  
    {  
        case 1:  
            printf("1\n");  
        default:  
            printf("2\n");  
    }  
}
```

1

2

3) What will be the output of the following C code?

```
#include <stdio.h>  
int main()  
{  
    int a = 1, b = 1;  
    switch (a)  
    {  
        case a*b:  
            printf("yes ");  
        case a-b:  
            printf("no\n");  
    }  
}
```



```
        break;
    }
}
```

Case not an integer constant

4) What is the output of the below program?

```
#include <stdio.h>
int main()
{
    int i = 0;
    switch (i)
    {
        case '0': printf("Geeks");
            break;
        case '1': printf("Quiz");
            break;
        default: printf("GeeksQuiz");
    }
    return 0;
}
```

GeeksQuiz

5) What will be the output of the following C program segment? (GATE CS 2012)

```
char inchar = 'A';
switch (inchar)
{
    case 'A' :
        printf ("choice A n") ;
    case 'B' :
        printf ("choice B ") ;
    case 'C' :
    case 'D' :
    case 'E' :
    default:
        printf ("No Choice") ;
}
```

Choice A nchoice BNo Choice

### Practical No: 5

Student Name and Roll Number: Saarmeen 22CSU119

Semester /Section: 1st CSE-C

Link to Code:

Date:

Faculty Signature:

Remarks:

▪

### Objective

To familiarize the students with while and do while loop

### Program Outcome

The students will be able to understand iterative control statements and how to apply while and do-while loop in programs.

### Background Study

#### *while loop*

The syntax of the while loop is:

```
while (testExpression)
{
    // statements inside the body of the loop
}
```

*How while loop works?*

- The while loop evaluates the test expression inside the parenthesis ().

- If the test expression is true, statements inside the body of while loop are executed. Then, the test expression is evaluated again.
- The process goes on until the test expression is evaluated to false.
- If the test expression is false, the loop terminates (ends).

### *do...while loop*

The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.

The syntax of the do...while loop is:

```
do
{
    // statements inside the body of the loop
}
while (testExpression);
```

### *How do...while loop works?*

- The body of do-while loop is executed once. Only then, the test expression is evaluated.
- If the test expression is true, the body of the loop is executed again and the test expression is evaluated.
- This process goes on until the test expression becomes false.
- If the test expression is false, the loop ends.

## **LIST OF PROGRAMS**

1. Find factorial of a number using while loop

```
#include<stdio.h>

int main(){

    int x,product=1;
```

```
printf("Enter a number to find factorial\n");  
  
scanf("%d",&x);  
  
while(x>=1){  
  
    product=product*x;  
  
    x--;  
  
}  
  
printf("The factorial is %d",product);  
  
return 0;  
  
}
```

2. Program to add numbers until the user enters zero

```
#include<stdio.h>  
  
int main(){  
    int x,y=0;  
    do{  
        printf("Enter a number\n");  
        scanf("%d",&x);  
        y=y+x;  
    }while(x!=0);  
    printf("The sum is %d",y);  
    return 0;  
}
```

3. Program to print the sum of 10 natural numbers using while loop

```
#include<stdio.h>  
  
int main(){  
  
    int x=1,y=0;  
  
    while(x<=10){  
  
        y=y+x;
```

```
x++;  
  
}  
  
printf("The sum is %d",y);  
  
return 0;  
  
}
```

### Suggested Question Bank

1. Differentiate between while and do while loop.

While loop checks the condition first before reading any further statements whereas, the do while loop reads all the statements first and checks the statement in the end.

2. What will be the output of the C program?

```
a) void main( )  
  
    {  
        int j =1;  
        while (j <= 10 )  
        {  
            printf ( "\n%d",j ) ;  
            j =j + 1 ; } }
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

b) void main( )

```
{  
    float x = 1.1 ;  
    while ( x == 1.1 )  
    {  
        printf ( "\n%f", x ) ;  
        x = x - 0.1 ; } }
```

Won't work as x is a float data type, it takes decimal places upto 6-7 places but condition is just 1.1 so it won't be true and loop wouldn't work.

## **LIST OF FLIP EXPERIMENTS**

### **1. What will be the output of the following?**

a) #include <stdio.h>

```
int main()  
{ int i = 0;  
    while (i = 0)  
        printf("True\n");  
    printf("False\n");  
}
```

False

b) #include<stdio.h>

```
int main()  
{ int i = 0;  
    while(i < 3, i = 0, i < 5)  
    { printf("Loop ");  
      i++; }  
    return 0; }
```

Infinite loop

c) #include<stdio.h>

```
int main()
{ int i = 0;
  while(i++)
  {
    printf("Loop ");
    if(i == 3) break; }
  return 0; }
```

Not work as updation statement given at place of condition

### **Practical No: 6**

Student Name and Roll Number: Saarmeen 22CSU119

Semester /Section: 1st CSE-C

Link to Code:

Date:

Faculty Signature:

Remarks:

▪

### **Objective**

To familiarize the students with for loop

### **Program Outcome**

The students will be able to understand iterative control statements and how to apply for loop in programs.

### **Background Study**

#### **for Loop**

*The syntax of the for loop is:*

for (initializationStatement; testExpression; updateStatement)

{ // statements inside the body of loop }

*How for loop works?*

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.
- This process goes on until the test expression is false. When the test expression is false, the loop terminates.

## LIST OF PROGRAMS

1. Write a program to find the factorial of a number using for loop

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

```
int main(){
```

```
    int x,product=1;
```

```
    printf("Enter a number\n");
```

```
    scanf("%d",&x);
```

```
    for(x;x>=1;x--)
```

```
        product=product*x;
```

```
    printf("The factorial is %d",product);
```

```
    return 0;
```

```
}
```

2. Write a program to print the following pattern:

a)                      1



1 2

1 2 3

1 2 3 4

DoD 1: The number should be given by the user.

DoD 2: The program should check if the number is a positive natural number.

DoD 3: The factorial of that number should be printed as the output.

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

```
int main(){
```

```
    int x,y;
```

```
    for(x=1;x<=4;x++){
```

```
        for(y=1;y<=x;y++)
```

```
            printf("%d",y);
```

```
            printf("\n");
```

```
    }
```

```
    return 0;
```

```
} .
```

b)           55555

              4444

              333

              22

              1

DoD 1: The program does not take any input.

DoD 2: The above mentioned pattern should be printed as the output.

```
#include<stdio.h>

int main(){
    int x,y;
    for(x=5;x>=1;x--){
        for(y=1;y<=x;y++)
            printf("%d",x);
        printf("\n");
    }
    return 0;
}
```

3. Write a program in C to print the table of a given number which is multiplied from 1 to 10 using for loop

```
#include<stdio.h>

int main(){
    int x,y;
    printf("Enter the number\n");
    scanf("%d",&x);
    for(y=1;y<=10;y++)
        printf("%d x %d = %d\n",x,y,x*y);
    return 0;
}
```

## **SUGGESTED QUESTION BANK**

1. To find the sum of individual digits of a positive integer and test given number is palindrome.

```
#include<stdio.h>
#include<math.h>
int main(){
    int e,f,g=0,x,y,z=0,a,b,c,d=0;
    printf("Enter a number\n");
    scanf("%d",&x);
    e=x;
    for(e;e!=0;e=e/10)
    {
        f=e%10;
        g=g+f;
    }
    printf("The sum of digits is %d\n",g);
    b=x;
    c=x;
    for(x;x!=0;x=x/10){
        z++;
    }
    for(b;b!=0;b=b/10){
        y=b%10;
```

```
    z--;  
    a=y*((int)(pow(10,z)));  
    d=d+a;  
}  
if(c==d){  
    printf("The number is palindrome.\n");  
}else{  
  
}  
return 0;  
}
```

2. That will read in a positive integer value and determine whether its prime or Fibonacci.

```
#include<stdio.h>  
  
int main(){  
    int i,x,y,z,a;  
    printf("Enter the number\n");  
    scanf("%d",&x);  
    for(i=2;i<=x/2;i++)  
    {  
        if((x%i)!=0){  
            printf("The numbers is prime.\n");  
        }else  
            break;  
    }
```

```
}  
y=x%10;  
x=x/10;  
z=x%10;  
x=x/10;  
a=x%10;  
if((a+z)==y){  
    printf("The numbers entered are in fibonacci series\n");  
}else{  
}  
return 0;  
}
```

3. WAP to calculate the sum of every third integer, beginning with i=2 ( i.e. calculate the sum 2+5+8+11+.....) for all values of i less than 100.

```
#include<stdio.h>  
  
int main(){  
    int x,y=0;  
    for(x=2;x<100;x=x+3)  
    {  
        y=y+x;  
    }  
    printf("The sum is %d",y);  
}
```

```
return 0;
```

```
}
```

4. WAP to convert a positive integer quantity to a roman numeral (i.e. 12 will be converted to XIV).

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

```
int main(){
```

```
int x;
```

```
printf("Enter a number\n");
```

```
scanf("%d",&x);
```

```
while(x!=0){
```

```
if(x>=1000){
```

```
printf("M");
```

```
x=x-1000;
```

```
}else if(x>=900){
```

```
printf("CM");
```

```
x=x-900;
```

```
}else if(x>=500){
```

```
printf("D");
```

```
x=x-500;
```

```
}else if(x>=400){
```

```
printf("CD");
```

```
x=x-400;
```

```
}else if(x>=100){
```

```
printf("C");  
x=x-100;  
}else if(x>=90){  
    printf("XC");  
    x=x-90;  
}else if(x>=50){  
    printf("L");  
    x=x-50;  
}else if(x>=40){  
    printf("XL");  
    x=x-10;  
}else if(x>=10){  
    printf("X");  
    x=x-10;  
}else if(x>=9){  
    printf("IX");  
    x=x-9;  
}else if(x>=5){  
    printf("V");  
    x=x-5;  
}else if(x>=4){  
    printf("IV");  
    x=x-4;
```

```
}else if(x>=1){  
    printf("I");  
    x--;  
}else;  
}  
return 0;  
}
```

## **LIST OF FLIP EXPERIMENTS**

1. How many times "FOCP" is get printed?

```
#include<stdio.h>  
int main()  
{  
    int x;  
    for(x=-1; x<=10; x++)  
    {  
        if(x < 5)  
            continue;  
        else  
            break;  
        printf("FOCP");  
    }  
    return 0;  
}
```

Output won't be printed as the continue statement in the if statement will take to the else statement, which has a break statement, which will eventually come out of the whole for loop.

2 What will be the output of the program?

```
#include<stdio.h>  
int main()  
{  
    int i=0;  
    for(; i<=5; i++);  
}
```



```
    printf("%d", i);  
    return 0;  
}  
loop won't work properly as semicolon is used after the for statement  
3. Point out the error, if any in the for loop.
```

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

There are no errors as initialization we did before the for loop with i=1. For updation we are using the concept of post-decrement, every time the loop runs, value of i is increased by one. And for condition we are using a break statement in a if statement. Which makes it a complete for loop statement.

### Practical No: 7

Student Name and Roll Number: Saarmeen 22csu119

Semester /Section: 1st CSE-C

Link to Code:

Date: 9-12-22

Faculty Signature:

Remarks:

▪

### Objective

To familiarize the students with functions in C

## Program Outcome

- The students will understand the need of functions, how to declare and define functions in C.
- Students will learn the two methods of function calling – call by value and call by reference.
- Students will be acquainted with the concept of recursion, their declaration and definition.

## Background Study

A function is a set of statements that take inputs, do some specific computation and produces output. The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

*Why do we need functions?*

Functions help us in reducing code redundancy. If functionality is performed at multiple places in software, then rather than writing the same code, again and again, we create a function and call it everywhere. This also helps in maintenance as we have to change at one place if we make future changes to the functionality.

Functions make code modular. Consider a big file having many lines of codes. It becomes really simple to read and use the code if the code is divided into functions. Functions provide abstraction. For example, we can use library functions without worrying about their internal working.

## LIST OF PROGRAMS

1. Write a C Program using functions to check Whether a Number can be Expressed as Sum of Two Prime Numbers  

```
#include<stdio.h>
int check(int);
int main(){
```

```
int x,a,b,c=0;
printf("Enter a number\n");
scanf("%d",&x);
for(a=2;a<=x/2;a++)
{
    b=x-a;
    if((check(a)==0)&&(check(b)==0))
    {
        printf("There exists a relation as %d and %d\n",a,b);
        c++;
    }
}
if(c==0)
    printf("There exists no such relation\n");
return 0;
}

int check(int a){
    int c,d,e=0;
    if(a==2)
        e=2;
    else{
        for(c=2;c<=a/2;c++){
            if(a%c==0)
            {
                e=1;
                break;
            }
        }
        else
            e=2;
    }
}
if(e==2)
    return 0;
else
```

```
        return 1;  
    }
```

2. Write a C Program using functions to convert Binary Number to Decimal and vice-versa

```
#include<stdio.h>  
int dec(int);  
void binary(int);  
int main(){  
    int a,b;  
    printf("1.Convert a number from binary to decimal\n2. Convert a number from decimal  
to binary\n3. Exit\n");  
    do{  
        printf("Choose a option\n");  
        scanf("%d",&a);  
        switch(a){  
            case 1: printf("Enter the binary number\n");  
                    scanf("%d",&b);  
                    if(dec(b)==0);  
                    else  
                        printf("The decimal value of %d is %d\n",b,dec(b));  
                    break;  
            case 2: printf("Enter the decimal number\n");  
                    scanf("%d",&b);  
                    binary(b);  
                    break;  
            case 3: printf("Exiting..\n");  
                    break;  
            default: printf("Try again\n");  
                    break;  
        }  
    }while(a!=3);  
    return 0;  
}
```

```
int dec(int b){
    int a,c,d=1,e=0,f=0;
    a=b;
    while(a!=0){
        c=a%10;
        a=a/10;
        if((c!=0)&&(c!=1))
        {
            f=1;
            break;
        }
        e=e+d*c;
        d=d*2;
    }
    if(f==1)
    {
        printf("Value entered is not correct\n");
        return 0;
    }
    else
        return e;
}

void binary(int b){
    int a[32],i=0;
    while(b!=0){
        a[i]=b%2;
        b=b/2;
        i++;
    }
    for(i=i-1;i>=0;i--)
        printf("%d",a[i]);
    printf("\n");
}
```

3. Write a C Program to Find the Sum of Natural Numbers using Recursion

**SUGGESTED QUESTION BANK**

1. Write four functions “add”, “Sub”, “mul” and “div” and use them to implement a calculator

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

```
int add(int,int);
```

```
int sub(int,int);
```

```
int divi(int,int);
```

```
int mul(int,int);
```

```
int main(){
```

```
    int a,b,c;
```

```
    printf("1. Add two numbers\n2. Subtract two numbers\n3. Multiply two numbers\n4. Divide  
two numbers\n5. Exit\n");
```

```
    do{
```

```
        printf("Choose a option\n");
```

```
        scanf("%d",&a);
```

```
        switch(a){
```

```
            case 1: printf("Enter two numbers\n");
```

```
                scanf("%d%d",&b,&c);
```

```
                printf("Sum = %d\n",add(b,c));
```

```
                break;
```

```
case 2: printf("Enter two numbers\n");

scanf("%d%d",&b,&c);

printf("Difference = %d\n",sub(b,c));

break;

case 3: printf("Enter two numbers\n");

scanf("%d%d",&b,&c);

printf("Product = %d\n",mul(b,c));

break;

case 4: printf("Enter two numbers\n");

scanf("%d%d",&b,&c);

printf("Qoutient = %d\n",divi(b,c));

break;

case 5: printf("Exiting..\n");

break;

default: printf("Try again\n");

break;

}

}while(a!=5);

return 0;

}

int add(int a,int b){
```

```
    a=a+b;

    return a;

}

int sub(int a,int b){

    a=a-b;

    return a;

}

int mul(int a,int b){

    a=a*b;

    return a;

}

int divi(int a,int b){

    a=a/b;

    return a;

}
```

2. Write two functions “area” and “perimeter” that computes area and perimeter of a triangle given the three sides. Write a main program that calls these two functions as per the choice of the user

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

```
#include<math.h>
```

```
float area(float,float,float);
```



```
int per(int,int,int);

int main(){

    int a,b,c,d;

    float e,f,g;

    printf("1. Finding area of triangle\n2. Finding perimeter of a triangle\n3. Exit\n");

    do{

        printf("Choose a option\n");

        scanf("%d",&a);

        switch(a){

            case 1: printf("Enter three sides\n");

                scanf("%f%f%f",&e,&f,&g);

                printf("Area = %f\n",area(e,f,g));

                break;

            case 2: printf("Enter three sides\n");

                scanf("%d%d%d",&b,&c,&d);

                printf("Perimeter = %d\n",per(b,c,d));

                break;

            case 3: printf("Exiting..\n");

                break;

            default: printf("Try again\n");

                break;
```

```
    }  
    }while(a!=3);  
    return 0;  
}  
  
float area(float a,float b,float c){  
    float s,d,e;  
    s=(a+b+c)/2;  
    d=(s*(s-a)*(s-b)*(s-c));  
    e=sqrt(d);  
    return e;  
}  
  
int per(int a,int b,int c){  
    a=a+b+c;  
    return a;  
}
```

3. Write a function “convert” that converts the temperature from Celsius to Fahrenheit and vice versa as per the choice of the user

```
#include<stdio.h>  
  
float far(float);  
float cel(float);
```

```
int main(){
    int a;
    float b;
    printf("1. Convert to Celcius\n2. Convert to Fahrenheit\n3. Exit\n");
    do{
        printf("Choose an option\n");
        scanf("%d",&a);
        switch(a){
            case 1: printf("Enter temperature in Celcius\n");
                    scanf("%f",&b);
                    printf("The temperature in Fahrenheit = %f\n",far(b));
                    break;
            case 2: printf("Enter temperature in Fahrenheit\n");
                    scanf("%f",&b);
                    printf("The temperature in Celcius = %f\n",cel(b));
                    break;
            case 3: printf("Exiting..\n");
                    break;
            default: printf("Try again\n");
                    break;
        }
    } while(a!=3);
    return 0;
}
```

```
float far(float a){  
    float b;  
    b=a*9/5+32;  
    return b;  
}  
  
float cel(float a){  
    float b;  
    b=(a-32)*5/9;  
    return b;  
}
```

## LIST OF FLIP EXPERIMENTS

1. What will be the output of the C program?

```
#include<stdio.h>  
int main()  
{ function();  
  return 0;  
}  
void function()  
{  
  printf("Function in C is awesome");  
}
```

Error function not declared.

2. #include<stdio.h>  
int main()  
{ main();  
return 0;  
}

Program won't work properly, it will exit the code, no functions will be performed.

3. #include<stdio.h>

```
int function();  
main()  
{ int i;  
i = function();  
printf("%d", i);  
return 0;  
}  
function()  
{  
int a;  
a = 250;  
}  
0
```

4.

```
#include<stdio.h>  
int function(int, int);  
int main()  
{  
int a = 25, b = 24 + 1, c;  
printf("%d", function(a, b));  
return 0;  
}  
int function(int x, int y)  
{ return (x - (x == y)); }  
24
```

### Practical No: 8

Student Name and Roll Number:Saarmeen 22csu119

Semester /Section:1/c

Link to Code:

Date:1/1/23

Faculty Signature:

Remarks:

■

## Objective

To familiarize the students with pointers in C

## Program Outcome

The students will be able to understand the concept of pointers and how to use them

## Background Study

Pointers in C language is a variable that stores/points the address of another variable. A Pointer in C is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data type such as int, float, char, double, short etc.

Pointer Syntax : `data_type *var_name;` Example : `int *p;` `char *p;`

Where, \* is used to denote that “p” is pointer variable and not a normal variable.

### *KEY POINTS TO REMEMBER ABOUT POINTERS IN C:*

- Normal variable stores the value whereas pointer variable stores the address of the variable.
- The content of the C pointer always be a whole number i.e. address.
- Always C pointer is initialized to null, i.e. `int *p = null;`
- The value of null pointer is 0.
- & symbol is used to get the address of the variable.
- \* symbol is used to get the value of the variable that the pointer is pointing to.
- If a pointer in C is assigned to NULL, it means it is pointing to nothing.
- Two pointers can be subtracted to know how many elements are available between these two pointers.
- But, Pointer addition, multiplication, division are not allowed.
- The size of any pointer is 2 byte (for 16 bit compiler).

## **LIST OF PROGRAMS**

- C program to create, initialize, assign and access a pointer variable.
- C program to demonstrate example of double pointer (pointer to pointer)
- C program to swap two numbers using pointers
- C program to reverse an array using pointers

## **SUGGESTED QUESTION BANK**

1. Explain null pointers.
2. Why do we use void pointer?
3. What is pointer to pointer? Explain its advantages.

## **LIST OF FLIP QUESTIONS:**

**What will be the output of the following?**

```
1. #include <stdio.h>
void main()
{
    char *s= "hello";
    char *p = s;
    printf("%c\t%c", 1[p], s[1]);
}
```

OUTPUT:- e e

```
2. #include <stdio.h>
int main()
{
    int ary[4] = {1, 2, 3, 4};
    int *p = ary + 3;
    printf("%d\n", p[-2]);
}
```

OUTPUT:- 2

```
3. # include <stdio.h>
```

```
void fun(int *ptr)
{   *ptr = 30; }
int main() {
    int y = 20;
    fun(&y);
    printf("%d", y);
    return 0; }
```

OUTPUT:- 30

```
4. #include<stdio.h>
int main()
{
    int a;
    char *x;
    x = (char *) &a;
    a = 512;
    x[0] = 1;
    x[1] = 2;
    printf("%dn",a);
    return 0;
}
```

OUTPUT:-513

```
5. #include <stdio.h>
int main() {
    int *ptr;
    int x;
    ptr = &x;
    *ptr = 0;
    printf(" x = %dn", x);
    printf(" *ptr = %dn", *ptr);
    *ptr += 5;
    printf(" x = %dn", x);
    printf(" *ptr = %dn", *ptr);
    (*ptr)++;
    printf(" x = %dn", x);
}
```



```
printf(" *ptr = %dn", *ptr);  
return 0; }
```

OUTPUT:- x = 0

\*ptr = 0

x = 5

\*ptr = 5

x = 6

\*ptr = 6

### **Practical No: 9**

Student Name and Roll Number:Saarmeen 22CSU119

Semester /Section:1st/C

Link to Code:

Date:

Faculty Signature:

Remarks:

.

### **Objective**

To familiarize the students with arrays in C

### **Program Outcome**

Through this lecture, students will understand the concept of arrays, types of arrays, application in real life. They will learn how to declare arrays in C

### **Background Study**

An array in C or C++ is a collection of items stored at contiguous memory locations and elements can be accessed randomly using indices of an array. They are used to store similar type of elements as in the data type must be the same for all elements. They can be used to store collection of primitive data types such as int, float, double, char, etc of any particular type. To add to it, an array in C or C++ can store derived data types such as the structures, pointers etc. Given below is the picturesque representation of an array.

40	55	63	17	22	68	89	97	89
0	1	2	3	4	5	6	7	8

<- Array Indices

**Array Length = 9**

**First Index = 0**

**Last Index = 8**

### Why do we need arrays?

We can use normal variables (v1, v2, v3, ..) when we have a small number of objects, but if we want to store a large number of instances, it becomes difficult to manage them with normal variables. The idea of an array is to represent many instances in one variable.

### LIST OF PROGRAMS

1. Write a C program to find the largest number in an array.
  - a) DoD 1: The size of the array should be given by the user. It has to be 1-D array.
  - b) DoD 2: The elements of the array should also be given by the user.
  - c) DoD 3: The largest element of the array should be printed as an output.

2. Write a C program to transpose the matrix.

DoD 1: The order of the matrix should be provided by the user.

DoD 2 : The elements of the matrix should also be provided by the user.

DoD 3: The original matrix should be displayed.

DoD 4: The transposed matrix should then be displayed.

### **SUGGESTED QUESTION BANK**

**Write a C program to perform the following:**

1. To insert an element in the array at specified position.
2. C Program to Put Even & Odd Elements of an Array in 2 Separate Arrays .
3. Merge the elements of two sorted arrays.

### **LIST OF FLIP QUESTIONS:**

Q1. What is the right way to Initialize array?

A. `int num[6] = { 2, 4, 12, 5, 45, 5 };//A`

B. `int n{} = { 2, 4, 12, 5, 45, 5 };`

C. `int n{6} = { 2, 4, 12 };`

D. `int n(6) = { 2, 4, 12, 5, 45, 5 };`

Q2. What will be the output of the program ?

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

```
void main()
```

```
{
```

```
    int a[5] = {5, 1, 15, 20, 25};
```

```
    int i, j, m;
```

```
i = ++a[1];  
j = a[1]++;  
m = a[i++];  
printf("%d, %d, %d", i, j, m);  
}
```

A. 3, 2, 15//A

B. 2, 3, 20

C. 2, 1, 15

D. 1, 2, 5

Q3. Consider the following type definition.

```
typedef char x[10];
```

```
x myArray[5];
```

What will sizeof(myArray) be ? (Assume one character occupies 1 byte)

A. 15

B. 10

C. 50

D. 30

E. None of these//E

Q4. What will be printed after execution of the following code?

```
void main()
{
    int arr[10] = {1,2,3,4,5};
    printf("%d", arr[5]);
}
```

- A. Garbage Value
- B. 5
- C. 6
- D. 0//D
- E. None of these

Q5. Which of the following statements are correct about the program below?

```
#include<stdio.h>

void main()
{
    int size, i;
    scanf("%d", &size);
    int arr[size];
    for(i=1; i<=size; i++)
    {
        scanf("%d", arr[i]);
    }
}
```

```
printf("%d", arr[i]);  
  
}  
  
}
```

### Practical No: 10

Student Name and Roll Number: Saarmeen 22CSU119

Semester /Section:1st/C

Link to Code:

Date:

Faculty Signature:

Remarks:

.

### Objective

To familiarize the students with strings in C

### Program Outcome

Through this lecture, students will understand the concept of strings and how to perform operations with and without using library functions.

### Background Study

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.

*Declaration of strings:* Declaring a string is as simple as declaring a one dimensional array.

*Syntax for declaring a string:* char str\_name[size];

In the above syntax `str_name` is any name given to the string variable and `size` is used to define the length of the string, i.e the number of characters strings will store. there is an extra terminating character which is the Null character (`'\0'`) used to indicate termination of string which differs strings from normal character arrays.

*Initializing a String:* A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as `str` and initialize it with "CProgramming".

1. `char str[] = " CProgramming";`
2. `char str[50] = " CProgramming";`

### **LIST OF PROGRAMS**

1. Write a C program to check whether a string is a palindrome or not.
2. Write a C program to convert string in lowercase to uppercase.
3. Write a C program to copy one string in another without using library functions.

### **SUGGESTED QUESTION BANK**

Write a C program to perform the following:

1. To convert each character including blank spaces to its ASCII equivalent.
2. To generate a positive random integer. Add this integer to ASCII equivalent
3. To delete a specified character from a string  
Eg. Input : PROGRAMMING, A    Output : PROGRMMING
4. To replace a specified character with another character in a string  
Eg. Input : SERIES, E, C    Output : SCRICS

### **LIST OF FLIP EXPERIMENTS**

**Find the error(if any ) and write the output of the following:**

- 1) `main( )`

```
{  
  
char c[2] = "A" ;  
  
printf ( "\n%c", c[0] ) ;  
  
printf ( "\n%s", c ) ; }
```

no error

2) main( )

```
{  
  
char s[ ] = "Get organised! learn C!!" ;  
  
printf ( "\n%s", &s[2] ) ;  
  
printf ( "\n%s", s ) ;  
  
printf ( "\n%s", &s ) ;  
  
printf ( "\n%c", s[2] ) ; }
```

no error

3. main( )

```
{  
  
printf ( 5 + "Good Morning " ) ;  
  
}
```

4. main( )



```
{ char str1[ ] = { 'H', 'e', 'l', 'l', 'o' } ;  
  
char str2[ ] = "Hello" ;  
  
printf ( "\n%s", str1 ) ;  
  
printf ( "\n%s", str2 ) ; }
```

5. main( )

```
{  
  
    printf ( "\n%d%d", sizeof ( '3' ), sizeof ( "3" ), sizeof ( 3 ) ) ;  
  
}
```

6. A string is terminated by a \_\_null\_\_ character, which is written as \_\_'\0'\_\_.

7. The array char name[10] can consist of a maximum of \_\_10\_\_ characters.

8. If the string "Alice in wonder land" is fed to the following scanf( ) statement, what will be the contents of the arrays str1, str2, str3 and str4? scanf ( "%s%s%s%s%s", str1, str2, str3, str4 ) ;

### Practical No: 11

Student Name and Roll Number:Saarmeen 22csu119

Semester /Section:1st/ C

Link to Code:

Date:

Faculty Signature:

Remarks:

■

## Objective

To familiarize the students with strings in C

## Program Outcome

Through this lecture, students will understand the concept of strings and how to perform operations with and without using library functions.

## Background Study

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.

*Declaration of strings:* Declaring a string is as simple as declaring a one dimensional array.

*Syntax for declaring a string:* `char str_name[size];`

In the above syntax `str_name` is any name given to the string variable and `size` is used to define the length of the string, i.e the number of characters strings will store. there is an extra terminating character which is the Null character ('\0') used to indicate termination of string which differs strings from normal character arrays.

*Initializing a String:* A string can be initialized in different ways. We will explain this with the help of an example. Below is an example to declare a string with name as `str` and initialize it with "CProgramming".

1. `char str[] = " CProgramming";`
2. `char str[50] = " CProgramming";`

## LIST OF PROGRAMS

4. Write a C program to check whether a string is a palindrome or not.
5. Write a C program to convert string in lowercase to uppercase.
6. Write a C program to copy one string in another without using library functions.

## **SUGGESTED QUESTION BANK**

1. Fill in the blanks:

A string is terminated by a \_\_\_\_\_ character, which is written as \_\_\_\_\_.

2. The array `char name[10]` can consist of a maximum of \_\_\_\_\_ characters.

3. If the string "Alice in wonder land" is fed to the following `scanf( )` statement, what will be the contents of the arrays `str1`, `str2`, `str3` and `str4`? `scanf ( "%s%s%s%s%s", str1, str2, str3, str4 ) ;`

## **LIST OF FLIP EXPERIMENTS**

**Find the error(if any ) and write the output of the following:**

1) `Void main( )`

```
{  
char c[2] = "A" ;  
printf ( "\n%c", c[0] ) ;  
printf ( "\n%s", c ) ; }
```

2) `Void main( )`

```
{  
char s[ ] = "Get organised! learn C!!" ;  
printf ( "\n%s", &s[2] ) ;  
printf ( "\n%s", s ) ;  
printf ( "\n%s", &s ) ;
```

```
printf ( "\n%c", s[2] ) ; }
```

3. void main( )

```
{  
    printf ( 5 + "Good Morning " ) ;  
}
```

4. void main( )

```
{ char str1[ ] = { 'H', 'e', 'l', 'l', 'o' } ;  
    char str2[ ] = "Hello" ;  
    printf ( "\n%s", str1 ) ;  
    printf ( "\n%s", str2 ) ; }
```

5. void main( )

```
{  
    printf ( "\n%d%d", sizeof ( '3' ), sizeof ( "3" ), sizeof ( 3 ) ) ;  
}
```

### Practical No: 12

Student Name and Roll Number:Saarmeen 22csu119

Semester /Section:1st/c

Link to Code:

Date:

Faculty Signature:

Remarks:

■

## Objective

To familiarize the students with structures in C

## Program Outcome

Through this lecture, students will understand the concept of structures and how to implement them to maintain the records.

## Background Study

Structure is another user defined data type available in C that allows to combine data items of different kinds. To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows –

```
struct [structure tag] {  
    member definition;  
    member definition;  
    ...  
    member definition;  
} [one or more structure variables];
```

The structure tag is optional and each member definition is a normal variable definition, such as `int i;` or `float f;` or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional.

## LIST OF PROGRAMS

1. Write a program to increase the salary depending on the number of hours of work per day as follows:

Hours of work per day	8	10	$\geq 12$
Increase in salary	\$50	\$100	\$150

DODs

- a) Write a structure to store the names, salary and hours of work per day of 10 employees in a company.
  - b) Print the name of all the employees along with their final salaries.
2. Write a program to create a menu for a library in which the following can be done.
    - 1 - Display book information
    - 2 - Add a new book
    - 3 - Display all the books in the library of a particular author
    - 4 - Display the number of books of a particular title
    - 5 - Display the total number of books in the library
    - 6 - Issue a book

DODs

- a) Create a structure containing book information like accession number, name of author, book title and flag to know whether book is issued or not.
  - b) If we issue a book, then its number gets decreased by 1 and if we add a book, its number gets increased by 1.
3. Write a program to read two complex numbers and perform addition, subtraction of these two complex numbers.
    - a) Define a structure “complex”.
    - b) Display the result

### **SUGGESTED QUESTION BANK**

1. Explain the difference between array and structures.
2. Explain the concept of union.
3. How can we create array of structure variables?

### **LIST OF FLIP EXPERIMENTS**

1. `#include<stdio.h>`

```
int main()
```

```
{
```

```
struct site { char name[] = "GeeksQuiz";
```

```
int no_of_pages = 200; };
```

```
struct site *ptr;
```

```
printf("%d ", ptr->no_of_pages);
```

```
printf("%s", ptr->name);
```

```
getchar();
```

```
return 0; }
```

2. Assume that size of an integer is 32 bit. What is the output of following program?

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

```
struct st {
```

```
int x;
```

```
static int y; };  
  
int main()  
{ printf("%d", sizeof(struct st));  
  return 0;}
```

3. main()

```
{  
struct employee  
{  
char name[25];  
int age;  
float bs;  
}  
struct employee e ;  
e.name = "Hacker" ;  
e.age = 25 ;  
printf ("%s %d", e.name, e.age); }
```

4. main()

```
{ struct {  
char name[25];  
char language[10].  
} a ;
```



```
static struct a = {"Hacker", "C"};  
printf ("%s %s", a.name, a.language);}
```

### Practical No: 13

Student Name and Roll Number:Saarmeen 22csu119

Semester /Section:1st/c

Link to Code:

Date:

Faculty Signature:

Remarks:

.

### Objective

To familiarize the students with file and its operations in C

### Program Outcome

Through this lecture, students will understand the file management in C

### Background Study

A File can be used to store a large volume of persistent data. Like many other languages 'C' provides following file management functions, Creation of a file, Opening a file, Reading a file, Writing to a file, and Closing a file.

### LIST OF PROGRAMS

1. C Program to read contents of file using fscanf(), fgetc() and fgets().

2. C Program to write contents of file using `fprintf()`, `fputs()` and `fputc()`.
3. C Program to Find the Number of Lines in a text file.
4. Program in C to write multiple lines in a text file

### **SUGGESTED QUESTION BANK**

1. Differentiate between `getc` and `getchar`.
2. How does an append mode is different from a write mode?
3. What is the significance of EOF?
4. Why do we use `rewind`?
5. Write the general syntax of `fseek`.

### **LIST OF FLIP EXPERIMENTS**

1. `#include<stdio.h>`

```
int main()
{
    int EOF = 0;
    printf("%d", EOF);
    return 0;
}
```

**A. -1   B. 0   C. 1   D. Compilation Error//D**

2. Which of the following true about `FILE *fp`

**A. FILE is a keyword in C for representing files and fp is a variable of FILE type.**

**B. FILE is a structure and fp is a pointer to the structure of FILE type//B**

**C. FILE is a stream**

**D. FILE is a buffered stream**

3. `getc()` returns EOF when

- A. End of files is reached**
- B. When `getc()` fails to read a character**
- C. Both of the above//C**
- D. None of the above**

4. In `fopen()`, the open mode "wx" is sometimes preferred "w" because. 1) Use of wx is more efficient. 2) If w is used, old contents of file are erased and a new empty file is created. When wx is used, `fopen()` returns NULL if file already exists.

- A. Only 1   B. Only 2//B   C. Both 1 and 2   D. Neither 1 nor 2**

5. `fseek()` should be preferred over `rewind()` mainly because

- A. `rewind()` doesn't work for empty files**
- B. `rewind()` may fail for large files**
- C. In `rewind`, there is no way to check if the operations completed successfully//C**
- D. All of the above**

6. `#include<stdio.h>`

```
int main()
{ printf("%d", EOF);
  return 0; }
```

- A. NULL   B. -1//B   C. 0   D. Compilation Error**

7. If `fopen()` functions is not able to open a file, it will returns?

- A. Compilation Error   B. 0   C. EOF   D. NULL//D**

8. What will be the output of the C program?

**/\*This program uses two files\*/**

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

```
int main()
```

```
{ char ch;
```

```
FILE *fptr1, *fptr2;
```

```
fp = fopen("datafile1.txt", "w");
```

```
fp = fopen("datafile2.txt", "w");
```

```
printf("Thank you.");
```

```
fclose(*fptr1, *fptr2);
```

```
return 0; }
```

**A. Thank you    B. Prints nothing    C. Compilation error //D    D. None of the above**

9. fopen() function will not opens \_\_\_\_\_

**A. bin files    B. c files    C. txt files    D. None of the above//D**

10. In C, when getc() function returns EOF?

**A. Never Returns    B. End of files is reached    C. When getc() fails to read a character  
D. Both B and C//D**

**Experiment No: (Mini Project)**

Student Name and Roll Number:Saarmeen 22csu119

Semester /Section:1st/C

Link to Code:

Date:

Faculty Signature:

Remarks:

**Project Title: Snake Game**

**Description of Project:**

**Problem Statement:**

**Problem Analysis:**

**Program Design:**

**Programming Requirements:**

## Data/Input Output Description:

## Pseudocode

## Algorithm

## Flowchart

## Program Implementation and Testing (module wise)

## Output (Screenshots)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int width=20,height=20,gameOver;
int x,y,fruitX,fruitY,score,flag,countTail=0;
int tailX[100],tailY[100];
void setup()
{
    gameOver=0;
    x=width/2;
    y=height/2;

    label1:
    fruitX=rand()%20;
    if(fruitX==0)
        goto label1;

    label2:
    fruitY=rand()%20;
    if (fruitY==0)
        goto label2;
}
//code to set up a box
void draw()
{
    system("cls");
    int i,j;
    system ("cls"); //Since draw function is in an infinite loop, we have to clear the previous screen before the draw function is called the 2nd time
    for(i=0;i<width;i++){
        for(j=0;j<height;j++){
            if (i==0||i==height-1||j==0||j==width-1)
            {
                printf("* ");
            }
            else
            {
                if(i==x && j==y)
                    printf("O ");
                else if(i==fruitX && j==fruitY)
                    printf("F ");
                else{
                    int m=0,k;
                    for(k=0;k<countTail;k++){
                        if(i==tailX[k] && j==tailY[k]){
                            printf("o ");
                            m=1;
                        }
                    }
                    if(m==0)
                        printf(" ");
                }
            }
        }
    }
}
```

```

        if(m==0)
            printf(" ");
    }

}

printf("\n");
}
printf("SCORE=%d\n",score);
}
//code to take the input of the keyboard by 'kbhit' and to read the value entered by the keyboard by 'getch'
void input ()
{
    if(kbhit())
    {
        switch(getch())
        {
            case 'a': //wasd
                flag=1;
                break;
            case 'd':
                flag=2;
                break;
            case 'w':
                flag=3;
                break;
            case 's':
                flag=4;
                break;
            case 'x':
                gameOver=1;
                break;
        }
    }
}
//code to make the snake run along the x and y axis
void Logic()
{
    int prevX=tailX[0], prevY=tailY[0], prev2X, prev2Y, i;
    tailX[0]=x;
    tailY[0]=y;
    for(i=1;i<countTail;i++){
        prev2X=tailX[i];
        prev2Y=tailY[i];
        tailX[i]=prevX;
        tailY[i]=prevY;
        prevX=prev2X;
        prevY=prev2Y;
    }
}

```

```

        prevY=prev2Y;
    }
    switch(flag)
    {
        case 1:
            y--;
            break;
        case 2:
            y++;
            break;
        case 3:
            x--;
            break;
        case 4:
            x++;
            break;
        default:
            break;
    }
    if(x<0||x>width-1||y<0||y>height-1)
        gameOver=1;
    for(i=0;i<countTail;i++){
        if(x==tailX[i] && y==tailY[i])
            gameOver=1;
    }
    if(x==fruitX && y==fruitY){
        label3:
        fruitX=rand()%20;
        if(fruitX==0)
            goto label3;

        label4:
        fruitY=rand()%20;
        if (fruitY==0)
            goto label4;
        score+=10;
        countTail++;
    }
}

int main()
{
    char a;
    int m,n;
label5:
    setup();
    while(!gameOver) //this infinite loop has been used to be able to take the input multiple times and run the logic multiple times
    {

int main()
{
    char a;
    int m,n;
label5:
    setup();
    while(!gameOver) //this infinite loop has been used to be able to take the input multiple times and run the logic multiple times
    {
        draw();
        input();
        Logic();
        for(m=0;m<1000;m++)
            for(n=0;n<1000;n++);
    }
    printf("\nPress Y to continue again:");
    scanf("%c",&a);
    if(a=='y'||a=='Y'){
        goto label5;
    }
    else
        return 0;
}

```





