A logo of a university

Description automatically generated

**FOCP-1 Lab Manual**

**CSL 106**

**Faculty Incharges**

Dr. Vandana Khanna

Dr. Pankaj Rakheja

Ms. Jyoti Yadav

Ms. Snehlata

Ms. Bhavana Sharma

Dr. Tamalika Chaira

Ms. Priya Arora

Dr. Anvesha Katti

A logo of a university

Description automatically generated

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2023-24

*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 The NorthCap 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**

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

**SYLLABUS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Department:CSE** | | **Department of CSE** | | |
| 1. **Course Name:**   Fundamentals of computer Programming-1 | | 1. **Course Code** | 1. **L- P** | 1. **Credits** |
| **Code: CSL106** | 2-4 | 4 |
| 1. **Type of Course (Check one):** | | Programme Core Programme Elective Open Elective  **✓** | | |
| √   1. **Frequency of offering (check one):** Odd Even Either Sem. Every Sem. | | | | |
| 1. **Brief Syllabus:**   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. | | | | |
| 1. **Total lecture and Practical Hours for this course**   **100 Hours**  The class size is maximum 30 learners. | | | | |
| 1. **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 | | | | |
| **CO 1** | Understand the working of a computer system with its all peripherals, and master the conversion from one base of number system to another base | | | |
| **CO 2** | Analyze the problem and apply logic to design solutions to problems | | | |
| **CO 3** | 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. | | | |
| **CO 4** | Address the concepts of arrays, pointers, and different storage classes in C | | | |
| **CO 5** | Apply concepts of pre-processor directives and files to develop management system and using functions of graphics to design different shapes and interface. | | | |
| 1. **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, | | | | |
| **Unit IV: Functions, Pointers, Arrays, Strings in C Hours: 21**  Function definition and declaration in C, Call by value and call by reference, Recursion with implementation. Pointers in C, Arrays and Strings. | | | | |
| **Unit V: Structures, Storage classes, File management in C**  **Hours: 16**  Structures in C, Storage classes with their importance, Using pointers in Structures with implementation, File Management in C, File operations | | | | |
| 1. **Guided Project (No. of Hours): 9**   **Unguided Project (No. of Hours): 9** | | | | |
| 1. **Brief Description of Self-learning component by students (through books/resource material etc.): Topics:** Translators (Compiler, Interpreter, Assembler, Linker and Loader) and current technology in real scenarios (case study). | | | | |
| 1. **Books Recommended:**   **Text Books**:  • Yashwant Karnetkar, "Let Us C", BPB Publications, 16th Edition  • Byron S. Gottfried, "Programming with C", McGraw-Hill, 4th Edition  **Reference Books**:  • J.B. Dixit, “Fundamentals of Computers & Programming in C”, Laxmi Publications, 2nd Edition  • Yashwant Karnetkar, “Test your C Skills”, BPB Publications, 5th Edition  **Reference websites:**  <https://www.tutorialspoint.com/cprogramming/>  <https://nptel.ac.in/courses/106104128/> | | | | |

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

1. **LAB REQUIREMENTS**

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

1. **GENERAL INSTRUCTIONS** 
   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.
   1. **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**.**
  1. **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

1. **LIST OF EXPERIMENTS**

|  |  |
| --- | --- |
| **S.No.** | **Experiments** |
| 1 | Algorithms and Flowcharts |
| 2 | Operators and Expressions |
| 3 | If-else |
| 4 | Switch-case |
| 5 | While and do-while loop |
| 6 | For loop |
| 7 | Functions |
| 8 | Pointers |
| 9 | Arrays |
| 10 | Strings |
| 11 | Structures |
| 12 | File Management |

1. **LIST OF PROJECTS**
2. Library Management System
3. Bank Management System
4. Inventory Management System
5. Flight Ticket booking system
6. Bus Reservation system
7. Garments Management System
8. College Record management system
9. Telephone Directory record management system
10. Blood Bank Management system
11. School Fee Management System
12. School Bus Transport System
13. QuizGame
14. **RUBRICS**

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE OF COURSE** | **PARTICULAR** | **ALLOTTED RANGE OF MARKS** | **PASS CRITERIA** |
| Theory+ Practical  (L-T-P/L-0-P) | Minor Test | 15% | Must Secure 30% Marks Out of Combined Marks of Major Test Plus Minor Test with Overall 40% Marks in Total. |
| Major Test | 35% |
| Continuous Evaluation Through Class Tests/Practice/Assignments/Presentation/Quiz | 10% |
| Online Quiz | 5% |
| Regular lab performance | 25% |
| End term lab evaluation | 10% |

**Annexure 1**

**FOCP-I**

**(CSL 106)**

Lab Practical Report

A logo of a university

Description automatically generated

Faculty name Student name

Roll No.:

Semester:

Group:

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2023-24

**INDEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Experiment** | **Page No.** | **Date of Experiment** | **Date of Submission** | **Marks** | **CO Covered** | **Signature** |
| 1 | Algorithms and flowcharts | 14 | 04-09-2023 |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Practical No. 1**

Student Name and Roll Number :Aditi Goyal

Semester /Section:1st /A

Link to Code: CSL 106

Date:04-09-2023

Faculty Signature:

Remarks:

**Objective**

To familiarize the students about Algorithms and Flowcharts

**Program Outcome**

* The students will be able to write algorithms and draw flowcharts for problem statements.

**Problem Statements**

1. Write algorithm and draw flowchart to check whether a number is prime or not.
2. Write algorithm and draw flowchart to find the Fibonacci series till the term less than 1000.

**Background Study:**

## *Algorithm:*

An algorithm is a procedure or set of rules that defines how a program is to be executed. Or we can say that it is a set of instructions for solving a well-defined computational problem.

## *Flowchart:*

A flowchart is a graphical representation of the steps a program takes to process data. In this, we can use several geometric patterns to illustrate the numerous actions the program carries out.

With the help of the flowchart, the designer can efficiently segregate the various elements of the process. Also, it facilitates the analysis by giving step-by-step instructions on the problem.

**Suggested Question Bank**

1. Write algorithm, flowchart and Pseudocode for the following:

1. Add all even numbers up to 100.

Step 1: Start

Step 2: Declare variables I, and sum

Step 3: Initialize variables

Sum – 0

I – 1

Step 4: Read variables

Step 5: Repeat the steps until I = 100

Step 6: If I%2 = = 0

Display sum – sum + I

I = I +1

Step 7: Display sum of even numbers, sum

Step 8: Stop

Initialize sum=0,n=1

false

Print sum

If n<101



If n%2==0

false



true

Sum=sum+n



n=n+1

2. Display all prime numbers between 1 to N.

Step 1: Start

Step 2: Declare variables b, x and flag

Step 3: Read variables b and x

Step 4: Initialize variables

X – 2

flag – 0

Step 5: Repeat the steps until x = b-1

Step 6: If

B%2 = =0

flag = 1

step 7: If flag = 0

Display x

Step 8: Stop

Read variables



Initialize i=1,b=2



If i==n



false

false

b=b+1

If i%b==0

true

Print i

i=i+1



1. Print all multiples of 5 between 1 and 100 (including both 1 and 100).

Step 1: Start

Start 2: Declare variables n and d

Step 3: Initialize

n – 5

d – 1

Step 4: Read variables

Step 5: Repeat steps until d =100

Step 6: if n%5 = = 0

d – d+1

display n

Step 7: Stop

i=1

false

If i<101

true

if i%5==0

false

true

Print i

I=i+1

1. Which of the following symbols is used to start a flowchart?
2. Oval 
3. Terminal
4. Connect
5. Rhombus
6. Which of the following boxes is used for displaying an output statement?
7. Rectangular box
8. Rhombus box
9. Parallelogram box
10. Oval box
11. Write the names and the symbols of the boxes for the given statements

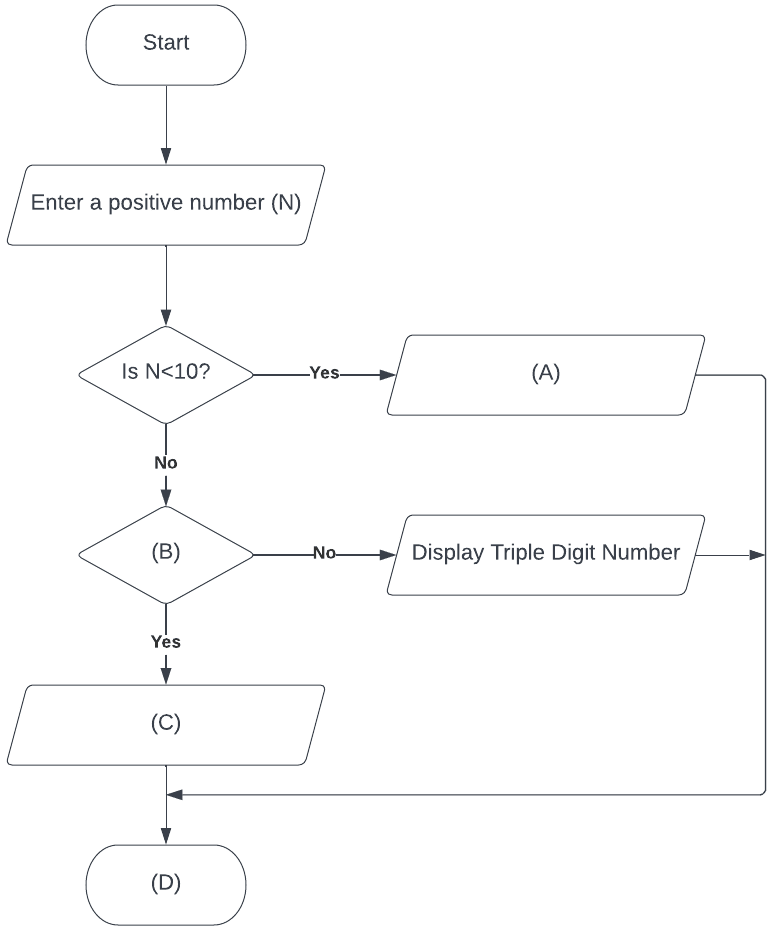
| **Statement** | **Name of the Box** | **Symbol** |
| --- | --- | --- |
| To assign a number | Rectangle |  |
| To display the result | Parallelogram |  |
| To start the process | Oval |  |
| To decide if a number is smaller than the other | Rhombus |  |

1. Anuj, a class 9 student, has designed the following flowchart to input a positive number and check and display whether the number is a single digit number, a double digit number or a triple digit number. To assess his brother who is in class 8, he hides four statements in the flowchart.

Based on the above case and the given flowchart, answer the following questions:

(a) Which of the following statements will be filled in place of (A)?

1. Display Single digit number
2. Display Double digit number
3. Display Negative number
4. Display odd number

(b) Identify (B) from the following:

1. N >= 100
2. N <= 100
3. N < 100
4. N > 100

(c) Which of the following statements will be filled in place of (C)?

1. Display Single digit number
2. Display Double digit number
3. Display Negative number
4. Display odd number

(d) Identify (D) from the following:

1. Start
2. End
3. Finish
4. Stop

C Programs

1. Hello world program in C

#include <stdio.h>

int main()

{

printf("hello world!");

return 0;

}



2. C program to print an integer

#include <stdio.h>

int main()

{

int x;

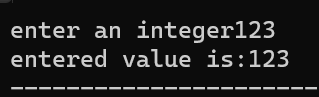
printf("enter an integer");

scanf("%d",&x);

printf("entered value is:%d",x);

return 0;

}



3. C program to add two integers

#include <stdio.h>

int main()

{

int i;

printf("enter 1st no");

scanf("%d",&i);

int n;

printf("enter 2nd no");

scanf("%d",&n);

int sum;

sum=i+n;

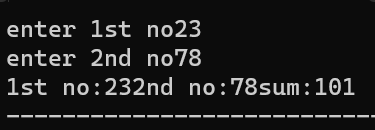
printf("1st no:%d",i);

printf("2nd no:%d",n);

printf("sum:%d",sum);

return 0;

}



4. Input 2 numbers from user and print their sum

#include <stdio.h>

int main()

{

int a,b,sum;

a=10;

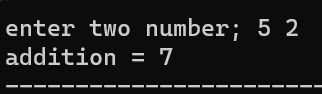
b=20;

sum=a+b;

printf("a+b=%d",sum);

return 0;

}



5. Program to take input of various data types in C

#include <stdio.h>

int main()

{

int i;

printf("enter roll no");

scanf("%d",&i);

float n;

printf("enter marks");

scanf("%f",&n);

char a;

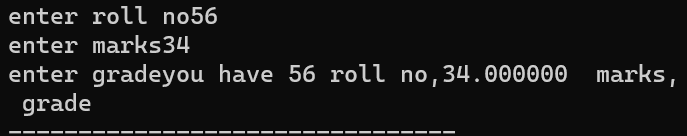
printf("enter grade");

scanf("%c",&a);

printf("you have %d roll no,%f marks,%c grade",i,n,a);

return 0;

}



6. Compute simple interest

#include <stdio.h>

int main()

{

float p,r,t,si;

printf("enter principle\n");

scanf("%f",&p);

printf("enter rate of interest\n");

scanf("%f",&r);

printf("enter time\n");

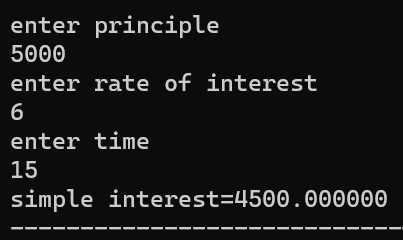
scanf("%f",&t);

si=(p\*r\*t)/100;

printf("simple interest=%f",si);

return 0;

}



7. Compute the perimeter and area of rectangle

#include <stdio.h>

int main()

{

int l,b,area,perimeter;

printf("enter sides:\n");

scanf("%d",&l);

scanf("%d",&b);

area=l\*b;

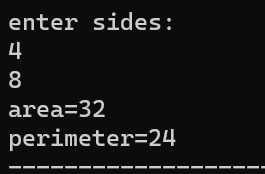
perimeter=2\*(l+b);

printf("area=%d\n",area);

printf("perimeter=%d",perimeter);

return 0;

}



8. Program to compute quotient and remainder

#include <stdio.h>

int main()

{

int dividend,divisor,remainder,quotient;

printf("enter dividend: ");

scanf("%d",&dividend);

printf("enter divisor: ");

scanf("%d",&divisor);

quotient=dividend/divisor;

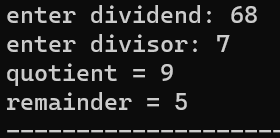
remainder=dividend % divisor;

printf("quotient = %d\n",quotient);

printf("remainder = %d",remainder);

return 0;

}



9. Program to find the size of all data types

#include <stdio.h>

int main()

{

int a;

char b;

float c;

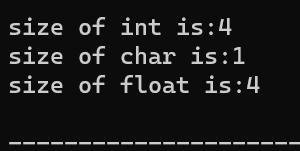
printf("size of int is:%d\n",sizeof (a));

printf("size of char is:%d\n",sizeof (b));

printf("size of float is:%d\n",sizeof (c));

return 0;

}



10. C program to add two float integer numbers

#include <stdio.h>

int main()

{

float a,b,sum;

printf("enter two floating numbers ");

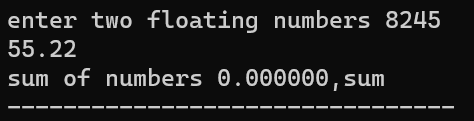
scanf("%f%f", &a, &b);

sum=a+b;

printf("sum of numbers %f,sum");

return 0;

}



11. Program to print ASCII value of a character

#include <stdio.h>

int main()

{

char c;

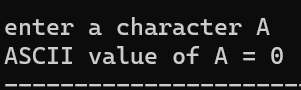
printf("enter a character ");

scanf("%c",&c);

printf("ASCII value of %c = %d ",c);

return 0;

}



12. Compute the perimeter and area of a circle

#include <stdio.h>

int main()

{

int r;

float area , perimeter;

printf("enter the radius of circle\n");

scanf("%d",&r);

printf("the area of circle\n");

printf("the perimeter of circle\n ");

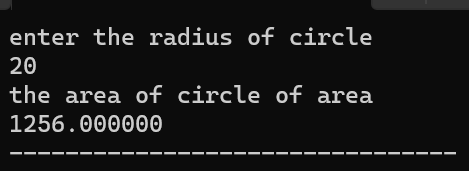
area=3.14\*r\*r;

perimeter=2\*3.14\*r;

printf("%f",area);

printf("%f",perimeter);

}



13. Program to calculate the volume of a sphere

#include <stdio.h>

int main()

{

int volume ,radius;

printf("enter radius of sphere");

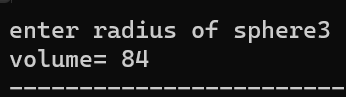
scanf("%d",&radius);

volume=4/3\*3.14\*radius\*radius\*radius;

printf("volume= %d",volume);

return 0;

}



14. C Program to multiply 2 integer ,floating numbers

#include <stdio.h>

int main()

{

int a,b,c;

printf("enter 1st number\n");

scanf("%d",&a);

printf("enter 2nd number\n");

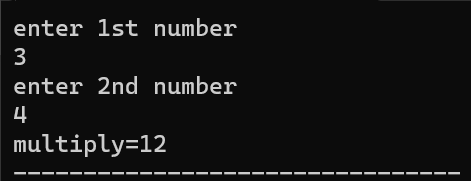
scanf("%d",&b);

c=a\*b;

printf("multiply=%d",c);

return 0;

}



15. C Programs to check whether the number is positive or negative

#include <stdio.h>

int main()

{

int no;

printf("enter any number\n");

scanf("%d",&no);

if(no>0)

{

printf("number is positive");

}

if(no<0)

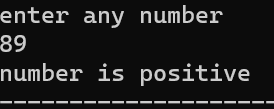
{

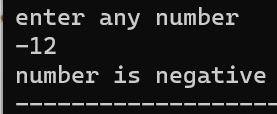
printf("number is negative");

}

return 0;

}





**practical No. 2**

Student Name and Roll Number:Aditi Goyal 23csu011

Semester /Section:1st /A

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 decisionmaking 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 x;

scanf("%f",&x);

printf("entered value is:%f",x);

int a;

a=x;

int n;

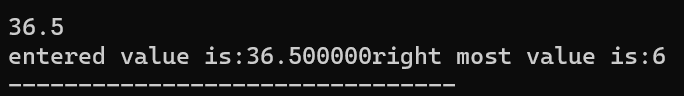
n=a%10;

printf("right most value is:%d",n);

return 0;

}

*Output (with Screenshots):*



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>

int main()

{

int i;

scanf("%d",&i);

int n;

n=i%1000;

printf("last 3 values are:%d\n",n);

int m;

m=n%100;

printf("last 2 values are:%d\n",m);

int l;

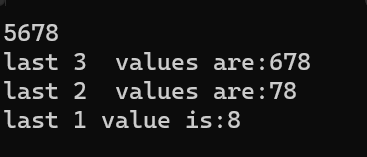
l=m%10;

printf("last 1 value is:%d\n",l);

return 0;

}

**Output: (with Screenshots)**

****

**Suggested Question Bank**

1. A C program contains the following declarations:

int i,j;

long ix ;

short s;

float x;

double dx;

char c;

printf( "%d %d %f %f", k, l, a, b ) ;.

(a) i+ c

(b) x + c (f) i x + j

(c) dx + x (g) s + c

(d) (( int ) dx) + ix

(e) i+ x

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;

}

**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>

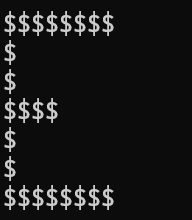
int main()

{

printf("$$$$$$$$\n$\n$\n$$$$\n$\n$\n$$$$$$$$");

return 0;

}



* + - 1. Write a program in C to convert 1000 meter per hour to miles per hour

#include <stdio.h>

int main()

{

int n;

scanf("%d",&n);

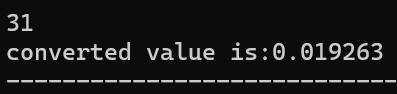
float a;

a=n/1609.344;

printf("converted value is:%f",a);

return 0;

}



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

#include <stdio.h>

int main()

{

int n;

scanf("%d",&n);

int hrs;

hrs=n/3600;

n=n%3600;

printf("%d\n",hrs);

int min;

min=n/60;

printf("%d\n",min);

n=n%60;

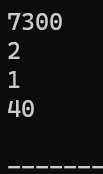
int sec;

sec=n;

printf("%d\n",sec);

return 0;

}



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

**Practical No: 3**

Student Name and Roll Number:Aditi goyal 23csu011

Semester /Section:1st/A

Link to Code:

Date:

Faculty Signature:

Remarks:

**Objective**

To familiarize the students with if-else statements.

**Program Outcome**

The students will learn the concept of checking the conditions 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 quantity;

float price;

printf("enter quantity and price:");

scanf("%f,d",&price,&quantity);

if(quantity>1000.0)

{

price=price-(0.1\*price);

printf("reduced price is %f",price);

}

else

{

printf("no discount can be applied!");

}

return 0;

}

1. 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.
2. 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 a,b,c;

printf("enter values for checking if the triangle is valid or not:");

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

int sum;

sum=a+b+c;

if(sum==180)

{

printf("triangle is valid:");

}

else

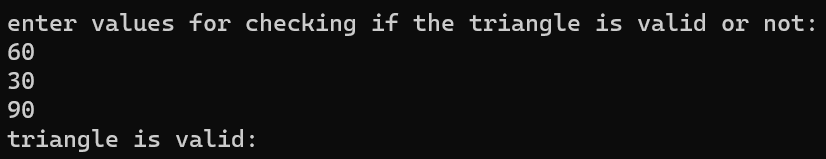
{

printf("triangle is invalid:");

}

return 0;

}



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

#include <stdio.h>

int main()

{

float age,x,n;

printf("enter your age:");

scanf("%f",&age);

x=(int)age;

n=age-x;

if(n>0.0);

{

printf("ENTER AGE IN INTEGER!");

scanf("%f",&age);

if(age>=18)

{

printf("eligible to vote:");

}

else

{

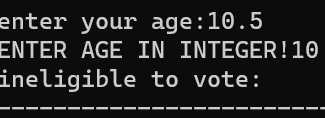
printf("ineligible to vote:");

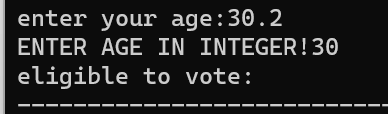
}

}

return 0;

}





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.

Arpasland has surrounded by attackers. A truck enters the city. The driver claims the load is food and medicine from Iranians. Ali is one of the soldiers in Arpasland. He doubts about the truck, maybe it's from the siege. He knows that a tag is valid if the sum of every two consecutive digits of it is even and its letter is not a vowel. Determine if the tag of the truck is valid or not.

We consider the letters **"A","E","I","O","U","Y**" to be vowels for this problem.

<https://www.hackerearth.com/practice/basic-programming/input-output/basics-of-input-output/practice-problems/algorithm/cartag-948c2b02/>

**LIST OF FLIP EXPERIMENTS**

1. What is the usage of nested if statements?

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;

}

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

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

**HACKEREARTH AND HACKERRANK**

**1.**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

    char s[100];

    scanf("%[^\n]%\*c", &s);

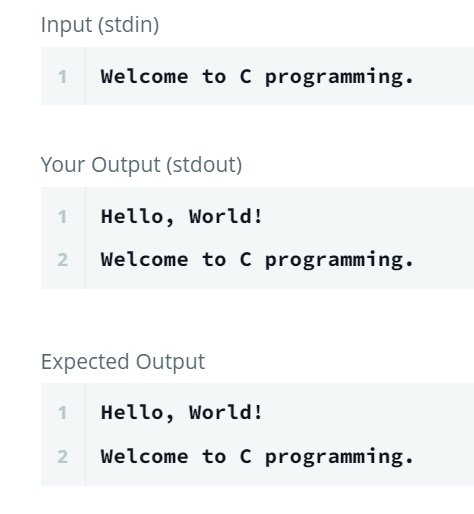
    /\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

    printf("Hello, World!\n");

    printf("%s", s);

    return 0;

}

****

**2.**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

    int var1,var2;

    float var3,var4;

    scanf("%d %d", &var1 ,&var2);

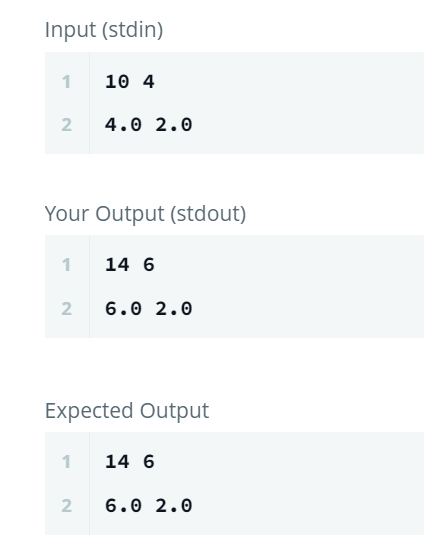
    scanf("%f %f",&var3 ,&var4);

    printf("%d %d\n",(var1 + var2),(var1 - var2));

    printf("%.1f %.1f\n",(var3 + var4),(var3 - var4));

    return 0;

}

****

**3.**

#include <stdio.h>

int main(){

    int num;

    scanf("%d", &num);

*//printf("Input number is %d.\n", num);*

    int i=1, fact=1;

    while (i<=num)

    {

        fact=fact\*i;

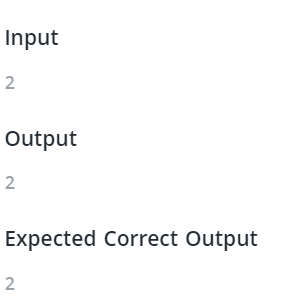
        i++;

    }

    printf("%d",fact);

    return 0;

}

****

**Practical No: 4**

Student Name and Roll Number:

Semester /Section:

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 checking conditions 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)

1. Duplicate case values are not allowed.
2. The default statement is optional. Even if the switch case statement do not have a default statement, it would run without any problem.
3. 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.
4. 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.
5. 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 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**

**Output: (with Screenshots)**

2. A character is entered through keyboard. Write a C program 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

**Code**

**Output: (with Screenshots)**

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 >= 90 : Grade A

Marks >= 70 && < 90 : Grade B

Marks >= 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**

**Output: (with Screenshots)**

**Suggested Question Bank**

1. Why do we use of break keyword?

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

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

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

1. Only int can be used in “switch” control expression
2. Both int and char can be used in “switch” control expression.
3. All types i.e. int, char and float can be used in “switch” control expression
4. “switch” control expression can be empty as well

**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;

}

}

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");

}

}

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;

}

}

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;  } |

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") ;  } |

**Practical No: 5**

Student Name and Roll Number:

Semester /Section:

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.

**Code:**

**Output (with screenshots)**

* + - 1. Program to add numbers until the user enters zero.

**Code:**

**Output (with screenshots)**

* + - 1. Program to print the sum of 10 natural numbers using while loop.

**Code:**

**Output (with screenshots)**

**Suggested Question Bank**

1. Differentiate between while and do while loop.

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 ; } }

b) void main( )

{

float x = 1.1 ;

while ( x == 1.1 )

{

printf ( "\n%f", x ) ;

x = x – 0.1 ; } }

**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");

}

b) #include<stdio.h>

int main()

{ int i = 0;

while(i < 3, i = 0, i < 5)

{ printf("Loop ");

i++; }

return 0; }

c) #include<stdio.h>

int main()

{ int i = 0;

while(i++)

{

printf("Loop ");

if(i == 3) break; }

return 0; }

**Practical No: 6**

Student Name and Roll Number:

Semester /Section:

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.

**Code:**

**Output (with screenshots)**

1. Write a program to print the following pattern:

1

* + 1. 2

1 2 3

  1 2 3 4

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

**Code:**

**Output (with screenshots)**

1. Write a program to print the following pattern:

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.

**Code:**

**Output (with screenshots)**

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

**Code:**

**Output (with screenshots)**

**hackerrank**

For each integer  in the interval  (given as input) :

* If , then print the English representation of it in lowercase. That is "one" for , "two" for , and so on.
* Else if  and it is an even number, then print "even".
* Else if  and it is an odd number, then print "odd".

https://www.hackerrank.com/challenges/for-loop-in-c/problem?isFullScreen=true

**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;

}

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;

}

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;

}

**Practical No: 7**

Student Name and Roll Number:

Semester /Section:

Link to Code:

Date:

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 program using function to check whether the number can be expressed as the sum of  
   two prime numbers.  
   Definition of Done:  
   a) The program asks the user to input an integer.  
   b) The user shall make a user-defined function named checksumPrime which takes one integer  
   arguments and returns an integer.  
   c) The program shall use for loop and if statement in the function.

**Code:**

**Output (with screenshots)**

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

**Code:**

**Output (with screenshots)**

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

**Code:**

**Output (with screenshots)**

**Hackerrank**

Bitwise operators:

<https://www.hackerrank.com/challenges/bitwise-operators-in-c/problem?isFullScreen=true>

**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");

}

2. #include<stdio.h>

int main()

{ main();

return 0;

}

3. #include<stdio.h>

int function();

main()

{ int i;

i = function();

printf("%d", i);

return 0;

}

function()

{

int a;

a = 250;

}

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)); }

**Practical No: 8**

Student Name and Roll Number:

Semester /Section:

Link to Code:

Date:

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 bytes (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]);

}

2. #include <stdio.h>

int main()

{

int ary[4] = {1, 2, 3, 4};

int \*p = ary + 3;

printf("%d\n", p[-2]);

}

3. # include <stdio.h>

void fun(int \*ptr)

{ \*ptr = 30; }

int main() {

int y = 20;

fun(&y);

printf("%d", y);

return 0; }

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;

}

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; }

**Practical No: 9**

Student Name and Roll Number:

Semester /Section:

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.

[](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2015/05/Arrays.png)

**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.

1. DoD 1: The size of the array should be given by the user. It has to be 1-D array.
2. DoD 2: The elements of the array should also be given by the user.
3. 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.

**Hackerearth**

palindrome lover

https://www.hackerearth.com/practice/basic-programming/bit-manipulation/basics-of-bit-manipulation/practice-problems/algorithm/palindrome-lover-d7beb2b0/

**LIST OF FLIP QUESTIONS:**

Q1. What is the right way to Initialize array?

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

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

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

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

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:

Semester /Section:

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 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 ) ; }

1. 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. 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 \_\_\_\_\_\_ character, which is written as \_\_\_\_\_\_.

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

Semester /Section:

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** | **>=12** |
| **Increase in salary** | **$50** | **$100** | **$150** |

DODs

1. Write a structure to store the names, salary, and hours of work per day of 10 employees in a company.
2. Print the name of all the employees along with their final salaries.
3. 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];

charlanguage[10].

} a ;

staticstructa = {"Hacker","C"};

printf ("%s %s", aname, alanguage);}

**Practical No: 12**

Student Name and Roll Number:

Semester /Section:

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 fprinf(), 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**

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**

**C. FILE is a stream**

1. **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**

**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.

1. **Only 1 B. Only 2 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**

1. **All of the above**

6. #include<stdio.h>

int main()

{ printf("%d", EOF);

return 0; }

**A. NULL B. -1 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**

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. None of the above**

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

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

**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**

**Experiment No: (Mini Project)**

Student Name and Roll Number:

Semester /Section:

Link to Code:

Date:

Faculty Signature:

Remarks:

**Project Title:**

**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)**