**Computer Programming**

CE-102

Semester 2nd

Name: Eman Ayaz\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll Number: \_\_\_\_\_\_\_\_\_\_15954\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Computer Systems Engineering Department

IQRA University, Karachi

List of Experiments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lab No** | **Topics** | **Mapping** | | **Remarks** |
| **PLO 2,3,5**  **(5)** | **PLO 9**  **(5)** |
|  | To get familiar with Programming Environment and Fundamentals of Programming Language |  |  |  |
|  | To get familiar with Operators and Math library |  |  |  |
|  | To get familiar with the use of Control Structures |  |  |  |
|  | To understand the structure and procedure of switch cases in C++ |  |  |  |
|  | To understand the structure and procedure of Loops in C++ |  |  |  |
|  | To develop understanding of nested For loop phenomenon. |  |  |  |
|  | The purpose of this lab is to get familiar with the concept of numbers and arrays |  |  |  |
|  | The purpose of this lab is to get a familiar with the concept and description multidimensional arrays |  |  |  |
|  | To be familiarized with Introduction of Pre-define Function (PDF) and User-define Function (UDF) |  |  |  |
|  | The purpose of this lab is to get understanding about structures in C++ |  |  |  |
|  | To familiarize with the usage of the Filling in C++ |  |  |  |
|  | **Total Marks Obtained** |  |  |  |

***Overall Score: \_\_\_\_\_\_\_\_\_\_\_ out of 110 Examined by****: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­*

***Overall Formula= (Obtained Score / Total Score)* x *25******(Name and Signature of Concerned)***

**Lab Assessment Rubrics (Software)**

**Method:**

Lab report and instructor observation during Lab session

**Outcome Assessed:**

1. Ability to analyze/design solutions to conduct experiments and comprehend modem engineering tools to develop skills. (Cognitive).
2. Ability to perform in a team and individually, also being able to design reports and documentation (Affective).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance** | | **Outstanding Performance** | **Good Performance** | **Average Performance** | **Below Average Performance** | **Unsatisfactory Performance** | **Taxonomy** LEVEL | **PLO** |
| **(5)** | **(4)** | **(3)** | **(2)** | **(0-1)** |
| I. | Conducting Experiment {a) | Students present a clear and concise plan/ solution to solve the problem. The code is completely functional with complete steps producing the correct outputs. | Students present a clear and concise plan/ solution to solve the problem. The code is functional with complete steps producing the outputs with acceptable error. | Student require effective guidance to present a clear and concise plan/ solution to solve the problem. The code is partially functional or has missing steps producing  the outputs with a minor error | Students require effective guidance to present a clear and concise plan/ solution to solve the problem. The code has substantial errors and has missing steps producing the outputs with errors. | The student fails to present a coherent plan to solve the problem. The student fails to code with complete steps to get the correct outputs. | C3,P3 | 2,3,5 |
| 2. | Individual  & Team work  [b] | An active cooperation develops among group members to support the execution of the experiment as a team and individually | Cooperation among group members co support the execution of the experiment as a team and individually was reasonable. | Shows average cooperation with the group members as a team and as an individual to perform the experiment. | Frequent conflict occurs among the group members as a team and as an individual to perform the experiment. | Negatively affects the other group members and creates hindrance in successful performance or completion of experiment as an individual or as a team member. | A2 | 9 |

Computer Programming (3+1)

### *Instructor:* Engr. Sidra Rehman **Office Hours: (9-5)**

***Prerequisites:* Nil**

***Objectives:***

This lab aims to cover overview of Computer Programming, Principles of Structured and Modular Programming, Overview of Structured Programming Languages, Algorithms and Problem Solving, Program Development: Analyzing Problem, Designing Algorithm/Solution, Testing Designed Solution, Translating Algorithms into Programs, Fundamental Programming Constructs.

***Contents:***

Basics of Input and Output, Selection and Decision (If, If-Else, Nested If-Else, Switch Statement and Condition Operator), Repetition (While and For Loop, Do-While Loops), Break Statement, Continue Statement, Control Structures, Functions, Arrays, Pointers, Records, Files (Input-Output), Testing & Debugging.

***Learning Outcomes:***

| **Mapping of CLOs and PLOs** | | | |
| --- | --- | --- | --- |
| **S. No** | **Course Learning Outcomes (CLOs)** | **Domain Level** | **PLO** |
| CLO1 | Describe and illustrate fundamentals of Programming. | C3 | 5 |
| CLO2 | Apply Programming skills on a particular condition and identify logical and syntax errors | P3 | 2 |
| CLO3 | Design, Implementation and Debug of small-medium scale programs to gives problem solution. | P3 | 3 |
| CLO4 | Function Individually as well as a team. | A2 | 9 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CLO Assessment Mechanism** | | | | |
| **Assessment tools** | **CLO\_1** | **CLO\_2** | **CLO\_3** | **CLO\_4** |
| Lab/Manual | 25% | 25% | 25% | 25% |
| Viva+ Project | 25% | 25% | 25% | 25% |
| Final Examination | 25% | 50% | 25% | - |

|  |  |
| --- | --- |
| **Overall Grading Policy** | |
| **Assessment Items** | **Percentage** |
| Lab/Manual | 25% |
| Project + Viva | 25% |
| Final Examination | 50% |

***ecommended Book:***

**C++ How to Programme, Paul Deitel and Harvey Deitel, Prentice Hall; 9th edition**

**(**ISBN-13: 9780133146141)

***Administrative Instructions:***

* Title and Group members name for Lab/Course project should be submitted by 5thweek of lab.
* According to institute policy, 75% attendance is *mandatory* to appear in the final examination but 100% will be expected. Approved leaves will not be considered towards attendance.
* Every student should bring calculator, book and manual in each lab.
* Every student is expected to be in lab before schedule starting time.
* In any case there will be no rescheduling and makeup of labs.

**General Laboratory Procedure**

While there is no specific document to be submitted at the beginning of the Lab –unless your instructor advises you otherwise-, you are expected to read the experiment fully before you come to the laboratory? Interestingly, you can even try parts of the experiment at home. Here is a list of programs that will equip you with a virtual lab at your home:

**Troubleshooting**

Things will not always go as expected; this is the nature of the learning process. While conducting the Experiment **think before you do anything.** If you do so you will avoid wasting time going down dead-end streets. Be logical and systematic. First, look for obvious errors that are easy to fix. Is your measuring device correctly set and connected? Are you looking at the proper scale? Is the power supply set for the correct voltage? Is the signal generator correctly set and connected? How are the variables in the code set? Is there a syntax error? And so on. Next, check for obvious misconnections or broken connections, at least in simple circuits.

As you work through your circuit, use your Lab Manual record tests and changes that you make as you go along; don't rely on your memory for what you have tried. Identify some test points in the system at which you know what the signal should be and work your way backwards from the output through the test points until you find a good signal.

**Neatness**

When you have finished for the day, return all modules to their proper storage bins, return all test leads and probes to their storage racks, return all equipment to its correct location, and clean up the lab station. If appropriate switch off the unneeded equipment. Save your files in the Computer and on any USB device for your records because you might not get the same PC System again for the next experiment. Also email your file contents to your email address as a backup.

**Laboratory Safety**



Always pay attention to what you are doing and you’re surrounding during the experiments, notify the Instructor for any unlikely event or mishap, and leave the Laboratory with the permission of Instructor immediately.

All students must read and understand the information in this document with regard to laboratory safety and emergency procedures prior to the first laboratory session.

**Your personal laboratory safety depends mostly on YOU**. Efforts have been made to address situations that may pose a hazard in the lab but the information and instructions provided cannot be considered all-inclusive.

Students must adhere to written and verbal safety instructions throughout the academic term. Since additional instructions may be given at the beginning of laboratory sessions, it is important that all students arrive at each session on time. With good judgment, the chance of an accident in this course is very small. Nevertheless, research and teaching workplaces (labs, shops, etc.) are full of potential hazards that can cause serious injury and or damage to the equipment. Working alone and unsupervised in laboratories is forbidden if you are working with hazardous substances or equipment. With prior approval, at least two people should be present so that one can shut down equipment and call for help in the event of an emergency. Safety training and/or information should be provided by a faculty member, teaching assistant, lab safety contact, or staff member at the beginning of a new assignment or when a new hazard is introduced into the workplace.

**Emergency Response**

1. It is your responsibility to read safety and fire alarm posters and follow the instructions during an emergency
2. Know the location of the fire extinguisher, eye wash, and safety shower in your lab and know how to use them.
3. Notify your instructor immediately after any injury, fire or explosion, or spill.
4. Know the building evacuation procedures.

**Common Sense**

Good common sense is needed for safety in a laboratory. It is expected that each student will work in a responsible manner and exercise good judgment and common sense. If at any time you are not sure how to handle a particular situation, ask your Teaching Assistant or Instructor for advice **DO NOT TOUCH ANYTHING WITH WHICH YOU ARE NOT COMPLETELY FAMILIAR**!!! It is always better to ask questions than to risk harm to yourself or damage to the equipment.

**Personal and General laboratory safety**

1. Never eat, drink, or smoke while working in the laboratory.
2. Read labels carefully.
3. Do not use any equipment unless you are trained and approved as a user by your supervisor.
4. Wear safety glasses or face shields when working with hazardous materials and/or equipment.
5. Wear gloves when using any hazardous or toxic agent.
6. Clothing: When handling dangerous substances, wear gloves, laboratory coats, and safety shield or glasses. Shorts and sandals should not be worn in the lab at any time. Shoes are required when working in the machine shops.
7. If you have long hair or loose clothes, make sure it is tied back or confined.
8. Keep the work area clear of all materials except those needed for your work. Coats should be hung in the hall or placed in a locker. Extra books, purses, etc. should be kept away from equipment that requires air flow or ventilation to prevent overheating.
9. Disposal - Students are responsible for the proper disposal of used material if any in appropriate containers.
10. Equipment Failure - If a piece of equipment fails while being used, report it immediately to your lab assistant or tutor. Never try to fix the problem yourself because you could harm yourself and others.
11. If leaving a lab unattended, turn off all ignition sources and lock the doors.
12. Never pipette anything by mouth.
13. Clean up your work area before leaving.
14. Wash hands before leaving the lab and before eating.
15. Unauthorized person(s) shall not be allowed in a laboratory for any reason

**Electrical safety**

1. Obtain permission before operating any high voltage equipment.
2. Maintain an unobstructed access to all electrical panels.
3. Wiring or other electrical modifications must be referred to the Electronics Shop or the Building Coordinator.
4. Avoid using extension cords whenever possible. If you must use one, obtain a heavy- duty one that is electrically grounded, with its own fuse, and install it safely. Extension cords should not go under doors, across aisles, be hung from the ceiling, or plugged into other extension cords.
5. Never, ever modify, attach or otherwise change any high voltage equipment.
6. Always make sure all capacitors are discharged (using a grounded cable with an insulating handle) before touching high voltage leads or the "inside" of any equipment even after it has been turned off. Capacitors can hold charge for many hours after the equipment has been turned off.
7. When you are adjusting any high voltage equipment or a laser which is powered with a high voltage supply, USE ONLY ONE HAND. Your other hand is best placed in a pocket or behind your back. This procedure eliminates the possibility of an accident where high voltage current flows up one arm, through your chest, and down the other arm.
8. Discard damaged cords, cords that become hot, or cords with exposed wiring.
9. Before equipment is energized ensure, (1) circuit connections and layout have been checked by a Teaching Assistant (TA) and (2) all colleagues in your group give their assent.
10. Know the correct handling, storage and disposal procedures for batteries, cells, capacitors, inductors and other high energy-storage devices.
11. Experiments left unattended should be isolated from the power supplies. If for a special reason, it must be left on, a barrier and a warning notice are required.
12. Equipment found to be faulty in any way should be reported to the Lab Engineer immediately and taken out of service until inspected and declared safe.
13. Voltages above 50 V rms AC and 120 V DC are always dangerous. Extra precautions should be considered as voltage levels are increased.
14. Never make any changes to circuits or mechanical layout without first isolating the circuit by switching off and removing connections to power supplies.
15. Know what you must do in an emergency.
16. Emergency Power Off: Every lab is equipped with and Emergency Power Off System.
17. Only authorized personnel are permitted to reset power once the Emergency Power Off system has been engaged.

**Electrical Emergency Response**

The following instructions provide guidelines for handling two types of electrical emergencies:

1. When someone suffers serious electrical shock, he or she may be knocked unconscious. If the victim is still in contact with the electrical current, immediately turn off the electrical power source. If you cannot disconnect the power source, depress the Emergency Power Off switch.
2. Do not touch a victim that is still in contact with a live power source; you could be electrocuted.
3. Have someone call for emergency medical assistance immediately. Administer first-aid, as appropriate.
4. If an electrical fire occurs, try to disconnect the electrical power source, if possible. If the fire is small and you are not in immediate danger; and you have been properly trained in fighting fires, use the correct type of fire extinguisher to extinguish the fire. When in doubt, push in the Emergency Power Off button.
5. NEVER use water to extinguish an electrical fire.

**Mechanical safety**

1. When using compressed air, use only approved nozzles and never direct the air towards any person.
2. Guards on machinery must be in place during operation.
3. Exercise care when working with or near hydraulically- or pneumatically-driven equipment. Sudden or unexpected motion can inflict serious injury.

**Additional Safety Guidelines**

1. Never do unauthorized experiments.
2. Never work alone in laboratory.
3. Keep your lab space clean and organized.
4. Do not leave an on-going experiment unattended.
5. Always inform your instructor if you break a thermometer. Do not clean mercury yourself!!
6. Never taste anything. Never pipette by mouth; use a bulb.
7. Never use open flames in laboratory unless instructed by TA.
8. Check your glassware for cracks and chips each time you use it. Cracks could cause the glassware to fail during use and cause serious injury to you or lab mates.
9. Maintain unobstructed access to all exits, fire extinguishers, electrical panels, emergency showers, and eye washes.
10. Do not use corridors for storage or work areas.
11. Do not store heavy items above table height. Any overhead storage of supplies on top of cabinets should be limited to lightweight items only. Also, remember that a 36" diameter area around all fire sprinkler heads must be kept clear at all times.
12. Areas containing lasers, biohazards, radioisotopes, and carcinogens should be posted accordingly. However, do not post areas unnecessarily and be sure that the labels are removed when the hazards are no longer present.
13. Be careful when lifting heavy objects. Only shop staff may operate forklifts or cranes.
14. Clean your lab bench and equipment, and lock the door before you leave the laboratory.

**Clothing**

1. Dress properly during a laboratory activity.
2. Long hair, dangling jewelry, and loose or baggy clothing are a hazard in the laboratory.
3. Long hair must be tied back, and dangling jewelry and baggy clothing must be secured.
4. Shoes must completely cover the foot.
5. No sandals allowed on lab days.
6. A lab coat or smock should be worn during laboratory experiments.

**Accidents and Injuries**

1. Do not panic.
2. Report any accident (spill, breakage, etc.) or injury (cut, burn, etc.) to the teacher immediately, no matter how trivial it seems.
3. If you or your lab partner is hurt, immediately (and loudly) yell out the teacher's name to get the teacher's attention.

**General Warning Signs**



**Lab - 01**

***To get familiar with Programming Environment and Fundamentals of***

***Programming Language***

**Theory:**

**The Development Environment - Integrated Development Environment (IDE):**

The C compiler has its own built-in text editor. You may also use a commercial text editor or word processor that can produce text files. The important thing is that whatever you write your program in, it must save simple, plain-text files, with no word processing commands embedded in the text. The files you create with your editor are called source files, and for

C++ they typically are named with the extension *.CPP*, *.CP*, or *.C*.

The C Developing Environment, also called as Programmers’ Platform, is a screen display with windows and pull-down menus. Code of the program, error messages and other information are displayed in separate windows. The menus may be used to invoke the operations necessary to develop the program, debug and execute the program.

**Invoking the IDE**

To invoke the IDE from the windows you need to double click the Dev C++ icon.

This makes you enter the IDE interface, which initially displays only a menu bar at the top of the screen and a status line below will appear. The menu bar displays the menu names and the status line tells what various function keys will do.

**Opening New Window**

To type a program, you need to open an Edit Window. For this, open file menu and click “new” or press [CTRL +N]. A window will appear on the screen where the program may be typed.

**Writing a Program**

When the Edit window is active, the program may be typed. Use the certain key combinations to perform specific edit functions.

**Saving a Program**

To save the program, select **save** command from the **file** menu. This function can also be performed by pressing the [CTRL+S]. A dialog box will appear asking for the path and name of the file. Provide an appropriate and unique file name. You can save the program after compiling too but saving it before compilation is more appropriate.

**Compiling the Source Code**

Although the source code in your file is somewhat cryptic, and anyone who doesn't know C will struggle to understand what it is for, it is still in what we call human-readable form. But, for the computer to understand this source code, it must be converted into machine-readable form. This is done by using a compiler. Hence, compiling is the process in which source code is translated into machine understandable language.

**Creating an Executable File with the Linker**

After your source code is compiled, an object file is produced. This file is often named with the extension .*OBJ*. This is still not an executable program, however. To turn this into an executable program, you must run your linker. C programs are typically created by linking together one or more OBJ files with one or more libraries. A library is a collection of linkable files that were supplied with your compiler.

**Project/Make**

Before compiling and linking a file, a part of the IDE called Project/Make checks the time and date on the file you are going to compile.

**Compiling and linking in the IDE**

In the Dev C++, compiling and linking can be performed together in one step. There are two ways to do this: you can select Make *EXE* from the compile menu, or you can press the F9 key.

**Executing a Program**

If the program is compiled and linked without errors, the program is executed by selecting Run from the Run Menu or by pressing the F10 key.

**The Development Cycle**

If every program worked the first time you tried it that would be the complete development cycle: Write the program, compile the source code, link the program, and run it.

Unfortunately, almost every program, no matter how trivial, can and will have errors, or bugs, in the program. Some bugs will cause the compile to fail, some will cause the link to fail, and some will only show up when you run the program. Whatever type of bug you find, you must fix it, and that involves editing your source code, recompiling and relinking, and then rerunning the program.

**Correcting Errors**

If the compiler recognizes some error, it will let you know through the Compiler window.

You’ll see that the number of errors is not listed as 0, and the word “Error” appears instead of the word “Success” at the bottom of the window. The errors are to be removed by returning to the edit window. Usually these errors are a result of a typing mistake. The compiler will not only tell you what you did wrong; they’ll point you to the exact place in your code where you made the mistake.

**Exiting IDE**

An Edit window may be closed in a number of different ways. You can click on the small square in the upper left corner, you can select **close** from the **window** menu, or you can press the [ALT+F4] combination.

**Building Blocks of Programming Language:**

In any language there are certain building blocks:

* Constants
* Variables
* Operators
* Methods to get input from user (cin , getch( ) etc.)
* Methods to display output (Escape Sequences etc.) and so on.

**Variables and Constants**

If the value of an item can be changed in the program then it is a variable. If it will not change then that item is a constant. The various variable types (also called *data type*) in C are: *int*, *float*, *char*, *long* etc. For constants, the keyword ***const*** is added before declaration.

**Operators**

There are various types of operators that may be placed in three categories:

***Basic*: + - \* / %**

***Assignment*: = += - = \*= /= %=**

(++, - - may also be considered as assignment operators)

***Relational*: < > <= >= == !=**

**Escape Sequences**

Escape Sequence causes the program to **escape** from the normal interpretation of a string, so that the next character is recognized as having a special meaning. The back slash “\” character is called the **Escape Character”**. The escape sequence includes the following:

\n => new line

\b => back space

\r => carriage return

\” => double quotations

**Getting Input From the User**

The input from the user can be taken by the following techniques: cin, getch( ), getche(), getchar( ) etc.

**Examples:**

1. **Implementing a Simple C Program**

#include<iostream>

using namespace std;

int main()

{

cout<<"\n Hello World";

return 0;

}



1. **Demonstrating the fundamentals of C Language**

#include<iostream>

using namespace std;

int main()

{

int num1,num2,sum,product;

cout<<" The program takes two numbers as input and prints their sum and product"<<endl;

cout<<" Enter first number:";

cin>>num1;

cout<<" Enter second number:";

cin>>num2;

sum=num1+num2;

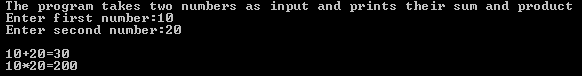
product=num1\*num2;

cout<<"\n "<<num1<<"+"<<num2<<"="<<sum;

cout<<"\n "<<num1<<"\*"<<num2<<"="<<product;

return 0;

}



**Lab Tasks**

1. Write a program to calculate the Area ( A= r2 ) and circumference of a circle (*C=r*), where r = radius is taken as input and is declared as a constant. The precision of should be the number of characters in your name. Display the result to 4 decimal places.
2. Write single C ++ statement to output the following on the screen:

My name is “*Your Name*”

And roll number is “*Your\_roll\_no*”

I am a student of “*Your Department*”

1. Write a program that perform basic arithmetic operations by getting values from User.
2. Write a program that accept value from user and display its square and cube without using predefined functions?

**Answer 01**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

float num1, sq, area, cir;

cout<<"Enter radius here= ";

cin>>num1;

sq=num1\*num1;

area=3.142\*sq;

cout<<"Area of circle is=";

cout<<area;

cir=2\*3.1428\*num1;

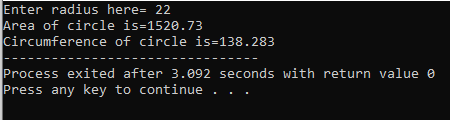
cout<<"\nCircumference of circle is=";

cout<<cir;

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include <iostream>

using namespace std;

int main()

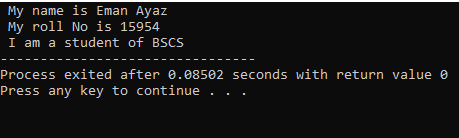
{

cout<<"My name is Eman Ayaz\nMy roll No is 15954\nI am a student of BSCS";

return 0;

}

**Output**



**Answer 03**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int num1, num2, mul, sub, sum;

float div;

cout<<"Enter First Number: ";

cin>>num1;

cout<<"\nEnter Second Number: ";

cin>>num2;

div=num1/num2;

cout<<"\n"<<num1<<"/"<<num2<<"="<<div<<"\n";

mul=num1\*num2;

cout<<num1<<"\*"<<num2<<"="<<mul<<"\n";

sum=num1+num2;

cout<<num1<<"+"<<num2<<"="<<sum<<"\n";

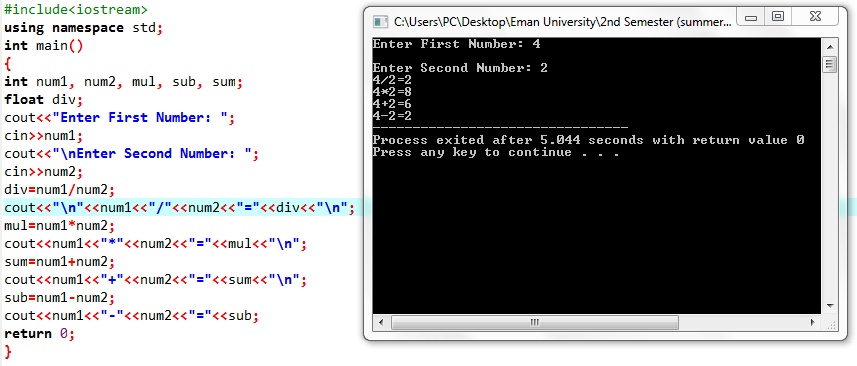
sub=num1-num2;

cout<<num1<<"-"<<num2<<"="<<sub;

return 0;

}

**Output**



**Answer 04**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int num1, cube, sq;

cout<<"Enter any number here: ";

cin>>num1;

sq=num1\*num1;

cout<<"\nThe square of the given number is: "<<sq <<"\n";

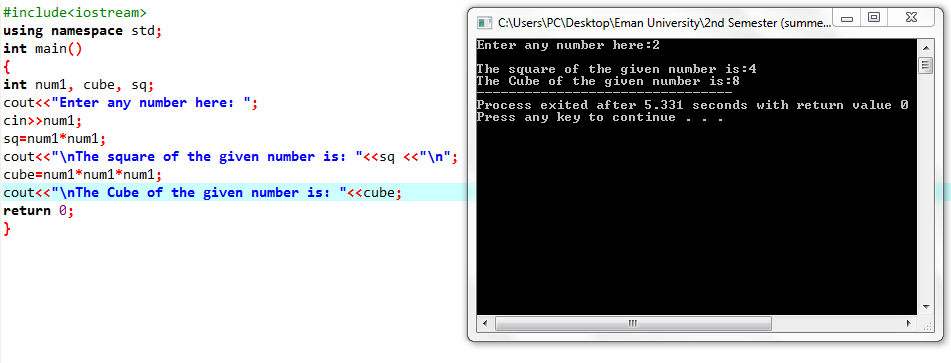
cube=num1\*num1\*num1;

cout<<"\nThe Cube of the given number is: "<<cube;

return 0;

}

**Output**



**Lab -02**

***To get familiar with Operators and Math library***

## Theory:

In the C/C++ programming language, the C/C++ Standard Library is a collection of classes and functions, which are written in the core language. The library includes several header files. Most frequently used header files are iostream, conio.h followed by preprocessor directives #include.

**#include<iostream>for C++ / #include<stdio.h> for C abbreviated for input/ output stream**

**#include<conio.h> abbreviated for console input/ output**

**#include<cmath> for C++ and #include<math.h> for C is used as mathematical functionalities like power, square root, cube root etc.**

**Note:** .h extension is used with those header files that are included in both C and C++.

Now, C++ has some collection of names referred to as the namespace. This includes various names which are not used for other purposes. For example **cin, cout**, basic input-output functions define in **std** namespace. After including header files **using namespace std** is added in global space in order to access various function in any program.

In different programs input is required to be entered from user at rum time and output is generate don the monitor screen so for this two functions are defined in **namespace std** that in Console input (**cin**), Console output (**cout**)

After declaring all header files and standard namespace, program will declare main function that is the gateway or the starting point of the program. In C++ all functions carried ( ) parenthesis.

**int** **main ()** In C++ main function has return type integer that indicates an integer will return to end up the functionality of main function and return a value to operating system.

**Syntax of Program:**

**#include <iostream>**

**#include<conio.h>**

**using namespace std;**

**int main(){**

**return 0; }**

Curly braces {} indicates the starting and ending of the program, return 0; statement returns the 0 value that indicates normal termination of program.

In main() function all the problem solving codes are written that involve different data. For example if a program is taking personal details from user to this program will includes name, mobile number, address, gender etc. that indicates some data are of only characters like name or some have numeric digits and alphabets like address.

**Data Types :**

Data types indicates type of data as string, character, integer, float, double, bool etc.

* **declaring a boolean variable which can only holds true or false value**

bool b1 = true;

* **declaring a string value**

string g= “Ali”;

string f= “6758”;

string y= “57.6”;

string d=“true”;

* **declaring a character value**

char z= ‘b’;

char j= ‘4’;

* **declaring a decimal value**

float h=25.2;

double u=612345.56;

* **declaring a integer value**

int u=3;

**Keywords:**

Like C and other languages C++ also have some keywords that cannot be used for variable names.

**Operators:**

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators and provide the following types of operators −

* Arithmetic Operators
* Relational Operators
* Logical Operators
* Bitwise Operators
* Assignment Operators

## Arithmetic Operators

There are following arithmetic operators supported by C++ language

*Assume variable A holds 10 and variable B holds 20, then*

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands | A + B will give 30 |
| - | Subtracts second operand from the first | A - B will give -10 |
| \* | Multiplies both operands | A \* B will give 200 |
| / | Divides numerator by de-numerator | B / A will give 2 |
| % | Modulus Operator and remainder of after an **integer** division | B % A will give 0 |
| ++ | [Increment operator](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), increases integer value by one | A++ will give 11 |
| -- | [Decrement operator](https://www.tutorialspoint.com/cplusplus/cpp_increment_decrement_operators.htm), decreases integer value by one | A-- will give 9 |

## Relational Operators:

There are following relational operators supported by C++ language

*Assume variable A holds 10 and variable B holds 20, then –*

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = = | Checks if the values of two operands are equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |

## Logical Operators:

There are following logical operators supported by C++ language.

Assume variable A holds 1 and variable B holds 0, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| && | Called Logical AND operator. If both the operands are non-zero, then condition becomes true. | (A && B) is false. |
| || | Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false. | ! (A && B) is true. |

## Bitwise Operators:

Bitwise operator works on bits and perform bit-by-bit operation. The truth tables for &, |, and ^ are as follows −

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **p** | **q** | **p & q** | **p | q** | **p ^ q** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Assume if A = 60; and B = 13; now in binary format they will be as follows −

A = 0011 1100

B = 0000 1101

-----------------------------

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A  = 1100 0011

The Bitwise operators supported by C++ language are listed in the following table. Assume variable A holds 60 and variable B holds 13, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| & | Binary AND Operator copies a bit to the result if it exists in both operands. | (A & B) will give 12 which is 0000 1100 |
| | | Binary OR Operator copies a bit if it exists in either operand. | (A | B) will give 61 which is 0011 1101 |
| ^ | Binary XOR Operator copies the bit if it is set in one operand but not both. | (A ^ B) will give 49 which is 0011 0001 |
| ~ | Binary Ones Complement Operator is unary and has the effect of 'flipping' bits. | (~A ) will give -61 which is 1100 0011 |
| << | Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand. | A << 2 will give 240 which is 1111 0000 |
| >> | Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand. | A >> 2 will give 15 which is 0000 1111 |

## Assignment Operators:

There are following assignment operators supported by C++ language −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Simple assignment operator, Assigns values from right side operands to left side operand. | C = A + B will assign value of A + B into C |
| += | Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand. | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand. | C -= A is equivalent to C = C – A |

**Example: Write a program that takes two integer input from user and apply assignment operators?**

#include <iostream>

#include<conio.h>

using namespace std;

int main()

{int a,b;

cout<< "Enter first number=";

cin>>a;

cout<<"Enter second number=";

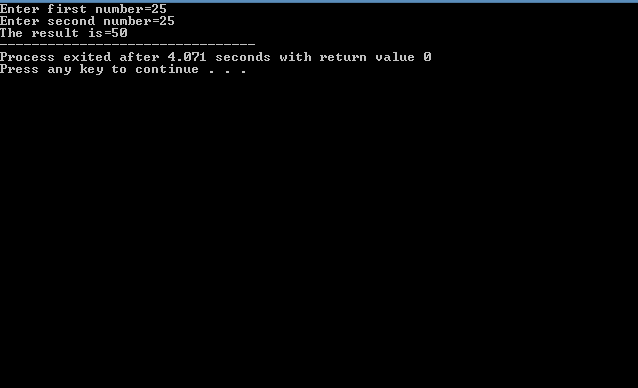
cin>>b;

a+=b;

cout<<"The result is="<<a;

return 0;

}

****

The <cmath>**header file** declares a set of functions to perform mathematical operations such as: sqrt() to calculate square root, log() to find natural logarithm of a number etc. Some of the frequently used functions of this library are mentioned as under;

* **Square root :** used to calculate the square root of the given argument

**double sqrt(double x)**

* **Cube root:** used to calculate the cube root of the given argument

**double cbrt(double x)**

* **Power:** determines the power for given exponent

**double pow(double base, double exponent);**

* **Absolute:** returns the absolute value of integer value

**int x = -5; int a = abs(x);**

* **Sine angle:** Use to calculate the sine value of given angle

**double sin (double x); //same as cos angle tangent angle**

* **Round off**: The round() function in C++ returns the integral value that is nearest to the argument, with halfway cases rounded away from zero

**double round(double x);**

* **Logarithm** returns logarithmic value of given argument

**double log10 (double x);**

* The **floor()** function in C++ returns the largest possible integer value which is less than or equal to the given argument.

**double floor(double x);**

* The **ceil()** function in C++ returns the smallest possible integer value which is greater than or equal to the given argument.

**double ceil(double x);**

**Lab Tasks**

1. Design a program to take integer input from user, store it in a variable named as num1 and apply post increment (num1 ++), pre increment (++num1), post decrement (num1 --) and pre decrement (num1--). Display each result in a new line with a track of previous value before applying each increment or decrement.
2. Write a program that takes float or double input from user and return ceiling and floor value of entered input.
3. Write a program to calculate cube root of every integer input value entered at run time.
4. Design a program to execute the expression a+b-c and print the overall final result where a,b,c are the console integer input from user?
5. Design a program that does not take input from user and calculates 65^3++pi? Where pi is a constant value of 3.142 should be declare as constant.

**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int num1;

cout << "\n\n Display the operation of pre and post increment and decrement \n";

cout << " Enter any number here: ";

cin>> num1;

cout << "--------------------------------------------------------------------\n";

cout <<" \n The number is : " << num1 << endl;

num1++;

cout <<" After post increment by 1 the number is : " << num1 << endl;

++num1;

cout <<" After pre increment by 1 the number is : " << num1 << endl;

num1--;

cout <<" After post decrement by 1 the number is : " << num1 << endl;

--num1;

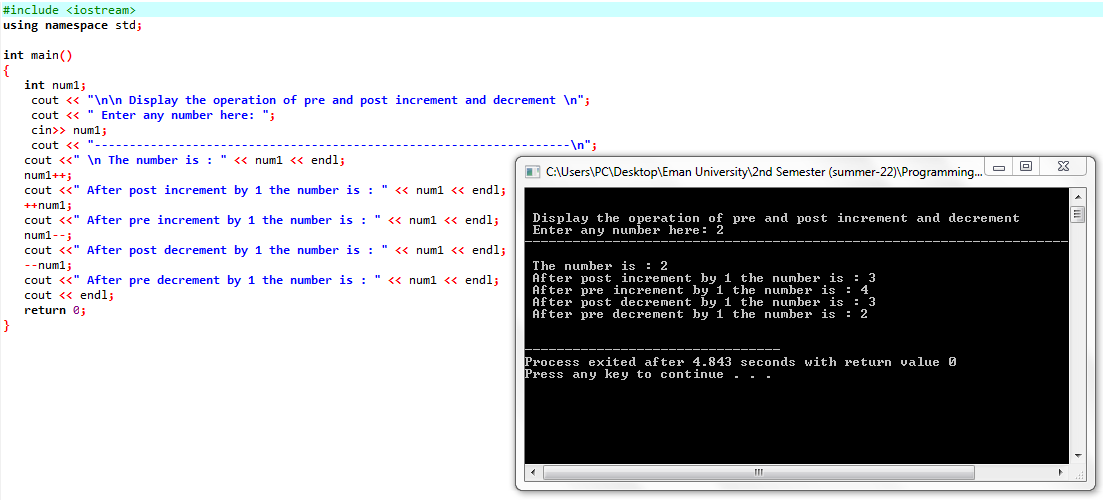
cout <<" After pre decrement by 1 the number is : " << num1 << endl;

cout << endl;

return 0;

}

**Output**

****

**Answer 02**

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

float num1;

cout <<"\nReturn Floor and Ceiling Value of Input Number";

cout <<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

cout <<"\n\nEnter any number to find floor and ceiling value: ";

cin >> num1;

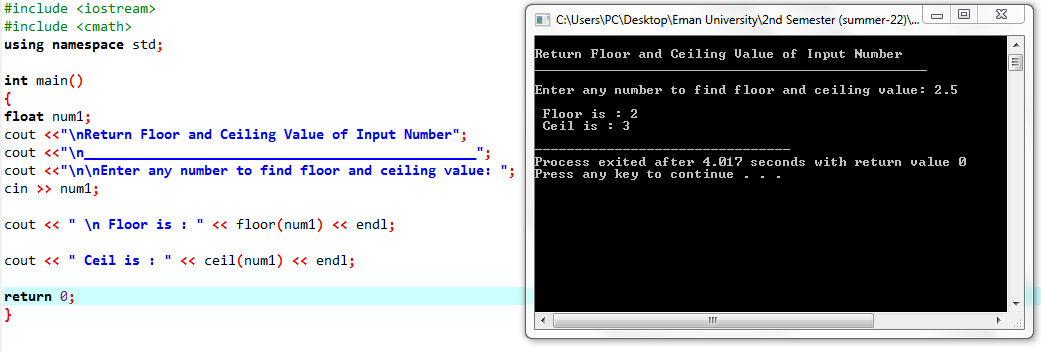
cout << " \n Floor is : " << floor(num1) << endl;

cout << " Ceil is : " << ceil(num1) << endl;

return 0;

}

**Output**



**Answer 03**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int num1, cube;

cout << " Program to Find the Cube Root of any Number";

cout << "\n\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

cout << "\n\n Enter any Number here to Find the cube root: ";

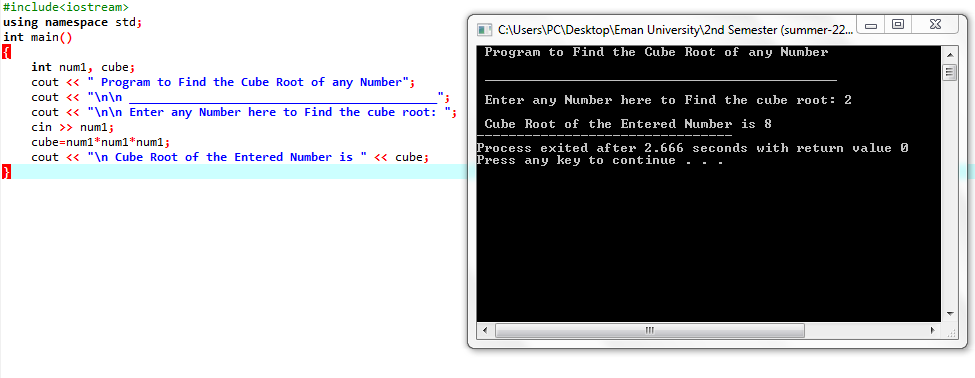
cin >> num1;

cube=num1\*num1\*num1;

cout << "\n Cube Root of the Entered Number is " << cube;

}

**Output**



**Answer 04**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int a,b,c,exp;

cout << " Calculate a+b-c";

cout << "\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

cout << "\n\n Enter the value of a: ";

cin >> a;

cout << "\n Enter the value of b: ";

cin >> b;

cout << "\n Enter the value of c: ";

cin >> c;

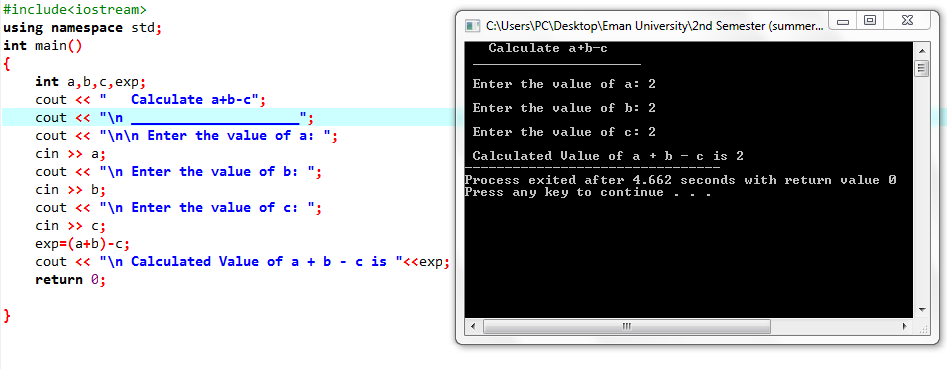
exp=(a+b)-c;

cout << "\n Calculated Value of a + b - c is "<<exp;

return 0;

}

**Output**



**Answer 05**

**Source Code:**

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

float p=3.142, cal;

cout << "\n" << " Program to Calculate the Value of 65 \* 3 + Root 625 + pi";

cout << "\n \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

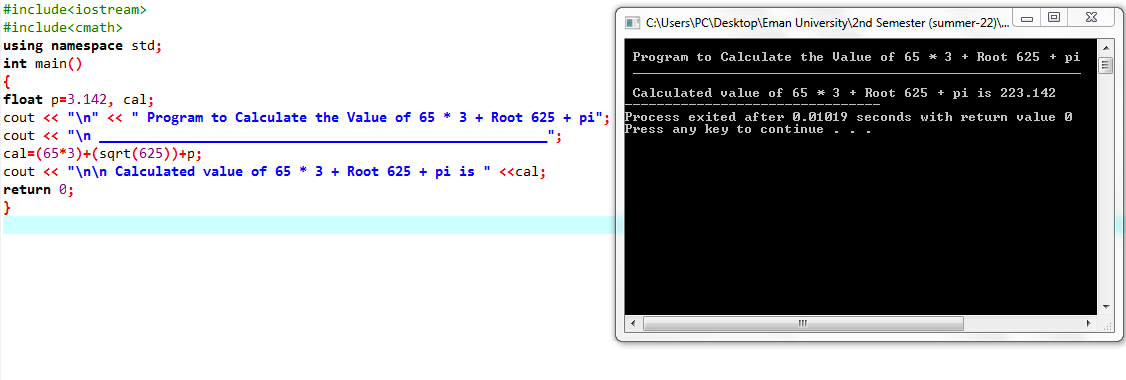
cal=(65\*3)+(sqrt(625))+p;

cout << "\n\n Calculated value of 65 \* 3 + Root 625 + pi is " <<cal;

return 0;

}

**Output**



**Lab- 03**

***To get familiar with the use of Control Structures***

**Theory:**

The ability to control the flow of your program, letting it make decisions i.e Control Structures on what code to execute, is valuable to the programmer. “if” statement allows you to control if a program enters a section of code or not based on whether a given condition is true or false, the meaning of TRUE and FALSE in computer terminology. A true statement is one that evaluates to a non-zero number. A false statement evaluates to zero, when you perform comparison with the relational operators.

1. **The IF Statement:**

This is the simplest form of the branching statements. It takes an expression in parenthesis and a statement or block of statements. If the expression is true, then the statement or block of statements gets executed otherwise these statements are skipped.

If a=10 and b=20

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | Checks if the value of two operands is equal or not, if yes then condition becomes true. | (A == B) is not true. |
| != | Checks if the value of two operands is equal or not, if values are not equal then condition becomes true. | (A != B) is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | (A > B) is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | (A < B) is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | (A >= B) is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | (A <= B) is true. |
| && | Called Logical AND operator. If both the operands are non-zero, then then condition becomes true. | (A && B) is true. |
| || | Called Logical OR Operator. If any of the two operands is non zero, then then condition becomes true. | (A || B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logic. | !(A && B) is false. |

**Syntax:**

**If (TRUE ) {**

**/\* between the braces is the body of the if statement \*/**

**Execute all statements inside the body**

**}**

**Example:**

#include<iostream>

using namespace std;

int main()

{

char ch;

cout<<"Enter any character\n";

if(ch=='y' || ch=='Y')

cout<<"\n IQRA UNIVERSITY";

else

cout<<"\n Invalid input";

return 0;

}

1. **The IF-ELSE Statement:**

Sometimes when the condition in if statement evaluates to false, it would be nice to execute some code instead of the code executed when the statement evaluates to true. The "else" statement effectively says that whatever code after it (whether a single line or code between brackets) is executed after if statement is FALSE.   
**Syntax:**

**if ( TRUE ) {**

**/\* Execute these statements if TRUE \*/**

**}**

**else {**

**/\* Execute these statements if FALSE \*/**

**}**

**Example:**

#include<iostream>

using namespace std;

int main()

{

int x,y;

cout<<" Enter value for x :";

cin>>x;

cout<<" Enter value for y :";

cin>>y;

if ( x > y )

cout<<" X is large number="<<x;

else

cout<<" Y is large number="<<y;

return 0;

}

1. **The ELSE-IF Statement:**

Another use of else is when there are multiple conditional statements that may all evaluate to true, yet you want only one if statement's body to execute. You can use an "else if" statement following an if statement and its body; that way, if the first statement is true, the "else if" will be ignored, but if the if statement is false, it will then check the condition for the else if statement. If the if statement was true, the else statement will not be checked. It is possible to use numerous else if statements to ensure that only one block of code is executed.

**Syntax:**

**if(condition)  
statement 1;  
else if (condition)  
          statement 2;  
          .....................  
          .....................  
else if(condition)  
         statement ;**

**Example:**

#include<iostream>

#include<conio.h>

using namespace std;

int main() /\* Most important part of the program! \*/

{ int age; /\* Need a variable... \*/

cout<< "\n Please enter your age: " ; /\* Asks for age \*/

cin>>age; /\* The input is put in age \*/

if ( age < 50) /\* If the age is less than 50 \*/

cout<<" You are young!\n" ; /\* Just to show you it works... \*/

else if ( age == 50 ) /\* I use else just to show an example \*/

cout<<" You are old\n" ;

else

cout<<" You are really old\n" ; /\* Executed if no other statement is \*/

return 0;

}

1. **Nested IF:**

If condition 1 is false, then condition 2 is checked and statements are executed if it is true. If condition 2 also gets failure, then else part is executed.

**Syntax:**

**if (condition1){ Statement1; }  
else\_if(condition2)   
{ Statement2; }   
else Statement 3;**

**Example:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

char username;

int password;

cout<<" Username:";

cin>>username;

cout<<" Password:";

cin>>password;

if(username=='s')

{

if(password==12345)

cout<<" Login successful";

else

cout<<" Password is incorrect, Try again.";

}

else

cout<<" Username is incorrect, Try again.";

return 0;}

**Conditional (Ternary) Operator:**

The conditional operator (?:) is C’s only ternary operator; that is, it is the only operator to take three terms.

**Syntax:**

The conditional operator takes three expressions and returns a value:

**(expression1) ? (expression2) : (expression3);**

This line is read as "If expression1 is true, return the value of expression2; otherwise, return the value of expression3." Typically, this value would be assigned to a variable.

**Example:**

#include<iostream>

using namespace std;

int main()

{ float per;

string result;

cout<<"\n Enter your percentage:";

cin>>per;

result= per >= 50 ?"Passed":"Failed";

cout<<"You are "<<result<<"!";

return 0;

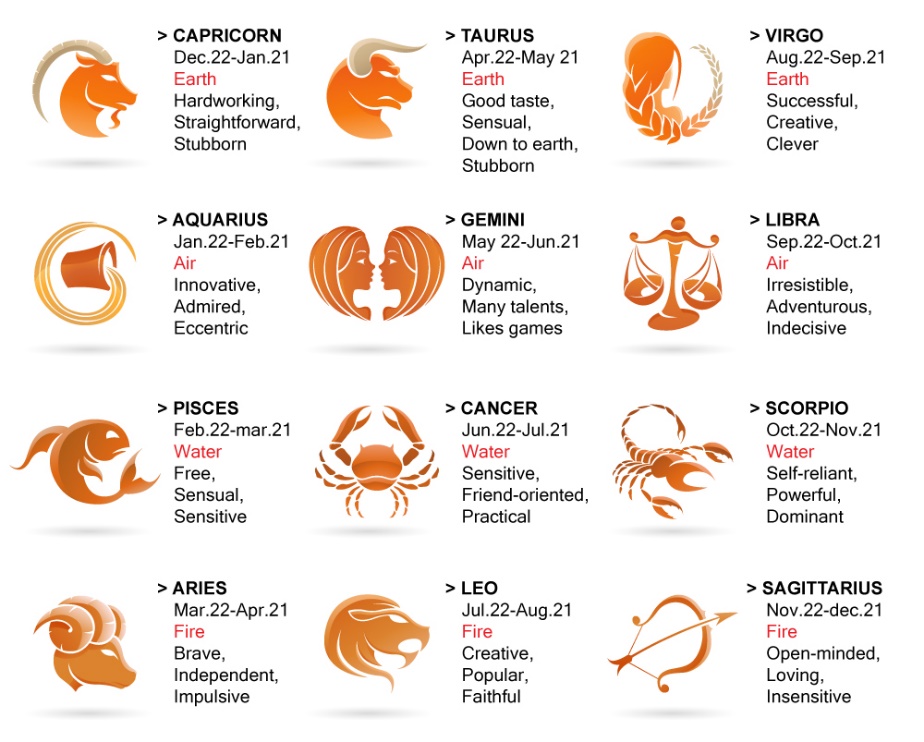
}

**Lab Tasks**

1. Write a program that display whether a number is even or odd?
2. Write a program that accept three values from a user and print the highest value from them?
3. Write a program that design a program which shows at the top Transport Route decision chart through nested if-else statement that takes integer distance in kilometers from residence to university and checks the condition in the following format:

* If distance is less than and equals to 1km than print “Come university by foot”
* If distance is above 1 km but below and equals to 3km than print “Come University by public transport”
* If distance is above 3km less than 5 km than print “Come university by university point”
* else change the university.
* Also, apply condition that indicates negative value of distance as invalid input.

1. Take integer input as the date and character as month from user and decide the zodiac sign

****

**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int n;

cout << "Program to Find the Entered Number is EVEN or ODD";

cout << "\n\nEnter an integer: ";

cin >> n;

if ( n % 2 == 0)

cout <<"\n" << n << " is even.";

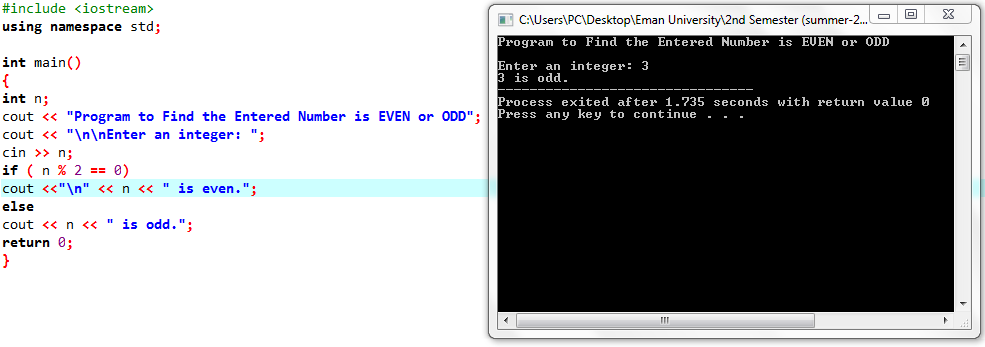
else

cout << n << " is odd.";

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int n1,n2,n3;

cout << "Program to Fine the Largest Number Among Three Numbers\n\n";

cout << "Enter First Number: ";

cin >> n1;

cout << "\nEnter Second Number: ";

cin >> n2;

cout << "\nEnter Third Number: ";

cin >> n3;

if (n1>n2 && n1>n3)

cout << n1 << " is the Largest Number\n";

else if (n2>n1 && n2>n3)

cout << n2 << " is the Largest Number\n";

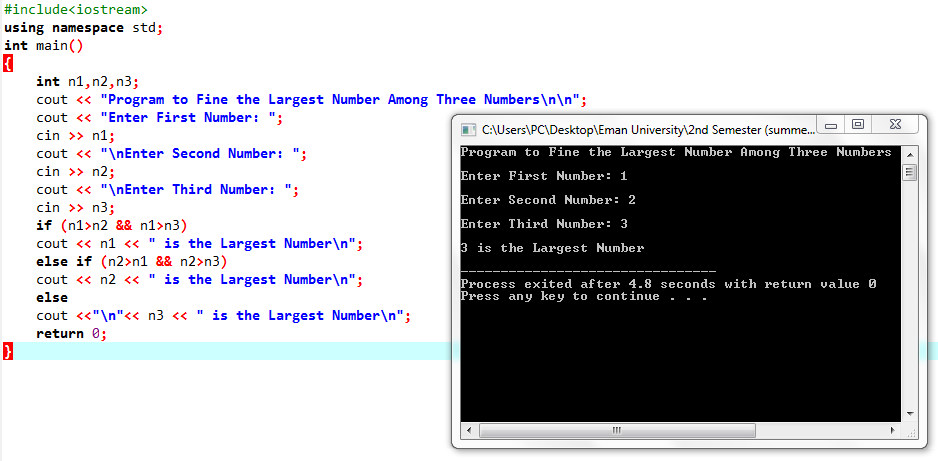
else

cout <<"\n"<< n3 << " is the Largest Number\n";

return 0;

}

**Output**



**Answer 03**

**Source Code:**

#include <iostream>

using namespace std;

int main( )

{

int age,dist;

cout<<"Welcome to Transport Route\n\n";

cout<<"Enter Your Age: ";

cin>>age;

if(age>=18)

cout<<"\nPermission granted";

else

{

cout<<"\nPermission not granted";

return 0;

}

cout<<"\n\nEnter Distance in Kilometer: \n\n";

cin>>dist;

if(dist<=1)

{

cout<<"\nCome University By Foot";

}

else if(dist>1 && dist <=3)

{

cout<<"\nCome University By Public Transport";

}

else if(dist>3 && dist<=5)

{

cout<<"\nCome University By Point";

}

else if (dist>5)

{

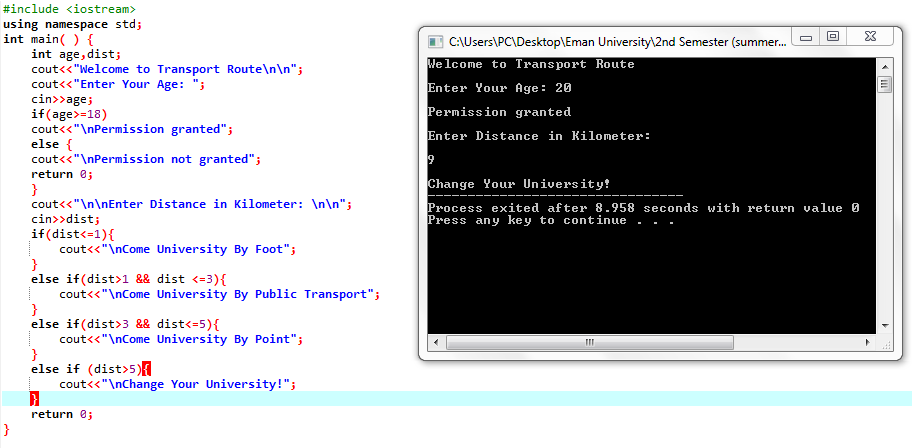
cout<<"\nChange Your University!";

}

return 0;

}

**Output**



**Answer 04**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int d;

string m, z;

cout << "\nEnter birth date: ";

cin >> d;

cout << "\nEnter birth month: ";

cin >> m;

if(d>=22 && d <= 31 && m == "december" || d <= 21 && m == "january")

{ z = "Capricorn"; }

else if(d>=22 && d <= 31 && m == "january" || d <= 21 && m == "february")

{ z = "Aquarius"; }

else if(d>=22 && d <= 31 && m == "february" || d <= 21 && m == "march")

{ z = "Pices"; }

else if(d>=22 && d <= 31 && m == "march" || d <= 21 && m == "april")

{ z = "Aries"; }

else if(d>=22 && d <= 31 && m == "april" || d <= 21 && m == "may")

{ z = "Taurus"; }

else if(d>=22 && d <= 31 && m == "may" || d <= 21 && m == "june")

{ z = "Gemini"; }

else if(d>=22 && d <= 31 && m == "june" || d <= 21 && m == "july")

{ z = "Cancer"; }

else if(d>=22 && d <= 31 && m == "july" || d <= 21 && m == "august")

{ z = "Leo"; }

else if(d>=22 && d <= 31 && m == "august" || d <= 21 && m == "september")

{ z = "Virgo"; }

else if(d>=22 && d <= 31 && m == "september" || d <= 21 && m == "october")

{ z = "Libra"; }

else if(d>=22 && d <= 31 && m == "october" || d <= 21 && m == "november")

{ z = "Scorpio"; }

else if(d>=22 && d <= 31 && m == "november" || d <= 21 && m == "december")

{ z = "Sagittarius"; }

else

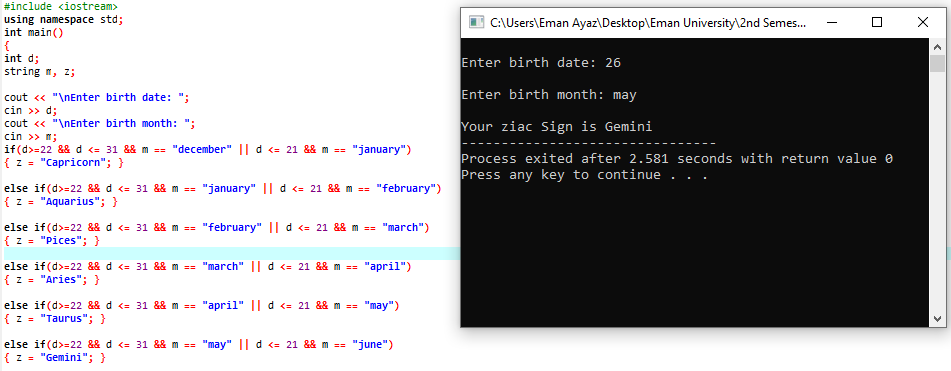
{ cout << "Invalid Input"; }

cout << "\nYour ziac Sign is " << z;

return 0;

}

**Output**



**Lab - 04**

***To understand the structure and procedure of switch cases in C++***

## Theory

**Switch Statement in C++**

Switch case statements are a substitute for long if statements that compare a variable to several integral values. The switch statement is a multi-way branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. Switch is a control statement that allows a value to change control of execution.

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

**Syntax for integer input:**

//here n is integer

switch (n)

{

case 1: // code to be executed if n = 1;

break;

case 2: // code to be executed if n = 2;

break;

default: // code to be executed if n doesn't match any cases

}

**Syntax for character input:**

//here ch is character

switch (ch)

{

case ‘a’: // code to be executed if ch = a;

break;

case ‘b’: // code to be executed if ch = b;

break;

default: // code to be executed if ch doesn't match any cases

}

***Note: character is written in single quotes ‘’ where integer, float values are written without ‘’.***

**EXAMPLE # 1: Write a program to make a Calculator using switch case**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

char c, op;

double operand1,operand2;

cout<<"Enter first number,operator,second number:";

cin>>operand1>>op>>operand2;

switch(op)

{

case '+':

cout<<"Answer="<<operand1+operand2;

break;

case '-':

cout<<"Answer="<<operand1-operand2;

break;

case '\*':

cout<<"Answer="<<operand1\*operand2;

break;

case '/':

cout<<"Answer="<<operand1/operand2;

break;

default:

cout<<"invalid operator";

}

return 0;

}

**EXAMPLE #2: Write a program to determine percentage from the grade.**

#include <iostream>

#include<conio.h>

using namespace std;

int main()

{

char grade;

cout<<"\n Enter your Grade: ";

grade=getche();

switch(grade)

{

case 'a':

cout<<"\n Your percentage is 80 or above 80 ";

break;

case 'b':

cout<<"\n Your percentage is in 79-70 ";

break;

case 'c':

cout<<"\n Your percentage is in 69-60 ";

break;

case 'd':

cout<<"\n Your percentage is in 59-50 ";

break;

default:

cout<<"\n Your percentage is below 50 ";

}

return 0;

}

**Lab Tasks**

1. **Write a program that display a Menu, and perform that operation**

*Menu*

*\*\*\*\*\*\**

1. *Even or Odd*
2. *Addition*
3. *Subtraction*
4. *Division*
5. *Multiplication*
6. *Quit*
7. **Write a program whose display heading on the top as “Admissions in Iqra University” that offers admission in Computer Science as choice 1, Software Engineering as choice 2, Computer Engineering as choice 3, Business management as choice 4, Pharmacy as choice 5.**

*The program takes integer input as choice number. And their percentage in float.*

*On the basis of their choice decide whether a student is eligible for this department or not.*

*If eligible display message “Your admission is confirmed”. Else display message “change your field”.*

*Selection criteria*

* *>=85 Computer Engineering*
* *Less than 85 and greater and equals to 82 Computer Science*
* *Less than 82 and greater and equals to 79 Software Engineering*
* *Less than 86 and greater and equals to 76 Business management*
* *Less than 88 and greater and equals to 70 Pharmacy*

1. **Write a program whose display heading on the top as “Grading Scheme in Iqra University” that takes grades as input as display their remarks as per university criteria.**

*A= You have shown Excellent performance*

*B= Your performance is average.*

*C= You are below average. Needs improvement!!*

*D= You need serious hard work!!*

*E= Your are failed*

**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int num, num1,num2,result;

char menu;

cout<<"\n\t MENU";

cout<<"\n\t\*\*\*\*\*\*\*\*\*\*";

cout<<"\n Select the action from menu:";

cout<<" \n 1.Even or Odd \n 2.Addition \n 3.Subtration \n 4.Division \n 5.Multiplication \n 6.Quit\n\n ";

cin>>menu;

cout<<"\n You entered "<<menu;

switch(menu)

{

case '1':

cout<<"\n Your Selection is EVEN or ODD ";

cout<<"\n Enter a number:";

cin>>num;

if(num%2==0)

cout<<" The number is EVEN";

else

cout<<" The number is ODD";

break;

case '2':

cout<<"\n Your Selection is ADDITION";

cout<<"\n Enter first number: ";

cin>> num1;

cout<<" Enter second number: ";

cin>> num2;

result=num1+num2;

cout<<" The Addition of "<<num1<<" + "<<num2<<" = "<<result;

break;

case '3':

cout<<"\n your selection is SUBTRACTION\n";

cout<<" Enter first number: ";

cin>>num1;

cout<<" Enter second number: ";

cin>>num2;

result=num1-num2;

cout<<" The Subtration of "<<num1<<" - "<<num2<<" = "<<result;

break;

case '4':

cout<<"\n your selection is DIVISION";

cout<<"\n Enter first number: ";

cin>>num1;

cout<<" Enter second number: ";

cin>>num2;

result=num1/num2;

cout<<" The Division of "<<num1<<" / "<<num2<<" = "<<result;

break;

case '5':

cout<<"\n your selection is MULTIPLICATION\n";

cout<<" Enter first number: ";

cin>>num1;

cout<<" Enter second number: ";

cin>>num2;

result=num2\*num1;

cout<<" The Multiplication of "<<num1<<" \* "<<num2<<" = "<<result;

break;

default:

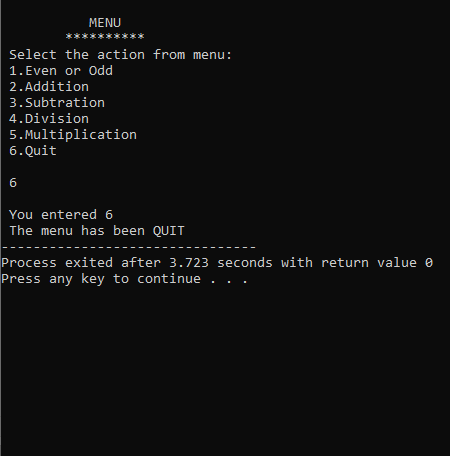
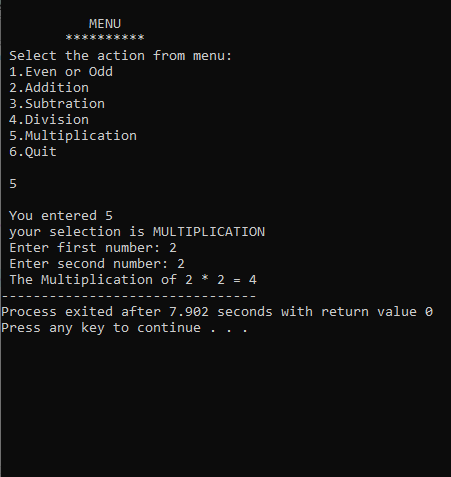
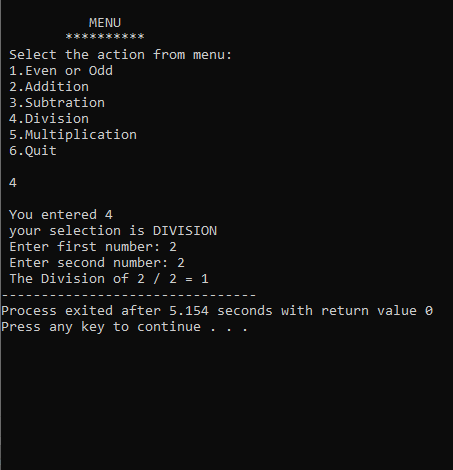
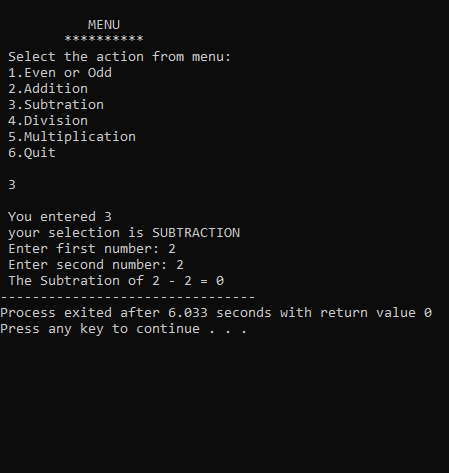
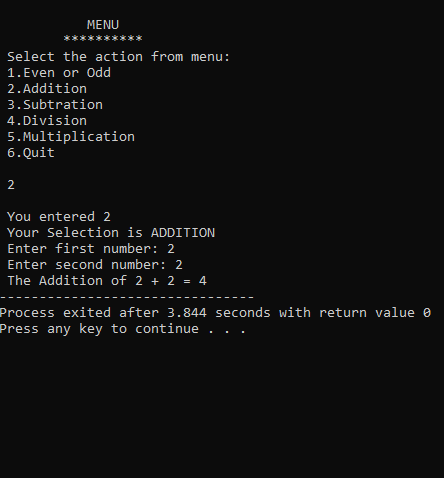
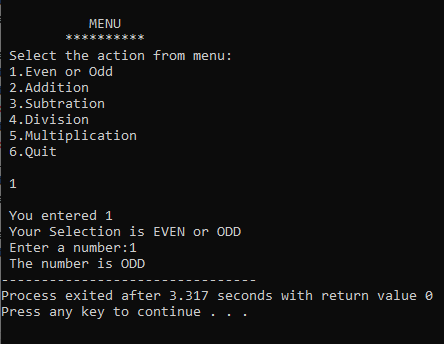
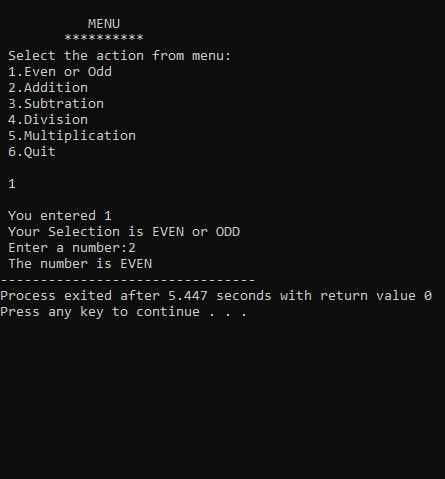
cout<<"\n The menu has been QUIT";

}

Return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

float p;

char menu;

cout<<"\n\t Admissions in IQRA UNIVERSITY";

cout<<"\n\t-------------------------------";

cout<<"\n Choose the department from the menu:";

cout<<" \n 1.Computer Science \n 2.Software Engineering \n 3.Computer Engineering \n 4.Business Management \n 5.Pharmacy\n\n ";

cin>>menu;

switch(menu)

{

case '1':

cout<<"\n You have select the department of COMPUTER SCIENCE ";

cout<<"\n Enter your percentage here:";

cin>>p;

if(p<85 && p>=82)

cout<<" Your admission is confirmed";

else

cout<<" Change your field";

break;

case '2':

cout<<"\n You have select the department of SOFTWARE ENGINEERING ";

cout<<"\n Enter your percentage here:";

cin>>p;

if(p<82 && p>=79)

cout<<" Your admission is confirmed";

else

cout<<" Change your field";

break;

case '3':

cout<<"\n You have select the department of COMPUTER ENGINEERING ";

cout<<"\n Enter your percentage here:";

cin>>p;

if(p>=85)

cout<<" Your admission is confirmed";

else

cout<<" Change your field";

break;

case '4':

cout<<"\n You have select the department of BUSIESS MANAGEMENT ";

cout<<"\n Enter your percentage here:";

cin>>p;

if(p<86 && p>=76)

cout<<" Your admission is confirmed";

else

cout<<" Change your field";

break;

case '5':

cout<<"\n You have select the department of PHARMACY ";

cout<<"\n Enter your percentage here:";

cin>>p;

if(p<88 && p>=70)

cout<<" Your admission is confirmed";

else

cout<<" Change your field";

break;

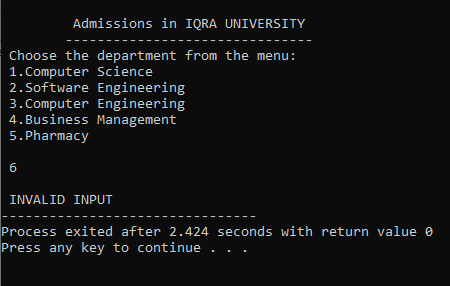
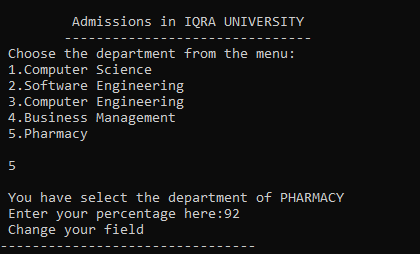
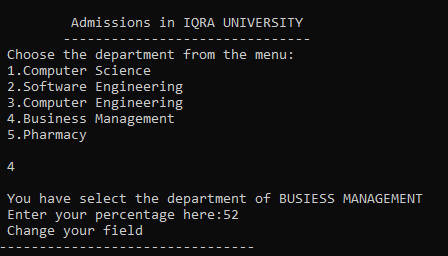
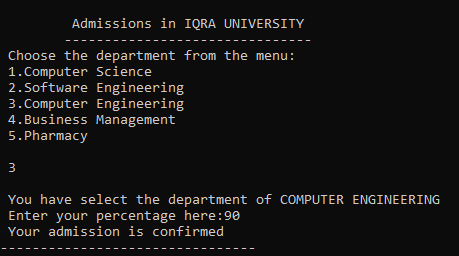
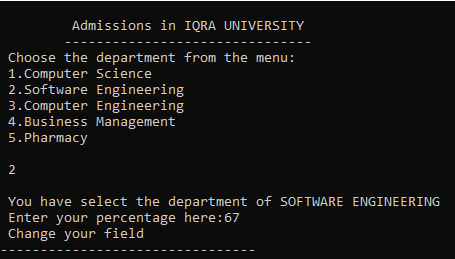
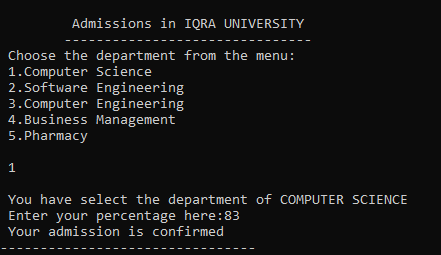
default:

cout<<"\n INVALID INPUT";

}

}

**Output**



**Answer 03**

**Source Code:**

#include<iostream>

using namespace std;

int main ()

{

char menu, grade;

cout << "\n Grading Scheme in IQRA UNIVERSITY" << endl;

cout << " -----------------------------------" << endl << endl;

cout << " Enter grade (A-E) to see the remarks: ";

cin >> menu;

switch(menu)

{

case 'A':

cout << "\n You have shown excellent performance";

break;

case 'B':

cout << "\n Your performance is average.";

break;

case 'C':

cout << "\n You are below average. Needs improvement!!";

break;

case 'D':

cout << "\n You need serious hard work!!";

break;

case 'E':

cout << "\n You are failed";

break;

default:

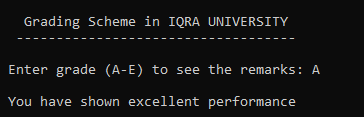
cout << "\n Invalid input";

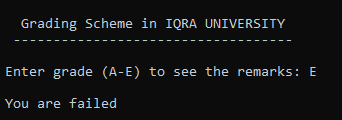
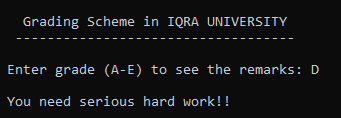
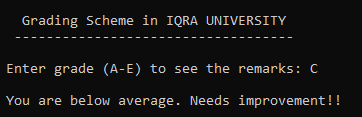
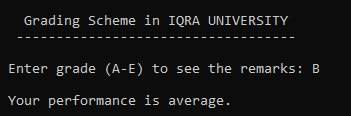
}

return 0;

}

**Output**





**Lab - 05**

**To understand the structure and procedure of Loops in C++**

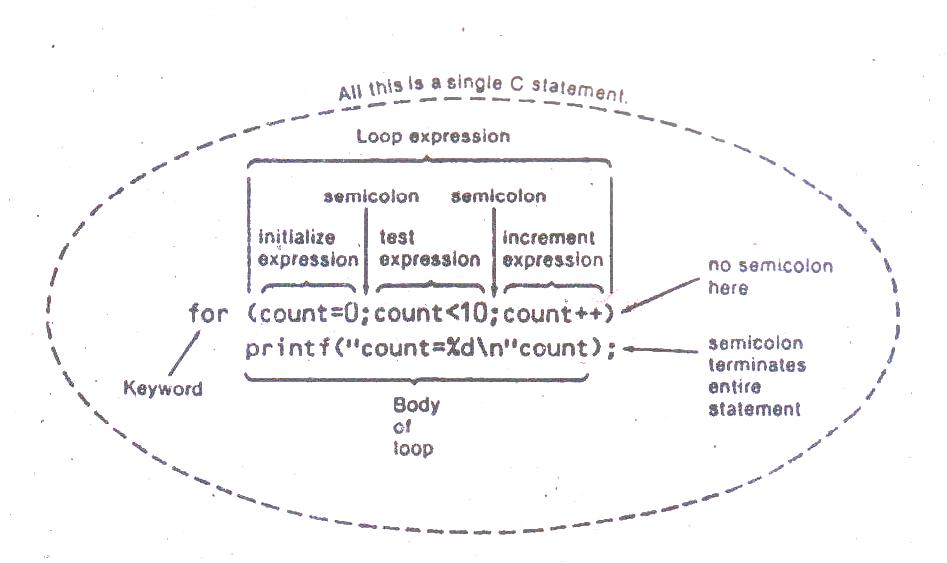
**Theory:**

In computer programming, a loop is a sequence of instruction s that is continually repeated until a certain condition is reached. Typically, a certain process is done, such as getting an item of data and changing it, and then some condition is checked such as whether a counter has reached a prescribed number or not.

1. **The For Loop:**

The FOR loop is found in many procedural languages which repeatedly executes some instructions until given condition is true. In C, the FOR loop is written in the form of expression in which initialization, testing of condition and increment or decrement value is include.

* **Structure of the For Loop:**

****

* **Initalization Expression:**

In initializing of expression, the given variables initialize the starting value. It always executed as soon as the loop is entered. We can start initialization at any number however in the given example variable count initializing or count from 0.

**Example:**

int var;

( var = 5; )

* **Test Expression:**

In secondpart of the expression variable testing the given condition. In given example count is less than 10, it test each time through the loop. If the test expression is true i.e. count<10 the body of the loop will be go to prntf statement and if the expression is false i.e. count is equal to 10 or more, the loop will be terminated.

**Example:**

int var;

( var >= 1;)

* **Increment / Decrement Expression:**

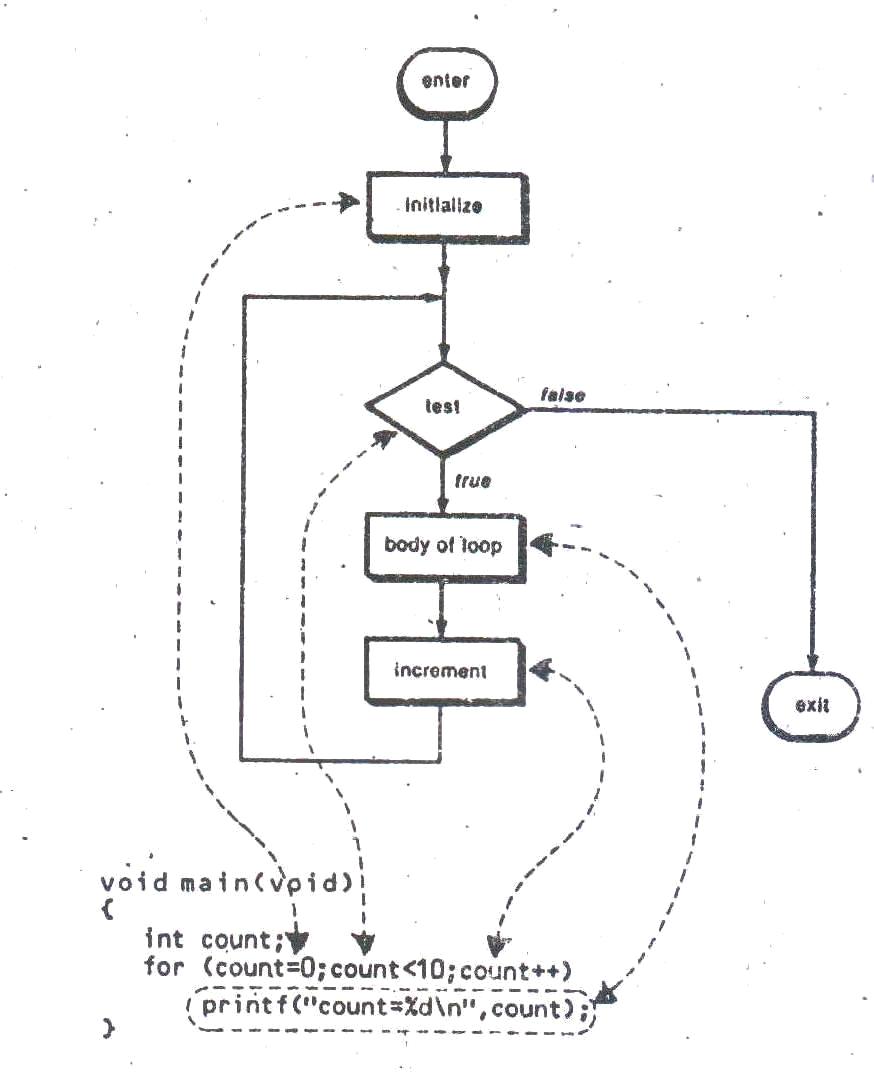
In third part of the expression we use increment or decrement operators to make continuous counter. In this example we use count ++ i.e. increment operator for increment in each step, the variable count each time the loop and then execute.

**Example:**

int var;

( var++ ) / ( var - - )

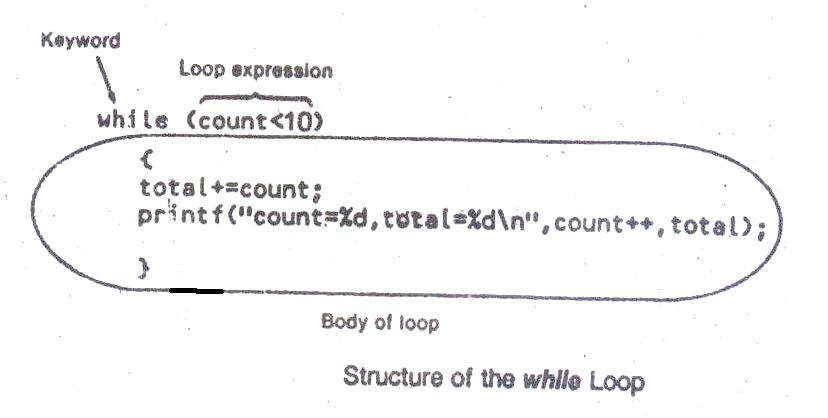
* **Operational Flow Chart of For Loop:**



1. **The While Loop:**

The while loop can be thought of as a repeating [if-statement](http://en.wikipedia.org/wiki/Conditional_%28programming%29). The while construct consists of a block of code and a condition. The condition is evaluated, and if the condition is true, the code within the block is executed. This repeats until the condition becomes false. Because while loop checks the condition before the block is executed, the control structure is often also known as a pre-test loop. The following flow chart illustrates the while loop in C.

* **Structure of While Loop**



* **Test Expression:**

The loop executes as long as the given logical expression between parentheses after whileis true. When expression is false, execution continues with the statement following the loop block.

**while (expression)**

**{**

**// statements**

**}**

**Example:**

#include <iostream>

#include<conio.h>

using namespace std;

int main()

{ int x = 10;

int i = 0; // using while loop statement

while(i < x){

i++;

cout<<i<<endl; //printf("%d\n",i);

} // when number 5 found, escape loop body

int numberFound= 5;

int j = 1;

while(j < x){

if(j == numberFound){

cout<<"number found"<<endl; //printf("number found\n");

}

cout<<j<< "..keep finding" <<endl; //printf("%d...keep finding\n",j);

j++;

}

getch();

}

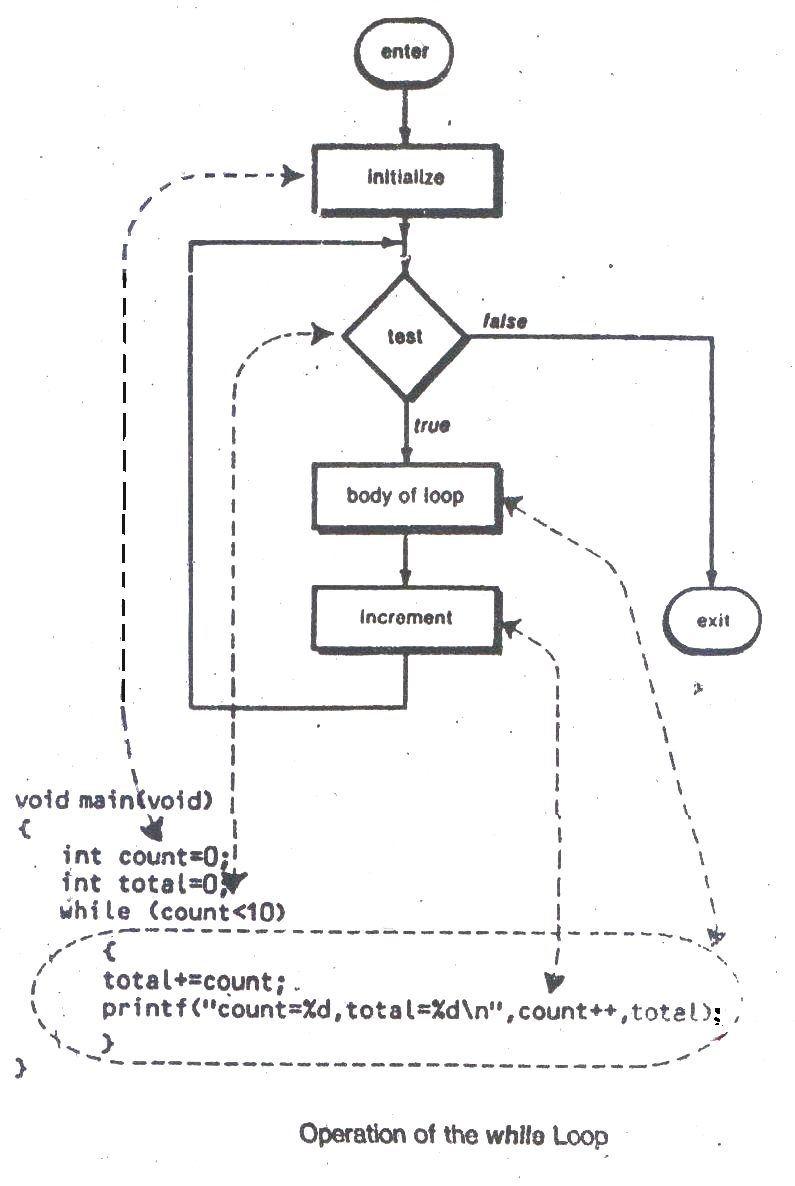
**Output:**

1  
2  
3  
4

5  
6

7  
8  
9  
10  
1...keep finding  
2...keep finding  
3...keep finding  
4...keep finding  
number found.

* **Operational Flow Chart:**

****

**Lab Tasks**

1. Write a program by using for loop which calculate square of first 30 natural numbers?
2. Write a program to print your name 20 times by using for loop?
3. Write a program to generate table of your required number by using while loop and for loop?
4. Write a program to calculate cube of a number and repeat the program until user requirement by using while loop?
5. Design a program to print

* n = 1\*2\*3\*...\*n Take integer input and generate product up to that input number.
* **n= 1+2+3+....+n.** Take integer input and generate sum up to that input number.

1. Design a program to print small and capital alphabets by assigning integer values and printing its character equivalent values.

**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int s;

for (int i = 1; i <= 30; i++)

{

s=i\*i;

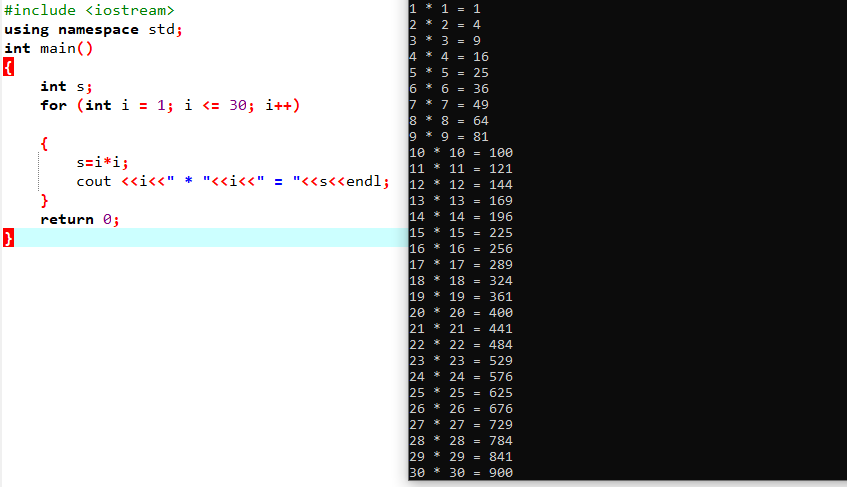
cout <<i<<" \* "<<i<<" = "<<s<<endl;

}

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

for (int i = 1; i <= 20; i++)

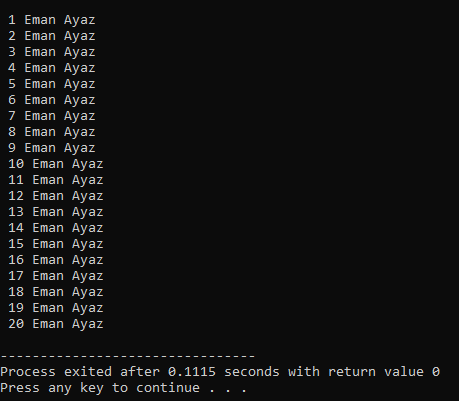
{

cout <<i<< "Eman Ayaz"<< endl;

}

return 0; }

**Output**



**Answer 03**

**Source Code:**

**USING FOR LOOP**

#include <iostream>

using namespace std;

int main()

{

int i, num;

cout << "Enter any number: ";

cin >> num;

for(i=1; i<=10; i++)

{

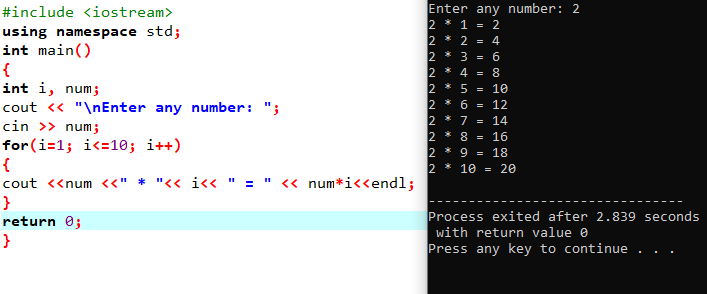
cout <<num <<" \* "<< i<< "=" << num\*i<<endl;

}

return 0;

}

**Output**



**USING WHILE LOOP**

#include <iostream>

using namespace std;

int main()

{

int i=1, num;

cout << "\nEnter any number: ";

cin >> num;

while(i<=10)

{

cout <<num <<" \* "<< i<< " = " << num\*i<<endl;

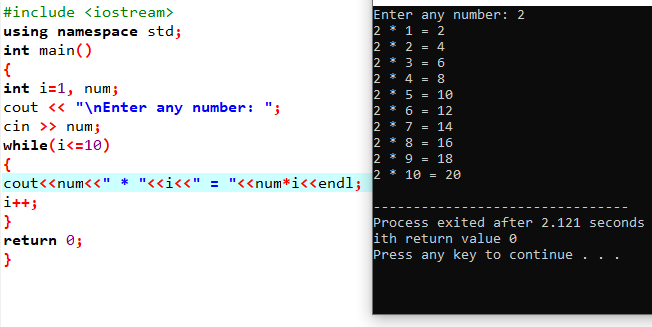
i++;

}

return 0;

}

**Output**



**Answer 04**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{ char yn;

int num, r;

do

{ cout << "Enter a number:\n";

cin >> num;

cout << "The cube of " << num << " is " << num \* num \* num << endl;

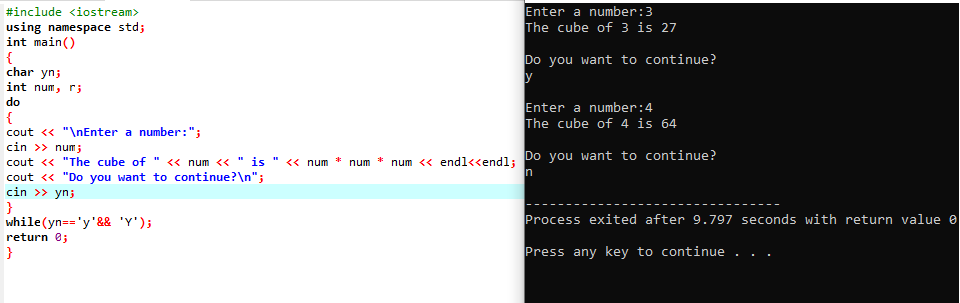
cout << "Do you want to continue?\n";

cin >> yn; }

while(yn=='y'&& 'Y');

return 0; }

**Output**



**Answer 05**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int i, r, pr=1;

cout <<"\n Find the product of the series 1\*2\*3\*...\*n\n";

cout << " --------------------------------------------\n";

cout << " Input the value for nth term: ";

cin>>r;

for(i=1;i<=r;i++)

{

pr\*=i;

}

cout<< " The product upto entered number is "<<pr<<endl;

cout <<"\n Find the sum of the series 1+2+3+...+n\n";

cout << " -----------------------------------------\n";

cout << " Input the value for nth term: ";

cin>>r;

int s=0;

for(i=1;i<=r;i++)

{

s+=i;

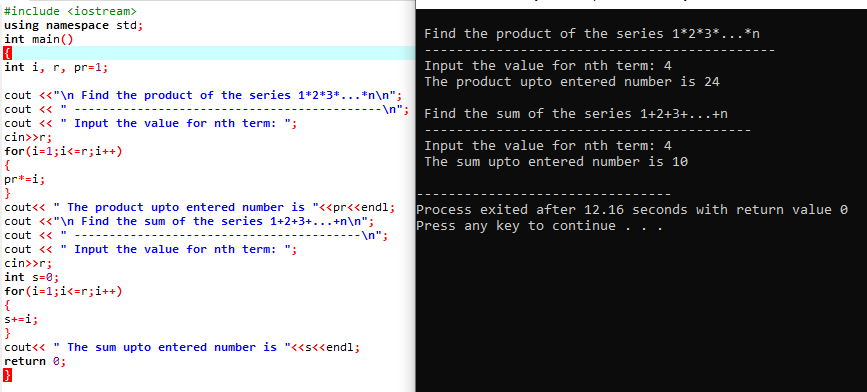
}

cout<< " The sum upto entered number is "<<s<<endl;

return 0;

}

**Output**



**Answer 06**

**Source Code:**

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

char ch;

cout<<"Uppercase Alphabets are: \n";

for(ch=65; ch<=90; ch++){

cout<<ch<<" ";

}

cout<<"\n";

cout<<"\nLowercase Alphabets are: \n";

for(ch=97; ch<=122; ch++){

cout<<ch<<" ";

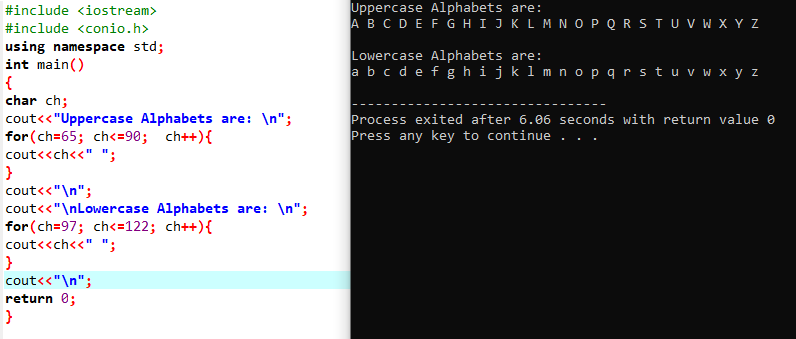
}

cout<<"\n";

return 0;

}

**Output**



**Lab-06**

***To develop understanding of nested For loop phenomenon.***

**Theory:**

Nested loop uses for loop structure within a for loop structure.

The syntax for a **nested for loop** statement in C++ is as follows −

**for ( init; condition; increment )**

**{**

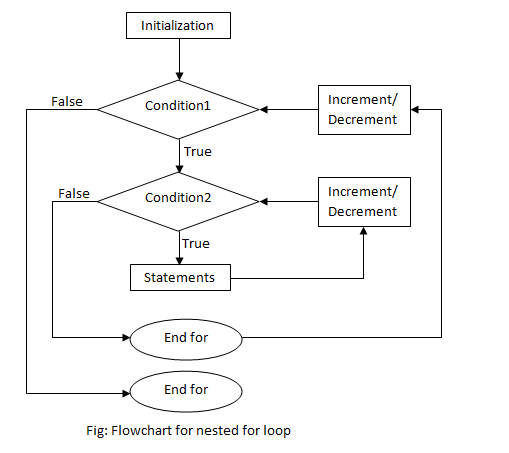
**for ( init; condition; increment )  
 {**

**statement(s);**

**}**

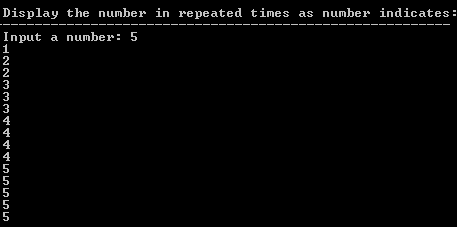
**statement(s); // you can put more statements.**

**}**



**Example 1: To Display the number in repeated times as number entered**

**Output:**



**Code:**

#include <iostream>

using namespace std;

int main() {

int num, r, sum = 0, i;

cout << "\n\n Display the number in repeated times as number indicates:\n";

cout << "---------------------------------------------------------\n";

cout << " Input a number: ";

cin >> num;

for (i = 1; i<=num; i++)

{ for (int j=1; j<=i; j++)

{ cout <<i <<endl;

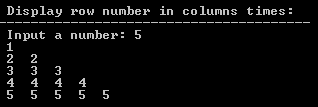
}

}

**}**

**Example 2: To print row number in columns times**

**Output:**



**Code:**

#include <iostream>

using namespace std;

int main()

{ int num, r, sum = 0, i;

cout << "\n\n Display row number in columns times:\n";

cout << "-----------------------------------------\n";

cout << " Input a number: ";

cin >> num;

for (i = 1; i<=num; i++)

{ for (int j=1; j<=i; j++)

{ cout << i <<" ";

}

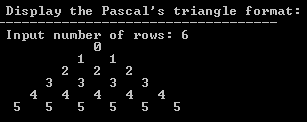
cout <<endl ;

}

}

**Example 3: Number of rows in the form of Pascal’s Triangle Format :**

**Output:**



**Code:**

#include <iostream>

using namespace std;

int main()

{ int row,s,i,j;

cout << "\n\n Display the Pascal's triangle format:\n";

cout << "-----------------------------------\n";

cout << " Input number of rows: ";

cin >> row;

for(i=0;i<row;i++)

{

for(s=1;s<=row-i;s++)

cout<<" ";

for(j=0;j<=i;j++)

{ cout<<i<<" "; }

cout<<endl;

}

}

**Example 4: The following program uses a nested for loop to find the prime numbers from 2 to 100:**

**Code:**

#include <stdio.h>

int main ()

{

/\* local variable definition \*/

int i, j;

for(i=2; i<100; i++) {

for(j=2; j <= (i/j); j++)

if(!(i%j)) break; // if factor found, not prime

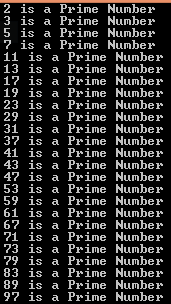
if(j > (i/j))

cout<<i<<" is prime";

}

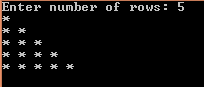
}

**Output:**

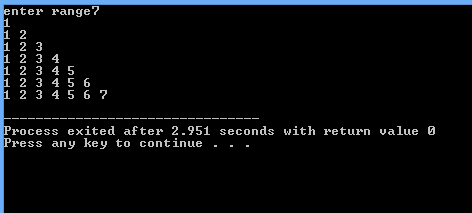


**Lab Tasks**

**1. Design a program, that print the pattern**

****

**2. Design a program, print FLOYD’s Triangle in following ways**



**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

int main() {

int rows;

cout << "Enter number of rows: ";

cin >> rows;

for(int i = 1; i <= rows; ++i) {

for(int j = 1; j <= i; ++j) {

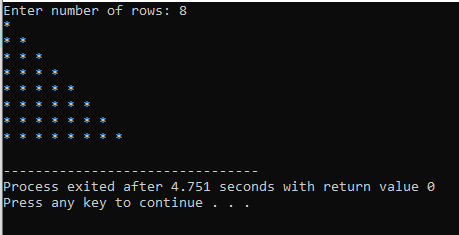
cout << "\* "; }

cout << "\n"; }

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int rows;

cout << "Enter number of rows: ";

cin >> rows;

for(int i = 1; i <= rows; ++i)

{

for(int j = 1; j <= i; ++j)

{

cout << j << " ";

}

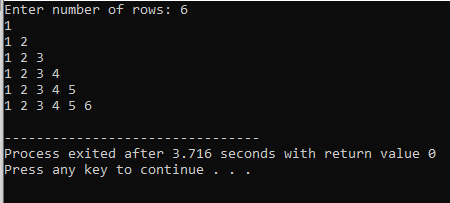
cout << "\n";

}

return 0;

}

**Output**



**Lab-07**

***The purpose of this lab is to get familiar with the concept of numbers and arrays***

## Theory:

* **Numbers:**

Normally, when we work with Numbers, we use primitive data types such as int, short, long, float and double, etc. The number data types, their possible values and number ranges have been explained while discussing C++ Data Types.

**Defining Numbers in C++**

You have already defined numbers in various examples given in previous chapters. Here is another consolidated example to define various types of numbers in C++ −

**Code:**

#include <iostream>

using namespace std;

int main () {

// number definition:

short s;

int i;

long l;

float f;

double d;

// number assignments;

s = 10;

i = 1000;

l = 1000000;

f = 230.47;

d = 30949.374;

// number printing;

cout << "short s :" << s << endl;

cout << "int i :" << i << endl;

cout << "long l :" << l << endl;

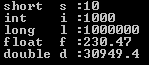
cout << "float f :" << f << endl;

cout << "double d :" << d << endl;

return 0;

**}**

**Output:**



**Math Operations in C++**

In addition to the various functions you can create, C++ also includes some useful functions you can use. These functions are available in standard C and C++ libraries and called **built-in** functions. These are functions that can be included in your program and then use.

|  |  |
| --- | --- |
| **Sr.No** | **Function & Purpose** |
| 1 | **double cos(double);**  This function takes an angle (as a double) and returns the cosine. |
| 2 | **double sin(double);**  This function takes an angle (as a double) and returns the sine. |
| 3 | **double tan(double);**  This function takes an angle (as a double) and returns the tangent. |
| 4 | **double log(double);**  This function takes a number and returns the natural log of that number. |
| 5 | **double pow(double, double);**  The first is a number you wish to raise and the second is the power you wish to raise it t |
| 6 | **double hypot(double, double);**  If you pass this function the length of two sides of a right triangle, it will return you the length of the hypotenuse. |
| 7 | **double sqrt(double);**  You pass this function a number and it gives you the square root. |
| 8 | **int abs(int);**  This function returns the absolute value of an integer that is passed to it. |
| 9 | **double fabs(double);**  This function returns the absolute value of any decimal number passed to it. |
| 10 | **double floor(double);**  Finds the integer which is less than or equal to the argument passed to it. |

C++ has a rich set of mathematical operations, which can be performed on various numbers. Following table lists down some useful built-in mathematical functions available in C++.

To utilize these functions you need to include the math header file **<cmath>**.

Following is a simple example to show few of the mathematical operations –

**Code:**

#include <iostream>

#include <cmath>

using namespace std;

int main () {

// number definition:

short s = 10;

int i = -1000;

long l = 100000;

float f = 230.47;

double d = 200.374;

// mathematical operations;

cout << "sin(d) :" << sin(d) << endl;

cout << "abs(i) :" << abs(i) << endl;

cout << "floor(d) :" << floor(d) << endl;

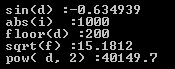
cout << "sqrt(f) :" << sqrt(f) << endl;

cout << "pow( d, 2) :" << pow(d, 2) << endl;

return 0;

}

**Output:**



**Random Numbers in C++**

There are many cases where you will wish to generate a random number. There are actually two functions you will need to know about random number generation. The first is **rand()**, this function will only return a pseudo random number. The way to fix this is to first call the **srand()** function.

Following is a simple example to generate few random numbers. This example makes use of **time()** function to get the number of seconds on your system time, to randomly seed the rand() function –

**Code:**

#include <iostream>

#include <ctime>

#include <cstdlib>

using namespace std;

int main () {

int i,j;

// set the seed

srand( (unsigned)time( NULL ) );

/\* generate 10 random numbers. \*/

for( i = 0; i < 10; i++ ) {

// generate actual random number

j = rand();

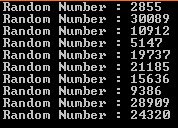
cout <<" Random Number : " << j << endl;

}

return 0;

}

**Output:**



* **Arrays**

C++ provides a data structure, **the array**, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

**Declaring Arrays**

To declare an array in C++, the programmer specifies the type of the elements and the number of elements required by an array as follows −

**type arrayName [ arraySize ];**

This is called a single-dimension array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C++ data type. For example, to declare a 10-element array called balance of type double, use this statement −

**double balance[10];**

**Initializing Arrays**

You can initialize C++ array elements either one by one or using a single statement as follows −

**double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};**

The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ]. Following is an example to assign a single element of the array −

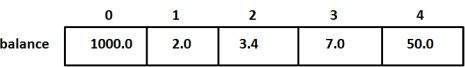
If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

**double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};**

You will create exactly the same array as you did in the previous example.

**balance[4] = 50.0;**

The above statement assigns element number 5th in the array a value of 50.0. Array with 4th index will be 5th, i.e., last element because all arrays have 0 as the index of their first element which is also called base index. Following is the pictorial representaion of the same array we discussed above −



**Accessing Array Elements**

An element is accessed by indexing the array name. This is done by placing the index of the element within square brackets after the name of the array. For example −

**double salary = balance[9];**

The above statement will take 10th element from the array and assign the value to salary variable.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

double temp[7],avg=0,sum=0;

for (int a=0;a<=6;a++)

{

cin>>temp[a];

sum = sum + temp[a];

}

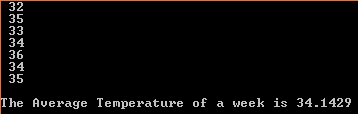
avg= sum/7;

cout<<"The Average Temperature of a week is "<<avg;

return 0;

}

**Output:**



**Lab Tasks**

1. Write a Program to Sort Array Elements.
2. Write a program, to calculate the average of marks of 10 students.
3. Write a program to find the maximum value in array.

**Answer 01**

**Source Code:**

#include<iostream>

using namespace std;

void selectionSort(int a[], int n) {

int i, j, min, temp;

for (i = 0; i < n - 1; i++) {

min = i;

for (j = i + 1; j < n; j++)

if (a[j] < a[min])

min = j;

temp = a[i];

a[i] = a[min];

a[min] = temp;

}

}

int main() {

int a[] = { 22, 91, 35, 78, 10, 8, 75, 99, 1, 67 };

int n = sizeof(a)/ sizeof(a[0]);

int i;

cout<<"Given array is:"<<endl;

for (i = 0; i < n; i++)

cout<< a[i] <<" ";

cout<<endl;

selectionSort(a, n);

printf("\nSorted array is: \n");

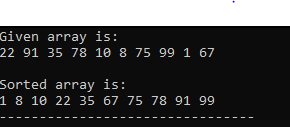
for (i = 0; i < n; i++)

cout<< a[i] <<" ";

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include <iostream>

using namespace std;

int main()

{

int i;

float num[9],sum=0.0, average;

cout << "Enter the marks of 10 students: ";

cin >> num[9];

for(i = 0; i < 10 ; ++i)

{

cout << i + 1 << ". Enter marks:";

cin >> num[i];

sum += num[i];

}

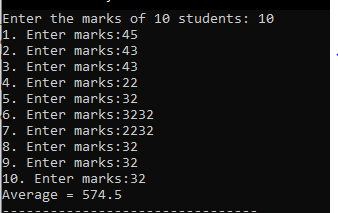
average = sum /10;

cout << "Average = " << average;

return 0;

}

**Output**



**Answer 03**

**Source Code:**

#include <iostream>

using namespace std;

int main() {

int i, n;

float arr[100];

cout << "Enter total number of elements(1 to 100): ";

cin >> n;

cout << endl;

for(i = 0; i < n; ++i) {

cout << "Enter Number " << i + 1 << " : ";

cin >> arr[i];

}

for(i = 1;i < n; ++i) {

if(arr[0] < arr[i])

arr[0] = arr[i];

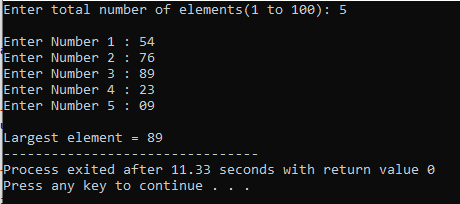
}

cout << endl << "Largest element = " << arr[0];

return 0;

}

**Output**



**Lab-08**

***The purpose of this lab is to get a familiar with the concept and description multidimensional arrays***

## Theory:

Arrays are important to C++ and should need lots of more detail. There are following few important concepts, which should be clear to a C++ programmer

C++ allows multidimensional arrays. Here is the general form of a multidimensional array declaration −

**type name[size1][size2]...[sizeN];**

For example, the following declaration creates a three dimensional 5 . 10 . 4 integer array :

**int threedim[5][10][4];**

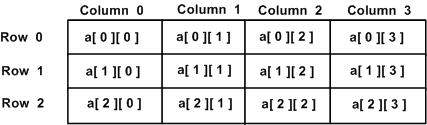
## Two-Dimensional Arrays

The simplest form of the multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size x,y, you would write something as follows −

**type arrayName [ x ][ y ];**

Where **type** can be any valid C++ data type and **arrayName** will be a valid C++ identifier.

A two-dimensional array can be think as a table, which will have x number of rows and y number of columns. A 2-dimensional array **a**, which contains three rows and four columns can be shown as below −



Thus, every element in array a is identified by an element name of the form **a[ i ][ j ]**, where a is the name of the array, and i and j are the subscripts that uniquely identify each element in a.

## Initializing Two-Dimensional Arrays

Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row have 4 columns.

**int a[3][4] = {**

**{0, 1, 2, 3} , /\* initializers for row indexed by 0 \*/**

**{4, 5, 6, 7} , /\* initializers for row indexed by 1 \*/**

**{8, 9, 10, 11} /\* initializers for row indexed by 2 \*/**

**};**

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to previous example −

**int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};**

## Accessing Two-Dimensional Array Elements

An element in 2-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example −

**int val = a[2][3];**

The above statement will take 4th element from the 3rd row of the array. You can verify it in the above diagram.

**Code:**

**#include <iostream>**

**using namespace std;**

**int main () {**

**// an array with 5 rows and 2 columns.**

**int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};**

**// output each array element's value**

**for ( int i = 0; i < 5; i++ )**

**for ( int j = 0; j < 2; j++ ) {**

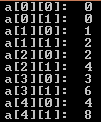
**cout << " a[" << i << "][" << j << "]: ";**

**cout << " "<<a[i][j]<< endl;**

**}**

**return 0; }**

**Output:**



**Lab Tasks**

1. Write a program to multiply two matrices using arrays.
2. Write a program that will add the two matrix of same order.

**Answer 01**

**Source Code:**

#include<iostream>

using namespace std;

int main(){

int first [3][3];

int second [3][3];

int result [3][3];

cout<<endl<<"[Input Matrix a]\n"; "\n\n";

for(int i=0;i<3;++i) {

for(int j=0; j<3;++j)

{

cout<<"Enter a "<<i+1<<j+1<<":\t";

cin>>first[i][j];

}

}

cout<<endl<< "[Input Matrix b]\n";"\n\n";

for(int i=0;i<3;++i){

for(int j=0;j<3;j++)

{

cout<<"Enter b "<<i+1<<j+1<<": \t";

cin>>second[i][j];

}

}

cout<<endl<< "[Input Matrix a and b]\n";"\n\n";

for(int i=0;i<3;++i){

for(int j=0;j<3;j++)

{

cout<<first[i][j];

}

cout<<"\t";

for(int j=0; j<3; ++j)

{

cout<<second[i][j]<<" ";

}

cout<<endl;

}

for(int i=0;i<3;++i){

for(int j=0;j<3;j++){

int sum =0;

for(int k=0;k<3;k++){

sum+=first[i][j]\*second [k][j];

}

result[i][j]=sum;

}

}

cout<<endl<<"[Resultant Matrix]"<< "\n\n";

for(int i=0;i<3;++i){

for(int j=0;j<3;j++){

cout <<" "<<result[i][j];

}

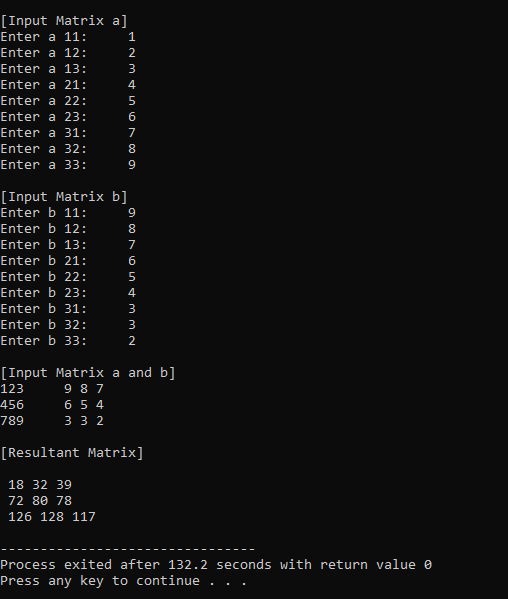
cout<<endl;

}

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include<iostream>

using namespace std;

int main()

{

int mat1[3][3], mat2[3][3], i, j, mat3[3][3];

cout<<"Enter Elements of First Matrix: ";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

cin>>mat1[i][j];

}

cout<<"Enter Elements of Second Matrix: ";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

cin>>mat2[i][j];

}

cout<<"\nAdding the Two Given Matrix...\n";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

mat3[i][j] = mat1[i][j]+mat2[i][j];

}

cout<<"Addition Result of Two Given Matrix is:\n";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

cout<<mat3[i][j]<<" ";

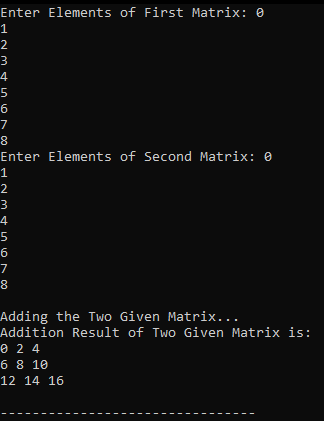
cout<<endl;

}

return 0;

}

**Output**



***Lab-09***

***To be familiarized with Introduction of Pre-define Function (PDF) and User-define Function (UDF)***

## Theory:

A function is a group of statements that together perform a task. Every C++ program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.

You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually is such that each function performs a specific task.

A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.

The C++ standard library provides numerous built-in functions that your program can call. For example, function **strcat()** to concatenate two strings, function **memcpy()** to copy one memory location to another location and many more functions.

A function is known with various names like a method or a sub-routine or a procedure etc.

## Defining a Function

The general form of a C++ function definition is as follows −

**return\_type function\_name( parameter list ) {**

**body of the function**

**}**

A C++ function definition consists of a function header and a function body. Here are all the parts of a function −

* **Return Type** − A function may return a value. The **return\_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return\_type is the keyword **void**.
* **Function Name** − This is the actual name of the function. The function name and the parameter list together constitute the function signature.
* **Parameters** − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.
* **Function Body** − The function body contains a collection of statements that define what the function does.

## Example

Following is the source code for a function called **max()**. This function takes two parameters num1 and num2 and return the biggest of both −

// function returning the max between two numbers

int max(int num1, int num2) {

// local variable declaration

int result;

if (num1 > num2)

result = num1;

else

result = num2;

return result;

}

## Function Declarations

A function **declaration** tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts −

**return\_type function\_name( parameter list );**

For the above defined function max(), following is the function declaration −

**int max(int num1, int num2);**

Parameter names are not important in function declaration only their type is required, so following is also valid declaration −

**int max(int, int);**

Function declaration is required when you define a function in one source file and you call that function in another file. In such case, you should declare the function at the top of the file calling the function.

## Calling a Function

While creating a C++ function, you give a definition of what the function has to do. To use a function, you will have to call or invoke that function.

When a program calls a function, program control is transferred to the called function. A called function performs defined task and when it’s return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value. For example –

**Code:**

**#include <iostream>**

**using namespace std;**

**// function declaration**

**int max(int num1, int num2);**

**int main () {**

**// local variable declaration:**

**int a = 100;**

**int b = 200;**

**int ret;**

**// calling a function to get max value.**

**ret = max(a, b);**

**cout << "Max value is : " << ret << endl;**

**return 0;**

**}**

**// function returning the max between two numbers**

**int max(int num1, int num2) {**

**// local variable declaration**

**int result;**

**if (num1 > num2)**

**result = num1;**

**else**

**result = num2;**

**return result;**

**}**

**Output:**



## Function Arguments

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the **formal parameters** of the function.

The formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit.

While calling a function, there are three ways that arguments can be passed to a function –

* **Call by Value**

This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

* **Call by Pointer**

This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

* **Call by Reference**

This method copies the reference of an argument into the formal parameter. Inside the function, the reference is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

By default, C++ uses **call by value** to pass arguments. In general, this means that code within a function cannot alter the arguments used to call the function and above mentioned example while calling max() function used the same method.

## Default Values for Parameters

When you define a function, you can specify a default value for each of the last parameters. This value will be used if the corresponding argument is left blank when calling to the function.

This is done by using the assignment operator and assigning values for the arguments in the function definition. If a value for that parameter is not passed when the function is called, the default given value is used, but if a value is specified, this default value is ignored and the passed value is used instead. Consider the following example –

**Code:**

**#include <iostream>**

**using namespace std;**

**int sum(int a, int b = 20) {**

**int result;**

**result = a + b;**

**return (result); }**

**int main () {**

**// local variable declaration:**

**int a = 100;**

**int b = 200;**

**int result;**

**// calling a function to add the values.**

**result = sum(a, b);**

**cout << "Total value is :" << result << endl;**

**// calling a function again as follows.**

**result = sum(a);**

**cout << "Total value is :" << result << endl;**

**return 0; }**

**Output:**



**Lab Tasks**

1. Write a program to print the sum of two numbers entered by user by defining your own function.
2. Write a program to print the circumference and area of a circle of radius entered by user by defining your own function.
3. Define two functions to print the maximum and the minimum number respectively among three numbers entered by user.
4. Define a program to find out whether a given number is even or odd.

**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

int addTwo(int x, int y);

int main(){

int a, b, sum;

cout << "Enter the first number: ";

cin >> a;

cout << "Enter the second number: ";

cin >> b;

sum = addTwo(a, b);

cout << "Sum of " << a << " and " << b << " is: " << sum << endl;

return 0;

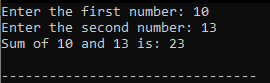
}

int addTwo(int x, int y){

return (x + y);

}

**Output**



**Answer 02**

**Source Code:**

#include<iostream>

using namespace std;

float area(float);

float circum(float);

int main()

{

int radius;

cout<<"\n Enter Radius of Circle: ";

cin>>radius;

cout<<"\n Area of Circle : "<<area(radius);

cout<<"\n Circumference of Circle : "<<circum(radius);

}

float area(float radius)

{

return (3.14 \* radius \* radius);

}

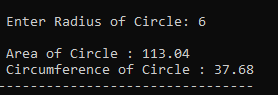
float circum(float radius)

{

return(2 \* 3.14 \* radius);

}

**Output**



A**nswer 03**

**Source Code:**

#include <iostream>

using namespace std;

#define SIZE 50

int main()

{

int array[SIZE];

int i, max, min, size;

cout<<"Enter size of the array: ";

cin>>size;

cout<<"\n Enter "<<size <<" elements in the array: ";

for(i=0; i<size; i++)

cin>>array[i];

max = array[0];

min = array[0];

for(i=1; i<size; i++)

{

if(array[i] > max)

max = array[i];

if(array[i] < min)

min = array[i];

}

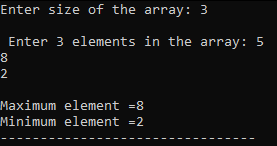
cout<<"\nMaximum element =" << max << "\n";

cout<<"Minimum element =" << min;

return 0;

}

**Output**



**Answer 04**

**Source Code:**

#include <iostream>

#include <conio.h>

using namespace std;

int find\_Oddeven(int);

int main()

{

int num;

cout << "Enter a number to check odd or even" << endl;

cin>>num;

find\_Oddeven(num);

getch();

return 0;

}

int find\_Oddeven(int num){

if(num%2==0)

cout<<num<<"is an even";

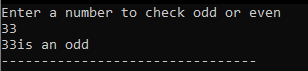
else

cout<<num<<"is an odd";

return 0;

}

**Output**



**Lab-10**

***The purpose of this lab is to get understanding about structures in C++***

## Theory:

C/C++ arrays allow you to define variables that combine several data items of the same kind, but **structure** is another user defined data type which allows you to combine data items of different kinds.

Structures are used to represent a record, suppose you want to keep track of your books in a library. You might want to track the following attributes about each book −

* Title
* Author
* Subject
* Book ID

## Defining a Structure

To define a structure, you must use the struct statement. The struct statement defines a new data type, with more than one member, for your program. The format of the struct statement is this −

**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. Here is the way you would declare the Book structure −

**struct Books {**

**char title[50];**

**char author[50];**

**char subject[100];**

**int book\_id;**

**} book;**

## Accessing Structure Members

To access any member of a structure, we use the **member access operator (.)**. The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use **struct** keyword to define variables of structure type. Following is the example to explain usage of structure −

**Code:**

#include <iostream>

#include <cstring>

using namespace std;

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id;};

int main() {

struct Books Book1; // Declare Book1 of type Book

struct Books Book2; // Declare Book2 of type Book

// book 1 specification

strcpy( Book1.title, "Learn C++ Programming");

strcpy( Book1.author, "Chand Miyan");

strcpy( Book1.subject, "C++ Programming");

Book1.book\_id = 6495407;

// book 2 specification

strcpy( Book2.title, "Telecom Billing");

strcpy( Book2.author, "Yakit Singha");

strcpy( Book2.subject, "Telecom");

Book2.book\_id = 6495700;

// Print Book1 info

cout << "Book 1 title : " << Book1.title <<endl;

cout << "Book 1 author : " << Book1.author <<endl;

cout << "Book 1 subject : " << Book1.subject <<endl;

cout << "Book 1 id : " << Book1.book\_id <<endl;

// Print Book2 info

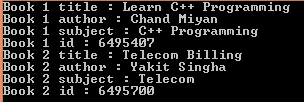
cout << "Book 2 title : " << Book2.title <<endl;

cout << "Book 2 author : " << Book2.author <<endl;

cout << "Book 2 subject : " << Book2.subject <<endl;

cout << "Book 2 id : " << Book2.book\_id <<endl;

return 0;}

**Output:**

## Structures as Function Arguments

You can pass a structure as a function argument in very similar way as you pass any other variable or pointer. You would access structure variables in the similar way as you have accessed in the above example –

**Code:**

#include <iostream>

#include <cstring>

using namespace std;

void printBook( struct Books book );

struct Books {

char title[50];

char author[50];

char subject[100];

int book\_id; };

int main() {

struct Books Book1; // Declare Book1 of type Book

struct Books Book2; // Declare Book2 of type Book

// book 1 specification

strcpy( Book1.title, "Learn C++ Programming");

strcpy( Book1.author, "Chand Miyan");

strcpy( Book1.subject, "C++ Programming");

Book1.book\_id = 6495407;

// book 2 specification

strcpy( Book2.title, "Telecom Billing");

strcpy( Book2.author, "Yakit Singha");

strcpy( Book2.subject, "Telecom");

Book2.book\_id = 6495700;

// Print Book1 info

printBook( Book1 );

// Print Book2 info

printBook( Book2 );

return 0;}

void printBook( struct Books book ) {

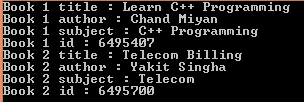
cout << "Book title : " << book.title <<endl;

cout << "Book author : " << book.author <<endl;

cout << "Book subject : " << book.subject <<endl;

cout << "Book id : " << book.book\_id <<endl;}

**Output:**



## Example: C++ Structure

C++ Program to assign data to members of a structure variable and display it.

**#include <iostream>**

**using namespace std;**

**struct Person**

**{**

char name[50];

int age;

float salary;

};

int main()

{

Person p1;

cout << "Enter Full name: ";

cin.get(p1.name, 50);

cout << "Enter age: ";

cin >> p1.age;

cout << "Enter salary: ";

cin >> p1.salary;

cout << "\nDisplaying Information." << endl;

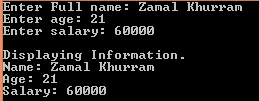
cout << "Name: " << p1.name << endl;

cout <<"Age: " << p1.age << endl;

cout << "Salary: " << p1.salary;

return 0;

}

**Output:**

**Lab Tasks**

1. Use one structure to represent four cars. (Attributes: brand, model, year)
2. Write a program that uses a structure to store Information of a Student and display in the following format.

Name: F.Name: ID:

Semester: Program: Exam Roll No

**Answer 01**

**Source Code:**

#include <iostream>

using namespace std;

struct Cars

{

char brand[50];

char model[10];

int year;

};

int main()

{

struct Cars car1, car2, car3, car4;

cout << "\n\_\_\_\_\_ Car 1 \_\_\_\_\_" << endl;

cout << "Enter brand: ";

cin >> car1.brand, 50;

cout << "Enter model: ";

cin >> car1.model, 10;

cout << "Enter year: ";

cin >> car1.year;

cout << "\n\_\_\_\_\_ Car 2 \_\_\_\_\_" << endl;

cout << "Enter brand: ";

cin >> car2.brand, 50;

cout << "Enter model: ";

cin >> car2.model, 10;

cout << "Enter year: ";

cin >> car2.year;

cout << "\n\_\_\_\_\_ Car 3 \_\_\_\_\_" << endl;

cout << "Enter brand: ";

cin >> car3.brand, 50;

cout << "Enter model: ";

cin >> car3.model, 10;

cout << "Enter year: ";

cin >> car3.year;

cout << "\n\_\_\_\_\_ Car 4 \_\_\_\_\_" << endl;

cout << "Enter brand: ";

cin >> car4.brand, 50;

cout << "Enter model: ";

cin >> car4.model, 10;

cout << "Enter year: ";

cin >> car4.year;

cout << "\n\t\t Displaying Information" << endl;

cout << "----------------------------------------------------------------";

cout << "\n\t\t\tCar 1"<<endl;

cout<< "Brand: " << car1.brand;

cout<<"\t\tModel: " << car1.model;

cout<< "\t\tYear: " << car1.year;

cout << "\n----------------------------------------------------------------";

cout << "\n\t\t\tCar 2";

cout<< "\nBrand: " << car2.brand ;

cout<<"\t\tModel: " << car2.model ;

cout<< "\t\tYear: " << car2.year;

cout << "\n----------------------------------------------------------------";

cout << "\n\t\t\tCar 3";

cout<< "\nBrand: " << car3.brand ;

cout<<"\t\tModel: " << car3.model ;

cout<< "\t\tYear: " << car3.year;

cout << "\n----------------------------------------------------------------";

cout << "\n\t\t\tCar 4";

cout<< "\nBrand: " << car4.brand;

cout<<"\t\tModel: " << car4.model;

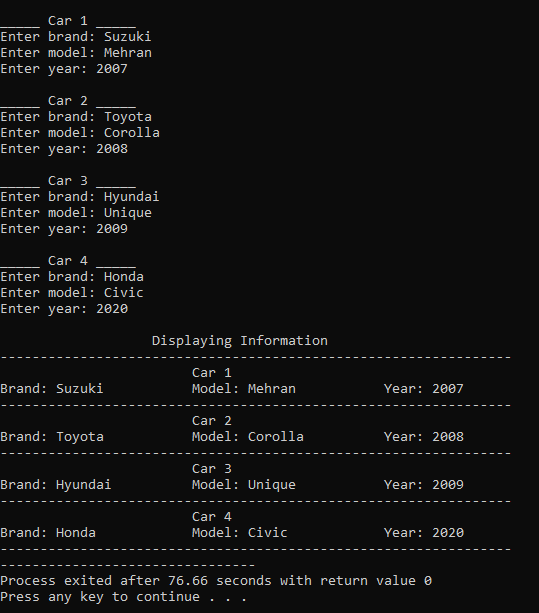
cout<< "\t\tYear: " << car4.year;

cout << "\n----------------------------------------------------------------";

return 0;

}

**Output**



**Answer 02**

**Source Code:**

#include <iostream>

using namespace std;

struct student

{

char name[50],fname[50],prog[50];

int roll,id,sem,ern;

float marks;

};

int main()

{

student s;

cout << "\nEnter Information of a student" << endl<<endl;

cout << "Name: ";

cin >> s.name;

cout << "F.Name: ";

cin >> s.fname;

cout << "ID: ";

cin >> s.id;

cout << "Semester: ";

cin >> s.sem;

cout << "Program: ";

cin >> s.prog;

cout << "Exam Roll no: ";

cin >> s.ern;

cout << "\n-----------------------" << endl;

cout << "Displaying Information" << endl;

cout << "-----------------------" << endl<<endl;

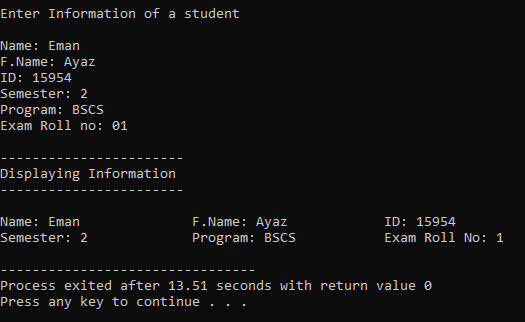
cout << "Name: " << s.name << "\t\tF.Name: " << s.fname << "\t\tID: " << s.id << endl;

cout << "Semester: " << s.sem << "\t\tProgram: " << s.prog << "\t\tExam Roll No: " << s.ern << endl;

return 0;

}

**Output**



**Lab -11**

***To familiarize with the usage of the Filling in C++***

**Theory**

File handling is used for store a data permanently in computer. Using file handling we can store our data in secondary memory (Hard disk).

How to achieve the File Handling

For achieving file handling we need to follow the following steps:-

STEP 1-Naming a file

STEP 2-Opening a file

STEP 3-Writing data into the file

STEP 4-Reading data from the file

STEP 5-Closing a file.

**Streams in C++** :

We give input to the executing program and the execution program gives back the output. The sequence of bytes given as input to the executing program and the sequence of bytes that comes as output from the executing program are called stream. In other words, streams are nothing but the flow of data in a sequence.

The input and output operation between the executing program and the devices like keyboard and monitor are known as “console I/O operation”. The input and output operation between the executing program and files are known as “disk I/O operation”.

**Classes for File stream operations** :

The I/O system of C++ contains a set of classes which define the file handling methods. These include ifstream, ofstream and fstream classes. These classes area derived from fstream and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.

The default mode for opening a file with ofstream's constructor is to create it if it does not exist, or delete everything in it if something does exist in it. If necessary, you can give a second argument that specifies how the file should be handled. They are listed below:  
ios::app -- Opens the file, and allows additions at the end  
ios::ate -- Opens the file, but allows additions anywhere  
ios::trunc -- Deletes everything in the file  
ios::nocreate -- Does not open if the file must be created  
ios::noreplace -- Does not open if the file already exists

**Example: To Create a file and write until “!” is encountered and display the result**

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

ofstream fout; // Creation of ofstream class object

string line;

// by default ios::out mode, automatically deletes

// the content of file. To append the content, open in ios:app

 // fout.open("sample.txt", ios::app)

fout.open("sample.txt");

while (fout) // Execute a loop If file successfully opened

{ getline(cin, line); // Read a Line from standard input

if (line == "!") // Press ! to exit

break;

fout << line << endl; // Write line in file

}

fout.close(); // Close the File

ifstream fin; // Creation of ifstream class object to read the file

fin.open("sample.txt"); // by default open mode = ios::in mode

while (fin) { // Execute a loop until EOF (End of File)

getline(fin, line); // Read a Line from File

cout << line << endl; // Print line in Console

}

fin.close(); // Close the file

return 0;

}

**Lab Task**

**Write a single file handling program in c++ to reading and writing data on a file.**

**Answer 01**

**Source Code:**

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

int rno,fee;

char name[50];

cout<<"Enter the Roll Number:";

cin>>rno;

cout<<"Enter Name:";

cin>>name;

cout<<"Enter the Fee:";

cin>>fee;

ofstream fout("Eman.doc");

fout<<rno<<"\t"<<name<<"\t"<<fee;

fout.close();

ifstream fin("Eman.doc");

fin>>rno>>name>>fee;

fin.close();

cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;

return 0;

}

**Output**

