

**Department of Computer Science**

**Lab Manual**

**CSC103 – Programming Fundamentals**

**BS (CS)**

**Semester: I**

**Section :1C**

**Mr. M. Ameen Khowaja (Course Supervisor)**

**Faraz Ahmed (Lab Instructor)**

**Working:**

**NAME: Id:**

Hafiz Mudassir Husain (CSC-22f-145)

Muhammad Asif (CS-22f- )

Mazhar Ali (CSC-22f- )

Zeeshan Bhutto (CSC-22f- )

**Table of Contents**

# Lab. No. Description Page No.

1. Introduction of Turbo C IDE and Programming Environment 3
2. C Building Blocks 6
3. Looping constructs in C-Language 9
4. Decision making the if-else and else if structure 12
5. Nested loops (for, while, do while) 16
6. Decision making switch and conditional (Ternary) operator 19
7. Functions in C-Language 22
8. Recursion functions in C 25
9. **Arrays** in C (single dimensional, Multidimensional) 29
10. **C** programming string 32
11. **Preprocessor** Directives 35
12. Pointers in C-Language 28
13. Files in C-language 35 **14** Structure in C-language 35

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 01**

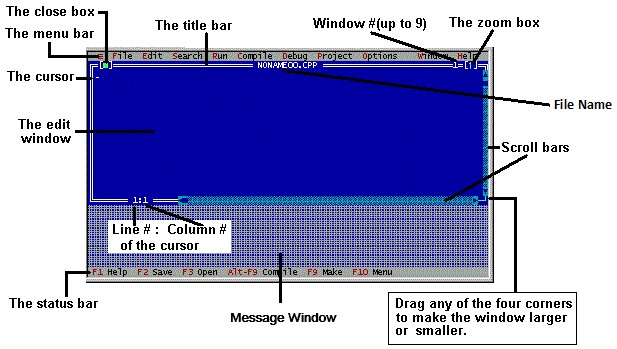
**Objective of Experiment:** Introduction of Turbo C IDE and Programming Environment

**Goals:** Students will be able to understand the Turbo C IDE and Programming Environment.

.

# Required Tools / Equipments:

* PC
* Turbo C (IDE)



# Theory:

**Integrated Development Environment (IDE):**

The Turbo C compiler has its own built-in text editor. The files you create with text 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 Programmer’s Platform, is a screen display with windows and pull-down menus. The program listing, error messages and other information are displayed in separate windows. The menus may be used to invoke all the operations necessary to develop the program, including editing, compiling, linking, and debugging and program execution.

**Default Directory**

The default directory of Turbo C compiler is c:\tc\bin

**Using Menus**

If the menu bar is inactive, it may be invoked by pressing the [**F10**] function key. To select different menu, move the highlight left or right with cursor (**arrow**) keys. You can also revoke the selection by pressing the key combination for the specific menu.

**Opening New Window**

To type a program, you need to open an **Edit** Window. For this, open **File** menu and click

“**New**”. 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 **[F2]** button. 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.

**Making an Executable File**

The Source file is required to be turned into an Executable file. This is called “Making” of the **.exe** file. The steps required to create an executable file are:

1. Create a source file, with a **.c** extension.
2. Compile the source code into a file with the **.obj** extension.
3. Link your .obj file with any needed libraries to produce an executable program.

All the above steps can be done by using **Run** option from the menu bar or using key combination **Ctrl+F9** (By this, linking and compiling is done in one step).

**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 to 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.

This can be done by selecting **Compile** option from menu bar or using key combination **Alt+F9**.

**Creating an Executable file with Linker**

After your source code is compiled, your object file is produced. This file is often name with extension **.OBJ**. This is still not an executable program however. To turn this into an executable program, you must learn 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.

**Compiling and Linking in IDE**

Compiling and linking can be done in one step. There are two ways to do this: you can select **MAKE EXE** from **Compile** menu, or you can press the **[F9]** key.

**Executing a program**

If the program is linked and compiled without errors, the program is executed by selecting **Run** from the **Run** menu or by pressing **Ctrl+F9** key combination.

**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][F3]** combination. To exit from the IDE, select **Exit** from **File** menu or press **[Alt][X]**

Combination.

**Technical Exits**

To minimize the screen of Turbo C editor, press **Alt+Enter**. If some where the program hangs up compiler at output or gets busy without passing control to programmer press **Ctrl+Pause/Break**.

# Student’s tasks:

# Answer the following questions:

**1. Write the following program in C Editor and execute it. Mention the Errors if any.**

#include<stdio.h> #include<conio.h> void main(void)

{ clrscr(); printf(“Hello World”);

getch(); }

**Ans:\_\_\_\_**HELLO WORLD**\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Make the following changes to the program. What Errors are observed?** 
   1. Write Printf instead of printf .
   2. Write void main (void);
   3. Erase any one of brace ‘{’ or ‘}’.
   4. Erase semicolon ‘;‘ after bracket ’)’ of Hello World.
   5. Erase Header Files #include<…> and #include<…>

**ANS:**

1. ERROR **1:** function ‘Printf’ should have a prototype.
2. ERROR2: **void main (void);** Declaration terminated incorrectly.
3. ERROOR3: **Erase brace };** Compound statement is missing }.
4. ERROR 4: **Erase semi-colon(;)** statement missing ; .
5. ERROR 5: **Erase header file(< -- >)** Function ‘clrscr’ should have prototype.

1. **Write a program in c language to print your bio-data on the screen by using printf function. (Name, Roll no, Semester, Batch, Department, University).**

#include<stdio.h>

#include<conio.h>

void main(void)

{

clrscr();

printf("\t My Name is Hafiz Mudassir Husain\n");

printf("\t My roll no is CSC-22F-145\n");

printf("\t I am a student of 1st Semester\n");

printf("\t Batch is Fall-22\n");

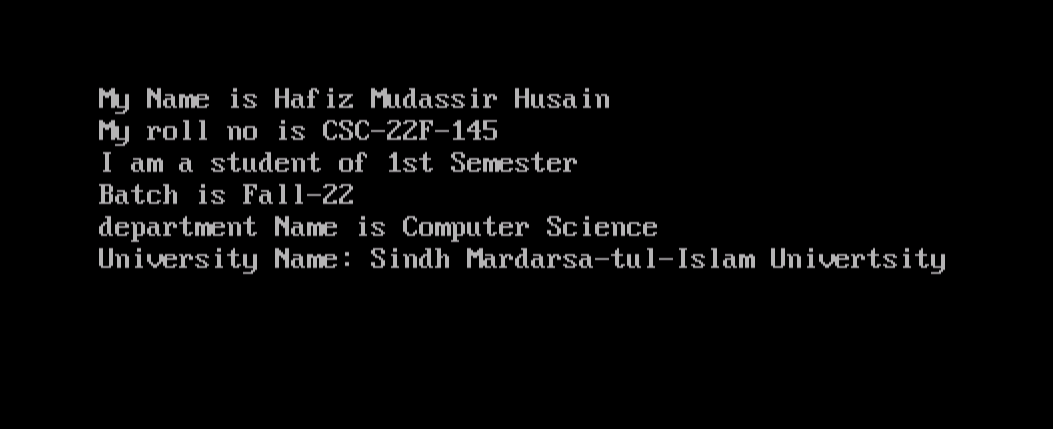
printf("\t department Name is Computer Science\n");

printf("\t University Name: Sindh Mardarsa-tul-Islam Univertsity\n");

getch();

}

**OUTPUT:**



1. **What is wrong in the following program:**

#include<stdio.h>

#include<conio.h> Void main[void];

( clrscr();

printf(Welcome to Computer Lab) getch();

}

**Ans:**

After runing the program we get

#include<stdio.h>

#include<conio.h>

int main(void)

{

printf("Welcome to Computer Lab");

getch();

}

**Error:**

Void main[void]

Semiclon with void.

No inverted comas & Semiclon in **printf** function.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date Lab Instructor Sign. Course Instructor Sign.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_\_\_\_** |  |  |  |  |  |  |  |  | **\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Student ID** |  |  |  |  |  |  |  |  | **Student Name** |

**Laboratory Experiment 02**

# Objective of Experiment: C Building Blocks

# Theory:

In any language there are certain building blocks:

* Constants
* Variables
* Operators
* Methods to get input from user (scanf( ), getch( ) etc.)
* Methods to display output (Format Specifier, Escape Sequences etc.) etc.

**Format Specifier**

Format Specifier tell the printf statement where to put the text and how to display the text. The various format specifiers are:

%d => integer

%c => character

%f => float

%e => displays number in scientific notation (float)

%c =>displays a character

%s =>displays a string

%o =>displays an octal number (unsigned)

%x =>displays a hexadecimal number (unsigned)

**Variables and Constants**

If the value of an item is to 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 ,double etc they are also of the type signed or unsigned.

**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

\t => horizontal tab

\b => back space

\r => carriage return

\’ =>single quote

\” => double quotations

\\ => back slash

\v =>vertical tab

\a =>alert

\? =>Question mark

\0 =>Null

\f =>form feed, page eject, page or section separator (move active position to the initial position of next logical page)

\xhh =>hexa decimal escape sequence

\ddd =>octal escape sequence

**Taking Input from the User**

The input from the user can be taken by the following techniques: scanf( ), getch(), getche( ), getchar( ) etc. **getchar()** is a standard function that gets a character from the stdin. e.g: char c;

printf(“Enter character”);

c= getchar();

**getch()** is a nonstandard function that gets a character from keyboard, does not echo to screen. **getche()** is a nonstandard function that gets a character from the keyboard, echoes to screen.

Use **getchar()** if you want it to work on all compilers. Use **getch()** or **getche()** on a system that supports it when you want keyboard input without pressing [Enter].

**Operators**

There are various types of operators that may be placed in three categories: Basic: + - \* / %

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

Increment / decrement operators ++, -- Relational: < > <= >= == != Logical: && , || , !

# Student’s tasks:

1. Write a following program and observe result:

void main (void)

{

char ch='A',str[]="My SMIU"; int num= 2; float fnum= 12.47;

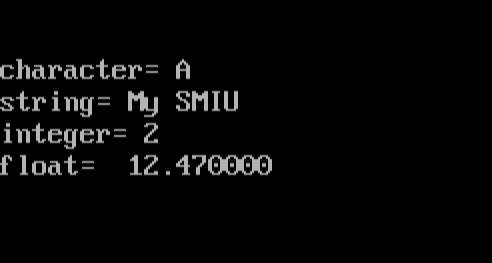
clrscr();

printf("character= %c\n",ch); printf("string= %s\n", str); printf("integer= %d\n", num); printf("float= %10.6f\n", fnum);

getch();

}

**OUTPUT:**



2. Write a program which take input Name, Age, Height, Gender and print them by using Escape Sequences and Format Specifiers.

#include<conio.h>

#include<stdio.h>

void main()

{

clrscr();

char name[20],gender[20];

float height,age;

printf("\n\n\t\t Enter Your name " );

scanf("%s", &name);

printf("\n\n\t\t Enter your gender ");

scanf("%s", &gender);

printf("\n\n\t\t Enter Your Age ");

scanf("%f", &age);

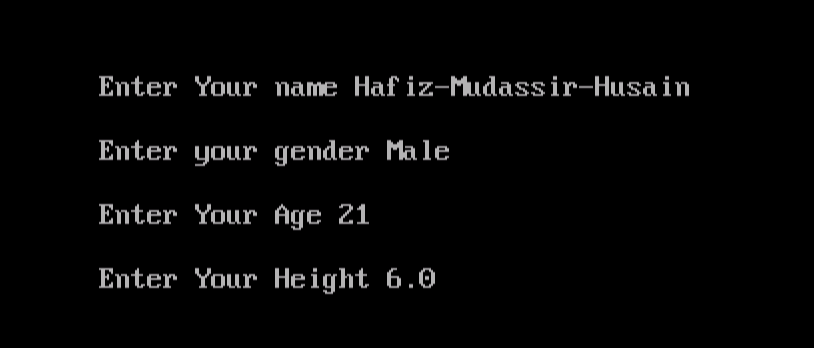
printf("\n\n\t\t Enter Your Height ");

scanf("%f", &height);

getch();

}

**OUTPUT:**



2. Write a following program and observe the result:

#include<stdio.h>

#includ<conio.h>

#include<math.h> #define pi 3.1415927 void main (void)

{

float altitude,base,radius,t,s;

clrscr ();

printf("\n\n\t\t Enter the value for Altitude of a Triangle. "); scanf("%f", &altitude);

printf("\n\n\t\t Enter the value for Base of a Triangle. "); scanf("%f", &base);

printf("\n\n\t\t Enter the value for Radius of a Sphere. "); scanf("%f", &radius); t=(1.0/2.0)\*(altitude\*base); s=(4.0/3.0)\*pi\*pow(radius,3);

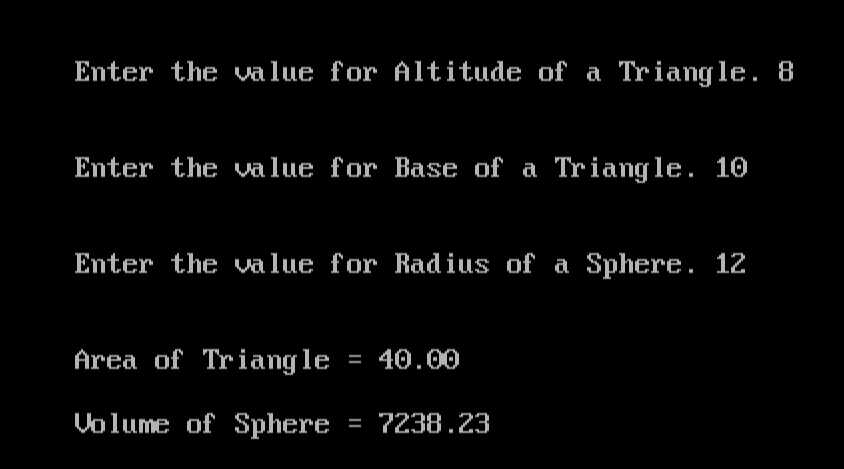
if(t<s) { printf("\n\n\t\t Area of Triangle = %.2f",t); printf("\n\n\t\t Volume of Sphere = %.2f",s);

} else { printf("\n\n\t\t Volume of Sphere = %.2f",s); printf("\n\n\t\t Area of Triangle = %.2f",t);

} getch ();

}

**OUTPUT:**



1. Write C program to perform basic arithmetic operations: addition, subtraction, multiplication and division of two numbers. Numbers should be integers and will be entered by the user.

#include<stdio.h>

#include<conio.h>

int main()

{

clrscr();

int num1,num2,ptr1,ptr2,result;

printf("Enter num1:");

scanf("%d",&num1);

printf("\nEnter num2:");

scanf("%d",&num2);

ptr1=num1;

ptr2=num2;

result=ptr1+ptr2;

printf("\n Addition=%d",result);

result=ptr1-ptr2;

printf("\n Subtraction=%d",result);

result=ptr1\*ptr2;

printf("\n Multiplication=%d",result);

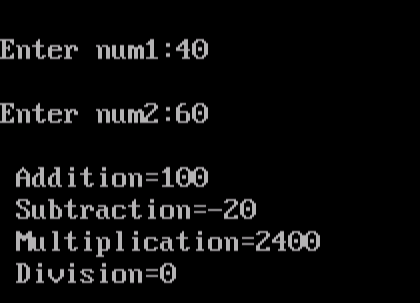
result=ptr1/(ptr2);

printf("\n Division=%d",result);

getch();

}

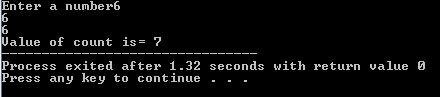
**OUTPUT:**



1. What will be the output if count=5, printf(“%d”,++count); printf(“%d”,count++);

printf(“Value of count is= %d”,count);

**OUTPUT:**

****

\_\_\_**\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date Lab Instructor Sign. Course Instructor Sign**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_\_\_\_** |  |  |  |  |  |  |  |  | **\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Student ID** |  |  |  |  |  |  |  |  | **Student Name** |

**Laboratory Experiment 03**

**Objective of Experiment:** Looping constructs in C-Language

# Theory:

There may be a situation, when you need to execute a block of code several numbers of times. In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages:

* You want to execute some code/s 100 times.
* You want to execute some code/s certain number of times depending upon input from user.

These types of task can be solved in programming using loops.

There are 3 types of loops in C programming:

* For Loop
* While Loop
* Do-While Loop

**For Loop**

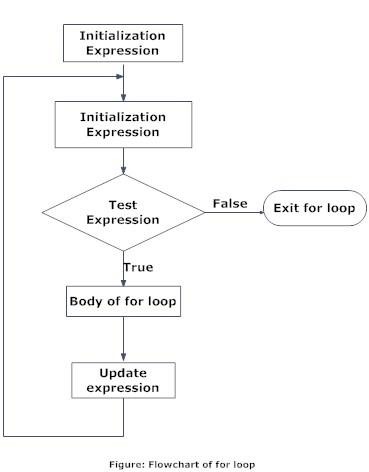
For example: Write a program to find the sum of first n natural numbers where n is entered by user. Note: 1,2,3... are called natural numbers. **The syntax of for loop**

for(initialize expression; test condition expression; increment expression)

{

set of statements

}



**Nested For loop**

for(initialize expression; test condition expression; increment expression)

{

for(initialize expression; test condition expression; increment expression)

{

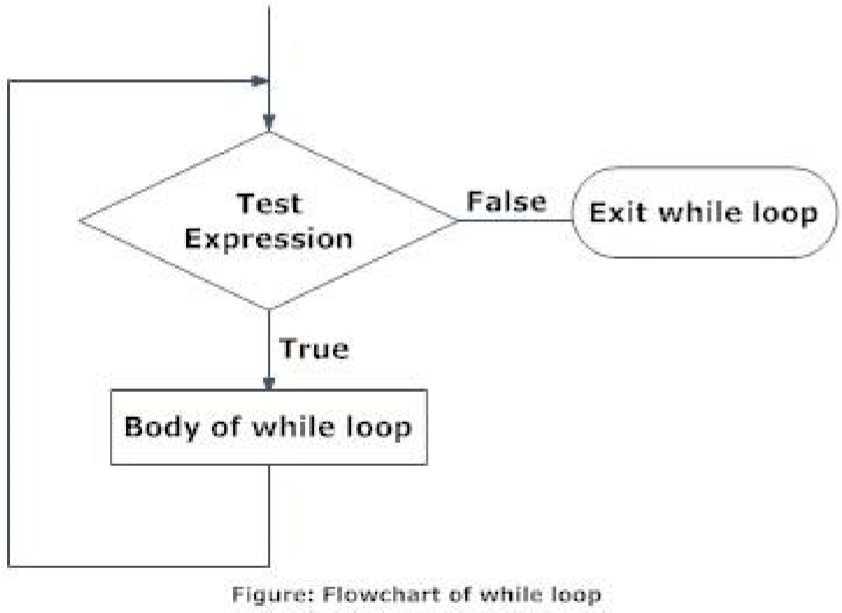
set of statements

}

**}**

**While loop**

The while loop checks whether the test expression is true or not. If it is true, code/s inside the body of while loop is executed, that is, code/s inside the braces { } are executed. Then again the test expression is checked whether test expression is true or not. This process continues until the test expression becomes false.



**Syntax of while loop**

**while** (test expression)

{ statement/s to be executed.

}

**Nested while loop while** (test expression)

**{ while** (test expression)

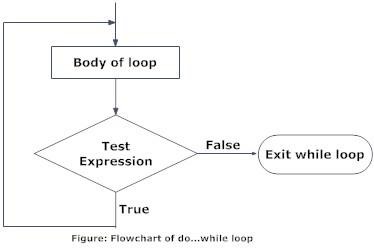
{ statement/s to be executed.

}

}

**Do while loop**

At first codes inside body of **do** is executed. Then, the test expression is checked. If it is true, code/s inside body of do are executed again and the process continues until test expression becomes false(zero).



**Syntax of do while loop**

**do** {

some code/s;

}

**while** (test expression);

**Nested do while loop do**

**{ do** {

some code/s;

}

**while** (test expression);

**}**

**while** (test expression);

**Student’s tasks:**

.

**1. Write a following program to add n number of elements.**

void main(void)

{

int n, count, sum=0; printf("Enter the value of n.\n");

scanf("%d",&n);

for(count=1;count<=n;++count) //for loop terminates if count>n

{

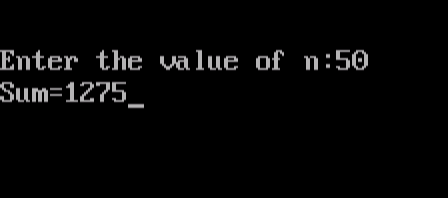
sum+=count; /\* this statement is equivalent to sum=sum+count \*/

}

printf("Sum=%d",sum); getch();

}

**OUTPUT:**



**2. Program generates Multiplication Table of number between 2 to 20.**

void main (void)

{ inti,n,j=0; clrscr ();

printf("\n\t\tEnter Table No. ");

scanf("%d",&n); if(n>=2 && n<=20) for(i=1;i<=12;i++)

{ j=n\*i;

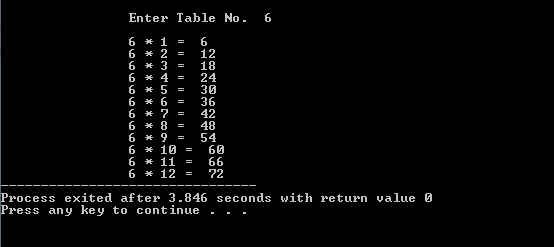
printf("\n\t\t%d \* %d = %d",n,i,j);

} else

printf("\nError!!! Enter number from 2 to 20"); getch ();

}

**OUTPUT:**

****

**3. Program to find factorial of a given number.**

void main(void)

{ clrscr(); double f; int num;

// computes the product n\*(n-1)\*(n-2)\*(n-3)\*...\*2\*1 while(num!=0)

{

printf("\n\n\n\t\tEnter the number to find its factorial: ");

scanf("%d",&num);

f=1;

while(num>1)

{

f=f\*num;

num--;

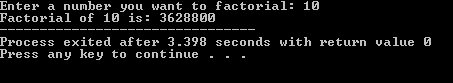
}

printf("\t\tThe factorial is %lf",f);

}

getch();

}



1. During execution of following program, how many lines of asterisks are displayed?

for(i=0;i<3;++i) for(j=0;j<4;++j) printf("\*\*\*\*\*\*\n");

Ans; 12 line of asterisk are display

**OUTPUT:**



1. **Write a program to print sum of even numbers from 1 to 100.**

**Ans:**

#include <stdio.h>

int main()

{

int i, n, sum=0;

printf("Enter a number : ");

scanf("%d", &n);

for(i=2; i<=n; i+=2)

{

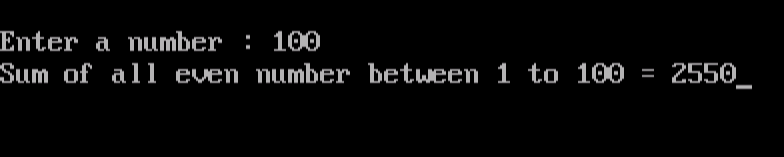
sum += i;

}

printf("Sum of all even number between 1 to %d = %d", n, sum);

}

**OUTPUT:**



1. **Write a program to print square of all the numbers from 1 to 20.**

**SOLUTION:**

#include <stdio.h>

#include<conio.h>

#include <math.h>

void main()

{

int i,n;

clrscr();

printf("Enter a number N : ");

scanf("%d",&n);

printf("No \tSquare \tCube \tSqrRoot\n",n);

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

{

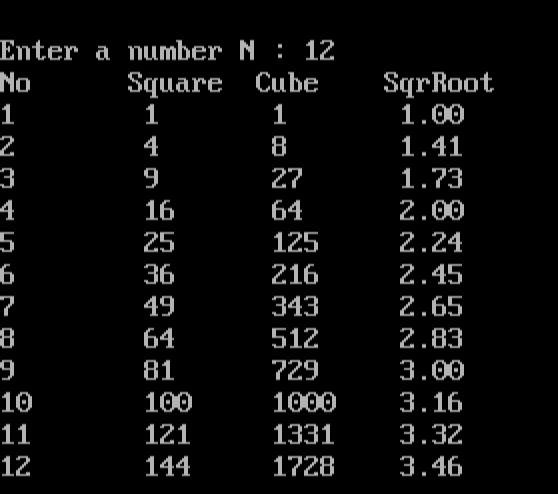
printf("%d\t %d\t %d\t %.2f\n",i,(i\*i),(i\*i\*i),sqrt((double)i));

}

getch();

}

**OUTPUT:**



1. **Write a program of nested loop that cause following output to be displayed.**

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

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

**\*\*\***

**\*\***

**\***

**\*\***

**\*\*\***

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

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

**SOLUTION:**

#include <stdio.h>

#include <conio.h>

int main(){

int i,j;

for(i=1;i<=5;i++){

for(j=5;j>=i; j--){

printf("\*");

}

printf("\n");

}

for(i=2;i<=5;i++){

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

printf("\*");

}

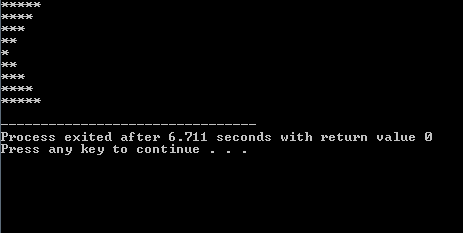
printf("\n");

}

getch();

}

**OUTPUT:**

****

|  |  |  |
| --- | --- | --- |
| **\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Date** | **Lab Instructor Sign.** | **Course Instructor Sign.** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **\_\_\_\_\_\_\_\_\_** |  |  |  |  |  |  |  |  | **\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Student ID** |  |  |  |  |  |  |  |  | **Student Name** |

**Laboratory Experiment 04**

**Objective:** Decision making structures: If and If-else

# Theory:

The if, if...else and nested if...else statement are used to make one-time decisions in C Programming, that is, to execute some code/s and ignore some code/s depending upon the test condition. Without decision making, the program runs in similar way every time. Decision making is an important feature of every programming language using C programming. Before you learn decision making, you should have basic understanding of relational operators.

* If Statement
* If…Else Statement
* Switch Statement

**If Statement**

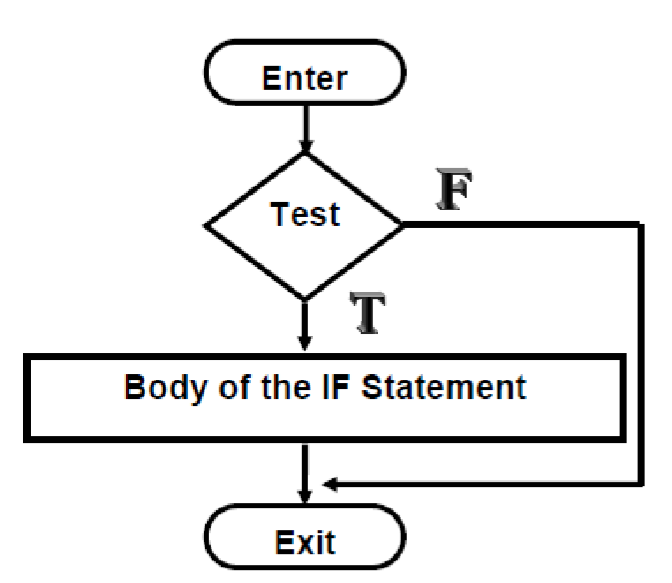
The if statement checks whether the test condition is true or not. If the test condition is true, it executes the code/s inside the body of if statement. But it the test condition is false, it skips the code/s inside the body of if statement.

**Syntax of If statement**

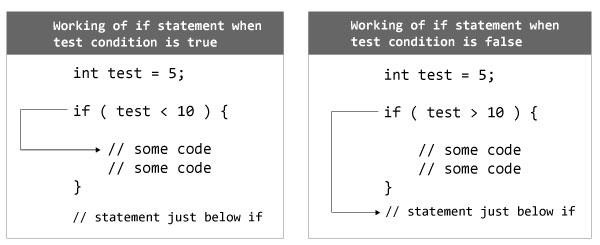
If(condition)

Statement;

**Operation of If statement**



**Working of if Statement**



The if keyword is followed by test condition inside parenthesis ( ). If the test condition is true, the codes inside curly bracket is executed but if test condition is false, the codes inside curly bracket { } is skipped and control of program goes just below the body of if as shown in figure above.

**The if-else statement**

Often your program will want to take one branch if your condition is true, another if it is false. If only one statement is to be followed by the if or else condition then there is no need of parenthesis. The keyword else is used to perform this functionality.

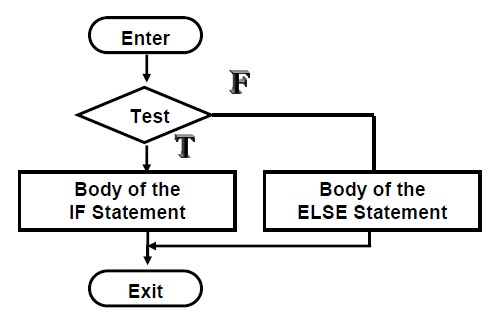
**Syntax of If –Else statement**

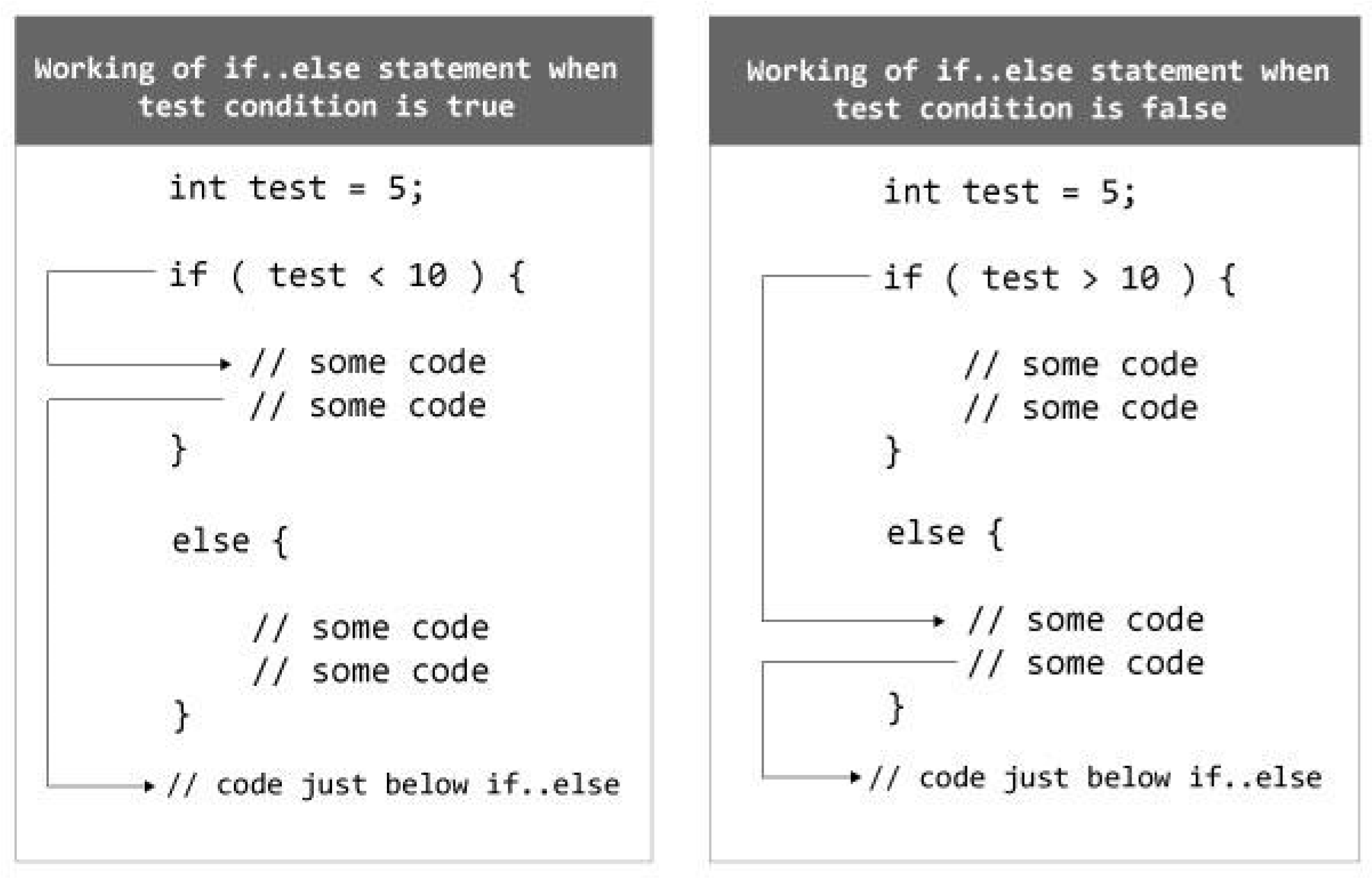
if (condition)

statement 1; //Executes if Condition is true else

statement 2; //Executes if condition is false

**Operation of If-Else Structure**





# Student’s tasks:

**1. Write a following program which guesses number user thinking of.**

void main(void)

{

float guess,incr; char ch;

printf("Think of a number between 1 and 99,and\n"); printf("I'will guess what it is. Type 'e' for equals,\n"); printf("'g' for greater than, and 'l' for less than.\n"); incr=guess=50; /\* two assignments at once \*/ while(incr>1.0) /\* while not close enough \*/

{

printf("\nIs your number greater or less than %.0f?\n",guess); incr=incr/2; if((ch=getche())=='e') /\* if guess it already \*/ break; /\* escape from loop \*/

else if(ch=='g') /\* if guess too low \*/ guess=guess+incr; /\* try higher \*/

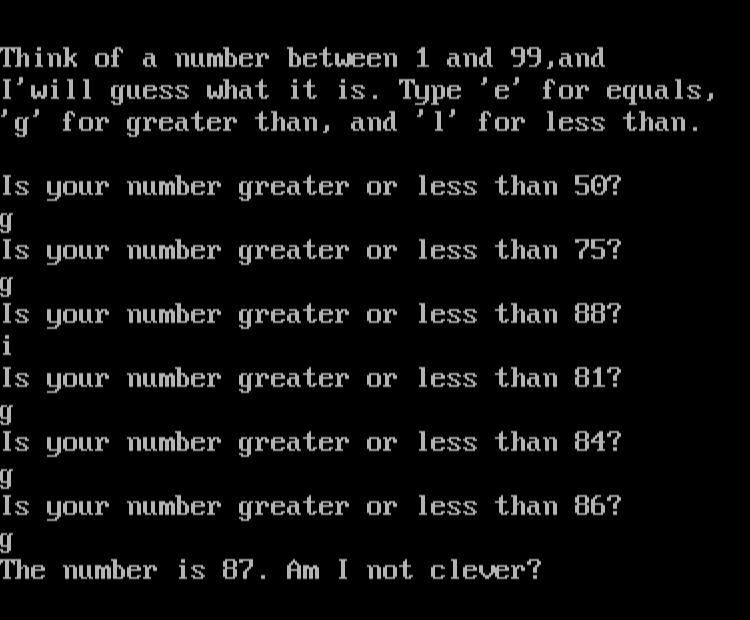
else /\* if guess too high \*/ guess=guess-incr; /\* try lower \*/

}

printf("\nThe number is %.0f. Am I not clever?",guess); getch();

}

**OUTPUT:**



1. **What is the output of the following program?**

if(temp<80)

{

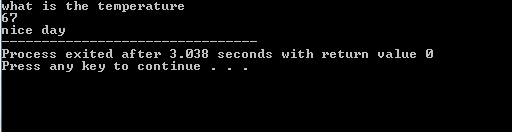
if(temp>60)

printf(“nice day”);

} else

printf(“sure it is hot”);

**OUTPUT:**

****

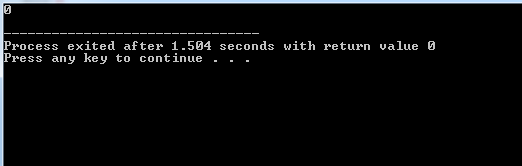
1. **What is the output of the following program?**

char x,y; clrscr();

scanf(“%c”,&x); scanf(“%c”,&y); if(x==’n’ && y==’o’)

printf(“you type no”);

**OUTPUT:**

****

1. **Write a program which takes a character input and checks whether it is vowel or consonant?**

**SOLUTION:**

#include <stdio.h>

#include<conio.h>

void main()

{

char ch;

int lowercase\_Vowel, uppercase\_Vowel;

clrscr();

printf("Please Enter an alphabet: \n");

scanf(" %c", &ch);

lowercase\_Vowel = (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');

uppercase\_Vowel = (ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U');

if (lowercase\_Vowel || uppercase\_Vowel) {

printf("\n %c The Enter Alphabet is a VOWEL.", ch);

}

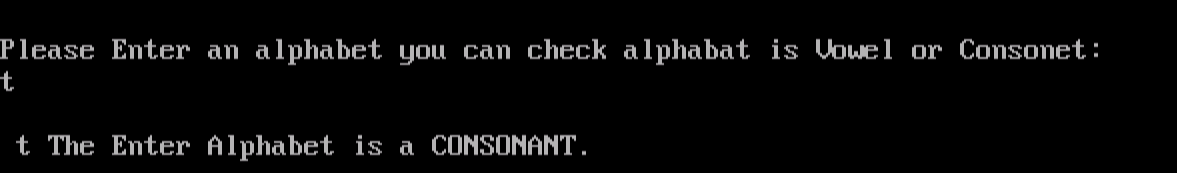
else {

printf("\n %c The Enter Alphabet is a CONSONANT.", ch);

}

getch();

}



**4. Write a program that inputs an integer and determine if it is even or odd.?**

**SOLUTION:**

#include<stdio.h>

#include<conio.h>

int main()

{

int num;

clrscr();

printf("enter number :");

scanf("%d",&num);

int rem=num%2;

if (rem==0)

{

printf("%d is Even number",num);

}

else

{

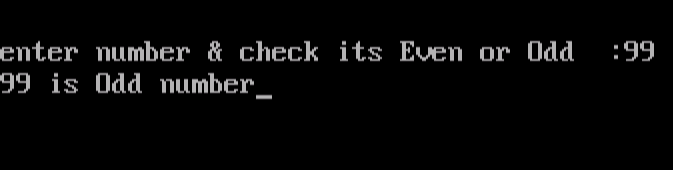
printf("%d is Odd number",num);

}

getch();

}

**OUTPUT:**



**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 05**

# Objective:

Decision making structure: switch statement and conditional (Ternary) operator

# Theory:

Normally, your program flows along line by line in the order in which it appears in your source code. But, it is sometimes required to execute a particular portion of code only if certain condition is true; or false i.e. you have to make decision in your program. There are three major decision making structures. The ‘if’ statement, the if-else statement, and the switch statement. Another less commonly used structure is the conditional operator.

# The switch Statement

Unlike if , which evaluates one value, switch statements allow you to branch on any of a number of different values. There must be break at the end of the statements of each case otherwise all the preceding cases will be executed including the default condition.

The general form of the switch statement is:

switch(expression)

{ case constant-expression :

statement(s); break; case constant-expression :

statement(s);

break;

/\* you can have any number of case statements \*/ default :

statement(s);

}

# Conditional Operator

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

The conditional operator takes three expressions and returns a value:

(expression1) ? (expression2) : (expression3)

It replaces the following statements of if else structure if(a>b) c=a; else c=b; can be replaced by

c=(a>b)?a:b

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.

# Student’s tasks:

**1. Write a following program and observe the output.**

void main ()

{

/\* local variable definition \*/ char grade = 'B';

switch(grade)

{ case 'A' : printf("Excellent!\n" );

break; case 'B' : case 'C' : printf("Well done\n" );

break; case 'D' : printf("You passed\n" );

break; case 'F' : printf("Better try again\n" );

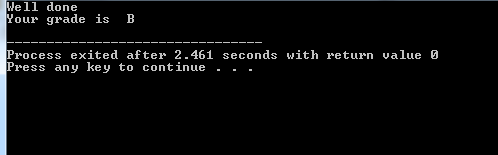
break; default : printf("Invalid grade\n" );

}

printf("Your grade is %c\n", grade ); getch();

}

**Output:**

****

1. **What is the output of the following program?**

color =’R’; switch(color)

{

case ‘R’: printf(“Red\n”); case ‘B’:

printf(“Blue\n”);

case ‘Y’: printf(“Yellow\n”); }

**Ans: RED**

1. **What is the output of the following program?** float i=1.5;

switch(i)

{ case 1:

printf(“1”); case 2:

printf(“2”); case 3:

printf(“0”);

}

**Ans 120.**

1. **What is the output of the following program?**

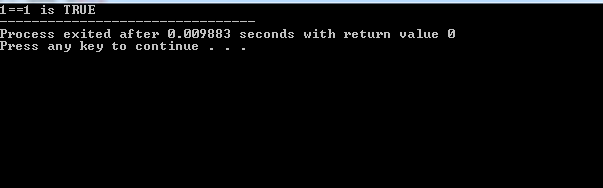
main() {

int k=1;

printf(“%d==1 is “”%s”,k,k==1?”TRUE”:FALSE”);

}

**Output:**



1. **Write a program that asks user an arithmetic operator ('+','-','\*', '/',%) and two operands and perform the corresponding calculation on the operands.**

**SOLUTION:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

int a,b;

int ch;

clrscr();

printf("\nEnter a and b :");

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

printf("\nEnter operation to be performed\n1. +\n2. -\n3. \*\n4. /\n5. %%\n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf(" %d + %d = %d",a,b,a+b);

break;

case 2:printf(" %d - %d = %d",a,b,a-b);

break;

case 3:printf(" %d \* %d = %d",a,b,a\*b);

break;

case 4:printf(" %d / %d = %.2f",a,b,a/b);

break;

case 5:printf(" %d",a%b);

break;

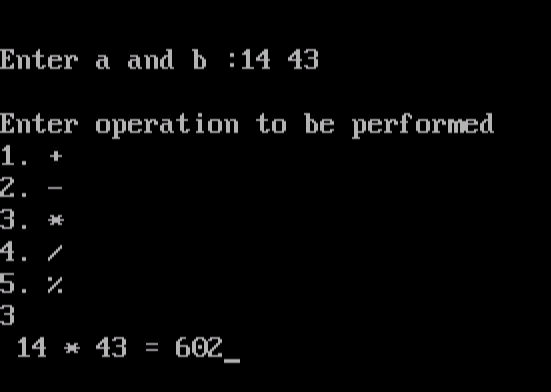
default:printf("\nEnter valid numbers or operator");

}

getch();

}

**OUTPUT:**

****

1. **Write a program which takes a text input and count total number of vowels, consonants and other special characters and print the result?**

#include <stdio.h>

#include<conio.h>

void main()

{

char line[150];

int i, vowels, consonants, digits, spaces;

vowels = consonants = digits = spaces = 0;

clrscr();

printf("Enter a line of string: ");

scanf("%[^\n]", line);

for(i=0; line[i]!='\0'; ++i)

{

if(line[i]=='a' || line[i]=='e' || line[i]=='i' || line[i]=='o' || line[i]=='u' || line[i]=='A' || line[i]=='E' || line[i]=='I' || line[i]=='O' || line[i]=='U')

{

++vowels;

}

else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))

{

++consonants;

}

else if(line[i]>='0' && line[i]<='9')

{

++digits;

}

else if (line[i]==' ')

{

++spaces;

}

}

printf("Vowels: %d",vowels);

printf("\nConsonants: %d",consonants);

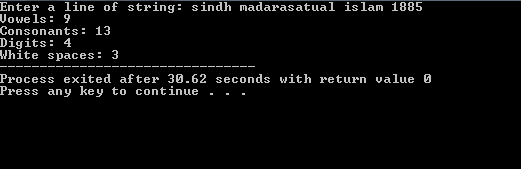
printf("\nDigits: %d",digits);

printf("\nWhite spaces: %d", spaces);

getch();

}

**OUTPUT:**

****

1. **Write a program which takes 10 integers as input and prints the largest one.**

ANS**:**

#include <stdio.h>

#include<conio.h>

void main() {

int n;

double arr[10];

clrscr();

printf("Enter the number of elements (1 to 10): ");

scanf("%d", &n);

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

printf("Enter number%d: ", i + 1);

scanf("%lf", &arr[i]);

}

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

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

arr[0] = arr[i];

}

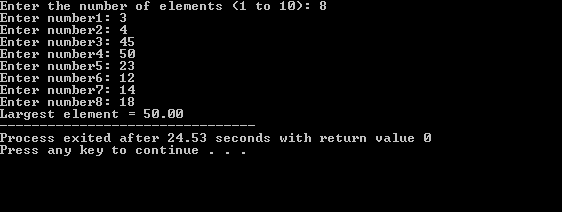
}

printf("Largest element = %.2lf", arr[0]);

getch();

}

**OUTPUT:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 06**

# Objective of Experiment:

Decision making switch and conditional (Ternary) operator

**Goals:** After completing this lab, the students should be able to understand the concept of switch and conditional (Ternary) operator.

# Required Tools / Equipment’s:

* PC
* Turbo C

# Theory:

Normally, your program flows along line by line in the order in which it appears in your source code. But, it is sometimes required to execute a particular portion of code only if certain condition is true; or false i.e. you have to make decision in your program. There are three major decision making structures. The ‘if’ statement, the if-else statement, and the switch statement. Another less commonly used structure is the conditional operator.

**The switch Statement**

Unlike if , which evaluates one value, switch statements allow you to branch on any of a number of different values. There must be break at the end of the statements of each case otherwise all the preceding cases will be executed including the default condition.

switch

(

expression

){

case

constant

-

expression

:

statement

(

s

;

)

break

;

/\* optional \*/

case

constant

-

expression

:

statement

(

s

;

)

break

;

/\* op

tional \*/

/\* you can have any number of case statements \*/

default

:

/\* Optional \*/

statement

(

s

)

;

}

The general form of the switch statement is:

|  |
| --- |
| #include <stdio.h> |
| int main ()  {  /\* local variable definition \*/ char grade = 'B'; switch(grade)  { case 'A' :  printf("Excellent!\n" ); |
| break; |
| case 'B' : case 'C' : printf("Well done\n" ); break; case 'D' : printf("You passed\n" ); break; case 'F' : printf("Better try again\n" ); break; default :  printf("Invalid grade\n" );  }  printf("Your grade is %c\n", grade ); return 0; } |

**Conditional Operator**

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

The conditional operator takes three expressions and returns a value:

(expression1) ? (expression2) : (expression3)

It replaces the following statements of if else structure

If(a>b) c=a; else c=b; can be replaced by

# c=(a>b)?:a:b Student’s task:

# Answer the following questions:

1. Write a program that asks user an arithmetic operator('+','-','\*' or '/') and two operands and perform the corresponding calculation on the operands.?

**Ans:**

**SOLUTION:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

int a,b;

int ch;

clrscr();

printf("\nEnter a and b :");

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

printf("\nEnter operation to be performed\n1. +\n2. -\n3. \*\n4. /\n5. %%\n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf(" %d + %d = %d",a,b,a+b);

break;

case 2:printf(" %d - %d = %d",a,b,a-b);

break;

case 3:printf(" %d \* %d = %d",a,b,a\*b);

break;

case 4:printf(" %d / %d = %.2f",a,b,a/b);

break;

case 5:printf(" %d",a%b);

break;

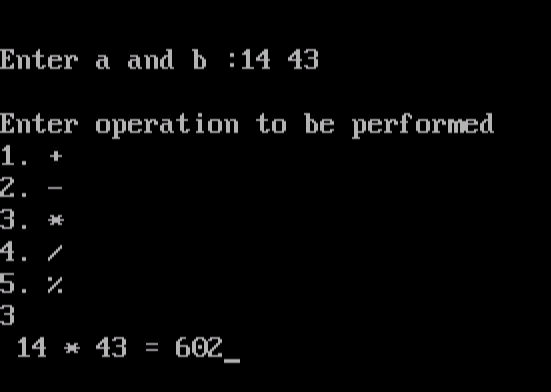
default:printf("\nEnter valid numbers or operator");

}

getch();

}

**OUTPUT:**

****

1. .Write a program which takes a text input counts total number of vowels, consonants and other special characters and prints the result**?**

**Ans:**

#include <stdio.h>

#include<conio.h>

void main()

{

char line[150];

int i, vowels, consonants, digits, spaces;

vowels = consonants = digits = spaces = 0;

clrscr();

printf("Enter a line of string: ");

scanf("%[^\n]", line);

for(i=0; line[i]!='\0'; ++i)

{

if(line[i]=='a' || line[i]=='e' || line[i]=='i' || line[i]=='o' || line[i]=='u' || line[i]=='A' || line[i]=='E' || line[i]=='I' || line[i]=='O' || line[i]=='U')

{

++vowels;

}

else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))

{

++consonants;

}

else if(line[i]>='0' && line[i]<='9')

{

++digits;

}

else if (line[i]==' ')

{

++spaces;

}

}

printf("Vowels: %d",vowels);

printf("\nConsonants: %d",consonants);

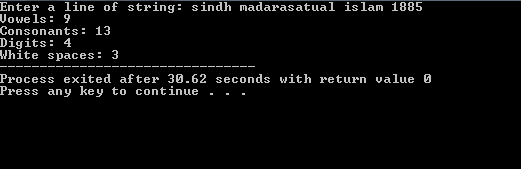
printf("\nDigits: %d",digits);

printf("\nWhite spaces: %d", spaces);

getch();

}

**OUTPUT:**

****

1. Write a program which takes 10 integers as input and prints the largest one**.**?

**Ans:**

#include <stdio.h>

#include<conio.h>

void main() {

int n;

double arr[10];

clrscr();

printf("Enter the number of elements (1 to 10): ");

scanf("%d", &n);

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

printf("Enter number%d: ", i + 1);

scanf("%lf", &arr[i]);

}

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

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

arr[0] = arr[i];

}

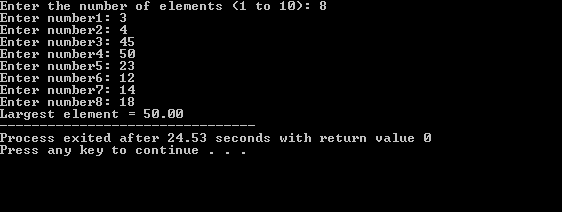
}

printf("Largest element = %.2lf", arr[0]);

getch();

}

**OUTPUT:**

****

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 07**

# Objective of Experiment:

To understand the concepts of function.

**Goals:** After completing this lab, the students should be able to understand the concept of function.

# Required Tools / Equipment’s:

* PC
* Turbo C(IDE)

# Theory:

The **function** is a good mechanism to encapsulate code used repeatedly in a program so that it can be called from other parts of the code. A function does not use a keyword called function but instead the programmer has to define function prototype before the main function and then define the function again later. A function has the following format:

type function\_name (optional parameter list )

{ function code ;

**return** value;

}

Return data type of function is in general the types of C++ variable types including int, double, char etc.

The function does some processing and the calculated value is returned using the return value.

In the main function or the other functions calling this function\_name, the value returned is used like the instruction:

calling\_value = function\_name (parameters);

A function does not need to always return a value.

A function not returning a value can omit the return statement and the function type is void in this case.

Function prototype has the following format: type function\_name (list of variable types);

Examples are:

Example 1: int compute\_sum (int);

Example 2:

void tryout ();

Function prototypes differ from the function definitions in two places: there is no code (no {} with code in between) and the variable names do not follow the types. A function prototype can return nothing, in which case void is the type returned; also it may have no parameters at all like example 2 above. The function prototype is declared before the main function with the function calls inside the main function or the other functions calling this function. The function definitions are put after the main function.

**C Standard Library** : is a collection of functions, which are written in the core language and part of the C ISO Standard itself to provides several generic functions that utilize and manipulate these containers, function objects, generic strings and streams (including interactive and file I/O), support for some language features, and everyday functions for tasks such as finding the square root of a number. **Using sqrt in <cmath> header file**.

# Student’s task:

# Answer the following questions:

1. Write a function called Sum\_even which takes as input two integers N and M entered by the user from keyboard , and returns to the main function the summation of even numbers from N to M. (note: N and M included if were even numbers) .?

**Ans:**

#include<stdio.h>

#include<conio.h>

void main()

{

int number;

clrscr();

printf("Enter an integer: ");

scanf("%d", &number);

if (number%2 == 0)

{

printf("%d is an Even integer. ",number);

}

else

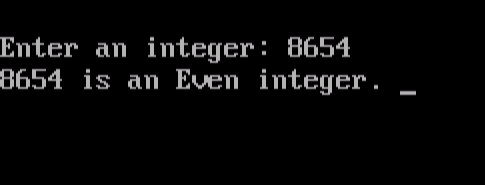
{

printf("%d is an Odd integer.SSSS ",number);

}

getch();

}



1. Write a function called Is\_Capital that has one parameter and receives a character and returns true if the character received is a capital letter, and false otherwise. Then write a main function that calls the Is\_Capital function to test its correctness.(you may solve it without using if-else statement).?

**Ans:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Write a C function to receive an integer of 3 digits then calculates and returns the sum of the MSD and the LSD. For example if your function received 345 it should return 8. ?

**Ans:**

#include<stdio.h>

#include<conio.h>

int l\_m(int x, int y, int z);

int main()

{

int x,y,z;

printf(" Enter First-Digit No. : ");

scanf("%d",&x);

printf(" Enter Second-Digit No. : ");

scanf("%d",&y);

printf(" Enter Third-Digit No. : ");

scanf("%d",&z);

printf(" The Sum of LSB & MSB : %d",l\_m(x,y,z));

}

int l\_m(int x, int y, int z)

{

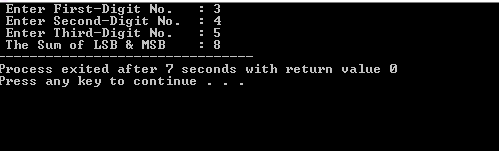
int sum;

sum = x+z;

return sum;

}

**OUTPUT:**

****

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 08**

# Objective of Experiment:

To understand the concepts of recursion function.

**Goals:** After completing this lab, the students should be able to understand the concept of recursion function.

# Required Tools / Equipment’s:

* PC
* Turbo C(IDE)

# Theory:

Recursion is the process of repeating items in a self-similar way. Same applies in programming languages as well where if a programming allows you to call a function inside the same function that is called recursive call of the function as follows.

void recursion()

{

recursion(); /\* function calls itself \*/

}

int main() {

recursion(); }

The C programming language supports recursion, i.e., a function to call itself. But while using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go in infinite loop.

Recursive function are very useful to solve many mathematical problems like to calculate factorial of a number, generating Fibonacci series, etc.

**Number Factorial**

Following is an example, which calculates factorial for a given number using a recursive function:

|  |
| --- |
| #include <stdio.h>    int factorial(unsigned int i)  {  if(i <= 1)  {  return 1;  }  return i \* factorial(i - 1);  }  int main()  {  int i = 15;  printf("Factorial of %d is %d\n", i, factorial(i)); return 0;  } |

When the above code is compiled and executed, it produces the following result:

Factorial of 6 is 720

**Fibonacci Series**

Following is another example, which generates Fibonacci series for a given number using a recursive function:

#include <stdio.h>

int fibonaci(int i)

{

if(i == 0)

{

return 0;

} if(i == 1)

{

return 1;

}

return fibonaci(i-1) + fibonaci(i-2);

}

int main()

{

int i;

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

{

printf("%d\t%n", fibonaci(i));

} return 0; }

When the above code is compiled and executed, it produces the following result:

0

1

1

2

3

5

8

13

21

34

# Student’s task:

# Answer the following questions:

1. Write a function called Sum\_even which takes as input two integers N and M entered by the user from keyboard , and returns to the main function the summation of even numbers from N to M. (note: N and M included if were even numbers) .?

**Ans:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Write a C function to receive an integer of 3 digits then calculates and returns the sum of the MSD and the LSD. For example if your function received 345 it should return 8. ?

**Ans:**

#include<stdio.h>

#include<conio.h>

int l\_m(int x, int y, int z);

int main()

{

int x,y,z;

printf(" Enter First-Digit No. : ");

scanf("%d",&x);

printf(" Enter Second-Digit No. : ");

scanf("%d",&y);

printf(" Enter Third-Digit No. : ");

scanf("%d",&z);

printf(" The Sum of LSB & MSB : %d",l\_m(x,y,z));

}

int l\_m(int x, int y, int z)

{

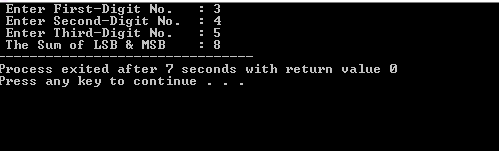
int sum;

sum = x+z;

return sum;

}

**OUTPUT:**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 09**

# Objective of Experiment:

Learn how to use the array data structure to represent a set of related data items.

Learn how to declare arrays, initialize arrays and refer to the individual elements of arrays. Learn how to pass arrays to functions.

**Goals:** After completing this lab, the students should be able to understand the concept of Array (Single dimensional).

# Required Tools / Equipment’s:

* PC
* Turbo C

# Theory:

**Array**: A collection of individual values, all of the same data type, stored in adjacent memory locations. One Dimensional Array: An array with a single variable index. Using the array name together with an integral valued index in square brackets refers to the individual values. The first array element always has the subscript 0. The second array element has the subscript 1, etc. The base address of an array is its beginning address in memory.

Declaring an Array: Use the following syntax below.

DataType Array Name [ConstIntExpression];

Whenever an array is passed as a parameter, its base address is sent to the called function. Generally, functions that work with arrays require 2 items of information as actual parameters: the beginning address of the array (in memory), and the number of elements to process in the array.

The example below shows the declaration of an integer array of size 10 with element 0

-

9

.

const int MAXSIZE = 10; int array[MAXSIZE];

Arrays can be initialized during declara

tion by equating the array to a listing of the array's

members in brackets. For example

int array[MAXSIZE] = {2 , 4, 6, 8, 10, 12, 14, 16, 18, 20};

Passing Arrays as Function Parameters In C++, arrays are always passed by reference by default.

For example (Function PrintArray):

66

|  |
| --- |
|  |
| void PrintArray(int Array[], int ArraySize) { for (int i = 0; i <= ArraySize - 1; i++) printf("array[" i"] “,Array[i]);    } |

Declare and manipulate Two-dimensional arrays Two-dimensional arrays store a tabular arrangement of values accessed by two indexes, for example matrix[i][j], where i is the row index and j is the column index. To declare and initialize a two-dimensional arrays, Use the

following syntax below.

DataType ArrayName [row][column];

The example below shows the declaring two-dimensional array b

|  |
| --- |
|  |
| int b[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } }; |

# Student’s task:

# Answer the following questions:

1. Write a program takes n number of element from user (where, n is specified by user), stores data in an array and calculates the average of those numbers?

**Ans:**

**#include <stdio.h>**

**int main()**

**{**

**int marks[10], i, n, sum = 0, average;**

**printf("Enter n: ");**

**scanf("%d", &n);**

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

**{**

**printf("Enter number%d: ",i+1);**

**scanf("%d",&marks[i]);**

**sum += marks[i];**

**}**

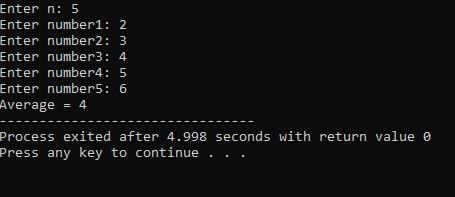
**average = sum/n;**

**printf("Average = %d",average);**

**return 0;**

**}**

**OUTPUT:**



**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 10**

# Objective:

To understand the concept of multidimensional arrays.

**Goals:** After completing this lab, the students should be able to understand the concept of Multidimensional array.

# Required Tools / Equipment’s:

* PC
* Turbo C

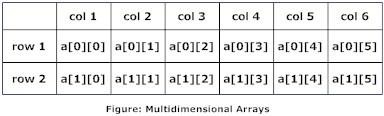
# Theory:

A Multidimensional array is a collection of data storage locations, each of which holds the same type of data. Each storage location is called an element of the array. to create arrays of arrays known as multidimensional arrays. For example:

float a[2][6];

Here, a is an array of two dimension, which is an example of multidimensional array. This array has 2 rows and 6 columns

For better understanding of multidimensional arrays, array elements of above example can be thanked of as below:



**Initialization of Multidimensional Arrays**

In C, multidimensional arrays can be initialized in different number of ways.

int c[2][3]={{1,3,0}, {-1,5,9}};

OR

int c[][3]={{1,3,0}, {-1,5,9}};

OR

int c[2][3]={1,3,0,-1,5,9};

**Initialization Of three-dimensional Array**

|  |
| --- |
| double cprogram[3][2][4]={  {{-0.1, 0.22, 0.3, 4.3}, {2.3, 4.7, -0.9, 2}},  {{0.9, 3.6, 4.5, 4}, {1.2, 2.4, 0.22, -1}},  {{8.2, 3.12, 34.2, 0.1}, {2.1, 3.2, 4.3, -2.0}} }; |

Suppose there is a multidimensional array arr[i][j][k][m]. Then this array can hold i\*j\*k\*m numbers of data.

# Student’s task:

# Answer the following questions:

**1**.Write a program which takes names of five countries as input and prints them in alphabetical order.?

**Ans:**

**#include<stdio.h> #include<string.h>**

**int main()**

**{**

**int i,j,count;**

**char str[25][25],temp[25];**

**puts("How many strings u are going to enter?: ");**

**scanf("%d",&count);**

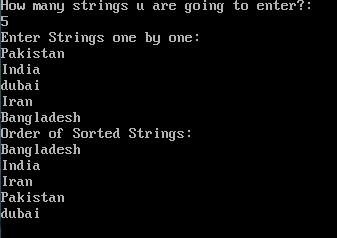
**puts("Enter Strings one by one: "); for(i=0;i<=count;i++)**

**gets(str[i]);**

**for(i=0;i<=count;i++)**

**for(j=i+1;j<=count;j++)**

**{**



**if(strcmp(str[i],str[j])>0)**

**{**

**strcpy(temp,str[i]); strcpy(str[i],str[j]);**

**strcpy(str[j],temp);**

**}**

**}**

**printf("Order of Sorted Strings:");**

**for(i=0;i<=count;i++)**

**puts(str[i]);**

**return 0;**

**}**

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 11**

# Objective of Experiment:

Explain the concepts of String in C language.

**Goals:** After completing this lab, the students should be able to understand the concept of preprocessor directives.

# Required Tools / Equipment’s:

* PC
* Turbo C(IDE)

# Theory:

The string in C programming language is actually a one-dimensional array of characters which is terminated by a **null** character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a **null**.

**Declaration of strings**

Strings are declared in C in similar manner as arrays. Only difference is that, strings are of char type.

**char** s[5];



Strings can also be declared using pointer.

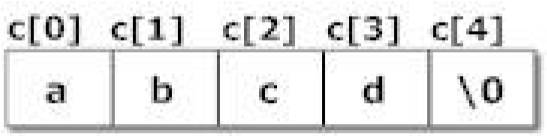
**char** \*p

**Initialization of strings**

In C, string can be initialized in different number of ways.

|  |
| --- |
| **char** c[]="abcd"; OR,  **char** c[5]="abcd"; OR,  **char** c[]={'a','b','c','d','\0'};  OR; |

**char** c[5]={'a','b','c','d','\0'};



String can also be initialized using pointers

**char** \*c="abcd";

**Reading Strings from user**.

Reading words from user.

|  |
| --- |
| **char** c[20]; **scanf("%s",c);** |
| String variable c can only take a word. It is beacause when white space is encountered, |

the scanf()function terminates.

**Write a C program to illustrate how to read string from terminal.**

|  |
| --- |
| #include <stdio.h> **int** main(){ **char** name[20]; printf("Enter name: "); scanf("%s",name);  printf("Your name is %s.",name); **return** 0;  } |

**Output**

Enter name: Dennis Ritchie Your name **is** Dennis.

# Student’s task:

# Answer the following questions:

1. Write a program which takes four integers a, b, c, d as input and prints the largest one using define directive**?**

**Ans:**

**#include<stdio.h>**

**void main()**

**{**

**int a,b,c,d;**

**clrscr();**

**printf("Enter the Four Numbers :");**

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

**if(a>b)**

**{**

**if(a>c)**

**{**

**if(a>d)**

**{**

**printf("%d is big",a);**

**}**

**else**

**{**

**printf("%d is big",d);**

**}**

**}**

**}**

**else if(b>c)**

**{**

**if(b>d)**

**{**

**printf("%d is big",b);**

**}**

**else**

**{**

**printf("%d is big",d);**

**}**

**}**

**else if(c>d)**

**{**

**printf("%d is big",c);**

**}**

**else**

**{**

**printf("%d is big",d);**

**}**

**getch();**

**}**

**Output:**



1. Write a function called Is\_Capital that has one parameter and receives a character and returns true if the character received is a capital letter, and false otherwise. Then write a main function that calls the Is\_Capital function to test its correctness.(you may solve it without using if-else statement).?

**Ans:**

**#include <stdio.h>**

**void check(char ch)**

**{**

**if (ch >= 'A' && ch <= 'Z')**

**{**

**printf("\n%c is an UpperCase character", ch);**

**}**

**else if (ch >= 'a' && ch <= 'z')**

**{**

**printf("\n%c is an LowerCase character", ch);**

**}**

**else**

**{**

**printf("\n%c is not an aplhabetic character", ch);**

**}**

**}**

**void main()**

**{**

**char ch;**

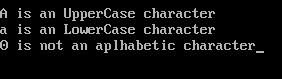
**ch = '0';**

**check(ch);**

**getch();**

**}**

**Output:**



1. Write a program which calculates and returns the area and volume of a sphere using define directive**.** ?

**Ans:**

**#include <stdio.h> #include <math.h>**

**void main()**

**{**

**float radius;**

**float surface\_area, volume;**

**printf("Enter radius of the sphere : \n");**

**scanf("%f", &radius);**

**surface\_area = 4 \* (22/7) \* radius \* radius;**

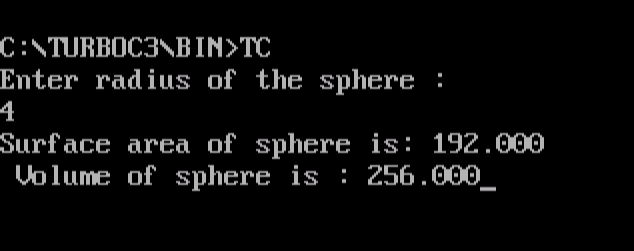
**volume = (4.0/3) \* (22/7) \* radius \* radius \* radius;**

**printf("Surface area of sphere is: %.3f", surface\_area);**

**printf("\n Volume of sphere is : %.3f", volume);**

**getch();**

**}**

****

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign. \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 12**

# Objective of Experiment:

Explain the concepts of preprocessor directives.

**Goals:** After completing this lab, the students should be able to understand the concept of preprocessor directives.

# Required Tools / Equipment’s:

* PC
* Turbo C(IDE)

# Theory:

**Preprocessor directives** are actually the instructions to the compiler itself. They are not translated but are operated directly by the compiler. The most common preprocessor directives are **i. include directive ii. define directive**

1. **include directive**: The include directive is used to include files like as we include header files in the beginning of the program using #include directive like

#include<stdio.h>

#include<conio.h>

1. **define directive**: It is used to assign names to different constants or statements which are to be used repeatedly in a program. These defined values or statement can be used by main or in the user defined functions as well.

They are used for

a. defining a constant b. defining a statement

c. defining a mathematical expression

for example

#define pi 3.142

#define p printf(“enter a new number”);

#define for(a) (4/3.0)\*pi\*(a\*a\*a);

**Predefined Macros**

ANSI C defines a number of macros. Although each one is available for your use in programming, the predefined macros should not be directly modified.

Let's try the following example:

#include

>

stdio.h

<

main

()

{

printf

(

"File :%s

\

n"

,

\_\_FILE\_\_

)

;

printf

(

"Date :%s

\

n"

,

\_\_DATE\_\_

)

;

printf

(

"Time :%s

\

n"

,

\_\_TIME\_\_

;

)

printf

(

"Line :%d

\

n"

,

\_\_LINE\_\_

)

;

printf

(

"ANSI :%d

\

n"

,

\_\_STDC\_\_

)

;

}

|  |  |
| --- | --- |
| **Macro** | **Description** |
| \_\_DATE\_\_ | The current date as a character literal in "MMM DD YYYY" format |
| \_\_TIME\_\_ | The current time as a character literal in "HH:MM:SS" format |
| \_\_FILE\_\_ | This contains the current filename as a string literal. |
| \_\_LINE\_\_ | This contains the current line number as a decimal constant. |
| \_\_STDC\_\_ | Defined as 1 when the compiler complies with the ANSI standard. |

# Student’s task:

# Answer the following questions:

1. Write a program which takes four integers a, b, c, d as input and prints the largest one using define directive**?**

**Ans:**

**#include<stdio.h>**

**void main()**

**{**

**int a,b,c,d;**

**clrscr();**

**printf("Enter the Four Numbers :");**

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

**if(a>b)**

**{**

**if(a>c)**

**{**

**if(a>d)**

**{**

**printf("%d is big",a);**

**}**

**else**

**{**

**printf("%d is big",d);**

**}**

**}**

**}**

**else if(b>c)**

**{**

**if(b>d)**

**{**

**printf("%d is big",b);**

**}**

**else**

**{**

**printf("%d is big",d);**

**}**

**}**

**else if(c>d)**

**{**

**printf("%d is big",c);**

**}**

**else**

**{**

**printf("%d is big",d);**

**}**

**getch();**

**}**

**Output:**



Write a function called Is Capital that has one parameter and receives a character and returns true if the character received is a capital letter, and false otherwise. Then write a main function that calls the Is Capital function to test its correctness.(you may solve it without using if-else statement).?

**Ans:**

**#include <stdio.h>**

**void check(char ch)**

**{**

**if (ch >= 'A' && ch <= 'Z')**

**{**

**printf("\n%c is an UpperCase character", ch);**

**}**

**else if (ch >= 'a' && ch <= 'z')**

**{**

**printf("\n%c is an LowerCase character", ch);**

**}**

**else**

**{**

**printf("\n%c is not an aplhabetic character", ch);**

**}**

**}**

**void main()**

**{**

**char ch;**

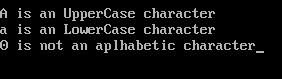
**ch = '0';**

**check(ch);**

**getch();**

**}**

**Output:**



1. Write a program which calculates and returns the area and volume of a sphere using define directive**.** ?

**Ans:**

**#include <stdio.h> #include <math.h>**

**void main()**

**{**

**float radius;**

**float surface\_area, volume;**

**printf("Enter radius of the sphere : \n");**

**scanf("%f", &radius);**

**surface\_area = 4 \* (22/7) \* radius \* radius;**

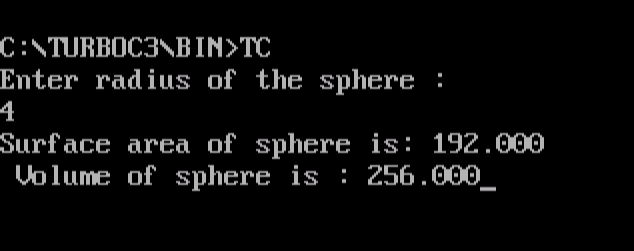
**volume = (4.0/3) \* (22/7) \* radius \* radius \* radius;**

**printf("Surface area of sphere is: %.3f", surface\_area);**

**printf("\n Volume of sphere is : %.3f", volume);**

**getch();**

**}**

****

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign. \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 13**

# Objective of Experiment:

Pointers in C-Language

**Goals:** After completing this lab, the students should be able to understand the concept of pointers in C-Language.

# Required Tools / Equipment’s:

* PC
* Turbo C

# Theory:

A pointer provides a way of accessing a variable without referring to the variable directly. The address of the variable is used.

Pointers are the powerful feature of C and (C++) programming, which differs it from other popular programming languages like: java and Visual Basic.

Pointers are used in C program to access the memory and manipulate the address.Reference operator(&)

If var is a variable then, &var is the address in memory.

|  |
| --- |
| /\* Example to demonstrate use of reference operator in C programming. \*/ |
| #include <stdio.h> int main(){ int var=5;  printf("Value: %d\n",var); |
| printf("Address: %d",&var); //Notice, the ampersand(&) before var. |
| return 0; } |
|  |

Note: You may obtain different value of address while using this code.

In above source code, value 5 is stored in the memory location 2686778. var is just the name given to that location.

You, have already used reference operator in C program while using scanf() function.scanf("%d",&var);

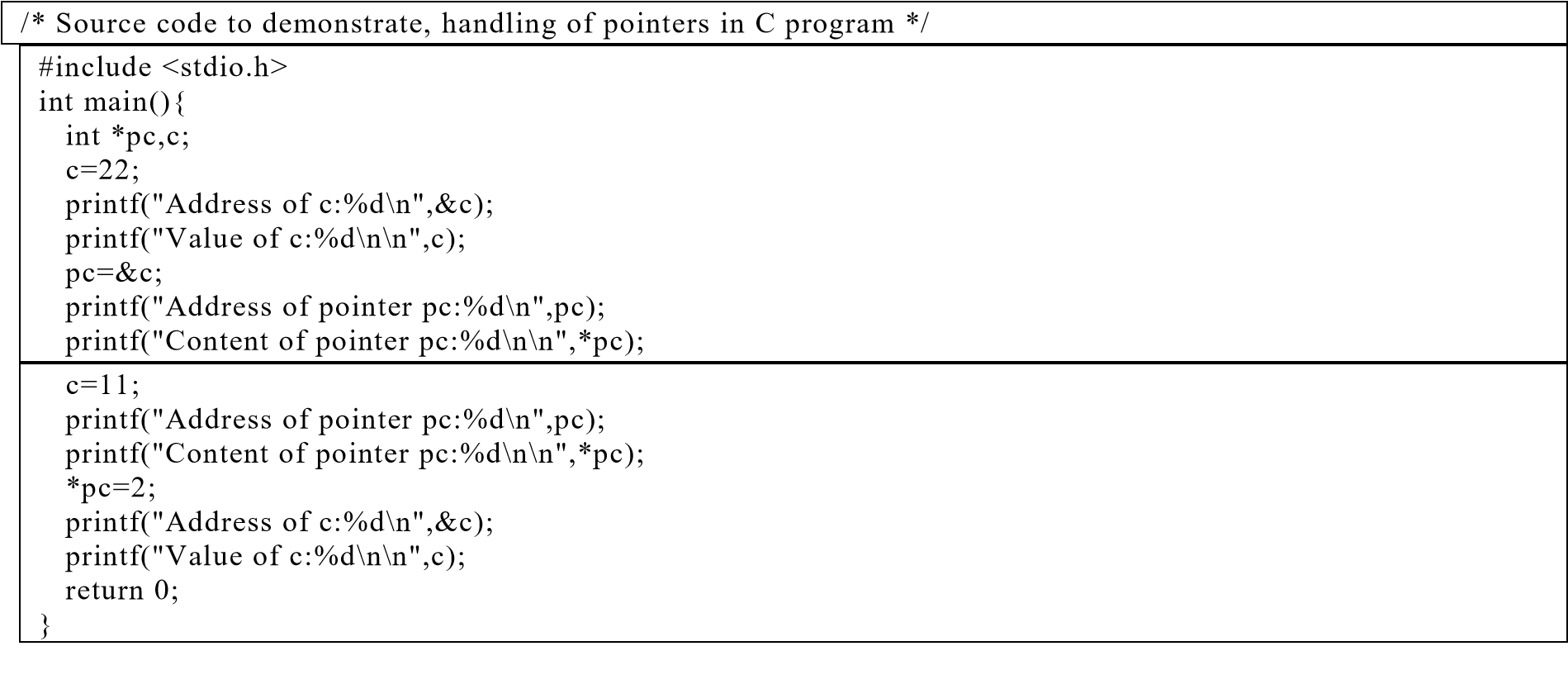
Reference operator(\*) and Pointer variables

Pointers variables are the special types of variables that holds memory address rather than data, that is, a variable that holds address value is called a pointer variable or simply a pointer.

Declaration of **Pointer**

Dereference operator(\*) are used to identify an operator as a pointer. data\_type \* pointer\_variable\_name; int \*p;

Above statement defines, p as pointer variable of type int.

Example To Demonstrate Working of Pointer 

|  |
| --- |
| **Output** |
| Address of c: 2686784  Value of c: 22 |
| Address of pointer pc: 2686784 |
| Content of pointer pc: 22    Address of pointer pc: 2686784 |
| Content of pointer pc: 11 |
| Address of c: 2686784 Value of c:2 |
|  |

# Student’s task:

# Answer the following questions:

1. Write down the number of bytes allocated for the following pointer variables:

int \*x;

char \*y; float \*z;

**Ans:**

#include<stdio.h>

#include<conio.h>

int main()

{

int \*x;

char \*y;

float \*z;

printf("Size of \*x: %ld bytes\n", sizeof(\*x));

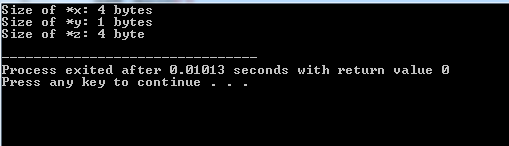
printf("Size of \*y: %ld bytes\n", sizeof(\*y));

printf("Size of \*z: %ld byte\n", sizeof(\*z));

return 0;

}

**Output:**

****

1. Write a c program that defines an integer variable var1 and a pointer Ptr that points to var1 . Assign and print value to var1 , then assign and print a new value to var1 using Ptr?

**Ans:**

#include <stdio.h>

int main () {

int var = 30;

int \*ip;

ip = &var;

printf("Address of var variable: %x\n", &var );

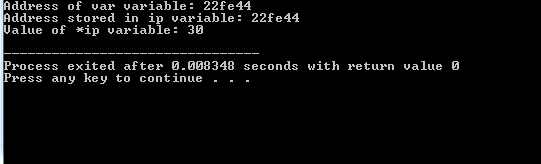
printf("Address stored in ip variable: %x\n", ip );

printf("Value of \*ip variable: %d\n", \*ip );

return 0;

}

**OUTPUT:**

****

1. Determine the output of the following program: void main(void)

{

int x=3,y=4,z=6; int \*p1,\*p2,\*p3; p1=&x; p2=&y; p3=&z; \*p1=\*p2+\*p3;

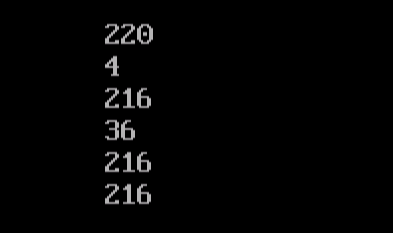
\*p1++;

\*p2--;

\*p1=(\*p2)\*(\*p3);

\*p2=(\*p2)\*(\*p1); x=y+z; printf(“%d”,x); printf(“%d”,y); printf(“%d”,z); printf(“%d”,\*p1); printf(“%d”,\*p2); printf(“%d”,\*p3); }

**Ans:**



**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 14**

# Objective of Experiment:

Structure in C- Language.

**Goals:** After completing this lab, the students should be able to understand the concept of structure in c language.

# Required Tools / Equipment’s:

* PC
* Turbo C

# Theory:

**Structure** is a user-defined data type in C which allows you to combine different data types to store a particular type of record. Structure helps to construct a complex data type in more meaningful way. It is somewhat similar to an Array. The only difference is that array is used to store collection of similar datatypes while structure can store collection of any type of data.

**For example:** You want to store the information about person about his/her name, citizenship number and salary. You can create these information separately but, better approach will be collection of these information under single name because all these information are related to person.

**Defining a structure** struct keyword is used to define a structure. struct define a new data type which is a collection of different type of data.

**struct**

Employee

{

char

name[20];

int

centizenship\_num;

int

salary;

}

|  |
| --- |
| struct employee |
| {  char ename[10];  int sal;  };  struct employee emp[5]; int i,j;  void main()  { |
| for(i=0;i<3;i++) |
| {  printf("\nEnter %dst employee record\n",i+1); printf("\nEmployee name\t"); scanf("%s",emp[i].ename); printf("\nEnter employee salary\t"); scanf("%d", &emp[i].sal);  }  printf("\nDisplaying Employee record\n"); for(i=0;i<3;i++)  {  printf("\nEmployee name is %s",emp[i].ename); printf("\nSlary is %d",emp[i].sal);  }  } |

# Student’s task:

# Answer the following questions:

1. Write a C Program to Store Information (name, roll and marks) of a Student Using Structure?

**Ans:**

#include <stdio.h>

#include<conio.h>

struct student {

char name[50];

int roll;

float marks;

} s;

int main() {

clrscr();

printf("Enter information:\n");

printf("Enter name: ");

fgets(s.name, sizeof(s.name), stdin);

printf("Enter roll number: ");

scanf("%d", &s.roll);

printf("Enter marks: ");

scanf("%f", &s.marks);

printf("\n\n");

printf("Displaying Information:\n");

printf("Name: ");

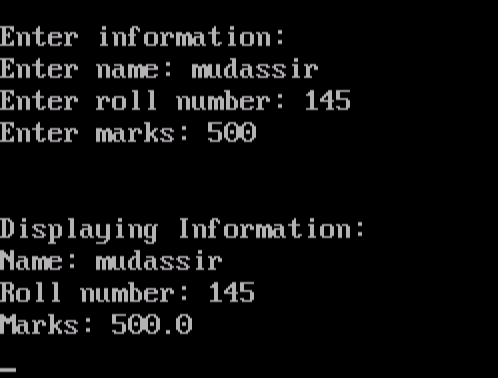
printf("%s", s.name);

printf("Roll number: %d\n", s.roll);

printf("Marks: %.1f\n", s.marks);

getch();

}



1. Write a C program to create a structure student, containing name and roll. Ask user the name and roll of a student in main function. Pass this structure to a function and display the information in that function?

**Ans:**

**#include <stdio.h>**

**#include<conio.h>**

**struct student**

**{**

**char name[50];**

**int roll;**

**};**

**void display(struct student stu);**

**// function prototype should be below to the structure declaration otherwise compiler shows error**

**int main()**

**{**

**clrscr();**

**struct student stud;**

**printf("Enter student's name: ");**

**scanf ("%[^\n]%\*c", stud.name);**

**printf("Enter roll number:");**

**scanf("%d", &stud.roll);**

**display(stud); // passing structure variable stud as argument**

**getch();**

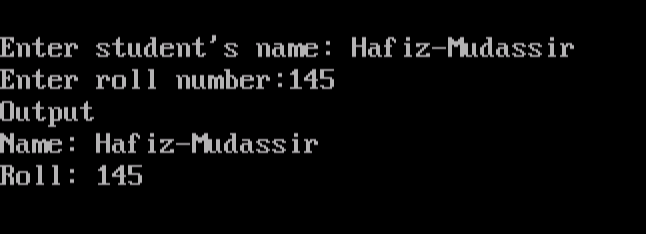
**}**

**void display(struct student stu){**

**printf("Output\nName: %s",stu.name);**

**printf("\nRoll: %d",stu.roll);**

**}**



**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**Student ID Student Name**

**Laboratory Experiment 14**

# Objective of Experiment:

Files in C- Language.

**Goals:** After completing this lab, the students should be able to understand the concept of files in C-Language.

# Required Tools / Equipment’s:

* PC
* Turbo C

# Theory:

Why files are needed?

When the program is terminated, the entire data is lost in C programming. If you want to keep large volume of data, it is time consuming to enter the entire data. But, if file is created, these information can be accessed using few commands.

There are large numbers of functions to handle file I/O in C language.

High level file I/O functions can be categorized as:

* Text file
* Binary file

File Operations

* Creating a new file
* Opening an existing file
* Reading from and writing information to a file
* Closing a file

**Working with file**

While working with file, you need to declare a pointer of type file. This declaration is needed for communication between file and program.

FILE \*ptr;

Opening a file

Opening a file is performed using library function fopen(). The syntax for opening a file in standard I/O is:

ptr=fopen("fileopen","mode")

For Example:

fopen("E:\\cprogram\program.txt","w");

/\* --------------------------------------------------------- \*/ E:\\cprogram\program.txt is the location to create file.

"w" represents the mode for writing.

|  |  |  |
| --- | --- | --- |
|  | Opening Modes in Standard I/O | |
| File Mode | Meaning of Mode | During Inexistence of file |
| R | Open for reading. | If the file does not exist, fopen() returns NULL. |
| W | Open for writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| A | Open for append. i.e, Data is added to end of file. | If the file does not exists, it will be created. |
| r+ | Open for both reading and writing. | If the file does not exist, fopen() returns NULL. |

/\* --------------------------------------------------------- \*/ Here, the program.txt file is opened for writing mode.

**Closing a File**

|  |  |  |
| --- | --- | --- |
|  | Opening Modes in Standard I/O | |
| File Mode | Meaning of Mode | During Inexistence of file |
| w+ | Open for both reading and writing. | If the file exists, its contents are overwritten. If the file does not exist, it will be created. |
| a+ | Open for both reading and appending. | If the file does not exists, it will be created. |

The file should be closed after reading/writing of a file. Closing a file is performed using library function fclose().fclose(ptr); //ptr is the file pointer associated with file to be closed.

The Functions fprintf() and fscanf() functions.

The functions fprintf() and fscanf() are the file version of printf() and fscanf(). The only difference while using fprintf() and fscanf() is that, the first argument is a pointer to the structure FILE **Writing to a file**

|  |
| --- |
| #include <stdio.h> |
| int main()  {  int n;  FILE \*fptr;  fptr=fopen("C:\\program.txt","w"); |
| if(fptr==NULL){ |
| printf("Error!"); exit(1);  }  printf("Enter n: "); scanf("%d",&n); fprintf(fptr,"%d",n); fclose(fptr); return 0;  } |

This program takes the number from user and stores in file. After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open that file, you can see the integer you entered.

Similarly, fscanf() can be used to read data from file.

**Reading from file**

|  |
| --- |
| #include <stdio.h> |
| int main()  {  int n;  FILE \*fptr;  if ((fptr=fopen("C:\\program.txt","r"))==NULL){ |
| printf("Error! opening file"); |
| exit(1); /\* Program exits if file pointer returns NULL. \*/  } |
| fscanf(fptr,"%d",&n); |
| printf("Value of n=%d",n);  fclose(fptr); return 0; } |

If you have run program above to write in file successfully, you can get the integer back entered in that program using this program, Other functions like fgetchar(), fputc() etc. can be used in similar way.

**Binary Files**

Depending upon the way file is opened for processing, a file is classified into text file and binary file, If a large amount of numerical data it to be stored, text mode will be insufficient. In such case binary file is used, Working of binary files is similar to text files with few differences in opening modes, reading from file and writing to file**.**

**Opening modes of binary files**

Opening modes of binary files are rb, rb+, wb, wb+,ab and ab+. The only difference between opening modes of text and binary files is that, b is appended to indicate that, it is binary file.

**Reading and writing of a binary file.**

Functions fread() and fwrite() are used for reading from and writing to a file on the disk respectively in case of binary files.

Function fwrite() takes four arguments, address of data to be written in disk, size of data to be written in disk, number of such type of data and pointer to the file where you want to write.

fwrite(address\_data,size\_data,numbers\_data,pointer\_to\_file);

# Student’s task:

# Answer the following questions:

1.Write a program to create file is called student. rec and store information , Name, Roll no, Batch, department?

**Ans:**

#include <stdio.h>

#include<conio.h>

struct student {

char name[50];

char department[30];

int batch;

int roll;

float marks;

} s;

void main() {

clrscr();

printf("Enter information:\n");

printf("Enter name: ");

fgets(s.name, sizeof(s.name), stdin);

printf("Enter Department: ");

fgets(s.department, sizeof(s.department), stdin);

printf("Enter roll number: ");

scanf("%d", &s.roll);

printf("Enter Batch: ");

scanf("%d", &s.batch);

printf("Enter marks: ");

scanf("%f", &s.marks);

printf("Displaying Information:\n");

printf("Name: ");

printf("%s", s.name);

printf("Department:");

printf("%s", s.department);

printf("Roll number: %d\n", s.roll);

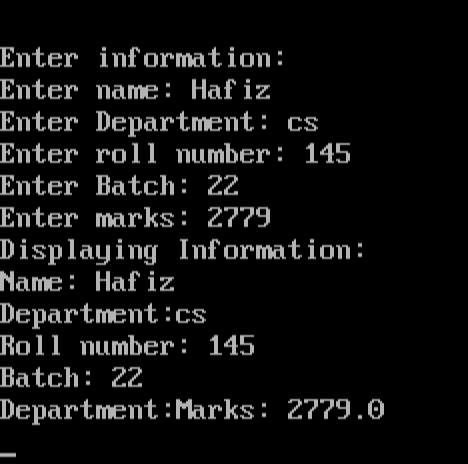
printf("Batch: %d\n", s.batch);

printf("Department:");

printf("Marks: %.1f\n", s.marks);

getch();

}



1. Write a C program to read name and marks of n number of students from user and store them in a file?

**Ans:**

#include<stdio.h>

#include<conio.h>

struct employee{

int id;

char name[20];

// char fname[20];

char position[20];

int grade;

int salary;

char company[20];

};

void main()

{

int i;

clrscr();

FILE \*ptr;

ptr = fopen("empInfo", "w");

struct employee s[25];

for(i=0; i<=24; i++)

{

printf(" insert employee id");

scanf("%d",&s[i].id);

// fprintf("%d", &s[i].id);

printf(" insert employee name");

scanf("%s",&s[i].name);

// fprintf("%s", &s[i].name);

// printf("\n insert employee father name");

// scanf("%s",&s[i].fname);

// fprintf("%s", &s[i].department);

printf(" insert position");

scanf("%s",&s[i].position);

// fprintf("%d", &s[i].mark);

printf(" insert grade");

scanf("%d",&s[i].grade);

// fprintf("%c", &s[i].sec);

printf(" insert salary");

scanf("%d",&s[i].salary);

// fprintf("%s", &s[i].clas);

printf(" insert company");

scanf("%s",&s[i].company);

}

for(i=0; i<=24; i++)

{

fprintf(ptr,"\n id : %d", s[i].id);

fprintf(ptr,"\n name : %s",s[i].name);

// fprintf(ptr,"\n father name = %s",s[i].fname);

fprintf(ptr,"\n poaition : %s", s[i].position);

fprintf(ptr,"\n grade : %d", s[i].grade);

fprintf(ptr,"\n salary : %d",s[i].salary);

fprintf(ptr,"\n company : %s",s[i].company);

fprintf(ptr,"\n");

}

fclose(ptr);

getch();

}

Write a program delete student. rec file?

**Ans:**

#include <stdio.h>

int main()

{

int status;

char file\_name[25];

printf("Enter name of a file you wish to delete\n");

gets(file\_name);

status = remove(file\_name);

if (status == 0)

printf("%s file deleted successfully.\n", file\_name);

else

{

printf("Unable to delete the file\n");

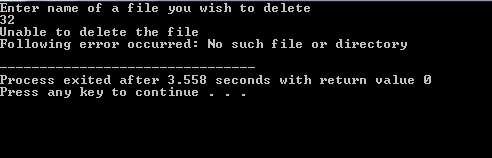
perror("Following error occurred");

}

return 0;

}

**Output:**

****

**\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date Lab Instructor Sign. Course Instructor Sign.**