LAB MANUAL

ADVANCED C LAB

Course code: 339

Department of Computer Engineering

Semester 3

ExpNo.1

INTRODUCTION TO C PROGRAM

AIM: (1) To understand and study the history, characteristics and key features of C programming language.

(2) To familiarize with the structure of C programs, need of header files and main functions and various fundamental data types in C language.

(3) To list the type qualifiers, keywords, type of statements and various categories of C language.

HISTORY OF C LANGUAGE

'C' seems a strange name for a programming language. But this strange sounding language is one of the most popular computer languages of today. 'C' was an offspring of the "Basic Combined Programming Language" [BCPL] called 'B' developed in the 1960's at Cambridge University. 'B' language was modified by Dennis Ritchie and was implemented at Bell laboratories in 1972.The new language was named C. Since it was developed along with the UNIX operating system, which was also developed at Bell laboratories, was coded almost entirely in C.

For many years, C was used mainly in academic environments, but eventually with the release of 'C' compilers for commercial use and the increasing popularity of UNIX, it began to gain wide spread support among computer professionals. Today C is running under a number of operating systems including MS-DOS. Since MS-DOS is a dominant operating system for microcomputers, it is natural that 'C' has begun to influence the microcomputer community at large.

CHARACTERISTICS OF C LANGUAGE

The increasing popularity of 'C' is probably due to its many desirable qualities

1. 'C' language is a robust language whose rich set of built-in functions and operators can be used to write any complex program.
2. 'C' language is well suited for writing both system software and business packages.
3. Program written in 'C' are efficient and fast. This due to its variety of data types and powerful operators.
4. 'C' language is highly portable. This means that 'C' programs written for one computer can be run on another with little or no modification.
5. 'C' language is well suited for structured programming.
6. 'C' language is its ability to extend itself.
7. Functions and data pointers permit advance runtime polymorphism.
8. In 'C' language user defined and compound types are possible.
9. Functions returns values can be ignored when not needed.
10. In 'C' more than one assignment may be performed in a single statements

CATEGORY OF VARIOUS OPERATORS IN C LANGUAGE

Arithmetic Operators

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division

%	Modulo division

Relational Operators

Operator	Meaning
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to

Bitwise Operators

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	Shift left
>>	Shift right
~	One's compliment

STRUCTURE OF C PROGRAM

A 'C' program can be viewed as a group of blocks are called functions. To write a c program, we first create functions and then put them together. A 'c' program may contain one or more sections.

Documentation Section: It consists of a set of comment lines giving the name of the program.

Link Section: It provides instructions to the compiler to link functions from the system library.

Definition Section: It defines all symbolic constants.

Main () function section: This section contains two parts declaration part and executable part. The declaration part declares all the variables used in the executable part. These two parts must appear between the opening and closing braces. The program execution begins at the opening brace and ends at the closing brace. All statements in the declaration and executable parts end with a semicolon.

Subprogram Section: It contains all the user defined functions that are called in the main function.

An overview of a 'C' program

<table>
<tr><td>Documentation section</td></tr>
<tr><td>Link section</td></tr>
<tr><td>Definition section</td></tr>
<tr><td>Global variable declaration section</td></tr>
<tr><td>Main() function section
{
<table><tr><td>Declaration part</td></tr><tr><td>Executable part</td></tr></table>
}</td></tr>
<tr><td>Subprogram section (user defined functions)
<table><tr><td>Function 1</td></tr><tr><td>Function 2</td></tr><tr><td>.
.
.</td></tr><tr><td>Function n</td></tr></table></td></tr>
</table>

NEED OF HEADER FILES AND MAIN FUNCTION

A header file is used to define constants, variables, macros and functions that may be common to several applications. When a system consist of multiple applications that all access the same data it become essential that each application uses identical definitions and it is safer for all of those application to use the same methods to read that data. Updating data should only be performed by a single function, form a single application, but reading data can be safely performed by using the common definitions found in header files.

The main function is the function that holds all the code that runs the program. If we don't use main function in C programs, the compiler will not execute the program. 'C' compiler starts execution from main function itself.

FUNDAMENTAL DATA TYPES

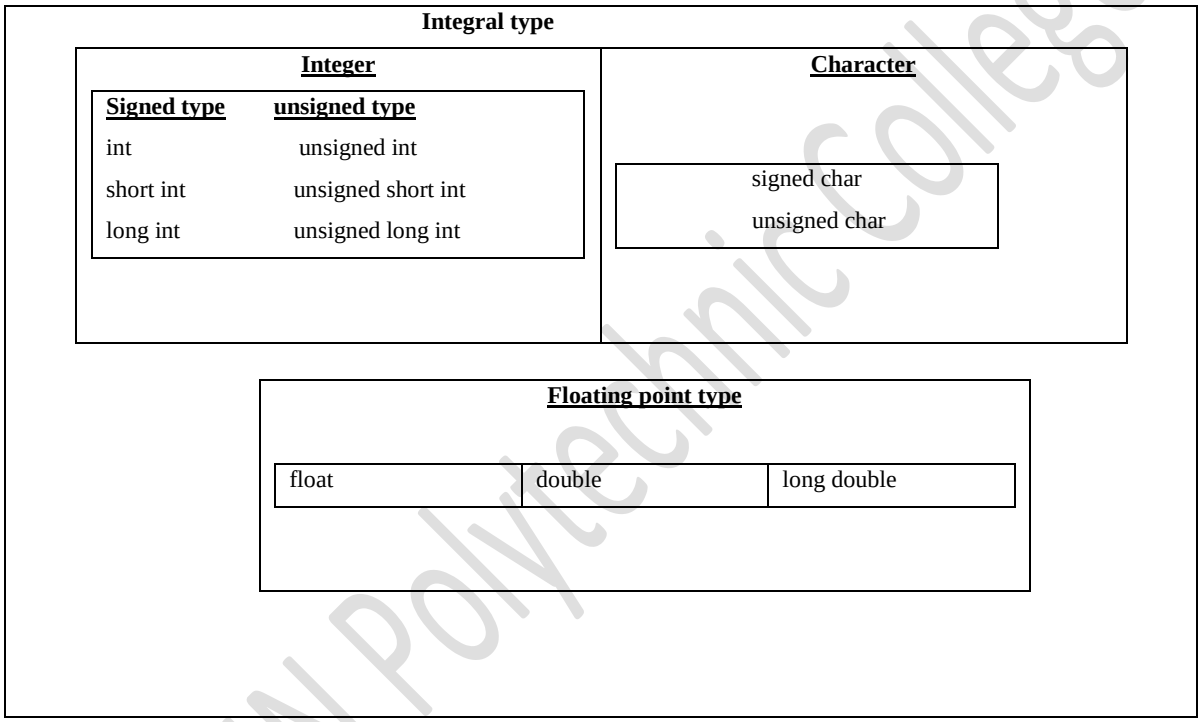
'C' language is rich in its datatypes.All C compilers support four fundamental data types, namely Integer (int), Character(char), Floating point(float) and double precision floating point(double).

Integer types: Integers are whole numbers with a range of values supported by a particular machine. The size of the integer value is limited to the range -32768 to +32767.

Floating point types: floating point numbers are stored in 32 bits with 6 digits of precision represent in float. When the accuracy provided by a float number is not sufficient, the double can be used to define the number. A double data type number uses 64 bits giving a precision of 14 digits. These are known as double precession numbers. To extend the precession further, we may use long double which uses 80 bits.

Character types: A single character can be defined as a character (char) data type. Characters are usually stored in 8 bits (one byte) of internal storage. The qualifier signed or unsigned may be explicitly applied to char while unsigned chars have values between 0 and 255, signed char's have values from -128 to 127.

Primary data types



Size and range of basic data types

Data types	Range of values
Char	-128 to 127
Int	-32,768 to 32,767
Float	3.4e-38 to 3.4e+38

Double	1.7e-308 to 1.7e+308

TYPE QUALIFIERS

Type qualifiers give one of two properties to an identifier. The constant type qualifier declares an object to be non modifiable. There are two types of qualifiers, constant and volatile, can appear only once in a declarations. Type qualifiers can appear with any type specifier however they cannot appear after the first comma in a multiple item declaration, to the following declaration are leg out.

E.g.: Type of volatile in v1;

constant in c;

KEYWORDS IN C

Every C word is classified as either a keyword or an identifier. All keywords have fixed meanings and these meanings cannot be changed. Keywords serve as basic building blocks for program statements. All keywords must be written in lower case.

Keywords

Auto	double	int	struct
break	else	long	switch
Case	enum	register	typedef
Char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
Do	if	static	while

TYPES OF STATEMENTS

1. Conditional statements: It is also called selection statement. This statement depends on the condition. If…else and switch statements always comes in conditional statements.
2. Iteration statements: These are used to run a particular blocks of statements repeatedly or in other words from a loop. for, while, do…while statements are came under this category.
3. Jump statements: These are used to make the flow of your statements from one point to another. break, continue, go to and return come under this statement.
4. Go to statements: It is a jump statement. It is used for jumping from one point to another point in your function.
5. Break statements: break statement when encountered within loop immediately terminates the loop by passing condition check and execution transfer to first statement after the loop.

6. Continue statement: It is analogues to break statement, unlike break which break the loop, continue statement forces the next execution of loop by passing any code in between. continue plays an important role in efficiently applying certain algorithms.

FORM OF STATEMENTS

if…else statement
if(condition) statement;
else statement;
switch statement
switch(input)
{ case value-1: statement-1;
break;
.
.
case value-n: statement-n;
break;
default : statement;
}
	for statement
	for(initialisation;condition;iteration)
	while statement
	while(condition)
	{
	Statement;
	}
	do…while statement
	do
	{
	Statement;
	}
	While(condition);

CATEGORY OF VARIOUS OPERATORS IN C LANGUAGE

Arithmetic operators: provides all the basic arithmetic operators. These can operate on any built- in data type allowed in C.

Relational operators: We often compare two quantities and depending on their relation, take certain decisions. The comparisons can be done with the help of relational operators.

Logical operators: The logical operators &&, || are used when we want to test more than one condition and make decision. An expression which combines two or more relational expression is termed as a logical expression.

Assignment operators: It is used to assign the result of an expression to a variable. We have seen that use all assignment operator, in addition, C has a set of 'shorthand 'assignment operator of the form

Op=Exp;

Increment and Decrement operators: C has two very useful operators not generally found in other languages. These are the increment and decrement operators ++ and --

Conditional operator: A ternary operator pair "?:" is available in C to construct conditional expression of the form EXP1?EXP2:EXP3;

Bitwise operators: 'C' has a distinction of supporting special operators known as bitwise operators for manipulation of data at bitlevel.Bitwise operators may not applied to float or double.

Special operators: 'C' supports some special operators of interest such as comma, size of operator, pointers and member selection operators.

CONCLUSION

Understood and studied about history, characteristics, key features, header file, main function, data types, qualifiers, keywords, various categories of operators and statements in C language.

ExpNo.2

SUM OF TWO INTEGERS

AIM:

To find sum of two numbers.

ALGORITHM:

Step1: Start

Step2: Read two values ‘A’ and ‘B’.

Step3: Compute SUM=A+B.

Step4: Display SUM.

Step5: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find sum of two numbers is executed successfully and verified the out put

ExpNo.3

AREA OF A CIRCLE

AIM:

To find the area of a circle when the radius is given

ALGORITHM:

Step1: Start

Step2: Read the radius of the circle in to the variable radius

Step3: Compute Area=3.14*radius*radius

Step4: Display the Area

Step5: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find area of a circle is executed successfully and verified the out put

ExpNo.4

LARGEST AMONG TWO NUMBERS

AIM:

To find largest among two numbers.

ALGORITHM:

Step1: Start

Step2: Read two integers ‘A’ and ‘B’.

Step3: If (A>B) then display “A is largest”.

Step4: Else display “B is largest”.

Step5: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find largest among two numbers is executed successfully and verified the out put

ExpNo.5

LARGEST AMONG 3 NUMBERS

AIM:

To find largest among three numbers.

ALGORITHM:

Step1: Start

Step2: Read two integers ‘A’, ‘B’and ‘C’

Step3: check If (A>B)

Step4: Then check if (A>C) then display “A is largest” else display “C is largest”.

Step5: Else check if (B>C) then display “B is largest” else display “C is largest”.

Step5: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find largest among three numbers is executed successfully and verified the out put

ExpNo.6

IMPLEMENTATION OF IF..ELSE LADDER

AIM:

To prepare mark list of SSLC using if...else ladder statement.

ALGORITHM:

Step1: Start

Step2: Read regno, name, and marks of physics, chemistry, and maths.

Step3: Set total= physics+chemistry+ maths.

Step4: Set percent=total/300*100

Step5: Print “School”, “SSLC Mark list”.

Step6: Print values of regno and name.

Step7: Print the marks of physics, chemistry, and maths.

Step8: print the value of total.

Step9: print the value of percent.

Step10: if (percent>=80), print “GRADE=Distinction”.

Step11: else if (percent>=60), print “GRADE=First class”.

Step12: else if (percent>=50), print “GRADE=Second class”.

Step13: else if (percent>=40), print “GRADE=Passed”.

Step14: else print “Failed”.

Step15: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement if…else ladder is executed successfully and verified the out put

ExpNo.7

IMPLEMENTATION OF SWITCH

AIM:

To prepare Electricity bill using switch statement.

ALGORITHM:

Step1: Start

Step2: Read consumer no, name, slab, units.

Step3: switch (slab).

Step4: case 1: amount=unit*.75; break.

Step5: case 2: amount=unit*1; break.

Step6: case 3: amount=unit*1.25; break.

Step7: case 4: amount=unit*1.50; break.

Step8: default: print “invalid slab no”.

Step9: Print “Electricity bill”

Step10: Print the values of consumer no, name, slab, units and amount.

Step11: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement switch is executed successfully and verified the out put

ExpNo.8

SUM OF DIGITS AND REVERSE OF A NUMBER

AIM:

To find sum of digits of a number and reverse.

ALGORITHM:

Step1: Start

Step2: Input a number ‘n’.

Step3: Store it in another variable ‘Temp’.

Step4: Set sum=0, rev=0.

Step5: Set d=Temp%10.

Step6: Set rev= (rev*10) +d. .

Step7: Set sum=sum+d.

Step8: Set Temp=Temp/10.

Step9: If Temp>0 go to step 5

Step10: Print the sum of digits and reverse.

Step11: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find sum of digits and reverse of a number is executed successfully and verified the out put

ExpNo.9

CHECK ARMSTRONG OR NOT

AIM:

To check the given number is Armstrong or not.

ALGORITHM:

Step1: Start.

Step2: Read a number ‘n’.

Step3: Set temp=0, sum=0.

Step4: Set d=n%10, sum=sum+d*d*d, n=n/10.

Step5: while (n>0), if (sum==temp).

Step6: Print “It is Armstrong number” else.

Step7: Print “It is not Armstrong number.

Step8: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to check the given number is Armstrong or not is executed successfully and verified the out put

ExpNo.10

IMPLEMENTATION OF FOR LOOP

AIM:

To print multiples of 10 between 100 and 200.

ALGORITHM:

Step1: Start.

Step2: Read ‘i’.

Step3: set sum=0.

Step4: for (i =100; i<=200; i=i+10)

Step5: Set sum=sum+i.

Step6: Display sum.

Step7: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement for loop is executed successfully and verified the out put

ExpNo.11

MULTIPLICATION TABLE

AIM:

To print multiplication tables up to a given number.

ALGORITHM:

Step1: Start.

Step2: Read n, l.

Step3: Print “Multiplication Table”.

Step4: for (i =1; i<=n; ++i)

Step5: for(j=1;j<=10;j++)

Step5: print j, i, i*j

Step6: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to print multiplication table is executed successfully and verified the out put

ExpNo.12

LARGEST FROM N NUMBERS

AIM:

Arrays- To find largest from ‘n’ numbers.

ALGORITHM:

Step1: Start.

Step2: Declare an array a[100].

Step3: Read limit and Input the numbers to array.

Step4: Set i=0, big=a[0].

Step5: for (i=0; i<n; ++i).

Step6: set a[i].

Step7: big=a[0].

Step8: for (i=0; i<n; ++i).

Step9: if (a[i]>big), big=a[i].

Step10: Print the value of big.

Step11: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find largest among n numbers is executed successfully and verified the out put

ExpNo.13

SEARCHING

AIM:

To search a given element from an array.

ALGORITHM:

Step1: Start.

Step2: Set an array a[100], count=0.

Step3: Read limit and Input the numbers to array.

Step4: for (i=0; i<n; ++i)

Step5: Display a[i].

Step6: Enter the number to be searched and store into variable ‘s’.

Step7: for(i=0;i<n;++i) ,if(s==a[i]) ,Print “Element is found”.

Step8: else print “Element is not found”.

Step9: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to search a given number is an array is executed successfully and verified the out put

ExpNo.14.

SORTING

AIM:

To sort N numbers of an array in ascending order.

ALGORITHM:

Step1: Start.

Step2: Set an array a[100].

Step3: Read limit and Input the numbers to array.

Step4: Consider the variables temp, i=0, j=i+1.

Step5: if (a[i]>a[j]), temp=a[i], a[i] =a[j], and a[j] =temp.

Step6: j=j+1.

Step7: if (j<=n-1) go to step5.

Step8: if (i< (n-1)), i=i+1, j= i+1, and go to step5

Step9: Print sorted array a[i].

Step10: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to sort a given array in ascending order is executed successfully and verified the out put

ExpNo.15

ADDING MATRICES

AIM:

To find sum of two matrices.

ALGORITHM:

Step1: Start

Step2: Declare three arrays a [100][100], b [100][100], c [100][100].

Step3: Read the arrays a and b.

Step4: Set i = 0, j = 0.

Step5: if (i <m)

Step6: if (j<n)

Step7: Set c [i][j]=a [i][j]+b [i][j], j=j+1.

Step8: go to step6

Step9: .end if.

Step10: Set=i+1.

Step11: go to step5

Step12: end if

Step13: Print array c.

Step14: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to add two matrices is executed successfully and verified the out put

ExpNo.16

PRODUCT OF MATRICES

AIM:

To find product of two matrices.

ALGORITHM:

Step1: Start

Step2: Declare three arrays a [100][100], b [100][100], c [100][100].

Step3: Read the arrays a and b.

Step4: Set i = 0, j = 0, k = 0.

Step5: if (i <m)

Step6: if (j<n)

Step7: Set c [i][j] = 0.

Step8: if (k < n)

Step9: Set c [i][j] = c [i][j]+ a[i][k] * b[k][j] , k=k+1.

Step10: go to step8.

Step11: end if.

Step12: Set j=j+1.

Step13: go to step6.

Step14: end if.

Step15: Set i=i+1.

Step16: go to step5.

Step17: end if.

Step18: Print the array c.

Step19: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find the product of two matrices is executed successfully and verified the out put

ExpNo.17

LENGTH OF A STRING

AIM:

To find length of a string.

ALGORITHM:

Step1: Start

Step2: Let the string read from the keyboard is ‘s1’.

Sep3: Set i=0.

Step4: if (s1 [i]! ='\0')

Step5: Set i=i+1.

Step6: go to step4.

Step7: end if.

Step8: Print (i-1)

Step9: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find length of a string is executed successfully and verified the out put

ExpNo.18

CONCATENATION OF 2 STRING

AIM:

To concatenate two strings.

ALGORITHM:

Step1: Start

Step2: Let the strings read from the keyboard are 's1' and 's2'.

Sep3: Set i=0, j=0.

Step4: if (s1 [i]! ='\0')

Step5: Set i=i+1.

Step6: go to step4.

Step7: end if.

Step8: if (s2 [j]! = '\0').

Step9: Set s1 [i] =s2 [j], i=i+1, j=j+1.

Step10: go to step8.

Step11: end if.

Step12: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to concatenate two string is executed successfully and verified the out put

ExpNo.19

COMPARE TWO STRING

AIM:

To compare two strings.

ALGORITHM:

Step1: Start

Step2: Let the strings read from the keyboard are ‘s1’ and ‘s2’.

Sep3: Set i=0, j=0, flag=0.

Step4: if (s1 [i]! =’\0’)

Step5: if s1 [i]! = s2[j].

Step6: Set flag=1.

Step7: break.

Step8: end if.

Step9: Set i =i+1, j =j+1.

Step10: go to step4.

Step11: end if.

Step12: if (flag==0)

Step13: Print “Strings are identical”.

Step14: else print “Strings are not identical.

Step15: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to compare two string is executed successfully and verified the out put

ExpNo.20

IMPLEMENTATION OF STRING FUNCTION

AIM:

String handling functions strlen, strcat, strcmp, strcat.

ALGORITHM:

Step1: Start

Step2: Declare the character arrays s1 [100], s2 [100], s3 [100].

Sep3: Input s1, s2.

Step4: Print length of s1 and s2.

Step5: if (s1! = s2)

Step6: Print “s1 and s2 not equal”.

Step7: Concatenate s1 and s2 to s1.

Step8: Print s1.

Step9: Copy s1 to s3.

Step10: Print s3.

Step11: else

Step12: Print “s1 and s2 equal.

Step13: Print length of s3.

Step14: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement string function is executed successfully and verified the out put

ExpNo.21

IMPLEMENTATION OF MATH FUNCTION

AIM:

To find area of triangle using mathematical function sqrt.

ALGORITHM:

Step1: Start

Step2: Read three sides of a triangle into ‘a’ , ‘b’, and ‘c’.

Sep3: compute s = (a +b +c)/2.

Step4: Compute Area = sqrt (s*(s-a)*(s-b)*(s-c))

Step5: Print Area.

Step6: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement mathematical function is executed successfully and verified the out put

ExpNo.22

IMPLEMENTATION OF USER DEFINED FUNCTION

AIM:

To learn User defined functions in C language.

ALGORITHM:

Step1: Start.

Step2: Display “Hello”.

Sep3: Calling the function printline ().

Step4: Called function printline (), print line ‘-‘and return to main function.

Step5: Calling the function welcome ().

Step6: Called the function welcome that print “Hello Good Morning” and return to main Function.

Step7: Call again function printline () as step4.

Step8: Calling the function printline1 ().

Step9: Called function printline1 (), set i=1.

Step10: if (i<=n) then
Step11: Print '- '.
Step12: Set i=i+1 repeat step7 to stp8 up to i>n.Then return to main function.
Step13: Calling function printx (ch, n).
Step14: Called function printx (set i=1).
Step15: if (i<=n) then
Step16: Print ch='x'.
Step17: Set i=i+1 repeat the step15 to step16up to i>n, return to main.
Step18: Calling function sum1 (a,b).
Step19: Called function sum1 (a,b) set c= a+b.
Step20: Print c and return to main.
Step21: Calling function sum2 (a,b).
Step22: Called function sum2 (a,b) set c=a+b.
Step23: Return the value of c.
Step24: Returning value stored to 'c'.
Step25: Print c.
Step26: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement user defined function is executed successfully and verified the out put

ExpNo.23

LARGEST AMOUNG 3 NUMBERS USING FUNCTION

AIM:

To find largest from three numbers using function.

ALGORITHM:

Step1: Start.
Step2: Input three numbers.
Sep3: Calling the function large (a,b,c).
Step4: Called function large (int a, int b, int c).
Step5: if (a>b) && (a>c) then.
Step6: Set l=a,return value of 'l'.
Step7: else if (b>c) then.
Step8: Set l=b, return value of l.
Step9: Returning value 'l' of function large, assigning with variable value.
Step10: Print value.
Step11: Stop.

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find largest among three numbers using function is executed successfully and verified the out put

ExpNo.24

FUNCTION WITH ARRAY AS ARGUMENTS

AIM:

To find largest from array elements using function.

ALGORITHM:

Step1: Start.

Step2: Input limit.

Sep3: Set i=0.

Step4: if i<=limit then.

Step5: Input number.

Step6: Set i=i+1 repeat step4 to step5 up to limit.

Step7: Calling the function largest (a,limit).

Step8: Called function largest (int b[],int l)

Step9: Set big=b [0], i=1.

Step10: if i<=l then.

Step11: if b [i]> big then.

Step12: Set big=b[i].

Step13: Set i=i+1 repeat step10 step12 up to i>l.

Step14: else return 'big' to main function.

Step15: Returning value is assigned to large.

Step16: Print large.

Step17: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find largest in an array using function is executed successfully and verified the out put

ExpNo.25

FUNCTION WITH 2DIMENSIONAL ARRAY AS ARGUMENT

AIM:

To find sum of elements in a matrix using function

ALGORITHM:

Step1: Start.

Step2: Input order m,n.

Sep3: Set i=0.

Step4: if i<m then,

Step5: Set j=0.

Step6: if j<n then

Step7: Input number a [i][j].

Step8: Set j=j+1 repeat step6 to step7 up to j>n

Step9: Set i=i+1 repeat step4 to step8 up to i>m.
Step10: Set i=0.
Step11: if i<m then.
Step12: Set j=0.
Step13: if j<n then.
Step14: Print number a [i][j].
Step15: Set j=j+1 repeat step13 to step 14 up to j>m
Step16: Set i=i+1 repeat step11 to step15 up to i>m
Step17: Calling the function sum (a [][], m,) .
Step18: Called function sum (int b [][], i nt m, int n)
Step19: Set sum=0,i=0.
Step20: if i<m then
Step21: Set j=0.
Step22: if j<n then
Step23: Set sum=sum+b[i][j]
Step24: Set j=j+1 repeat step22 to 23 up to j>n.
Step25: Set i=i+1 repeat step20 to 24 up to i>m.
Step26: return sum to main function.
Step27: return sum assigned to result.
Step28: Print result.
Step29: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find sum of two matrices using function is executed successfully and verified the out put

ExpNo.26

LENGTH OF A STRING USING FUNCTION

AIM:

To find length of a string using function.

ALGORITHM:

Step1: Start

Step2: Set i=0, count=0.

Sep3: Input string s[i].

Step4: Calling function mystrlen(s).

Step5: Called function mystrlen (char s[])

Step6: Set i=0, count=0.

Step7: if s[i]! =’\0’ then

Step8: Print s[i].

Step9: Set count=count+1.

Step10: Set i=i+1 repeat step6 to step8 up to s[i] ='\0'.

Step11: Return count to main function.

Step12: Print count.

Step13: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find length of a string using function is executed successfully and verified the out put

ExpNo.27

IMPLEMENTATION OF STRUCTURES

AIM:

To implement Student details using structure.

ALGORITHM:

Step1: Start.

Step2: Declaring structure student containing member data regno, name, physics, Chemistry, maths, total, percent.

Sep3: Declaring variable stud1.

Step4: Input structure member stud1.regno

Step5: Input structure member stud1.name

Step6: Input structure member stud1.physics

Step7: Input structure member stud1.chemistry

Step8: Input structure member stud1.maths

Step9: Set stud1.total= stud1.physics+ stud1. chemistry+ stud1.maths.

Step10: Set stud1.percent= stud1.total/3.

Step11: Print stud1.regno.

Step12: Print stud1.name

Step13: Print stud1. physics.

Step14: Print stud1. chemistry.

Step15: Print stud1.maths.

Step16: Print stud1.total.

Step17: Print stud1.percent.

Step18: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to find implement structure is executed successfully and verified the out put

ExpNo.28

CREATING EMPLOYEE DETAILS

AIM:

To implement employee details using structure.

ALGORITHM:

Step1: Start.

Step2: Declare structure employee

Step3: member data empno, gp, da, ta, total pay, netpay, pf, loan, empno [20], Designation [10]

Step4: Declare structure variable empl

Step5: input structure member empl.empno

Step6: input structure member empl.designation

Step7: input structure member empl.bp

Step8: input structure member empl.loan

Step9: set empl.da= (60/100*empl.bp)

Step10: set empl.ta= (5/100*empl.bp)

Step11: set empl.pf= (10/100*empl.bp)

Step12: set empl.totalpay = emp.bp+empl.da+empl.ta

Step13: set empl.netpay = empl.totalpay - (empl.da+empl.ta)

Step14: Print empl.empno

Step15: Print empl.empname

Step16: Print empl.designation

Step17: Print empl.totalpay

Step18: Print empl.netpay

Step19: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to create employee details using structure is executed successfully and verified the out put

ExpNo.29

CREATING CASH BILL USING STRUCTURE

AIM

Print cash bill using structure.

ALGORITHM:

Step1: Start
Step2: Declare structure sales bill with member data itemcode, itemname, qty, rate, tax, and totalprice.
Step3: Declare structure variable 'in'.
Step4: Input structure members.
Step5: Call computebill function
Step6: Called computebill (struct salesbill in)
Step7: Compute in. tax=in. rate* (in. tax/100)* in.qty
Step8: Compute in.totalprice =in. tax + (in. rate * in.qty).
Step9: Display itemcode, itemname, qty, rate, tax, and totalprice.
Step10: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to creating cash bill using structure is executed successfully and verified the out put

ExpNo.30

ILLUSTRIATION OF POINTERS

AIM:

Illustration of pointers.

ALGORITHM:

Step1: Start.
Step2: Declare char a= 'x', int b=25 and float c=12.6
Step3: Set char *p1= a, int *p2=b, float *p3= c
Step4: Print the character *p1.
Step5: Print the integer *p2.
Step6: Print the float *p3.
Step7: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to illustrate pointers is executed successfully and verified the out put

ExpNo.31

FILE HANDLING

AIM:

To find total no. of words, alphabets, digits, uppercase letters and lines in a file.

ALGORITHM:

Step1: Start.

Step2: Declare a file using the keyword FILE as *FP

Sep3: Declare c as character.

Step4: Declare w, i, u, d, io as integer

Step5: Assign w =i=u=io=d=0

Step6: Create a file using the function open in write mode.

Step7: Print “Enter the content of the file”.

Step8: while(c=getchar ()! =EOF)

Step9: Write characters in the file using putc function.
Step10: Close the file using fclose function.
Step11: Open the file in read mode
Step12: while(c= getc (*p)! =EOF)
Step13: if isalpha(c)
Step14: Set io++
Step15: end if
Step16: if isdigit(c)
Step17: Set d++
Step18: end if
Step19: if isspace(c)
Step20: Set w++
Step21: end if
Step22: if c= ‘\n’
Step23: Set i++
Step24: end if
Step25: if (isupper (c))
Step26: Set u++
Step27: end if
Step28: end while
Step29: Print “The file contains”
Step30: Print “number of words”, w.
Step31: Print “number of alphabet”, io.
Step32: Print “number of digits”, d.
Step33: Print “number of lines”, i
Step34: Print “number of block letters”, u
Step35: Close file
Step36: Stop

PROGRAM:

Program and output to be written by the students

RESULT:

Program to implement file handling is executed successfully and verified the out put