II SEMESTER

1901010 - ‘C’ PROGRAMMING LABORATORY

Regulation – 2019

Academic Year 2019 – 2020 (Even Semester)

Prepared by

| Mr. S.K. Saravanan AP (Sel. G) and Head |
| Mr. M. Asan Nainar AP (S.G)            |
| Mrs. M. Shanthi AP (S.G)               |
| Mr N.Sowrirajan A.P (OG)               |
SYLLABUS

1901010 C PROGRAMMING LABORATORY L T P C

0 0 4 2

OBJECTIVES:
- To develop programs in C using basic constructs.
- To develop applications in C using arrays and functions.
- To develop applications in C using Strings and Structures.

List of Programs

1. Programs using I/O statements and expressions.
2. Programs using decision-making constructs.
3. Write a program to find whether the given year is leap year or Not? (Hint: not every centurion year is a leap. For example 1700, 1800 and 1900 is not a leap year)
4. Write a program to perform the Calculator operations, namely, addition, subtraction, multiplication, division and square of a number.
5. Check whether a given number is Armstrong number or not?
6. Check whether a given number is odd or even?
7. Write a program to perform factorial of a number.
8. Write a C program to find out the average of 4 integers.
9. Show how to display array elements using two dimensional array.
10. Write a C program to perform swapping using function.
11. Display all prime numbers between two intervals using functions.
12. Reverse a sentence using recursion.
13. Write a program in C to get the largest element of an array using the function.
14. Write a C program to concatenate two string.
15. Write a C program to find the length of String.
16. Find the frequency of a character in a string.
17. Write a C program to Store Student Information in Structure and Display it.
18. The annual examination is conducted for 10 students for five subjects. Write a program to read the data and determine the following:
   (a) Total marks obtained by each student.
   (b) The highest marks in each subject and the marks of the student who secured it.
   (c) The student who obtained the highest total marks.

TOTAL: 60 PERIODS

COURSE OUTCOMES:

Upon completion of the course, the students will be able to
- Develop C programs for simple applications making use of basic constructs, arrays and strings.
- Develop C programs involving functions, recursion, pointers, and structures.
- Design applications using sequential and random access file processing.
# INDEX

<table>
<thead>
<tr>
<th>S.No</th>
<th>Topic</th>
<th>Page No</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Programs using I/O statements and expressions</td>
<td>4</td>
</tr>
<tr>
<td>2.</td>
<td>Programs using decision-making constructs.</td>
<td>11</td>
</tr>
<tr>
<td>3.</td>
<td>Write a program to find whether the given year is leap year or Not? (Hint: not every centurion year is a leap. For example 1700, 1800 and 1900 is not a leap year)</td>
<td>16</td>
</tr>
<tr>
<td>4.</td>
<td>Write a program to perform the Calculator operations, namely, addition, subtraction, multiplication, division and square of a number.</td>
<td>18</td>
</tr>
<tr>
<td>5.</td>
<td>Check whether a given number is Armstrong number or not?</td>
<td>20</td>
</tr>
<tr>
<td>6.</td>
<td>Check whether a given number is odd or even?</td>
<td>22</td>
</tr>
<tr>
<td>7.</td>
<td>Write a program to perform factorial of a number.</td>
<td>23</td>
</tr>
<tr>
<td>8.</td>
<td>Write a C program to find out the average of 4 integers.</td>
<td>25</td>
</tr>
<tr>
<td>9.</td>
<td>Show how to display array elements using two dimensional array.</td>
<td>27</td>
</tr>
<tr>
<td>10.</td>
<td>Write a C program to perform swapping using function.</td>
<td>29</td>
</tr>
<tr>
<td>11.</td>
<td>Display all prime numbers between two intervals using functions.</td>
<td>31</td>
</tr>
<tr>
<td>12.</td>
<td>Reverse a sentence using recursion.</td>
<td>34</td>
</tr>
<tr>
<td>13.</td>
<td>Write a program in C to get the largest element of an array using the function.</td>
<td>35</td>
</tr>
<tr>
<td>14.</td>
<td>Write a C program to concatenate two string.</td>
<td>37</td>
</tr>
<tr>
<td>15.</td>
<td>Write a C program to find the length of String.</td>
<td>39</td>
</tr>
<tr>
<td>16.</td>
<td>Find the frequency of a character in a string.</td>
<td>40</td>
</tr>
<tr>
<td>17.</td>
<td>Write a C program to Store Student Information in Structure and Display it.</td>
<td>42</td>
</tr>
<tr>
<td>18.</td>
<td>The annual examination is conducted for 10 students for five subjects. Write a program to read the data and determine the following: (a) Total marks obtained by each student. (b) The highest marks in each subject and the marks of the student who secured it. (c) The student who obtained the highest total marks</td>
<td>44</td>
</tr>
</tbody>
</table>
Ex.No: 1 Programs using I/O statements and expressions.

1a: Display the statement using simple printf() statement

Aim:
To write a C program using printf() statement.

Algorithm:
1. Start the program
2. Use the printf() function for output statement
3. Display the result
4. Stop the program.

Program:
#include <stdio.h>
int main()
{
    // Displays the string inside quotations
    printf("C Programming");
    return 0;
}

Output
C Programming

1b: Display the integer value

Aim:
To write a C program using printf() statement.

Algorithm:

1. Start the program
2. Initialize the integer variable.
3. Use the printf() function for output statement
4. Display the result
5. Stop the program.
**Program:**
```
#include <stdio.h>
int main()
{
    int testInteger = 5;
    printf("Number = %d", testInteger);
    return 0;
}
```

**Output**
Number = 5

**1c: Display the float and double value**

**Aim:**
To write a C program using printf() statement.

**Algorithm:**

1. Start the program
2. Initialize the float and double variable.
3. Use the printf() function for output statement
4. Display the result
5. Stop the program.

**Program:**
```
#include <stdio.h>
int main()
{
    float number1 = 13.5;
    double number2 = 12.4;

    printf("number1 = %f\n", number1);
    printf("number2 = %lf", number2);
    return 0;
}
```
1d: Display the character

Aim:
   To write a C program using printf() statement.

Algorithm:

1. Start the program
2. Initialize the character variable.
3. Use the printf() function for output statement
4. Display the result
5. Stop the program.

Program:
#include <stdio.h>
int main()
{
    char chr = 'a';
    printf("character = %c.", chr);
    return 0;
}

Output
character = a

1e: Integer Input/Output

Aim:
   To write a C program using printf() and scanf() statement.

Algorithm:

1. Start the program
2. Get the integer variable using scanf() function
3. Use the printf() function for output statement
4. Display the result
5. Stop the program.

Program:
#include <stdio.h>
int main()
{
    int testInteger;
    printf("Enter an integer: ");
    scanf("%d", &testInteger);
    printf("Number = %d", testInteger);
    return 0;
}

Output
Enter an integer: 4
Number = 4

1f. Display the Character I/O statement

Aim:
To write a C program using printf() and scanf() statement.

Algorithm:

1. Start the program
2. Get the character variable using scanf() function
3. Use the printf() function for output statement
4. Display the result
5. Stop the program.

Program:
#include <stdio.h>
int main()
{
    char chr;
    printf("Enter a character: ");
    scanf("%c", &chr);
    printf("You entered %.c: ", chr);
    return 0;
}

Output
Enter a character: g
1g. Display I/O statement using multiple values

Aim:
To write a C program using printf() and scanf() statement.

Algorithm:
1. Start the program
2. Get the integer and floating point variable using scanf() function
3. Use the printf() function for output statement
4. Display the result
5. Stop the program

Program:
#include <stdio.h>
int main()
{
    int a;
    float b;
    printf("Enter integer and then a float: ");
    // Taking multiple inputs
    scanf("%d%f", &a, &b);
    printf("You entered %d and %f", a, b);
    return 0;
}

Output
Enter integer and then a float: -3
3.4
You entered -3 and 3.400000

1h (a). Expressions: Type Conversion

Aim:
To write a C program on “Type Conversion” using expression.

Algorithm:
1. Start the program
2. Initialize the char variable.
3. Declare the integer variable.
4. Add the char variable and assign it to integer variable. (Implicit type conversion).
5. Use the printf() function for output statement
6. Display the result
7. Stop the program

Program:
#include <stdio.h>
void main()
{
char a = 100;
char b = 28;
int c;
char d;

c = a + b;
d = a + b;
printf( "c = %d, d = %d\n", c, d );
}

Output:
c = 128, d = -128

1h (b) Expressions : Explicit Type Conversion

Aim:
To write a C program on “Explicit Type Conversion” using expression.

Algorithm:
1. Start the program
2. Initialize the integer and double variable.
3. Assign integer variable to double variable using explicit type conversion.
4. Use the printf() function for output statement
5. Display the result
6. Stop the program

Program:
#include <stdio.h>
void main()
{
    int i = 5;
    double d = i / 6;
    printf( "d = %fn", d );
        d = (double) i / 6;
    printf( "d = %fn", d );
}

**Output:**
d = 0.000000 d = 0.833333

**Result:**
Thus the C program using I/O statement and expressions has been verified and executed successfully.

Ex.No: 2 Programs using decision-making constructs.

If statement
If statement is used for branching when a single condition is to be checked. The condition enclosed in if statement decides the sequence of execution of instruction. If the condition is true, the statements inside if statement are executed, otherwise they are skipped.

2a: C program to print the square of a number if it is less than 10.

Aim:
To write a C program to print the square of a number if it is less than 10 using if statement.

Algorithm:
1. Start the program
2. Get the number
3. Check the number if it is less than 10 using if statement.
4. Display the result
5. Stop the program.

Program:
#include<stdio.h>
int main()
{
 int n;
 printf("Enter a number: ");
 scanf("%d", &n);
 if(n<10)
  {
   printf("%d is less than 10\n",n);
   printf("Square = %d\n", n*n);
  }
 return 0;
}

Output:
Enter a number 8
8 is less than 10

2b. if ... else statement
if ... else statement is a two way branching statement. It consists of two blocks of statements each enclosed inside if block and else block respectively. If the condition inside if statement is true,
statements inside if block are executed, otherwise statements inside else block are executed. Else block is optional and it may be absent in a program.

**C program to find if a number less than 10.**

**Aim:**
To write a C program to find a larger of two numbers using if statement.

**Algorithm:**

1. Start the program
2. Get the numbers a and b
3. Check the number if it is less than 10 or greater than 10 using if statement.
4. Display the result
5. Stop the program.

**Program:**

```c
#include<stdio.h>
int main()
{
    int n;
    printf("Enter two numbers:");
    scanf("%d%d",&a,&b);
    if(a>b)
        printf("%d is greater",a);
    else
        printf("%d is greater ",b);
    return 0;
}
```

**Output**

Enter two numbers:18 25
25 is greater

---

**2c. Example of if ... else if ... else statement**
C program to find if a number is negative, positive or zero.

Aim:
To write a C program to find if a number is negative, positive or zero using if ... else if ... else statement.

Algorithm:
1. Start the program
2. Get the number
3. Check the number if it is negative, positive or equal to using if statement.
4. If the number is < 0 print number is negative, else if the number is > 0 print it is positive else the number = 0.
5. Display the result
6. Stop the program.

Program:

#include<stdio.h>
int main()
{
int n;
printf("Enter a number:");
scanf("%d", &n);
if(n<0)
printf("Number is negative");
else if(n>0)
printf("Number is positive");
else
printf("Number is equal to zero");
return 0;
}

Output
Enter a number: 109
Number is positive
Enter a number: -56
Number is negative
Enter a number: 0
Number is equal to zero

2d. Switch Statement
Switch case can be considered as simplified version of if statement. When there are large number of conditions to be tested, it is difficult to use if statement as the number of repeated if statements may cause confusion and makes the program unreadable. So, switch case is preferred in such cases to simplify programmer’s job and increases code readability.

C program to Check if entered alphabet is vowel or a consonant.

Aim:
To write a C program to check if entered alphabet is vowel or a consonant using switch case.

Algorithm:
1. Start the program
2. Get the character for choice
3. Give the multiple choices using case statement whether one of the choices are in the consonants and print the same.
4. In the default case print that it is a consonant.
5. Display the result
6. Stop the program.

Program:

#include <stdio.h>
int main()
{
    char alphabet;
    printf("Enter an alphabet:");
    scanf("%c",&alphabet);
    switch(alphabet)
    {
        case 'a':
            printf("Alphabet a is a vowel.\n");
        case 'e':
            printf("Alphabet e is a vowel.\n");
        case 'i':
            printf("Alphabet i is a vowel.\n");
        case 'o':
            printf("Alphabet o is a vowel.\n");
        case 'u':
            printf("Alphabet u is a vowel.\n");
        default:
            printf("You entered a consonant.\n");
    }
}
return 0;

Output

Enter an alphabet:i
Alphabet i is a vowel.
Alphabet o is a vowel.
Alphabet u is a vowel.
You entered a consonant.

Result:
Thus the C program using decision-making construct has been verified and executed successfully.
Ex.No: 3. Write a program to find whether the given year is leap year or Not? (Hint: not every centurion year is a leap. For example 1700, 1800 and 1900 is not a leap year)

Aim:
To write a C program to find whether the given year is leap year or not using switch case.

Algorithm:
1. Start the program
2. Get the year to find whether it’s a leap year
3. Check if the year % 4 == 0 and year % 100 == 0 and year %400 == 0 for the leap year, else it’s not a leap year
4. Display the result
5. Stop the program.

Program:

```c
#include <stdio.h>
int main()
{
    int year;
    printf("Enter a year: ");
    scanf("%d",&year);
    if(year%4 == 0)
    {
        if( year%100 == 0)
        {
            // year is divisible by 400, hence the year is a leap year
            if ( year%400 == 0)
                printf("%d is a leap year.", year);
            else
                printf("%d is not a leap year.", year);
        }
        else
            printf("%d is a leap year.", year);
    }
    else
        printf("%d is not a leap year.", year);

    return 0;
}
```

return 0;
}

Output
Enter a year: 1900
1900 is not a leap year.

Result:
Thus the C program using to find whether the given year is leap or not year has been successfully verified and executed.
Ex.No: 4. Write a program to perform the Calculator operations, namely, addition, subtraction, multiplication, division and square of a number.

Aim:
To write a C program to perform calculator operation using the switch case.

Algorithm:
1. Start the program
2. Get the first and second number
3. Choose the choice for addition, subtraction, multiplication, division and square of a number using switch case.
4. If the choice is addition add operation is performed and other operations are performed respectively till square of the number.
5. If the choice is default it shows “invalid operation”.
5. Display the result
6. Stop the program.

Program:

#include <stdio.h>

int main()
{
  int num1,num2;
  float result 0.0 ;
  char ch;  //to store operator choice

  printf("Enter first number: ");
  scanf("%d",&num1);
  printf("Enter second number: ");
  scanf("%d",&num2);

  printf("Choose operation to perform (+,-,\,*\,\/%): ");
  scanf(" %s",&ch);

  result=0;
  switch(ch)
  {
    case '+':
      result=num1+num2;
      break;
    case '-':
      result=num1-num2;
      break;
    case '*':
      result=num1*num2;
      break;
    case '/':
      result=num1/num2;
      break;
    case '%':
      result=num1%num2;
      break;
    default:
      printf("invalid operation");
      break;
  }

  printf("Result: ");
  printf("%f",result);

  return 0;
}
```
break;

case '*':
    result = num1 * num2;
    break;

case '/':
    result = (float)num1 / (float)num2;
    break;

case 's':
    result = num1 * num1;
    break;

case '
':
    printf("Invalid operation.\n");
    default:
        printf("Result: %d %c %d = %f\n", num1, ch, num2, result);
        return 0;
    }

printf("Result: %d %c %d = %f\n", num1, ch, num2, result);
return 0;
}

Output

First run:
Enter first number: 10
Enter second number: 20
Choose operation to perform (+,-,*,/,%): +
Result: 10 + 20 = 30.000000

Second run:
Enter first number: 10
Enter second number: 3
Choose operation to perform (+,-,*,/,%): /
Result: 10 / 3 = 3.333333

Third run:
Enter first number: 10
Enter second number: 3
Choose operation to perform (+,-,*,/,%): >
Invalid operation.
Result: 10 > 3 = 0.000000

Result:

Thus the C program using to perform calculator operation has been successfully verified and executed.

Ex.No: 5. Check whether a given number is Armstrong number or not?
```
Aim:
To write a C program to check whether a given number is Armstrong number or not using switch case.

Algorithm:
1. Start the program
2. read number
3. set result=0 and originalnumber=number
4. reminder=originalnumber%10
5. result+=(reminder*reminder*reminder)
6. number=originalnumber/10
7. repeat steps 4 to 6 until number > 0
8. if result = number
9. display number is armstrong
10. else
11. display number is not armstrong
12. stop the program

Program:
#include <stdio.h>

int main()
{
    int number, originalNumber, remainder, result = 0;
    printf("Enter a three digit integer: ");
    scanf("%d", &number);
    originalNumber = number;
    while (originalNumber != 0)
    {
        remainder = originalNumber%10;
        result += remainder*remainder*remainder;
        originalNumber /= 10;
    }
    if(result == number)
        printf("%d is an Armstrong number.",number);
    else
        printf("%d is not an Armstrong number.",number);
    return 0;
}
Output

Enter a three digit integer: 371
371 is an Armstrong number.

Result:

Thus the C program to check whether a given number is armstrong or not has been successfully verified and executed.
Ex.No: 6.   Write a C program to check whether a given number is odd or even.

Aim:
To write a C program to find an odd or even number.

Algorithm:
1. Start the program
2. Get the number
3. Check the number if it is odd or even using if statement.
4. If the number is even check the condition as n%2 == 0 else it is even.
5. Display the result
6. Stop the program.

Program:

```c
#include <stdio.h>
int main()
{
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number);
    // True if the number is perfectly divisible by 2
    if(number % 2 == 0)
        printf("%d is even.", number);
    else
        printf("%d is odd.", number);
    return 0;
}
```

Output
Enter an integer: -7
-7 is odd.

Result:
Thus the C program to find an odd or even number has been successfully verified and executed.

Ex.No:7.   Write a program to perform factorial of a number.
Aim:
To write a C program to find a factorial of a given number.

Algorithm:

1. Start the program
2. Get the number
3. If the number < 0 print “Error for finding a factorial”
4. Else Initialize variables
   factorial ← 1
   i ← 1
5: Read value of n
6: Repeat the steps until i=n
6.1: factorial ← factorial*i
6.2: i ← i+1
7: Display factorial
8 Stop the program.

Program:

```c
#include <stdio.h>
int main()
{
    int n, i;
    long factorial = 1;
    printf("Enter an integer: ");
    scanf("%d",&n);
    // show error if the user enters a negative integer
    if (n < 0)
        printf("Error! Factorial of a negative number doesn't exist.");
    else
    {
        for(i=1; i<=n; ++i)
        {
            factorial *= i;    // factorial = factorial*i;
        }
        printf("Factorial of %d = %lu", n, factorial);
    }
    return 0;
}
```

Output
Enter an integer: 10
Factorial of 10 = 3628800

Result:
Thus the C program to find the factorial of a given number has been successfully executed and verified.

Ex.No: 8. Write a C program to find out the average of 4 integers.
Aim:
To write a C program to find average of 4 integers.

Algorithm:
1. Start the program
2. Get the integer numbers using an array.
3. Use the for loop to iterate the values of four numbers and assign the added values in the variable called “sum”.
4. Find the average of four numbers.
5. Display the result
6. Stop the program.

Program:

```c
#include<stdio.h>
int main()
{
    int avg=0;
    int sum =0;
    int x=0;

    /* Array- declaration – length 4*/
    int num[4];

    /* We are using a for loop to traverse through the array 
     * while storing the entered values in the array 
     */
    for(x=0; x<4;x++)
    {
        printf("Enter number %d \n",(x+1));
        scanf("%d",&num[x]);
    }
    for(x=0; x<4;x++)
    {
        sum=sum+num[x];
    }

    avg= sum/4;
    printf("Average of entered number is: %d\n",avg);
    return0;
}
```

Output:

25
Enter number 1
10
Enter number 2
20
Enter number 3
30
Enter number 4
40
Average of entered number is: 25

**Result:**
Thus the C program to find average of 4 integers has been successfully executed and verified.
Ex.No:9. Show how to display array elements using two dimensional arrays.

Aim:

To write a C program to display the array elements using two dimensional arrays.

Algorithm:

1. Start the program
2. Declare the integer using a two dimensional array.
3. Use the for loop get the values and display the same
4. Print the format as 2 dimensional matrix.
5. Display the result
6. Stop the program.

Program:

```c
#include<stdio.h>
int main(){
    /* 2D array declaration*/
    int disp[2][3];
    /*Counter variables for the loop*/
    int i, j;
    for(i=0;i<2;i++){
        for(j=0;j<3;j++){
            printf("Enter value for disp[%d][%d]:",i, j);
            scanf("%d", &disp[i][j]);
        }
    }
    //Displaying array elements
    printf("Two Dimensional array elements:\n");
    for(i=0;i<2;i++){
        for(j=0;j<3;j++){
            printf("%d ",disp[i][j]);
            if(j==2){
                printf("\n");
            }
        }
    }
    return0;
}
```
**Output:**

Enter value for disp[0][0]: 1
Enter value for disp[0][1]: 2
Enter value for disp[0][2]: 3
Enter value for disp[1][0]: 4
Enter value for disp[1][1]: 5
Enter value for disp[1][2]: 6
TwoDimensional array elements:
123
456

**Result:**

Thus the C program to display array elements using two dimensional arrays has been successfully executed and verified.

---

**Ex.No:10.** Write a C program to perform swapping using function.
**Aim:**

To write a C program to perform swapping using function.

**Algorithm:**

1. Start the program
2. Declare and get the two integer variables a and b.
3. call the swap() function
   3.1 In swap definition use the temporary variable and assign temp = a
   3.2 a = b
   3.3 b = temp
4. Print the a and b value.
5. Display the result
6. Stop the program.

**Program:**

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

void main()
{
    void swap(int,int);
    int a,b,r;
    clrscr();
    printf("enter value for a&b: ");
    scanf("%d%d",&a,&b);
    swap(a,b);
    getch();
}

void swap(int a,int b)
{
    int temp;
    temp=a;
    a=b;
    b=temp;
    printf("after swapping the value for a & b is : %d %d",a,b);
}
```

**Output:**

enter value for a&b: 4
after swapping the value for a &b : 5 4

Result:

Thus the C program to perform swapping using function has been successfully executed and verified.
Ex.No: 11. Write a C program to display all prime numbers between two intervals using functions.

Aim:

To write a C program to display all prime numbers between two intervals using functions.

Algorithm:

1. Start the program
2. Enter the lower and upper limit of prime number list.
3. Call `printPrimes(lowerLimit, upperLimit);`
   
   3.1 Check while (lowerLimit<=upperLimit) do
      
      3.2 Increase the lowerlimit and check using the while loop. If the number is divisible by 1 and self then it is prime
      
      3.3 Repeat the step until the upper limit is reached
4. Print the prime numbers in the interval.
5. Display the result
6. Stop the program.

Program:

```c
#include <stdio.h>
/* Function declarations */
int isPrime(int num);
void printPrimes(int lowerLimit, int upperLimit);

int main()
{
    int lowerLimit, upperLimit;
    printf("Enter the lower and upper limit to list primes: ");
    scanf("%d%d", &lowerLimit, &upperLimit);

    // Call function to print all primes between the given range.
    printPrimes(lowerLimit, upperLimit);
    return 0;
}

int isPrime(int num)
{
    return 0;
}

void printPrimes(int lowerLimit, int upperLimit)
{
    // Function to print all primes between the given range.
}
```

31
```c
void printPrimes(int lowerLimit, int upperLimit)
{
    printf("All prime number between %d to %d are: ", lowerLimit, upperLimit);
    while(lowerLimit <= upperLimit)
    {
        // Print if current number is prime.
        if(isPrime(lowerLimit))
        {
            printf("%d, ", lowerLimit);
        }
        lowerLimit++;
    }
}

/**
 int isPrime(int num)
 {
     int i;
     for(i = 2; i <= num/2; i++)
     {
         /*
             * If the number is divisible by any number
             * other than 1 and self then it is not prime
             */
         if(num % i == 0)
         {
```
return 0;

}

}

return 1;

}

**Output**
Enter the lower and upper limit to list primes 20 50
Prime numbers between 20 and 50 are: 23 29 31 37 41 43 47

**Result:**

Thus the C program to display all prime numbers between two intervals using functions has been successfully executed and verified.

---

**Ex.No:12.** Write a C program to reverse a sentence using recursion.
Aim:

To write a C program to reverse a sentence using recursion

Algorithm:

1. Start the program
2. Read the sentence
3. Declare the reverse sentence () function.
4. Read the sentence reversely until newline is reached as (if c!='\n')
5. Display the result
6. Stop the program

Program:

#include <stdio.h>
VoidreverseSentence();
int main()
{
    printf("Enter a sentence: ");
    reverseSentence();
    return 0;
}
VoidreverseSentence()
{
    char c;
    scanf("%c", &c);
    if( c != 'n')
    {
        reverseSentence();
        printf("%c",c);
    }
}

Output

Enter a sentence: margorpemosewa
awesome program

Result:

Thus the C program to reverse a sentence using recursion has been successfully executed and verified.
Ex.No: 13. Write a program in C to get the largest element of an array using the function.

Aim:

To write a C program to get the largest element of an array.

Algorithm:

1. Start the program
2. Read the number of array elements as size.
3. Read array elements array i= 0,1,2,3,…n-1
4. Assume first element array[0] to be maximum
5. Compare each element array[i] with maximum
6. If( maximum > array[i]) then maximum = array[i]
7. Display the result.
8. Stop the program.

Program:

```c
int main()
{
    int array[100], maximum, size, c, location = 1;

    printf("Enter the number of elements in array\n");
    scanf("%d", &size);

    printf("Enter %d integers\n", size);

    for (c = 0; c < size; c++)
        scanf("%d", &array[c]);

    maximum = array[0];

    for (c = 1; c < size; c++)
    {
        if (array[c] > maximum)
        {
            maximum = array[c];
            location = c+1;
        }
    }

    printf("Maximum element is present at location %d and it's value is %d.\n", location, maximum);
    return 0;
}
```

Output:

35
Enter the number of elements in array
5
Enter 5 integers
4
5
6
8
2
Maximum element present at location 4 and its value is 8

**Result:**
Thus the C program to get the largest element of an array using the function has been successfully executed and verified.
Ex.No:14. Write a C program to concatenate two strings.

Aim:

To write a C program to concatenate two strings.

Algorithm:

1. Start the program
2. Read the strings
3. Use the for loop to get the first sentence and use the second for loop for second sentence and increment the counter i.
4. Now assign the sentence s2[j] to s1[i] until the string reaches the condition s2[j]!=0 and the sentences are concatenated
5. Display the result
6. Stop the program

Program:

```c
#include <stdio.h>

int main()
{
    char s1[100], s2[100], i, j;
    printf("Enter first string: ");
    scanf("%s", s1);
    printf("Enter second string: ");
    scanf("%s", s2);
    // calculate the length of string s1
    // and store it in i
    for(i = 0; s1[i] != '\0'; ++i);
    for(j = 0; s2[j] != '\0'; ++j, ++i)
    {
        s1[i] = s2[j];
    }
    s1[i] = '\0';
    printf("After concatenation: %s", s1);
    return 0;
}
```

Output
Enter first string: hi
Enter second string: welcome
After concatenation: Hi welcome

**Result:**

Thus the C program to concatenate two strings has been successfully executed and verified.
Ex.No: 15. Write a C program to find the length of String.

Aim:
To write a C program to find the length of the string.

Algorithm:
1. Start the program
2. Read the sentence
3. Use the for loop and test the condition that s[i] != '\0'
4. Use the i value to display the length
5. Stop the program

Program:
```c
#include <stdio.h>
int main()
{
    char s[1000];
    int i;
    printf("Enter a string: ");
    scanf("%s", s);
    for(i = 0; s[i] != '\0'; ++i);
    printf("Length of string: %d", i);
    return 0;
}
```

Output:
Enter a string: Program
Length of string: 7

Result:
Thus the C program to find the length of String has been successfully executed and verified.
Ex.No: 16. Find the frequency of a character in a string.

Aim:

To write a C program to find the frequency of a character in the string.

Algorithm:

1. Start the program.
2. Read the sentence
3. Calculate the frequency of the character repeated in the string.
4. Read each character of string to find its frequency
5. Display the result
6. Stop the program

Program:

```c
#include <stdio.h>
int main()
{
    Charstr[1000], ch;
    int i, frequency = 0;
    printf("Enter a string: ");
    gets(str);
    printf("Enter a character to find the frequency: ");
    scanf("%c",&ch);
    for(i = 0; str[i] != '\0'; ++i)
    {
        if(ch == str[i])
            ++frequency;
    }
    printf("Frequency of %c = %d", ch, frequency);
    return 0;
}
```

Output
Enter a string: This website is awesome.
Enter a character to find the frequency: e
Frequency of e = 4

**Result:**
Thus the C program to find the frequency of a character in a string has been successfully executed and verified.

Ex.No: 17. **Write a C program to Store Student Information in Structure and Display it.**
Aim:

To write a C program to store the student information using structure.

Algorithm:

1. Start the program
2. Declare the variables in the structure data type.
3. Read the values of the variables in the structure.
4. Display the result
5. Stop the program

Program:

```c
#include <stdio.h>
struct student
{
    char name[50];
    int roll;
    float marks;
} s[10];
int main()
{
    int i;
    printf("Enter information of students:\n");
    // storing information
    for(i=0; i<10; ++i)
    {
        s[i].roll = i+1;
        printf("nameFor roll number%d,\n",s[i].roll);
        printf("Enter name: ");
        scanf("%s",s[i].name);
        printf("Enter marks: ");
        scanf("%f",s[i].marks);
        printf("\n");
    }
    printf("Displaying Information:\n\n");
    // displaying information
    for(i=0; i<10; ++i)
    {
        printf("Roll number: %d\n",i+1);
        printf("Name: ");
```
puts(s[i].name);
printf("Marks: %.1f",s[i].marks);
printf("\n");
}
return 0;
}

**Output:**

Enter information of students:

For roll number 1,
Enter name: Tom
Enter marks: 98

For roll number 2,
Enter name: Jerry
Enter marks: 89

.. Displaying Information:

Roll number: 1
Name: Tom
Marks: 98

Roll number: 2
Name: Jerry
Marks: 89

**Result:**
Thus the C program to store Student Information in Structure has been successfully executed and verified.

**Ex.No: 18.** The annual examination is conducted for 10 students for five subjects. Write a program to read the data and determine the following:
(a) Total marks obtained by each student.
(b) The highest marks in each subject and the marks of the student who secured it.
(c) The student who obtained the highest total marks.

Aim:
To write a C program to get various details regarding the marks obtained by the students.

Algorithm:
1. Start the program
2. Read the inputs of the student details.
3. Declare the structure and its size along with the variables.
4. Read the values for the variables.
5. Display the result
6. Stop the program

Program:
```c
#include<stdio.h>
#define SIZE 50

struct student {
    char name[30];
    int rollno;
    int sub[3];
};

void main() {
    int i, j, max, count, total, n, a[SIZE], ni;
    struct student st[SIZE];
    clrscr();

    printf("Enter how many students: ");
    scanf("%d", &n);

    /* for loop to read the names and roll numbers*/
    for (i = 0; i < n; i++) {
        printf("Enter name and roll number for student %d: ", i);
        scanf("%s", &st[i].name);
        scanf("%d", &st[i].rollno);
    }

    /* for loop to read ith student's jth subject*/
    for (i = 0; i < n; i++) {
        for (j = 0; j <= 2; j++) {
            printf("Enter marks of student %d for subject %d: ", i, j);
        }
    }

    /* display the result */
    for (i = 0; i < n; i++) {
        for (j = 0; j <= 2; j++) {
            total = st[i].sub[j] + total;
        }
        printf("Total marks of student %d: %d\n", i, total);
    }
```
```c
scanf("%d", &st[i].sub[j]);
}
}
/* (i) for loop to calculate total marks obtained by each student*/
for (i = 0; i< n; i++) {
    total = 0;
    for (j = 0; j < 3; j++) {
        total = total + st[i].sub[j];
    }
    printf("Total marks obtained by student %s are %d\n", st[i].name, total);
    a[i] = total;
}
/* (ii) for loop to list out the student's roll numbers who have secured the highest marks in each subject */
/* roll number who secured the highest marks */
for (j = 0; j < 3; j++) {
    max = 0;
    for (i = 0; i< n; i++) {
        if (max < st[i].sub[j]) {
            max = st[i].sub[j];
            ni = i;
        }
    }
    printf("Student %s got maximum marks = %d in Subject : %d\n", st[ni].name, max, j);
}
max = 0;
for (i = 0; i< n; i++) {
    if (max < a[i]) {
        max = a[i];
        ni = i;
    }
}
printf("%s obtained the total highest marks.\n", st[ni].name);
getch();
```
Output:
Enter how many students: 2
Enter name and roll number for student 0 : Pritesh 1
Enter name and roll number for student 1 : Suraj 2
Enter marks of student 0 for subject 0 : 90
Enter marks of student 0 for subject 1 : 89
Enter marks of student 0 for subject 2 : 78
Enter marks of student 1 for subject 0 : 67
Enter marks of student 1 for subject 1 : 88
Enter marks of student 1 for subject 2 : 99
Total marks obtained by student Pritesh are 257
Total marks obtained by student Suraj are 254
Student Pritesh got maximum marks = 90 in Subject : 0
Student Pritesh got maximum marks = 89 in Subject : 1
Student Suraj got maximum marks = 99 in Subject : 2
Pritesh obtained the total highest marks.

Result:
Thus the C program to get various details regarding the marks obtained by the students has been successfully executed and verified.

Additional C Programs for exercise
1. Program to find the row sum and column sum of a given .

#include<stdio.h>

#include<conio.h>
void main()
{
    int mat[10][10];
    int i,j;
    int m,n;
    int sumrow,sumcol;
    clrscr();
    printf("\nTO FIND THE ROW SUM AND COLUMN SUM OF A GIVEN MATRIX: ");
    printf("\n-- ---- --- --- --- --- ------ --- -- - ----- ------: ");
    printf("\nEnter the order of matrix: ");
    scanf("%d%d",&m,&n);
    printf("\nEnter elements of a matrix:");
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
            scanf("%d",&mat[i][j]);
    }
    printf("\n\nOUTPUT: ");
    printf("\n--------------------------------------");
    for(i=0;i<m;i++)
    {
        sumrow=0;
        for(j=0;j<n;j++)
            sumrow=sumrow+mat[i][j];
        printf("\nTHE SUM OF %d ROW IS %d",i+1,sumrow);
    }
}
2. Program to read a string and print the first two characters of each word in the string.

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

void main( )
{
    char s[100];
    int i,l;
    clrscr( );
    printf("Enter a string");
    gets(s);
    l=strlen(s);
    for(i=0;i<l;i++)
```
if(s[i]!=’ ‘ && s[i]=’ ‘)
{
    printf("%c %c",s[i],s[i+1])
    i=i+2;
    while(s[i]!=’ ‘)
    i++;
}

getch();

3. Program to print current system date.
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <dos.h>
int main()
{
    struct date d;
    getdate(&d);
    printf("Current system date is %d/%d/%d",d.da_day,d.da_mon,d.da_year);
    getch();
    return 0;
}.

4. Program to calculate Standard Deviation.
#include <stdio.h>
#include <math.h>
float standard_deviation(float data[], int n);

int main()
{
  int n, i;
  float data[100];
  printf("Enter number of datas (should be less than 100): ");
  scanf("%d", &n);
  printf("Enter elements: ");
  for(i=0; i<n; ++i)
    scanf("%f", &data[i]);
  printf("\n");
  printf("Standard Deviation = %.2f", standard_deviation(data, n));
  return 0;
}

float standard_deviation(float data[], int n)
{
  float mean=0.0, sum_deviation=0.0;
  int i;
  for(i=0; i<n; ++i)
  {
    mean+=data[i];
  }
  mean=mean/n;
  for(i=0; i<n; ++i)
    sum_deviation+=(data[i]-mean)*(data[i]-mean);
  return sqrt(sum_deviation/n);
}
5. Program to calculate the Power of a Number using Recursion.

```c
#include <stdio.h>

int power(int n1, int n2);

int main()
{
    int base, exp;
    printf("Enter base number: ");
    scanf("%d", &base);
    printf("Enter power number(positive integer): ");
    scanf("%d", &exp);
    printf("%d^%d = %d", base, exp, power(base, exp));
    return 0;
}

int power(int base, int exp)
{
    if (exp!=1)
        return (base * power(base, exp-1));
    return base;
}
```

6. Program to find the ASCII value of a Character.

```c
#include <stdio.h>

int main()
{
    char c;
    printf("Enter a character: ");
    scanf("%c", &c); /* Takes a character from user */
    printf("ASCII value of %c = %d", c, c);
}
7. Program to find biggest of four no by using ternary numbers.

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

void main( )
{
    int a,b,c,d,big;
    clrscr( );
    printf(“enter value a”);
    scanf(“%d”,&a);
    printf(“enter the value of b”);
    scanf(“%d”,&b);
    printf(“enter the value of c”);
    scanf(“%d”,&c);
    printf(“enter the value of d”);
    scanf(“%d”,&d);
    big=(a>b)?(a>c)?(a>d)?a:d:(c>d)?c:d:(b>c)?(b>d)?b:d:(c>d)?c:d;
    printf(“Biggest of the given 4 numbers is %d”,big);
    getch();
}
```

8. Matrix Multiplication.

```c
#include <stdio.h>

int main()
{
    int m, n, p, q, c, d, k, sum = 0;
```
int first[10][10], second[10][10], multiply[10][10];

printf("Enter the number of rows and columns of first matrix:\n");
scanf("%d%d", &m, &n);

/*//Entering elements of first matrix
printf("Enter the elements of first matrix\n");
for ( c = 0 ; c < m ; c++ )
    for ( d = 0 ; d < n ; d++ )
        scanf("%d", &first[c][d]);*/

printf("Enter the number of rows and columns of second matrix:\n");
scanf("%d%d", &p, &q);

//Checking if Matrix Multiplication is possible
if ( n != p )
{
    printf("Matrices with entered orders can't be multiplied with each other.\n");
    printf("The column of first matrix should be equal to row of second.\n");
}
else
{
    //Entering elements of first matrix
    printf("Enter the elements of first matrix:\n");
    for ( c = 0 ; c < m ; c++ )
        for ( d = 0 ; d < n ; d++ )
            scanf("%d", &first[c][d]);

    //Entering elements of second matrix
    printf("Enter the elements of second matrix:\n");
    for ( c = 0 ; c < p ; c++ )
        for ( d = 0 ; d < q ; d++ )
            scanf("%d", &second[c][d]);

    //Carrying out matrix multiplication operation
    for ( c = 0 ; c < m ; c++ )
    {
        for ( d = 0 ; d < q ; d++ )
        {
            for ( k = 0 ; k < p ; k++ )
            {
                sum = sum + first[c][k]*second[k][d];
            }
            multiply[c][d] = sum;
            sum = 0;
        }
    }
}
C Language Questions and Answers Viva - Voce

1. What is C language?

C is a programming language developed at AT&T's Bell Laboratories of USA in 1972. The C programming language is a standardized programming language developed in the early 1970s by Ken Thompson and Dennis Ritchie for use on the UNIX operating system. It has
since spread to many other operating systems, and is one of the most widely used programming languages.

2. What are the types of constants in C?
In C language two types of constants are available:
- Primary constants
- Secondary constants

3. What are the types of C instructions?
There are basically three types of instructions in C:
- Type Declaration Instruction
- Arithmetic Instruction
- Control Instruction

4. What is a pointer?
Pointers are variables which stores the address of another variable. That variable may be a scalar (including another pointer), or an aggregate (array or structure). The pointed-to object may be part of a larger object, such as a field of a structure or an element in an array.

5. What is an array?
Array is a variable that hold multiple elements which has the same data type.

6. Differentiate between arrays and pointers?
Pointers are used to manipulate data using the address. Pointers use * operator to access the data pointed to by them.

Array is a collection of similar data type. Array use subscripted variables to access and manipulate data. Array variables can be equivalently written using pointer expression.

7. What is “this” pointer?
The “this” pointer is a pointer accessible only within the member functions of a class, struct, or union type. It points to the object for which the member function is called. Static member functions do not have a “this” pointer.
8. What are the uses of a pointer?

Pointer is used in the following cases
- It is used to access array elements.
- It is used for dynamic memory allocation.
- It is used in Call by reference.
- It is used in data structures like trees, graph, linked list etc.

9. What is the purpose of main() function?

The function main() invokes other functions within it. It is the first function to be called when the program starts execution.
- It is the starting function.
- It returns an int value to the environment that called the program.
- Recursive call is allowed for main( ) also.
- It is a user-defined function.

10. What are the different storage classes in C?

There are four types of storage classes in C language.
- Automatic
- Extern
- Register
- Static

11. What is a structure?

Structure constitutes a super data type which represents several different data types in a single unit. A structure can be initialized if it is static or global.
12. Define Constructors?
A constructor is a member function with the same name as its class. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the values of data members of the class.

13. Define destructors?
A destructor is called for a class object when that object passes out of scope or is explicitly deleted. A destructor as the name implies is used to destroy the objects that have been created by constructors. Like a constructor, the destructor is a member function whose name is the same as the class name but is preceded by a tilde.

14. What is the use of default constructor?
A constructor that accepts no parameters is called the default constructor. If no user-defined constructor exists for a class A and one is needed, the compiler implicitly declares a default parameterless constructor A::A(). This constructor is an inline public member of its class. The compiler will implicitly define A::A() when the compiler uses this constructor to create an object of type A. The constructor will have no constructor initializer and a null body.

15. What is a macro?
Macros are the identifiers that represent statements or expressions.

16. What is the difference between #include<> and #include “ ”?
#include<> ----> specifically used for built in header files.
#include “ ” ----> Specifically used for used for user defined/created header file.

17. What are the advantages of the functions?
- It reduces the Complexity in a program by reducing the code.
- Functions are easily understanding and reliability and execution is faster.
- It also reduces the Time to run a program. In other way, it’s directly proportional to Complexity.
- It’s easy to find-out the errors due to the blocks made as function definition outside the main function.
18. How do declare an array?

We can declare an array by specify its data type, name and the number of elements the array holds between square brackets immediately following the array name.

syntax :

data_type array_name[size];

19. What are the differences between structures and union?

A structure variable contains each of the named members, and its size is large enough to hold all the members. Structure elements are of same size.

A Union contains one of the named members at a given time and is large enough to hold the largest member. Union element can be of different sizes.

20. What is the difference between an Array and a List?

The main difference between an array and a list is how they internally store the data whereas Array is collection of homogeneous elements. List is collection of heterogeneous elements.

21. What is the difference between a string copy (strcpy) and a memory copy (memcpy)?

The strcpy() function is designed to work exclusively with strings. It copies each byte of the source string to the destination string and stops when the terminating null character () has been moved.

On the other hand, the memcpy() function is designed to work with any type of data. Because not all data ends with a null character, you must provide the memcpy() function with the number of bytes you want to copy from the source to the destination.

22. What is the difference between const char*p and char const* p?

const char*p - p is pointer to the constant character. i.e value in that address location is constant.
const char* const p – p is the constant pointer which points to the constant string, both value and address are constants.

23. What is the purpose of realloc()?  
Realloc(ptr,n) function uses two arguments.  
- The first argument ptr is a pointer to a block of memory for which the size is to be altered.  
- The second argument n specifies the new size. The size may be increased or decreased.

24. What is the use of typedef? 
The typedef help in easier modification when the programs are ported to another machine. A descriptive new name given to the existing data type may be easier to understand the code.

25. What are the differences between new and malloc?  
- New initializes the allocated memory by calling the constructor. Memory allocated with new should be released with delete.  
- Malloc allocates uninitialized memory.  
- The allocated memory has to be released with free. New automatically calls the constructor while malloc(dosen’t)

26. What is the difference between strdup and strcpy?  
Both copy a string. strcpy wants a buffer to copy the string. strdup allocates a buffer using malloc(). Unlike strcpy(), strdup() is not specified by ANSI.

27. What is this pointer?  
It is a pointer that points to the current object. This can be used to access the members of the current object with the help of the arrow operator.

28. What is recursion?  
A recursion function is one which calls itself either directly or indirectly it must halt at a
29. What are the characteristics of arrays in C?

- An array holds elements that have the same data type.
- Array elements are stored in subsequent memory locations.
- Two-dimensional array elements are stored row by row in subsequent memory locations.

30. Differentiate between for loop and a while loop. What are their uses?

- For executing a set of statements fixed number of times, we use a for loop.
- When the number of iterations to be performed is not known in advance, we use a while loop.

31. What is the difference between printf(...) and sprintf(…)?

- printf(…): standard output statement
- sprintf(…): formatted output statement

32. What is an explicit constructor?

A conversion constructor declared with the explicit keyword. The compiler does not use an explicit constructor to implement an implied conversion of types. Explicit constructors are simply constructors that cannot take part in an implicit conversion.

33. What is a copy constructor?

Copy constructor is a constructor function with the same name as the class and used to make a deep copy of objects.

34. What is the difference between malloc() and calloc?

- malloc() is used for memory allocation and initializes garbage values. malloc() for allocating the single block of memory.
- calloc() is used for memory allocation and initializes the values of the allocated memory with 0s.
35. What is null pointer?
NULL pointer is a pointer which is pointing to nothing.
Examples:
int *ptr=(char *)0;
float *ptr=(float *)0;

36. What is dynamic array?
The dynamic array is an array data structure which can be resized during runtime which means elements can be added and removed.

37. What are macros? What are its advantages and disadvantages?
Macros are abbreviations for lengthy and frequently used statements. When a macro is called the entire code is substituted by a single line though the macro definition is of several lines.
The advantage of macro is that it reduces the time taken for control transfer as in case of function. The disadvantage of it is here the entire code is substituted so the program becomes lengthy if a macro is called several times.

38. What are register variables? What are the advantages of using register variables?
If a variable is declared with a register storage class, it is known as register variable. The register variable is stored in the CPU register instead of main memory. Frequently used variables are declared as register variable as it’s access time is faster.

39. What is storage class? What are the different storage classes in C?
Storage class is an attribute that changes the behavior of a variable. It controls the lifetime, scope and linkage. The storage classes in C are auto, register, and extern, static, typedef.
40. What the advantages of using Unions?

When the C compiler is allocating memory for unions it will always reserve enough room for the largest member.

41. In C, why is the void pointer useful? When would you use it?

The void pointer is useful because it is a generic pointer that any pointer can be cast into and back again without loss of information.

42. What is the difference between the functions memmove() and memcpy()?

The arguments of memmove() can overlap in memory. The arguments of memcpy() cannot.

43. Can a Structure contain a Pointer to itself?

Yes such structures are called self-referential structures.

44. What is dynamic memory allocation?

A dynamic memory allocation uses functions such as malloc() or calloc() to get memory dynamically. If these functions are used to get memory dynamically and the values returned by these function are assigned to pointer variables, such a way of allocating memory at run time is known as dynamic memory allocation.

45. What is pointer to a pointer?

If a pointer variable points another pointer value. Such a situation is known as a pointer to a pointer.

Example:

```c
int *p1, **p2,
v=10;
P1=&v;
p2=&p1;
```

Here p2 is a pointer to a pointer.

46. What is a function?

A large program is subdivided into a number of smaller programs or subprograms. Each subprogram specifies one or more actions to be performed for the larger program. Such
subprograms are called functions.

47. What is an argument?
An argument is an entity used to pass data from the calling to a called function.

48. What is the difference between syntax vs logical error?
Syntax Error
- These involve validation of syntax of language.
- Compiler prints diagnostic message.
Logical Error
- Logical error are caused by an incorrect algorithm or by a statement mistyped in such a way that it doesn’t violate syntax of language.
- Difficult to find.

49. Explain enumerated types.
- Enumerated types allow the programmers to use more meaningful words as values to a variable.
- Each item in the enumerated type variable is actually associated with a numeric code.

50. Differentiate between the expression “++a” and “a++”?
- With ++a, the increment happens first on variable a, and the resulting value is used. This is called as prefix increment.
- With a++, the current value of the variable will be used in an operation. This is called as postfix increment.