

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY COIMBATORE-641008

COIMBATORE-641008.

**(An Autonomous Institution)**

**DEPARTMENT OF SCIENCE AND HUMANITIES**

**20CS202- DATA STRUCTURES LABORATORY**

**ACADEMIC YEAR 2021-2022**

**Practical Record**

**Name : RASIKA B**

**Reg.No : 727721EUIT126**

**Branch : B.TECH -INFORMATION TECHNOLOGY**

**Class/Section : IT-C**

**Year/ Sem : I/II**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE-641 008.

**DEPARTMENT OF SCIENCE AND HUMANITIES**

**20CS202- DATA STRUCTURES LABORATORY**

**PRACTICAL RECORD**

**Name : RASIKA B Reg.no:727721EUIT126**

**Class : IT-C Branch:INFORMATION TECHNOLOFY**

**BONAFIDE CERTIFICATE**

Certified bonafide record of work done by Mr. /Ms **RASIKA B**

Reg.No **727721EUIT126** during the academic year 2021-2022(EVEN SEM)

##### Mr. S. PALANI, Dr.V. RAGAVI

##### AP/S&H HOD, S&H

##### 

Submitted for End Semester practical Examination, held on……………………….

EXTERNAL EXAMINER INTERNAL EXAMINER

| **EX.NO** | **DATE** | **NAME OF THE PROGRAM** | **SIGNATURE** |
| --- | --- | --- | --- |
| 1.a | 25/04/2022 | Finding the type of triangle- Equilateral/Scalene/Isosceles |  |
| 1.b | 25/04/2022 | Get marks as input and print the grade |  |
| 1.c | 25/04/2022 | Printing the series 1,4,9,16,........ |  |
| 1.d | 25/04/2022 | Printing the factors of a number |  |
| 2.a | 30/04/2022 | Factorial of the number using recursion |  |
| 2.b | 30/04/2022 | Swapping the numbers using function |  |
| 2.c | 3/04/2022 | Arithmetic operations using functions |  |
| 3.a | 07/05/2022 | Delete an element at a particular position in the array |  |
| 3.b | 07/05/2022 | Palindrome or not without using string library functions. |  |
| 3.c | 07/05/2022 | String Manipulation functions. |  |
| 3.d | 07/05/2022 | Matrix Addition, Subtraction using Array. |  |
| 4.a | 13/05/2022 | Employee information system using nested structure |  |
| 4.b | 13/05/2022 | Modulus of two numbers using pointers |  |
| 4.c | 13/05/2022 | Student Information System using Union data type |  |
| 4.d | 13/05/2022 | Swapping of numbers using function with call by reference |  |
| 5.a | 19/05/2022 | Reading from and Writing into a file |  |
| 5.b | 19/05/2022 | Print the number of lines in the text file |  |
| 6.a | 25/05/2022 | Implementation of Singly, Doubly and circular Linked List |  |
| 6.b | 25/05/2022 | Implementation of Polynomial manipulation |  |
| 7 | 31/05/2022 | Implementation of stack using array |  |
| 8 | 06/06/2022 | Implementation of stack using Linked List |  |
| 9 | 17/06/2022 | Conversion of an infix expression to postfix expression using Stack |  |
| 10 | 23/06/2022 | Implementation of queue using Arrays |  |
| 11 | 23/06/2022 | Implementation of queue using Linked List |  |
| 12 | 29/06/2022 | Implementation of Priority Queue |  |
| 13 | 29/06/2022 | Implementation of queue applications |  |
| 14 | 05/07/2022 | Implementation of Binary Search Tree |  |
| 15 | 11/07/2022 | Implementation of hashing techniques |  |

|  |  |
| --- | --- |
| **EXPT NO:1(a)**  **DATE:25/4/22** | **FIND WHETHER THE TRIANGLE IS SCALENE OR EQUILATERAL OR ISOSCELES** |

**AIM:**

To write a program whether a triangle is equilateral/scalene/isosceles.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variables s1,s2 and s3
4. Read the inputs s1,s2 and s3
5. Display the inputs s1,s2,s3
6. Check IF( s1==s2 && s2==s3 && s3==s1),if the condition is true
7. Display ”Equilateral triangle”
8. Check ELSE IF (s1==s2 || s1==s3 || s2==s3),if the condition is true
9. Display” Isosceles triangle”
10. Otherwise ELSE
11. Display “Scalene triangle”
12. Stop

**SOURCE CODE:**

#include<stdio.h>

int main()

{

int s1,s2,s3;

scanf("%d",&s1);

scanf("%d",&s2);

scanf("%d",&s3);

printf("Side 1:%d\n",s1);

printf("Side 2:%d\n",s2);

printf("Side 3:%d\n",s3);

if(s1==s2 && s1==s3 && s2==s3)

{

printf("Equilateral triangle");

}

else if(s1==s2 || s1==s3 || s2==s3)

{

printf("Isosceles triangle");

}

else

{

printf("Scalene triangle");

}

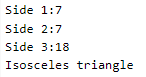
return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to determine the type of triangle is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:1(b)**  **DATE:25/4/22** | **GET MARKS AS INPUT AND PRINT THE GRADE** |

**AIM:**

To create a C program to get a marks as input and print the grade.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variable mark
4. Read mark
5. Display mark
6. Check IF(mark>=91 && mark<=100),if the condition is true
7. Display”Grade A”
8. Check ELSE IF(mark>=71 && mark<=90),if the condition is true
9. Display”Grade B”
10. Check ELSE IF(mark>=51 && mark<=70),if the condition is true
11. Display”Grade C”
12. Check ELSE IF(mark>=31 && mark<=50),if the condition is true
13. Display”Grade D”
14. Check ELSE IF(mark<31),if the condition is true
15. Display”Fail”
16. Stop

**SOURCE CODE:**

#include<stdio.h>

int main()

{

int mark;

scanf("%d",&mark);

printf("Mark: %d\n",mark);

if(mark>=91 && mark<=100)

{

printf("Grade A");

}

else if(mark>=71 && mark<=90)

{

printf("Grade B");

}

else if(mark>=51 && mark<=70)

{

printf("Grade C");

}

else if(mark>=31 && mark<=50)

{

printf("Grade D");

}

else if(mark<31)

{

printf("Fail");

}

return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to determine the grade is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:1(c)**  **DATE:25/4/22** | **PRINTING THE SERIES 1,4,9,16,........** |

**AIM:**

To create a program to print the series number.

**ALGORITHM:**

1. Start

2. Include all the header files which are used in the program

3. Take input number from the user, and store it in variable num.

4. Then use while statement (i<=num ) then print the i is a power of num.

5.Stop

**SOURCE CODE:**

#include <stdio.h>

int main()

{

int num,i=1;

printf("Enter the number:");

scanf("%d",&num);

printf("%d",num);

while (i<=num)

{

printf("%d ",i\*i);

i++;

}

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT**:



**RESULT:**

Thus the program to print the series is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:1(d)**  **DATE:25/4/22** | **PRINTING THE FACTORS OF A NUMBER** |

**AIM:**

To create a program to print the factors of a number**.**

**ALGORITHM:**

1. Start

2. Include all the header files which are used in the program

3. Take input number from the user, and store it in variable num.

4.Start loop from 1 to num by incrementing 1 in each iteration.

5.something like this (i=1;i<=num;i++)

6. Then use if statement inside for loop to print factors of a number if (num%i==0)

then i is a factor of num.

7.Stop

**SOURCE CODE:**

#include <stdio.h>

int main()

{

int num, i;

printf("Enter a positive number: ");

scanf("%d", &num);

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

printf("Factors of %d are:\n", num);

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

{

if (num%i==0)

{

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

}

}

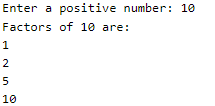
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program to print the factors is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:2(a)**  **DATE:30/4/22** | **FACTORIAL OF THE NUMBER USING RECURSION** |

**AIM:**

To create a program to find factorial of a number using recursion.

**ALGORITHM:**

1.Start

2.Read the variables a, fact

3.Declare the function, int factorial (a)

4.Call the function, factorial(a)

5.If a= =1 or a= =0, return 1

6.Else fact =a\*factorial(a-1)

7.Display fact

8.Stop

**SOURCE CODE:**

#include<stdio.h>

int factorial(int);

int main ()

{

int a, fact;

printf("\nEnter a number: ");

scanf("%d",&a);

printf("%d",a);

fact =factorial(a);

printf("\nFactorial of %d is: %d",a, fact);

return 0;

}

int factorial(int a)

{

if(a==0||a==1)

{

return 1;

}

else

{

return(a\*factorial(a-1));

}

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program to print the factorial using recursion is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:2(b)**  **DATE:30/4/22** | **SWAPPING THE NUMBERS USING FUNCTION** |

**AIM:**

To create a program to swap numbers using function.

**ALGORITHM:**

1.Start

2.Read the variables a,b,temp

3.Declare the function void swap(int x,int y)

4.Call the function(swap a,b)

5.Assign temp=x

6.Assign x=y

7.Assign y=temp

8.Print the numbers after swapping ( a and b)

9.Stop

**SOURCE CODE:**

#include<stdio.h>

void swap(int, int);

int main()

{

int a, b;

scanf("%d", &a);

scanf("%d", &b );

printf("\nBefore swapping: a = %d and b = %d\n", a, b);

swap(a, b);

return 0;

}

void swap(int x, int y)

{

int temp;

temp = x;

x = y;

y = temp;

printf("After swapping: a = %d and b = %d", x, y);

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program to swap the numbers using function is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:2(c)**  **DATE:30/4/22** | **ARITHMETIC OPERATIONS USING FUNCTIONS** |

**AIM:**

To create a C program to calculate arithmetic operations using functions.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variables a and b
4. Read the inputs a and b
5. Display the inputs a and b
6. Call the add function
7. Declare the variables res
8. res=a+b
9. Display res
10. Call the sub function
11. res=a-b
12. Display res
13. Call the multi function
14. res=a\*b
15. Display res
16. Call the division function
17. res=a/b
18. Display res
19. Call the mod function
20. res=a%b
21. Display res

22.Stop

**SOURCE CODE:**

#include<stdio.h>

int add(int a,int b)

{

int res=a+b;

printf("Addition:%d\n",res);

}

int sub(int a,int b)

{

int res=a-b;

printf("Subtraction:%d\n",res);

}

int multi(int a,int b)

{

int res=a\*b;

printf("Multiplication:%d\n",res);

}

float division(float a,float b)

{

float res=a/b;

printf("Division:%f\n",res);

}

int mod(int a,int b)

{

int res=a%b;

printf("Modulus:%d\n",res);

}

int main()

{

int a,b;

scanf("%d",&a);

scanf("%d",&b);

printf("Num 1:%d\n",a);

printf("Num 2:%d\n",b);

add(a,b);

sub(a,b);

multi(a,b);

division(a,b);

mod(a,b);

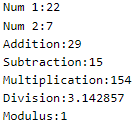
return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to perform arithmetic operations is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:3(a)**  **DATE:7/5/22** | **DELETE AN ELEMENT AT A PARTICULAR POSITION IN THE ARRAY** |

**AIM:**

To create a C program to delete an element at a particular position in the array.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Declare the variable n
4. Read the input n
5. Read the array elements of arrays arr[n]
6. Display the array elements of arrays arr[n]
7. Declare the variables del
8. Read the input del
9. Initialize the variable k=0,count=0
10. Initialize the variable i=0
11. Initialize temp[n]
12. Check FOR condition, if the condition is true iterate the loop for n times
13. Check IF Condition,if(arr[i]!=del)
14. temp[k]=arr[i]
15. Increment the k value
16. Otherwise ELSE condition
17. Increment the count value
18. Finally check IF condition,if(count!=0)
19. Check FOR condition,if the condition is true iterate the loop for k times
20. Display temp[i]
21. Otherwise ELSE condition
22. Display "Element dosen't found in the array"
23. Stop

**SOURCE CODE:**

#include<stdio.h>

int main()

{

int n,i,del,k=0,count=0;

scanf("%d",&n);

int arr[n],temp[n];

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

{

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

}

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

{

printf("%d ",arr[i]);

}

scanf("%d",&del);

printf("\nDelete element:%d",del);

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

{

if(arr[i]!=del)

{

temp[k]=arr[i];

k++;

}

else

{

count+=1;

}

}

if(count!=0)

{

printf("\nArray after deletion:\n");

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

{

printf("%d ",temp[i]);

}

}

else

{

printf("Element dosen't found in the array");

}

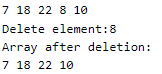
return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to delete a particular element in an array is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:3(b)**  **DATE:7/5/22** | **PALINDROME OR NOT WITHOUT USING STRING LIBRARY FUNCTIONS.** |

**AIM:**

To write a program to check the given string is a palindrome or not without using string library functions.

**ALGORITHM:**

1. Start
2. Include all the head files
3. Declare two character array a[50],str[50]
4. Declare three integer i, k=0, c
5. Read a
6. Initialize a for loop with int i=0 , condition a[i]!=’\0’ with increment i++
7. Initialize a for loop with int j=i-1 , condition j>=0 with increment j++
8. Iterate ‘a’ and store it in ‘str’
9. Initialize a for loop with int i=0 , condition a[i]!=0 with increment i++
10. Iterating to check whether the each character in ‘a’,’str’ are same
11. If two are same upgrade the c=1 else c=0
12. If c==1 print palindrome
13. Else if c==0 print not a palindrome
14. Stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

int main()

{

char a[50],str[50];

int i,k=0,c;

scanf("%s",a);

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

for(int j=i-1;j>=0;j--)

{

str[k]=a[j];

k++;

}

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

{

if(a[i]==str[i])

{

c=1;

}

else

{

c=0;

break;

}

}

if(c==1)

{

printf("Palindrome");

}

else

{

printf("Not a Palindrome");

}

return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program find whether the given string is a palindrome or not without using string library functions is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:3(c)**  **DATE:7/5/22** | **STRING MANIPULATION FUNCTIONS.** |

**AIM:**

To create a program using strcat(),strcmp(),strcpy(),strlen(),strrev() using two strings input.

**ALGORITHM:**

1. Start
2. Include all the head files
3. Include string.h header file to perform string operations
4. Declare two char array a[50],b[50],s,c[50]
5. Read a and b
6. Initialize l which is length of a and l1which is the length of b using strlen() function
7. Print the copy of a which is stored in c using strcopy()
8. Print the result of the concatenation using strcat()
9. Print l and l1
10. Compare a and b using strcmp()
11. If s==0 print strings are same
12. Else strings are not same
13. Stop

**SOURCE CODE:**

#include<stdio.h>

#include<string.h>

int main()

{

char a[50],b[50],s,c[50];

scanf("%s",a);

scanf("%s",b);

int l,l1;

l=strlen(a);

l1=strlen(b);

printf("%s\n",strcpy(c,a));

printf("%s\n",strcat(a,b));

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

printf("%d\n",l1);

s=strcmp(a,b);

if(s==0)

{

printf("Strings are same\n");

}

else

{

printf("Strings are not same");

}

return 0;

}

#include<stdio.h>

#include<string.h>

int main()

{

Char str[50]=”WELCOME”;

printf(“The given string is %s\n”,str);

printf(“Reversing string is %s\n”,strrev(str));

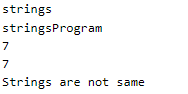
Return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**



**RESULT:**

Thus the program using strcat(),strcmp(),strcpy(),strlen(),strrev() is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:3(d)**  **DATE:7/5/22** | **ARITHMETIC OPERATIONS USING FUNCTIONS** |

**AIM:**

To write a program for matrix addition, subtraction using array.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Read n
4. Read the array elements of arrays a[n], b[n]
5. Declare the variables sum, diff
6. Initialize the variable i=0
7. Check FOR condition, if the condition is true iterate the loop for n times
8. sum=a[i] + b[i]
9. Display sum
10. Initialize the variable i=0
11. Check FOR condition, if the condition is true iterate the loop for n times
12. diff=a[i]-b[i]
13. Display diff
14. Stop

**SOURCE CODE:**

#include<stdio.h>

int main(){

int n;

scanf("%d",&n);

int a[n],b[n];

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

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

}

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

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

}

int sum;

int diff;

printf("Addition:");

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

sum=a[i]+b[i];

printf("%d ",sum);

}

printf("\n");

printf("Subtraction:");

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

diff=a[i]-b[i];

printf("%d ",diff);

}

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program for matrix addition, subtraction using array is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:4(a)**  **DATE:13/5/22** | **EMPLOYEE INFORMATION SYSTEM USING NESTED STRUCTURE** |

**AIM:**

To create a C program for employee information system using nested structure.

**ALGORITHM:**

1. Start
2. Create a structure employee with the members name, id, sal and the nested structure dob which contains the members date, mon, yr.
3. Create the structure variables d for the structure dob and the variable e for the employee
4. Read the members of the structure
5. Display the members of the structure
6. Stop

**SOURCE CODE:**

#include<stdio.h>

struct employee{

char name[50];

int id;

int sal;

struct dob{

int date;

char mon[20];

int yr;

}d;

}e;

int main(){

scanf("%s",e.name);

scanf("%d",&e.id);

scanf("%d",&e.sal);

scanf("%d",&e.d.date);

scanf("%s",e.d.mon);

scanf("%d",&e.d.yr);

printf("Name:%s\n",e.name);

printf("ID:%d\n",e.id);

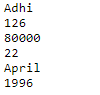
printf("Salary:%d\n",e.sal);

printf("DOB:%d-%s-%d",e.d.date,e.d.mon,e.d.yr);

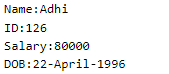
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program for employee information system using nested structure is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:4(b)**  **DATE:13/5/22** | **MODULUS OF TWO NUMBERS USING POINTERS** |

**AIM:**

To write a program to find the modulus of two numbers using pointers.

**ALGORITHM:**

1. Start
2. Include all the header files which are used in the program
3. Take the quotient and reminder from the user as a and b
4. Calulate modulus of a and b and store it in the variable m
5. Store the address of m in mod
6. Display \*mod
7. Stop

**SOURCE CODE:**

#include <stdio.h>

int main(){

int a,b,m,\*mod;

scanf("%d",&a);

scanf("%d",&b);

m=a%b;

mod=&m;

printf("Modulus : %d",\*mod);

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program to find the modulus of two numbers using pointers is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:4(c)**  **DATE:13/5/22** | **STUDENT INFORMATION SYSTEM USING UNION DATA TYPE** |

**AIM:**

To write a program containing the union student with the given members and display

them.

**ALGORITHM:**

1. Start
2. Include all the header files which are used in the program
3. Declare the union student and include the members rn, name, mark, avg and grad
4. Collect the data such as roll number and name from the user and display it using union variable immediately
5. Collect the marks of five subject from the user and calculate the average and display it using the union variable
6. Then calculate the grade using average. If the average is greater than 70, grade is 1. Else if greater 49 and less than 71 grade is 2. Else grade is 3.
7. Display the grade using the union variable.
8. Stop

**SOURCE CODE:**

#include<stdio.h>

union student{

int rn;

char name[50];

int mark[5];

float avg;

int grad;

};

int main()

{

int total;

union student s;

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

printf("Roll.no: ");

printf("%d\n",s.rn);

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

printf("Name: ");

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

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

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

total=total+s.mark[i];

}

s.avg=total/5;

printf("Average: ");

printf("%f\n",s.avg);

printf("Grade: ");

if(s.avg>70){

s.grad=1;

printf("%d",s.grad);

}

else if(s.avg>=50 && s.avg<=70){

s.grad=2;

printf("%d",s.grad);

}

else{

s.grad=3;

printf("%d",s.grad);

}

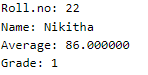
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program using union is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:4(d)**  **DATE:13/5/22** | **SWAPPING OF NUMBERS USING FUNCTION WITH CALL BY REFERENCE** |

**AIM:**

To write a program to swap two numbers.

**ALGORITHM:**

  1. Start.

2. Get a and b.

3. Call the function swap(a,b).

4. Start a function.

5. Assign temp=a.

6. Assign a=b.

7. Assign b=temp.

8. Print a and b.

9. Stop.

**SOURCE CODE:**

#include <stdio.h>

void swap(int \*a,int \*b);

int main()

{

int a,b;

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

swap(&a,&b);

printf("Inside main:%d %d",a,b);

return 0;

}

void swap(int \*a,int \*b)

{

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

printf("Inside swap:%d %d\n",\*a,\*b);

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to swap the number using function with call by reference is executed successfully.

|  |  |
| --- | --- |
| **EX.NO:5(a)**  **DATE:19/5/22** | **READING FROM AND WRITING INTO A FILE** |

**AIM:**

To create a program for creating a file "ITB.txt".Open the file and read the content and print it.

**ALGORITHM:**

Creating a file:

1. Start
2. Declare variables and the file pointer
3. Enter name of file
4. Open file in write mode
5. Print file is not created if no file is created else print file created successfully
6. End

For Reading a Text File

1. Start
2. Declare variables and the file pointer
3. Open the file
4. If the file is not opened print error massage
5. Print the content of the text file using the loop
6. Close the file
7. Stop

**SOURCE CODE:**

#include<stdio.h>

#include <stdlib.h>

int main()

{

FILE \*fp;

char fName[200];

scanf("%s",&fName);

printf("Enter file name to create :%s\n",fName);

fp=fopen(fName,"w");

if(fp==NULL)

{

printf("File is not created!!!");

exit(0);

}

printf("File created successfully.");

return 0;

}

#include <stdio.h>

#include <stdlib.h>

int main()

{

FILE \*fptr;

char filename[100], c;

scanf("%s", &filename);

printf("Enter the filename to open:%s\n ",filename);

fptr = fopen(filename, "r");

if (fptr == NULL)

{

printf("Cannot open file \n");

exit(0);

}

else

c = fgetc(fptr);

while (c != EOF)

{

printf ("%c", c);

c = fgetc(fptr);

}

fclose(fptr);

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

****

**RESULT:**

Thus the above program for creating a file “ITB.txt” and read the content is executed successfully.

|  |  |
| --- | --- |
| **EX.NO:5(b)**  **DATE:19/5/22** | **PRINT THE NUMBER OF LINES IN THE TEXT FILE** |

**AIM:**

To create a program to create a file “ITC.txt” and find the number of lines in the text file.

**ALGORITHM:**

Creating a file:

1. Start
2. Declare variables and the file pointer
3. Enter name of file
4. Open file in write mode
5. Print file is not created if no file is created else print file created successfully
6. End

Number of lines:

1. Start
2. Read file name
3. Open file in read-mode using a File pointer 'fp'.
4. Characters from the file are read into a char variable 'sample\_chr' with the help of getc
5. function.
6. If a new line character(' ') is encountered, the integer variable 'no\_lines' is
7. incremented.
8. If the character read into 'sample\_char' is not a new line character, next character is
9. read from the file.
10. Continues until the last character of the file(EOF) is encountered.
11. The file pointer is then closed
12. The total number of lines is shown as output.
13. Stop

**SOURCE CODE:**

#include<stdio.h>

#include <stdlib.h>

int main()

{

FILE \*fp;

char fName[200];

scanf("%s",&fName);

printf("Enter file name to create :%s\n",fName);

fp=fopen(fName,"w");

if(fp==NULL)

{

printf("File is not created!!!");

exit(0);

}

printf("File created successfully.");

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

#include <stdio.h>

#define MAX\_FILE\_NAME 100

int main()

{

FILE \*fp;

int count = 0;

char filename[MAX\_FILE\_NAME];

char c;

scanf("%s", &filename);

printf("Enter file name:%s\n ",filename);

fp = fopen(filename, "r");

if (fp == NULL)

{

printf("Could not open file %s", filename);

return 0;

}

for (c = getc(fp); c != EOF; c = getc(fp))

if (c == '\n')

count = count + 1;

fclose(fp);

printf("The file %s has %d lines\n ", filename, count);

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the above program to create a file “ITC.txt” and find the number of lines in the text file is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:6(a)**  **DATE:25/5/22** | **IMPLEMENTATION OF SINGLY, DOUBLY AND CIRCULAR LINKED LIST** |

**AIM:**

To write a program to implementation of Singly, Doubly and circular Linked List.

**ALGORITHM:**

i)Singly linked list:

1. Start
2. Include all the required header files
3. Define a structure, define an integer and a structure pointer to store the next node address inside the structure.
4. Define a function insertend having parameters as head address and a value to insert at the end of the linked list
5. Create a newnode pointer to structure node
6. Assign the value from use as its value and define its address as null
7. If head is null initially change head as newnode
8. Else by using while find the last node and change its address to newnode
9. Define a function insertfront having same parametes as insertend to insert an element before the linked list
10. If head is null initially change head as newnode
11. Else change the newnode’s address as head and newnode as head
12. Define a function display to print the elements in the linked list,head as its parameter
13. Using while loop print the values in each node
14. In the main function define head as a null initially which is a pointer to the structure
15. By calling the above functions create a linked list and display it
16. Stop

ii)Doubly linked list:

1. Start
2. Include all the required header files
3. Define a structure, define an integer and a two structure pointer to store the next node address and previous node address inside the structure.
4. Define a function insertend having parameters as head address and a value to insert at the end of the linked list
5. Create a newnode pointer to structure node
6. Assign the value from user as its value and define its both address as null
7. If head is null initially change head as newnode
8. Else by using while find the last node and change its next address to newnode and newnode’s previous address as lastnode address
9. Define a function insertfront having same parametes as insertend to insert an element before the linked list
10. If head is null initially change head as newnode
11. Else change the newnode’s next address as head and newnode as head
12. Define a function display to print the elements in the linked list,head as its parameter
13. Using while loop print the values in each node
14. In the main function define head as a null initially which is a pointer to the structure
15. By calling the above functions create a linked list and display it
16. Stop

iii)Circular linked list:

1. Start
2. Include all the required header files
3. Define a structure, define an integer and a structure pointer to store the next node address inside the structure.
4. Define a function insertend having parameters as last address and a value to insert at the end of the linked list
5. Create a newnode pointer to structure node
6. If last is null initially change head as newnode and its address also newnode
7. Else newnode’s address as last node’s address,last node address as newnode and chane last node also newnode
8. Define a function insertfront having parameters as last address and a value to insert at the end of the linked list
9. Create a newnode pointer to structure node
10. If last is null initially change head as newnode and its address also newnode
11. Else newnode’s address as last’s address and last node address as newnode
12. Define display function to print values by using dowhile loop
13. In the main function define head as a null initially which is a pointer to the structure
14. By calling the above functions create a linked list and display it
15. Stop

**SOURCE CODE:**

i)Singly linked list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int val;

struct node\* addr;

};

void insertend(struct node\*\* q,int x)

{

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->val=x;

newnode->addr=NULL;

if(\*q==NULL)

{

\*q=newnode;

}

else

{

struct node\* temp=\*q;

while(temp->addr!=NULL)

{

temp=temp->addr;

}

temp->addr=newnode;

}

}

void insertfront(struct node\*\*q,int x)

{

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->val=x;

newnode->addr=NULL;

if(\*q==NULL)

{

\*q=newnode;

}

else

{

newnode->addr=\*q;

\*q=newnode;

}

}

void display(struct node\* q)

{

while(q!=NULL)

{

printf("%d ",q->val);

q=q->addr;

}

}

int main()

{

struct node\* head=NULL;

insertfront(&head,10);

insertfront(&head,9);

insertfront(&head,8);

insertfront(&head,7);

insertend(&head,20);

insertend(&head,30);

insertend(&head,40);

display(head);

return 0;

}

ii)Doubly linked list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int val;

struct node\* nxt;

struct node\* pre;

};

void insertend(struct node\*\* q,int x)

{

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->val=x;

newnode->nxt=NULL;

newnode->pre=NULL;

if(\*q==NULL)

{

\*q=newnode;

}

else

{

struct node\* temp=\*q;

while(temp->nxt!=NULL)

{

temp=temp->nxt;

}

temp->nxt=newnode;

newnode->pre=temp;

}

}

void insertfront(struct node\*\*q,int x)

{

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->val=x;

newnode->nxt=NULL;

newnode->pre=NULL;

if(\*q==NULL)

{

\*q=newnode;

}

else

{

newnode->nxt=\*q;

(\*q)->pre=newnode;

\*q=newnode;

}

}

void display(struct node\* q)

{

while(q!=NULL)

{

printf("%d ",q->val);

q=q->nxt;

}

}

int main()

{

struct node\* head=NULL;

insertfront(&head,10);

insertfront(&head,9);

insertfront(&head,8);

insertfront(&head,7);

insertend(&head,20);

insertend(&head,30);

insertend(&head,40);

display(head);

return 0;

}

iii)Circular linked list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int val;

struct node\* addr;

};

void insertend(struct node\*\* q,int x)

{

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->val=x;

newnode->addr=NULL;

if(\*q==NULL)

{

\*q=newnode;

(\*q)->addr=\*q;

}

else

{

newnode->addr=(\*q)->addr;

(\*q)->addr=newnode;

\*q=newnode;

}

}

void insertfront(struct node\*\*q,int x)

{

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->val=x;

newnode->addr=NULL;

if(\*q==NULL)

{

\*q=newnode;

(\*q)->addr=\*q;

}

else{

newnode->addr=(\*q)->addr;

(\*q)->addr=newnode; }

}

void display(struct node\* q)

{

struct node\* x=q->addr;

do

{

printf("%d ",x->val);

x=x->addr;

}while(x!=q->addr);

}

int main()

{

struct node\* last=NULL;

insertfront(&last,10);

insertfront(&last,9);

insertfront(&last,8);

insertfront(&last,7);

insertend(&last,20);

insertend(&last,30);

insertend(&last,40);

display(last);

return 0;

}

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program using singly, doubly, circular linked list is verified successfully.

|  |  |
| --- | --- |
| **EXPT NO:6(b)**  **DATE:31/5/22** | **IMPLEMENTATION OF POLYNOMIAL MANIPULATION** |

**AIM:**

To create a C program to implement Polynomial manipulation

**ALGORITHM:**

1. Start
2. A structure array [], b[] and c[] is declared.
3. The number of coefficients of both polynomials are read to the variables n1 and n2 respectively.
4. Coefficients and exponents of both the polynomials are read one by one to the arrays a[] and b[] using a for loop.
5. Then the program uses while loop to find the sum of these polynomials and stores it in the array c[].
6. Then the result of the addition is displayed.
7. Stop

**SOURCE CODE:**

#include<stdio.h>

struct poly

{

int coeff;

int expo;

};

struct poly p1[10],p2[10],p3[10];

int readPoly(struct poly []);

int addPoly(struct poly [],struct poly [],int ,int ,struct poly []);

void displayPoly( struct poly [],int terms);

int main()

{

int t1,t2,t3;

t1=readPoly(p1);

printf(" \nFirst polynomial : ");

displayPoly(p1,t1);

printf("\n \n");

t2=readPoly(p2);

printf(" \nSecond polynomial : ");

displayPoly(p2,t2);

printf("\n \n");

t3=addPoly(p1,p2,t1,t2,p3);

printf(" \n\nResultant polynomial after addition : ");

displayPoly(p3,t3);

printf("\n");

return 0;

}

int readPoly(struct poly p[10])

{

int t1,i;

printf("\nEnter the total number of terms in the polynomial:");

scanf("%d",&t1);

printf("\nEnter the COEFFICIENT and EXPONENT in DESCENDING ORDER\n");

for(i=0;i<t1;i++){

printf("\nEnter the Coefficient(%d): ",i+1);

scanf("%d",&p[i].coeff);

printf("%d",p[i].coeff);

printf("\nEnter the exponent(%d): ",i+1);

scanf("%d",&p[i].expo);

printf("%d",p[i].expo);

}

return(t1);

}

int addPoly(struct poly p1[10],struct poly p2[10],int t1,int t2,struct poly p3[10])

{

int i,j,k; i=0; j=0; k=0;

while(i<t1 && j<t2){

if(p1[i].expo==p2[j].expo){

p3[k].coeff=p1[i].coeff + p2[j].coeff;

p3[k].expo=p1[i].expo;

i++;

j++;

k++;

}

else if(p1[i].expo>p2[j].expo){

p3[k].coeff=p1[i].coeff;

p3[k].expo=p1[i].expo;

i++;

k++;

}

else{

p3[k].coeff=p2[j].coeff;

p3[k].expo=p2[j].expo;

j++;

k++;

}

}

while(i<t1)

{

p3[k].coeff=p1[i].coeff;

p3[k].expo=p1[i].expo;

i++;

k++;

}

while(j<t2){

p3[k].coeff=p2[j].coeff;

p3[k].expo=p2[j].expo;

j++;

k++;

}

return(k);

}

void displayPoly(struct poly p[10],int term){

int k;

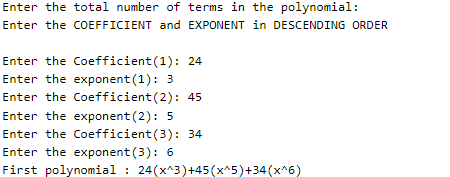
for(k=0;k<term-1;k++)

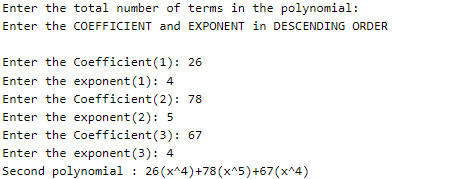
printf("%d(x^%d)+",p[k].coeff,p[k].expo);

printf("%d(x^%d)",p[term-1].coeff,p[term-1].expo);

}

**SAMPLE INPUT AND OUTPUT:**

****

****

****

**RESULT:**

The C program to implement Polynomial manipulation is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:7**  **DATE:31/5/22** | **IMPLEMENTATION OF STACK USING ARRAY** |

**AIM:**

To write a program to implement stack using array.

**ALGORITHM:**

1. Start
2. Include all the required header files
3. Before main function define an array of integers and top identifier with value -1 to define top element of the stack
4. Define a function push with insertion value as its parameter
5. If top is greater than or equals to size of array-1 then print stack is overflow
6. Else increase the top by one and assign the value to array[top]
7. Define a function pop to delete the top element of the stack
8. If top is less than or equals to -1 then print stack is underflow
9. Else print the popped element is array[top] and reduce top by one
10. Define a function display to print the stack elements
11. If top is greater or equals to zero by using for loop print the array[top] element and reduce top by one run the loop until top reaches zero
12. Else print the stack is empty
13. In main function by using the above mentioned functions push pop and display the stack elements
14. Stop

**SOURCE CODE:**

#include<stdio.h>

int arr[10];

int n=10;

int top=-1;

void push(int val)

{

if(top>=n-1)

{

printf("Stack is Overflow");

}

else

{

top++;

arr[top]=val;

}

}

void pop()

{

if(top<=-1)

{

printf("Stack is Underflow");

}

else

{

printf("The poped element is %d \n",arr[top]);

top--;

}

}

void display()

{

if(top>=0)

{

for(int i=top;i>=0;i--)

{

printf("%d ",arr[i]);

}

}

else

{

printf("Stack is empty");

}

}

int main()

{

push(1);

push(2);

push(3);

push(10);

display();

printf("\n");

pop();

display();

return 0;

}

**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to implement stack using array is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:8**  **DATE:6/6/22** | **IMPLEMENTATION OF STACK USING LINKED LIST** |

**AIM:**

To create a program to implement a stack using Linked List.

**ALGORITHM:**

1. start
2. Include all the **header files** which are used in the program
3. Declare all the **user defined functions**.
4. Create a node pointer head and make it as NULL
5. Read X
6. Call the insert function to insert the elements at front.
7. Create the node and insert node data and its address as NULL

8.Check if the stack is empty or not.

9.If it is e**mpty**, then set **newnode -> next** = **NULL**

10.If it is n**ot empty**, then set **newnode -> address** = head

**11.**Finally, set head = **newnode**.

12.To display the stack call the function display

13.Check if the stack is empty or not

14.If it is e**mpty**, then display s**tack is empty** and terminate the function.

15.If it is not empty, display head->address until head!=NULL

16.Stop

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\*addr;

};

void insert(struct node\*\*q,int val)

{

struct node\*nn=(struct node\*)malloc(sizeof(struct node));

nn->data=val;

nn->addr=NULL;

if(\*q==NULL)

{

\*q=nn;

}

else

{

nn->addr=(\*q);

(\*q)=nn;

}

}

void display(struct node\*q)

{

if(q==NULL)

{

printf("Stack is empty");

}

else

{

while(q!=NULL)

{

printf("%d\t",q->data);

//cout<<q->data<<" ";

q=q->addr;

}

}

}

int main()

{

struct node\*head=NULL;

int x;

while(1)

{

scanf("%d",&x);

if(x>0)

{

insert(&head,x);

}

else

{

break;

}

}

display(head);

}

**SAMPLE INPUT :**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the program to implement a stack using Linked List is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:9**  **DATE:17/6/22** | **CONVERSION OF AN INFIX EXPRESSION TO POSTFIX EXPRESSION USING STACK** |

**AIM:**

To create a program to convert an infix expression to postfix expression using stack implementation

**ALGORITHM:**

1.Start

2.Include all the required header files

3.Declare all user defined functions

4.Scan the infix expression from left to right

5.If the scanned operator is greater than the precedence order of the operator in the stack, or if the stack is empty or contains open ‘(‘ , then push it into the stack

6.Else

7.Pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator. Then push the scanned operator into the stack

8.If the scanned character is an ‘(‘ , then push it into the stack

9.If the scanned character is an ‘)’ , then pop the stack and print it until a ‘(‘ is encountered and eliminate both the paranthesis

10.Continue steps 3 to 8 until the infix expression is scanned

11.Print the output

12.Pop and output from the stack until it is not empty

13.Stop.

**SOURCE CODE:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

struct Stack

{

int top;

int capacity;

int\* array;

};

struct Stack\* createStack( int capacity )

{

struct Stack\* stack = (struct Stack\*)

malloc(sizeof(struct Stack));

stack->top = -1;

stack->capacity = capacity;

stack->array = (int\*) malloc(stack->capacity \*sizeof(int));

return stack;

}

int isEmpty(struct Stack\* stack)

{

return stack->top == -1 ;

}

char peek(struct Stack\* stack)

{

return stack->array[stack->top];

}

char pop(struct Stack\* stack)

{

if (!isEmpty(stack))

return stack->array[stack->top--] ;

return '$';

}

void push(struct Stack\* stack, char op)

{

stack->array[++stack->top] = op;

}

int isOperand(char ch)

{

return (ch >= 'a' && ch <= 'z') ||

(ch >= 'A' && ch <= 'Z');

}

int Prec(char ch)

{

switch (ch)

{

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

int infixToPostfix(char\* exp)

{

int i, k;

struct Stack\* stack = createStack(strlen(exp));

if(!stack)

return -1 ;

for (i = 0, k = -1; exp[i]; ++i)

{

if (isOperand(exp[i]))

exp[++k] = exp[i];

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

push(stack, exp[i]);

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

{

while (!isEmpty(stack) && peek(stack) != '(')

exp[++k] = pop(stack);

if (!isEmpty(stack) && peek(stack) != '(')

return -1;

else

pop(stack);

}

else

{

while (!isEmpty(stack) &&

Prec(exp[i]) <= Prec(peek(stack)))

exp[++k] = pop(stack);

push(stack, exp[i]);

}

}

while (!isEmpty(stack))

exp[++k] = pop(stack );

exp[++k] = '\0';

printf( "%s", exp );

}

int main()

{

char exp[100];

printf("Enter a expression");

scanf("%s",exp);

printf("%s\n",exp);

infixToPostfix(exp);

return 0;

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**

****

**RESULT:**

Thus the above program to convert an infix expression to postfix expression using Stack implementation is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:10**  **DATE:23/6/22** | **IMPLEMENTATION OF QUEUE USING ARRAYS** |

**AIM:**

To create a program to implement queue using arrays.

**ALGORITHM:**

1. Start.

2. Include all the head files.

3. Declare the function void display(int r,int\* arr).

4. Declare the variables n,r=-1,f=-1,element.

5. Read n.

6. Initialize a for loop with int i=0 ,condition i<n with increment i++.

7. If r == n-1 print Queue is full.

8. Else if f == -1 declare f=0.

9. Increment r by 1.

10. Assign arr[r]=element.

11. Print Queue elements are.

12. Call the function void display(r,arr).

13. If r == -1 print Queue is empty.

14. Else initialize a for loop with int i=0,condition i< = r with increment i++.

15. Print arr[i].

16.Stop.

**SOURCE CODE:**

#include <stdio.h>

void display(int r,int\* arr);

int main(){

int n,r=-1,f=-1,element;

scanf("%d",&n);

int arr[n];

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

scanf("%d",&element);

if(r==(n-1)){

printf("Queue is full \n");

}

else{

if(f==-1){

f=0;

}

r++;

arr[r]=element;

}

}

printf("Queue elements are :\n");

display(r,arr);

return 0;

}

void display(int r,int\* arr){

if(r==-1){

printf("\n Queue is empty");

}

else{

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

printf("%d",arr[i]);

printf(" ");

}

}

}

**SAMPLE INPUT:**



**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to implement a queue using Arrays is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:11**  **DATE:23/6/22** | **IMPLEMENTATION OF QUEUE USING LINKED LIST** |

**AIM:**

To implement a queue using using linked list.

**ALGORITHM:**

1. Start.
2. Include all the required header files
3. Declare all user defined functions
4. Define node with two members data and addr
5. Read num
6. Call insert function and insert elements at the end
7. Create the node and insert the node data and its addr as NULL
8. If queue is empty then head=newnode
9. Else store head in temp
10. While temp->addr!=NULL
11. Temp=temp->addr
12. Temp->addr=newnode
13. To display the function , call the display function
14. Display head->addr until head!=NULL
15. Stop

**SOURCE CODE:**

// You are using GCC

#include <stdio.h>

#include<cstdlib>

struct node

{

int data;

struct node\* addr;

};

void insertatend(struct node\*\* q,int value)

{

struct node\* newnode;

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=value;

newnode->addr=NULL;

if((\*q)==NULL)

{

(\*q)=newnode;

}

else

{

struct node\* temp=(\*q);

while(temp->addr!=NULL)

{

temp=temp->addr;

}

temp->addr=newnode;

}

}

void display(struct node\* q)

{

while(q!=NULL)

{

printf("%d",q->data);

printf(" ");

(q)=(q)->addr;

}

}

int main()

{

//let num be the numbers in the list

struct node\* head=NULL;

int num;

while(true)

{

scanf("%d",&num);

if(num>=0)

{

insertatend(&head,num);

}

else

{

break;

}

}

printf("Implementation of Queue using linked list :\n");

display(head);

return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**



**RESULT:**

Thus the program to implement a queue using Linked List is executed successfully.

|  |  |
| --- | --- |
| **EXP NO: 12**  **DATE:29/6/22** | **IMPLEMENTATION OF PRIORITY QUEUE** |

**AIM:**

To create a program for implementing Priority Queue.

**ALGORITHM:**

PUSH(HEAD, DATA, PRIORITY)

Step 1: Create new node with DATA and PRIORITY

Step 2: Check if HEAD has lower priority. If true follow Steps 3-4 and end. Else goto Step 5.

Step 3: NEW -> NEXT = HEAD

Step 4: HEAD = NEW

Step 5: Set TEMP to head of the list

Step 6: While TEMP -> NEXT != NULL and TEMP -> NEXT -> PRIORITY > PRIORITY

Step 7: TEMP = TEMP -> NEXT [END OF LOOP]

Step 8: NEW -> NEXT = TEMP -> NEXT

Step 9: TEMP -> NEXT = NEW

Step 10: End POP(HEAD) Step 2: Set the head of the list to the next node in the list. HEAD = HEAD -> NEXT. Step 3: Free the node at the head of the list Step 4: End

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int priority;

int data;

struct node\*next;

};

struct node\*front=NULL;

void insert();

void de();

void display();

void main(){

int ch;

while(1){

printf("\nPerform operations");

printf("\nEnter the choice:");

scanf("%d",&ch);

switch(ch){

case 1:

insert();

break;

case 2:

de();

break;

case 3:

display();

break;

default:

printf("error");

}}}

void insert(){

struct node\*temp,\*p;

temp=(struct node\*)malloc(sizeof(struct node));

printf("\nEnter the priority:");

scanf("%d",&temp->priority);

printf("\nenter the data:");

scanf("%d",&temp->data);

if(front==NULL){

front=temp;

temp->next=NULL;

}

else{

p=front;

while(p->next!=NULL&&p->next->priority<=temp->priority){

p=p->next;}

p->next=temp;

temp->next=NULL;}

}

void display(){

struct node\*p;

p=front;

while(p!=NULL){

printf("\nThe priority is:%d and the element is:%d",p->priority,p->data);

p=p->next;

}}

void de(){

struct node\*p;

p=front;

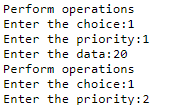
printf("Deleted element is:%d",p->data);

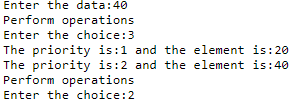
front=p->next;

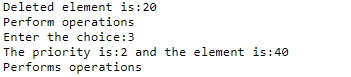
free(p);

}

**SAMPLE INPUT AND OUTPUT:**

****

****

****

**RESULT:**

The above program for implementing Priority Queue is executed successfully.

|  |  |
| --- | --- |
| **EXP NO: 13**  **DATE:29/6/22** | **IMPLEMENTATION OF QUEUE APPLICATION** |

**AIM:**

To write a program to implement queue applications.

**ALGORITHM:**

1. Create a previous with given value and set to NULL.

2 . Create a currrent with given value and set to head.

3 .Create a next with given value and set to head->addr

4. WHILE (head->addr!=NULL).

5. current->addr=previous.

6. current=previous

7.next->next ->addr

8.head = current.

9. Stop WHILE

10. head->addr = previous

11.Stop

**PROGRAM CODE:**

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node\* addr;

};

void Enqueue(struct node\*\* temp,struct node\*\* tail,int value){

struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=value;

newnode->addr=NULL;

if((\*temp)==NULL && (\*tail)==NULL){

(\*temp)=newnode;

(\*tail)=newnode;

}else{

struct node\* q=(\*temp);

while(q->addr!=NULL){

q=q->addr;

}

q->addr=newnode;

(\*tail)=newnode;

}

}

void reverse(struct node\*\* q){

struct node\* previous=NULL;

struct node\* current=(\*q);

struct node\* next=current->addr;

while((\*q)->addr!=NULL){

current->addr=previous;

previous=current;

current=next;

next=next->addr;

(\*q)=current;

}

(\*q)->addr=previous;

}

int Dequeue(struct node\*\* head){

int x=(\*head)->data;

struct node\* temp=(\*head);

(\*head)=(\*head)->addr;

free(temp);

return x;

}

int head(struct node\* head){

return head->data;

}

int tail(struct node\* tail){

return tail->data;

}

void display(struct node\* q){

while(q!=NULL){

printf("%d ",q->data);

q=q->addr;

}

}

int main(){

struct node\* front=NULL;

struct node\* rear=NULL;

int val;

while(1){

scanf("%d",&val);

if(val<0){break;}

Enqueue(&front,&rear,val);

}

printf("Before reversing:\n");

display(front);

printf("\nAfter reversing:\n");

reverse(&front);

display(front);

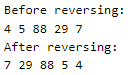
return 0;

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**



**RESULT:**

Hence the program to implement queue applications is implemented and executed.

|  |  |
| --- | --- |
| **EXP NO: 14**  **DATE:5/7/22** | **IMPLEMENTATION OF BINARY SEARCH TREE** |

**AIM:**

To create a program to implement Binary Search Tree.

**ALGORITHM:**

1. Search (root, item)

2. Step 1 - if (item = root → data) or (root = NULL)

3. Return root

4. Else if (item < root → data)

5. Return Search(root → left, item)

6. Else

7. Return Search(root → right, item)

8. END if

9. Step 2 – END

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

struct node {

int key;

struct node \*left, \*right;

};

struct node \*newNode(int item) {

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root) {

if (root != NULL) {

inorder(root->left);

printf("%d -> ", root->key);

inorder(root->right);

}

}

struct node \*insert(struct node \*node, int key) {

if (node == NULL) return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

return node;

}

struct node \*minValueNode(struct node \*node) {

struct node \*current = node;

while (current && current->left != NULL)

current = current->left;

return current;

}

struct node \*deleteNode(struct node \*root, int key) {

if (root == NULL) return root;

if (key < root->key)

root->left = deleteNode(root->left, key);

else if (key > root->key)

root->right = deleteNode(root->right, key);

else {

if (root->left == NULL) {

struct node \*temp = root->right;

free(root);

return temp;

} else if (root->right == NULL) {

struct node \*temp = root->left;

free(root);

return temp;

}

struct node \*temp = minValueNode(root->right);

root->key = temp->key;

root->right = deleteNode(root->right, temp->key);

}

return root;

}

int main() {

struct node \*root = NULL;

int n;

scanf("%d",&n);

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

int val;

scanf("%d",&val);

root = insert(root, val);

}

printf("Inorder traversal: ");

inorder(root);

printf("\nAfter deleting \n");

root = deleteNode(root, 6);

printf("Inorder traversal: ");

inorder(root);

}

**SAMPLE INPUT:**

****

**SAMPLE OUTPUT:**

****

**RESULT:**

The above program to implement Binary Search Tree is executed successfully.

|  |  |
| --- | --- |
| **EXPT NO:15**  **DATE:11/7/22** | **IMPLEMENTATION OF HASHING TECHNIQUES** |

**AIM:**

To create a C program to implement hashing techniques.

**ALGORITHM:**

1. Start

2. Create an array of structure, data (i.e a hash table).

3. Take a key to be stored in hash table as input.

4. Corresponding to the key, an index will be generated.

5. In case of absence of data in that index of array, create one and insert the data item (key and value) into it and increment the size of hash table.

6. In case the data already exists, the new data cannot be inserted if the already present data does not match given key.

7. To display all the elements of hash table, data at each index is extracted and elements (key and value) are read and printed.

8. To remove a key from hash table, we will first calculate its index and extract its data, delete the key in case it matches the given key.

9. Stop

**SOURCE CODE:**

**//Hashing with Separate Chaining**

#include <stdio.h>

#include <stdlib.h>

#define TABLE\_SIZE 10

struct node

{

               int data;

               struct node \*next;

};

struct node \*head[TABLE\_SIZE]={NULL},\*c;

void insert()

{

    int i,key;

    printf("\nEnter a value to insert into hash table\n");

    scanf("%d",&key);

    i=key%TABLE\_SIZE;

    struct node \* newnode=(struct node \*)malloc(sizeof(struct node));

    newnode->data=key;

    newnode->next = NULL;

    if(head[i] == NULL)

        head[i] = newnode;

    else

    {

        c=head[i];

        while(c->next != NULL)

        {

           c=c->next;

        }

        c->next=newnode;

    }

}

void search()

{

    int key,index;

    printf("\nEnter the element to be searched\n");

    scanf("%d",&key);

    index=key%TABLE\_SIZE;

    if(head[index] == NULL)

        printf("\n Search element not found\n");

    else

    {

        for(c=head[index];c!=NULL;c=c->next)

        {

            if(c->data == key)

                {

                    printf("search element found\n");

                    break;

                }

        }

        if(c==NULL)

            printf("\n Search element not found\n");

    }

}

void display()

{

    int i;

    for(i=0;i<TABLE\_SIZE;i++)

          {

           printf("\nEntries at index %d\n",i);

               if(head[i] == NULL)

               {

               printf("No Hash Entry");

               }

               else

               {

                              for(c=head[i];c!=NULL;c=c->next)

                              printf("%d->",c->data);

               }

          }

}

void main()

{

    int opt,key,i;

    while(1)

    {

        printf("\nEnter your option \n 1. Insert \n 2. Display \n 3. Search \n 4.Exit \n");

        scanf("%d",&opt);

        switch(opt)

        {

            case 1:

                insert();

                break;

            case 2:

                display();

                break;

            case 3:

                search();

                break;

            case 4:exit(0);

        }

    }

}

### **//Hashing using linear probing**

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 10

int h[TABLE\_SIZE]={NULL};

void insert()

{

 int key,index,i,flag=0,hkey;

 printf("\nenter a value to insert into hash table\n");

 scanf("%d",&key);

 hkey=key%TABLE\_SIZE;

 for(i=0;i<TABLE\_SIZE;i++)

    {

     index=(hkey+i)%TABLE\_SIZE;

     if(h[index] == NULL)

     {

        h[index]=key;

         break;

     }

    }

    if(i == TABLE\_SIZE)

     printf("\nelement cannot be inserted\n");

}

void search()

{

 int key,index,i,flag=0,hkey;

 printf("\nenter search element\n");

 scanf("%d",&key);

 hkey=key%TABLE\_SIZE;

 for(i=0;i<TABLE\_SIZE; i++)

 {

    index=(hkey+i)%TABLE\_SIZE;

    if(h[index]==key)

    {

      printf("value is found at index %d",index);

      break;

    }

  }

  if(i == TABLE\_SIZE)

    printf("\n value is not found\n");

}

void display()

{

  int i;

  printf("\nelements in the hash table are \n");

  for(i=0;i< TABLE\_SIZE; i++)

  printf("\nat index %d \t value =  %d",i,h[i]);

}

void main()

{

    int opt,i;

    while(1)

    {

        printf("\nEnter the option  \n 1. Insert \n 2. Display \n 3. Search \n 4.Exit \n");

        scanf("%d",&opt);

        switch(opt)

        {

            case 1:

                insert();

                break;

            case 2:

                display();

                break;

            case 3:

                search();

                break;

            case 4:exit(0);

        }

    }

}

### 

**//Quadratic Probing**

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 10

int h[TABLE\_SIZE]={NULL};

void insert()

{

 int key,index,i,flag=0,hkey;

 printf("\nenter a value to insert into hash table\n");

 scanf("%d",&key);

 hkey=key%TABLE\_SIZE;

 for(i=0;i<TABLE\_SIZE;i++)

{

     index=(hkey+i\*i)%TABLE\_SIZE;

     if(h[index] == NULL)

     {

        h[index]=key;

         break;

     }

}

if(i == TABLE\_SIZE)

     printf("\nelement cannot be inserted\n");

}

void search()

{

 int key,index,i,flag=0,hkey;

 printf("\nenter search element\n");

 scanf("%d",&key);

 hkey=key%TABLE\_SIZE;

 for(i=0;i<TABLE\_SIZE; i++)

 {

    index=(hkey+i\*i)%TABLE\_SIZE;

    if(h[index]==key)

    {

      printf("value is found at index %d",index);

      break;

    }

  }

  if(i == TABLE\_SIZE)

    printf("\n value is not found\n");

}

void display()

{

  int i;

  printf("\nelements in the hash table are \n");

  for(i=0;i< TABLE\_SIZE; i++)

              printf("\nat index %d \t value =  %d",i,h[i]);

}

void main()

{

    int opt,i;

    while(1)

    {

        printf("\nEnter your option \n 1. Insert \n 2. Display \n 3. Search \n 4.Exit \n");

        scanf("%d",&opt);

        switch(opt)

        {

            case 1:

                insert();

                break;

            case 2:

                display();

                break;

            case 3:

                search();

                break;

            case 4:exit(0);

        }

    }

}

### 

### **//Double Hashing**

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 10

int h[TABLE\_SIZE]={NULL};

void insert()

{

 int key,index,i,flag=0,hkey,hash2;

 printf("\nEnter a value to insert into hash table\n");

 scanf("%d",&key);

 hkey=key%TABLE\_SIZE;

 hash2 = 7-(key %7);

 for(i=0;i<TABLE\_SIZE;i++)

 {

    index=(hkey+i\*hash2)%TABLE\_SIZE;

    if(h[index] == NULL)

    {

        h[index]=key;

        break;

    }

 }

 if(i == TABLE\_SIZE)

     printf("\nElement cannot be inserted\n");

}

void search()

{

 int key,index,i,flag=0,hash2,hkey;

 printf("\nEnter search element\n");

 scanf("%d",&key);

  hkey=key%TABLE\_SIZE;

 hash2 = 7-(key %7);

 for(i=0;i<TABLE\_SIZE; i++)

 {

    index=(hkey+i\*hash2)%TABLE\_SIZE;

    if(h[index]==key)

    {

      printf("value is found at index %d",index);

      break;

    }

  }

  if(i == TABLE\_SIZE)

    printf("\n value is not found\n");

}

void display()

{

  int i;

  printf("\nElements in the hash table are \n");

  for(i=0;i< TABLE\_SIZE; i++)

    printf("\nat index %d \t value =  %d",i,h[i]);

}

void main()

{

    int opt,i;

    while(1)

    {

        printf("\nEnter your option \n 1. Insert \n 2. Display \n 3. Search \n 4.Exit \n");

        scanf("%d",&opt);

        switch(opt)

        {

            case 1:

                insert();

                break;

            case 2:

                display();

                break;

            case 3:

                search();

                break;

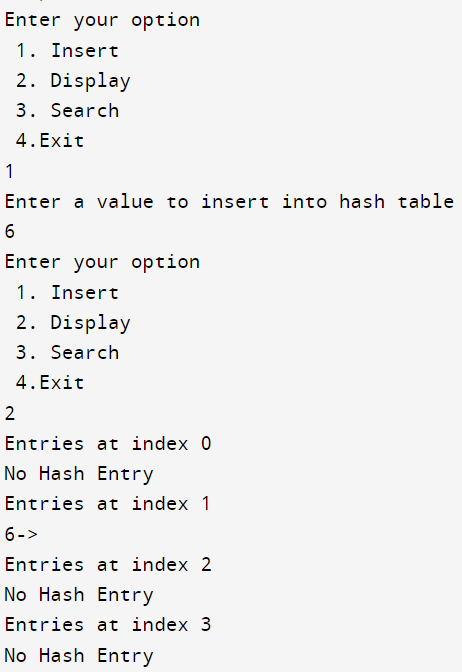
            case 4:exit(0);

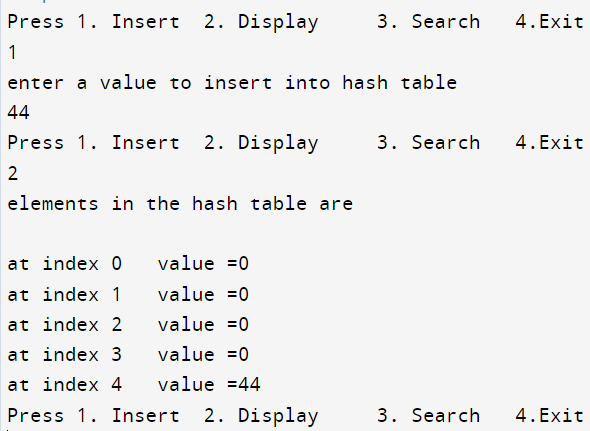
        }

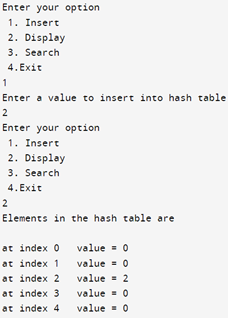
    }

}

**SAMPLE INPUT AND OUTPUT:**







**RESULT:**

The C program to implement hashing techniques is executed successfully.