|  |
| --- |
| **SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES** |
| **COMPUTER SCIENCE AND ENGINEERING PROGRAMME** |

**CSA03 - DATA STRUCTURES**

**LIST OF PROGRAMS**

1. **Write a C program to perform Matrix Multiplication**

#include<stdio.h>

int main()

{

int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;

printf("enter the number of row=");

scanf("%d",&r);

printf("enter the number of column=");

scanf("%d",&c);

printf("enter the first matrix element=\n");

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

{

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

{

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

}

}

printf("enter the second matrix element=\n");

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

{

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

{

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

}

}

printf("multiply of the matrix=\n");

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

{

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

{

mul[i][j]=0;

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

{

mul[i][j]=a[i][k]\*b[k][j];

}

}

}

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

{

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

{

printf("%d\t",mul[i][j]);

}

printf("\n");

}

return 0;

}

**Output**

enter the number of row=2

enter the number of column=2

enter the first matrix element=

1

2

3

4

enter the second matrix element=

5

6

7

8

multiply of the matrix=

14 16

28 32

1. **Write a C program to** **find Odd or Even number from a given set of numbers**

#include<stdio.h>

int main()

{

int array[10],i,num;

printf("enter the no of elements in array: ");

scanf("%d",&num);

printf("enter the elements of array\n");

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

{

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

}

printf("even numbers in a array: ");

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

{

if(array[i]%2==0)

{

printf("%d\t",array[i]);

}

}

printf("\nodd numbers in a array: ");

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

{

if(array[i]%2!=0)

{

printf("%d\t",array[i]);

}

}

}

**Output**

enter the no of elements in array: 3

enter the elements of array

2

3

4

even numbers in a array: 2 4

odd numbers in a array: 3

1. **Write a C program to find Factorial of a given number without using Recursion**

#include<stdio.h>

int main(){

int a=1,numbr;

long int fact=1;

printf("Enter a number: ");

scanf("%d",&numbr);

for(a=1;a<=numbr;a++)

fact=fact\*a;

printf("Factorial of %d is: %ld",numbr,fact);

return 0;

}

**Output**

Enter a number: 5

Factorial of 5 is: 120

1. **Write a C program to find Fibonacci series without using Recursion**

#include<stdio.h>

int main()

{

int n1=0,n2=1,n3,i,num;

printf("Number of elements:");

scanf("%d",&num);

printf("\n%d %d",n1,n2);

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

{

n3=n1+n2;

printf(" %d",n3);

n1=n2;

n2=n3;

}

return 0;

}

**Output**

Number of elements:6

0 1 1 2 3 5

1. **Write a C program to find Factorial of a given number using Recursion**

#include<stdio.h>

long int multiplyNumbers(int n);

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d",&n);

printf("Factorial of %d = %ld", n, multiplyNumbers(n));

return 0;

}

long int multiplyNumbers(int n) {

if (n>=1)

return n\*multiplyNumbers(n-1);

else

return 1;

}

**Output**

Enter a positive integer: 5

Factorial of 5 = 120

1. **Write a C program to find Fibonacci series using Recursion**

#include <stdio.h>

int fibonacci(int num)

{

if (num == 0)

{

return 0;

}

else if (num == 1)

{

return 1;

}

else

{

return fibonacci(num - 1) + fibonacci(num - 2);

}

}

int main()

{

int num;

printf("Enter the number of elements to be in the series : ");

scanf("%d", &num);

for (int i = 0; i < num; i++)

{

printf("%d, ", fibonacci(i));

}

return 0;

}

**Output**

Enter the number of elements to be in the series : 5

0, 1, 1, 2, 3,

1. **Write a C program to implement Array operations such as Insert, Delete and Display.**

#include<stdio.h>

int array[10],size,r,i,op;

int insert();

int delete1();

int display();

int main()

{

printf("Enter the array size: ");

scanf("%d",&size);

do

{

printf("\nEnter the operations: ");

printf("\n1=Insert\n2=delete\n3=display\n4=Exit");

scanf("%d",&r);

switch(r)

{

case 1:

{

insert();

break;

}

case 2:

{

delete1();

break;

}

case 3:

{

display();

break;

}

case 4:

{

break;

}

default:

{

printf("invalid operations");

break;

}

}

}

while(r!=4);

return 0;

}

int insert()

{

printf("enter the number to insert: ");

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

{

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

}

printf("\nInsertion completed!\n");

}

int delete1()

{

printf("enter the position :");

scanf("%d",&op);

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

{

array[i]=array[i+1];

}

printf("\nDeletion completed");

}

int display()

{

printf("Elements in Array: \n");

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

{

printf("%d\t",array[i]);

}

}

**Output**

Enter the array size: 5

Enter the operations:

1=Insert

2=delete

3=display

4=Exit1

enter the number to insert: 2

3

4

5

6

Insertion completed!

1. **Write a C program to search a number using Linear Search method**

#include<stdio.h>

int main(){

int array[10],i,num,s,temp=0;

printf("enter the no of elements in array:");

scanf("%d",&num);

printf("enter the elements of array:\n");

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

{

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

}

printf("enter the element to search: ");

scanf("%d",&s);

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

{

if(array[i]==s)

{

break;

}

}

if(i<num)

{

printf("entered elements found, index = %d",i);

}

else

{

printf("elemnt is not found");

}

}

**Output**

enter the no of elements in array:4

enter the elements of array:

1

2

3

4

5

enter the element to search: 4

entered elements found, index = 2

1. **Write a C program to search a number using Binary Search method**

int main()

{

int c, first, last, middle, n, search, array[100];

printf("Enter number of elements\n");

scanf("%d",&n);

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

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

scanf("%d",&array[c]);

printf("Enter value to find\n");

scanf("%d",&search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while( first <= last )

{

if ( array[middle] < search )

first = middle + 1;

else if ( array[middle] == search )

{

printf("%d found at location %d.\n", search, middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if ( first > last )

printf("Not found! %d is not present in the list.\n", search);

return 0;

}

**Output**

Enter number of elements

3

Enter 3 integers

1

2

3

Enter value to find

3

3 found at location 3.

1. **Write a C program to implement Linked list operations**

#include<stdio.h>

struct node

{

int nData;

struct node\* pNode;

};

struct node\* createLL(int\* nArr, int n)

{

static int i=0;

struct node\* t = NULL;

if(n==0)

return NULL;

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

t->nData = nArr[i++];

t->pNode = createLL(nArr, --n);

return t;

}

void displayLL(struct node \*t)

{

while(t)

{

printf("%d ", t->nData);

t=t->pNode;

}

}

int main()

{

int n=0, i=0, arr[100]={0};

struct node \*t = NULL;

printf("\nEnter the number of elements: ");

scanf("%d", &n);

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

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

printf("\nCreate linked list from array");

t =createLL(arr, n);

printf("\nDisplay Linked List : \n");

if(t)

displayLL(t);

}

**Output**

Enter the number of elements: 4

1

2

4

9

Create linked list from array

Display Linked List :

1 2 4 9

1. **Write a C program to implement Stack operations such as PUSH, POP and PEEK**

#include<stdio.h>

int stack[10],choice,n,top,x,i;

void push();

void pop();

void display();

int main()

{

top = -1;

printf("\n Enter the size of STACK : ");

scanf("%d",&n);

printf("\nSTACK IMPLEMENTATION USING ARRAYS\n");

do

{

printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT\n");

printf("\nEnter the choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

break;

}

default:

{

printf ("\nInvalid Choice\n");

}}}

while(choice!=4);

return 0;

}

void push()

{

if(top >= n - 1)

{

printf("\nSTACK OVERFLOW\n");

}

else

{

printf("Enter a value to be pushed : ");

scanf("%d",&x);

top++; // TOP is incremented after an element is pushed

stack[top] = x; // The pushed element is made as TOP

}}

void pop()

{

if(top <= -1)

{

printf("\nSTACK UNDERFLOW\n");

}

else

{

printf("\nThe popped element is %d",stack[top]);

top--;

}}

void display()

{

if(top >= 0)

{

printf("\nELEMENTS IN THE STACK\n\n");

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

printf("%d\t",stack[i]);

}

else

{

printf("\nEMPTY STACK\n");

}

}

**Output**

Enter the size of STACK : 4

STACK IMPLEMENTATION USING ARRAYS

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter the choice : 1

Enter a value to be pushed : 20

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter the choice : 1

Enter a value to be pushed : 22

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter the choice : 3

ELEMENTS IN THE STACK

22 20

1. **Write a C program to implement the application of Stack (Notations)**
2. **Write a C program to implement Queue operations such as ENQUEUE, DEQUEUE and Display**

#include <stdio.h>

void insert();

void deletef();

void display();

int queue\_array[10];

int rear = - 1;

int front = - 1;

int MAX;

int main()

{

printf("Enter the Queue Size: ");

scanf("%d",&MAX);

int choice;

while (choice!=4)

{

printf("\n1.Insert element to queue \n");

printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert();

break;

case 2:

deletef();

break;

case 3:

display();

break;

case 4:

printf("thank you");

break;

default:

printf("Wrong choice \n");

}

}

}

void insert()

{

int add\_item;

if (rear == MAX - 1)

printf("Queue Overflow \n");

else

{

if (front == - 1)

front = 0;

printf("Inset the element in queue : ");

scanf("%d", &add\_item);

rear = rear + 1;

queue\_array[rear] = add\_item;

}

}

void deletef()

{

if (front == - 1 || front > rear)

{

printf("Queue Underflow \n");

return ;

}

else

{

printf("Element deleted from queue is : %d\n", queue\_array[front]);

front = front + 1;

}

}

void display()

{

int i;

if (front == - 1)

printf("Queue is empty \n");

else

{

printf("Queue is : \n");

for (i = front; i <= rear; i++)

printf("%d ", queue\_array[i]);

printf("\n");

}

}

**Output**

Enter the Queue Size: 4

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 1

Inset the element in queue : 23

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 1

Inset the element in queue : 24

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 1

Inset the element in queue : 25

1.Insert element to queue

2.Delete element from queue

3.Display all elements of queue

4.Quit

Enter your choice : 3

Queue is :

23 24 25

1. **Write a C program to implement** **the Tree Traversals (Inorder, Preorder, Postorder)**

#include <stdio.h>

#include <stdlib.h>

struct node {

int item;

struct node\* left;

struct node\* right;

};

// Inorder traversal

void inorderTraversal(struct node\* root) {

if (root == NULL) return;

inorderTraversal(root->left);

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

inorderTraversal(root->right);

}

// preorderTraversal traversal

void preorderTraversal(struct node\* root) {

if (root == NULL) return;

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

preorderTraversal(root->left);

preorderTraversal(root->right);

}

// postorderTraversal traversal

void postorderTraversal(struct node\* root) {

if (root == NULL) return;

postorderTraversal(root->left);

postorderTraversal(root->right);

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

}

struct node\* createNode(value) {

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

newNode->item = value;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

// Insert on the left of the node

struct node\* insertLeft(struct node\* root, int value) {

root->left = createNode(value);

return root->left;

}

struct node\* insertRight(struct node\* root, int value) {

root->right = createNode(value);

return root->right;

}

int main() {

struct node\* root = createNode(1);

insertLeft(root, 12);

insertRight(root, 9);

insertLeft(root->left, 5);

insertRight(root->left, 6);

printf("Inorder traversal \n");

inorderTraversal(root);

printf("\nPreorder traversal \n");

preorderTraversal(root);

printf("\nPostorder traversal \n");

postorderTraversal(root);

}

**Output**

Inorder traversal

5 ->12 ->6 ->1 ->9 ->

Preorder traversal

1 ->12 ->5 ->6 ->9 ->

Postorder traversal

5 ->6 ->12 ->9 ->1 ->

1. **Write a C program to implement hashing using Linear Probing method**
2. **Write a C program to arrange a series of numbers using Insertion Sort**
3. **Write a C program to arrange a series of numbers using Merge Sort**
4. **Write a C program to arrange a series of numbers using Quick Sort**
5. **Write a C program to implement Heap sort.**
6. **Write a program to perform the following operations:**

**a) Insert an element into a AVL tree.**

**b) Delete an element from a AVL tree.**

**c) Search for a key element in a AVL tree.**

1. **Write a C program to Graph traversal using Breadth First Search**
2. **Write a C program to Graph traversal using Depth First Search**
3. **Implementation of** **Shortest Path Algorithms using Dijkstra’s Algorithm**
4. **Implementation of Minimum Spanning Tree using** **Prim’s Algorithm**
5. **Implementation of Minimum Spanning Tree using** **Kruskal Algorithm**