1. Find and return the maximum and minimum of n given numbers.

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

float max(float arr[], int n) {

float maxnum = arr[0];

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

if (arr[i] > maxnum)

maxnum = arr[i];

}

return maxnum;

}

float min(float arr[], int n) {

float minnum = arr[0];

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

if (arr[i] < minnum)

minnum = arr[i];

}

return minnum;

}

void main() {

int n;

cout << "Enter number of elements:";

cin >> n;

float arr[30];

cout << "Enter elements:";

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

cin >> arr[i];

}

cout << "\nEnter 1 for maximum element \nEnter 2 for minimum element \nChoice:";

int choice;

cin >> choice;

switch (choice) {

case 1:

cout << "\nMaximum element: " << max(arr, n);

break;

case 2:

cout << "\nMinimum element: " << min(arr, n);

break;

default:

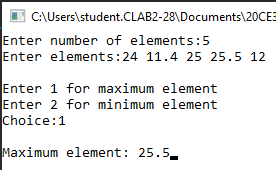
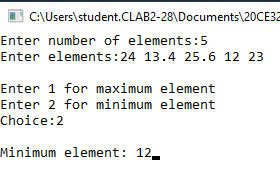
cout << "Invalid choice";

}

\_getch();

}

Output:

2. Selection Sort

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

void SelSort(float arr[], int n) {

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

//int j = i;

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

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

float temp = arr[k];

arr[k] = arr[i];

arr[i] = temp;

}

}

}

}

void main() {

int n;

cout << "Enter number of elements:";

cin >> n;

float arr[30];

cout << "Enter elements:";

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

cin >> arr[i];

}

cout << "\nArray after selection sort: ";

SelSort(arr, n);

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

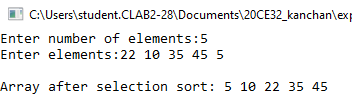
cout << arr[i] << " ";

}

\_getch();

}

Output:



3. Iterative Sum without counting statements

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

float sum(float arr[], int n) {

float total = 0.0;

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

total += arr[i];

}

return total;

}

void main() {

int n;

cout << "Enter number of elements:";

cin >> n;

float arr[30];

cout << "Enter elements:";

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

cin >> arr[i];

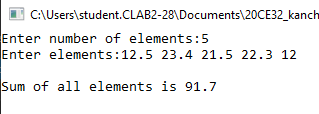
}

cout << "\nSum of all elements is " << sum(arr, n);

\_getch();

}

Output:



4. Iterative Sum with count statements

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

float sum(float arr[], int n, int &count) {

float total = 0.0;

count += 1;

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

count += 1;

total += arr[i];

count += 1;

}

count += 1; //for loop termination

count += 1; // for return statement

return total;

}

void main() {

int n;

cout << "Enter number of elements:";

cin >> n;

float arr[30];

cout << "Enter elements:";

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

cin >> arr[i];

}

int count = 0;

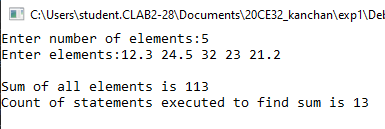
cout << "\nSum of all elements is " << sum(arr, n, count);

cout << "\nCount of statements executed to find sum is " << count;

\_getch();

}

Output:



5. Add two m x n matrices

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

void sumMat(float mat1[][30], float mat2[][30], float sum[][30], int row, int col) {

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

for (int j = 0; j < col; j++)

sum[i][j] = mat1[i][j] + mat2[i][j];

}

void main() {

int r1, r2, c1, c2;

float mat1[30][30], mat2[30][30], sum[30][30];

cout << "Enter number of rows(<30) and columns(<30) for matrix 1 :";

cin >> r1 >> c1;

cout << "Enter number of rows(<30) and columns(<30) for matrix 2 :";

cin >> r2 >> c2;

if (r1 != r2 || c1 != c2)

cout << "\nRows and columns of matrices unequal hence cannot add.";

else {

}

cout << "Enter elements for matrix 1:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++)

cin >> mat1[i][j];

}

cout << "Enter elements for matrix 2:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++)

cin >> mat2[i][j];

}

sumMat(mat1, mat2, sum, r1, c1);

cout << "\nSum of matrices:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++)

cout << sum[i][j] << " ";

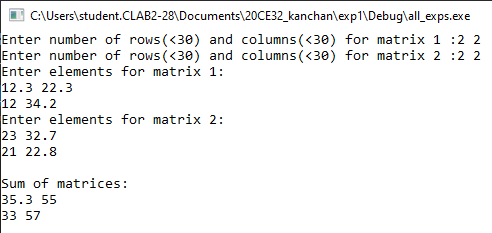
cout << endl;

}

\_getch();

}

Output:



6.Fibonacci Numbers

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

void fibonacci(int n) {

int fib1 = 0, fib2 = 1;

if (n < 1)

cout << "Invalid number";

else if (n == 1)

cout << fib1;

else {

cout << fib1 << " " << fib2 << " ";

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

int fib3 = fib1 + fib2;

cout << fib3 << " ";

fib1 = fib2;

fib2 = fib3;

}

}

}

void main() {

int n;

cout << "Enter limit for Fibonacci Series:";

cin >> n;

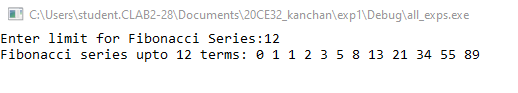
cout << "Fibonacci series upto " << n << " terms: ";

fibonacci(n);

\_getch();

}

Output:



7. Exponentiation

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

float exponentiate(float base, int power) {

float exp = 1;

while (power > 0) {

while ((power % 2) == 0) {

power = power / 2;

base = base \* base;

}

power = power - 1;

exp = exp \* base;

}

return exp;

}

void main() {

float base; int power;

cout << "Enter base and power:";

cin >> base >> power;

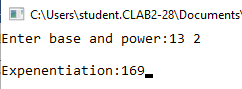
cout << "\nExpenentiation:";

cout << exponentiate(base, power);

\_getch();

}

Output:



8.Insertion Sort

#include<iostream>

#include<conio.h>

using namespace std;

void insertSort(float arr[], int size) {

for (int j = 1; j < size; j++) {

int item = arr[j];

int i = j - 1;

while ((i >= 0) && (item < arr[i])) {

arr[i + 1] = arr[i];

i -= 1;

}

arr[i + 1] = item;

}

}

void main() {

int n;

cout << "Enter number of elements:";

cin >> n;

float arr[30];

cout << "Enter elements:";

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

cin >> arr[i];

}

cout << "\nArray after Insertion sort: ";

insertSort(arr, n);

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

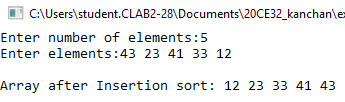
cout << arr[i] << " ";

}

\_getch();

}

Output:



9. Sequential Search

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

int SeqSearch(float arr[], float elem, int size) {

while (size >= 0) {

if (arr[size] == elem)

return size;

size -= 1;

}

return -1;

}

void main() {

float arr[20], elem; int size;

cout << "Enter array size(<20):";

cin >> size;

cout << "Enter elements:";

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

cin >> arr[i];

cout << "Enter element to search for:";

cin >> elem;

int index = SeqSearch(arr, elem, size);

if (index == -1)

cout << "Element not in array";

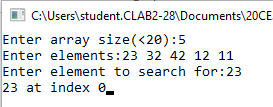
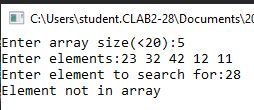
else

cout << elem << " at index " << index;

\_getch();

}

Output:



10. Matrix Transpose

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

void transpose(float arr[][30], int n) {

for (int i = 0; i < n - 1; i++) {

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

int temp = arr[i][j];

arr[i][j] = arr[j][i];

arr[j][i] = temp;

}

}

}

void main() {

float arr[30][30];

int size;

cout << "Enter number of rows (assuming same number of rows and columns):";

cin >> size;

cout << "Enter matrix elements:\n";

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

for (int j = 0; j < size; j++)

cin >> arr[i][j];

}

transpose(arr, size);

cout << "\nArray after transpose:\n";

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

for (int j = 0; j < size; j++)

cout << arr[i][j] << " ";

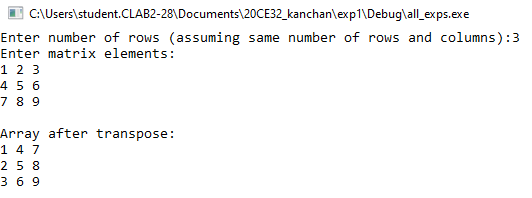
cout << endl;

}

\_getch();

}

Output:



11. Multiplication of n x n matrix

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

void product(float mat1[][30], float mat2[][30], float pdt[][30], int size) {

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

for (int j = 0; j < size; j++) {

pdt[i][j] = 0;

for (int k = 0; k < size; k++) {

pdt[i][j] += (mat1[i][k] \* mat2[k][j]);

}

}

}

}

void main() {

float mat1[30][30], mat2[30][30], pdt[30][30];

int size;

cout << "Enter n for (n x n) matrix 1 and matrix 2:";

cin >> size;

cout << "Enter matrix 1 elements:\n";

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

for (int j = 0; j < size; j++)

cin >> mat1[i][j];

}

cout << "Enter matrix 2 elements:\n";

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

for (int j = 0; j < size; j++)

cin >> mat2[i][j];

}

product(mat1, mat2, pdt, size);

cout << "\nProduct of matrices:\n";

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

for (int j = 0; j < size; j++)

cout << pdt[i][j] << " ";

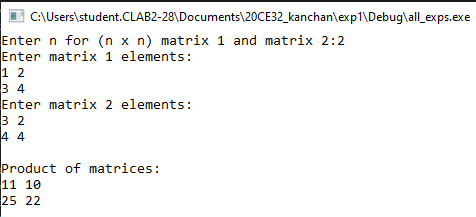
cout << endl;

}

\_getch();

}

Output:



12. Multiplication of m x n and n x p matrices

Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

void product(float mat1[][30], float mat2[][30], float pdt[][30], int r1, int c1, int c2) {

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

pdt[i][j] = 0;

for (int k = 0; k < c1; k++) {

pdt[i][j] += (mat1[i][k] \* mat2[k][j]);

}

}

}

}

void main() {

float mat1[30][30], mat2[30][30], pdt[30][30];

int r1, r2, c1, c2;

cout << "Enter rows and columns for matrix 1:";

cin >> r1 >> c1;

cout << "Enter rows and columns for matrix 2:";

cin >> r2 >> c2;

if (c1 != r2)

cout << "Multiplication not possible";

else {

cout << "Enter matrix 1 elements:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++)

cin >> mat1[i][j];

}

cout << "Enter matrix 2 elements:\n";

for (int i = 0; i < r2; i++) {

for (int j = 0; j < c2; j++)

cin >> mat2[i][j];

}

product(mat1, mat2, pdt, r1, c1, c2);

cout << "\nProduct of matrices:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++)

cout << pdt[i][j] << " ";

cout << endl;

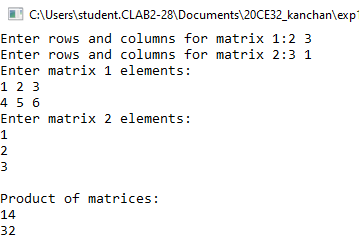
}

\_getch();

}

}

Output:



RECURSION

Program 1: Towers of Hanoi

#include<iostream>

#include<conio.h>

using namespace std;

void towersOfHanoi(int n, char src, char aux, char dest) {

if (n == 1)

{

cout << "Move Disk " << n << " from " << src << " to " << dest << endl;

return;

}

towersOfHanoi(n - 1, src, dest, aux);

cout << "Move Disk " << n << " from " << src << " to " << dest << endl;

towersOfHanoi(n - 1, aux, src, dest);

}

int main() {

char x = 'A';

char y = 'B';

char z = 'C';

int n;

cout << "\n\t\t\t\tTOWER OF HANOI\n";

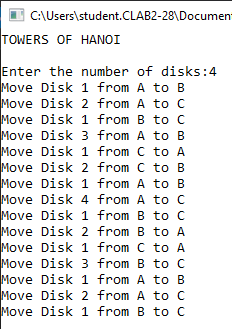
cout << "\nEnter the number of disks: \t";

cin >> n;

towersOfHanoi(n, x, y, z);

}

Output:



Program 2: Permutation Generator. E.g. input is {a, b}, output will be { (a, b), (b, a)}

#include<iostream>

#include<conio.h>

using namespace std;

void swap(char &x, char &y) {

char temp = x;

x = y;

y = temp;

}

void printArr(char arr[], int start, int end) {

for (int i = start; i < end; i++)

cout << arr[i];

cout << endl;

}

void permutations(char arr[], int start, int size) {

if (start >= size) {

printArr(arr, 0, size + 1);

}

else {

char temp;

for (int i = start; i <= size; i++) {

swap(arr[start], arr[i]);

permutations(arr, start + 1, size);

swap(arr[start], arr[i]);

}

}

}

void main() {

int n;

char arr[20];

cout << "Enter number of elements:";

cin >> n;

cout << "Enter elements:";

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

cin >> arr[i];

}

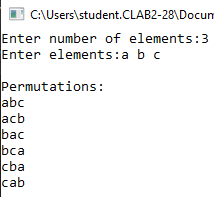
cout << "\nPermutations:\n";

permutations(arr, 0, n-1);

\_getch();

}

Output:



Program 3: Recursive Sum without counting statements.

#include<iostream>

#include<conio.h>

using namespace std;

const int SIZE = 20;

int Sum(int a[SIZE], int n) {

if (n < 0)

return 0;

else

return a[n] + Sum(a, n - 1);

}

int main() {

int n, arr[SIZE];

cout << "Enter the number of elements in the array:";

cin >> n;

cout << "\nEnter the elements: ";

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

cin >> arr[i];

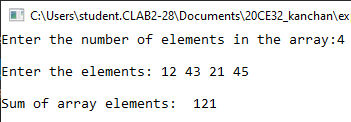
cout << "\nSum of array elements: \t" << Sum(arr, n - 1);

\_getch();

return 0;

}

Output:



Program 4: Recursive Sum with counting statements.

Add statements to increment count at each program step and display the total number of program steps at the end.

#include<iostream>

#include<conio.h>

using namespace std;

const int SIZE = 20;

int cnt = 0;

int Sum(int a[SIZE], int n) {

cnt++; // for if

if (n < 0) {

cnt++; // for return

return 0;

}

else {

cnt++; // for addition, invocation, return

return a[n] + Sum(a, n - 1);

}

}

int main() {

int n, arr[SIZE];

cout << "Enter the number of elements in the array:";

cin >> n;

cout << "\nEnter the elements:";

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

cin >> arr[i];

cout << "\nSum of array elements:" << Sum(arr, n - 1);

cout << "\nCount of statements in Sum():" << cnt;

\_getch();

return 0;

}

Output:

