Procedural Programming Lab Assignments

Name- Tejas Waghmare, Div-N, Roll no-72

Practical no.4(a)

Code:-

#include <stdio.h>

int main() {

int n;

// Input the number of elements

printf("Enter the number of elements: ");

scanf("%d", &n);

if (n <= 0) {

printf("The number of elements should be greater than 0.\n");

return 1;

}

int arr[n];

// Input array elements

printf("Enter %d elements:\n", n);

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

printf("Element %d: ", i + 1);

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

}

// Initialize largest and smallest with the first element

int largest = arr[0], smallest = arr[0];

// Loop to find the largest and smallest elements

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

if (arr[i] > largest) {

largest = arr[i];

}

if (arr[i] < smallest) {

smallest = arr[i];

}

}

// Output the results

printf("Largest element: %d\n", largest);

printf("Smallest element: %d\n", smallest);

return 0;

}

Output:-

A screenshot of a computer

Description automatically generated

A close-up of numbers

Description automatically generated

Practical no.4(b)

Code:-

#include <stdio.h>

// Function to multiply two matrices

void multiplyMatrices(int m, int n, int p, int mat1[m][n], int mat2[n][p], int result[m][p]) {

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

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

result[i][j] = 0;

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

result[i][j] += mat1[i][k] \* mat2[k][j];

}

}

}

}

// Function to input a matrix

void inputMatrix(int rows, int cols, int matrix[rows][cols], const char\* name) {

printf("Enter elements of %s matrix (%dx%d):\n", name, rows, cols);

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

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

printf("Enter element [%d][%d]: ", i + 1, j + 1);

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

}

}

}

// Function to display a matrix

void displayMatrix(int rows, int cols, int matrix[rows][cols]) {

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

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

printf("%d ", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int m, n, p;

// Input dimensions of the matrices

printf("Enter the number of rows and columns of the first matrix (MxN): ");

scanf("%d %d", &m, &n);

printf("Enter the number of columns of the second matrix (Nxp): ");

scanf("%d", &p);

int mat1[m][n], mat2[n][p], result[m][p];

// Input matrices

inputMatrix(m, n, mat1, "first");

inputMatrix(n, p, mat2, "second");

// Multiply matrices

multiplyMatrices(m, n, p, mat1, mat2, result);

// Display the result

printf("Resultant matrix after multiplication:\n");

displayMatrix(m, p, result);

return 0;

}

Output:-

A screenshot of a computer code

Description automatically generated

A close-up of a white background

Description automatically generated

Practical no.5

Code:-

#include <stdio.h>

// Function to compare two strings

int compareStrings(char str1[], char str2[]) {

int i = 0;

while (str1[i] != '\0' && str2[i] != '\0') {

if (str1[i] != str2[i]) {

return 0; // Strings are not equal

}

i++;

}

return (str1[i] == '\0' && str2[i] == '\0'); // Check if both strings end together

}

// Function to reverse a string

void reverseString(char str[], char reversed[]) {

int length = 0;

while (str[length] != '\0') {

length++;

}

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

reversed[i] = str[length - i - 1];

}

reversed[length] = '\0'; // Null-terminate the reversed string

}

// Function to check if a string is a palindrome

int isPalindrome(char str[]) {

char reversed[100];

reverseString(str, reversed);

return compareStrings(str, reversed);

}

int main() {

char str1[100], str2[100];

// Input strings

printf("Enter the first string: ");

scanf("%s", str1);

printf("Enter the second string: ");

scanf("%s", str2);

// Compare strings

if (compareStrings(str1, str2)) {

printf("The strings are equal.\n");

} else {

printf("The strings are not equal.\n");

}

// Check for palindrome

printf("Checking if the first string is a palindrome...\n");

if (isPalindrome(str1)) {

printf("The first string is a palindrome.\n");

} else {

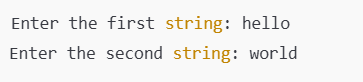
printf("The first string is not a palindrome.\n");

}

return 0;

}

Output:-



A close up of words

Description automatically generated

Practical no.6(a)

Code:-

#include<stdio.h>

int PrimeOrNot(int);

int main()

{

int n1,prime;

printf("\n\n Function : check whether a number is prime number or not :\n");

printf("---------------------------------------------------------------\n");

printf(" Input a positive number : ");

scanf("%d",&n1);

prime = PrimeOrNot(n1);

if(prime==1)

printf(" The number %d is a prime number.\n",n1);

else

printf(" The number %d is not a prime number.\n",n1);

return 0;

}

int PrimeOrNot(int n1)

{

int i=2;

while(i<=n1/2)

{

if(n1%i==0)

return 0;

else

i++;

}

return 1;

}

Output:-

A white paper with black text

Description automatically generated

Practical no.6(b)

1. Without Recursive Function

Code:-

#include <stdio.h>

void generateFibonacci(int n)

{

int a = 0, b = 1, next;

printf("Fibonacci Series: ");

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

{

printf("%d, ", a);

next = a + b;

a = b;

b = next;

}

printf("\n");

}

int main()

{

int n;

printf("Enter the number of terms in the Fibonacci series: ");

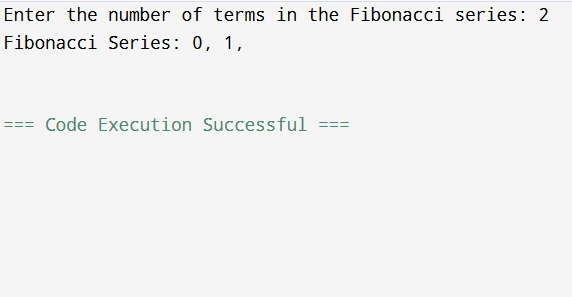
scanf("%d", &n);

generateFibonacci(n);

return 0;

}

Output:-



2. With Recursive Function

Code:-

#include <stdio.h>

int fibonacci(int n)

{

if (n <= 1)

return n;

else

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

}

void generateFibonacciRecursive(int n)

{

printf("Fibonacci Series: ");

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

{

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

}

printf("\n");

}

int main()

{

int n;

printf("Enter the number of terms in the Fibonacci series: ");

scanf("%d", &n);

generateFibonacciRecursive(n);

return 0;

}

Output:-

A screenshot of a computer

Description automatically generated

Practical no.7(a)

Code:-

#include <stdio.h>

#include <stdio.h>

int main()

{

int n;

// Get the number of elements from the user

printf("Enter the number of elements: ");

scanf("%d", &n);

// Declare an array of size n

int arr[n];

// Input n elements into the array

printf("Enter %d elements:\n", n);

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

{

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

}

// Initialize pointer to base address of the array

int \*ptr = arr;

// Print the elements using the pointer

printf("Elements in the array: ");

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

{

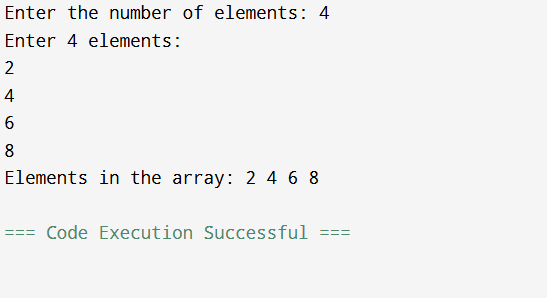
printf("%d ", \*(ptr + i));

}

return 0;

}

Output:-



Practical no.7(b)

Code:-

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n, i, \*ptr, sum = 0;

printf("Enter number of elements: ");

scanf("%d", &n);

ptr = (int\*) calloc(n, sizeof(int));

if(ptr == NULL)

{

printf("Error! memory not allocated.");

exit(0);

}

printf("Enter elements: ");

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

{

scanf("%d", ptr + i);

sum += \*(ptr + i);

}

printf("Sum = %d", sum);

free(ptr);

return 0;

}

Output:-

A screenshot of a computer code

Description automatically generated

Practical no.8

Code:-

#include <stdio.h>

#include <string.h>

#define STUDENTS 10

#define SUBJECTS 5

#define PASS\_MARKS 40

#define DISTINCTION\_MARKS 75

// Structure to store student information

typedef struct {

char name[50];

int marks[SUBJECTS];

} Student;

// Function to find the student with highest marks in a particular subject

void highestInSubject(Student students[], int subjectIndex) {

int highest = -1;

char topStudent[50];

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

if (students[i].marks[subjectIndex] > highest) {

highest = students[i].marks[subjectIndex];

strcpy(topStudent, students[i].name);

}

}

printf("Highest marks in subject %d: %s with %d marks.\n",

subjectIndex + 1, topStudent, highest);

}

// Function to calculate overall percentage result of the class

float overallPercentage(Student students[]) {

int totalMarks = 0, maxMarks = STUDENTS \* SUBJECTS \* 100;

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

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

totalMarks += students[i].marks[j];

}

}

return (float)totalMarks / maxMarks \* 100;

}

// Function to count passing students

int countPassingStudents(Student students[]) {

int passingStudents = 0;

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

int pass = 1;

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

if (students[i].marks[j] < PASS\_MARKS) {

pass = 0;

break;

}

}

if (pass) {

passingStudents++;

}

}

return passingStudents;

}

// Function to count students failing in at least one subject

int countFailingStudents(Student students[]) {

int failingStudents = 0;

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

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

if (students[i].marks[j] < PASS\_MARKS) {

failingStudents++;

break;

}

}

}

return failingStudents;

}

// Function to count distinctions

int countDistinctions(Student students[]) {

int distinctions = 0;

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

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

if (students[i].marks[j] >= DISTINCTION\_MARKS) {

distinctions++;

}

}

}

return distinctions;

}

// Main function

int main() {

Student students[STUDENTS];

// Input student data

printf("Enter student data (name and marks in 5 subjects):\n");

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

printf("Student %d name: ", i + 1);

scanf("%s", students[i].name);

printf("Enter marks in 5 subjects: ");

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

scanf("%d", &students[i].marks[j]);

}

}

// Function calls

int subjectIndex;

printf("Enter subject index (1-5) to find highest marks: ");

scanf("%d", &subjectIndex);

highestInSubject(students, subjectIndex - 1);

printf("Overall class percentage: %.2f%%\n", overallPercentage(students));

printf("Total passing students: %d\n", countPassingStudents(students));

printf("Total failing students (in at least one subject): %d\n", countFailingStudents(students));

printf("Total distinctions in the class: %d\n", countDistinctions(students));

return 0;

}

Output:-

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated