Algorithmic languages – 22. 01. 2019

Lab.1.w.1:

/\* Program for finding the greatest common divisor (GCD) \*/

#include <stdio.h>

int main () {

int a, b, c;

printf ( "Please, enter two nonnegative integers:\n " );

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

while ( b != 0 ) {

c = a % b;

a = b;

b = c;

}

printf ("GCD is %d", a);

return 0;

}

Lab.1.w.2:

/\* Program for calculation the sum of natural even numbers nor esceeding given m \*/

#include <stdio.h>

int main () {

int m, counts = 0, sum = 0;

printf ("Please enter the integer numbers\n");

while ( m > -1) {

scanf ("%d", &m);

if ( m % 2 == 0 ) {

counts++;

sum += m;

}

}

printf ("Sum is %d\nCounts: %d\n", sum, counts);

return 0;

}

Lab1.w.3:

/\* Given a natural number m. It's necessary to determine the number of dividers of this number, less than itself \*/

#include <stdio.h>

int main () {

int m, n, counts = 0;

printf ( "\nPlease, type the integer number\n" );

scanf ( "%d", &m );

while ( m > counts ) {

n = m % 10;

m = m / 10;

counts++;

}

printf ( "The number of divisors is %d", counts+1 );

return 0;

}

Lab.2.w.1:

/\* It's necessary to obtain a new number by writing down the digits of the given number in the reverse order \*/

#include <stdio.h>

int main () {

int a, b;

printf ( "\nPlease enter a positive number\n" );

scanf ("%d", &a);

a > 0;

while ( a != 0 ) {

b = a % 10;

printf ("%d", b);

a = a / 10;

}

return 0;

}

Lab.2.w.2:

/\* It's necessary to check whether the given number is a pelindrome (f.e 404, 1221 and etc.) \*/

#include <stdio.h>

int main () {

int a, b, c;

printf ( "Please type any positive integer number:\n" );

scanf ("%d", &a);

c = a;

while ( a != 0 ) {

b = b \* 10 + a % 10;

a = a / 10;

}

if ( c == b ) {

printf ( "%d\nThis number is pelindrome", b );

}

else {

printf ( "%d\nThe number isn't pelindrome", b );

}

return 0;

}

Lab.2.w.3:

/\* It's necessary to determine how many time the sequence of N arbitrary number changes sign \*/

#include <stdio.h>

int main () {

int n; // number //

int m; // auxilary variables //

int counts = 0;

printf ( "\nPlease type some integers:\n" );

scanf ("%d", &n);

while ( n != 0) {

m = n;

scanf ("%d", &n);

if (m \* n < 0) {

counts++;

}

}

printf ("%d times", counts);

return 0;

}

Lab.2.w.4:

/\* N sets of nonzero integers are given. Each set contains at least 2 elements. A sign of completion of the set is the number of zero.

Find the number of sets whose elements increase \*/

#include <stdio.h>

int main () {

int n; // numbers

int m, k; // auxilary variables

int counts = 0;

while ( n != 0 ) {

scanf ( "%d", &n );

}

do {

m = n;

scanf ( "%d", &n );

k = n;

if ( k > m ) {

counts++;

}

}

while ( n != 0 );

printf ( "%d", counts-1 );

return 0;

}

Lab.3.w.1:

/\* It's necessary to determine whether a positive number is perfect (6, 28, 496, 8128) (This means that the sum of divisors equals to number itself\*/

#include <stdio.h>

#include <math.h>

int main () {

int number, sum=1;

int i;

printf ( "Please, type a positive number:\n" );

scanf ( "%d", &number );

// Step 1: to find the divisors of the number //

for ( i = 2; i < sqrt (number); i++) { // starts from 2, because of dividing by 0 imposibble, and ti divide 1 gets the same result //

if ( number % i == 0 ) {

sum += i + number/i; // for decrasing the number of loop steps //

}

}

// Step 2: to check if it is perfect number //

if ( sum == number )

{

printf ( "%d\tThis number is a perfect number", sum ); }

else {

printf ( "%d\tThis number isn't perfect number" );

}

return 0;

}

Lab.3.w.2:

/\* Write the program that determines friendly number(220 & 284; 5020 & 5564 => The sum of divisors) \*/

#include <stdio.h>

int main () {

int num1, num2 = 0;

int i = 2;

printf ("Please type a positive integer number:\n");

scanf ("%d", &num1);

num1 > 0;

do {

if ( num1 % i == 0 )

num2 += i;

i++;

}

while ( i < num1 );

printf("%d\t%d\nThese numbers are friendly numbers", num1, num2+1); // num2+1 - because during dividing num1 by i, there isn't considered 1 as divisor //

return 0;

}

Lab.3.w.3:

/\* It's necessary to determine whether the 2 numbers are simultaneously primitive numbers (qarwiliqli sade) \*/

#include <stdio.h>

int main () {

printf("Primitive numbers:\n\n");

int num1, num2; // numbers //

int temp;

printf ("Please type the numbers:\n");

scanf("%d%d", &num1, &num2);

// To find the Greatest Common Divisor(GCD) of the numbers //

while ( num2 != 0 ){

temp = num1 % num2;

num1 = num2;

num2 = temp;

}

if ( num1 == 1 ) printf ("These numbers are simultaneously primitive");

else printf ( "These numbers are not simultaneously primitive" );

return 0;

}

Lab.3.w.4:

/\* Write the function to determine the given number is an armstrong or not ( 153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3 ) \*/

#include <stdio.h>

#include <math.h>

int main () {

int num, i, temp;

int sum = 0, counts = 0;

printf("\nPlease enter the positive number:\n\n");

scanf("%d", &num);

// Step 1: To find how many digits the number contains //

temp = num;

do {

temp /= 10;

counts++;

} while ( temp > 0);

printf ("counts: %d\n", counts);

// Step 2: To determine the digits and calculate as main rule //

temp = num;

do {

i = num % 10;

num /= 10;

sum += pow (i, counts);

}

while ( num > 0 );

printf ("sum:%d\n", sum);

// Step 3: To determine this number is an armstrong or not //

if ( sum == temp )

printf ("%d\tThis number is an armstrong\n", sum);

else

printf ("%d\tThis number isn't an armstrong\n", sum);

return 0;

}

Lab.4.w.1:

/\* Write a program to display Pascal's triangle like right triangle \*/

#include <stdio.h>

long factorial(int);

int main () {

printf ("to display Pascal's triangle like right triangle\n");

int i, n, j;

printf ("Please type the number rows:\n");

scanf("%d", &n);

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

for ( j = 0; j <= ( n - i - 2); j++ )

printf (" ");

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

printf ("%ld ", factorial( i ) / (factorial( j )\*factorial( i - j )));

printf ("\n");

}

return 0;

}

long factorial ( int n ){

int j;

long result = 1;

for ( j = 1; j <= n; j++ )

result \*= j;

return ( result );

}

Lab.4.w.2:

/\* Write a program in C to input any number and print it in words \*/

#include <stdio.h>

int main () {

int num, reverse = 0, remainder;

printf ("Please type the number:\n");

scanf ("%d", &num);

while ( num > 0 ) {

remainder = num % 10;

reverse = reverse \* 10 + remainder;

num /= 10;

}

num = reverse;

while ( num != 0 ){

remainder = num % 10;

num /= 10;

switch ( remainder ){

case 0: printf ("Zero\t"); break;

case 1: printf ("One\t"); break;

case 2: printf ("Two\t"); break;

case 3: printf ("Three\t"); break;

case 4: printf ("Four\t"); break;

case 5: printf ("Five\t"); break;

case 6: printf ("Six\t"); break;

case 7: printf ("Seven\t"); break;

case 8: printf ("Eight\t"); break;

case 9: printf ("Nine\t"); break;

default: printf (" ");

}

} return 0;

}

Lab.4.w.3:

/\* Write a program to convert a decimal number to hexadecimal number \*/

#include <stdio.h>

int main () {

int number;

printf ("Please type the decimal number:\n");

scanf ("%d", &number);

printf ("The hexadecimal form is: %X", number);

return 0;

}

Lab.5.w.1:

#include <stdio.h>

long int Factorial( int a ){

do {

if ( a >= 0 ){

return Factorial (a);

}

else return 0;

} while ( a <= 0);

}

int main ()

{

int num;

scanf ( "%d" , &num);

if ( num > 0) {

num = Factorial(num);

printf ("Factorial is %d", num);

}

else

printf ("The problem refers factorial of positive numbers\n");

return 0;

}

Lab.5.w.2:

Sehv elemiwem. Fikirlewecem hele usdunde Vaxt tapsam.)

Lab.5.w.3:

/\* Write a program to convert from decimal to binary \*/

#include <stdio.h>

int main () {

printf ("to convert from decimal to binary\n\n");

int num;

printf ("Please type the decimal number:\n");

scanf ("%d", &num);

int remainder;

while ( num != 0 ){

remainder = num % 2;

int temp = remainder;

}

int i, reverse = 0;

while ( remainder > -1 ){

i = remainder % 10;

reverse = reverse \* 10 + i;

remainder /= 10;

}

printf ("%d", reverse);

return 0;

}

Lab.6.w.1:

#include <stdio.h>

int main ()

{

int a[15];

int max, nmax;

int i;

for ( i = 0; i < 15; i++ )

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

max = a[0];

nmax = 0;

for ( i = 0; i < 15; i++ )

if ( a[i] > max ) {

max = a[i];

nmax = i;

}

printf (" The max element of this array is %d\n And its position is %d", max, nmax );

return 0;

}

Lab.6.w.2:

Elememiwem(((

Lab.6.w.3:

#include <stdio.h>

int main ()

{

// an array //

int a[10];

// index

int i;

// I use loop, because I input an array //

for ( i = 0; i < 10; i++ ){

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

}

// maximal and minimal numbers //

int max\_num, min\_num, imin, imax;

min\_num = a[0];

imin = 0;

max\_num = a[0];

imax = 0;

// to find min number //

for ( i = 1; i < 10; i++){

if ( a[i] < min\_num ){

min\_num = a[i];

imin = i;

}

if ( a[i] > max\_num ){

max\_num = a[i];

imax = i;

}

}

//swap their places//

int c;

c = a[imin];

a[imin] = a[imax];

a[imax] = c;

printf ("max=%d\ti\_max=%d\nmin=%d\ti\_min=%d", mim\_num, max\_num);

// we

return 0;

}

Lab.6.w.4:

/\* Write a program to find two minimal elements in an array \*/

#include <stdio.h>

int main() {

printf ("to find two minimal elements in an array\n\n");

int kvo, i, n, nmin1, nmin2;

int size;

printf ("Please type the size of an array:\n");

scanf ("%d", &size);

int x[size];

printf ("Please type the elements of an array:\n ");

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

scanf ("%lf", &x[i]);

for ( nmin1 = 0, i = 1; i < size; i++){

if ( x[i] < x[nmin1] ) nmin1 = i;

}

for ( kvo = i, i = 0; i < size; i++ ){

if ( i != nmin1 ){

kvo++;

if ( kvo == 1 ) nmin2 = 2;

else if ( x[i] < x[nmin2]) nmin2 = 2;

}

}

printf ( "%lf\t%lf", x[nmin1], x[nmin2]);

return 0;

}

Lab.7.w.1:

#include <stdio.h>

int main () {

int array\_a[10], array\_b[10];

int i,j, counts = -1;

for ( i = 0; i < 10; i++ ) {

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

}

for ( i = 0; i < 10; i++ ) {

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

counts++;

array\_b[counts] = array\_a[i];

}

else {

printf ("\t");

}

}

for ( j = 0; j < counts; j++) {

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

}

return 0;

}

Lab.7.w.2:

#include <stdio.h>

#include <math.h>

int main () {

int array[10];

int max\_a , min\_a; // max and min elements of array //

int imin, imax; // indices of max and min elements //

int i,j, c[10], counts = 0, geo\_mean; // aux value //

// Step 1: to find the max and min elements and theirs index of an array //

printf ("\nPlease type the elements of an array:\n");

for ( i = 0; i < 10; i++ ) {

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

}

max\_a = array[0]; imax = 0;

min\_a = array[0]; imin = 0;

for ( i = 1; i < 10; i++ ){

if ( array[i] > max\_a ) {

max\_a = array[i]; imax = i;

}

if ( array[i] < min\_a ) {

min\_a = array[i]; imin = i;

}

}

// Step 2: check whether max and min elements how located in array //

int q; // auxilary index //

if ( imin > imax ) {

q = imax;

imax = imin;

imin = q;

}

// Step 3: find the elements located between imin and imax, then find the products of prime ones //

long int p = 1;

for ( i = imin; i < imax; i++ ) {

int isPrime = 1;

for ( j = 2; j < sqrt (array[i]); j++) {

if ( array[i] % j == 0){

isPrime = 0;

break;

}

}

if ( isPrime == 1) {

counts++;

p \*= array[i];

}

}

// Step 4: to find geometric mean of these prime numbers //

geo\_mean = pow ( p, 1/counts );

// Step 5: to get final result //

printf ("%d\n%d\n%d\n", max\_a, min\_a, geo\_mean );

return 0;

}

Lab.8.w.1:

/\* Write the elements in increasing order as using Quick sort algorithm of the given array \*/

#include <stdio.h>

void quick\_Sort ( int arr[10], int a, int b );

int main () {

int size;

printf("Please type the size of an array:\n");

scanf ("%d", &size);

int array\_q[size];

int i;

printf ("Please type the elements:\n");

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

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

}

quick\_Sort( array\_q, 0, size - 1 );

printf ( "Sorted elemets:\n" );

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

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

}

return 0;

}

void quick\_Sort ( int arr[10], int left, int right ){

int i = left, j = right;

int temp;

int pivot = arr[(left+right)/2];

while ( i <= j ) {

while ( arr[i] < pivot )

i++;

while ( arr[j] > pivot )

j--;

if ( i <= j ){

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

}

// recursion //

if ( left < j )

quick\_Sort ( arr, left, j );

if ( i < right )

quick\_Sort ( arr, i, right );

}

Lab.8.w.2:

/\* Write the elements in increasing order as using Bubble sort algorithm of the given array \*/

#include <stdio.h>

int main () {

int array\_b[10];

int i,j;

int c;

printf ("\nPlease type 10 elements of an array:\n");

for ( i = 0; i < 10; i++ ) {

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

}

for ( i = 0; i < 9; i++ ) { // because, bubble sort algorithm begins with the last element of an array //

for ( j = 8; j >= i; j-- ) {

if ( array\_b[j] > array\_b[j+1] ) {

c = array\_b[j];

array\_b[j] = array\_b [j+1];

array\_b [j+1] = c;

}

}

}

printf ("Sorted array:\n");

for ( i = 0; i < 10; i++ ){

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

}

}

Lab.9.w.1:

/\* Sequential searching algorithm \*/

#include <stdio.h>

int main () {

int size;

printf ("\nPlease type the size of list(array)\n");

scanf ("%d", &size); // because the size of list can be more //

int list[size], i;

int t\_val; // target value //

printf ("\nPlease type the target value:\n");

scanf ("%d", &t\_val);

printf ("\nPlease type elements of list(array):\n");

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

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

}

int it\_val; // index of target value //

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

if (t\_val == list[i])

it\_val = i;}

printf("target value is located in %dth element", it\_val +1 );

return 0;

}

Lab.9.w.2:

/\* Binary search algorithm \*/

#include <stdio.h>

int main () {

int m; // searching value //

printf ("The codes of letters are:\n");

printf ("A=01; a=001; B=02; b=002; C=03; c=003; D=04; d=004; E=05; e=005\n;F=06; f=006; G=07; g==007; H=08; h==008 ");

printf ("I=09; i=009; J==10\n; j==010; K=11; k=011; L=12; l=012; M=13; m=013; N=14; n=014; O=15; o=015;\nP=16; p=016 ");

printf ("Q=17; q=017; R=18; r=018; S=19; s=019; T=20; t=020;\nU=21; u=021; V=22; v=022; W=23; w=023; X=24; x=024; Y=25; y=025; Z=26; z=026\n");

int size;

printf ("Please, type the size of an array:\n");

scanf ("%d", &size);

int a[size];

printf("Please tye the elements of an array:\n");

int i, j;

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

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

}

printf ("\nPlease type the searched element:\n");

scanf ("%d", &m);

int temp1;

for ( i = 0, j = size-1; i <= (i+j)/2; i++ ) {

if ( a[i] == m ) {

temp1 = i;

break;

}

else { for ( i = (i+j)/2, j = size -1; i < j; i++){

if ( a[i] == m ){

temp1 = i;

break;

}

}

}

}

printf ("%d", temp1+1);return 0;

}

Lab.10.w.1:

/\* Write a program multiplication of two square matrices: \*/

#include <stdio.h>

int main (){

printf ("Multiplication of two matrices:");

int temp1, temp2, temp3, temp4;

int rows, columns, i, j;

// Step 1: declare the sizes of first matrix and input it //

printf ("\nPlease type the number of rows and columns of 1st matrix:\n");

scanf ("%d%d", &rows, &columns);

temp1 = rows;

temp2 = columns;

int a[rows][columns];

printf ("\nPlease type the elements of 1st two dimensional matrix:\n");

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

for ( j = 0; j < columns; j++ ){

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

}

}

// Step 2: declare the sizes of second matrix //

printf ("\nPlease type the number of rows and columns of 2nd matrix:\n");

scanf ("%d%d", &rows, &columns);

temp3 = rows;

temp4 = columns;

// Step 3: to check the main condition of multiplication of matrices //

if (temp2 == temp3){

//Input the 2nd matrix //

int b[rows][columns];

printf ("\nPlease type the elements of 2nd two dimensional matrix:\n");

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

for ( j = 0; j < columns; j++ ){

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

}

}

//take another matrix to represent the result //

int c[temp1][temp4], sum = 0;

for ( i = 0; i < temp1; i++ ) {

for ( j = 0; j < temp4; j++ ){

for ( int k = 0; k < temp2; k++){ // k < temp2 || k < temp3 not difference, because temp2 = temp3; //

// we take auxilary k variable to represent the location of element in resulting matirx //

sum += a[i][k] \* b[k][j];

}

c[i][j] = sum;

sum = 0;

}

}

// Step 4: Output of our result //

for ( i = 0; i <temp1; i++ ) {

for ( j = 0; j < temp4; j++ ) {

printf ("%4d", c[i][j]);

} printf ("\n");

}

}

else {

printf ("In this case, matrices can't be multiplied.\n");

}

return 0;

}

Lab.10.w.2:

/\* Write a program to find transpose of a given matrix \*/

#include <stdio.h>

int main () {

printf("Transpose of a matrix:\n");

// declare the size of matrix and input elements: //

printf("\nPlease type the sizes of the matrix:\n");

int rows, columns;

scanf ("%d%d", &rows, &columns);

printf("\nPlease type the elements of matrix:\n");

int a[rows][columns];

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

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

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

}

}

// to find the transpose of that matrix //

int c;

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

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

if ( i == j ){

break;

}else{

c = a[i][j];

a[i][j] = a[j][i];

a[j][i] = c;}

}

}

// to change the sizes of resulting matrix //

c = columns;

columns = rows;

rows = c;

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

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

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

}printf ("\n");

}

return 0;

}

Lab.11.w.1:

/\* Write a program to print lower triangular matrix \*/

#include <stdio.h>

int main () {

printf ("The lower triangular form:\n\n");

int rows, columns;

printf ("Please type the size of matrix:\n");

printf("Type rows:");

scanf ("%d", &rows);

printf ("Type columns:");

scanf("%d", &columns);

printf ("\n");

if ( rows == columns ){

int mat[rows][columns];

int i, j;

printf ("\nPlease type the elements:\n");

for ( i = 1; i <= rows; i++ ){

for ( j = 1; j <= columns; j++ ){

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

}

}

int n = 1;

for ( i = 1; i <= rows; i++ ){

for ( j = 1; j <= columns; j++ ){

if ( i < j && mat[i][j] != 0 ) {

n = 0;

break;

}

}

}

if ( n == 0 ) printf ("Error...");

else if ( n == 1 ){

printf ("Is Lower:\n");

printf ("\n");

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

for ( i = 1; i <= rows; i++ ){

for ( j = 1; j <= columns; j++ ){

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

} printf ("\n");

}

}

}

else printf("The lower form can occur in square matrix...");

return 0;

}

Lab.11.w.2:

/\* Write a program to print lower triangular matrix \*/

#include <stdio.h>

int main () {

printf ("The lower triangular form:\n\n");

int rows, columns;

printf ("Please type the size of matrix:\n");

printf("Type rows:");

scanf ("%d", &rows);

printf ("Type columns:");

scanf("%d", &columns);

printf ("\n");

if ( rows == columns ){

int mat[rows][columns];

int i, j;

printf ("\nPlease type the elements:\n");

for ( i = 1; i <= rows; i++ ){

for ( j = 1; j <= columns; j++ ){

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

}

}

int n = 1;

for ( i = 1; i <= rows; i++ ){

for ( j = 1; j <= columns; j++ ){

if ( i < j && mat[i][j] != 0 ) {

n = 0;

break;

}

}

}

if ( n == 0 ) printf ("Error...");

else if ( n == 1 ){

printf ("Is Lower:\n");

printf ("\n");

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

for ( i = 1; i <= rows; i++ ){

for ( j = 1; j <= columns; j++ ){

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

} printf ("\n");

}

}

}

else printf("The lower form can occur in square matrix...");

return 0;

}

Lab.13.w.1:

/\* Write a program in C to find the length of a string without using library function \*/

#include <stdio.h>

#include <string.h>

int main () {

char L[30];

printf ("Enter the L string:\n");

gets (L);

int counts = 1, i;

while ( L[i] != '\0' ){

if ( L[i] != '\0' ) counts++;

i++;

}

printf ( "Length of string = %d", counts );

return 0;

}

Lab.13.w.2:

/\* Write a program to count the total number of words in string \*/

#include <stdio.h>

#include <string.h>

int main () {

char BLH[100];

printf ("Enter the BLH string:\n");

gets(BLH);

int counts = 1, i;

while ( BLH[i] != '\0'){

if ( BLH[i] == ' ' || BLH[i] == '\n' || BLH[i] == '\t') {

counts++;

} i++;

}

printf ("%d", counts);

return 0;

}

Lab.13.w.3:

/\* Write a program to extract a substring from a given string \*/

#include <stdio.h>

#include <string.h>

int main () {

char Kovser[100], Rena[100];

printf ("Enter the Kovser string:\n");

gets(Kovser);

strncpy(Rena, Kovser+23, 55);

puts(Rena);

return 0;

}

Arada eliye bilmediyim tek – tuk meseleler var, bir de ki, son 3 meselede stringlerin adina gulmiyin :D :D :D

Umid edirem iwe yarayacaq, **!GOOD LUCK TO EVERYONE!**