# Q1. Use structures and find and print difference between two time periods (hh:mm:ss format).

#include <stdio.h>

#include <stdlib.h>

struct time{

int h;

int m;

int s;

}t1,t2;

typedef struct time t;

void difference(t t1,t t2)

{

int h,m,s;

if(t1.s<t2.s){

--t1.m;

t1.s+=60;

}

s=t1.s-t2.s;

if(t1.m<t2.m){

--t1.h;

t1.m+=60;

}

m=t1.m-t2.m;

h=t1.h-t2.h;

printf("\nThe difference of the two timelines is -->\n");

printf("%d:%d:%d",h,m,s);

}

int main()

{

int i,j;

printf("Enter the valid time values inside t1\n");

printf("Value of h1 : ");

scanf("%d",&(t1.h));

printf("Value of m1 : ");

scanf("%d",&(t1.m));

printf("Value of s1 : ");

scanf("%d",&(t1.s));

printf("\nEnter the time values inside t2\n");

printf("Value of h2 : ");

scanf("%d",&(t2.h));

printf("Value of m2 : ");

scanf("%d",&(t2.m));

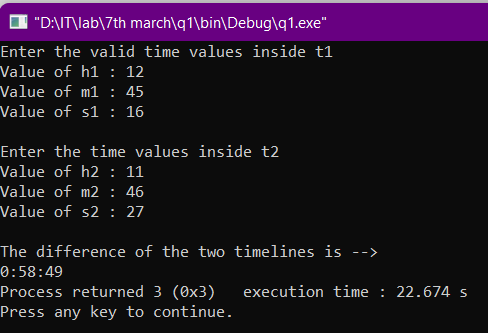
printf("Value of s2 : ");

scanf("%d",&(t2.s));

difference(t1,t2);

return 3;

}



# Q2. Use structures and find and print difference between two dates

# (dd:mm:yyyy) format.

#include <stdio.h>

#include <stdlib.h>

struct date{

int d;

int m;

int y;

}d1,d2;

typedef struct date dt;

void difference(dt d1,dt d2)

{

int dd,mm,yy;

if(d1.m==1||3||5||7||8||10||12){

if(d2.m==1||3||5||7||8||10||12){

if(d1.d<d2.d){

d1.d+=31;

--d1.m;

}

dd=d1.d-d2.d;

if(d1.m<d2.m){

d1.m+=12;

--d1.y;

}

mm=d1.m-d2.m;

yy=d1.y-d2.y;

}

}

else{

if(d1.d<d2.d){

d1.d+=30;

--d1.m;

}

dd=d1.d-d2.d;

if(d1.m<d2.m){

d1.m+=12;

--d1.y;

}

mm=d1.m-d2.m;

yy=d1.y-d2.y;

}

printf("\nThe difference of the dates is -->\n");

printf("%d/%d/%d",dd,mm,yy);

}

int main()

{

printf("Enter the valid date values inside d1\n");

printf("Value of dd : ");

scanf("%d",&(d1.d));

printf("Value of mm : ");

scanf("%d",&(d1.m));

printf("Value of yyyy : ");

scanf("%d",&(d1.y));

printf("\nEnter the time values inside d2\n");

printf("Value of dd : ");

scanf("%d",&(d2.d));

printf("Value of mm : ");

scanf("%d",&(d2.m));

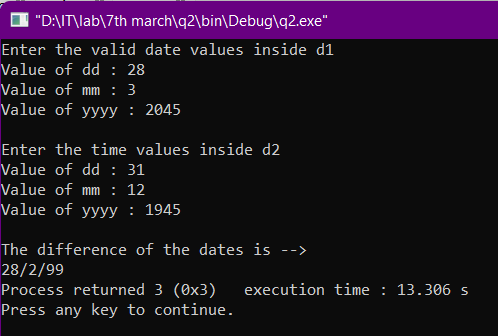
printf("Value of yyyy : ");

scanf("%d",&(d2.y));

difference(d1,d2);

return 3;

}



## Q3. Find a person’s age from their birth date.

#include <stdio.h>

#include <stdlib.h>

struct date

{

int d;

int m;

int y;

} d1, d2;

typedef struct date dt;

void difference(dt d1, dt d2)

{

int dd, mm, yy;

if (d1.m == 1 || 3 || 5 || 7 || 8 || 10 || 12)

{

if (d1.d < d2.d)

{

d1.d += 31;

--d1.m;

}

dd = d1.d - d2.d;

if (d1.m < d2.m)

{

d1.m += 12;

--d1.y;

}

mm = d1.m - d2.m;

yy = d1.y - d2.y;

}

else if (d1.m == 2 && leap(d1.y) == 0)

{

if (d1.d < d2.d)

{

d1.d += 28;

--d1.m;

}

dd = d1.d - d2.d;

if (d1.m < d2.m)

{

d1.m += 12;

--d1.y;

}

mm = d1.m - d2.m;

yy = d1.y - d2.y;

}

else if (d1.m == 2 && leap(d1.y) == 1)

{

if (d1.d < d2.d)

{

d1.d += 29;

--d1.m;

}

dd = d1.d - d2.d;

if (d1.m < d2.m)

{

d1.m += 12;

--d1.y;

}

mm = d1.m - d2.m;

yy = d1.y - d2.y;

}

else

{

if (d1.d < d2.d)

{

d1.d += 30;

--d1.m;

}

dd = d1.d - d2.d;

if (d1.m < d2.m)

{

d1.m += 12;

--d1.y;

}

mm = d1.m - d2.m;

yy = d1.y - d2.y;

}

printf("\nYour age is -->\n");

printf("%d years, %d months and %d days", yy, mm, dd);

}

int leap(int year)

{

if (year % 400 == 0)

{

return 1;

}

// not a leap year if divisible by 100

// but not divisible by 400

else if (year % 100 == 0)

{

return 0;

}

// leap year if not divisible by 100

// but divisible by 4

else if (year % 4 == 0)

{

return 1;

}

// all other years are not leap years

else

{

return 0;

}

}

int main()

{

printf("Enter the present date\n");

printf("Value of dd : ");

scanf("%d", &(d1.d));

printf("Value of mm : ");

scanf("%d", &(d1.m));

printf("Value of yyyy : ");

scanf("%d", &(d1.y));

printf("\nEnter the birth date\n");

printf("Value of dd : ");

scanf("%d", &(d2.d));

printf("Value of mm : ");

scanf("%d", &(d2.m));

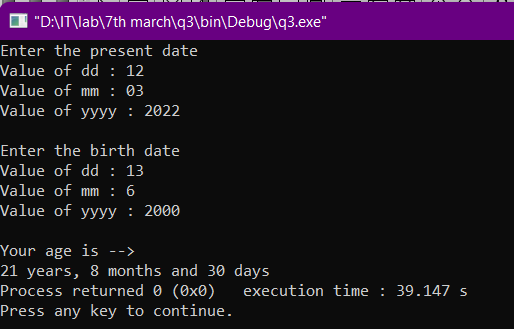
printf("Value of yyyy : ");

scanf("%d", &(d2.y));

difference(d1, d2);

return 0;

}



## Q4. Perform basic DMAS operations on two complex numbers.

#include <stdio.h>

#include <stdlib.h>

int i;

struct input

{

float rl;

float cmp;

}n1,n2;

typedef struct input in;

void division(in n1,in n2)

{

float real,imaginary;

real=((n1.rl\*n2.rl)+(n1.cmp\*n2.cmp))/((n2.rl\*n2.rl)+(n2.cmp\*n2.cmp));

imaginary=((n1.cmp\*n2.rl)-(n1.rl\*n2.cmp))/((n2.rl\*n2.rl)+(n2.cmp\*n2.cmp));

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

printf("----");

}

printf("\nResult after division : %.3f + %.3fi \n",real,imaginary);

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

printf("----");

}

printf("\n");

}

void multiply(in n1,in n2)

{

float real,imaginary;

real=((n1.rl\*n2.rl)-(n1.cmp\*n2.cmp));

imaginary=((n1.rl\*n2.cmp)+(n1.cmp\*n2.rl));

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

printf("----");

}

printf("\nResult after multiplication : %.3f + %.3fi \n",real,imaginary);

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

printf("----");

}

printf("\n");

}

void add(in n1,in n2)

{

float real,imaginary;

real=n1.rl+n2.rl;

imaginary=n1.cmp+n2.cmp;

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

printf("----");

}

printf("\nResult after addition : %.3f + %.3fi \n",real,imaginary);

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

printf("----");

}

printf("\n");

}

void subtract(in n1,in n2)

{

float real,imaginary;

real=n1.rl-n2.rl;

imaginary=n1.cmp-n2.cmp;

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

printf("----");

}

printf("\nResult after subtraction : %.3f + %.3fi \n",real,imaginary);

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

printf("----");

}

printf("\n");

}

int main()

{

char smb;

printf("Enter the values for complex numbers --> \n\n");

printf("R1 : ");

scanf("%f",&n1.rl);

printf("I1 : ");

scanf("%f",&n1.cmp);

printf("\n");

printf("R2 : ");

scanf("%f",&n2.rl);

printf("I2 : ");

scanf("%f",&n2.cmp);

Label:

printf("\nPress '/', '\*', '+', '-' for respective operations & 0 for exit\n");

Label1:

scanf("%s",&smb);

switch (smb)

{

case '/' :

{

division(n1,n2);

goto Label;

break;

}

case '\*' :

multiply(n1,n2);

goto Label;

break;

case '+' :

{

add(n1,n2);

goto Label;

break;

}

case '-' :

{

subtract(n1,n2);

goto Label;

break;

}

case '0':

{

break;

}

default :

{

printf("\nINVALID INPUT\nENTER AGAIN : ");

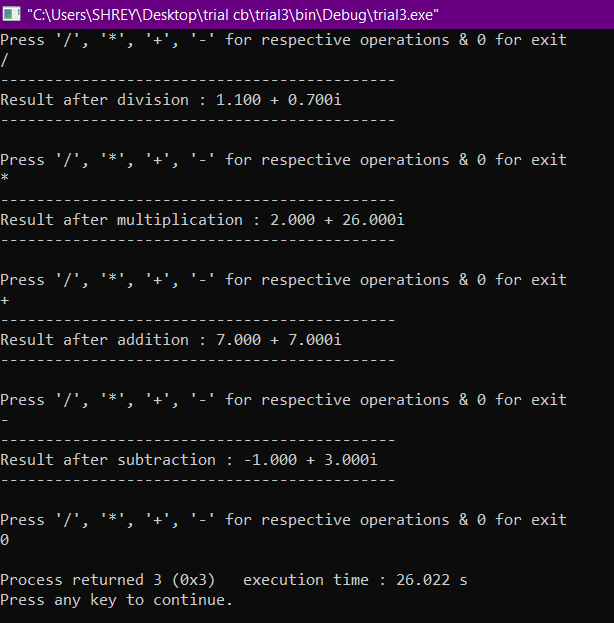
goto Label1;

}

}

return 3;

}



## Q5. With the truth table of various logic gates using structures.

#include <stdio.h>

#include <stdlib.h>

struct data

{

int a;

int b;

}d;

void AND(int m,int n)

{

if((m&&n)==1) printf("OUTPUT : '1'");

else printf("OUTPUT : '0'");

}

void OR(int m,int n)

{

if((m||n)==1) printf("OUTPUT : '1'");

else printf("OUTPUT : '0'");

}

void NAND(int m,int n)

{

if((m&&n)==1) printf("OUTPUT : '0'");

else printf("OUTPUT : '1'");

}

void NOR(int m,int n)

{

if((m&&n)==0) printf("OUTPUT : '1'");

else printf("OUTPUT : '0'");

}

void XOR(int m,int n)

{

if(((m&&n)==0)||((m&&n)==1)) printf("OUTPUT : '0'");

else printf("OUTPUT : '1'");

}

void XAND(int m,int n)

{

if(((m&&n)==0)||((m&&n)==1)) printf("OUTPUT : '1'");

else printf("OUTPUT : '0'");

}

int main()

{

printf("\nPRINTING VARIOUS LOGIC GATES --> \n");

printf("\nPRESS 1 FOR 'AND' GATE\n");

printf("PRESS 2 FOR 'OR' GATE\n");

printf("PRESS 3 FOR 'NAND' GATE\n");

printf("PRESS 4 FOR 'NOR' GATE\n");

printf("PRESS 5 FOR 'XOR' GATE\n");

printf("PRESS 6 FOR 'XAND' GATE\n");

printf("PRESS 7 FOR EXIT\n");

label:

printf("\n\nEnter your choice : ");

char input;

scanf("%s",&input);

if(input=='7'){

return 3;

}

// if(input!=('1'||'2'||'3'||'4'||'5'||'6'||'7')){

// goto label2;

// }

printf("Number 1 : ");

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

printf("Number 2 : ");

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

switch (input)

{

case '1':

{AND(d.a,d.b);

goto label;

break;}

case '2':

{OR(d.a,d.b);

goto label;

break;}

case '3':

{NAND(d.a,d.b);

goto label;

break;}

case '4':

{NOR(d.a,d.b);

goto label;

break;}

case '5':

{XOR(d.a,d.b);

goto label;

break;}

case '6':

{XAND(d.a,d.b);

goto label;

break;}

case '7':

{break;}

// label2:

default:

{printf("//Enter a valid input//");

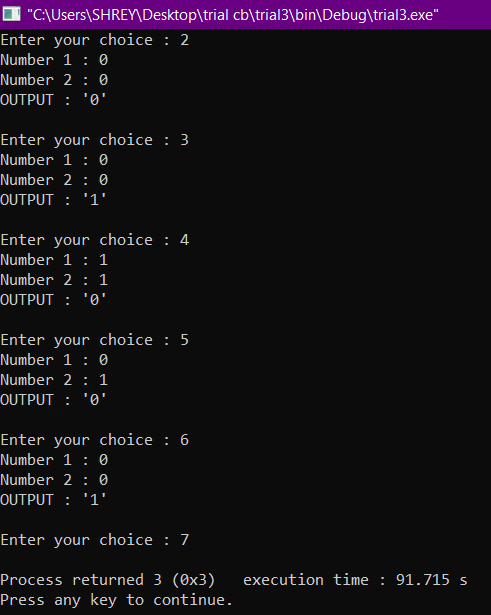
goto label;

break;}

}

return 3;

}



## Q6. Use structures to initialize various triangles (sides and/or angles) and print their types.

#include <stdio.h>

#include <stdlib.h>

int i,j;

struct triangle

{

float s1;

float s2;

float s3;

float a1,a2,a3;

}tr;

typedef struct triangle T;

void sides(float a,float b,float c)

{

if(a==b==c) printf("EQUILATERAL TRIANGLE\n");

else if((a==b)||(b==c)||(c==a)) printf("ISOSCELES TRIANGLE\n");

else printf("SCALANE TRIANGLE\n");

}

void angle(float a,float b,float c)

{

if((a<90)&&(b<90)&&(c<90)) printf("ACUTE ANGLED TRIANGLE\n");

if((a==90.000000)||(b==90.000000)||(c==90.000000)){

printf("RIGHT ANGLED TRIANGLE\n");

if((a==b)||(b==c)||(c==a)) printf("ISOSCELES TRIANGLE\n");

}

else if((a>90)||(b>90)||(c>90)) printf("OBTUSE ANGLED TRIANGLE\n");

}

int main()

{

printf("Enter the dimensions of a triangle -->\n");

for(i=0;i<30;++i){

printf("--");

}

printf("\n");

char sym;

label:

printf("Press S for sides || Press A for angles : ");

scanf("%c",&sym);

if(sym=='s'||sym=='S'){

label1:

printf("S1 : ");

scanf("%f",&tr.s1);

printf("S2 : ");

scanf("%f",&tr.s2);

printf("S3 : ");

scanf("%f",&tr.s3);

float max=tr.s1;

if(tr.s2>=max) max=tr.s2;

if(tr.s3>=max) max=tr.s3;

if(max==tr.s1){

if((tr.s2+tr.s3)>max) sides(tr.s1, tr.s2, tr.s3);

else {

printf("INVALID SIDES||ENTER AGAIN\n");

goto label1;

}

}

if(max==tr.s2){

if((tr.s1+tr.s3)>max) sides(tr.s1, tr.s2, tr.s3);

else {

printf("INVALID SIDES||ENTER AGAIN\n");

goto label1;

}

}

if(max==tr.s3){

if((tr.s2+tr.s1)>max) sides(tr.s1, tr.s2, tr.s3);

else {

printf("INVALID SIDES||ENTER AGAIN\n");

goto label1;

}

}

}

else if(sym=='a'||sym=='A'){

label2:

printf("A1 : ");

scanf("%f",&tr.a1);

printf("A2 : ");

scanf("%f",&tr.a2);

printf("A3 : ");

scanf("%f",&tr.a3);

if((tr.a1+tr.a2+tr.a3)==180.00000) angle(tr.a1,tr.a2,tr.a3);

else {

printf("INVALID ANGLES||ENTER AGAIN\n");

goto label2;

}

}

else{

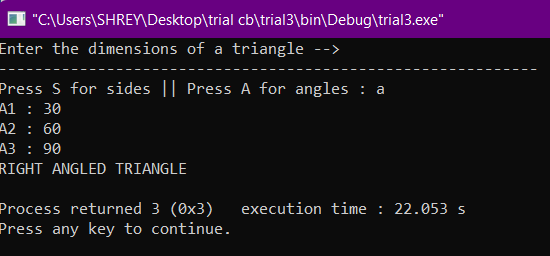
printf("|INVALID INPUT||PLZ ENTER AGAIN|\n");

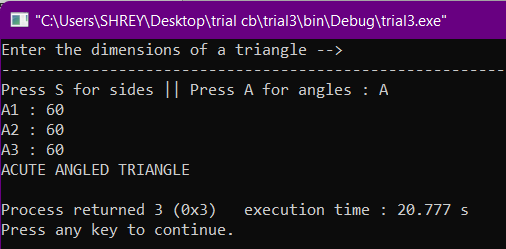
goto label;

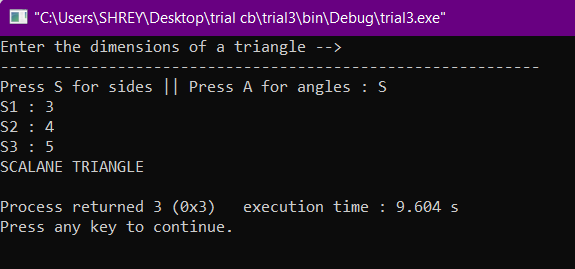
}

return 3;

}







## Q7. Modify q6 to initialize using vertex coordinates instead of sides/angles.

#include <stdio.h>

#include <ctype.h>

#include <math.h>

#include <stdlib.h>

int i,j;

struct vertex

{

float x1,x2,x3,y1,y2,y3;

}v;

typedef struct vertex ver;

float distance(float a,float b,float m,float n)

{

float dis = sqrt(((float)pow((a-b),2)+(float)pow((m-n),2)));

return dis;

}

//TYPES OF TRIANGLE ON BASIS OF SIDES

void triangle(float a,float b,float c)

{

if(a==b||b==c||c==a){

if(a==b && b==c) printf("\n||EQUILATERAL TRIANGLE||\n");

else printf("\n||ISOSCELES TRIANGLE||\n");

}

else printf("\n||SCALENE TRIANGLE||\n");

}

//TYPES OF TRIANGLE ON BASIS OF ANGLES

void angle(float a,float b,float c)

{

if((a<90)&&(b<90)&&(c<90)) printf("||ACUTE ANGLED TRIANGLE||\n");

if((a==90.000000)||(b==90.000000)||(c==90.000000)){

printf("||RIGHT ANGLED TRIANGLE||\n");

}

else if((a>90)||(b>90)||(c>90)) printf("||OBTUSE ANGLED TRIANGLE||\n");

}

int main()

{

printf("Finding the type of triangle ---\n");

for(i=0;i<30;++i){

printf("--");

}

printf("\n");

printf("Enter the value of the vertex of the triangle -->\n");

label:

printf("X1 : ");

scanf("%f",&v.x1);

printf("X2 : ");

scanf("%f",&v.x2);

printf("X3 : ");

scanf("%f",&v.x3);

for(i=0;i<30;++i){

printf("--");

}

printf("\n");

printf("Y1 : ");

scanf("%f",&v.y1);

printf("Y2 : ");

scanf("%f",&v.y2);

printf("Y3 : ");

scanf("%f",&v.y3);

// triangle(v.x1,v.x2,v.x3,v.y1,v.y2,v.y3);

float s1 = distance(v.x1,v.x2,v.y1,v.y2);

float s2 = distance(v.x2,v.x3,v.y2,v.y3);

float s3 = distance(v.x3,v.x1,v.y3,v.y1);

float max = s1;

if(s2>max) max=s2;

if(s3>max) max=s3;

//INVALID IF ANY SIDE LENGTH IS 0

if(s1==0 || s2==0 || s3==0){

printf("\n\*INVALID INPUT\*\n");

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

goto label;

}

//INVALID IF AREA OF THE TRIANGLE IS 0 OR POINTS ARE COLLINEAR

float area = (1/2.0)\*(abs((v.x1\*(v.y2-v.y3)+v.x2\*(v.y1-v.y3)+v.x3\*(v.y1-v.y2))));

if(area==0){

printf("\n\*INVALID INPUT\*\n");

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

goto label;

}

//CHECKING IF SUM OF TWO SMALLER SIDES OF THE TRIANGLE IS LESS THAN THE LARGEST SIDE OR NOT

if(max==s1){

if((s2+s3)<max){

printf("\n\*INVALID INPUT\*\n");

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

goto label;

}

}

else if(max==s2){

if((s1+s3)<max){

printf("\n\*INVALID INPUT\*\n");

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

goto label;

}

}

else if(max==s3){

if((s1+s2)<max){

printf("\n\*INVALID INPUT\*\n");

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

goto label;

}

}

//CALCULATION ANGLE USING COSINE LAW

float A = acos(((s2\*s2)+(s3\*s3)-(s1\*s1))/(2\*s2\*s3))\*(180/3.14159265);

float B = acos(((s1\*s1)+(s3\*s3)-(s2\*s2))/(2\*s1\*s3))\*(180/3.14159265);

float C = acos(((s1\*s1)+(s2\*s2)-(s3\*s3))/(2\*s2\*s1))\*(180/3.14159265);

//PRINTING VALUES OF ANGLES AND SIDES FOR VERIFICATION

printf("\n%f\n",A);

printf("%f\n",B);

printf("%f\n",C);

printf("\n%f\n",s1);

printf("%f\n",s2);

printf("%f\n",s3);

triangle(s1,s2,s3);

angle(A,B,C);

return 3;

}

