

Chapter 04

Exercises

1. The height of an object dropped from a starting height of 100 meters is given by the equation $h = 100 - 4.9 * t^2$ where t is the time in seconds. Write a C program that asks the user for a time less than 20 seconds and displays the height of the object at that time using this formula. The program output should look similar to:

Enter the time:2

The object is at 80.4 meters

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

main()
{
    float h,t;

    printf("\nEnter the time: ");

    scanf("%f",&t);

    h = 100-4.9*pow(t,2);

    printf("\nThe object is at %f meters",h);

    getch();
}
```

■ Examples

```
/* necklace*/
#include <stdio.h>
#include <stdlib.h>
main()
{
    int i,first,second,temp,f,s;
    printf("\nEnter the first number: ");
    scanf("%d",&first);
    f = first;
    printf("\nEnter the second number: ");
    scanf("%d",&second);
    s = second;
    printf("\n%d %d ",first,second);
    for(i=1;i<=100;i++)
    {
        temp = (first+second)%10;
        printf("%d ",temp);
        first = second;
        second = temp;
        if(f == first && s==second)
        {
            printf("\nYour numbers required %d steps",i);
            break;
        }
    }

    getch();
}
```

1. Write a C program to solve the equation

$$x^5 - 3x^2 + 10 = 0$$

The program must implement the trial-and-error procedure (**iterative** procedure)

```
/* solve equation numerically */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
    int i;
    float x; /* initial guess */
    printf("\nInitial guess: ");
    scanf("%f",&x);
    double xn;
    for(i=1;i<100;i++)
    {
        xn = pow(10-3*pow(x,2),0.2);
        printf("\nIteration number: %d\t x = %f",i,xn);
        if(fabs(xn-x)<0.001)
            break;
        x = xn;
    }

    getch();
}
```

2. Write a C program to solve the equation

$$x^5 - 3x^2 + 10 = 0$$

The program must implement the trial-and-error procedure (**iterative** procedure)

```
/* solve equation numerically using while*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main()
{
    int i=1;
    float x; /* initial guess */
    double xn = 100,e=5;
    printf("\nInitial guess: ");
    scanf("%f",&x);
    while(fabs(e)> 0.001)
    {
        xn = pow(10-3*pow(x,2),0.2);
        printf("\nIteration number: %d\t x = %f",i,xn);
        i++;
        e = xn - x;
        x = xn;
    }

    getch();
}
```

Trace

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int i,j,x=0;
    for(i = 0;i<5;++i)
        for(j=0;j<i;++j)
        {
            x += (i+j-1);
            printf("%d",x);
            break;
        }
    printf("\nx=%d",x);
}
```

Solution

0136

X = 6

The velocity of a moving fluid can be found from the difference between the total and static pressures P_t and P_s . For water this is given by $V = 1.016\sqrt{P_t - P_s}$. Write a C program that will receive as input the total and static pressures and will print the velocity of the water

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{
    float Pt,Ps, V;
    printf("\nTotal Pressure: ");
    scanf("%f",&Pt);
    printf("\nStatic Pressure: ");
    scanf("%f",&Ps);
    V = 1.016*sqrt(Pt-Ps);
    printf("\nVelocity = %f",V);
    getch();
}
```

3. Projectile trajectories using equations for ideal projectile motion:

$$y(t) = y_0 - \frac{1}{2} g t^2 + (v_0 \sin(\theta_0)) t$$

$$x(t) = x_0 + (v_0 \cos(\theta_0)) t$$

Where $y(t)$ is the vertical distance and $x(t)$ is the horizontal distance traveled by the projectile in meters, g is the acceleration due to Earth's gravity = **9.8 m/s²** and t is time in seconds. Let us assume that the initial velocity of the projectile $v_0 =$ **50.75 m/s** and the projectile's launching angle $\theta_0 = \frac{5\pi}{12}$ radians. The initial vertical and horizontal positions of the projectile are given by $y_0 =$ **0 m** and $x_0 =$ **0 m**. Print the value of y and x at $t = 0$ to 20 with step equal to 1

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.14159
main()
{
    float x0,y0,g,v0,theta0;
    float x,y;
    int i;
    x0 = 0;
    y0 = 0;
    v0 = 50.75;
    theta0 = 5*PI/12;
    for(i = 0;i<=20;i++)
    {
        y = y0 -(1/2)*g*pow(i,2)+v0*sin(theta0)*i;
        x = x0 + v0*cos(theta0)*i;
        printf("\n %d\t%f\t%f",i,x,y);
    }
}
```

Guess a number

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    float guess = 50,incr;
    int flag = 0;
    char ch ;
    incr = guess;
    printf("guess a number between 1 and 99,and\n");
    printf("I'll guess what it is. type 'e' for equals, \n");
    printf("'g' for greater than, and 'l' for less than. \n");
    while(flag != 1)
    {
        incr = incr/2;
        printf("\nIs your number greater than or equal to %f\n",guess);
        if ((ch = getche()) == 'e')
            flag = 1;
        else if(ch == 'g')
            guess += incr/2;
        else
            guess -= incr/2;

    }
    printf("\nThe number is %f",guess);
    getche();
}
```

4. Calculate the geometric average of a list of numbers, using the formula

$$x_{average} = [x_1 x_2 x_3 \dots x_n]^{1/n}$$

Test your program with the following six data items 6.2, 12.3, 5.0, 18.8, 7.1, 12.8
Compare the results obtained with the arithmetic average of the same data. Which average is larger

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{
    int i,n;
    float x,xg,xa;
    printf("\n#numbers: ");
    scanf("%d",&n);
    xa = 0;
    xg = 1;
    for(i=1;i<=n;i++)
    {
        printf("\nNumber %d: ",i);
        scanf("%f",&x);
        xa += x;
        xg *= x;
    }
    xa = xa/n;
    xg = pow(xg,1/(float)n);
    printf("\nArithmetic average = %f",xa);
    printf("\nGeometric average = %f",xg);
    if(xg>xa)
        printf("\nGeometric average is larger");
    else
        printf("\nArithmetic average is larger");
}
```

5. "chaos theory" is a subfield of mathematics which relies heavily on the computer. A simple chaos experiment is:

Take any real number X between 0 and 1 as a starting value. Generate a new number using the "logistic" equation:

$$X = 2 * X * (1 - X)$$

Write a C program to display this new X , and repeat the process. Perform this cycle 50 times

```
#include <stdio.h>
#include <stdlib.h>

main()
{
    int i;
    float x;
    printf("\nChoose any real x value...");
    printf("\n x should be between 0 and 1: ");
    scanf("%f",&x);
    for(i=1;i<=50;i++)
    {
        x = 2*x*(1-x);
    }
    printf("\n End value: %f",x);
    getch();
}
```

Solve n quadratic equation

Write a **C program** that will calculate and display the **real roots** of n quadratic equation

$ax^2 + bx + c = 0$ using the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Assume that a , b and c are floating-point arguments whose values are given, and that x_1 and x_2 are floating-point variables.

// solve n equation of quadratic form

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

main()
{
    float a,b,c,D,xr,x1,x2;

    int i,n;

    printf("Type #equations: ");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
    {
        printf("\nType a: ");
        scanf("%f",&a);

        printf("\nType b: ");
        scanf("%f",&b);

        printf("\nType c: ");
        scanf("%f",&c);

        D = pow(b,2)-4*a*c;

        if (D >= 0)
        {
            x1 = (-b+sqrt(D))/(2*a);
            x2 = (-b-sqrt(D))/(2*a);

            printf("\nx1 = %f\tX2 = %f",x1,x2);
```

```
    }  
    else  
    {  
        D = sqrt(-D)/(2*a);  
        xr = -b/(2*a);  
        printf("\nx1 = %f+i%f\tx2 = %f-i%f",xr,D,xr,D);  
    }  
}  
}
```

Type #equations: 3

Type a: 1

Type b: 5

Type c: 6

x1 = -2.000000 X2 = -3.000000

Type a: 1

Type b: 4

Type c: 4

x1 = -2.000000 X2 = -2.000000

Type a: 1

Type b: 4

Type c: 16

x1 = -2.000000+i3.464102 x2 = -2.000000-i3.464102

Solve a number of quadratic equation as long as the user typed any character except *n*

Write a **C program** that will calculate and display the **roots** of quadratic equation

$ax^2 + bx + c = 0$ using the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Assume that *a*, *b* and *c* are floating-point arguments whose values are given, and that *x*₁ and *x*₂ are floating-point variables.

```
// solve n equation of quadratic form
//type n for stop solving the problem
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{
    float a ,b,c,D,xr,x1,x2;
    char ch = 'y';
    while(ch != 'n')
    {
        printf("\nType a: ");
        scanf("%f",&a);
        printf("\nType b: ");
        scanf("%f",&b);
        printf("\nType c: ");
        scanf("%f",&c);
        D = pow(b,2)-4*a*c;
        if (D >= 0)
        {
            x1 = (-b+sqrt(D))/(2*a);
            x2 = (-b-sqrt(D))/(2*a);
            printf("\nx1 = %f\tX2 = %f",x1,x2);
        }
    }
}
```

```
    }  
    else  
    {  
        D = sqrt(-D)/(2*a);  
        xr = -b/(2*a);  
        printf("\nx1 = %f+i%f\tx2 = %f-i%f",xr,D,xr,D);  
  
    }  
    printf("\n Y/N: ");  
    ch = getche();  
  
}  
}
```

Write a program that inputs one **five-digit number**, separates the number into its individual digits and prints the digits separated from one another by three spaces each. [Hint: Use combinations of integer division and the remainder operation.] For example, if the user types in 42139, the program should print

4 2 1 3 9

//five digit number to its individual

#include <stdio.h>

#include <stdlib.h>

main()

{

int x,a,f = 10000;

int i;

printf("\nType a number: ");

scanf("%d",&x);

printf("\n");

for(i = 1;i<=5;i++)

{

a = x/f;

x = x%f;

f /=10;

printf("%d\t",a);

}

}

Check whether it was right triangle — if this is the case print out Right Triangle and exit the program

Write a program that reads a number of three nonzero integers and determines and prints whether they could be the sides of a right triangle **قائم الزاوية**.

if this is the case, print out it is Right triangle and stops the program

```
// search for right triangle - and exit
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <math.h>
```

```
main()
```

```
{
```

```
    float x,y,z,x2,y2,z2;
```

```
    int i,n;
```

```
    printf("# Triangle: ");
```

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

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

```
    {
```

```
        printf("\nSide 1: ");
```

```
        scanf("%f",&x);
```

```
        printf("\nSide 2: ");
```

```
        scanf("%f",&y);
```

```
        printf("\nSide 3: ");
```

```
        scanf("%f",&z);
```

```
        x2 = pow(x,2);
```

```
        y2 = pow(y,2);
```

```
        z2 = pow(z,2);
```

```
        if((x2+y2) == z2)
```

```
        {
```

```
            printf("\nRight triangle");
```

```
            break;
```

```
        }
```

```
    else if((x2+z2)==y2)
```

```
    {
```

```
        printf("\nRight triangle");
```

```
        break;
```

```
    }
```

```
    else if((y2+z2)==x2)
```

```
    {
```

```
        printf("\nRight triangle");
```

```
        break;
```

```
    }
```

```
    else
```

```
        printf("\nNot Right Triangle");
```

```
    }  
}  
  
# Triangle: 5  
Side 1: 1  
Side 2: 1  
Side 3: 1  
Not Right Triangle  
Side 1: 2  
Side 2: 2  
Side 3: 2  
Not Right Triangle  
Side 1: 12  
Side 2: 5  
Side 3: 13  
Right triangle  
Process returned 15 (0xF)   execution time : 50.592 s  
Press any key to continue.
```