Exercise 2: Expressions, Variables & Assignment

Nanda H Krishna

8 February 2018

1 Area and Perimeter of a Circle

Problem description: Write a program to calculate the area and the perimeter of a circle. Read the radius from the user and print the outputs on the display.

Specification: The function `area()` takes radius of circle as input and returns the area of the circle while the function `perimeter()` takes radius as input and returns the circle's perimeter.

Prototype:

```
float area(float r)
float perimeter(float r)
```

Program design: The program consists of 3 functions, `area(float r)` to calculate area of the circle, `perimeter(float r)` to calculate the circle's perimeter and `main()` to get input and call the functions.

Algorithm: The algorithm to calculate area and perimeter of a circle is as follows:

```
area(r):
  return 3.14 * (r^2)
perimeter(r):
  return 3.14 * 2 * r
```

Program:

```
#include<stdio.h>
float area(float r) //To calculate area of circle given radius
{
  return 3.14*r*r;
}
float perimeter(float r) //To calculate perimeter of circle given radius
{
  return 2*3.14*r;
}
int main()
{
  float r;
  scanf("%f", &r); //Enter radius
  printf("%f \n %f \n", area(r), perimeter(r));
  return 0;
}
```

Test Input:

```
10
```

Output:

```
314.000000
62.799999
```

2 Leap Year

Problem description: Write a Boolean function `is_leap()` for testing whether a year is leap year or not. Test the function from `main()`.

Specification: The boolean function `is_leap()` takes the year as the parameter and returns `true` if year is a leap year or `false` if it is not.

Prototype:

```
bool is_leap(int year)
```

Program design: The program consists of 2 functions, `is_leap(int year)` to check if a year is a leap year or not, and `main()` to get input and call `is_leap()` for testing.

Algorithm: The algorithm to check for leap years is as follows:

```
is_leap(year):
  if year % 400 is 0:
    return true
  else if year % 4 is 0 and year % 100 is not 0:
    return true
  else:
    return false
```

Program:

```
#include<stdio.h>
#include<stdbool.h>
bool is_leap(int year) // Takes year as parameter
{
  if(year % 400 == 0) // Leap year rule
    return true;
  else if(year % 4 == 0 && year % 100 != 0)
    return true;
  else return false;
}
int main()
{
  int year;
  scanf("%d",&year);
  bool t = is_leap(year);
  if(t == true)
    printf("Leap year");
  else printf("Non Leap year");
  return 0;
}
```

Test Input:

```
2000
```

Output:

```
Leap year
```

3 Quadratic Equation

Problem description: Read the coefficients `a`, `b`, and `c` of a quadratic equation. Calculate the discriminant. Define a function `sign()` that returns `-1` or `0` or `1` for a negative number, zero or a positive number, respectively. Use it to test the discriminant. If the discriminant is non-negative, find the roots of the equation, and print them.

Specification: The function `sign()` takes the discriminant as the input and based on its sign, returns `-1`, `0` or `1` as the output. The calculation of roots is done in main. The inputs for `a`, `b` and `c` are received, followed by `D` calculation, calling of `sign()` and then root calculation.

Prototype:

```
int sign(float d)
```

Program design: The program consists of `sign(float d)` which takes input as discriminant and returns `-1`, `0` or `1` depending on the sign, and `main()` which takes inputs for `a`, `b` and `c` and also calls `sign()` and calculates the roots.

Algorithm: The algorithm to get discriminant sign and roots of a quadratic equation are:

```
sign(d):
  if d < 0:
    return -1
  else if d is 0:
    return 0
  else:
    return 1
if sign returns -1:
  print Complex roots
else:
  if sign returns 0:
    print Real and equal roots, -b/2a
  else:
    print Real and distinct roots, (-b+D^0.5)/2a, (-b-D^0.5)/2a
```

Program:

```
#include<stdio.h>
#include<math.h>
int sign(float d) // Discriminant parameter
{
  if(d < 0)
    return -1;
  else if(d == 0)
    return 0;
  else
    return 1;
}
int main()
{
  float a, b, c, d;
  int s;
  scanf("%f %f %f", &a, &b, &c);
  d = b*b - 4*a*c;
  s = sign(d);
  if(s == -1)
    printf("Complex roots");
  else {
      if(s == 0) {
  printf("Real and equal roots, %f", -b/(2*a));
      }
      else {
  float x = -b/(2*a), y = sqrt(d)/(2*a);
  printf("Real distinct roots, %f and %f", x+y, x-y);
      }
  }
  return 0;
}
```

Test Input:

```
1 2 1
```

Output:

```
Real and equal roots, -1.000000
```

4 Distance between 2 points

Problem description: Write a program to compute the distance between two points. To read a point, the program should read 2 numbers from the user for the x and y coordinates. Print the output on the stdout. Implement a function `distance(x1, y1, x2, y2)` that takes two points `(x1, y1)` and `(x2, y2)` as 4 parameters and returns the distance between the two points.

Specification: The function `distance(x1, y1, x2, y2)` takes the x and y coordinates of the 2 points and returns the distance between them.

Prototype:

```
float distance(int x1, int y1, int x2, int y2)
```

Program design: The program consists of `distance(int x1, int y1, int x2, int y2)` which calculates the distance between the 2 points, and `main()` to get input and for testing.

Algorithm: The algorithm to find distance between 2 points is as follows:

```
distance(x1, y1, x2, y2):
  return sqrt((x2-x1)^2 + (y2-y1)^2)
```

Program:

```
#include<stdio.h>
#include<math.h>
float distance(int x1, int y1, int x2, int y2)
{
  return sqrt(pow(x2-x1,2)+pow(y2-y1,2));
}
int main()
{
  int x1, x2, y1, y2;
  scanf("%d %d %d %d",&x1,&y1,&x2,&y2); // Coordinates
  printf("%f",distance(x1,y1,x2,y2));
  return 0;
}
```

Test Input:

```
2 1 1 2
```

Output:

```
1.414214
```

5 Swap 2 variables

Problem description: Initialize two variables with values read from the user and exchange (swap) their contents. Print them before and after the swap.

Specification: The swapping of the 2 numbers is done in `main()`, with inputs from `stdin` and the outputs being the numbers before and after the process.

Program design: The program consists of `main()` which gets the input of 2 numbers from `stdin`, swaps them and prints them on `stdout`.

Algorithm: The algorithm to swap 2 numbers is as follows:

```
temp = a
a = b
b = temp
```

Program:

```
#include<stdio.h>
int main()
{
  int a, b;
  scanf("%d %d",&a,&b);
  int t = a; //Temporary t
  a = b;
  b = t;
  printf("%d %d",a,b);
  return 0;
}
```

Test Input:

```
2 3
```

Output:

```
2 3
3 2
```

6 Circulate numbers

Problem description: Read four numbers `a`, `b`, `c`, `d` from `stdin`. Circulate them so that `a` gets the value of `b`, and so on: a <- b <- c <- d <- a

Specification: The inputs are the 4 numbers to circulate, the output is the numbers after circulation, all carried out in `main()`.

Program design: The program consists of `main()` which gets the input from `stdin`, circulates the numbers and prints them on `stdout`.

Algorithm: The algorithm to circulate 4 numbers is as follows:

```
t = a
a = b
b = c
c = d
d = t
```

Program:

```
#include<stdio.h>
int main()
{
  int a, b, c, d;
  scanf("%d %d %d %d",&a,&b,&c,&d);
  int t = a; // Temporary
  a = b;
  b = c;
  c = d;
  d = t;
  printf("%d %d %d %d",a,b,c,d);
  return 0;
}
```

Test Input:

```
2 3 4 5
```

Output:

```
3 4 5 2
```

7 Rearrange 3 numbers

Problem description: Read three numbers `a, b, c` from `stdin`. Write a program to rearrange them so that `a < b < c`.

Specification: Input of 3 numbers is received from `stdin` and the output is the numbers rearranged as per the given condition.

Program design: The program consists of `main()` which receives inputs from `stdin`, rearranges the numbers and prints output on `stdout`.

Algorithm: The algorithm to rearrange 3 numbers in ascending order is as follows:

```
if a > b:
  swap a & b
if a > c:
  swap a & c
if b > c:
  swap b & c
```

Program:

```
#include<stdio.h>
// Rearrange 3 numbers as a < b < c.
int main()
{
  int a, b, c, t;
  scanf("%d %d %d", &a, &b, &c);
  if(a > b) {
    t = a; a = b; b = t;
  }
  if(a > c) {
    t = a; a = c; c = t;
  }
  if(b > c) {
    t = b; b = c; c = t;
  }
  printf("%d %d %d", a, b, c);
  return 0;
}
```

Test Input:

```
4 2 9
```

Output:

```
2 4 9
```

8 Rearrange 3 numbers in an array

Problem description: Fill an array of 3 numbers with numbers read from `stdin`. Write a program to rearrange them so that `a[0] < a[1] < a[2]`.

Specification: Input of 3 numbers is received from `stdin` and the output is the numbers rearranged as per the given condition.

Program design: The program consists of `main()` which receives inputs from `stdin`, rearranges the numbers in the array and prints output on `stdout`.

Algorithm: The algorithm to rearrange 3 numbers in ascending order is as follows:

```
if a[0] > a[1]:
  swap a[0] & a[1]
if a[0] > a[2]:
  swap a[0] & a[2]
if a[1] > a[2]:
  swap a[1] & a[2]
```

Program:

```
#include<stdio.h>
// Rearrange 3 numbers in the array
int main()
{
  int a[3], t;
  scanf("%d %d %d", &a[0], &a[1], &a[2]);
  if(a[0] > a[1]) {
    t = a[0]; a[0] = a[1]; a[1] = t;
  }
  if(a[0] > a[2]) {
    t = a[0]; a[0] = a[2]; a[2] = t;
  }
  if(a[1] > a[2]) {
    t = a[1]; a[1] = a[2]; a[2] = t;
  }
  printf("%d %d %d", a[0], a[1], a[2]);
  return 0;
}
```

Test Input:

```
7 6 8
```

Output:

```
6 7 8
```