C Programming Introduction

week 9: Function

Functions

- a group of declarations and statements that is assigned a name
  - effectively, a named statement block
  - usually has a value
- a sub-program
  - when we write our program we always define a function named **`main`**
  - inside **`main`** we can call other functions
    - which can themselves use other functions, and so on…

Example: Square

```
double square(double a)
{
    return a * a;
}
```

This is a function defined outside main

```
int main(void)
{
    double num = 0.0, sqr = 0.0;

    printf("enter a number\n");
    scanf("%lf",&num);

    sqr = square(num);

    printf("square of %g is %g\n", num, sqr);

    return 0;
}
```

Here is where we call the function square

Why use functions?

- Break your problem down into smaller sub-tasks
  - easier to solve complex problems
- generalize a repeated set of instructions
  - we don’t have to keep writing the same thing over and over
  - printf and scanf are good examples…
- They make a program much easier to read and maintain

Characteristics of Functions

```
return-type name(argument-list)
{
   local-declarations
   statements
   return return-value;
}
```

- When invoking a function call, we can include **function parameters** in the parameter list.
- Declaring a **function parameter** is accomplished by simply including the **prototype of the function** in the parameter list

Exercise 12.1

- Write a function to calculate the kinetic energy of the element

  ke= $mv^2/2$, for m is mass (kg) and v is speed (m/s)
- Use this function in a program.

Solution

```
#include <stdio.h>
double kineticEnergy(double m, double v){
   return m*v*v/2;
}
void main(){
   double m, v;
   do {
       printf("Enter mass:"); scanf("%f",&m);
       printf("Enter speed:"); scanf("%f",&v);
   } while (m>0 && v >=0);
   printf("Kinetic Energy of element is:%f",
   kineticEnergy(m,v));
}
```

Exercise 12.2

1. Write a function `is_prime` that accepts a positive integer and returns 1 if it's a prime number, and 0 otherwise.
   prototype: `int is_prime(int n);`
2. Now write a program that gets a positive integer from the user and prints all the prime numbers from 2 up to that integer.
   Use the function from (1)!

Solution: function

```
int is_prime(int n)
{
    int i = 0;

    /* Check if any of the numbers 2, ... , n-1
   divide it. */
    for (i = 2; i < sqrt(n); ++i)
    {
        if (n % i == 0)
        {
            return 0;
        }
    }
    return 1;
    /* If we got here - n is necessarily prime */
}
```

Solution: main program

```
int main(void)
{
    int num = 0, i = 0;
    /* Get input from user */
    printf("enter a positive integer\n");
    scanf("%d", &num);

    printf("prime numbers up to %d:\n", num);
    for (i = 2; i <= num; ++i)
    {
        if (is_prime(i))
        {
            printf("%d\n",i);
        }
    }
    return 0;
}
```

Pass by value

- Function arguments are passed to the function by **copying** their **values** rather than giving the function direct access to the actual variables
- A change to the value of an argument in a function body will not change the value of variables in the calling function

Exercise 12.3

- Write programs to setup these following functions. Use them in a main program
  - A function to find the sum of the cube of integers from 1 to n
  - A function to list all submutiples of the integer n
  - A function to list the n first perfect square numbers

Solution: sum of cube and List of submultiples

```
long sumcube(int n)
{
     int i = 0;
      long s=0;
      for(i=1; i<=n; i++) s+=i*i*i; return s;
}
void printsubmultiples(int n)
{
    int i;
      for(i=2; i<n; i++)
            if (n%i ==0) printf("%d ",i);
      printf("\n");
}
```

Solution: n first perfect square

```
void printsquares(int n)
{
     int i;
     for(i=1; i<=n; i++)
          printf("%d ",i*i);
     printf("\n");
}
```

Exercise

- Write a program to calculate the worker's salary by a week. The average wage is 15000 VND for one hour working. And workers have to do 40 hours a week. If they work overtime, the money is paid more 1.5 time for each hour.
- Data validation: A worker can not work less than 10 hours or more than 65 hours a week.

Solution:Salary Function

```
#include <stdio.h>
long salary(int hours)
{
  if (hours >40)
   return 15000*40+15000(hours-40)*3/2;
   else      return hours*40;
}
int main()
{
    int n;
     do {
   printf("Enter number of working hours:");
   scanf("%d",&n);
     } while (m<10 || n>=65);
    printf("The salary you get:%ld\n",salary(n));
     return 0;
}
```

Exercise 12.5

- Write the function

  void printnchars(int ch, int n) to display a character for n time. Use this function to print "* - triangle" which has edges of 4, 5.

Solution

```
void printnchars(int ch, int n)
{
   int i;
   for(i = 0; i < n; i++)
       printf("%c", ch);
}
```

Exercise 12.6

- The formula for converting a temperature from Fahrenheit to Celcius is C = 5/9(F-32)
- Write a function named celsius that accepts a Fahrenheit temperature as an argument. Function should return the temperature in Celcius. Display a table of the Fahrenheit temperature 0 though 20 and their Celsius equivalents.

Solution

```
// function to convert fahrenheit to celsius
double celsius(double);

int main() {
    double fahr = 0;

    printf("Fahrenheit\tCelsius\n");
    while (fahr < 21) {
        printf("%6.1f\t%6.1f\n", fahr, celsius(fahr);
        fahr += 1;
    }

    return 0;
}

double celsius(double f) {
    return 5 * (f - 32) / 9;
}
```

Exercise 12.7

- Given a positive number n which is k-figure number. Write a function to verify whether n has all figures being odd numbers or even numbers.

Solution

```
#include <stdio.h>

int DigitAllSame(int n){
    int digit;
    int count =0;
    int flagEven,flagOdd;
    flagEven=1; flagOdd=1;
    while (n>0 && count<5){
        digit = n%10;
        n= n/10;
        count++;
        if (digit%2 == 0) {
                flagEven= flagEven*1;
                flagOdd= flagOdd*0;
        }
        else {
                flagEven= flagEven*0;
                flagOdd= flagOdd*1;
        }
    }
    printf("count = %d\n", count);
    if (count>=5) return -1;
    if (flagEven || flagOdd) return 1;
    else return 0;
}
```

```
main() {

        printf("Hello.\n");
        printf("So %d co gia tri
        ham la %d\n", 44668,
        DigitAllSame(46668));
}
```

Exercise

- The program Vietnamese Idol has 5 judges, each of whom awards a score between 0 and 10 for each performer. Performer's final score is determined by dropping the highest and lowest score received, the averaging th 3 remaining scores. Write a program that uses this method to calculate a contestant's score using two following functions:
  - void getJudgeData() should ask the user for a judge's score, store it in a reference parameter variable, and validate it.
  - void calcScore() should calculate and display the average score of performer.

Solution

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

// function to get Judge's score
void getJudgeData(double *);
// function to calculate competitor's score
void calcScore(double, double, double, double, double);
double findLowest(double, double, double, double, double);
double findHighest(double, double, double, double, double);

int main() {
    double s1, s2, s3, s4, s5;

    getJudgeData(&s1);
    getJudgeData(&s2);
    getJudgeData(&s3);
    getJudgeData(&s4);
    getJudgeData(&s5);
    calcScore(s1, s2, s3, s4, s5);
    return 0;
}
```

Solution

```
void getJudgeData(double *s) {
   do {
     printf("Enter a judge's score: "); scanf("%f",s);
   } while (s < 0 || s > 10);
}

double findLowest(double s1, double s2, double s3, double
   s4, double s5) {
   double min = s1;
   if (s2 < min) min = s2;
   if (s3 < min) min = s3;
   if (s4 < min) min = s4;
   if (s5 < min) min = s5;
   return min;
}
```

Solution

```
double findHighest(double s1, double s2, double s3, double s4, double
   s5) {
   double max = s1;
   if (s2 > max) max = s2;
   if (s3 > max) max = s3;
   if (s4 > max) max = s4;
   if (s5 > max) max = s5;
   return max;
}

void calcScore(double s1, double s2, double s3, double s4, double s5) {
   double sum = s1 + s2 + s3 + s4 + s5;
   double max = findHighest(s1, s2, s3, s4, s5);
   double min = findLowest(s1, s2, s3, s4, s5);
   sum -= (max + min);
   printf("Max = %1.2f\n", max);
   printf("Min = %1.2f\n", min);
   printf("Final score: %1.2f\n", sum / 3);
}
```

Exercise: Leap Year

- Write an algorithm *isLeapYear* as a function that determines whether a given year is a leap year. Pass the year as a parameter. A year is a leap year if
  - It is a multiple of 4 but not a multiple of 100 OR
  - It is a multiple of 400
  - So, for example, 1996 and 2000 are leap years, but 1900, 2002 and 2100 are not.