PDS Lab

Section 16

Autumn-2018

Tutorial 5

Functions

- The C language is termed as function-oriented programming

- Every C program consists of one or more functions.

    - The concept is based on the “divide-and conquer” policy.
    - A large program can be decomposed into a number of relatively smaller segments
    - Easy to code, debug, maintain, etc.

- In C, there is no limit on the number of such functions in a program.

- In C programs, any function can call any other function, as many time as it may be.

- One of these functions, there shall be one must be called “main”.
    - Execution of a program always begins by carrying out the instructions in “main”.

Functions in C

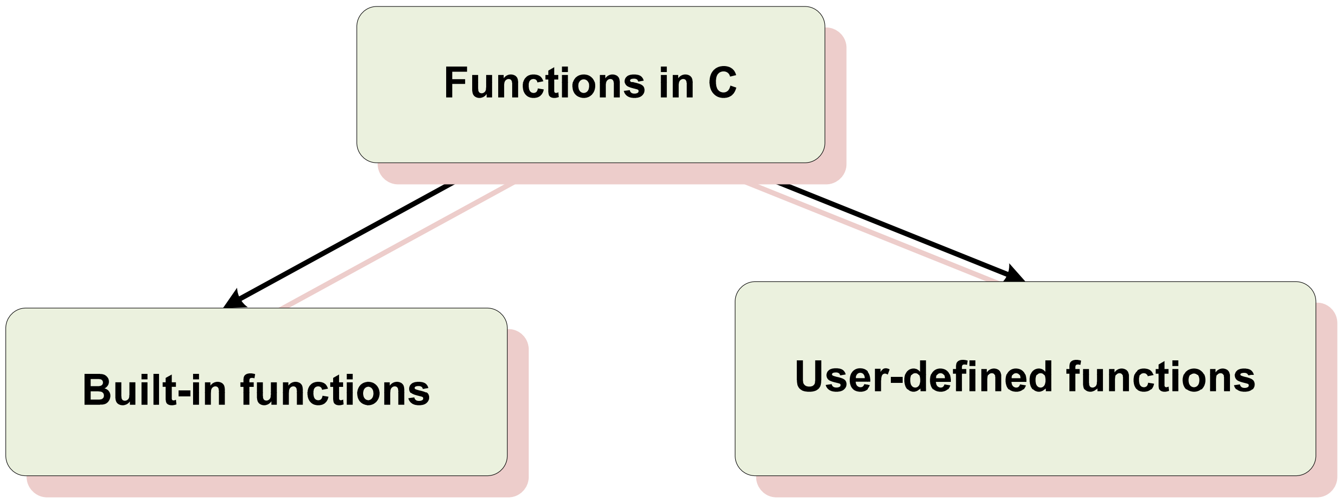
A function is a block of code that performs a set of pre-defined commands to produce desired result.

Example 1:

```
#include  <stdio.h>

int  factorial (int m)
{
    int i, temp=1;
    for (i=1; i<=m; i++)
       temp = temp * i;
    return (temp);
}
main()
{
    int  n;
    for  (n=1; n<=10; n++)
        printf (“%d! = %d \n”, n,
       factorial (n) );
}
```

Functions definition (prototype) includes basic structural information: it tells the compiler what the function will return, how the function will be called, as well as the arguments that can be can be passed.



Built-in functions

<assert.h>	<float.h>	**<math.h>**	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	**<string.h>**
<errno.h>	<locale.h>	<signal.h>	**<stdio.h>**	<time.h>

Example 2:

printf(), scanf(), rand(), getchar(), etc.

Note: A function returns a value of predefined type.

User Defined Functions

Way 1

- Usually, a function is defined before it is called.
    - main() is the last function in the program.
    - Easy for the compiler to identify function definitions in a single scan through the file.

Way 2

- However, many programmers prefer a top-down approach, where the functions follow main().
    - Must be some way to tell the compiler.
    - Function prototypes are used for this purpose.
        - Only needed if function definition comes after use.

Example 3:

```
Header section
```

```
Global declaration
```

Function declaration section

```
<type 1> f1(<arg list 1>);
<type 2> f2(<arg list 2>);


<type n> fn(<arg list n>);
```

Main body of the program

```
<type>main(<arg list>
{
    ….
    ….
    ….
    return (…);
}
```

Declaration of f1

```
<type 1> f1(<arg list 1>)
{
   …
   …
   return(…);
}
```

Example 4:

```
#include<stdio.h>
…..
<function declaration>;
…..
main( )
{
   ……..
   …….
   ……….
}

…….
…….
<function definition>
{
   ……..
   …….
   ……….
}
```

Example 5:

```
return-value-type  function-name ( parameter-list )
        {
                declarations
                        …
                 statements
                        …
                return (…);
        }
```

Example 6:

```
# include <stdio.h>
  float eval_quad_poly( float a, float b, float c);

int main()
{
  float A,B,C,X, Z;
  int numb_X_val =1;
  printf ( “Enter the coefficients of the polynomial”);
  scanf( “%f %f %f”,&A, &B,&C);
   printf ( “A= %f, B=%f, C =%f\n”, A,B,C);
   //Evaluate the quadratic polynomial for 10 values of X
    for (numb_X_val=1; numb_X_Val <=10; ++numb_X_val)
           {  printf ( “Enter value of X”);
              scanf( “%f”,&X);
             Z = eval_quad_poly(A,B,C);          //calling the user defined function
              printf( “For X = %f  Z= %f \n”, X, Z);
           }
return 0;
}


float eval_quad_poly( float a, float b, float c)
//Defining the user defined function to evaluate a quadratic polynomial a* x^2 + b*x +c
{
    float y;
    y= a*(x*x) +b*x +c;
    return (y);
}
```

Nested Functions

- A function cannot be defined within another function.
  - All function definitions must be disjoint.

- Nested function calls are allowed.
  - A calls B, B calls C, C calls D, etc.
  - The function called last will be the first to return.

- A function can also call itself, either directly or in a cycle.
  - A calls B, B calls C, C calls back A.
    - Called recursive call or recursion.

Example 7:

```
#include  <stdio.h>
int ncr (int n, int r);
int fact (int n);
main()
{
    int i, m, n, sum=0;
    scanf (“%d %d”, &m, &n);
    for (i=1; i<=m; i+=2)
      sum = sum + ncr(n, i);
    printf (“Result: %d \n”,
sum);
}
```

```
int  ncr (int n, int r)
{
  return(fact(nfac(r)/fact(n-r));
}

int  fact (int n)
{
    int  i, temp=1;
    for (i=1; i<=n; i++)
    temp *= i;
    return (temp);
}
```

Example 8:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main()
{
      int i;
      time_t t;
      unsigned seed;
    /* Initialize random number generator */
      srand((unsigned)time(&t));

    /* Print 10 random numbers between 1 and 6 */
      for(i=1;i<=10;i++)
      {
        printf("%d",1+rand()%6);
        if (i%5 == 0)printf("\n");
      }
  return 0;
}
```

Tutorial Problems

Problem 1

Determine what each of the following foomatic functions computes:

1. ---

```
unsigned int foo1 ( unsigned int n )
    {
       unsigned int t = 0;

       while (n > 0) {
          if (n % 2 == 1) ++t;
          n = n / 2;
       }
       return t;
    }
```

2. ---

```
unsigned int foo2 ( unsigned int n )
    {
       unsigned int t = 0;

       while (n > 0) {
          if (n & 1) ++t;
          n >>= 1;
       }
       return t;
   }
```

3. ---

```
double foo3 ( double a , unsigned int n )
    {
       double s, t;

       s = 0;
       t = 1;
       while (n > 0) {
          s += t;
          t *= a;
          --n;
       }
       return s;
    }
```

4. ---

```
double foo4 ( float A[] , int n )
    {
       float s, t;

       s = t = 0;
       for (i=0; i<n; ++i) {
          s += A[i];
          t += A[i] * A[i];
       }
       return (t/n)-(s/n)*(s/n);
    }
```

Problem 2

A set of number is given. Write a function to find the minimum number form the set of numbers.

```
#include <stdio.h>

int x[100];
int size;

int minimum()
{
   int i, min = 99999;
   for (i=0; i<size; i++)
       if (min > x[i])
            min = x[i];
   return (min);
}

void main()
{
      int i;
      scanf (“%d”, &size);
      for (i=0; i<size; i++)
         scanf (“%d”, &x[i]);
      printf(“\n Minimum is %d”,minimum());
}
```

Problem 3

What this main() function does?

```
#include <stdio.h>
#define PI 3.1415926
main()
{
  float r = 4.0, area;
  area = PI*r*r;
}
```

```
#define   sqr(x)   x*x

        r = sqr(a) + sqr(30);            // r = a*a + 30*30;
        r = sqr(2+5);                    // r = 2+5*2+5;

#define  sqr(x)  (x)*(x)

        r = sqr(a+b);     r = (a+b)*(a+b);
```

Problem 4

Which of the following function definitions are invalid? Why?

a) average(x, y, z);

b) sqrt(int n);

c) int rand();

d) power (int x, int n)

e) double minimum(float x; float y);

Problem 5

The following is the function prototype to retune the value of x/y.

```
double divide(float x, float y)
{
        return (x/y);
}
```

What will be the value of the following function calls?

a) divide (10, 2);
b) divide (4.5, 1.0);
c) divide (1, 0);

Problem 6

Determine the output of the following program.

```
#include <stdio.h>
int p, q;

int product(int i, int j);

void main () {
    int x = 10, y = 20;
    p = product(x, y);
     q = product(p, product(x,2));
     printf(“%d %d \n”, p, q);
  }

  int product(int a, int b){
      return(a*b);
  }
```

Problem 7

What will be the output of the following programs given that

s = “d%samanta” and s = “Debasis Samanta”?

```
printf(s);

printf(“%s”, s);
```

Important links:

http://cse.iitkgp.ac.in/~dsamanta/courses/pds/index.html