# Functions in C

Rab Nawaz Jadoon

Assistant Professor

COMSATS IIT, Abbottabad

Pakistan

Department of Computer Science

**DCS**

COMSATS Institute of Information Technology

**Introduction to Computer Programming (ICP)**

## Class Quiz 1

- Write a program using function that prints square of first ten digits???
- Do it in just 5 minutes?

# Program

```c
int square( int y ); /* function prototype */

/* function main begins program execution */
int main( void )
{
   int x; /* counter */

   /* loop 10 times and calculate and output square of x each
   for ( x = 1; x <= 10; x++ ) {
      printf( "%d  ", square( x ) ); /* function call */
   } /* end for */

   printf( "\n" );
   return 0; /* indicates successful termination */
} /* end main */

/* square function definition returns square of parameter */
int square( int y ) /* y is a copy of argument to function */
{
   return y * y; /* returns square of y as an int */
} /* end function square */
```

## Class Quiz 2

- Write a program using function that finds the maximum of three given numbers?
- Do it in just 5 minutes?

## Progam

```c
int main( void )
{
   int number1; /* first integer */
   int number2; /* second integer */
   int number3; /* third integer */

   printf( "Enter three integers: " );
   scanf( "%d%d%d", &number1, &number2, &number3 );

   /* number1, number2 and number3 are arguments
      to the maximum function call */
   printf( "Maximum is: %d\n", maximum( number1, number2, number3 ) );
   return 0; /* indicates successful termination */
} /* end main */

/* Function maximum definition */
/* x, y and z are parameters */
int maximum( int x, int y, int z )
{
   int max = x; /* assume x is largest */

   if ( y > max ) { /* if y is larger than max, assign y to max */
      max = y;
   } /* end if */

   if ( z > max ) { /* if z is larger than max, assign z to max */
      max = z;
   } /* end if */

   return max; /* max is largest value */
} /* end function maximum */
```

## Math Library function

| Function | Description | Example |
|---|---|---|
| `sqrt( x )` | square root of *x* | `sqrt( 900.0 )` is `30.0`<br>`sqrt( 9.0 )` is `3.0` |
| `exp( x )` | exponential function $e^x$ | `exp( 1.0 )` is `2.718282`<br>`exp( 2.0 )` is `7.389056` |
| `log( x )` | natural logarithm of *x* (base *e*) | `log( 2.718282 )` is `1.0`<br>`log( 7.389056 )` is `2.0` |
| `log10( x )` | logarithm of *x* (base 10) | `log10( 1.0 )` is `0.0`<br>`log10( 10.0 )` is `1.0`<br>`log10( 100.0 )` is `2.0` |
| `fabs( x )` | absolute value of *x* | `fabs( 13.5 )` is `13.5`<br>`fabs( 0.0 )` is `0.0`<br>`fabs( -13.5 )` is `13.5` |

## Math Library function

| | | |
|---|---|---|
| `ceil( x )` | rounds *x* to the smallest integer not less than *x* | `ceil( 9.2 )` is 10.0<br>`ceil( -9.8 )` is -9.0 |
| `floor( x )` | rounds *x* to the largest integer not greater than *x* | `floor( 9.2 )` is 9.0<br>`floor( -9.8 )` is -10.0 |
| `pow( x, y )` | *x* raised to power *y* ($x^y$) | `pow( 2, 7 )` is 128.0<br>`pow( 9, .5 )` is 3.0 |
| `fmod( x, y )` | remainder of *x/y* as a floating-point number | `fmod( 13.657, 2.333 )` is 1.992 |
| `sin( x )` | trigonometric sine of *x* (*x* in radians) | `sin( 0.0 )` is 0.0 |
| `cos( x )` | trigonometric cosine of *x* (*x* in radians) | `cos( 0.0 )` is 1.0 |
| `tan( x )` | trigonometric tangent of *x* (*x* in radians) | `tan( 0.0 )` is 0.0 |

## Random number generation

- The element of chance can be introduced into computer applications by using the C Standard Library function rand from the <stdlib.h> header.
- Consider the following statement:
  - i=rand()
    - The rand function generates an integer between 0 and RAND_MAX(a symbolic constant)
    - Standard C states that the value of RAND_MAX must be at least 32767 (i.e., 16-bit) integer.

## Sixe sided dice roller

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

/* function main begins program execution */
int main( void )
{
   int i; /* counter */
   /* loop 20 times */
   for ( i = 1; i <= 20; i++ ) {

      /* pick random number from 1 to 6 and output it */
      printf( "%10d", 1 + ( rand() % 6 ) );

      /* if counter is divisible by 5, begin new line of output */
      if ( i % 5 == 0 ) {
         printf( "\n" );
      } /* end if */
   } /* end for */

   return 0; /* indicates successful termination */
} /* end main */
```

## Rolling a six sided die 6000 times

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

/* function main begins program execution */
int main( void )
{
   int frequency1 = 0; /* rolled 1 counter */
   int frequency2 = 0; /* rolled 2 counter */
   int frequency3 = 0; /* rolled 3 counter */
   int frequency4 = 0; /* rolled 4 counter */
   int frequency5 = 0; /* rolled 5 counter */
   int frequency6 = 0; /* rolled 6 counter */

   int roll; /* roll counter, value 1 to 6000 */
```

## Cont…

```
int face; /* represents one roll of the die, value 1 to 6 */

/* loop 6000 times and summarize results */
for ( roll = 1; roll <= 6000; roll++ ) {
   face = 1 + rand() % 6; /* random number from 1 to 6 */

   /* determine face value and increment appropriate counter */
   switch ( face ) {

      case 1: /* rolled 1 */
         ++frequency1;
         break;

      case 2: /* rolled 2 */
         ++frequency2;
         break;

      case 3: /* rolled 3 */
         ++frequency3;
         break;
```

## Cont…

```
            case 4: /* rolled 4 */
               ++frequency4;
               break;

            case 5: /* rolled 5 */
               ++frequency5;
               break;

            case 6: /* rolled 6 */
               ++frequency6;
               break; /* optional */
         } /* end switch */
      } /* end for */

      /* display results in tabular format */
      printf( "%s%13s\n", "Face", "Frequency" );
      printf( "   1%13d\n", frequency1 );
      printf( "   2%13d\n", frequency2 );
      printf( "   3%13d\n", frequency3 );
      printf( "   4%13d\n", frequency4 );
      printf( "   5%13d\n", frequency5 );
      printf( "   6%13d\n", frequency6 );
      return 0; /* indicates successful termination */
   } /* end main */
```

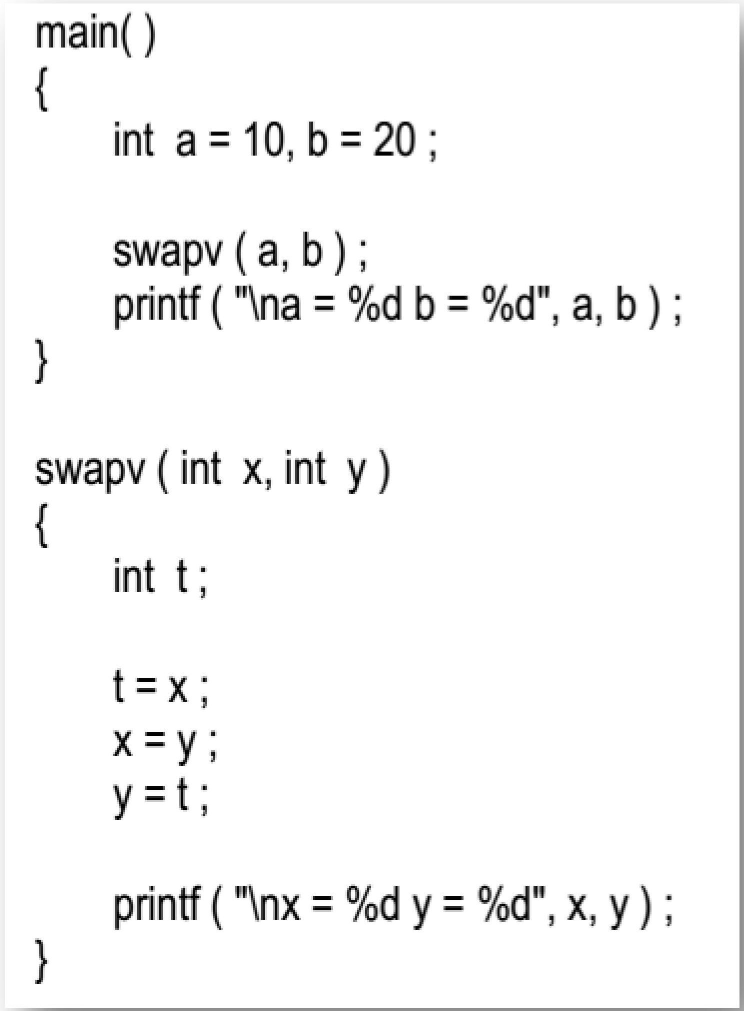
## Output

```
1          987
2          984
3          1029
4          974
5          1004
6          1022
```

## Functions call by value

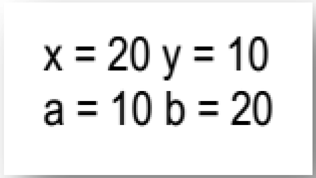
- There are two ways to invoke functions in many programming languages
  - Call-by-value and
  - Call-by-reference.
    - When arguments are passed by value, a copy of the argument's value is made and passed to the called function.
    - Changes to the copy do not affect an original variable's value in the caller.
    - Call-by-value should be used whenever the called function does not need to modify the value of the caller's original variable.
    - In C, all calls are by value.

## Program



```
main( )
{
        int  a = 10, b = 20 ;

        swapv ( a, b ) ;
        printf ( "\na = %d b = %d", a, b ) ;
}

swapv ( int  x, int  y )
{
        int  t ;

        t = x ;
        x = y ;
        y = t ;

        printf ( "\nx = %d y = %d", x, y ) ;
}
```

output



```
x = 20 y = 10
a = 10 b = 20
```

## Call by Reference

- When an argument is passed by reference, the caller allows the called function to modify the original variable’s value.
  - Call-by-reference should be used only with trusted called functions that need to modify the original variable.
  - Call-by reference is used by using address operators and indirection operators.
  - For example arrays are passed automatically by reference (see later)

THANKS