

COSC1101 – Programming Fundamentals

Maham Khan

Lecture – 15

Demonstration of Examples used in previous lectures

```
void main( )  
{  
message( ) ;  
printf ( "\nI wrote my first function in C!" ) ;  
}  
message( )  
{  
printf ( "\nThis is my first function in C.." ) ;  
}
```

Simple Example

```
void main()
{
    printline();
    printf("I love my parents !!!");
    printline();
}

void printline()
{
    int i;
    for(i=0;i<40;i++)
        printf ("-");
    printf ("\n");
}
```

Sample Function

```
int add2nums( int firstnum, int secondnum )
{
    int sum;

    sum = firstnum + secondnum;

    // just to make a point
    firstnum = 0;
    secondnum = 0;

    return(sum) ;
}
```

Testing add2nums

```
int main(void)
{
    int y,a,b;

    cout << "Enter 2 numbers\n";
    cin >> a >> b;

    y = add2nums(a,b) ;

    cout << "a is " << a << endl;
    cout << "b is " << b << endl;
    cout << "y is " << y << endl;
    return(0) ;
}
```

```
// Creating and using a programmer-defined function.
```

```
#include <iostream.h>
```

```
int square( int );    // function prototype
```

Function prototype: specifies data types of arguments and return values. **square** expects an **int**, and returns an **int**.

```
int main()
```

```
{
```

```
    // loop 10 times and calculate and output
```

```
    // square of x each time
```

```
    for ( int x = 1; x <= 10; x++ )
```

```
        cout << square( x ) << " ";    // function call
```

```
    cout << endl;
```

```
    return 0;    // indicates successful termination
```

Parentheses () cause function to be called. When done, it returns the result.

```
// square function definition returns square of an integer
```

```
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
```

Definition of **square**. **y** is a copy of the argument passed. Returns **y * y**, or **y** squared.

```
1  4  9 16 25 36 49 64 81 100
```

compute square and cube of numbers [1..10] using functions

```
#include<iostream.h>

int square(int); // prototype
int cube(int);   // prototype
main()
{   int i;
    for (int i=1;i<=10;i++){

        cout<< i<< "square=" << square(i) << endl;
        cout<< i<< "cube="    <<cube(i) << endl;
    } // end for
    return 0;
} // end main function
int square(int y) //function definition
{
    return  y*y; // returned Result
}

int cube(int y) //function definition
{
    return  y*y*y; // returned Result
}
```

Output
1 square=1
1 cube=1
2 square=4
2 cube=8
.
.
.
.
10 square=100
10 cube=1000


```
// Finding the maximum of three floating-point (real) numbers.
```

```
#include <iostream.h>
```

```
double maximum( double, double, double ); // function prototype
```

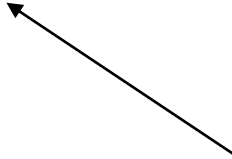
```
int main()
```

```
{  
    double number1, number2;  
    double number3;  
  
    cout << "Enter three real numbers: ";  
    cin >> number1 >> number2 >> number3;  
  
    // number1, number2 and number3 are arguments to the maximum function call  
    cout << "Maximum is: "  
        << maximum( number1, number2, number3 ) << endl;  
    return 0; // indicates successful termination  
  
} // end main
```

```
// function maximum definition. x, y and z are parameters
```

```
double maximum( double x, double y, double z )
```

```
{  
    double max = x; // assume x is largest  
    if ( y > max ) // if y is larger,  
        max = y; // assign y to max  
    if ( z > max ) // if z is larger,  
        max = z; // assign z to max  
    return max; // max is largest value  
} // end function maximum
```



Function **maximum** takes 3 arguments (all **double**) and returns a **double**.

```
Enter three real numbers: 99.32 37.3 27.1928  
Maximum is: 99.32
```

```
Enter three real numbers: 1.1 3.333 2.22  
Maximum is: 3.333
```

Finding Errors in Function Code

```
void f(float a);  
{  
    float a;  
    cout<<a<<endl;  
}
```

→; found after function definition header.

→ redefining the parameter **a** in the function

```
void f(float a)  
{  
    float a2 = a + 8.9;  
    cout <<a2<<endl;  
}
```

Random Number Generator

- **rand** function generates an integer between 0 and RAND-MAX(~32767) a symbolic constant defined in `<stdlib.h>`
- You may use modulus operator (%) to generate numbers within a specifically range with **rand**.

```
//generate 10 random numbers open-range
```

```
int x;  
for( int i=0; i<=10; i++){  
    x=rand();  
    cout<<x<<" ";  
}
```

```
//generate 10 integers between 0.....49
```

```
int x;  
for( int i=0; i<10; i++){  
    x=rand()%50;  
    cout<<x<<" ";  
}
```

Randomizing with `srand`

```
#include<iostream.h>
#include<stdlib.h>
int main()
{
    int i;
    unsigned num;
    // we will enter a different number each time we run
    cin>>num;
    srand(num);
    for(i=1; i<=5; i++)
        cout<<setw(10)<< 1+rand()%6;
    return 0;
}
```

Output for Multiple Runs

19→	6	1	1	4	2	1
18→	6	1	5	1	4	4
3→	1	2	5	6	2	4
0→	1	5	5	3	5	5
3→	1	2	5	6	3	4

Different-set of Random
numbers



Example

```
void abc (int&);  
void main ( )  
{  
    int temp, x;  
    abc(temp);  
    abc(x);  
    cout<<temp<<endl<<x;  
}  
void abc(int &t)  
{  
    cout<<"enter number=";  
    cin>>t;  
}
```

Default arguments example

```
#include <iostream.h>
void starline(char = ' # ', int = 45 ); //function declaration
void main ( )
{
    starline ( ); //function Call
    cout<<"Hello World";
    starline ( '@' ); //function call
    starline('$',30);
    getch( );
}
// starline function definition
void starline (char ch, int n ) //function Declarator
{
    for ( int j=0; j < n; j++ ) //function body
        cout<<ch;
    cout<<endl;
}
```

Finding Factorial Recursively

```
//Recursive factorial Function
#include<iostream.h>
unsigned long factorial(unsigned long); //prototype
int main()
{
    int num;
    cout<<"enter a positive integer:";
    cin>>num;
    cout<<"factorial="<<factorial(num);
    return 0;
}
unsigned long factorial(unsigned long n)
{
    if ( n <= 1) //the base case
        return 1;
    else
        return n * factorial (n - 1);
}
```

Example of Recursion

```
aint isprime(int, int = 2); //prototype
int main() {
    int num;
    cout<<" \n enter a positive integer:\n";
    cin>>num;
    if (isprime(num))
        cout << num << " is a prime number\n";
    else
        cout << "Not a Prime Number\n";
    system ("pause");
    return 0;
}
int isprime(int p, int i) {
    if (i==p)                                //or better if (i*i>p) return 1;
        return 1;
    if (p%i == 0)
        return 0;
    else
        return isprime (p, i+1);
}
```