PIC 10A

Lecture 11: Functions

What is a function

In mathematics a function is a recipe for constructing an output for a given input.

eg.

$f(x) = x^2+2x+7$

We put in a number for x and get out a new number.

In computer science, function often behaves in same manner:

```
double y = sqrt(x);
```

Functions in computer science

Function is a block of code that can be executed with a function call.

For example:

This is a function call!

```
double y = sqrt(x);
```

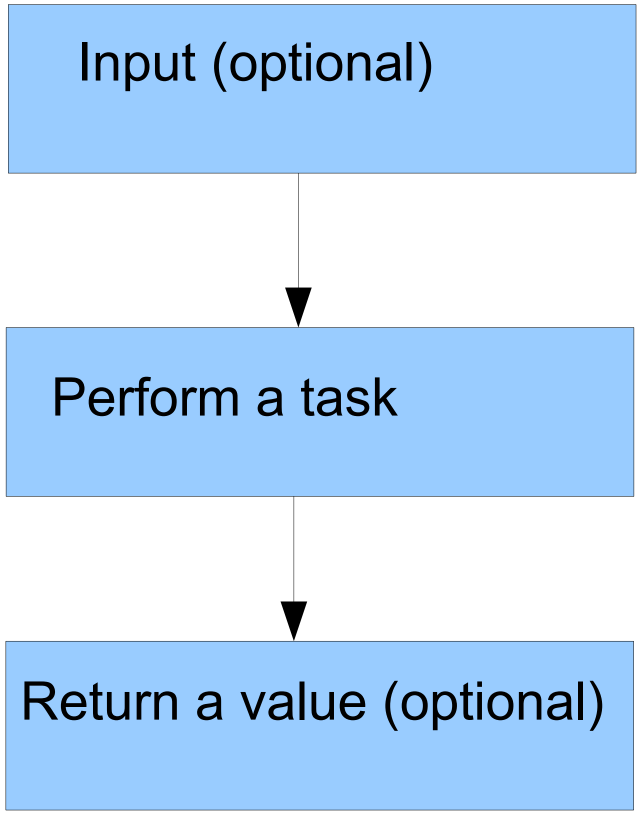
The block of code that runs is separate from the code in the main() part of your program.

In this example the code for sqrt is located in the cmath library.

Just as in math, a function can give us output. We say a function *returns a value.*

The expression sqrt(x) has the value that the function returns.

Functions in computer science



Benefits of a function

- Can execute a block of code anywhere without rewriting it.
- Makes code easy to read and debug.
- Modular code is reusable.
- Ideally a function is written like a black box.

We have met functions before!

We have not written functions before but we have called them.

Member functions are a particular kind of functions.

```
int length = s.length();
```

We have seen math functions

```
double y=sqrt(x);
```

Ok I lied

We have written one function before!

```
int main()
{

// Code

return 0;
}
```

Writing our own functions

```cpp
// function definition (implementation)
int max(int a, int b)
{
   if(a > b)
      return a;
   else
      return b;
}

int main()
{
   int x,y,z;
   cout << "Input two integers: ";
   cin >> x >> y;
   z = max(x, y); // syntax for function call
   cout << "The max is " << z;
   return 0;
}
```

There is lot to notice already in this simple example:

- When we define our function we must put data types of arguments.
- We must put the return type of the function.
- Despite having multiple inputs the function has only one output. Only one of the returns gets executed.
- Note the syntax for the function. We have braces. But as in if and while statements we don't have ; after the function name.
- When we call our function we do not state the data types!!
- Names of the variables in my function call do not have to match the names of the variables in my function.

More function facts

- Function definition is put just under using namespace (although we will soon learn another way)
- The values of the arguments in the function call are copied to variables in the function in the same order

Syntax for function definition

Return data type

```
type function_name(type par1, type par2, ...)
{
```

data type of the parameter

Parameters

```
/* code */

return value; // May be optional!

}
```

Parameters are variables that can be used in the function without declaring them!

Calling functions

When we call a function we do not put in any data types!

The parameters passed must be passed in the correct order as specified in the function definition.

The parameters must have the data types as specified in the function definition.

Syntax:

```
function_name(par1,par2,...)
```

parameters here can be expressions or variables.

How to use functions

There are many ways to use functions in your code.

Functions can return a value and so used in assignments:

```
z = pow(x,y);
```

Or in boolean expressions:

```
if (total_cost(number,price)>1000)
{
...
}
```

Or in output:

```
cout << “My total money: “ << get_total(savings,checking);
```

Where can function call be placed

When a function returns a value a call to that function can be placed anywhere where the return data type makes sense.

Examples:

```
if (total_cost(x,y) > 1000)


setw(s.length())


max( max(a,b), c )
```

Example

```
#include <iostream>
using namespace std;

double perimeter(double side1, double side2)
{
return 2*side1 + 2*side2; // oooh efficient code!
}

double area(double side1, double side2)
{
return side1 * side2;
}

int main()
{
double side1,side2;
cout << "Enter the length and the width of a rectangle: ";
cin >> side1 >> side2;
cout << "The area is: " << area(side1,side2)
<< " and the perimeter is: " << perimeter(side1,side2) << "\n";
return 0;
}
```

Example from HW2

Write a function that removes an s from the end of a word or leaves the word alone if it does not have an s at the end.

```
string dropTheS (string word)
{
    int len = word.length( );
    if ( word.substr(len-1,1) == "s" )
        return word.substr(0,len-1);
    else
        return word;
}
```

In main we might have:

```
string plural, singular;
cin >> plural;
singular = dropTheS(plural);
```

Factoring numbers

A prime number is a positive integer that is only divisible by itself and 1.

First 5 prime numbers are;

2,3,5,7,11

Common problem is to factor a number into its prime factors.

For example:

6 = 2 * 3

9 = 3 * 3

24 = 2 * 2 * 2 * 3

Finding factors of a number: Pseudo code

User enters a number N

Find the smallest number (bigger than one) that divides N. This number is prime and so is one of the factors we are after.

Output the smallest divisor.

Divide N by the smallest divisor.

Find the smallest divisor of this new N.

Keep going until smallest divisor is 1.

Factorization example

```
#include <iostream>
using namespace std;
int findSmallestDivisor (int num)
{
    int d = 2;
    while ( (num % d != 0) && d < num )
        d++;
    return d;
}

int main ( )
{
    int x;
    cout << "Enter a number: ";
    cin >> x;
    cout << "The factors of " << x << " are ";
    while ( findSmallestDivisor(x) <= x )
    {
        cout << findSmallestDivisor(x) << " ";
        x = x / findSmallestDivisor(x);
    }
    return 0;
}
```

void functions

Functions do not always have to return a value

Functions that do not return anything have a type void.

```
void output(double money)
{
   cout << "Your parking fee is $" << money;
   cout << "Please pay the cashier.";
}
```

If a function does not return anything then it can not be used in assignment.

```
error = this_function_is_void(x,y,z);
```

Compiler gives an error if you are assigning output of a void function.

Rectangle example

```
See example
```

bool functions

Boolean functions return either true or false.

```
Syntax:

bool function_name(type par1, type par2,...)
{

// Code

return boolean expression;

}
```

Boolean functions are best used when deciding if something is true or false is a complicated task.

bool function example

```
bool is_vowel(string letter);
{
    if (letter.length() > 1)
        return false;
    if (letter == "a" || letter == "e" || letter == "i"
                || letter =="o" || letter =="u")
        return true;
    else if (letter == "A" || letter == "E" || letter == "I"
                || letter =="O" || letter =="U")
        return true;
    else
        return false;

}
```

In main we might have:

```
if (is_vowel(letter))
{
// Do stuff
}
```