Functions

C++ Standard Functions

- C++ language is shipped with a lot of functions which are known as standard functions
- These standard functions are groups in different libraries which can be included in the C++ program, e.g.
  - Math functions are declared in <math.h> library
  - Character-manipulation functions are declared in <ctype.h> library
  - C++ is shipped with more than 100 standard libraries, some of them are very popular such as <iostream.h> and <stdlib.h>, others are very specific to certain hardware platform, e.g. <limits.h> and <largeInt.h>

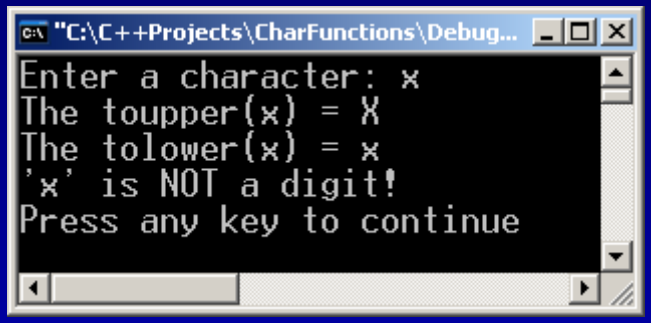
Example of Using Standard C++ Math Functions

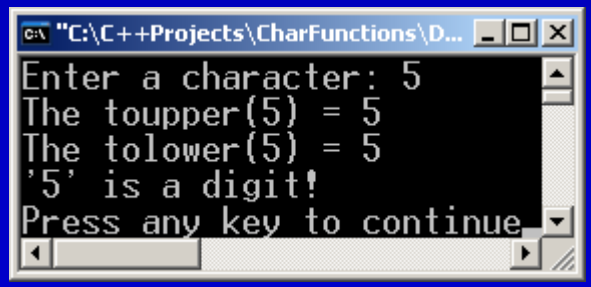
```
#include <iostream.h>
#include <math.h>
void main()
{
    // Getting a double value
    double x;
    cout << "Please enter a real number: ";
    cin >> x;
    // Compute the power and the square root of the real number
    cout << "The power(" << x << ") = " << Math.pow(x,2) << endl;
    cout << "The root(" << x << ") = " << Math.sqrt(x) << endl;
```

Example of Using Standard C++ Character Functions

```
#include <iostream.h>  // input/output handling
#include <ctype.h>    // character type functions
void main()
{
    char ch;
    cout << "Enter a character: ";
    cin >> ch;
    cout << "The toupper(" << ch << ") = " << (char) toupper(ch) << endl;
    cout << "The tolower(" << ch << ") = " << (char) tolower(ch) << endl;
    if (isdigit(ch))
        cout << "'" << ch <<"' is a digit!\n";
    else
        cout << "'" << ch <<"' is NOT a digit!\n";
}
```

Explicit casting





C++ Header Files

- cmath - declares functions for mathematical operations
- cstdlib - usually general purpose functions
- iostream - functions for standard I/O
- cstring - functions to manipulate C-style string
- cctype - functions to classify (and transform) individual characters
- csignal - to handle signals
- clocale - internationalization support task such as date/time formatting
- cwctype - for classifying and transforming individual wide characters
- cstdio - C Standard Input and Output Library
- cwchar - to work with C wide string
- cuchar - convert between multibyte characters and UTF-16 or UTF-32
- csetjmp - bypass the normal function call and return discipline
- cfenv - access floating point environment
- ctime - functions to work with date and time

Function Definition

- Define *function header* and *function body*
- *Value-returning* functions

```
return-data-type function-name(parameter list)
{
 constant declarations
 variable declarations

 other C++ statements

 return value
}
```

Function Definition (cont.)

- *Non value-returning* functions

```
void function-name(parameter list)
{
 constant declarations
 variable declarations

 other C++ statements
}
```

Function Definition (cont.)

- The argument names in the function header are referred to as *formal parameters*.

```
int FindMax(int x, int y)
{
 int maximum;

 if(x>=y)
   maximum = x;
 else
   maximum = y;

 return maximum;
}
```

Function Prototype

- Every function should have a *function prototype*.
- The function prototype specifies the *type* of the value that the function returns (if any) and the *type*, *number*, and *order* of the function's arguments.

return-data-type function-name(argument data types);

or

void function-name(argument data types);

Function Prototype (cont.)

- The use of function prototypes permits *error checking* of data types by the compiler.
- It also ensures conversion of all arguments passed to the function to the declared argument data type when the function is called.

Calling a function

- A function is *called* by specifying its name followed by its arguments.
- Non-value returning functions:

  function-name (data passed to function);
- Value returning functions:

  results = function-name (data passed to function);

Calling a function (cont.)

```
#include <iostream.h>

int FindMax(int, int); // function prototype

int main()
{
 int firstnum, secnum, max;

 cout << "\nEnter two numbers: ";
 cin >> firstnum >> secnum;

 max=FindMax(firstnum, secnum); // the function is called here

 cout << "The maximum is " << max << endl;

 return 0;
}
```

- The argument names in the function call are referred to as *actual parameters*

Function Overloading

- C++ provides the capability of using the same function name for more than one function (*function overloading*).
- The compiler must be able to determine which function to use based on the number and data types of the parameters.

Function Overloading (cont.)

```
void cdabs(int x)
{
 if (x<0)
   x = -x;
 cout << "The abs value of the integer is " << x << endl;
}

void cdabs(float x)
{
 if (x<0)
   x = -x;
 cout << "The abs value of the float is " << x << endl;
}
```

- *Warning*: creating overloaded functions with identical parameter lists and different return types is a syntax error !!