* + theRoot = sqrt(9.0);  
    /\* returns a double containing the square root of the double argument \*/
  + thePower = pow(myBase, 3);   
    /\* returns a double containing the first double argument raised to the power of the second double argument \*/
  + cout << fabs(userValue) << endl;   
    /\* returns a double containing the absolute value (positive form) of the double argument \*/
  + roundUp = ceil(startVal);  
    /\* returns a double containing the **ceiling** (rounded-up value) of the double argument \*/
  + roundDown = floor(startVal);  
    /\* returns a double containing the **floor** (rounded-down value) of the double argument \*/

<cstdlib>:

* + absoluteInt = abs(myInt);   
    /\* returns an int containing the absolute value of the int argument \*/
  + absoluteLong = labs(myLong);   
    /\* returns a long containing the absolute value of the long argument \*/
  + rand for pseudorandom number
    - A **parameter** is a local name for an argument being used in a function
* A void function call is a standalone statement: it should not be placed within another statement
* A parameter that should not be allowed to change the value of the argument in the calling function should remain pass-by-value, not pass-by-reference
* **A function never contains another function’s definition** (the function definitions are independent, not nested)
  + A **driver** program calls the function with various arguments to verify that the results are consistent with what is expected
* To test a program before an individual function has been written, you may replace it with a **stub**
  + Same prototype and header as the real function
  + Behavior is simplified (display the parameter values to ensure they are received, return the same literal value every time)
* The assert macro (provided by the library <cassert>) prevents a program from continuing execution if a Boolean expression evaluates to false
  + Prevents execution when preconditions are not met
  + Add the line #define NDEBUG to disable these macros
  + If fail() returns true, display an error message and call the function exit (which may require #include <cstdlib>)
* Both cout and ofstream variable output can be formatted
* The method setf supports additional flags:
  + ios::fixed ensures that scientific notation is avoided
  + ios::scientific ensures that scientific notation is used
  + ios::showpoint requires the decimal on a floating-point value
  + ios::showpos displays a + before positive integer values
  + ios::right and ios::left specify alignment within fields
* Any of these formatting changes can be applied using setf and removed using unsetf
* A program can use the get method to clear out any extra input remaining in the current line
* Another way of clearing unwanted characters is to use the ignore method
* The library cctype provides a set of functions to examine and convert characters of various forms

|  |  |
| --- | --- |
| **Function** | **Returns true if char\_exp is:** |
| isupper(char\_exp) | An uppercase letter |
| islower(char\_exp) | A lowercase letter |
| isalpha(char\_exp) | A letter (regardless of case) |
| isdigit(char\_exp) | A digit (e.g., '0') |
| isspace(char\_exp) | Any whitespace character (' ', '\t', '\n') |

// Boniface Sindala

// CIS 251 (Spring 2013)

// Laboratory Assignment 6

// Wednesday, February 20, 2013

// In this laboratory assignment, I am going to use the void function

// with a reference parameter and a value parameter

// to adjust one exam score for each student based on a curve.

#include <iostream>

using namespace std;

void curveExamScore(int examScore, int curvePoints); //Function Declaration

int main()

{

int curve\_points, exam\_score;

cout << "Exam Score Curve Program.\n" << endl;

cout << "How many points do you want to add "

<< "to each exam? ";

cin >> curve\_points;

cout << endl;

cout << "Enter the first exam score (-1 to exit): ";

cin >> exam\_score;

while (exam\_score >= 0) //Function called inside loop

{

curveExamScore(exam\_score, curve\_points);

cout << "Exam Score after Curve: " << exam\_score + curve\_points << endl << endl; //Curve Score Calculation and Display

cout << "Enter another exam score (-1 to exit) : ";

cin >> exam\_score;

}

cout << endl << "End of Program!" << endl << endl;

return 0;

}

void curveExamScore(int examScore, int curvePoints) //Function Defenition

{

examScore = examScore + curvePoints;

}

#include <fstream>

#include <iostream>

#include <cstdlib>

#include <iomanip>

using namespace std;

int main()

{

ifstream originalFile;

ofstream newFile;

char fileName[31];

int val1, val2, val3;

cout << "What file would you like to open? ";

cin >> fileName;

originalFile.open(fileName);

while (originalFile.fail())

{

cout << "Error! " << fileName << " file does not exist." << endl;

cout << "Enter a different file name: ";

cin >> fileName;

originalFile.open(fileName);

}

newFile.open("New.txt");

originalFile >> val1 >> val2 >> val3;

newFile << val1 << setw(8) << val2 << setw (8) << val3 << endl;

originalFile.close();

newFile.close();

return 0;

}

//Program to create a file called cplusad.dat theat is identical to the file

//cad.dat, except that all occurences of 'C' are replaced by "C++"

//Assumes that the uppercase letter 'C' does not occur in cad.dat except

//as the name of the C programming language.

#include <fstream>

#include <iostream>

#include <cstdlib>

using namespace std;

void add\_plus\_plus(ifstream& in\_stream, ofstream& out\_stream);

//Precondition: in\_stream has been connected to an input file with open.

//out\_stream has been connected to an output file with open.

//Postcondition: The contents of the file connected to in\_stream have been

//copied into the file connected to out\_stream, but with each 'C' replaced

//by "C++". (The files are not closed by this function.)

int main()

{

ifstream fin;

ofstream fout;

cout << "Begin editing files.\n";

fin.open("cad.dat");

if (fin.fail())

{

cout << "Input file opening failed.\n";

exit(1);

}

fout.open("cplusad.dat");

if (fout.fail())

{

cout << "Output file opening failed.\n";

exit(1);

}

add\_plus\_plus(fin, fout);

fin.close();

fout.close();

cout << "End of editing files.\n";

return 0;

}

void add\_plus\_plus(ifstream& in\_stream, ofstream& out\_stream)

{

char next;

in\_stream.get(next);

while (! in\_stream.eof())

{

if (next == 'C')

out\_stream << "C++";

else

out\_stream << next;

in\_stream.get(next);

}

}