NEW THINK TANK

# C++ Tutorial 12

Posted by Derek Banas on Apr 25, 2018 in C Video Tutorial | 0 comments

In this part of my C++ tutorial we'll cover Operator Overloading, File I/O and will present you with another problem to solve. We'll learn to use unary and Binary operators. We'll specifically overload ++, print customer strings, +, [], ==, <, >, and =. Then we'll write and read from files.

All of the code and a transcript follows the video below. I hope you enjoy the tutorial.

If you like tutorials like this, consider donating $1, or simply turn off Ad Blocking software when watching my videos. Either helps me to keep my videos 100% free.

## Code & Transcript

```cpp
// ---------- C++ TUTORIAL 12 ----------

// ---------- OPERATOR OVERLOADING ----------

#include <cstdlib>
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <numeric>
#include <cmath>

// Needed for ostringstream
#include <sstream>

// Create a custom Box class with overloaded operators
class Box{
public:
    double length, width, breadth;

    // Used to hold a string representation of a box
    std::string boxString;

    Box(){
        length = 1, width = 1, breadth = 1;
    }
    Box(double l, double w, double b){
        length = l, width = w, breadth = b;
    }

    // You can define customer operators just like
    // you define functions
    // This is a unary operator because it operates
    // on 1 object

    // Other Unary Operators : --, *(pointer dereference),
    // -> (Member Selection), !, & (Address of), +, -
    Box& operator ++ (){
        length++;
        width++;
        breadth++;
        return *this;
    }

    // Creates a C string representation which is a
    // pointer to an array that is null terminated
    operator const char*() {
        // Creates a stream that can be loaded with
        // characters that can then be accessed as
        // a string object
        std::ostringstream boxStream;
        boxStream << "Box : " <<
                length << ", " <<
                width << ", " <<
                breadth;

        // Return a string representation of the stream
        boxString = boxStream.str();

        // Returns the pointer to the string array
        return boxString.c_str();
    }

    // Binary operators operate on 2 objects
    // +, -, *, /, %, ==, !=, >, <, >=, <=, &&, ||,
    // !, =, +=, -=, *=, /=, ^, [], &, |

    // Let's add boxes
    Box operator + (const Box& box2){
        Box boxSum;
        boxSum.length = length + box2.length;
        boxSum.width = width + box2.width;
        boxSum.breadth = breadth + box2.breadth;
        return boxSum;
    }

    // Access items using a subscript operator
    double operator [] (int x){
        if(x == 0){
            return length;
        } else if(x == 1){
            return width;
        } else if(x == 2){
            return breadth;
        } else {
            return 0;
        }
    }

    // Check for box equality
    bool operator == (const Box& box2){
        return ((length == box2.length) &&
                (width == box2.width) &&
                (breadth == box2.breadth));
    }

    // Check for which is bigger
    bool operator < (const Box& box2){
        double thisSize = length + width + breadth;
        double box2Size = box2.length + box2.width +
```

```cpp
        box2.breadth;
        if (thisSize < box2Size){
            return true;
        } else {
            return false;
        }
    }

    bool operator > (const Box& box2){
        double thisSize = length + width + breadth;
        double box2Size = box2.length + box2.width +
        box2.breadth;
        if (thisSize > box2Size){
            return true;
        } else {
            return false;
        }
    }

    // Overload the assignment operator
    void operator = (const Box& boxToCopy){
        length = boxToCopy.length;
        width = boxToCopy.width;
        breadth = boxToCopy.breadth;
    }
};

int main()
{
    Box box(10,10,10);

    // Will increment all values in the box by 1
    ++box;
    std::cout << box << "\n";

    // Add boxes
    Box box2(5,5,5);
    std::cout << "Box1 + Box2 = " <<
            box + box2 << "\n";

    // Access data with subscript operator
    std::cout << "Box Length : " <<
            box[0] << "\n";

    // Displays true or false for bolleans
    std::cout << std::boolalpha;
    std::cout << "Are boxes equal : " <<
            (box == box2) << "\n";

    // Is box < box2
    std::cout << "Is box < box2 : " <<
            (box < box2) << "\n";

    // Is box > box2
    std::cout << "Is box < box2 : " <<
            (box > box2) << "\n";

    box = box2;
    std::cout << box << "\n";
    return 0;
}

// ---------- OPERATOR OVERLOADING ----------

// ---------- FILE I/O & PROBLEM ----------
```

```

#include <cstdlib>
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <numeric>
#include <cmath>
#include <sstream>

// iostream allows use to read from the standard
// input (keyboard) and write to the standard output
// (console)
// fstream is needed for working with files
#include <fstream>

// ----- PROBLEM FUNCTION PROTOTYPE -----

std::vector<std::string> StringToVector(std::string,
        char separator);

// ----- END OF PROBLEM FUNCTION PROTOTYPE -----

int main()
{
    std::ofstream writeToFile;
    std::ifstream readFromFile;
    std::string txtToWrite = "";
    std::string txtFromFile = "";

    // We open the file by providing a name and then either
    // ios::app : Append to the end of the file
    // ios::trunc : If the exists delete content
    // ios::in : Open file for reading
    // ios::out : Open file for writing
    // ios::ate : Open writing and move to the end of the file
    writeToFile.open("test.txt", std::ios_base::out |
            std::ios_base::trunc);

    if(writeToFile.is_open()){
        // You can write with the stream insertion operator
        writeToFile << "Beginning of File\n";

        // You can write data in a string
        std::cout << "Enter data to write : ";
        getline(std::cin, txtToWrite);
        writeToFile << txtToWrite;

        // Close the file
        writeToFile.close();
    }

    // Open the file for reading
    readFromFile.open("test.txt", std::ios_base::in);

    if(readFromFile.is_open()){

        // Read text from file
        while(readFromFile.good()){
            getline(readFromFile, txtFromFile);

            // Print text from file
            std::cout << txtFromFile << "\n";

            // ----- PROBLEM -----
            // After each line print both the number of
            // words in each line and the average word length

            std::vector<std::string> vect =
                    StringToVector(txtFromFile, ' ');

            int wordsInLine = vect.size();

            std::cout << "Words in Line : " <<
                     wordsInLine << "\n";

            int charCount = 0;

            for(auto word: vect){
                for(auto letter: word){
                    charCount++;
                }
            }

            int avgNumChars = charCount/wordsInLine;

            std::cout << "Avg Word Length : " <<
            avgNumChars << "\n";

            // ----- END OF PROBLEM -----
        }
        readFromFile.close();
    }

    return 0;
}

// ----- PROBLEM FUNCTION -----

std::vector<std::string> StringToVector(std::string theString,
        char separator){

    // Create a vector
    std::vector<std::string> vecsWords;

    // A stringstream object receives strings separated
    // by a space and then spits them out 1 by 1
    std::stringstream ss(theString);

    // Will temporarily hold each word in the string
    std::string sIndivStr;

    // While there are more words to extract keep
    // executing
    // getline takes strings from a stream of words stored
    // in the stream and each time it finds a blanks space
    // it stores the word proceeding the space in sIndivStr
    while(getline(ss, sIndivStr, separator)){

        // Put the string into a vector
        vecsWords.push_back(sIndivStr);
    }

    return vecsWords;
}

// ----- END OF PROBLEM FUNCTION -----

// ---------- FILE I/O & PROBLEM ----------
```

## Leave a Reply

Your email address will not be published.

Comment

Name

Email

Website

Submit Comment

Search

Search

Social Networks

Facebook
YouTube
Twitter
LinkedIn

Buy me a Cup of Coffee
"Donations help me to keep the site running. One dollar is greatly appreciated." - (Pay Pal Secured)

Donate

My Facebook Page
Like Share 6K people like this. Be the first of your friends.

Archives

- October 2018
- September 2018
- August 2018
- July 2018
- June 2018
- May 2018
- April 2018
- March 2018
- February 2018
- January 2018
- December 2017
- November 2017
- October 2017

- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015
- November 2015
- October 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- November 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013