# Conquer

Conquer is a programming language designed and developed by **Daniel Puckowski**.

*Current version: 0.6.0.0 (beta-6)*

# Purpose

Conquer facilitates easy multithreaded image processing and manipulation.

# Language overview

- Conquer uses a lazy parser. The parsing process of a function body is deferred until it is completely needed.
- Child threads have access to every variable defined by their parents.
- The main thread will automatically wait for all background threads to terminate before terminating itself.
- Conquer is implemented in C++.

# Supported features

## Datatypes

- int

- double
- string
- bool
- file

## Variable declaration

Examples:

```
int x;
double x;
string s;
bool b;
file f;
```

## Variable statements

Examples:

```
intX = 100;
y = 512;

doubleD = 1.005;
pi = 3.14;

welcome = "Hello, world!";
prompt = ">> ";
```

```
result = true;
eval2 = false;

textFile = "input.txt";
imageFile = "img.ppm";
```

## Array declaration

Examples:

```
int x[512];
int x2[]; /* Array of size 0 */
double d [100];
double d2[]; /* Array of size 0 */
string s[20];
string s2[]; /* Array of size 0 */
bool b [4];
bool b2[]; /* Array of size 0 */
file f[128];
file f2; /* Array of size 0 */
```

## Array statements

Examples:

```
xArr[0] = 100;
```

```
int index;
index = 128;

doubleArr[index] = 1.50;

string next;
next = stringArr[2];
```

## Function declaration

Examples:

```
test
{
    int x;
}

test2 {
    innerFunc1 {
        string s;
    }

    double d;
}
```

# Comments

Examples:

```
/*
This is a comment.
*/
commentedFunction1 {
    /*
    Do something...
    */
}
```

# Output

Examples:

```
/*
Prompt without newline.
*/
print ">> ";

string welcome;
welcome = "Hello, world!";
/*
"endl" will output a newline.
*/
```

```
print welcome << " How are you today?" << endl;

println "A newline will be output after this string.";

file f;
f = "out.txt";
f << welcome; /* Append welcome string variable value to
out.txt */
```

# Input

Examples:

```
string word1;
string word2;

/*
Get two "words" from user via the console. Word ends at s
pace character. First word stored in word1 variable. Seco
nd word stored in word2 variable.
*/
read word1 >> word2;

/*
Read a line of user input from the console. Line ends wit
h Enter key/a newline character.
*/
```

```
string line;
readln >> line;

file f;
f = "data.txt";
string fileContents;
f >> fileContents; /* Read entirety of data.txt into file
Contents string variable. */
```

## If statements

Examples:

```
int x;
x=5;
int y;
y = 6;

if x == y {
    print "x and y are equal!";
}

if x != y
{
    print "x and y are not equal.";
}
```

# While statements

Examples:

```
/*
index 0 by default
*/
int index;

while index < 1000 {
    println << index;
}

index = 0;
int target;
target = 1000;

while index < target
{
    print "index: " << index << endl;
    print << "target: " << target << endl;
}
```

# Operators

Available operators:

| | | |
|---|---|---|

| Operator | Name | Datatypes supported |
|---|---|---|
| = | Assignment | int, double, string, bool, file |
| == | Equals | int, double, string, bool, file |
| != | Not equals | int, double, string, bool, file |
| += | Plus equals | int, double, string |
| -= | Minus equals | int, double |
| *= | Multiply equals | int, double |
| /= | Divide equals | int, double |
| ^= | Power equals | int, double |
| . | Dot | int, double, string, bool, file |
| < | Less than | int, double, string, bool, file |
| <= | Less than or equals | int, double, string, bool, file |
| > | Greater than | int, double, string, bool, file |
| >= | Greater than or equals | int, double, string, bool, file |
| << | Output | int, double, string, bool, file |
| >> | Input | int, double, string, bool |

# Reserved keywords

- true
- false
- call

- if
- while
- import
- print
- println
- detach
- join
- read
- readln
- delete
- system
- lock
- unlock
- endl

## Call keyword

Examples:

```
test {
    print "test function called.";
}

/*
Run the test function.
*/
call test;

```

```
call test2; /* Error. test2 not yet defined. */

test2
{
    int x;
    x = 5;
    x ^= 5;
}
```

## Import keyword

Examples:

```
import "mathFunctions.txt"; /* Process code in mathFuncti
ons.txt and then return */
```

## Detach keyword

Examples:

```
countFunc1
{
    int x;

    while x < 1000
    {
        print x << endl;
```

```
        x += 1;
    }
}

call countFunc1; /* countFunc1 is executed on main thread
. */
detach countFunc1; /* countFunc1 is executed on a backgro
und thread. */
detach countFunc1; /* countFunc1 executed on another back
ground thread. */
```

## Join keyword

Examples:

```
countFunc2 {
    int y;

    while y < 2000 {
        println y << "...";

        y += 1;
    }
}

run
{
```

```
    detach countFunc2;
    detach countFunc2;
}

/*
Execute run on another thread and wait for that thread to
 finish processing before continuing.

Will wait for run to finish, as well as any background th
reads created within run via the detach keyword.
*/
join run;
```

## Delete keyword

Examples:

```
double largeArray[10000];
delete largeArray; /* All 10,000 elements are deleted. */

int toDelete;
toDelete = 5;
delete toDelete;
```

## System keyword

Examples:

```
system "ping google.com";

/*
Download a file using wget.
*/
system "wget twitter.com -O twitter.html";

/*
Create a temporary file "temp.txt" and append some system
 information, CPU information, and USB device information
 to the file.
*/
system "touch temp.txt; uname -a >> temp.txt; lscpu >> te
mp.txt; lsusb >> temp.txt";

/*
Calculate factorial of 5 and print result using perl.
*/
system "perl -MMath::BigInt -le 'print Math::BigInt->new(
5)->bfac()'";
```

## Lock and Unlock keywords

Examples:

```
string names[5];
int countdown;
```

```
int index;

countdown = 5;

simpleNameTask {
    lock countdown; /* Wait for this thread to get exclus
ive access to countdown variable. */
    lock index; /* Wait for this thread to get exclusive
access to index variable. */
    names[index] = "Placeholder";
    index += 1;
    countdown -= 1;
    unlock countdown; /* Allow other threads to modify co
untdown variable. */
    unlock index; /* Allow other threads to modify index
variable. */
}

detach simpleNameTask;
detach simpleNameTask;

while countdown > 3 {
    /* Busy wait... */
}
```

## String functions

Available string functions:

| Function | Name | Argument list |
|---|---|---|
| length | Length | int variable |
| find | Find | string variable or string, int variable |
| substr | Substring | int variable or integer, int variable or integer, string variable |
| to_int | String to int | int variable |
| to_double | String to double | double variable |
| to_bool | String to bool | bool variable |
| take_line | Take line | string variable |

Examples:

```
int str1Length;
string str1;
str1 = "12345";
str1.length(str1Length); /* str1Length is now 5 */

int indexOfTwentyThree;
str1.find("23", indexOfTwentyThree); /* indexOfTwentyThre
e is now 1 */
int indexAbc;
```

```
str1.find("abc", indexAbc); /* indexAbc is now -1 as "abc
" does not exist in str1 */

int numCharsToCopy;
numCharsToCopy = 3;
string substr1;
str1.substr(0, numCharsToCopy, substr1); /* substr1 is no
w "123". First argument indicates start index of substrin
g. Second argument indicates number of characters to copy
 into substring. Third argument is the string variable to
 copy to. */

int numFromString;
string someNumString;
someNumString = "5555";
someNumString.to_int(numFromString); /* numFromString is
now 5555 */

bool boolFromString;
string tString;
tString = "true";
tString.to_bool(boolFromString); /* boolFromString is now
 true */

double pi;
string piString;
piString = "3.14";
piString.to_double(pi); /* pi is now 3.14 */
```

```
string irisDataFromFile;
string oneLine;
file f;
f = "iris.txt";
f >> irisDataFromFile; /* irisDataFromFile now contains c
ontents of iris.txt */
irisDataFromFile.take_line(oneLine); /* oneLine is set to
 the first line of the irisDataFromFile string. The first
 line from irisDataFromString is removed. */
```

# File functions

Available file functions:

| Function | Name | Argument list |
|---|---|---|
| exist | Exist | bool variable |
| remove | Remove | bool variable |
| open | Open | bool variable |
| close | Close | bool variable |
| read_into_array | Read into array | int[] variable, int variable, int variable, int variable or integer |
| write_array | Write array to file | int[] variable, int variable, int variable, int variable or integer |
| list_files | List files in directory | string[] variable |

Examples:

```
file testFile;
testFile = "test.txt";

bool isExist;
testFile.exist(isExist); /* isExist set to true if file e
xists */

if testFile == true {
    println "test.txt already exists.";
}

if testFile == true
{
    bool successfullyRemoved;
    testFile.remove(successfullyRemoved); /* remove set t
o true if file successfully removed */
}

int imgArray[]; /* Array of size 0 */
int width;
int height;

file myBitmap;
myBitmap = "Landscape.bmp";
myBitmap.read_into_array(imgArray, width, height, 3); /*
```

```
Width and height of image stored in width and height vari
ables. 3-dimensional/single channel data */

println "Image data: " << imgArray;

bool fileOpResult;
testFile = "test2.txt";

testFile.open(fileOpResult);
testFile << "This string is written to "test2.txt"" << en
dl;
testFile.close(fileOpResult);

file mySecondBitmap;
mySecondBitmap = "Landscape_Two.bmp";
mySecondBitmap.write_array(imgArray, width, height, 1); /
* Write contents of imgArray to file. 1-dimensional/singl
e channel data */

file currentDirectory;
currentDirectory = ".";
string currentDirectoryListing[]; /* Empty string array *
/
currentDirectory.list_files(currentDirectoryListing); /*
currentDirectoryListing now contains a list of all of the
 files in the current directory (pointed to by ".") */
```

# Integer functions

Available integer functions:

| Function | Name | Argument list |
|---|---|---|
| to_double | To double | double variable |
| to_string | To string | string variable |
| normalize | Normalize | int variable or integer, int variable or integer |
| abs | Absolute value | int variable |

Examples:

```
double d; /* d is 0 by default */
int x;

x = 5;
x.to_double(d); /* d is now 5.0 */

string xString;
x.to_string(xString);

int y;
y = -5;
```

```
y.abs(y); /* y is now 5 */

int values[10];
values.normalize(0, 255); /* Values in int array "values"
 normalized to range [0, 255] */

int value;
value = 512;
value.normalize(0, 100); /* Error. Normalize may only be
called on an array */
```

## Double functions

Available double functions:

| Function | Name | Argument list |
|---|---|---|
| to_int | To integer | int variable |
| to_string | To string | string variable |
| abs | Absolute value | double variable |
| rand | Next random | int variable or integer, or none |

Examples:

```
int x; /* x is 0 by default */
double d;
```

```
d = 5.123;
x.to_int(d); /* x is now 5 (5.123 is truncated) */

string dString;
d.to_string(dString);

double negativePi;
negativePi = -3.14;

negativePi.abs(negativePi); /* negativePi is now 3.14 */

double nextRand;
nextRand.rand(); /* nextRand is now a random value betwee
n 0.0 and 1.0 */

nextRand.rand(443); /* nextRand is a now a random value b
etween 0.0 and 1.0. A seed of 443 is used for the pseudor
andom number generator. */
nextRand.rand(0); /* Use the current time as the seed */
nextRand.rand(-1); /* Negative values also use the curren
t time as the seed */

int seed;
seed = 8080;
nextRand.rand(seed);
```

## Boolean functions

Available boolean functions:

| Function | Name | Argument list |
| --- | --- | --- |
| to_string | To string | string variable |

Examples:

```
bool b;
b = true;

string bString;
b.to_string(bString);
```

## Converting non-string datatypes to strings

Examples:

```
int x;
double d;
bool b;

x = -16;
d = 3.14;
b = false;
```

```
string xString;
string dString;
string bString;

x.to_string(xString);
d.to_string(dString);
b.to_string(bString);

/* Alternative, slower method using file IO... */
file f;
bool fileOpResult;

f = "temp.tmp";
f << x;
f >> xString; /* xString is now "-16" */
f.remove(fileOpResult);

f = "temp.tmp";
f << d;
f >> dString; /* dString is now "3.14" */
f.remove(fileOpResult);

f = "temp.tmp";
f << b;
f >> bString; /* bString is now "false" */
f.remove(fileOpResult);
```

## Supported image formats

- RAW
- ASC (Ascii)
- HDR (Analyze 7.5)
- INR (Inrimage)
- PPM/PGM (Portable Pixmap)
- BMP (uncompressed)
- PAN (Pandore-5)
- DLM (Matlab ASCII)

Reading of image files handled by the CImg Library. The CImg Library is a small, open-source, and modern C++ toolkit for image processing. For more information about the CImg Library, refer to the project's website: http://cimg.eu/.

## Test script – test.txt

```
println "Test script...";


println endl << "Testing variable declarations...";
int x;
double d;
string s;
bool b;
file f;
intx2;
```

```
doubled2;
strings2;
boolb2;
filef2;
int  x3;
double  d3;
string  s3;
bool  b3;
file  f3;

println endl << "Testing variable statements...";
x = 100;
d = 3.14;
s = "Hello, world!";
b = true;
f = "testFile1.txt";
x2=-100;
d2=-5.005;
s2="Test string 2.";
b2=false;
f2="testFile2.txt";
x3  =  999;
d3  =  0.001;
s3  =  "Test string 3.";
b3  =  false;
f3  =  "testFile3.txt";

/*x="s";          *//* Expected error */
```

```
/*x2 = 1.2345;    *//* Expected error */
/*x3  =  false;   *//* Expected error */
/*d=5;            *//* Expected error */
/*d2 = "double";  *//* Expected error */
/*d3  =  false;   *//* Expected error */
/*s=5;            *//* Expected error */
/*s2 = true;      *//* Expected error */
/*s3  =  5.43;    *//* Expected error */
/*b=0;            *//* Expected error */
/*b2 = "bool";    *//* Expected error */
/*b3  =  0.0;     *//* Expected error */
/*f=1000;         *//* Expected error */
/*f2 = 1024.2401; *//* Expected error */
/*f3  =  true;    *//* Expected error */

println endl << "Testing array declarations...";
int xArr[5];
double dArr[5];
string sArr[5];
bool bArr[5];
file fArr[5];
int x2Arr[0];

/*int x3Arr[abc];      *//* Expected error */
/*int x4Arr[size];     *//* Expected error */
int size;
size = 5;
int x5Arr[size];
```

```
/*string sArr2[-1];    *//* Expected error */
int size2;
size2 = -10;
/*double dArr2[size2]; *//* Expected error */

println endl << "Testing array statements...";
int index;
xArr[index] = -123;
dArr[index] = 99.9;
sArr[index] = "array assign test";
bArr[index] = false;
fArr[index] = "testFile4.txt";
xArr[0] = -123;
dArr[0] = 99.9;
sArr[0] = "array assign test";
bArr[0] = false;
fArr[0] = "testFile4.txt";
xArr[index] = x;
dArr[index] = d;
sArr[index] = s;
bArr[index] = b;
fArr[index] = f;
xArr[0] = x;
dArr[0] = d;
sArr[0] = s;
bArr[0] = b;
fArr[0] = f;
x = xArr[0];
```

```
d = dArr[0];
s = sArr[0];
b = bArr[0];
f = fArr[0];
x = xArr[index];
d = dArr[index];
s = sArr[index];
b = bArr[index];
f = fArr[index];

/*xArr[-1] = 0;    *//* Expected error */
index = -1;
/*xArr[index] = 0; *//* Expected error */
/*xArr[-1] = x;    *//* Expected error */
/*xArr[index] = x; *//* Expected error */

println endl << "Testing print keyword...";
print "Test 1";
print << "Test 2";
print << "Test 3" <<;
println "Test 4";
println << "Test 5";
println << "Test 6" <<;
print "Test 7" << endl;
print endl << "Test 8" << endl;
println "Test 9" << endl;
print;
println;
```

```
println << x;
print << x << endl;
println << d;
print << d << endl;
println << s;
print << s << endl;
println << b;
print << b << endl;
println << f;
print << f << endl;
println x;
println d;
println s;
println b;
println f;
print x;
print d;
print s;
print b;
print f;
print "x: " << x << "." << endl;
print "d: " << d << "." << endl;
print "s: " << s << "." << endl;
print "b: " << b << "." << endl;
print "f: " << f << "." << endl;
println << x << " " << d << " " << s << " " << b << " " <
< f << " " << endl;
print << x << " " << d << " " << s << " " << b << " " <<
```

```
f << " " << endl;
print << "<<" << ".";
println << "<<" << ".";
print << "Operator <<" << ".";
println << "Operator <<" << ".";
print << "==" << ".";
println << "==" << ".";
print << "Operator ==" << ".";
println << "Operator ==" << ".";
print << "!=" << ".";
println << "!=" << ".";
print << "Operator !=" << ".";
println << "Operator !=" << ".";
print << "<" << ".";
println << "<" << ".";
print << "Operator <" << ".";
println << "Operator <" << ".";
print << "<=" << ".";
println << "<=" << ".";
print << "Operator <=" << ".";
println << "Operator <=" << ".";
print << ">" << ".";
println << ">" << ".";
print << "Operator >" << ".";
println << "Operator >" << ".";
print << ">=" << ".";
println << ">=" << ".";
print << "Operator >=" << ".";
```

```
println << "Operator >=" << ".";
print << "+=" << ".";
println << "+=" << ".";
print << "Operator +=" << ".";
println << "Operator +=" << ".";
print << "-=" << ".";
println << "-=" << ".";
print << "Operator -=" << ".";
println << "Operator -=" << ".";
print << "*=" << ".";
println << "*=" << ".";
print << "Operator *=" << ".";
println << "Operator *=" << ".";
print << "/=" << ".";
println << "/=" << ".";
print << "Operator /=" << ".";
println << "Operator /=" << ".";
print << "^=" << ".";
println << "^=" << ".";
print << "Operator ^=" << ".";
println << "Operator ^=" << ".";

println endl << "Declaring functions...";
test {
    println "test running...";

    string s4;
    s4 = "String declared in function.";
```

```
    innerFunc1
    {
        println "innerFunc1 running...";
    }
}

test2
{
    println "test2 running...";

    innerFunc2 {
        println "innerFunc2 running...";
    }
}

test3 { println "test3 running..."; innerFunc3 { println
"innerFunc3 running..."; } }

countFunc1 {
    println "countFunc1 called...";

    int c1;
    c1 = 0;

    while c1 < 2000 {
        c1 += 1;
        totalCount += 1;
```

```
        println "Count: " << totalCount;
    }
}

countFunc2 {
    println "countFunc2 called...";

    int c2;
    c2 = 0;

    while c2 < 2000 {
        c2 += 1;
        totalCount += 1;
        println "Count: " << totalCount;
    }
}

countFunc3 {
    println "countFunc3 called...";

    int c3;
    c3 = 0;

    while c3 < 2000 {
        c3 += 1;
        totalCount += 1;
        println "Count: " << totalCount;
    }
```

```
}

run
{
    println "Run called...";

    detach countFunc1;
    detach countFunc2;
    detach countFunc3;
}

println endl << "Running test functions...";
call test;
call innerFunc1;
call test2;
call innerFunc2;

/*call innerFunc3; *//* Expected error */

call test3;
call innerFunc3;

println endl << "Testing code comments...";
/*
This is a comment.
*/

/* This is also a comment. */ println "Print statement be
```

```
tween comments."; /* Comment after statement. */

println endl << "Testing operators...";
int xOp1;
int xOp2;

xOp1 = 5;
xOp2 = 5;

xOp1 += 1;
xOp1 += xOp2;
xOp1 -= 1;
xOp1 *= 2;
xOp1 /= 2;
xOp1 ^= 2;

double dOp1;
double dOp2;

dOp1 = 5.0;
dOp2 = 5.0;

dOp1 += 1.0;
dOp1 += dOp2;
dOp1 -= 1.0;
dOp1 *= 2.0;
dOp1 /= 2.0;
dOp1 ^= 2.0;
```

```
string sOp1;
string sOp2;

sOp1 = "str1";
sOp2 = "str2";

sOp1 += " ";
sOp1 += sOp2;

file fOp1;
file fOp2;

fOp1 = "fOp1";
fOp2 = "txt";

/*fOp1 += ".";  *//* Expected error */
/*fOp1 += fOp2; *//* Expected error */

xArr[0] += 1;
xArr[0] -= 1;
xArr[0] *= 2;
xArr[0] /= 2;
xArr[0] ^= 2;
index = 0;
xArr[index] += 1;
xArr[index] -= 1;
xArr[index] *= 2;
```

```
xArr[index] /= 2;
xArr[index] ^= 2;

dArr[0] += 1.0;
dArr[0] -= 1.0;
dArr[0] *= 2.0;
dArr[0] /= 2.0;
dArr[0] ^= 2.0;
index = 0;
dArr[index] += 1.0;
dArr[index] -= 1.0;
dArr[index] *= 2.0;
dArr[index] /= 2.0;
dArr[index] ^= 2.0;

sArr[0] += "added to index";
index = 0;
sArr[index] += "added to index";

fArr[0] += ".txt";
index = 0;
fArr[index] += ".txt";

println endl << "Testing if statements...";
if xOp1 == 100 {
    println "xOp1 success!";
}
```

```
if xOp1 != 100 {
    println "xOp1 fail!";
}

if dOp1 == 100.0
{
    println "dOp1 success!";
}

if dOp1 != 100.0
{
    println "dOp1 fail!";
}

int xIf1;
int xIf2;
xIf1 = 5;
xIf2 = 5;

if xIf1 == xIf2
{
    println "xIf1 == success!";
}

xIf2 += 1;

if xIf1 != xIf2
{
```

```
    println "xIf1 != success!";
}

println endl << "Testing while statements...";
index = 0;

while index < 100 {
    println "While loop running... index: " << index;
    index += 1;
}

if index == 100 {
    println "While loop success!";
}

if index != 100
{
    println "While loop fail!";
}

int target;
target = 100;
index = 0;

while index < target {
    println "While loop 2 running... index: " << index;
    index += 1;
}
```

```
if index == 100 {
    println "While loop 2 success!";
}

if index != 100
{
    println "While loop 2 fail!";
}

println << endl << "Testing detach and join keywords...";

int totalCount;
join run;

if totalCount == 6000 {
    println "Detach and join success!";
}

if totalCount != 6000 {
    println "Detach and join fail!";
}

/*
println << endl << "Testing read keywords...";
string readStr1;
string readStr2;
string readStr3;
```

```
print << "> ";
read >> readStr1 >> readStr2 >> readStr3;

print << "Result: ";
println << readStr1 << ", " << readStr2 << ", " << readSt
r3;

readln >> readStr1 >> readStr2 >> readStr3;

print << "Result: ";
println << readStr1 << ", " << readStr2 << ", " << readSt
r3;
*/

println << endl << "Testing import keyword...";
import "testing\scripts\draw.txt";
call drawPpmFunc;

import "testing\scripts\histogramTest.txt";
import "testing\scripts\threadedHistogramTest.txt";
/*
import "testing\scripts\grayscaleTest.txt";
import "testing\scripts\homogeneityTest.txt";
*/

call compareHistograms;
call threadedCompareHistograms;
```

```
println << endl << "Testing string functions...";
/*string s4; *//* Expected error */
s4 = "Hello, world!";
int s4Length;
s4.length(s4Length);
println "String: " << s4 << endl << "Length: " << s4Lengt
h;

int indexOfComma;
s4.find(",", indexOfComma);
println "String: " << s4 << endl << "Index of comma: " <<
 indexOfComma;

string commaString;
indexOfComma = 0;
commaString = ",";
s4.find(commaString, indexOfComma);
println "String: " << s4 << endl << "Index of comma 2: "
<< indexOfComma;

string s4Substr;
s4.substr(0, indexOfComma, s4Substr);
println "String: " << s4 << endl << "Substring (0 to comm
a): " << s4Substr;

int startIndex;
s4.substr(startIndex, indexOfComma, s4Substr);
```

```
println "String: " << s4 << endl << "Substring (0 to comm
a) 2: " << s4Substr;
```

## Second test script – draw.txt

```
drawPpmFunc
{
    /*file f; *//* Expected error when imported by test.t
xt */
    bool fileOpResult;

    f = "temp.ppm";
    f.exist(fileOpResult);

    println << "temp.ppm exists? " << fileOpResult;

    if fileOpResult == false
    {
        int width;
        width = 32;

        int height;
        height = 32;

        /*int size; *//* Expected error when imported by
test.txt */
        size = width;
```

```
        size *= height;

        string ppmArr[size];

        int bgColor;
        bgColor = 255;
        string bgColorString;
        bgColor.to_string(bgColorString);

        /*int index; *//* Expected error when imported by
test.txt */
        index = 0;

        /*
        Index starts at 0.
        */
        while index < size
        {
            ppmArr[index] = bgColorString;
            ppmArr[index] += " ";
            ppmArr[index] += bgColorString;
            ppmArr[index] += " 0";
            /*ppmArr[index] += "0";*/

            index += 1;
            println index;
        }
```

```
        int eyeIndex;
        eyeIndex = 196;

        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 17;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";

        eyeIndex += 9;

        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
```

```
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 17;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";

        eyeIndex += 9;

        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 17;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
```

```
        eyeIndex += 9;

        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 17;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";
        eyeIndex += 1;
        ppmArr[eyeIndex] = "0 0 255";


        int mouthIndex;
        mouthIndex = 580;

        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
```

```
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 17;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";

        mouthIndex += 9;

        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 17;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";

```

```
        mouthIndex += 13;

        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 9;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";

        mouthIndex += 17;

        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 9;
```

```
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";

        mouthIndex = 716;

        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";

        mouthIndex += 25;
```

```
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";
        mouthIndex += 1;
        ppmArr[mouthIndex] = "255 0 0";

        println << ppmArr;


        string widthString;
        width.to_string(widthString);

        string heightString;
        height.to_string(heightString);

        string depthString;
        depthString = "255";
```

```
        f = "temp.ppm";
        f.open(fileOpResult);
        f << "P3" << endl;
        f << widthString << " " << heightString << endl;
        f << depthString << endl;

        index = 0;
        string color;

        while index < size
        {
            color = ppmArr[index];
            f << color << " ";

            index += 1;
            println index;
        }

        f.close(fileOpResult);

        f.exist(fileOpResult);
        println << "temp.ppm exists? " << fileOpResult;

        if fileOpResult == true
        {
            f.remove(fileOpResult);
            println << "temp.ppm remove success? " << fil
eOpResult;
```

```
        }

        println "Done.";
    }

    if fileOpResult == true
    {
        println << "Could not draw PPM image... temp.ppm
already exists.";
    }
}
```

## Third test script – histogramTest.txt

```
compareHistograms
{
    println "Generating and comparing two histograms...";

    /*
    Chi-Square Distance, defined as
    sum( (xi - yi)^2 / (xi + yi) )
    */

    int imageOne[];
    int widthOne;
    int heightOne;
    file fOne;
```

```
    fOne = "testing\images\Landscape_Small.bmp";
    fOne.read_into_array(imageOne, widthOne, heightOne, 3
);

    imageOne.normalize(0, 255); /* [floor, ceiling] */

    int imageSize;
    imageSize = widthOne;
    imageSize *= heightOne;

    int histoOne[256];
    int loopIndex;
    int histoIndex;

    while loopIndex < imageSize
    {
        histoIndex = imageOne[loopIndex];
        histoOne[histoIndex] += 1;

        loopIndex += 1;
    }

    println "Histogram for: "Landscape_Small.bmp"";
    println << histoOne;


    int imageTwo[];
```

```
    int widthTwo;
    int heightTwo;
    file fTwo;

    fTwo = "testing\images\Landscape_Small_Edited.bmp";
    fTwo.read_into_array(imageTwo, widthTwo, heightTwo, 3
);

    int imageSizeTwo;
    imageSizeTwo = widthTwo;
    imageSizeTwo *= heightTwo;

    int histoTwo[256];
    int loopIndexTwo;
    int histoIndexTwo;

    while loopIndexTwo < imageSizeTwo
    {
        histoIndexTwo = imageTwo[loopIndexTwo];
        histoTwo[histoIndexTwo] += 1;

        loopIndexTwo += 1;
    }

    println "Histogram for: "Landscape_Small_Edited.bmp""
;
    println << histoTwo;
```

```
    double distance;
    int xInt;
    int yInt;
    double xVal;
    double yVal;
    double calc;
    double calc2;

    loopIndex = 0;

    while loopIndex < 256
    {
        xInt = histoOne[loopIndex];
        xInt.to_double(xVal);

        yInt = histoTwo[loopIndex];
        yInt.to_double(yVal);

        calc = xVal;
        calc -= yVal;
        calc ^= 2.0;

        calc2 = xVal;
        calc2 += yVal;

        if calc2 != 0.0
        {
```

```
            calc /= calc2;
            distance += calc;
        }

        loopIndex += 1;
    }

    println "Histogram chi-square distance: " << distance
;
}
```

## Fourth test script – threadedHistogramTest.txt

```
threadedCompareHistograms
{
    println "Generating and comparing two histograms...";

    /*
    Chi-Square Distance, defined as
    sum( (xi - yi)^2 / (xi + yi) )
    */

    int imageOne[];
    int widthOne;
    int heightOne;
    file fOne;
```

```
    fOne = "testing\images\Landscape_Small.bmp";
    fOne.read_into_array(imageOne, widthOne, heightOne, 3
);

    int imageSize;
    int fourthImageSize;
    imageSize = widthOne;
    imageSize *= heightOne;
    fourthImageSize = imageSize;
    fourthImageSize /= 4;

    int imageTwo[];
    int widthTwo;
    int heightTwo;
    file fTwo;

    fTwo = "testing\images\Landscape_Small_Edited.bmp";
    fTwo.read_into_array(imageTwo, widthTwo, heightTwo, 3
);

    histoThreadOne
    {
        int histoOne[256];
        int histoOne2[256];
        int loopIndexOne;
        int histoIndexOne;
```

```
        println "Thread 1 -- Start index: " << loopIndexO
ne << ", Target index: " << fourthImageSize;

        while loopIndexOne < fourthImageSize
        {
            histoIndexOne = imageOne[loopIndexOne];
            histoOne[histoIndexOne] += 1;

            loopIndexOne += 1;
        }

        loopIndexOne = 0;

        while loopIndexOne < fourthImageSize
        {
            histoIndexOne = imageTwo[loopIndexOne];
            histoOne2[histoIndexOne] += 1;

            loopIndexOne += 1;
        }
    }

    histoThreadTwo
    {
        int histoTwo[256];
        int histoTwo2[256];
        int loopIndexTwo;
        int histoIndexTwo;
```

```
        loopIndexTwo = fourthImageSize;

        int targetTwo;
        targetTwo = fourthImageSize;
        targetTwo *= 2;

        println "Thread 2 -- Start index: " << loopIndexT
wo << ", Target index: " << targetTwo;

        while loopIndexTwo < targetTwo
        {
            histoIndexTwo = imageOne[loopIndexTwo];
            histoTwo[histoIndexTwo] += 1;

            loopIndexTwo += 1;
        }

        loopIndexTwo = fourthImageSize;

        while loopIndexTwo < targetTwo
        {
            histoIndexTwo = imageTwo[loopIndexTwo];
            histoTwo2[histoIndexTwo] += 1;

            loopIndexTwo += 1;
        }
    }
```

```
    histoThreadThree
    {
        int histoThree[256];
        int histoThree2[256];
        int loopIndexThree;
        int histoIndexThree;

        loopIndexThree = fourthImageSize;
        loopIndexThree *= 2;

        int targetThree;
        targetThree = fourthImageSize;
        targetThree *= 3;

        println "Thread 3 -- Start index: " << loopIndexT
hree << ", Target index: " << targetThree;

        while loopIndexThree < targetThree
        {
            histoIndexThree = imageOne[loopIndexThree];
            histoThree[histoIndexThree] += 1;

            loopIndexThree += 1;
        }

        loopIndexThree = fourthImageSize;
        loopIndexThree *= 2;
```

```
        while loopIndexThree < targetThree
        {
            histoIndexThree = imageTwo[loopIndexThree];
            histoThree2[histoIndexThree] += 1;

            loopIndexThree += 1;
        }
    }

    histoThreadFour
    {
        int histoFour[256];
        int histoFour2[256];
        int loopIndexFour;
        int histoIndexFour;

        loopIndexFour = fourthImageSize;
        loopIndexFour *= 3;

        println "Thread 4 -- Start index: " << loopIndexF
our << ", Target index: " << imageSize;

        while loopIndexFour < imageSize
        {
            histoIndexFour = imageOne[loopIndexFour];
            histoFour[histoIndexFour] += 1;
```

```
            loopIndexFour += 1;
        }

        loopIndexFour = fourthImageSize;
        loopIndexFour *= 3;

        while loopIndexFour < imageSize
        {
            histoIndexFour = imageTwo[loopIndexFour];
            histoFour2[histoIndexFour] += 1;

            loopIndexFour += 1;
        }
    }

    runThreadedHisto
    {
        detach histoThreadOne;
        detach histoThreadTwo;
        detach histoThreadThree;
        detach histoThreadFour;
    }

    join runThreadedHisto;

    println "Histogram 1 for: "Landscape_Small.bmp"";
    println << histoOne;
    println "Histogram 1 for: "Landscape_Small_Edited.bmp
```

```
"";
    println << histoOne2;

    println;

    println "Histogram 2 for: "Landscape_Small.bmp"";
    println << histoTwo;
    println "Histogram 2 for: "Landscape_Small_Edited.bmp
"";
    println << histoTwo2;

    println;

    println "Histogram 3 for: "Landscape_Small.bmp"";
    println << histoThree;
    println "Histogram 3 for: "Landscape_Small_Edited.bmp
"";
    println << histoThree2;

    println;

    println "Histogram 4 for: "Landscape_Small.bmp"";
    println << histoFour;
    println "Histogram 4 for: "Landscape_Small_Edited.bmp
"";
    println << histoFour2;
```

```
compareThreadOne
{
    double distanceOne;
    int xIntOne;
    int yIntOne;
    double xOne;
    double yOne;
    double calcOne;
    double calc2One;

    loopIndexOne = 0;

    while loopIndexOne < 256
    {
        xIntOne = histoOne[loopIndexOne];
        xIntOne.to_double(xOne);

        yIntOne = histoOne2[loopIndexOne];
        yIntOne.to_double(yOne);

        calcOne = xOne;
        calcOne -= yOne;
        calcOne ^= 2.0;

        calc2One = xOne;
        calc2One += yOne;

        if calc2One != 0.0
```

```
            {
                calcOne /= calc2One;
                distanceOne += calcOne;
            }

            loopIndexOne += 1;
        }

        println "Histogram 1 chi-square distance: " << di
stanceOne;
    }

    compareThreadTwo
    {
        double distanceTwo;
        int xIntTwo;
        int yIntTwo;
        double xTwo;
        double yTwo;
        double calcTwo;
        double calc2Two;

        loopIndexTwo = 0;

        while loopIndexTwo < 256
        {
            xIntTwo = histoTwo[loopIndexTwo];
            xIntTwo.to_double(xTwo);
```

```
            yIntTwo = histoTwo2[loopIndexTwo];
            yIntTwo.to_double(yTwo);

            calcTwo = xTwo;
            calcTwo -= yTwo;
            calcTwo ^= 2.0;

            calc2Two = xTwo;
            calc2Two += yTwo;

            if calc2Two != 0.0
            {
                calcTwo /= calc2Two;
                distanceTwo += calcTwo;
            }

            loopIndexTwo += 1;
        }

        println "Histogram 2 chi-square distance: " << di
stanceTwo;
    }

    compareThreadThree
    {
        double distanceThree;
        int xIntThree;
```

```
        int yIntThree;
        double xThree;
        double yThree;
        double calcThree;
        double calc2Three;

        loopIndexThree = 0;

        while loopIndexThree < 256
        {
            xIntThree = histoThree[loopIndexThree];
            xIntThree.to_double(xThree);

            yIntThree = histoThree2[loopIndexThree];
            yIntThree.to_double(yThree);

            calcThree = xThree;
            calcThree -= yThree;
            calcThree ^= 2.0;

            calc2Three = xThree;
            calc2Three += yThree;

            if calc2Three != 0.0
            {
                calcThree /= calc2Three;
                distanceThree += calcThree;
            }
```

```
            loopIndexThree += 1;
        }

        println "Histogram 3 chi-square distance: " << di
stanceThree;
    }

    compareThreadFour
    {
        double distanceFour;
        int xIntFour;
        int yIntFour;
        double xFour;
        double yFour;
        double calcFour;
        double calc2Four;

        loopIndexFour = 0;

        while loopIndexFour < 256
        {
            xIntFour = histoFour[loopIndexFour];
            xIntFour.to_double(xFour);

            yIntFour = histoFour2[loopIndexFour];
            yIntFour.to_double(yFour);
```

```
            calcFour = xFour;
            calcFour -= yFour;
            calcFour ^= 2.0;

            calc2Four = xFour;
            calc2Four += yFour;

            if calc2Four != 0.0
            {
                calcFour /= calc2Four;
                distanceFour += calcFour;
            }

            loopIndexFour += 1;
        }

        println "Histogram 4 chi-square distance: " << di
stanceFour;
    }

    println;

    runThreadedHistoCompare
    {
        detach compareThreadOne;
        detach compareThreadTwo;
        detach compareThreadThree;
        detach compareThreadFour;
```

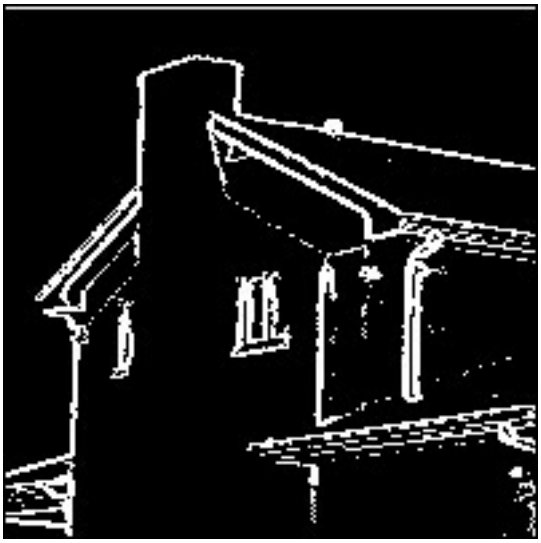
```
    }

    join runThreadedHistoCompare;

    println << "Done.";
}
```

## Fifth test script – homogeneityTest.txt

Original image (left) and result of edge detection (right).

```
int inputImage[];
int inputWidth;
int inputHeight;

file inputFile;
inputFile = "testing\images\House.bmp";
inputFile.read_into_array(inputImage, inputWidth, inputHe
```

```
ight, 1);

int sizeOfInputImage;
sizeOfInputImage = inputWidth;
sizeOfInputImage *= inputHeight;

int homogeneityOutput[sizeOfInputImage];

int high;
int new_hi;
int new_low;

/*high = 50;*/

int userThreshold;

print "Enter threshold for edge detection [0, 255]: ";
readln >> userThreshold;

high = userThreshold;

new_hi = 255;
new_low = 0;

int rowStart;
int colStart;
int rowEnd;
int colEnd;
```

```
rowStart = 1;
rowEnd = inputHeight;
rowEnd -= 1;
colEnd = inputWidth;
colEnd -= 1;

int maxDiff;

int aStart;
int aEnd;
int bStart;
int bEnd;

aEnd = 1;
bEnd = 1;

int index;
int diff;
int diff2;

int tempRow;
int tempThreshold;

println "Performing homogeneity edge detection function..
.";

while rowStart < rowEnd
```

```
{
    println "Row: " << rowStart;

    colStart = 1;

    while colStart < colEnd
    {
        /* println "Col: " << colStart; */

        maxDiff = 0;

        aStart = -1;
        bStart = -1;

        while aStart <= aEnd
        {
            while bStart <= bEnd
            {
                index = rowStart;
                index *= inputWidth;
                index += colStart;

                diff = inputImage[index];

                index = rowStart;
                index *= inputWidth;

                tempRow = aStart;
```

```
                tempRow *= inputWidth;
                index += tempRow;

                index += colStart;
                index += bStart;

                diff2 = inputImage[index];

                diff -= diff2;
                diff.abs(diff);

                if diff > maxDiff
                {
                    maxDiff = diff;
                }

                bStart += 1;
            }

            aStart += 1;
        }

        index = rowStart;
        index *= inputWidth;
        index += colStart;

        homogeneityOutput[index] = maxDiff;
```

```
        colStart += 1;
    }

    rowStart += 1;
}

println "Thresholding values...";

rowStart = 0;

while rowStart < rowEnd
{
    println "Row: " << rowStart;

    colStart = 0;

    while colStart < colEnd
    {
        /* println "Col: " << colStart; */

        index = rowStart;
        index *= inputWidth;
        index += colStart;

        tempThreshold = homogeneityOutput[index];

        if tempThreshold > high
        {
```

```
            homogeneityOutput[index] = new_hi;
        }

        if tempThreshold <= high
        {
            homogeneityOutput[index] = new_low;
        }

        colStart += 1;
    }

    rowStart += 1;
}

println "Writing homogeneity array...";

file outputFile;
outputFile = "homogeneity_test.bmp";
outputFile.write_array(homogeneityOutput, inputWidth, inp
utHeight, 1);
```

## Sixth test script – grayscaleTest.txt

Convert color bitmap to grayscale image. Five methods tested.

Original image in top left.

Methods used:

- Conversion to single channel (Red)
- Conversion to single channel (Green)
- RGB average (1/3 Red, Green, and Blue channel)
- Weighted RGB average (29.9% Red, 58.7% Green, and 11.4% Blue channel)
- Save as single channel (Defaults to RGB average)

(Order: left to right, top to bottom)

```
int inputImage[];
int inputWidth;
int inputHeight;

file inputFile;
inputFile = "testing\images\Landscape_Small.bmp";
inputFile.read_into_array(inputImage, inputWidth, inputHe
ight, 3);
```

```
int sizeOfInputImage;
sizeOfInputImage = inputWidth;
sizeOfInputImage *= inputHeight;

int outputImage[sizeOfInputImage];

sizeOfInputImage *= 3;

int r;
int g;
int b;
int grayValueInt;
int index;
int index2;

double doubleR;
double doubleG;
double doubleB;
double grayValue;
double temp;

println "Grayscale method 1...";
println "Converting to single channel (R)..." << endl;

while index < sizeOfInputImage
{
    r = inputImage[index];
```

```
    outputImage[index2] = r; /* Grayscale value */
    index += 3;              /* Point to next pixel */
    index2 += 1;

    /* println "Processing... " << index; */
}

file outputFile;
outputFile = "grayscale_test_1.bmp";
outputFile.write_array(outputImage, inputWidth, inputHeig
ht, 1);

println "Grayscale method 2...";
println "Converting to single channel (G)..." << endl;

index = 0;
index2 = 0;

while index < sizeOfInputImage
{
    index += 1;              /* Pixels are stored as RGB.
.. point to G */
    g = inputImage[index];   /* Get G. The human eye is m
ost sensitive to green, especially for high resolution lu
minance information. */
    index -= 1;

    outputImage[index2] = g; /* Grayscale value */
```

```
    index += 3;                 /* Point to next pixel */
    index2 += 1;

    /* println "Processing... " << index; */
}

outputFile = "grayscale_test_2.bmp";
outputFile.write_array(outputImage, inputWidth, inputHeig
ht, 1);

println "Grayscale method 3...";
println "Computing average of RGB..." << endl;

index = 0;
index2 = 0;

while index < sizeOfInputImage
{
    grayValue = 0.0;

    r = inputImage[index];
    r.to_double(doubleR);
    index += 1;

    g = inputImage[index];
    g.to_double(doubleG);
    index += 1;
```

```
    b = inputImage[index];
    b.to_double(doubleB);
    index += 1;

    temp = 0.33;
    temp *= doubleR;
    grayValue += temp;

    temp = 0.33;
    temp *= doubleG;
    grayValue += temp;

    temp = 0.33;
    temp *= doubleB;
    grayValue += temp;

    grayValue.to_int(grayValueInt);

    outputImage[index2] = grayValueInt;
    index2 += 1;

    /* println "Processing... " << index; */
}

outputFile = "grayscale_test_3.bmp";
outputFile.write_array(outputImage, inputWidth, inputHeig
ht, 1);
```

```
println "Grayscale method 4...";
println "Computing average of RGB (weighted)..." << endl;

index = 0;
index2 = 0;

while index < sizeOfInputImage
{
    grayValue = 0.0;

    r = inputImage[index];
    r.to_double(doubleR);
    index += 1;

    g = inputImage[index];
    g.to_double(doubleG);
    index += 1;

    b = inputImage[index];
    b.to_double(doubleB);
    index += 1;

    temp = 0.299;
    temp *= doubleR;
    grayValue += temp;

    temp = 0.587;
    temp *= doubleG;
```

```
    grayValue += temp;

    temp = 0.114;
    temp *= doubleB;
    grayValue += temp;

    grayValue.to_int(grayValueInt);

    outputImage[index2] = grayValueInt;
    index2 += 1;

    /* println "Processing... " << index; */
}

outputFile = "grayscale_test_4.bmp";
outputFile.write_array(outputImage, inputWidth, inputHeig
ht, 1);

println "Grayscale method 5...";
println "Saving as new single channel image..." << endl;

outputFile = "grayscale_test_5.bmp";
outputFile.write_array(inputImage, inputWidth, inputHeigh
t, 1);

println "Color save test...";

outputFile = "grayscale_test_6.bmp";
```

```
outputFile.write_array(inputImage, inputWidth, inputHeigh
t, 3);
```