Conquer

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

Current version: 0.7.2.0 (beta-7)

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:

```
/*
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

Keywords
true
false
call
if
while
import
print
println
detach

join
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	

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 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	
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 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 – 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();

                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);
```

Second test script – threadedEdgeDetection.txt

Original image (top) and result of edge detection (bottom). Original image and result image are 10,000 pixels wide by 3,768 pixels tall.

Link to unscaled original image:

https://lh3.googleusercontent.com/KBEJHuF0J-nX0CHEP_ecuzhoFzxFNU6OHQOOgpb6vVswbJzH7v6_knp6NTNDQFNXZh1014-rw

Link to unscaled result image:

https://lh5.googleusercontent.com/yCOVraLB0ajm7gpf6Il3TSSNCZjU_YJch1014-rw

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

file inputFile;
inputFile = "..\testing\images\Castle_10000px_3678px.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;

int userThreshold;

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

```
*/

userThreshold = 40;
high = userThreshold;

new_hi = 255;
new_low = 0;

func {
    int rowStart;
    int colStart;

    int rowEnd;
    int colEnd;

    rowStart = start;
    rowEnd = end;
    rowEnd -= 1;

    unlock start;
    unlock end;

    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;

    println "Performing homogeneity edge detection functi
on...";

    while rowStart <= rowEnd
    {
        colStart = 1;

        while colStart < colEnd
        {
            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();

                    if diff > maxDiff
                    {
                        maxDiff = diff;
                    }
```

```
                    bStart += 1;
                }

                aStart += 1;
            }

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

            homogeneityOutput[index] = maxDiff;

            colStart += 1;
        }

        rowStart += 1;
        println rowStart;
    }
}

int targetNumberOfThreads;
targetNumberOfThreads = 4;

int numberOfThreads;

int start;
int end;
```

```
int rowsPerThread;

func2 {
    rowsPerThread = inputHeight;
    rowsPerThread /= targetNumberOfThreads;

    lock start;
    lock end;

    start = 1;
    end = rowsPerThread;

    println "Starting at " << start << " and ending at "
<< end;
    detach func;

    numberOfThreads = 1;

    while numberOfThreads < targetNumberOfThreads {
        lock start;
        lock end;

        start = rowsPerThread;
        start *= numberOfThreads;
        start -= 1;

        end += rowsPerThread;
```

```
        println "Starting at " << start << " and ending a
t " << end;
        detach func;

        numberOfThreads += 1;
    }
}

println "Starting...";

join func2;

/*
func3 {
    println "Thresholding values...";

    int rowStart;
    int colStart;

    int rowEnd;
    int colEnd;

    rowStart = start;
    rowEnd = end;
    rowEnd -= 1;

    unlock start;
    unlock end;
```

```
    colEnd = inputWidth;
    colEnd -= 1;

    int index;
    int tempThreshold;

    while rowStart < rowEnd
    {
        colStart = 0;

        while colStart < colEnd
        {
            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;
    }
}

func4 {
    start = 1;
    end = rowsPerThread;

    lock end;
    lock start;

    println "Starting at " << start << " and ending at "
<< end;
    detach func3;

    numberOfThreads = 1;

    while numberOfThreads < targetNumberOfThreads {
        lock end;
        lock start;

        start = rowsPerThread;
        start *= numberOfThreads;
        start -= 1;
```

```
        end += rowsPerThread;

        println "Starting at " << start << " and ending a
t " << end;
        detach func3;

        numberOfThreads += 1;
    }
}

numberOfThreads = 1;
join func4;
*/

println "Writing homogeneity array...";

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

println "Done.";
```