

# Project 2 - Expression evaluator

---

**Author:** Tyler D Clark

**Date:** 12 October 2020

**Description** This assignment is completing and extending a C++ program that evaluates statements of an expression language. The program reads in the arithmetic expression from a file and encodes the expression as a binary tree. After the expression has been read in, the variable assignments are read in and the variables and their values of the variables are placed into the symbol table. Finally the expression is evaluated recursively.

---

## File Layout

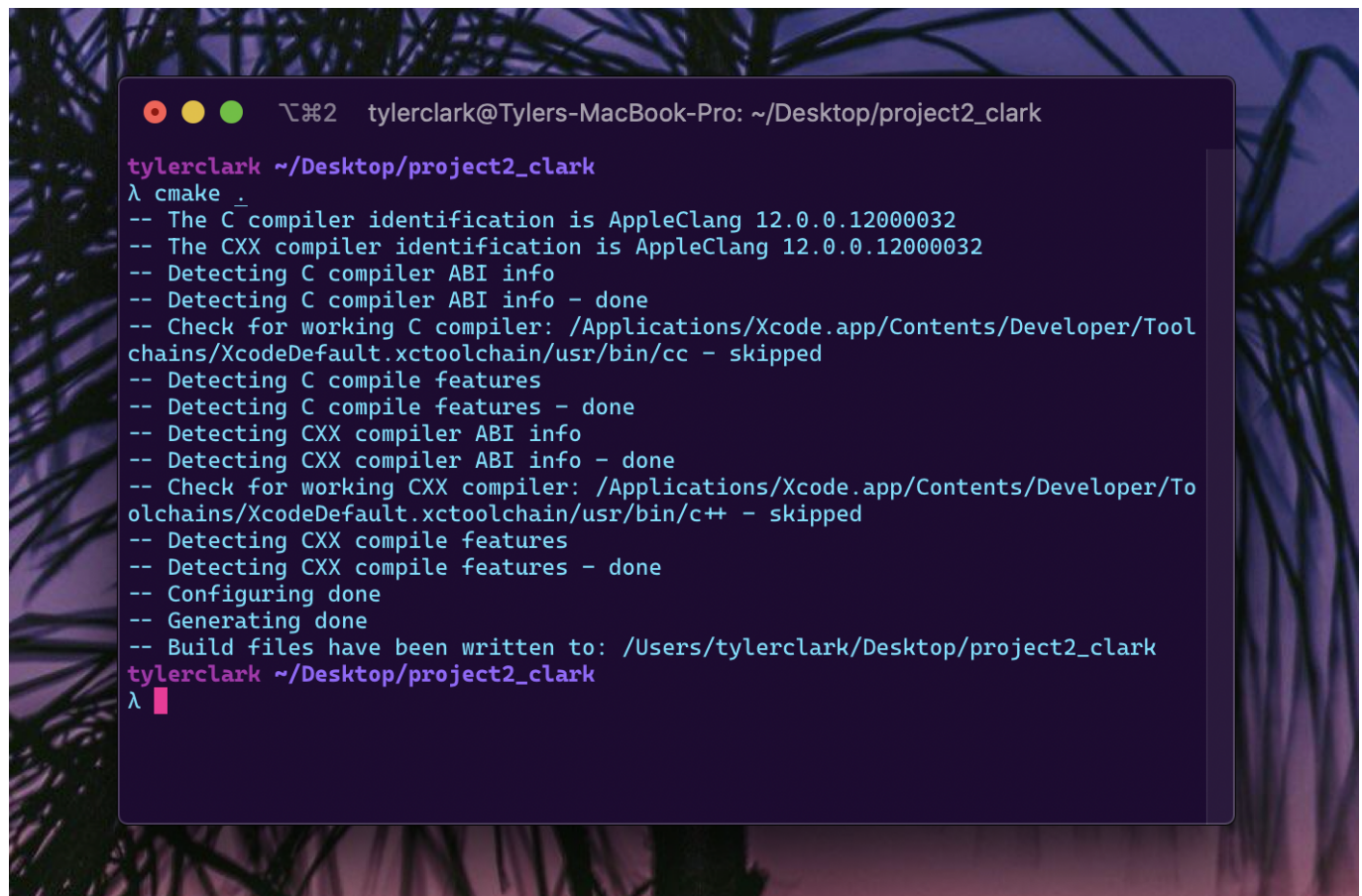
```
project2
|
|___CMakeLists.txt
|
|___include
| |___and.h
| |___lessthan.h
| |___divide.h
| |___times.h
| |___negate.h
| |___subexpression.h
| |___parse.h
| |___operand.h
| |___literal.h
| |___variable.h
| |___symboltable.h
| |___equalto.h
| |___expression.h
| |___plus.h
| |___or.h
| |___minus.h
| |___greaterthan.h
| |___conditional.h
|
|___input.txt
|
|___doc
| |___project2.md
| |___img
| |___Project2_instructions.pdf
|
|___src
| |___subexpression.cpp
| |___parse.cpp
| |___module3.cpp (main is here)
| |___symboltable.cpp
| |___variable.cpp
```

```
| |__operand.cpp
```

## Running the Program

To run this program, it can be done from an IDE or using the command-line and CMake. The following instructions show running it in the command-line. The first step is to enter the project 2 folder and enter the command:

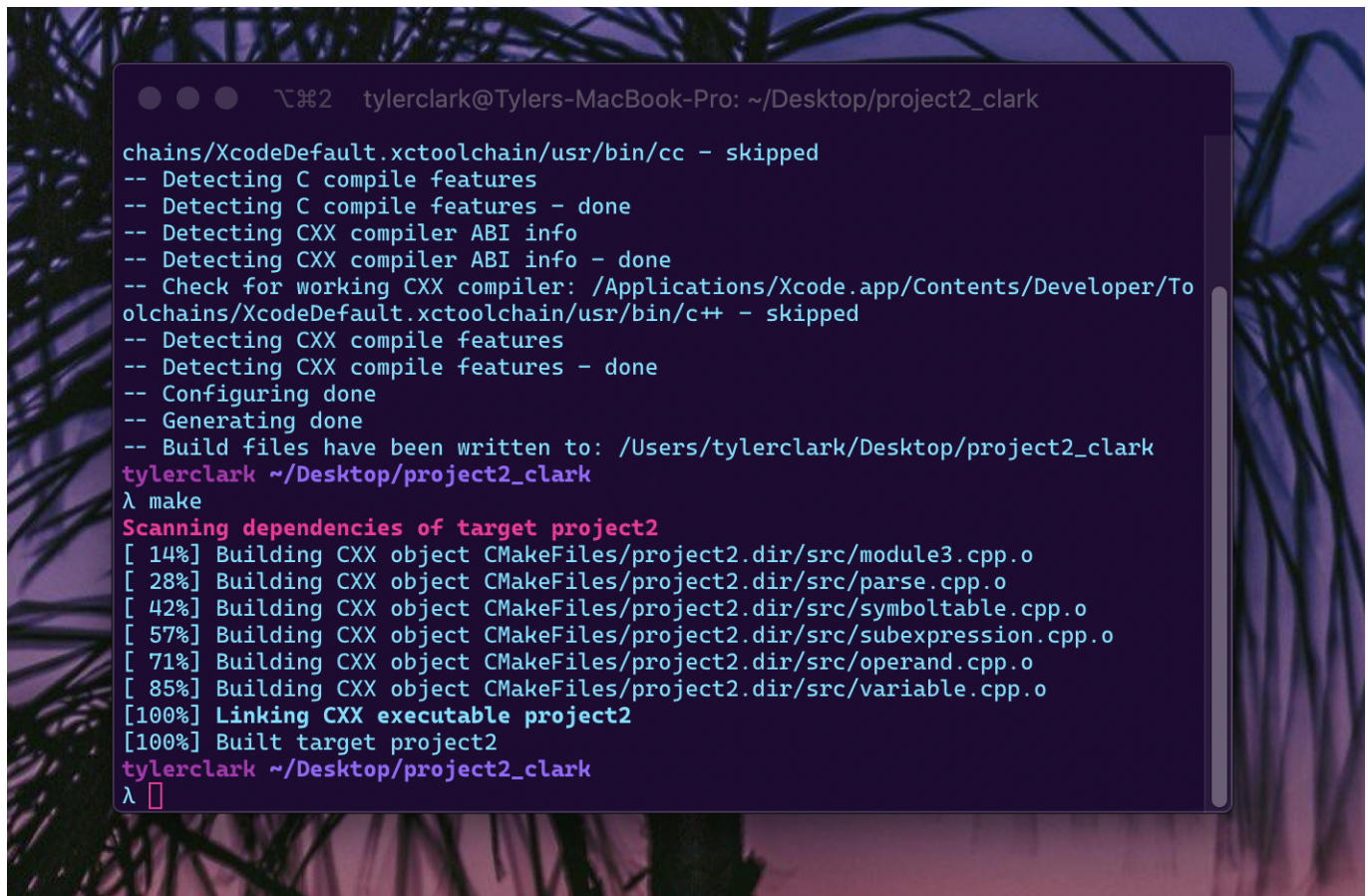
```
cmake .
```

A screenshot of a macOS terminal window with a dark purple background and a palm tree pattern. The window title is 'tylerclark@Tylers-MacBook-Pro: ~/Desktop/project2\_clark'. The prompt is 'tylerclark ~/Desktop/project2\_clark'. The user has entered 'λ cmake .' and the terminal shows the output of CMake configuration, including compiler identification for AppleClang 12.0.0.12000032, detection of C and CXX compiler ABI info, and generation of build files. The prompt returns to 'tylerclark ~/Desktop/project2\_clark λ' with a red cursor.

```
tylerclark@Tylers-MacBook-Pro: ~/Desktop/project2_clark
tylerclark ~/Desktop/project2_clark
λ cmake .
-- The C compiler identification is AppleClang 12.0.0.12000032
-- The CXX compiler identification is AppleClang 12.0.0.12000032
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/tylerclark/Desktop/project2_clark
tylerclark ~/Desktop/project2_clark
λ
```

then the command:

```
make
```

A terminal window on a Mac with a dark purple background and light green text. The window title is 'tylerclark@Tylers-MacBook-Pro: ~/Desktop/project2\_clark'. The output shows CMake configuration steps: detecting C and CXX compiler features, checking for a working CXX compiler, and generating build files. Then, the user runs 'make', which shows progress bars for building CXX objects and finally linking the executable 'project2'.

```
tylerclark@Tylers-MacBook-Pro: ~/Desktop/project2_clark
chains/XcodeDefault.xctoolchain/usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/tylerclark/Desktop/project2_clark
tylerclark ~/Desktop/project2_clark
λ make
Scanning dependencies of target project2
[ 14%] Building CXX object CMakeFiles/project2.dir/src/module3.cpp.o
[ 28%] Building CXX object CMakeFiles/project2.dir/src/parse.cpp.o
[ 42%] Building CXX object CMakeFiles/project2.dir/src/symboltable.cpp.o
[ 57%] Building CXX object CMakeFiles/project2.dir/src/subexpression.cpp.o
[ 71%] Building CXX object CMakeFiles/project2.dir/src/operand.cpp.o
[ 85%] Building CXX object CMakeFiles/project2.dir/src/variable.cpp.o
[100%] Linking CXX executable project2
[100%] Built target project2
tylerclark ~/Desktop/project2_clark
λ
```

Lastly, the program can be run with the command:

```
./project2
```

```

tylerclark@Tylers-MacBook-Pro: ~/Desktop/project2_clark
olchains/XcodeDefault.xctoolchain/usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/tylerclark/Desktop/project2_clark
tylerclark ~/Desktop/project2_clark
λ make
Scanning dependencies of target project2
[ 14%] Building CXX object CMakeFiles/project2.dir/src/module3.cpp.o
[ 28%] Building CXX object CMakeFiles/project2.dir/src/parse.cpp.o
[ 42%] Building CXX object CMakeFiles/project2.dir/src/symboltable.cpp.o
[ 57%] Building CXX object CMakeFiles/project2.dir/src/subexpression.cpp.o
[ 71%] Building CXX object CMakeFiles/project2.dir/src/operand.cpp.o
[ 85%] Building CXX object CMakeFiles/project2.dir/src/variable.cpp.o
[100%] Linking CXX executable project2
[100%] Built target project2
tylerclark ~/Desktop/project2_clark
λ ./project2
((x:y?z)>y),x=2,y=3,z=1; Value = 0
(((x+3)<6)&((y+2)=5)),x=2,y=3; Value = 1
(((x*3)=6)|(((y-2)>5)!)),x=2,y=10; Value = 1
((x+3)/6),x=21; Value = 4
tylerclark ~/Desktop/project2_clark
λ

```

## Test Cases

The following test cases will cover all of the operators included in the program. Since the file can read multiple inputs, all of the expressions to be evaluated will be in a single file `"input.txt"`.

### Test case #1

`((x:y?z)>y),x=2,y=3,z=1;`

with this expression we can check off two operations:

- ☐ and
- ☒ conditional
- ☐ divide
- ☐ equal to
- ☒ greater than
- ☐ less than
- ☐ minus
- ☐ negate
- ☐ or
- ☐ plus
- ☐ times

The value return is the correct value of 0 (false), so it passes.

### Test case #2

```
((x+3)<6)&((y+2)=5),x=2,y=3;
```

This expression can check off even more operators:

- ☒ and
- ☒ conditional
- ☐ divide
- ☒ equal to
- ☒ greater than
- ☒ less than
- ☐ minus
- ☐ negate
- ☐ or
- ☒ plus
- ☐ times

The value return is the correct value of 1 since both sides of the & operator evaluate to true, so it passes.

### Test case #3

```
((x*3)=6)|((y-2)>5)!),x=2,y=10;
```

This expression can check off three more operators:

- ☒ and
- ☒ conditional
- ☐ divide
- ☒ equal to
- ☒ greater than
- ☒ less than
- ☒ minus
- ☒ negate
- ☒ or
- ☒ plus
- ☒ times

Because the left side of the or operator is true, we get the correct return value of 1. The left side is made false by negate operation.

### Test case #4

```
((x+3)/6),x=21;
```

This expression will take care of the last operator, divide:

- ☒ and
- ☒ conditional
- ☒ divide
- ☒ equal to
- ☒ greater than



- ☒ less than
- ☒ minus
- ☒ negate
- ☒ or
- ☒ plus
- ☒ times

After evaluating, we see we get the correct value of 4, so this last test passes as well.

## Lessons learned / Conclusion

Through this program, I learned a lot when it came to multi-file C++ projects and a bit on file-handling with C++. The separation of specification and implementation makes much more sense after completing this project and having multiple header files in use. The project instructions stated to assume all input will be correct, yet I did spend some time between IDE and command-line trying to make sure the input file would be read. Perhaps it was due to me being a novice at CMake, but I had to include the line in module3.cpp:

```
std::ifstream file("../input.txt"); //works in IDE
if (!file)
{
    file = std::ifstream("input.txt"); //works in command-line
}
```

To ensure the file would be read with the ide or command-line. This had to do with where the output executable would be placed. Each time I ran the project in CLion IDE, it created a folder for the CMake output.

Overall, I learned a great deal and enjoyed the project! I look forward to more opportunities to use C++!