

Day7 Circuit

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Circuit Class Reference	5
3.1.1 Constructor & Destructor Documentation	5
3.1.1.1 Circuit()	5
3.1.2 Member Function Documentation	5
3.1.2.1 assembleCircuit()	5
3.1.2.2 evaluate()	6
3.1.2.3 getSignal()	6
3.2 Parser::Instruction Struct Reference	6
3.2.1 Member Data Documentation	6
3.2.1.1 input1	6
3.2.1.2 input2	6
3.2.1.3 op	7
3.2.1.4 outputWire	7
3.3 Parser Class Reference	7
3.3.1 Member Enumeration Documentation	7
3.3.1.1 Operation	7
3.3.2 Member Function Documentation	8
3.3.2.1 parseInstruction()	8
3.3.2.2 trim()	8
4 File Documentation	9
4.1 day7.cpp File Reference	9
4.1.1 Function Documentation	9
4.1.1.1 main()	9
Index	11

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Circuit	5
Parser::Instruction	6
Parser	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

day7.cpp	9
------------------------------------	---

Chapter 3

Class Documentation

3.1 Circuit Class Reference

Public Member Functions

- [Circuit](#) ()=default
- void [assembleCircuit](#) (const vector< string > &instructions)
Assembles the circuit from a list of instruction strings.
- uint16_t [evaluate](#) (const string &wire)
Evaluates the signal on a given wire.
- uint16_t [getSignal](#) (const string &wire)
Gets the signal on a specified wire.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Circuit()

```
Circuit::Circuit ( ) [default]
```

3.1.2 Member Function Documentation

3.1.2.1 assembleCircuit()

```
void Circuit::assembleCircuit (
    const vector< string > & instructions ) [inline]
```

Assembles the circuit from a list of instruction strings.

Parameters

<i>instructions</i>	Vector of instruction strings
---------------------	-------------------------------

Each instruction is parsed and stored in the circuitKit map with the output wire as the key.

3.1.2.2 evaluate()

```
uint16_t Circuit::evaluate (
    const string & wire ) [inline]
```

Evaluates the signal on a given wire.

Parameters

<i>wire</i>	The wire identifier
-------------	---------------------

Returns

The signal value as uint16_t

This function recursively evaluates the signal on the specified wire, using memoization to cache previously computed values.

3.1.2.3 getSignal()

```
uint16_t Circuit::getSignal (
    const string & wire ) [inline]
```

Gets the signal on a specified wire.

The documentation for this class was generated from the following file:

- [day7.cpp](#)

3.2 Parser::Instruction Struct Reference

Public Attributes

- [Operation op](#)
- string [input1](#)
- string [input2](#)
- string [outputWire](#)

3.2.1 Member Data Documentation

3.2.1.1 input1

```
string Parser::Instruction::input1
```

3.2.1.2 input2

```
string Parser::Instruction::input2
```

3.2.1.3 op

`Operation` `Parser::Instruction::op`

3.2.1.4 outputWire

`string` `Parser::Instruction::outputWire`

The documentation for this struct was generated from the following file:

- [day7.cpp](#)

3.3 Parser Class Reference

Classes

- struct [Instruction](#)

Public Types

- enum class [Operation](#) {
 [AND](#) , [OR](#) , [LSHIFT](#) , [RSHIFT](#) ,
 [NOT](#) , [ASSIGN](#) , [UNDEFINED](#) = 6 }

Public Member Functions

- [Instruction](#) [parseInstruction](#) (const string &line)
Parses a single instruction line into an [Instruction](#) struct.

Static Public Member Functions

- static string [trim](#) (const string &str)

3.3.1 Member Enumeration Documentation

3.3.1.1 Operation

enum class `Parser::Operation` [strong]

Enumerator

AND	
OR	
LSHIFT	
RSHIFT	
NOT	
ASSIGN	
UNDEFINED	

3.3.2 Member Function Documentation

3.3.2.1 `parseInstruction()`

```
Instruction Parser::parseInstruction (
    const string & line ) [inline]
```

Parses a single instruction line into an [Instruction](#) struct.

Parameters

<i>line</i>	The instruction line as a string
-------------	----------------------------------

Note

Supports operations: AND, OR, LSHIFT, RSHIFT, NOT, ASSIGN
removes leading/trailing whitespace from inputs and outputs to avoid bugs due to formatting issues.

Returns

Parsed [Instruction](#) struct

3.3.2.2 `trim()`

```
static string Parser::trim (
    const string & str ) [inline], [static]
```

The documentation for this class was generated from the following file:

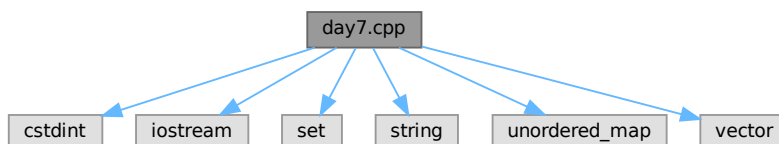
- [day7.cpp](#)

Chapter 4

File Documentation

4.1 day7.cpp File Reference

```
#include <cstdint>
#include <iostream>
#include <set>
#include <string>
#include <unordered_map>
#include <vector>
Include dependency graph for day7.cpp:
```



Classes

- class [Parser](#)
- struct [Parser::Instruction](#)
- class [Circuit](#)

Functions

- int [main](#) ()

4.1.1 Function Documentation

4.1.1.1 main()

```
int main ( )
```


Index

AND
 Parser, [7](#)
assembleCircuit
 Circuit, [5](#)
ASSIGN
 Parser, [7](#)

Circuit, [5](#)
 assembleCircuit, [5](#)
 Circuit, [5](#)
 evaluate, [5](#)
 getSignal, [6](#)

day7.cpp, [9](#)
 main, [9](#)

evaluate
 Circuit, [5](#)

getSignal
 Circuit, [6](#)

input1
 Parser::Instruction, [6](#)
input2
 Parser::Instruction, [6](#)

LSHIFT
 Parser, [7](#)

main
 day7.cpp, [9](#)

NOT
 Parser, [7](#)

op
 Parser::Instruction, [6](#)
Operation
 Parser, [7](#)
OR
 Parser, [7](#)
outputWire
 Parser::Instruction, [7](#)

parseInstruction
 Parser, [8](#)
Parser, [7](#)
 AND, [7](#)
 ASSIGN, [7](#)
 LSHIFT, [7](#)

 NOT, [7](#)
 Operation, [7](#)
 OR, [7](#)
 parseInstruction, [8](#)
 RSHIFT, [7](#)
 trim, [8](#)
 UNDEFINED, [7](#)
Parser::Instruction, [6](#)
 input1, [6](#)
 input2, [6](#)
 op, [6](#)
 outputWire, [7](#)

RSHIFT
 Parser, [7](#)

trim
 Parser, [8](#)

UNDEFINED
 Parser, [7](#)