Institute of Technology, Carlow
B.Sc. in Software Engineering

CW228

User Manual

C Maintenance Tool

Name: Anna-Christina Friedrich

ID: C00132716

Supervisor: Dr. Christophe Meudec

Submission Date: 16.04.2010

Table of Contents

1. Introduction

Congratulations on becoming an user of CMT – C Maintenance Tool!

CMT is a tool that analyses C source files and generates text or html files regarding to all identifiers and their occurrences. It helps to understand and maintain C source code and is useful for Software Engineers, Software Developers, Students of Software Engineering and for everybody else interested in alleviating getting into unknown C code.

If you like to have a simple text file which lists all variable identifiers and all function identifiers you can use the text file creation facility. CMT provides html generation as well. It is an interactive file displaying the C source code and it provides a drop down menu for each identifier to highlight the next or previous occurrence. Furthermore it is possible to select a certain identifier and replace it by a new name. The special thing about this facility is that the scope of the identifier is considered. Assume any development environment and you like to rename an identifier: usually each occurrence in the whole file is replaced without regard to its scope. CMT is concerned about renaming only those occurrences of an identifier that belong to the specified declaration. Another function of CMT is actually implemented to test the tools output, but maybe it is useful for you as well: there is the facility to rename all identifiers automatically.

2. Requirements

CMT is delivered as an executable jar file. To run the tool you need to install the JRE – Java Runtime Environment – if you haven't already installed it. The JRE can be downloaded from the Sun Microsystems web page http://java.sun.com/javase/downloads/index.jsp.

Furthermore you need an editor to display text files and a browser to display the html files. If you are using Microsoft Internet Explorer make sure that you allow active content respectively Scripts.

In order to get maximum efficiency of the tool put your self written header files and their definition files into the directory of your main file, but at most 2 folders prior to the folder where your main file is situated. Otherwise the files cannot be found.

If you create html files an additional file “prototype.js” is copied into the same folder. This file is needed to enable highlighting of identifiers. Hence if you want to move the html file you have to move “prototype.js” as well. Only one prototype file is needed for a folder.

3. Limitations

First of all it is important to ensure the C file you like to reference has ANSI C standard, because CMT works merely on ANSI C files. If there is any expression that cannot be recognized it is displayed on the command prompt like:

```
line 16:7 no viable alternative at input  ' ”ABC” '
```

That means the string “ABC” in line 16, column 7, cannot be recognized.

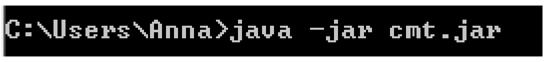
Such an error message need not mean that the files CMT has created are wrong. Sometimes it does not affect the output. To make sure it is working fine for your file, use the renameAll argument to create a modified C file. Then compile both and compare the output whether it is identical.

There is still a problem with referencing structures, which leads unfortunately to the fact that you cannot execute renaming on structures and their attributes. In this case text and html files are not correct as well.

4. How it Works

4.1 Command Line Syntax

Now as the environment is set up open your command prompt, go to the directory where cmt.jar is situated. To run the tool type in: java -jar cmt.jar



```
C:\Users\Anna>java -jar cmt.jar
```

Now the correct syntax is displayed. You can use the following syntax explanation as well.

Arguments are passed with the jar file, therefore you have to type 'java -jar cmt.jar' whenever you want to use a new command. The output files will be located in the same folders as the original files.

Specifying the C source file

Whenever you run the tool you have to specify the path to the C file, that is to be referenced, by:

src 'path to c file'

Make sure that the path always follows the expression 'src'.

Creating a text file

In addition to the previous expression type in:

createText

The order does not matter.

Creating Html output

In addition to specifying the source file type in:

createHtml

The order does not matter as well.

Renaming all identifiers

You can rename all identifiers automatically by:

```
renameAll
```

Again, the order does not matter but you have to name the C source file.

Renaming one identifier

For renaming one identifier ensure to specify the source file and you have to specify more attributes:

```
rename
id 'name of identifier'
newid 'new name of identifier'
scope 'number of scope where identifier is declared'
```

First create a text file so that you can see the scope of the identifier declaration. The order of the attributes does not matter as well, but you have to make sure that an attribute is always followed by its value.

You can combine those properties as you like. See the following example for further comprehension.

Getting help

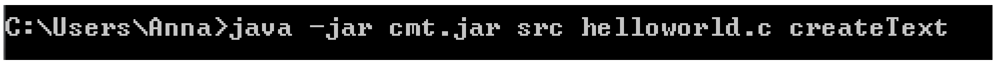
To see the syntax just type:

```
info
```

4.2 Syntax Example

Assuming the C source file helloworld.c which includes the library mylib.h. We also suppose that helloworld.c is in the same folder as cmt.jar: 'C:\Users\Anna'. If not, you have to give the full path to the C file.

Creating a text file



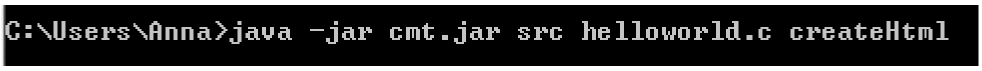
```
C:\Users\Anna>java -jar cmt.jar src helloworld.c createText
```

output files: filename + suffix + textfile suffix

here: helloworldc.txt

mylibh.txt

Creating Html output



```
C:\Users\Anna>java -jar cmt.jar src helloworld.c createHtml
```

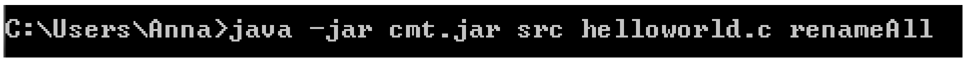
output files: filename + suffix + html suffix

here: helloworld.c.html

mylib.h.html

Use Mozilla Firefox to behold html files.

Renaming all identifiers



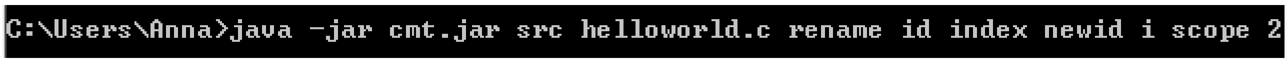
```
C:\Users\Anna>java -jar cmt.jar src helloworld.c renameAll
```

output files: filename + Modified + file suffix

here: helloworldModified.c

mylibModified.h

Renaming one identifier

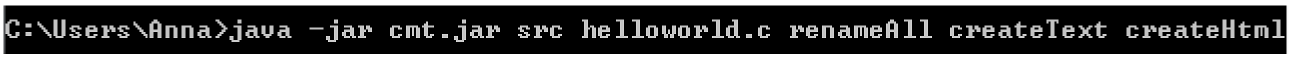


```
C:\Users\Anna>java -jar cmt.jar src helloworld.c rename id index newid i scope 2
```

The identifier 'index' is now called 'i'. You can see the scope number in the created textfile. Output files are the same as renaming all identifiers.

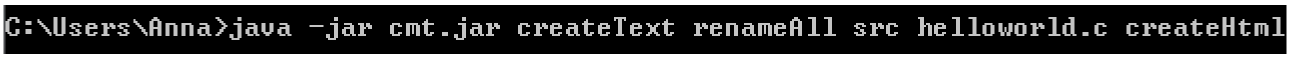
Combining properties

Supposing we like to have a textfile html files and rename all identifiers:



```
C:\Users\Anna>java -jar cmt.jar src helloworld.c renameAll createText createHtml
```

As mentioned above the order doesn't matter, so that following is possible as well:



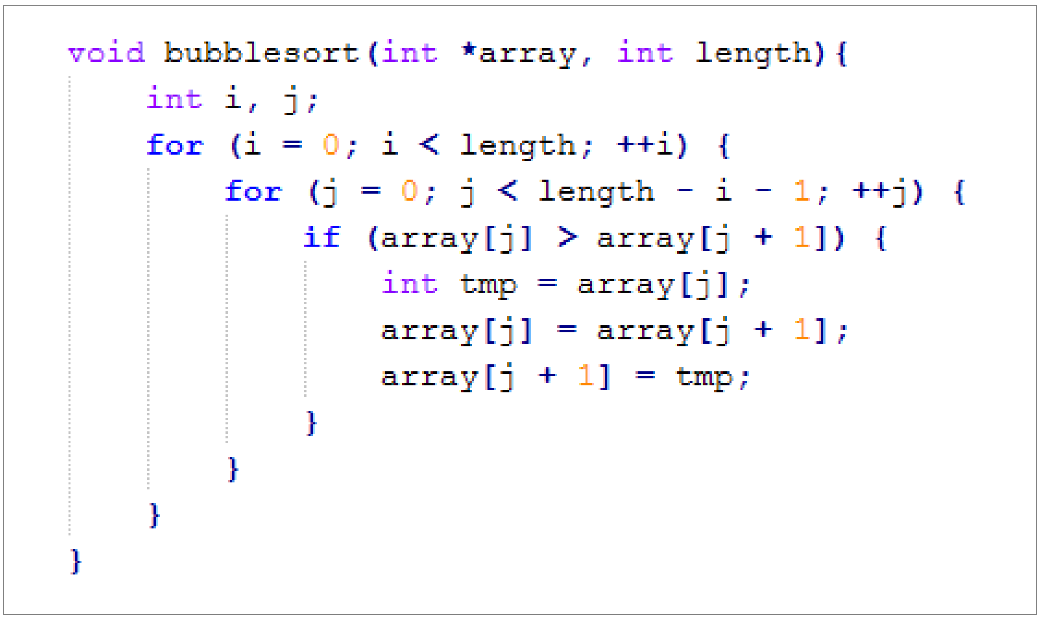
```
C:\Users\Anna>java -jar cmt.jar createText renameAll src helloworld.c createHtml
```

You see that lots of compositions are possible.

5. Output

That is how the output files look like for a simple C file.

5.1 Exemplary C File: bubble.c



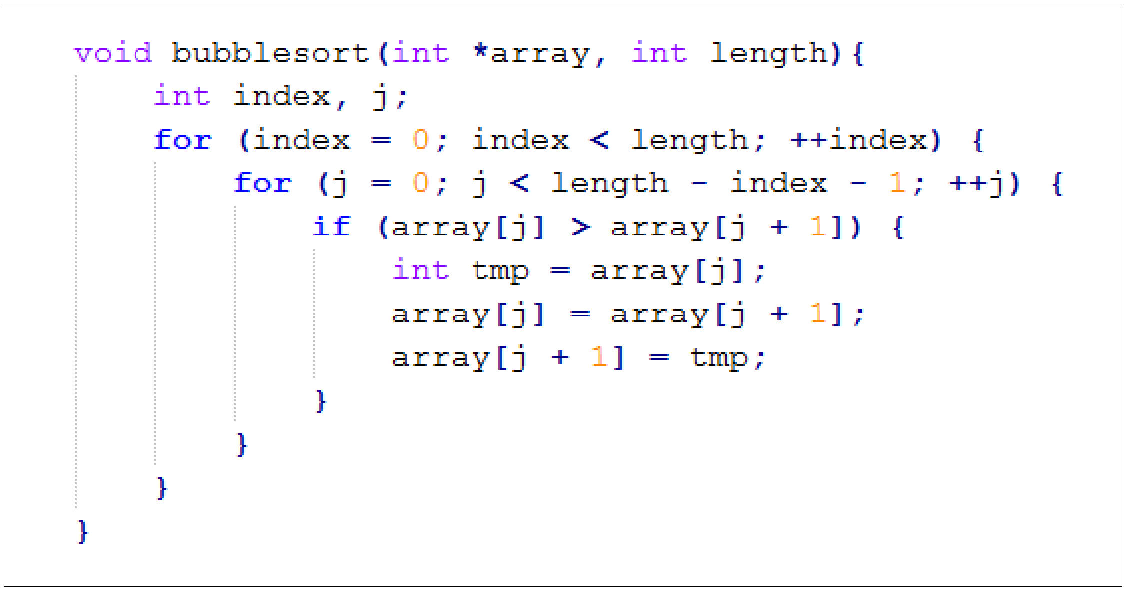
```c
void bubblesort(int *array, int length){
    int i, j;
    for (i = 0; i < length; ++i) {
        for (j = 0; j < length - i - 1; ++j) {
            if (array[j] > array[j + 1]) {
                int tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
            }
        }
    }
}
```

5.2 Text File Sample Output : bubblec.txt

```
VARIABLE LIST
[Identifier  array
scope=1, line=2, source=bubble.c,
Occurrences=
[Occurrence array : scope=4, line=6, source=bubble.c
, Occurrence array : scope=4, line=6, source=bubble.c
, Occurrence array : scope=5, line=7, source=bubble.c
, Occurrence array : scope=5, line=8, source=bubble.c
, Occurrence array : scope=5, line=8, source=bubble.c
, Occurrence array : scope=5, line=9, source=bubble.c
]

, Identifier  length
scope=1, line=2, source=bubble.c,
Occurrences=
[Occurrence length : scope=2, line=4, source=bubble.c
, Occurrence length : scope=3, line=5, source=bubble.c
]

, Identifier  i
scope=2, line=3, source=bubble.c,
Occurrences=
[Occurrence i : scope=2, line=4, source=bubble.c
, Occurrence i : scope=2, line=4, source=bubble.c
, Occurrence i : scope=2, line=4, source=bubble.c
, Occurrence i : scope=3, line=5, source=bubble.c
]

, Identifier  j
scope=2, line=3, source=bubble.c,
Occurrences=
[Occurrence j : scope=3, line=5, source=bubble.c
, Occurrence j : scope=3, line=5, source=bubble.c
, Occurrence j : scope=3, line=5, source=bubble.c
, Occurrence j : scope=4, line=6, source=bubble.c
, Occurrence j : scope=4, line=6, source=bubble.c
, Occurrence j : scope=5, line=7, source=bubble.c
, Occurrence j : scope=5, line=8, source=bubble.c
, Occurrence j : scope=5, line=8, source=bubble.c
, Occurrence j : scope=5, line=9, source=bubble.c
]

, Identifier  tmp
scope=5, line=7, source=bubble.c,
Occurrences=
[Occurrence tmp : scope=5, line=9, source=bubble.c
]

]
FUNCTION LIST
[Identifier  bubblesort
scope=0, declaration line=2, definition line=0, source=bubble.c,
occurrences=
[]
```

5.3 Sample of the Renaming: bubbleModified.c

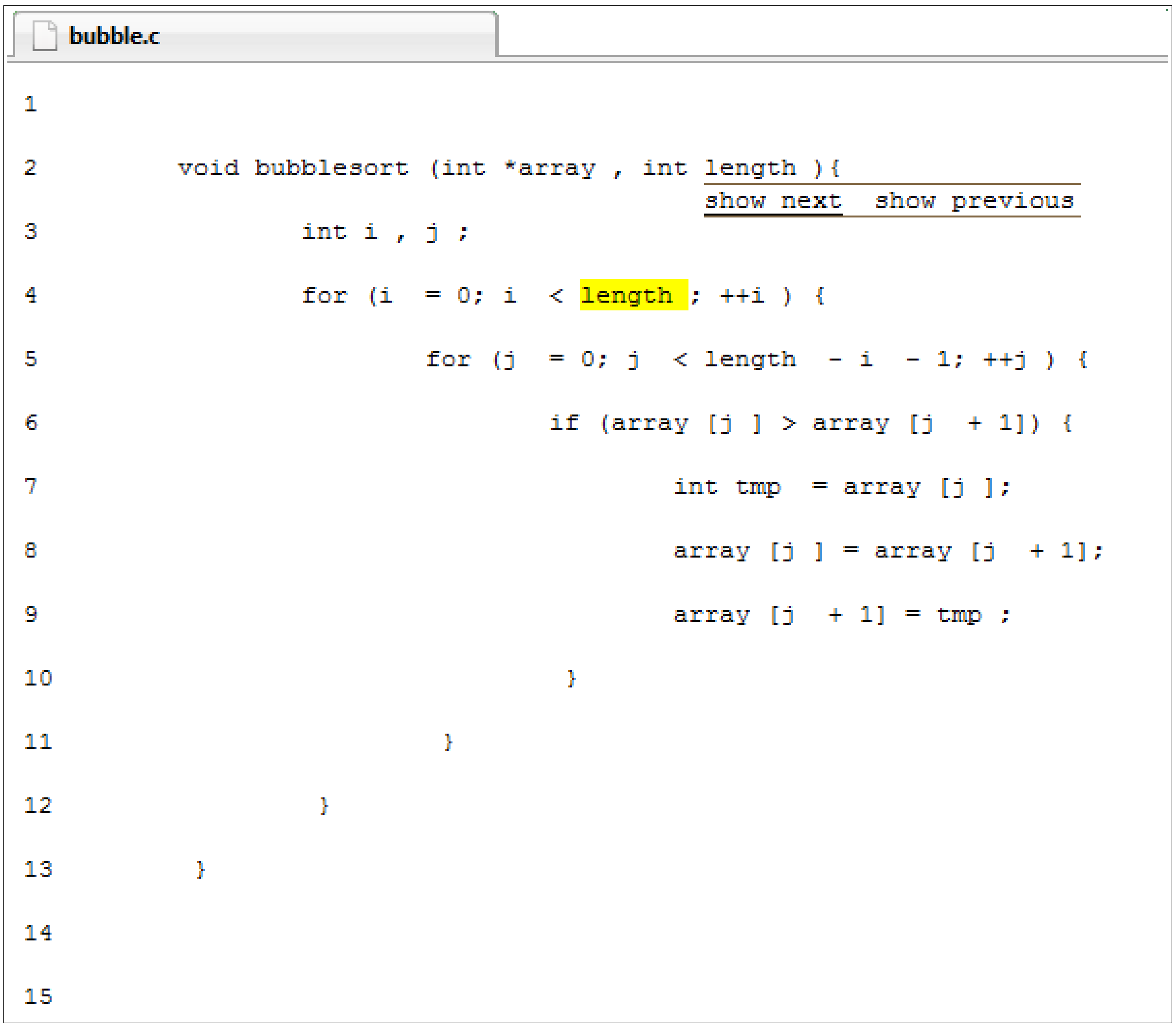
We renamed identifier 'i' to 'index'.



```
void bubblesort(int *array, int length){
    int index, j;
    for (index = 0; index < length; ++index) {
        for (j = 0; j < length - index - 1; ++j) {
            if (array[j] > array[j + 1]) {
                int tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
            }
        }
    }
}
```

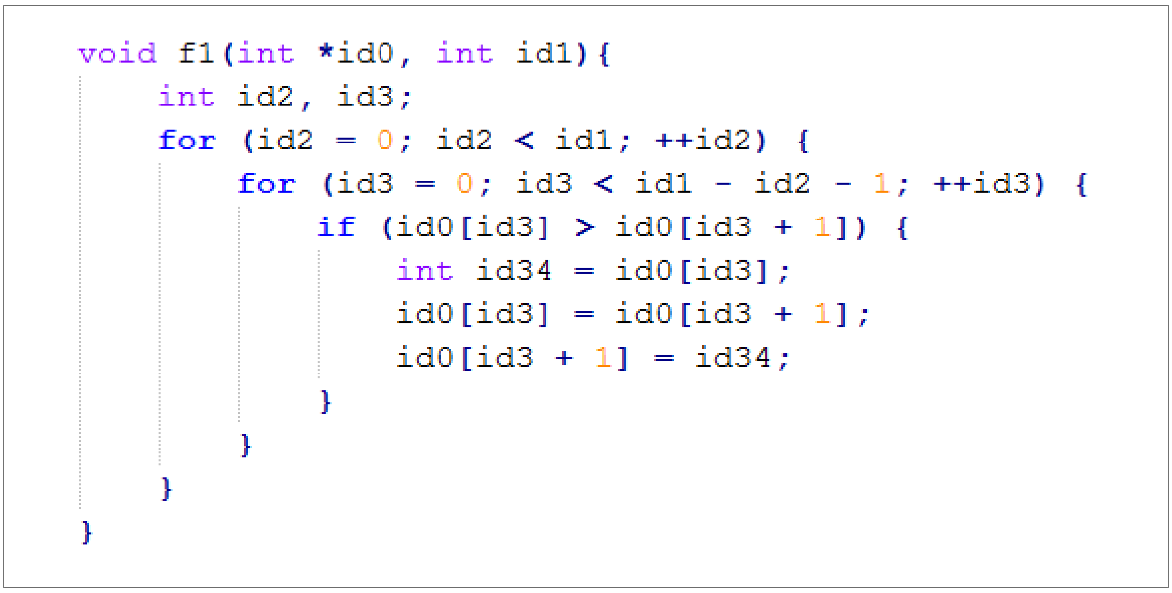
5.4 Screenshot of the Html Result: bubble.c.html

By hovering over 'length' in line 2 the menu drops down. Clicking on 'show next' highlights the next occurrence of 'length' in line 4.



```
1
2        void bubblesort (int *array , int length ){
3                int i , j ;
4                for (i  = 0; i  < length ; ++i ) {
5                        for (j  = 0; j  < length  - i  - 1; ++j ) {
6                                if (array [j ] > array [j  + 1]) {
7                                        int tmp  = array [j ];
8                                        array [j ] = array [j  + 1];
9                                        array [j  + 1] = tmp ;
10                                }
11                        }
12                }
13        }
14
15
```

5.5 Sample of Renaming All Identifiers



```
void f1(int *id0, int id1){
    int id2, id3;
    for (id2 = 0; id2 < id1; ++id2) {
        for (id3 = 0; id3 < id1 - id2 - 1; ++id3) {
            if (id0[id3] > id0[id3 + 1]) {
                int id34 = id0[id3];
                id0[id3] = id0[id3 + 1];
                id0[id3 + 1] = id34;
            }
        }
    }
}
```