THODORIS MAXIMILIANOS CHYTIS
sdi1700197

Askhsh1:

Seperate Chaining:

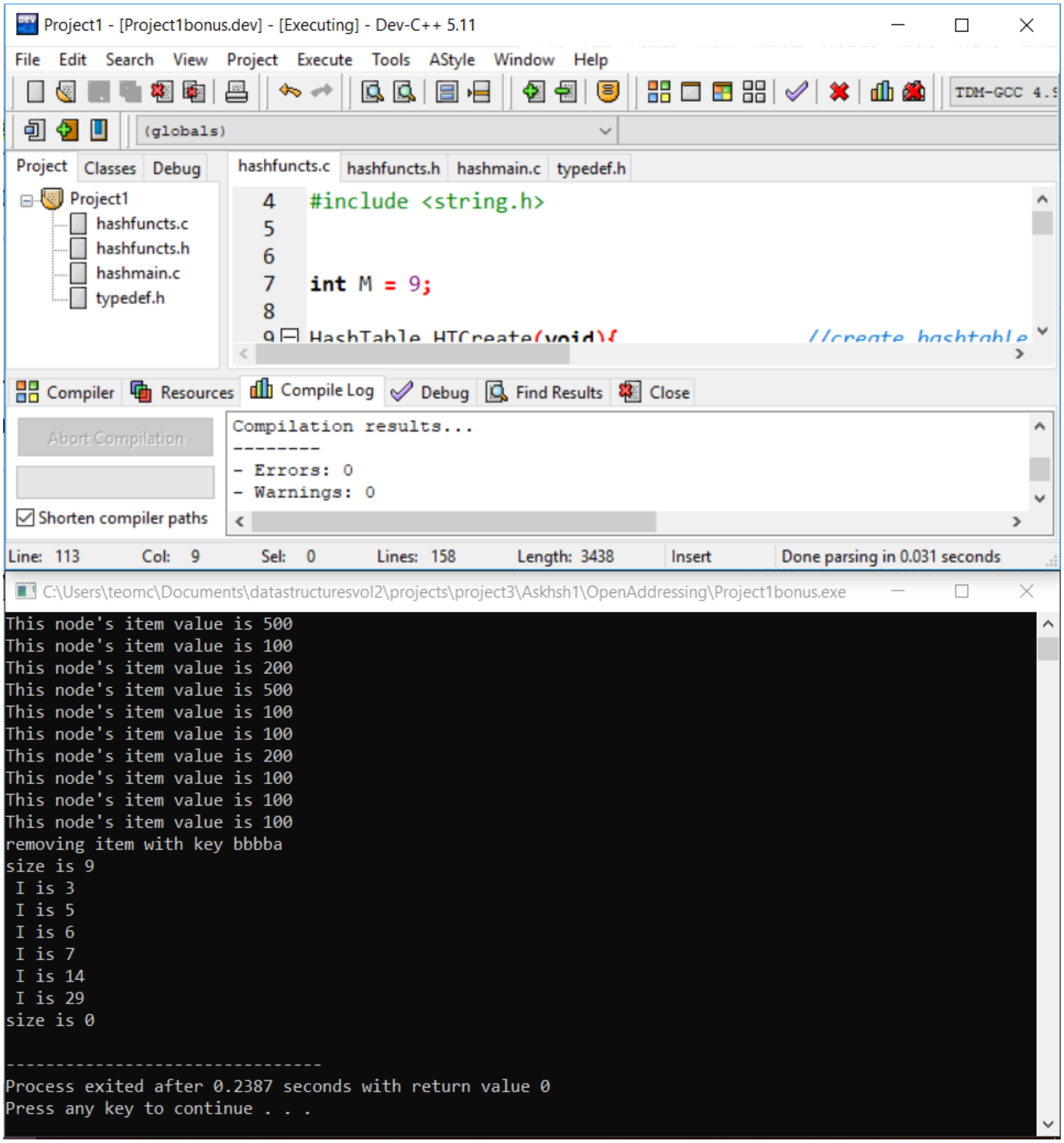
I create a HashTable with the seperate chaining method and succesfully implement all the wanted functions. As you can see in the typdef.h file a have a pointer named HashTable pointing to a pointer named HTNode pointing to a datastructure named HTHash which is the data type I work with. Inside the HTHash structure there is a char * key which is the Key of the Hash data, an item of type Htitem which is int and a pointer to the next Hash data (because of seperate chaining).

```
To compile and run : make
                     ./runporgram
                     make clean
```

OpenAddressing:

This is the bonus exercise I did. The same exact functions are implemented as before. This time around as you will see there is no pointer inside the data structure of HTHash because the open addressing method does not use lists.I was running this programm in DEVC++ without any error.

```
To compile and run : make
                     ./runporgram
                     make clean
```


Project1 - [Project1bonus.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9
(globals)
Project Classes Debug
hashfuncts.c hashfuncts.h hashmain.c typedef.h
Project1
hashfuncts.c
hashfuncts.h
hashmain.c
typedef.h
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 113 Col: 9 Sel: 0 Lines: 158 Length: 3438 Insert Done parsing in 0.031 seconds
C:\Users\teomc\Documents\datastructuresvol2\projects\project3\Askhsh1\OpenAddressing\Project1bonus.exe

```c
#include <string.h>


int M = 9;

HashTable HTCreate(void){                          //create hashtable
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

```
This node's item value is 500
This node's item value is 100
This node's item value is 200
This node's item value is 500
This node's item value is 100
This node's item value is 100
This node's item value is 200
This node's item value is 100
This node's item value is 100
This node's item value is 100
removing item with key bbbba
size is 9
 I is 3
 I is 5
 I is 6
 I is 7
 I is 14
 I is 29
size is 0

--------------------------------
Process exited after 0.2387 seconds with return value 0
Press any key to continue . . .
```

<u>Askhsh2 :</u> In this exercise I create a Graph and succesfully manage to implement all the wanted functions. The graph is made generic so that I can use the same data structure again in a function as suitable option to implement Dijstra's Algorithm.As you will see in the typedef file I use the same HashTable as in the first exercise but with a few alterations. This time the data type of the HTHash is Htitem which is a pointer to structure containing an int dist, an int W or a char * prev as well as pointer to a list.This list is the one I had created in the first projects with a few alterations.

```
        To compile and run : make
                             ./runporgram
                             make clean
```

<u>Askhsh3:</u> In this exercise I use the the graph of the second exercise with all its functions as well as some functions from the exercise number 5 from the second project. I also created an function to Create a graph Dictionary of words with the same length. The I use the Similar function to insert the words of the Dictionary with only one different letter between them into a list. Next, I create the edges between the words of the above list with CreateEdges. After that I can just call UGShortestPath and the wanted procedure is completed.

For more detail please look out for comments next to the code after the // signs.