listed

This is an introduction to listed which is a tutorial of a C++ project so that it is very clear for every beginner how to set up a project. You do need some knowledge of the language for this.

What are we going to make?

A very simple program that can load a text document, edit it and save it. There will be two functions, a function that filters and a function that sorts.

Setting up main.cpp

At the beginning of our file we need our libraries. We add `iostream`, and `uforia`.

```
#include <iostream>
#include <getopt.h>
#include “uforia.h”
```

Then we place the next line so that we don't have to enter a `namespace` every time.

```
using namespace std;
using namespace uforia;
```

After this we have to place a number of variables, some of which are flags. A flag is a `bool` and can be false or true.

```
bool flag_file = false, flag_write = false,
      flag_include = false, flag_sort = false;
vector <string> buffer, list;
ofstream write;
string include;
```

Now we need to enter `int main()` along with `getopt`. This allows us to use options in our program. Arguments can be placed in the string that is in the

while loop, if there is a ':' after it, this means that a secondary argument will take place when using the program. In the program itself this will be called optarg.

The following arguments have been put forward in the downward cases:

For the file we are going to work with we have 'f', so when that is activated we set flag_file to true. And we give optarg as an argument in the ifstream variable read.

For the file we are going to write to we have argument 'o'. And when it is activated we set flag_write to true and give optarg as an argument in the ofstream variable write.

For the option that filters lines from the file by an entered word, we use the argument 'i'. Then flag_include is activated and the string include becomes optarg.

For the option that sorts the lines we use the argument 's'. Then flag_sort is activated.

```
int main(int argc, char * argv[]){

     int c;

     while ((c = getopt (argc, argv, "f:o:i:s")) != -1)
    switch (c)
      {
      case 'f':
         flag_read = true;
         list = from(optarg, true);
         break;
      case 'o':
         flag_write = true;
         write.open(optarg);
         break;
      case 'i':
         flag_include = true;
         include = optarg;
         break;
      case 's':
         flag_sort = true;
         break;
      }

     return Ø;
```

```
}
```

Adding if statements

Now that we've got the basics down, let's move on to filling in the if statements.

Above `return Ø` we need to put our first `if` statement, which checks if a file location is given. Because otherwise we can't go on anyway.

The `if(flag_include)` statement checks if there are any lines that match the given word and collects them into a temporary vector called buffer. Then the vector list becomes buffer.

The `if(flag_sort)` statement checks if `flag_sort` is `true` and then sorts the `vector list` alphabetically.

The `if(flag_write)` statement checks if `flag_write` is `true` and then writes all lines to the given file.

Then all lines present in `vector list` are printed on the screen, so you can see what the result is.

```
  if (!flag_file) {
    cout << "No file given (-f)" << endl;
    exit(Ø);
  }

  if (flag_include) {
    for (int i = Ø; i < list.size(); ++i) {
      string line = list[i];
      if (uforia::has(include, line)) {
        buffer.push_back(line);
      }
    }
    list = buffer;
    buffer.clear();
  }

  if (flag_sort) {
    sort(list.begin(), list.end());
  }

  if (flag_write) {
    for (int i = Ø; i < list.size(); ++i) {
      string line = list[i];
      write << line;
```

```
    }
    write.close();
  }

  for (int i = Ø; i < list.size(); ++i) {
    string line = list[i];
    cout << line << endl;
  }
```

Compile and create the files

Now the code is complete and can be tested. Now we need to compile the files with cmake and then create them with make.

```
cmake CMakeLists.txt
make
```

The program can be installed too.

```
sudo make install
```

Testing

The program is now ready for testing. Inside the folder is a file called fruits.txt. With this we can test the program. For that we enter the following line in the terminal.

```
./listed -f fruits.txt -i p -s
```

The result:

```
Apple
Grape
Raspberry
```