```
Script started on 2024-12-05 12:08:06-06:00 [TERM="xterm-256color" TTY="/dev/pts/8'
e_prevost@ares:~/Lab_3_p3$ pwd
/home/students/e_prevost/Lab_3_p3
e_prevost@ares:~/Lab_3_p3$ cat sorting.info
****************

Robert Prevost

CSC 122 W01



sorting lab



a multi-faceted sorting program that works with a wide range of data.



Base Level: Level 2

Total Level: Level 2

****************e_prevost@ares:~/Lab_3_p3$ show-code main.cpp


main.cpp:


#include <iostream>
#include "sort.h"

int main() {
    short shorts[] = {5, 2, 8, 1, 9};
    double doubles[] = {3.14, 1.41, 2.71, 0.58};
    char chars[] = {'z', 'a', 'c', 'b'};
    std::string strings[] = {"zebra", "apple", "cat", "bear"};
    const char* cstrings[] = {"dog", "cat", "bird"};
    Descending desc;

    sort(shorts, 5, ascending<short>);
    sort(doubles, 4, ascending<double>);
    sort(chars, 4, desc);
    sort(strings, 4, ascending<std::string>);
    sort(cstrings, 3, descendingCString);

    for (int i = 0; i < 3; i++) {
        std::cout << shorts[i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 3; i++) {
        std::cout << doubles[i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 3; i++) {
        std::cout << chars[i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 3; i++) {
        std::cout << strings[i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < 3; i++) {
        std::cout << cstrings[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
e_prevost@ares:~/Lab_3_p3$ show-code sort.h


sort.h:


#ifndef SORT_H
#define SORT_H

#include <string>
#include <cstring>

template<typename T>
void swap(T& a, T& b);

template<typename T, typename Compare>
void sort(T arr[], int size, Compare comp);

template<typename T>
bool ascending(T a, T b);

bool ascendingCString(const char* a, const char* b);
bool descendingCString(const char* a, const char* b);

class Descending {
public:
    template<typename T>
    bool operator()(T a, T b) {
        return a < b;
    }
};

template<typename T>
void swap(T& a, T& b) {
    T temp = a;
    a = b;
    b = temp;
}

template<typename T, typename Compare>
void sort(T arr[], int size, Compare comp) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (comp(arr[j], arr[j + 1])) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

template<typename T>
bool ascending(T a, T b) {
    return a > b;
}

#endif
e_prevost@ares:~/Lab_3_p3$ show-code sort.cpp


sort.cpp:


#include "sort.h"

bool ascendingCString(const char* a, const char* b) {
    return strcmp(a, b) > 0;
}

bool descendingCString(const char* a, const char* b) {
    return strcmp(a, b) < 0;
}
e_prevost@ares:~/Lab_3_p3$ CPP main sort
main.cpp***
sort.cpp...


e_prevost@ares:~/Lab_3_p3$ ./main.out
1 2 5
0.58 1.41 2.71
z c b
apple bear cat
dog cat bird
e_prevost@ares:~/Lab_3_p3$ cat sorting.tpq
What things might cause your new sort template to fail to instantiate?



If the type cant be copied (in order to be swapped) or if the type doesnt

support comparison operators.


Where should the overloaded swap form go? (In the library or the main

application?) If in the library, should it be in the header with the swap

template or in the implementation file? Does it matter to the compiler

where it is? Why/Why not?


The overloaded swap form should go in the header file because it needs to

be visible at point of instantiation and this can only be seen in the

header.


Did you need to make any changes to your original swap template?


I wrote this program from scratch but the swap was very straightforward and

reminded me of swap functions ive made before so I would like to say no.


Which comparisons did you write as plain functions? Function objects? Were

any of them templated? Could/Should they have been? (Hint: you should use

at least one plain function and one function object class to show you can

do both.)


ascending, ascendingCstring and descendingCstring are all plain functions

and descending class is a function object. the ascending and descending

function/class were templated to work with multiple types. C string

functions were a little different so they required their own function and

therefore couldnt be tempalted. e_prevost@ares:~/Lab_3_p3$ exit
exit
```

```
Script done on 2024-12-05 12:08:48-06:00 [COMMAND_EXIT_CODE="0"]
```