

A2:T1: Generic quicksort

Programming Languages and Paradigms Seminar

Approach

This program uses c++ templates in order to achieve the generic type behaviour. The implementation of the quicksort is a classic lomuto partitioning. The partition function receives vectors of generic types and uses the `auto` feature of c++ in order to initialise the value of the pivot. This is crucial in order to keep type compatibility.

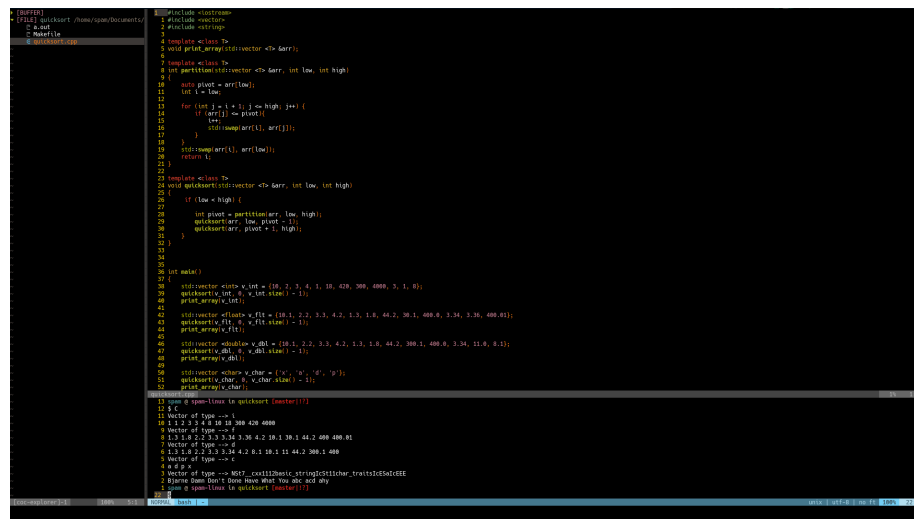
Running the program

Navigate to the directory and run the `make` command. This will produce a binary executable for you to run like `./a.out`.

Deleting the program

You may use the `make clean` command to delete the produced binary.

Screenshot



```
[root@19]
+ [FILE] quicksort /home/ryan/Documents/
+ 2 out
+ Makefile
+ make

1 #include <iostream>
2 #include <vector>
3 #include <string>
4 template <class T>
5 void print_array(const vector<T> &arr);
6
7 template <class T>
8 int partition(vector<T> &arr, int low, int high)
9 {
10     auto pivot = arr[low];
11     int i = low;
12     for (int j = i + 1; j <= high; j++) {
13         if (arr[j] <= pivot) {
14             swap(arr[i], arr[j]);
15             i++;
16         }
17     }
18     swap(arr[i], arr[low]);
19     return i;
20 }
21
22 template <class T>
23 void quicksort(vector<T> &arr, int low, int high)
24 {
25     if (low < high) {
26         int pivot = partition(arr, low, high);
27         quicksort(arr, low, pivot - 1);
28         quicksort(arr, pivot + 1, high);
29     }
30 }
31
32 int main()
33 {
34     vector<int> v_int = {10, 2, 3, 4, 1, 10, 420, 200, 4000, 3, 1, 0};
35     print_array(v_int);
36     vector<float> v_flt = {10.1, 2.2, 3.3, 4.2, 1.5, 1.8, 44.2, 30.1, 400.0, 3.34, 1.30, 400.01};
37     print_array(v_flt);
38     vector<double> v_dbl = {10.1, 2.2, 3.3, 4.2, 1.5, 1.8, 44.2, 30.1, 400.0, 3.34, 1.34, 11.0, 0.1};
39     print_array(v_dbl);
40     vector<char> v_chr = {'a', 'b', 'c', 'd', 'e'};
41     print_array(v_chr);
42     vector<string> v_str = {"a", "b", "c", "d", "e"};
43     print_array(v_str);
44 }
45
46 g++ -std=c++11 -O2 quicksort.cpp -o a.out
47 ./a.out
48 10 2 3 4 1 10 420 200 4000 3 1 0
49 10.1 2.2 3.3 4.2 1.5 1.8 44.2 30.1 400.0 3.34 1.30 400.01
50 10.1 2.2 3.3 4.2 1.5 1.8 44.2 30.1 400.0 3.34 1.34 11.0 0.1
51 a b c d e
52 a b c d e
```

Figure 1: quicksort