```
Script started on 2024-05-05 15:27:58-05:00 [TERM="xterm" TTY="/dev/pts/0" COLUMNS=
ee43254@ares:~$ pwd
/home/students/ee43254
ee43254@ares:~$ cat median.info
Name: Kyle Enkhzul

Class: CSC122-W01



Activity: Smack in the Middle of Nowhere

Level: 2.5, 1.5 (base program), 1 (repeating prompt)



Description:



This lab provides practice with dynamic memory not mixed wth classes. Given

a text file of numbers, we are to calculate the median in the most memory

efficient manner.

ee43254@ares:~$ show-code median.cpp


median.cpp:
```

```cpp
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

// Function to find the median of a set of data
double findMedian(vector<double>& data) {
    size_t numItems = data.size();
    // Bubble sort the data
    for (size_t i = 0; i < numItems - 1; ++i) {
        for (size_t j = 0; j < numItems - i - 1; ++j) {
            if (data[j] > data[j + 1]) {
                // Swap data[j] and data[j+1]
                double temp = data[j];
                data[j] = data[j + 1];
                data[j + 1] = temp;
            }
        }
    }
    // Calculate the median
    if (numItems % 2 == 0) {
        return (data[numItems / 2 - 1] + data[numItems / 2]) / 2.0;
    } else {
        return data[numItems / 2];
    }
}

int main() {
    char choice;
    do {
        // Open the file
        ifstream file;
        string filename;
        cout << "Enter filename: ";
        cin >> filename;
        file.open(filename);
        if (!file) {
            cerr << "Error: Unable to open file!" << endl;
            continue; // Skip to the next iteration of the loop
        }

        // Read data from the file
        vector<double> data;
        double value;
        while (file >> value) {
            data.push_back(value);
        }

        // Close the file
        file.close();

        // Find the median
        double median = findMedian(data);

        // Print the median
        cout << "Median: " << median << endl;

        // Ask user if they want to repeat
        cout << "Do you want to calculate median with another file? (y/n): ";
        cin >> choice;
    } while (choice == 'y' || choice == 'Y');

    return 0;
}
```

```
ee43254@ares:~$ CPP median
median.cpp***


ee43254@ares:~$ cat median1.txt
12

43

23
```

```
40

10

20

46

23

54

2ee43254@ares:~$ cat median2.txt
12

34

45

23

30

70

80

60

50

80

93

47

3ee43254@ares:~$ ./median.out
Enter filename: median100.txt
Error: Unable to open file!
ee43254@ares:~$ ./median.ou
bash: ./median.ou: No such file or directory
ee43254@ares:~$ ./median.out
Enter filename: median1.txt
Median: 23
Do you want to calculate median with another file? (y/n): y
Enter filename: median2.txt
Median: 47
Do you want to calculate median with another file? (y/n): y
Enter filename: median3.txt
Error: Unable to open file!
Enter filename: median1.txt
Median: 23
Do you want to calculate median with another file? (y/n): n
```

```
ee43254@ares:~$ cat median.tpq
1. What (data) types of values can your program handle?

The program can handle any numeric data values present in the input file.

This includes integers, floating-point numbers, and scientific notation.


2. How do you count the number of data items which are in the file?

Once you've counted them, how do you start over and read them in?

The number of data items in the file is counted using the size() method of the

vector. After counting them, the program proceeds to read them in by

iterating over the file until the end is reached (while (file >> value)).


3. How do you allocate memory on the heap? Is it guaranteed to work?

Memory on the heap is allocated implicitly when the vector dynamically grows

to accommodate more data elements. The vector class manages memory allocation

and resizing internally, and it typically allocates more memory than necessary

to prevent frequent reallocations.


4. When do you allocate memory for your values array? Before or after counting

the values in the file?

Memory for the values array (the vector data in this case) is allocated

dynamically when the program reads data from the file. It grows dynamically

as more elements are read from the file.


5. When you are done with dynamic memory, what should you do?

When you're done with dynamic memory, you should release it to prevent memory

leaks. In this code, memory management is handled automatically by the vector

class. If you were manually managing memory using pointers, you would use

delete[] to deallocate memory allocated with new[].
```

6. What's this nullptr value for, anyway?

It is used to represent a null pointer, which indicates that the pointer does not point to any memory location. In this code, nullptr is not explicitly used because memory management is handled by vector

7. Which sorting algorithm did you use? How efficient is it?

(i.e. What is it's complexity?)

I used the bubble sort. It does its job well but it is not the most efficient. I tweaked it a little from the notes that Professor James provided.

8. Does it matter if the data are sorted in ascending or descending order?

Why/Why not?

It does not matter if it is sorted in order because median calculation depends on the middle value. The bubble sort would work equally as well if it was descending since I used ascending order.ee43254@ares:~$ exit
exit

Script done on 2024-05-05 15:29:52-05:00 [COMMAND_EXIT_CODE="0"]