“ In the name of God ”

Amirkabir university of technology
HomeWork 6

Advanced Programming C++
Dr.Jahanshahi

Writer : Sajad Ghadiri
Student Number : 9723067

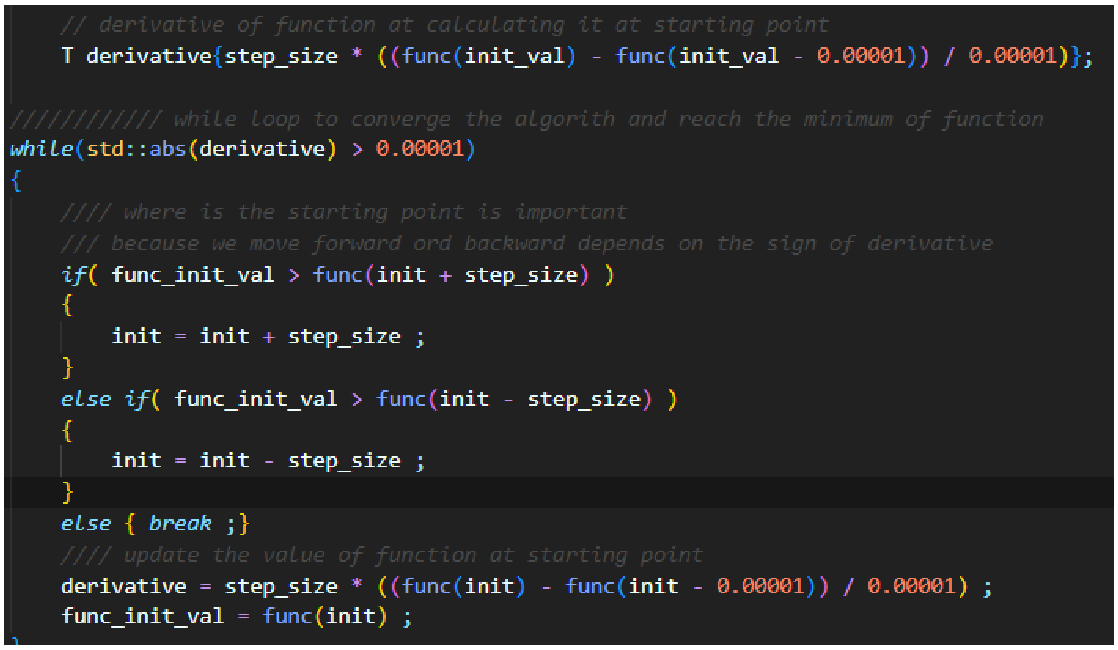
Teaching assistant : Mr K.Behzad

In this homework I Solved 4 Questions with STL approach and implementing some classes according to the structure of the README.md

Q1 :

1.

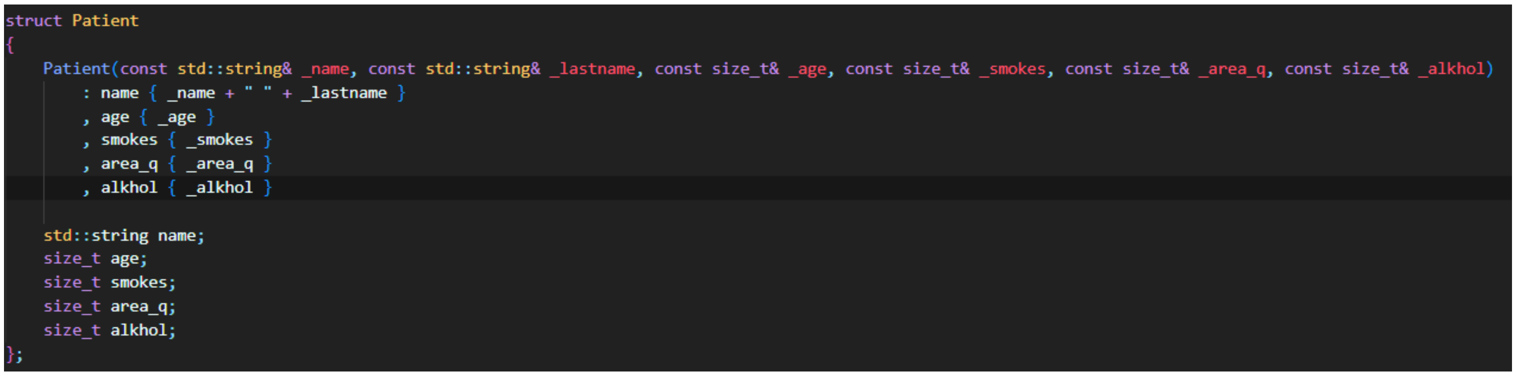
I calculate the derivative term and then I assume 2 situations. If the starting point is lower than the minimum point so we should go forward every step and if the starting point is greater than the minimum point we should go backward. I implement it by if and elseif. Finally I update the derivative term and the y-position of the present point.



```
    // derivative of function at calculating it at starting point
    T derivative{step_size * ((func(init_val) - func(init_val - 0.00001)) / 0.00001)};

//////////// while loop to converge the algorith and reach the minimum of function
while(std::abs(derivative) > 0.00001)
{
    //// where is the starting point is important
    /// because we move forward ord backward depends on the sign of derivative
    if( func_init_val > func(init + step_size) )
    {
        init = init + step_size ;
    }
    else if( func_init_val > func(init - step_size) )
    {
        init = init - step_size ;
    }
    else { break ;}
    //// update the value of function at starting point
    derivative = step_size * ((func(init) - func(init - 0.00001)) / 0.00001) ;
    func_init_val = func(init) ;
```

Q2:

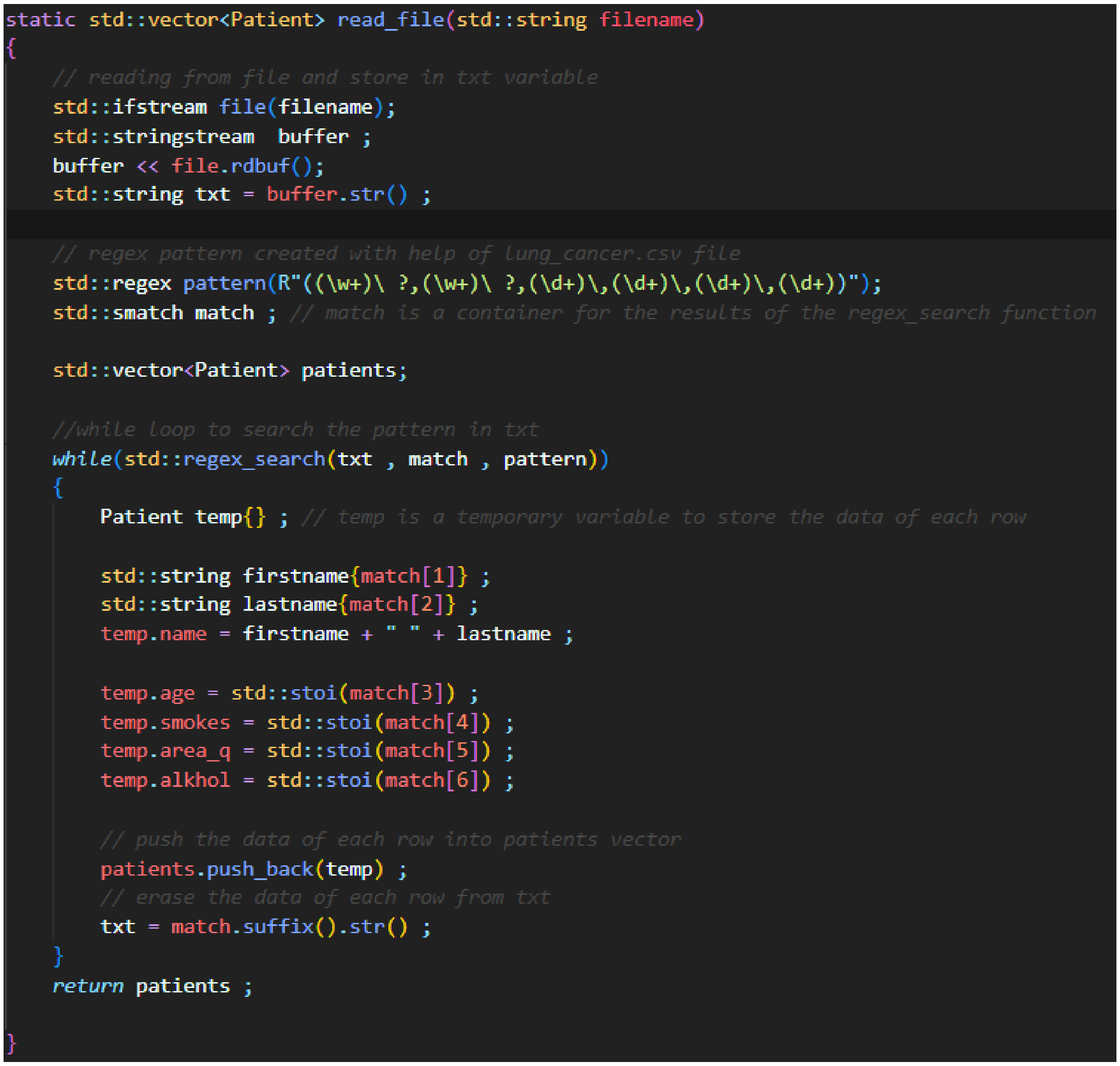
At first I replaced the suggested structure for patient in the readme file and added constructor to it.



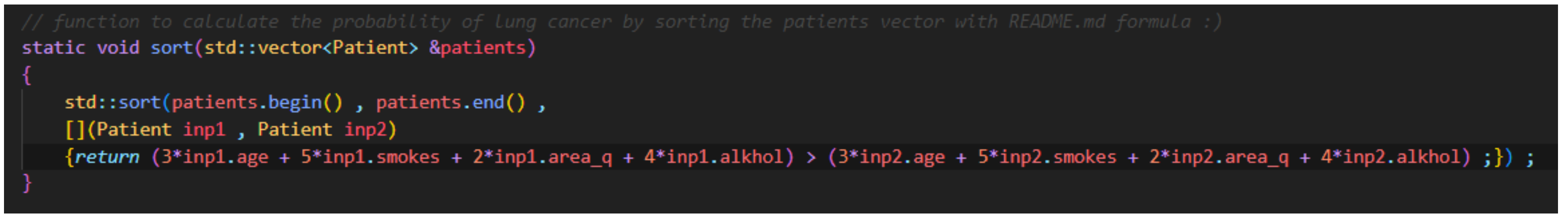
```
struct Patient
{
    Patient(const std::string& _name, const std::string& _lastname, const size_t& _age, const size_t& _smokes, const size_t& _area_q, const size_t& _alkhol)
        : name { _name + " " + _lastname }
        , age { _age }
        , smokes { _smokes }
        , area_q { _area_q }
        , alkhol { _alkhol }

    std::string name;
    size_t age;
    size_t smokes;
    size_t area_q;
    size_t alkhol;
};
```

Then I implemented the read_file function which takes a string filename as its input.
I used regex library and read all file data and assigned in “ txt ” string.
I implemented a pattern to search by regular expression method and then defined match variable(for search loop) and a vector object from patient structure and then in the while loop I assigned every group of my pattern which I declared by () to a variable of temp. At the end I used push_back temp to patient Vector.

* notice that using std::stoi is necessary and at the end of every iteration i used match.suffix().str() to look forward of “ txt “ string which include our main file data.



```
static std::vector<Patient> read_file(std::string filename)
{
    // reading from file and store in txt variable
    std::ifstream file(filename);
    std::stringstream  buffer ;
    buffer << file.rdbuf();
    std::string txt = buffer.str() ;

    // regex pattern created with help of lung_cancer.csv file
    std::regex pattern(R"((\w+)\ ?,(\w+)\ ?,(\d+)\,(\d+)\,(\d+)\,(\d+))");
    std::smatch match ; // match is a container for the results of the regex_search function

    std::vector<Patient> patients;

    //while loop to search the pattern in txt
    while(std::regex_search(txt , match , pattern))
    {
        Patient temp{} ; // temp is a temporary variable to store the data of each row

        std::string firstname{match[1]} ;
        std::string lastname{match[2]} ;
        temp.name = firstname + " " + lastname ;

        temp.age = std::stoi(match[3]) ;
        temp.smokes = std::stoi(match[4]) ;
        temp.area_q = std::stoi(match[5]) ;
        temp.alkhol = std::stoi(match[6]) ;

        // push the data of each row into patients vector
        patients.push_back(temp) ;
        // erase the data of each row from txt
        txt = match.suffix().str() ;
    }
    return patients ;

}
```



```
// function to calculate the probability of lung cancer by sorting the patients vector with README.md formula :)
static void sort(std::vector<Patient> &patients)
{
    std::sort(patients.begin() , patients.end() ,
    [](Patient inp1 , Patient inp2)
    {return (3*inp1.age + 5*inp1.smokes + 2*inp1.area_q + 4*inp1.alkhol) > (3*inp2.age + 5*inp2.smokes + 2*inp2.area_q + 4*inp2.alkhol) ;}) ;
}
```

For the sort function I defined a lambda function like above.
* The most important error in this homework which I faced was " multiple definitions " because all headers file included together and I was not allowed to edit main.cpp and delete some includes or comment them so I was forced to use " STATIC " before my lambda functions and other functions.
You can also use the " INLINE " command but INLINE doesn't redefine.

Q3:

This question approach is the same as Q2 .
Some point i want to mention are below:

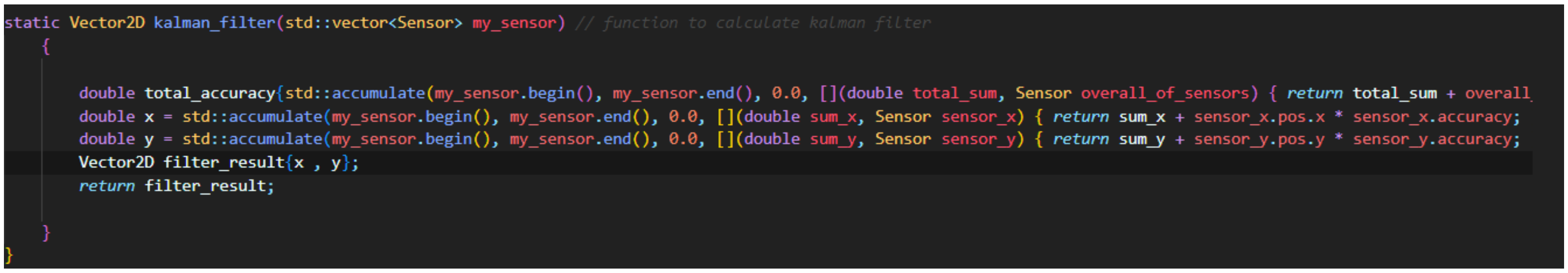
1- i got some insensitive errors which finally i understood that i should use static_cast<size_t > for some " match indexes " that contained std::stoi (match is variable for searching throw our string file)

2-using STATIC before functions and …

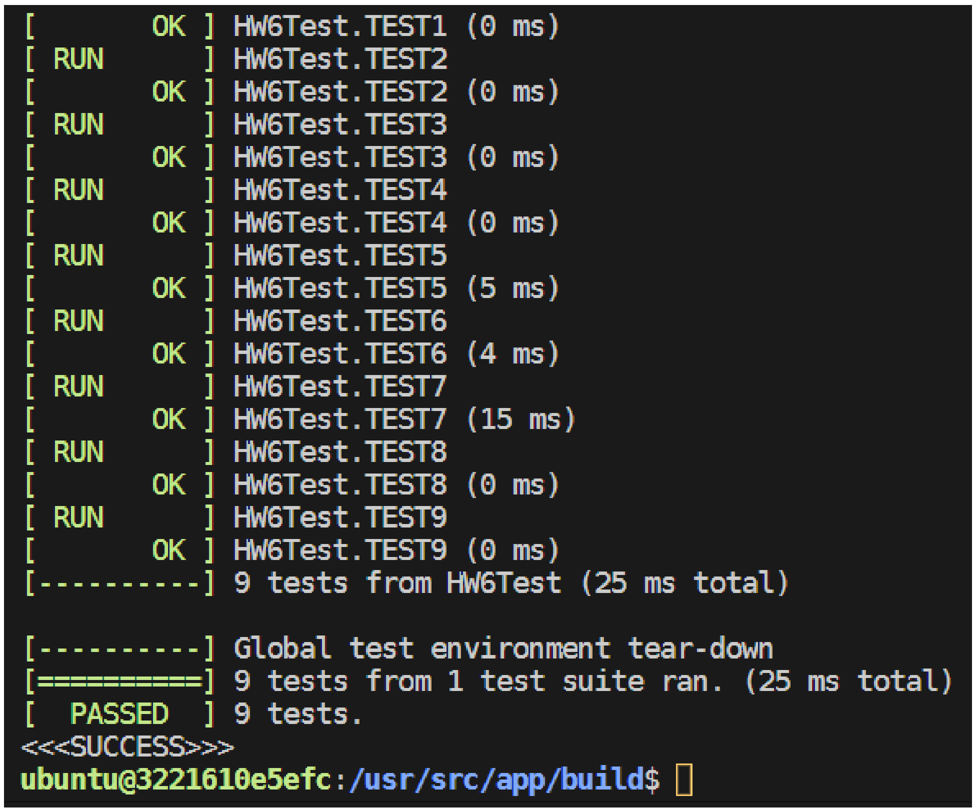
3-defining constructor is important too

Q4:
I replaced the suggested structure for Vector2D and Sensor like readme file and wrote their constructor.

The Kalman_filter function takes a vector of sensor and returns a Vector2D as its output. By using numeric library calculate weighted accuracy of each elements of sensor of input By defining lambda function and then return it in a Vector2D.



```cpp
static Vector2D kalman_filter(std::vector<Sensor> my_sensor) // function to calculate kalman filter
    {

        double total_accuracy{std::accumulate(my_sensor.begin(), my_sensor.end(), 0.0, [](double total_sum, Sensor overall_of_sensors) { return total_sum + overall
        double x = std::accumulate(my_sensor.begin(), my_sensor.end(), 0.0, [](double sum_x, Sensor sensor_x) { return sum_x + sensor_x.pos.x * sensor_x.accuracy;
        double y = std::accumulate(my_sensor.begin(), my_sensor.end(), 0.0, [](double sum_y, Sensor sensor_y) { return sum_y + sensor_y.pos.y * sensor_y.accuracy;
        Vector2D filter_result{x , y};
        return filter_result;

    }
}
```



```
[       OK ] HW6Test.TEST1 (0 ms)
[ RUN      ] HW6Test.TEST2
[       OK ] HW6Test.TEST2 (0 ms)
[ RUN      ] HW6Test.TEST3
[       OK ] HW6Test.TEST3 (0 ms)
[ RUN      ] HW6Test.TEST4
[       OK ] HW6Test.TEST4 (0 ms)
[ RUN      ] HW6Test.TEST5
[       OK ] HW6Test.TEST5 (5 ms)
[ RUN      ] HW6Test.TEST6
[       OK ] HW6Test.TEST6 (4 ms)
[ RUN      ] HW6Test.TEST7
[       OK ] HW6Test.TEST7 (15 ms)
[ RUN      ] HW6Test.TEST8
[       OK ] HW6Test.TEST8 (0 ms)
[ RUN      ] HW6Test.TEST9
[       OK ] HW6Test.TEST9 (0 ms)
[----------] 9 tests from HW6Test (25 ms total)

[----------] Global test environment tear-down
[==========] 9 tests from 1 test suite ran. (25 ms total)
[  PASSED  ] 9 tests.
<<<SUCCESS>>>
ubuntu@3221610e5efc:/usr/src/app/build$ 
```

Final Result :)