```cpp
//Bernard J. Gole Cruz, CS 202-2002, Assignment 5
//This program compare the sorting time of bubble sort and quicksort algorithm
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#include <iomanip>

using namespace std;

const int ARR_SIZE = 100000;
const int ID_MIN = 1000;
const int ID_MAX = 99999;

//class definition
class Student {
    public:
        Student(int sid = 0, double sgpa = 0.0){
            id = sid;
            gpa = sgpa;
        }
        int id;
        double gpa;
};
//display student id and gpa separated by colon
void display(Student array[]){
    for (int index = 0; index < ARR_SIZE; index++){
        cout << array[index].id << ":" << array[index].gpa << " ";
    }
}
//bubble sort
void sortBubble(Student array[]){
    for (int pass = 0; pass < ARR_SIZE; pass++){
        for (int index = 0; index < ARR_SIZE - pass; index++){
            // check if element at index is less than element at
            // index + 1
            if (array[index].id > array[index + 1].id){
                // perform swap
                int temp = array[index].id;
                double tempGpa = array[index].gpa;
                array[index].id = array[index + 1].id;
                array[index].gpa = array[index + 1].gpa;
                array[index + 1].id = temp;
                array[index + 1].gpa = tempGpa;
            }
        }
    }
}
//swap two elements in the array
void swapValues(Student &student1, Student &student2){
    int temp = student1.id;
    double tempGpa = student1.gpa;
    student1.id = student2.id;
    student1.gpa = student2.gpa;
    student2.id = temp;
    student2.gpa = tempGpa;
}

//move element that are less than the pivot to the left
//and move element greater than the pivot to the right
int qPartition(Student array[], int lowIndex, int highIndex){
    int pivotItem = array[highIndex].id;

    // index of the place where element will be placed
    int putIndex = lowIndex - 1;

    for (int currentIndex = lowIndex; currentIndex < highIndex; currentIndex++){
        // if element at currentIndex is less than the pivotItem, move it to left
side
        if (array[currentIndex].id <= pivotItem){
            putIndex++;
            swapValues(array[currentIndex], array[putIndex]);
        }
    }

    // once the loop completes, move the pivotItem to its right place
    swapValues(array[highIndex], array[putIndex + 1]);

    // return the index of pivotItem
    return putIndex + 1;
}

//quick sort
void qsort(Student array[], int lowIndex, int highIndex){
    if (lowIndex < highIndex){
        int partitionIndex = qPartition(array, lowIndex, highIndex);

        // sort the left and right side of partitionIndex
        qsort(array, lowIndex, partitionIndex - 1);
        qsort(array, partitionIndex + 1, highIndex);
    }
}

int main(){
    // create an array of 1M elements
    Student students[ARR_SIZE];
    // fill with random values: id = ID MIN - ID MAX)
    // initialize the seed
    srand(time(0));

    for (int index = 0; index < ARR_SIZE; index++){
        int id = ID_MIN + (rand() % (ID_MAX - ID_MIN + 1));
        double gpa = (rand() % 41) / 10.0;

        students[index] = Student(id, gpa);
    }

    //display unsorted only for smaller inputs
    //display(students);

    cout <<"Choice:" <<endl;
    cout <<"1 = bubblesort" << endl;
    cout <<"2 = quicksort" << endl;

    //choice variable
    int choice;
    cin >> choice;

    //start time and end time variables;
    clock_t start_time, end_time;

    //start measuring time
    start_time = clock();

    // sort by selected algorithm
    if (choice == 1){
        sortBubble(students);
    }
    else if(choice == 2){
        qsort(students, 0, ARR_SIZE - 1);
    }

    //stop measuring time
    end_time = clock() - start_time;

    //display sorted array only for smaller inputs
    //display(students);

    // display table
    int total_seconds = ((double)end_time / CLOCKS_PER_SEC);


    //display the result time
    cout << "Total time taken: " << total_seconds << endl;
}
```