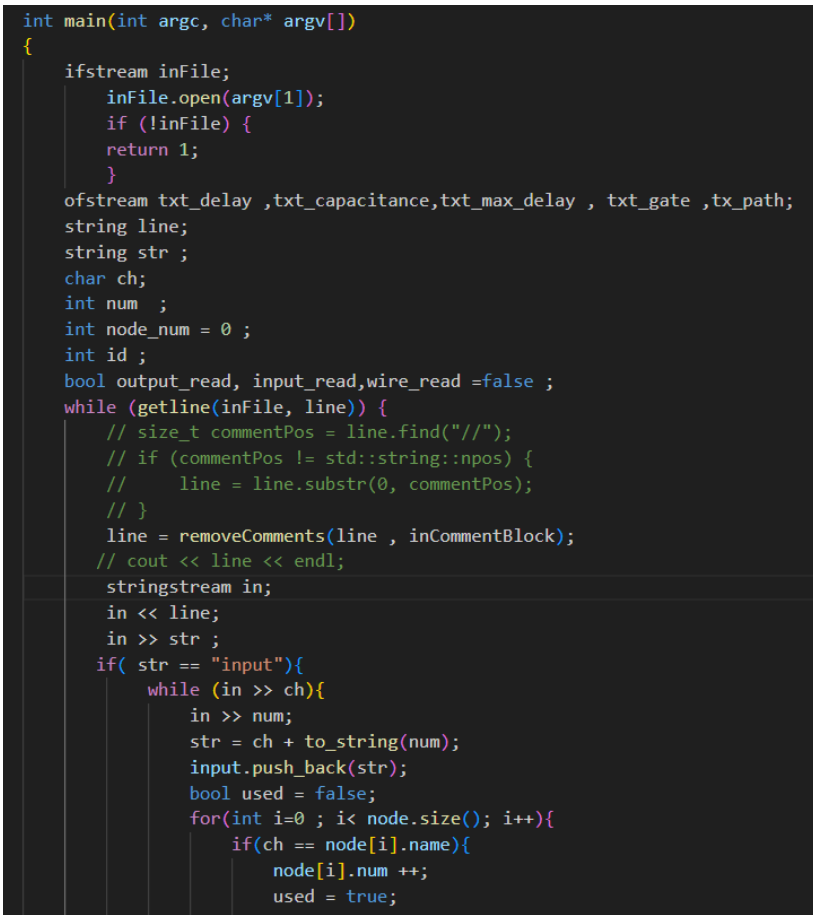
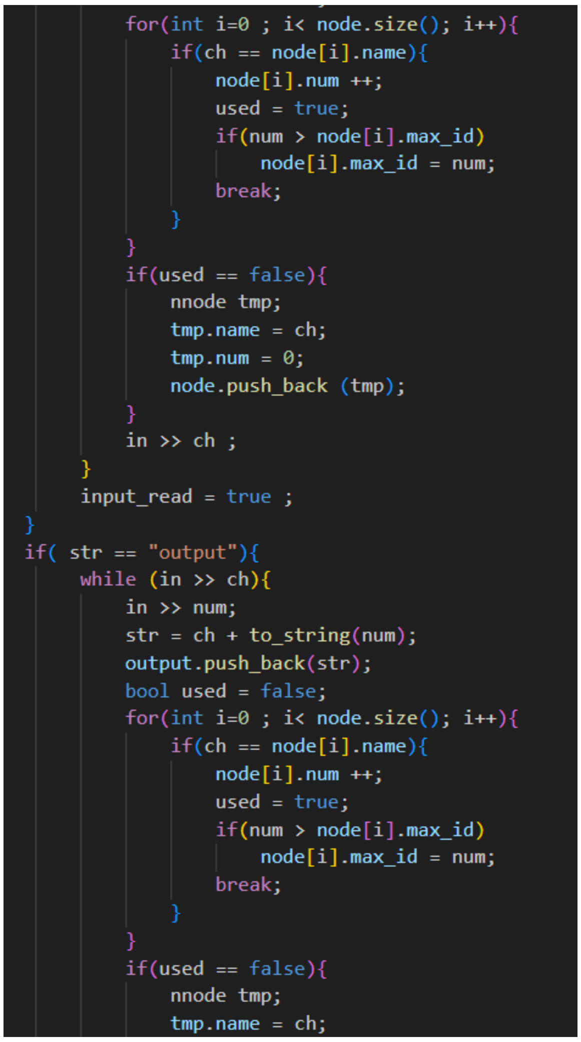
計算機輔助設計特論 HW2 311510207 江尹凡

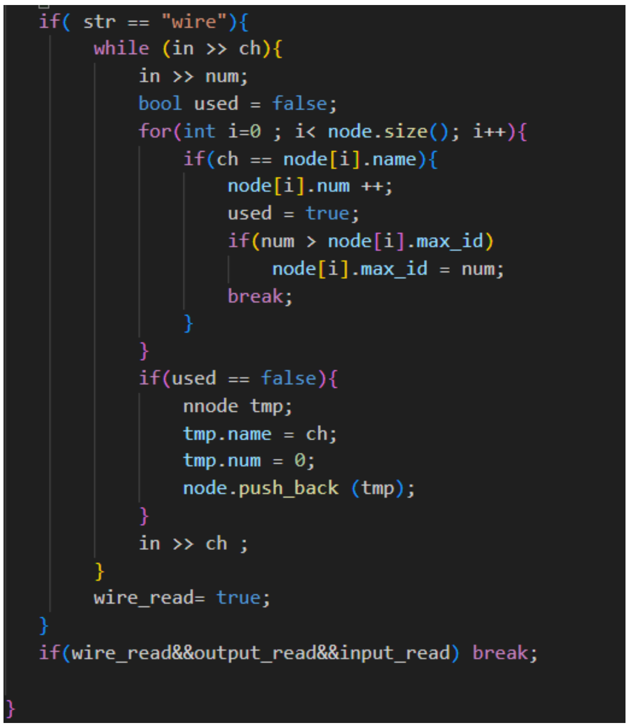
讀檔



```cpp
int main(int argc, char* argv[])
{
    ifstream inFile;
        inFile.open(argv[1]);
        if (!inFile) {
        return 1;
        }
    ofstream txt_delay ,txt_capacitance,txt_max_delay , txt_gate ,tx_path;
    string line;
    string str ;
    char ch;
    int num  ;
    int node_num = 0 ;
    int id ;
    bool output_read, input_read,wire_read =false ;
    while (getline(inFile, line)) {
        // size_t commentPos = line.find("//");
        // if (commentPos != std::string::npos) {
        //     line = line.substr(0, commentPos);
        // }
        line = removeComments(line , inCommentBlock);
       // cout << line << endl;
        stringstream in;
        in << line;
        in >> str ;
       if( str == "input"){
            while (in >> ch){
                in >> num;
                str = ch + to_string(num);
                input.push_back(str);
                bool used = false;
                for(int i=0 ; i< node.size(); i++){
                    if(ch == node[i].name){
                        node[i].num ++;
                        used = true;
```

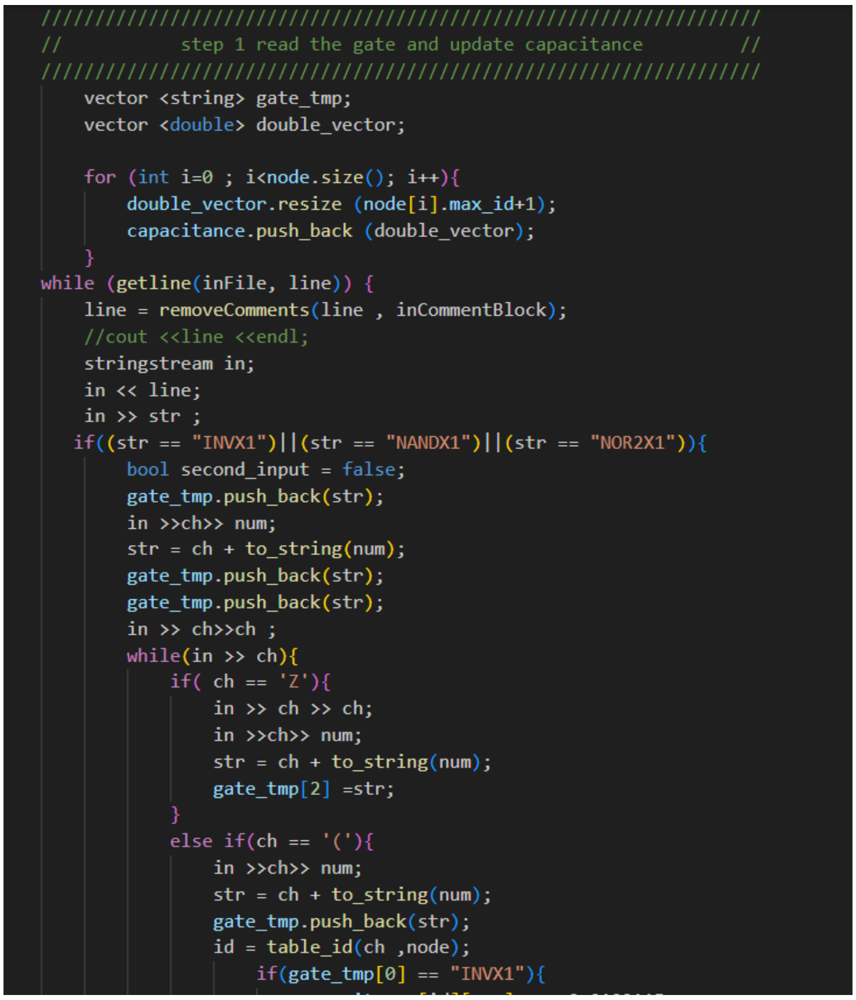


```cpp
                for(int i=0 ; i< node.size(); i++){
                    if(ch == node[i].name){
                        node[i].num ++;
                        used = true;
                        if(num > node[i].max_id)
                            node[i].max_id = num;
                        break;
                    }
                }
                if(used == false){
                    nnode tmp;
                    tmp.name = ch;
                    tmp.num = 0;
                    node.push_back (tmp);
                }
                in >> ch ;
            }
            input_read = true ;
        }
        if( str == "output"){
            while (in >> ch){
                in >> num;
                str = ch + to_string(num);
                output.push_back(str);
                bool used = false;
                for(int i=0 ; i< node.size(); i++){
                    if(ch == node[i].name){
                        node[i].num ++;
                        used = true;
                        if(num > node[i].max_id)
                            node[i].max_id = num;
                        break;
                    }
                }
                if(used == false){
                    nnode tmp;
                    tmp.name = ch;
```

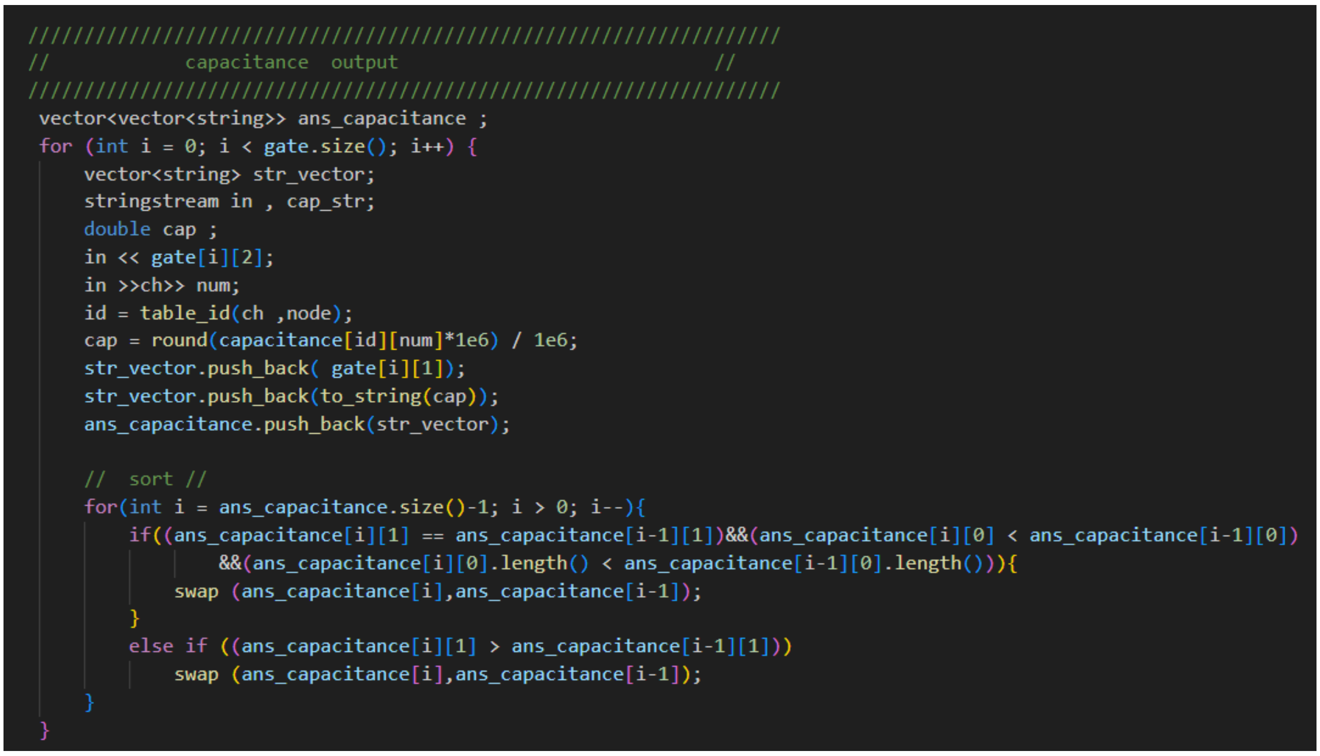


```cpp
    if( str == "wire"){
        while (in >> ch){
            in >> num;
            bool used = false;
            for(int i=0 ; i< node.size(); i++){
                if(ch == node[i].name){
                    node[i].num ++;
                    used = true;
                    if(num > node[i].max_id)
                        node[i].max_id = num;
                    break;
                }
            }
            if(used == false){
                nnode tmp;
                tmp.name = ch;
                tmp.num = 0;
                node.push_back (tmp);
            }
            in >> ch ;
        }
        wire_read= true;
    }
    if(wire_read&&output_read&&input_read) break;

}
```

算電容

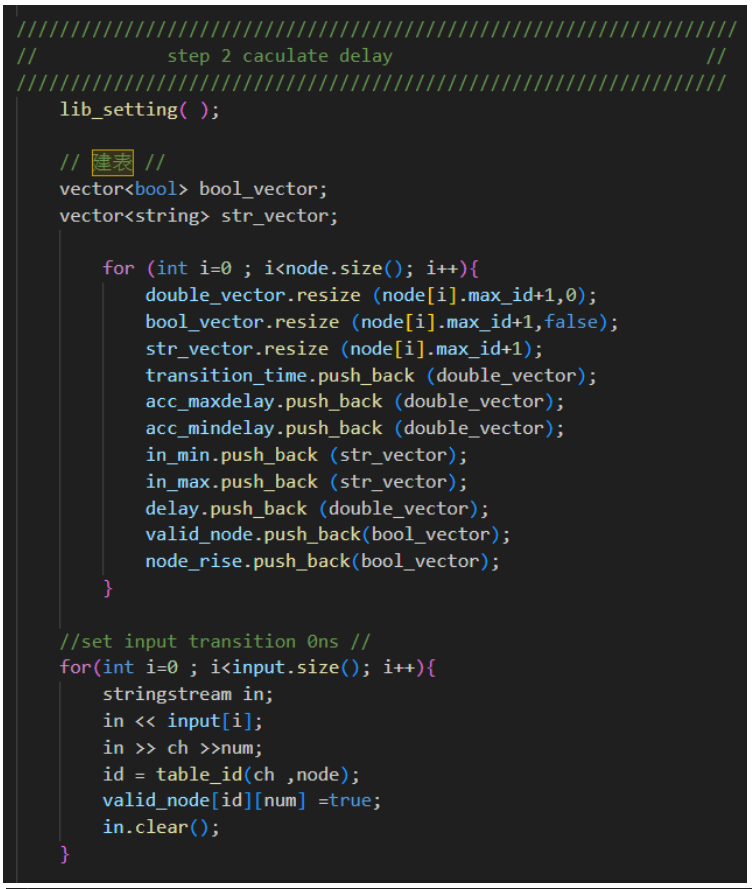


```cpp
/////////////////////////////////////////////////////////////////////
//            step 1 read the gate and update capacitance          //
/////////////////////////////////////////////////////////////////////
    vector <string> gate_tmp;
    vector <double> double_vector;

    for (int i=0 ; i<node.size(); i++){
        double_vector.resize (node[i].max_id+1);
        capacitance.push_back (double_vector);
    }
while (getline(inFile, line)) {
    line = removeComments(line , inCommentBlock);
    //cout <<line <<endl;
    stringstream in;
    in << line;
    in >> str ;
   if((str == "INVX1")||(str == "NANDX1")||(str == "NOR2X1")){
        bool second_input = false;
        gate_tmp.push_back(str);
        in >>ch>> num;
        str = ch + to_string(num);
        gate_tmp.push_back(str);
        gate_tmp.push_back(str);
        in >> ch>>ch ;
        while(in >> ch){
            if( ch == 'Z'){
                in >> ch >> ch;
                in >>ch>> num;
                str = ch + to_string(num);
                gate_tmp[2] =str;
            }
            else if(ch == '('){
                in >>ch>> num;
                str = ch + to_string(num);
                gate_tmp.push_back(str);
                id = table_id(ch ,node);
                    if(gate_tmp[0] == "INVX1"){
```

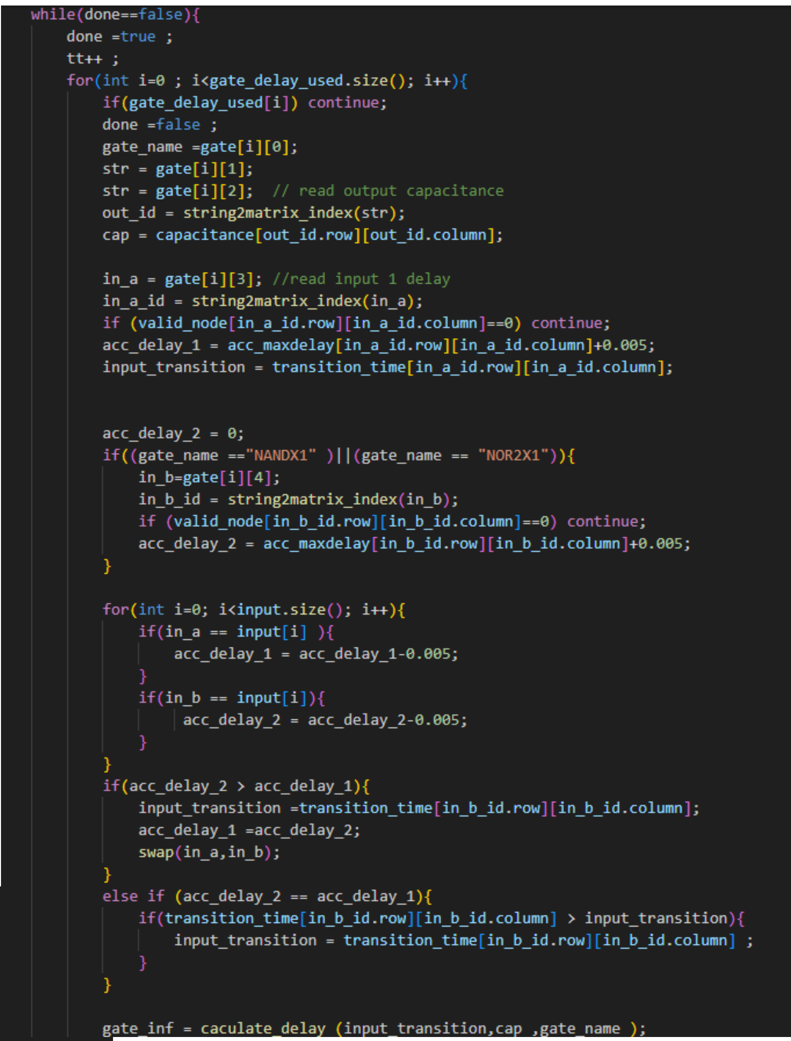


```cpp
/////////////////////////////////////////////////////////////////////
//             capacitance  output                                 //
/////////////////////////////////////////////////////////////////////
 vector<vector<string>> ans_capacitance ;
 for (int i = 0; i < gate.size(); i++) {
     vector<string> str_vector;
     stringstream in , cap_str;
     double cap ;
     in << gate[i][2];
     in >>ch>> num;
     id = table_id(ch ,node);
     cap = round(capacitance[id][num]*1e6) / 1e6;
     str_vector.push_back( gate[i][1]);
     str_vector.push_back(to_string(cap));
     ans_capacitance.push_back(str_vector);

     //  sort //
     for(int i = ans_capacitance.size()-1; i > 0; i--){
         if((ans_capacitance[i][1] == ans_capacitance[i-1][1])&&(ans_capacitance[i][0] < ans_capacitance[i-1][0])
                 &&(ans_capacitance[i][0].length() < ans_capacitance[i-1][0].length())){
             swap (ans_capacitance[i],ans_capacitance[i-1]);
         }
         else if ((ans_capacitance[i][1] > ans_capacitance[i-1][1]))
             swap (ans_capacitance[i],ans_capacitance[i-1]);
     }
 }
```

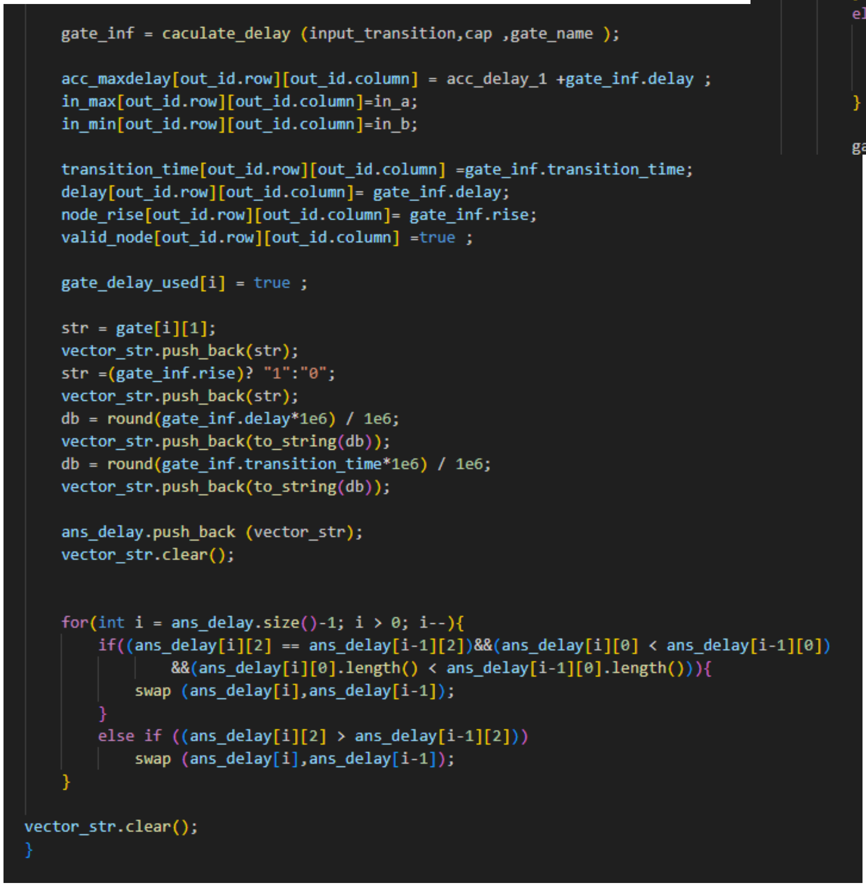
算 delay



```
////////////////////////////////////////////////////////////////////
//            step 2 caculate delay                               //
////////////////////////////////////////////////////////////////////
    lib_setting( );

    // 建表 //
    vector<bool> bool_vector;
    vector<string> str_vector;

        for (int i=0 ; i<node.size(); i++){
            double_vector.resize (node[i].max_id+1,0);
            bool_vector.resize (node[i].max_id+1,false);
            str_vector.resize (node[i].max_id+1);
            transition_time.push_back (double_vector);
            acc_maxdelay.push_back (double_vector);
            acc_mindelay.push_back (double_vector);
            in_min.push_back (str_vector);
            in_max.push_back (str_vector);
            delay.push_back (double_vector);
            valid_node.push_back(bool_vector);
            node_rise.push_back(bool_vector);
        }

    //set input transition 0ns //
    for(int i=0 ; i<input.size(); i++){
        stringstream in;
        in << input[i];
        in >> ch >>num;
        id = table_id(ch ,node);
        valid_node[id][num] =true;
        in.clear();
    }
```

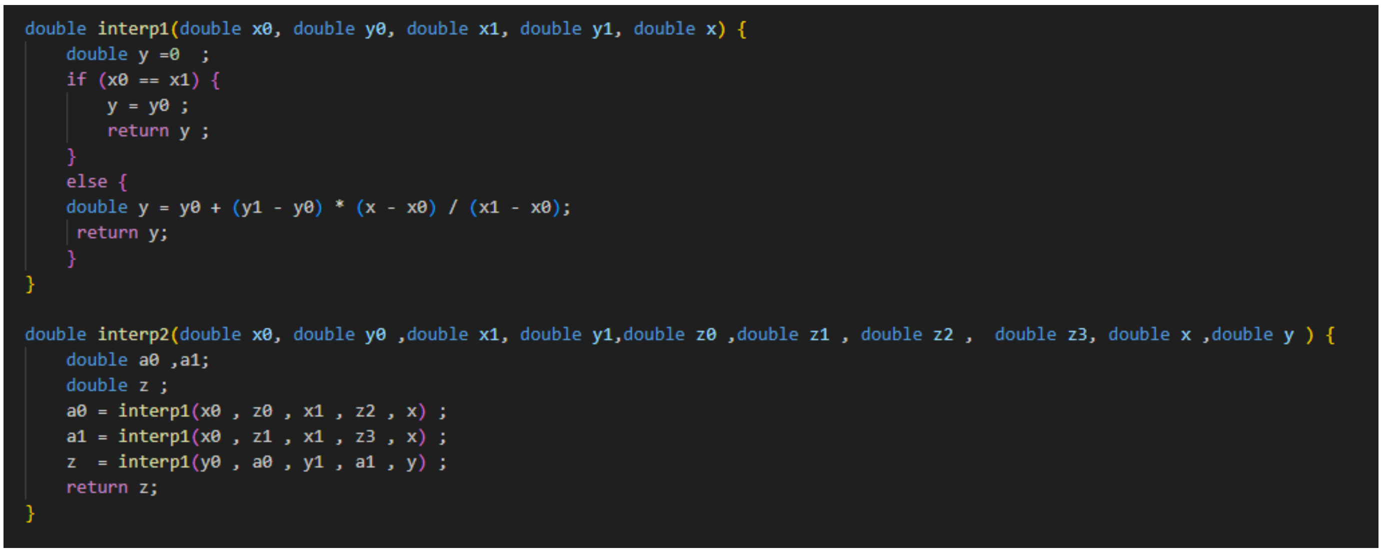


```
    while(done==false){
        done =true ;
        tt++ ;
        for(int i=0 ; i<gate_delay_used.size(); i++){
            if(gate_delay_used[i]) continue;
            done =false ;
            gate_name =gate[i][0];
            str = gate[i][1];
            str = gate[i][2];  // read output capacitance
            out_id = string2matrix_index(str);
            cap = capacitance[out_id.row][out_id.column];

            in_a = gate[i][3]; //read input 1 delay
            in_a_id = string2matrix_index(in_a);
            if (valid_node[in_a_id.row][in_a_id.column]==0) continue;
            acc_delay_1 = acc_maxdelay[in_a_id.row][in_a_id.column]+0.005;
            input_transition = transition_time[in_a_id.row][in_a_id.column];


            acc_delay_2 = 0;
            if((gate_name =="NANDX1" )||(gate_name == "NOR2X1")){
                in_b=gate[i][4];
                in_b_id = string2matrix_index(in_b);
                if (valid_node[in_b_id.row][in_b_id.column]==0) continue;
                acc_delay_2 = acc_maxdelay[in_b_id.row][in_b_id.column]+0.005;
            }

            for(int i=0; i<input.size(); i++){
                if(in_a == input[i] ){
                    acc_delay_1 = acc_delay_1-0.005;
                }
                if(in_b == input[i]){
                     acc_delay_2 = acc_delay_2-0.005;
                }
            }
            if(acc_delay_2 > acc_delay_1){
                input_transition =transition_time[in_b_id.row][in_b_id.column];
                acc_delay_1 =acc_delay_2;
                swap(in_a,in_b);
            }
            else if (acc_delay_2 == acc_delay_1){
                if(transition_time[in_b_id.row][in_b_id.column] > input_transition){
                    input_transition = transition_time[in_b_id.row][in_b_id.column] ;
                }
            }

            gate_inf = caculate_delay (input_transition,cap ,gate_name );
```



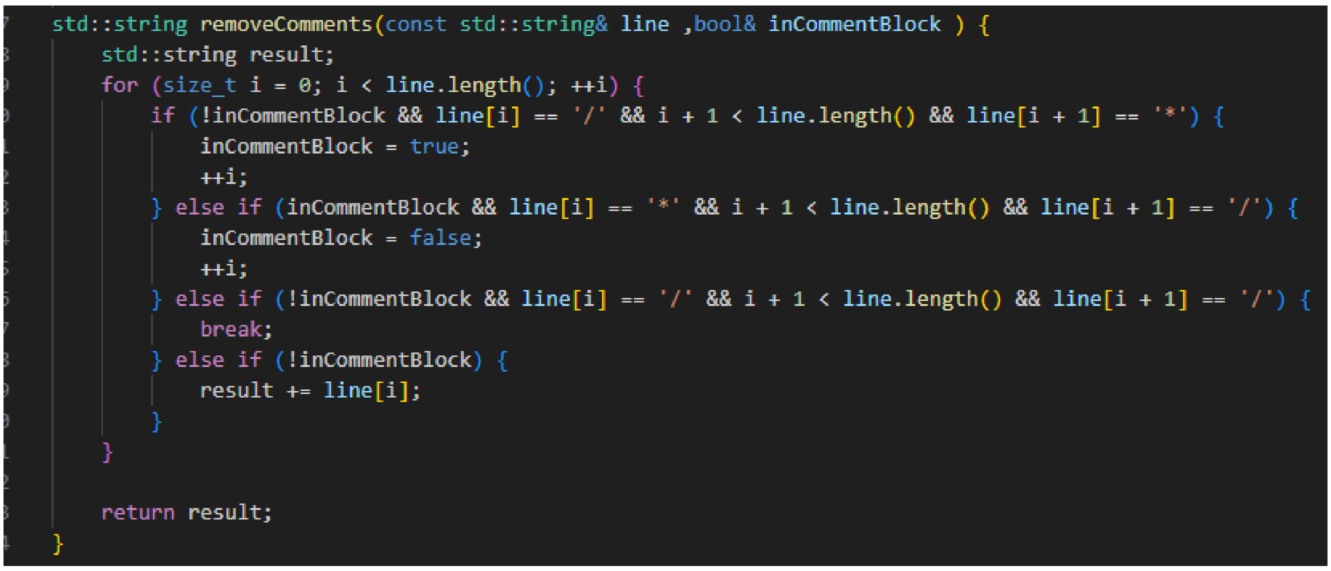
```
            gate_inf = caculate_delay (input_transition,cap ,gate_name );

            acc_maxdelay[out_id.row][out_id.column] = acc_delay_1 +gate_inf.delay ;
            in_max[out_id.row][out_id.column]=in_a;
            in_min[out_id.row][out_id.column]=in_b;

            transition_time[out_id.row][out_id.column] =gate_inf.transition_time;
            delay[out_id.row][out_id.column]= gate_inf.delay;
            node_rise[out_id.row][out_id.column]= gate_inf.rise;
            valid_node[out_id.row][out_id.column] =true ;

            gate_delay_used[i] = true ;

            str = gate[i][1];
            vector_str.push_back(str);
            str =(gate_inf.rise)? "1":"0";
            vector_str.push_back(str);
            db = round(gate_inf.delay*1e6) / 1e6;
            vector_str.push_back(to_string(db));
            db = round(gate_inf.transition_time*1e6) / 1e6;
            vector_str.push_back(to_string(db));

            ans_delay.push_back (vector_str);
            vector_str.clear();


            for(int i = ans_delay.size()-1; i > 0; i--){
                if((ans_delay[i][2] == ans_delay[i-1][2])&&(ans_delay[i][0] < ans_delay[i-1][0])
                        &&(ans_delay[i][0].length() < ans_delay[i-1][0].length())){
                    swap (ans_delay[i],ans_delay[i-1]);
                }
                else if ((ans_delay[i][2] > ans_delay[i-1][2]))
                    swap (ans_delay[i],ans_delay[i-1]);
            }

        vector_str.clear();
        }
```

內插



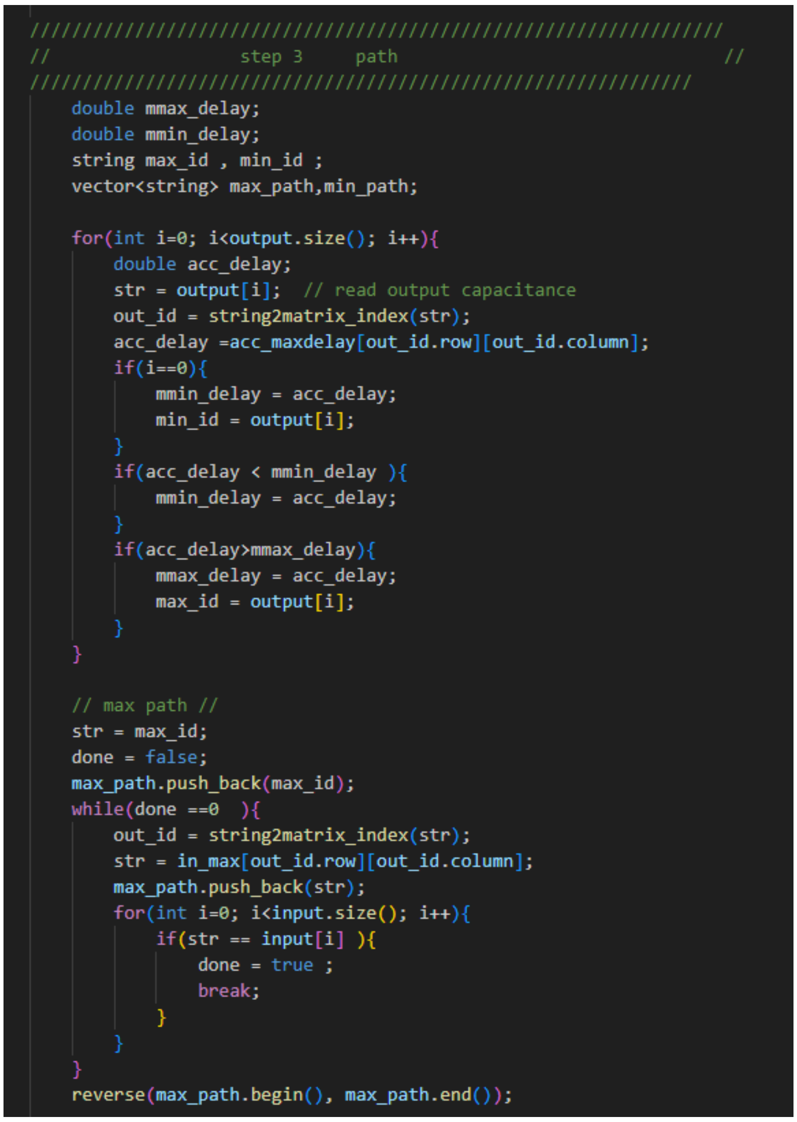
```
double interp1(double x0, double y0, double x1, double y1, double x) {
    double y =0  ;
    if (x0 == x1) {
        y = y0 ;
        return y ;
    }
    else {
    double y = y0 + (y1 - y0) * (x - x0) / (x1 - x0);
     return y;
    }
}

double interp2(double x0, double y0 ,double x1, double y1,double z0 ,double z1 , double z2 ,  double z3, double x ,double y ) {
    double a0 ,a1;
    double z ;
    a0 = interp1(x0 , z0 , x1 , z2 , x) ;
    a1 = interp1(x0 , z1 , x1 , z3 , x) ;
    z  = interp1(y0 , a0 , y1 , a1 , y) ;
    return z;
}
```

移除註解

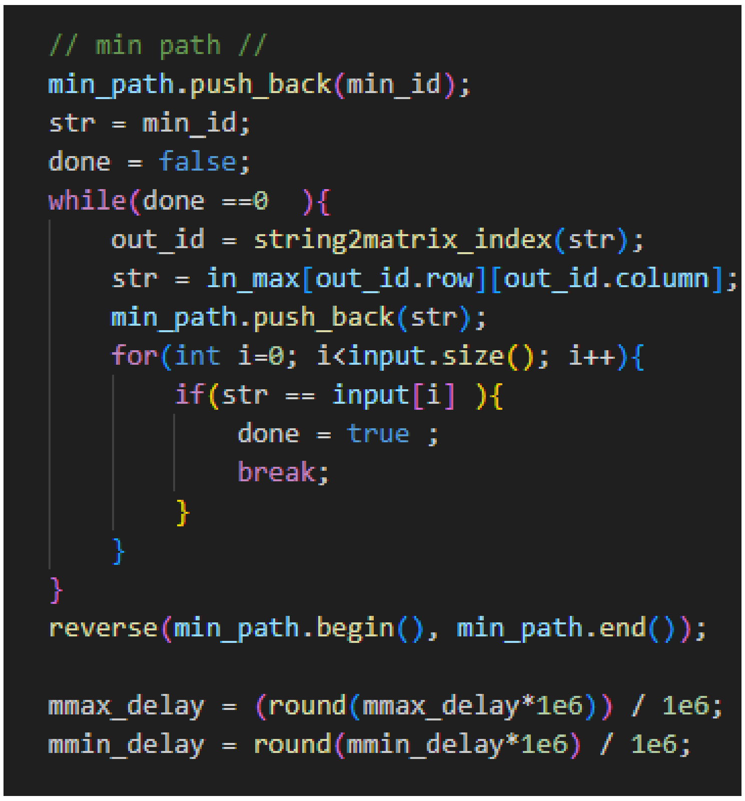


```
std::string removeComments(const std::string& line ,bool& inCommentBlock ) {
    std::string result;
    for (size_t i = 0; i < line.length(); ++i) {
        if (!inCommentBlock && line[i] == '/' && i + 1 < line.length() && line[i + 1] == '*') {
            inCommentBlock = true;
            ++i;
        } else if (inCommentBlock && line[i] == '*' && i + 1 < line.length() && line[i + 1] == '/') {
            inCommentBlock = false;
            ++i;
        } else if (!inCommentBlock && line[i] == '/' && i + 1 < line.length() && line[i + 1] == '/') {
            break;
        } else if (!inCommentBlock) {
            result += line[i];
        }
    }

    return result;
}
```

算 max、min path

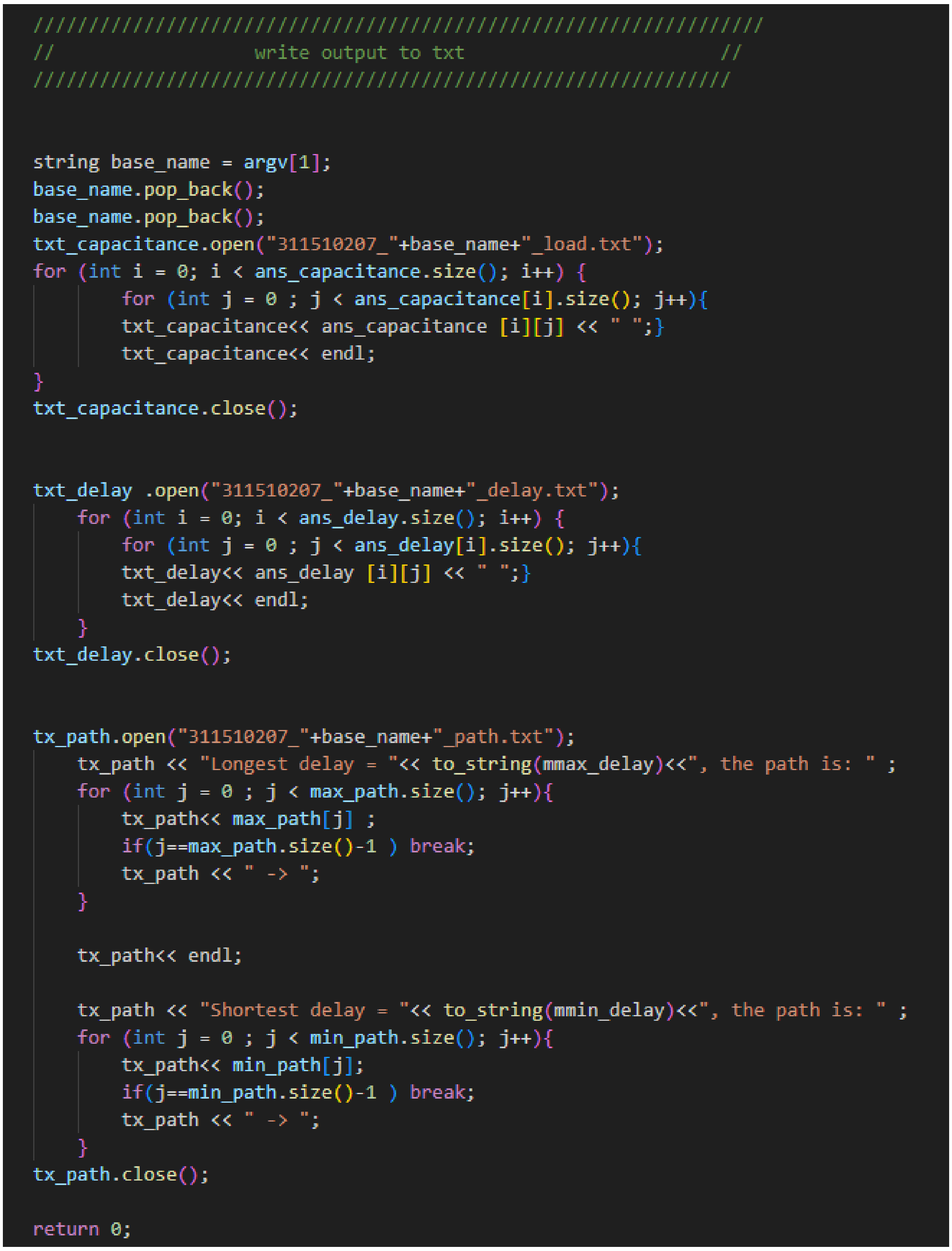


```
//////////////////////////////////////////////////////////////////
//                   step 3     path                            //
///////////////////////////////////////////////////////////////
    double mmax_delay;
    double mmin_delay;
    string max_id , min_id ;
    vector<string> max_path,min_path;

    for(int i=0; i<output.size(); i++){
        double acc_delay;
        str = output[i];  // read output capacitance
        out_id = string2matrix_index(str);
        acc_delay =acc_maxdelay[out_id.row][out_id.column];
        if(i==0){
            mmin_delay = acc_delay;
            min_id = output[i];
        }
        if(acc_delay < mmin_delay ){
            mmin_delay = acc_delay;
        }
        if(acc_delay>mmax_delay){
            mmax_delay = acc_delay;
            max_id = output[i];
        }
    }

    // max path //
    str = max_id;
    done = false;
    max_path.push_back(max_id);
    while(done ==0  ){
        out_id = string2matrix_index(str);
        str = in_max[out_id.row][out_id.column];
        max_path.push_back(str);
        for(int i=0; i<input.size(); i++){
            if(str == input[i] ){
                done = true ;
                break;
            }
        }
    }
    reverse(max_path.begin(), max_path.end());
```



```
// min path //
min_path.push_back(min_id);
str = min_id;
done = false;
while(done ==0  ){
    out_id = string2matrix_index(str);
    str = in_max[out_id.row][out_id.column];
    min_path.push_back(str);
    for(int i=0; i<input.size(); i++){
        if(str == input[i] ){
            done = true ;
            break;
        }
    }
}
reverse(min_path.begin(), min_path.end());

mmax_delay = (round(mmax_delay*1e6)) / 1e6;
mmin_delay = round(mmin_delay*1e6) / 1e6;
```

Output 寫回 txt



```
//////////////////////////////////////////////////////////////////
//                  write output to txt                       //
///////////////////////////////////////////////////////////////


string base_name = argv[1];
base_name.pop_back();
base_name.pop_back();
txt_capacitance.open("311510207_"+base_name+"_load.txt");
for (int i = 0; i < ans_capacitance.size(); i++) {
        for (int j = 0 ; j < ans_capacitance[i].size(); j++){
        txt_capacitance<< ans_capacitance [i][j] << " ";}
        txt_capacitance<< endl;
}
txt_capacitance.close();


txt_delay .open("311510207_"+base_name+"_delay.txt");
    for (int i = 0; i < ans_delay.size(); i++) {
        for (int j = 0 ; j < ans_delay[i].size(); j++){
        txt_delay<< ans_delay [i][j] << " ";}
        txt_delay<< endl;
    }
txt_delay.close();


tx_path.open("311510207_"+base_name+"_path.txt");
    tx_path << "Longest delay = "<< to_string(mmax_delay)<<", the path is: " ;
    for (int j = 0 ; j < max_path.size(); j++){
        tx_path<< max_path[j] ;
        if(j==max_path.size()-1 ) break;
        tx_path << " -> ";
    }

    tx_path<< endl;

    tx_path << "Shortest delay = "<< to_string(mmin_delay)<<", the path is: " ;
    for (int j = 0 ; j < min_path.size(); j++){
        tx_path<< min_path[j];
        if(j==min_path.size()-1 ) break;
        tx_path << " -> ";
    }
tx_path.close();

return 0;
```