10/10/2024

**Lab 04 Report**

Ryan Beck

Jared Bronson

**Objectives**

Write a simulated annealing program in C or C++ that solves the FPGA placement problem. The program will ingest a text file with a grid size, number of vertices, and edges between two nodes. Using the values given in the text file, the program will place the edges on a grid. The nodes will then go through an annealing process to find a solution to the placement problem. When the program finds an acceptable solution, it will print to an output file.

**Procedures**

Create a C++ file with global variables for the initial temperature, cooling rate, and stop threshold. In the main function, take the given input file and read the parameters. In the input file, the type of input will be denoted by the first character on each line. The characters used will be g for grid size, v for vertices, and e for edges. Use the parameters given by the input file to make a properly sized grid, and place the given nodes at random coordinates on the grid. Do not allow a node to be placed in the same spot as another in the beginning.

After the nodes are placed, pass them into the anneal function. First, the anneal function will evaluate the initial score of the placement. Then, while the temperature is greater than the stop threshold, the next nodes are set to equal the current nodes. After changing the next node, it goes to be altered. In the alter function a random number from 0-2 for both the x and y positions. These random numbers will be used to change the x and y positions. If the number is two, the position will be moved one in the negative direction. The new positions are put into evaluation to check the new score. The old score and new score are then compared to determined if the new score will be accepted. If the old score is better, there is still a chance the new score will be accepted because it may lead to a better solution. Then temperature is reduced through the cooling function. This will repeat until the temperature is less than the stop threshold.

After the annealing process, the results need to be printed to an output file. The output file should show the new placement of each node, as well as the length of the edges between two nodes.

**Statistical Analysis**

The team ran a statistical analysis to assess the relationship between the cooling rate and both the execution time and solution quality. This analysis is not complete, as results were only collected from the first input.txt file due to time constraints. In addition, more executions at each cooling rate would also provide more accurate data.

The program was run with 5 different cooling rates, and the corresponding solutions tried and final score are recorded. The number of solutions tried is proportional to the execution time, as more loops through the anneal function are required. As a note, the INITIAL\_TEMPERATURE is 100000, and STOP\_THRESHOLD is 0.001.

Cooling Rate: 0.9

|  |  |  |
| --- | --- | --- |
| Run Number | Solutions Tried | Final Score |
| 1 | 182 | 16 |
| 2 | 182 | 22 |
| 3 | 182 | 20 |
| 4 | 182 | 19 |
| 5 | 182 | 17 |

Cooling Rate: 0.99

|  |  |  |
| --- | --- | --- |
| Run Number | Solutions Tried | Final Score |
| 1 | 1840 | 13 |
| 2 | 1840 | 11 |
| 3 | 1840 | 14 |
| 4 | 1840 | 12 |
| 5 | 1840 | 12 |

Cooling Rate: 0.999

|  |  |  |
| --- | --- | --- |
| Run Number | Solutions Tried | Final Score |
| 1 | 18419 | 9 |
| 2 | 18419 | 12 |
| 3 | 18419 | 10 |
| 4 | 18419 | 9 |
| 5 | 18419 | 8 |

Cooling Rate: 0.9999

|  |  |  |
| --- | --- | --- |
| Run Number | Solutions Tried | Final Score |
| 1 | 184205 | 8 |
| 2 | 184205 | 9 |
| 3 | 184205 | 7 |
| 4 | 184205 | 8 |
| 5 | 184205 | 9 |

Cooling Rate: 0.99999

|  |  |  |
| --- | --- | --- |
| Run Number | Solutions Tried | Final Score |
| 1 | 1842066 | 7 |
| 2 | 1842066 | 7 |
| 3 | 1842066 | 8 |
| 4 | 1842066 | 8 |
| 5 | 1842066 | 7 |

**Results**

The place.cpp file and header file, which can be found in figures, successfully takes the input of an input file. It takes the input and creates a grid with nodes placed randomly throughout. These nodes are sent through an annealing process to find the best possible solution allowed given the time restraints. When done cooling off, the function prints the node placement and edge length to an output file. Using an input file with six edges and running the program, the best score achieved is seven.

The statistical analysis shows a positive correlation between cooling rate, solutions tried, and final score. As the cooling rate is increased (approaches 1), more solutions are checked and the score is improved. It can also be noted that as another 9 is added to the cooling rate, the solutions tried, and therefore the execution time, increases by a factor of ~10.

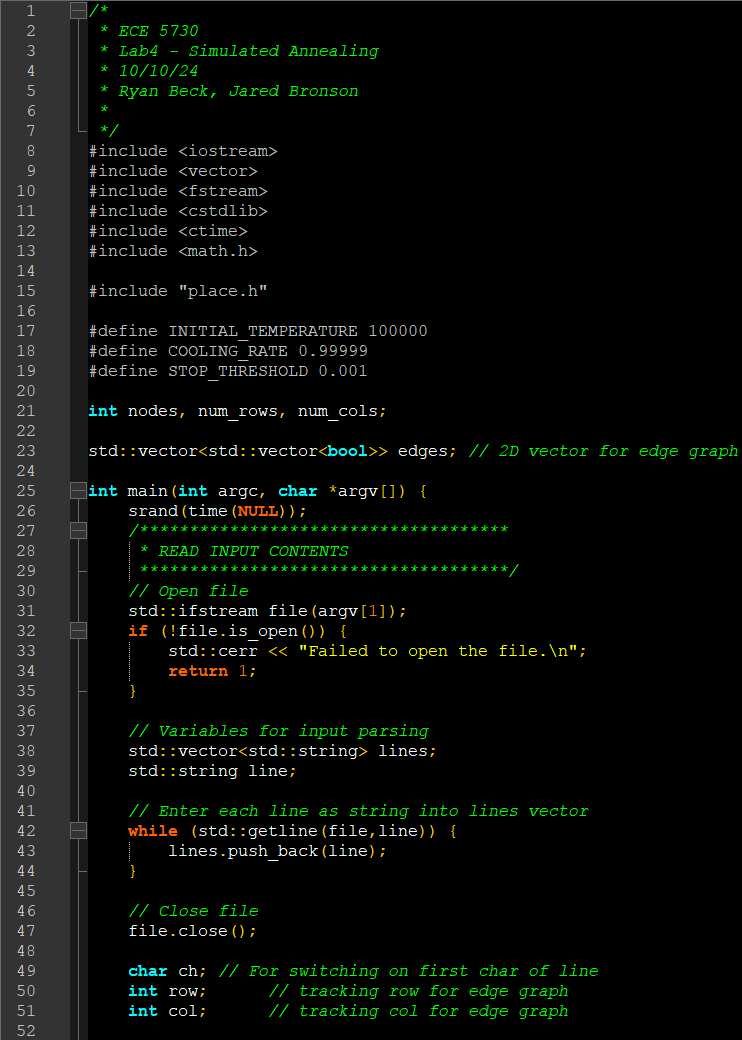
**Figures**

Figure 1: Main function Pt. 1 in Place.cpp

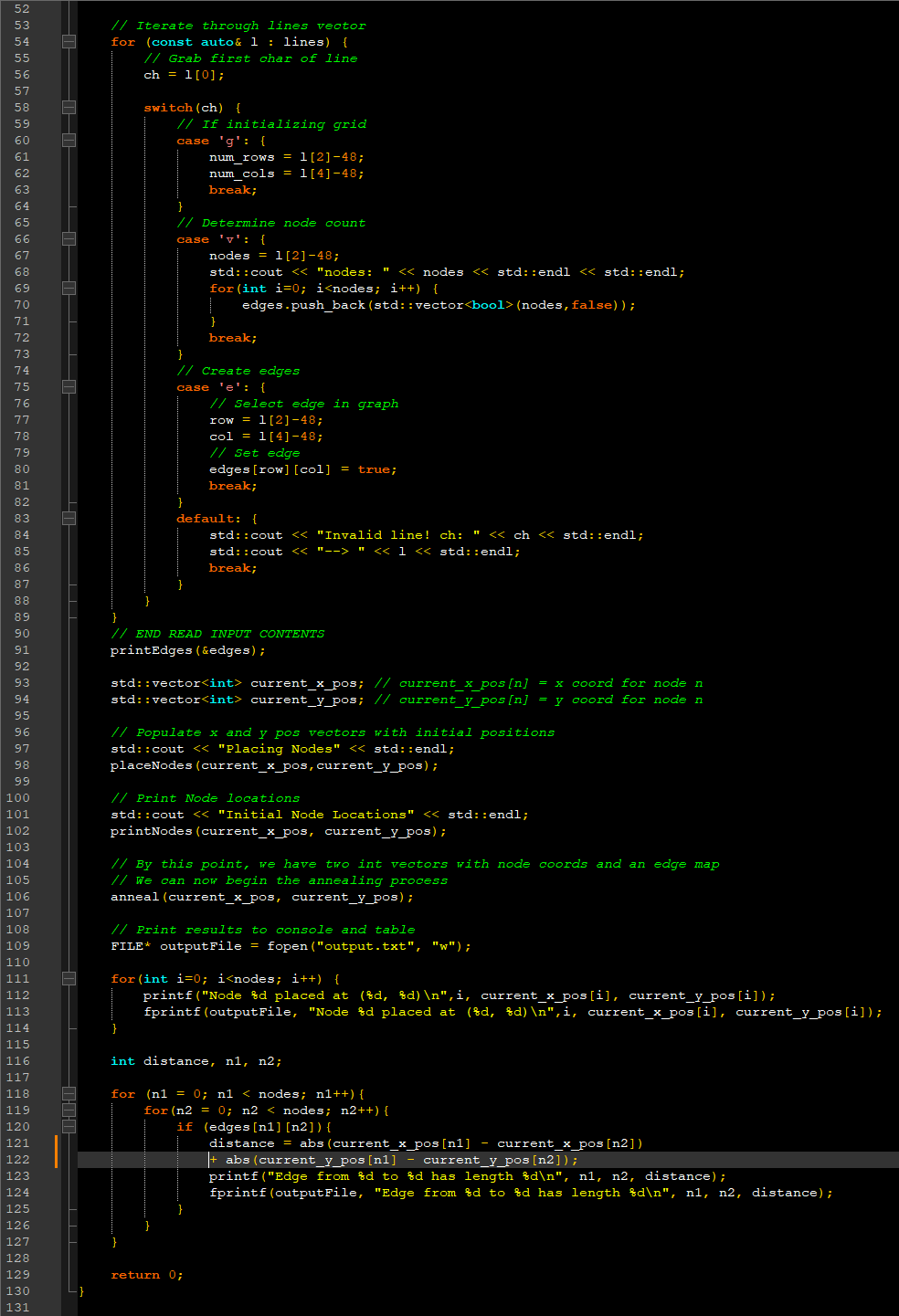


Figure 2: Main function Pt. 2 in Place.cpp

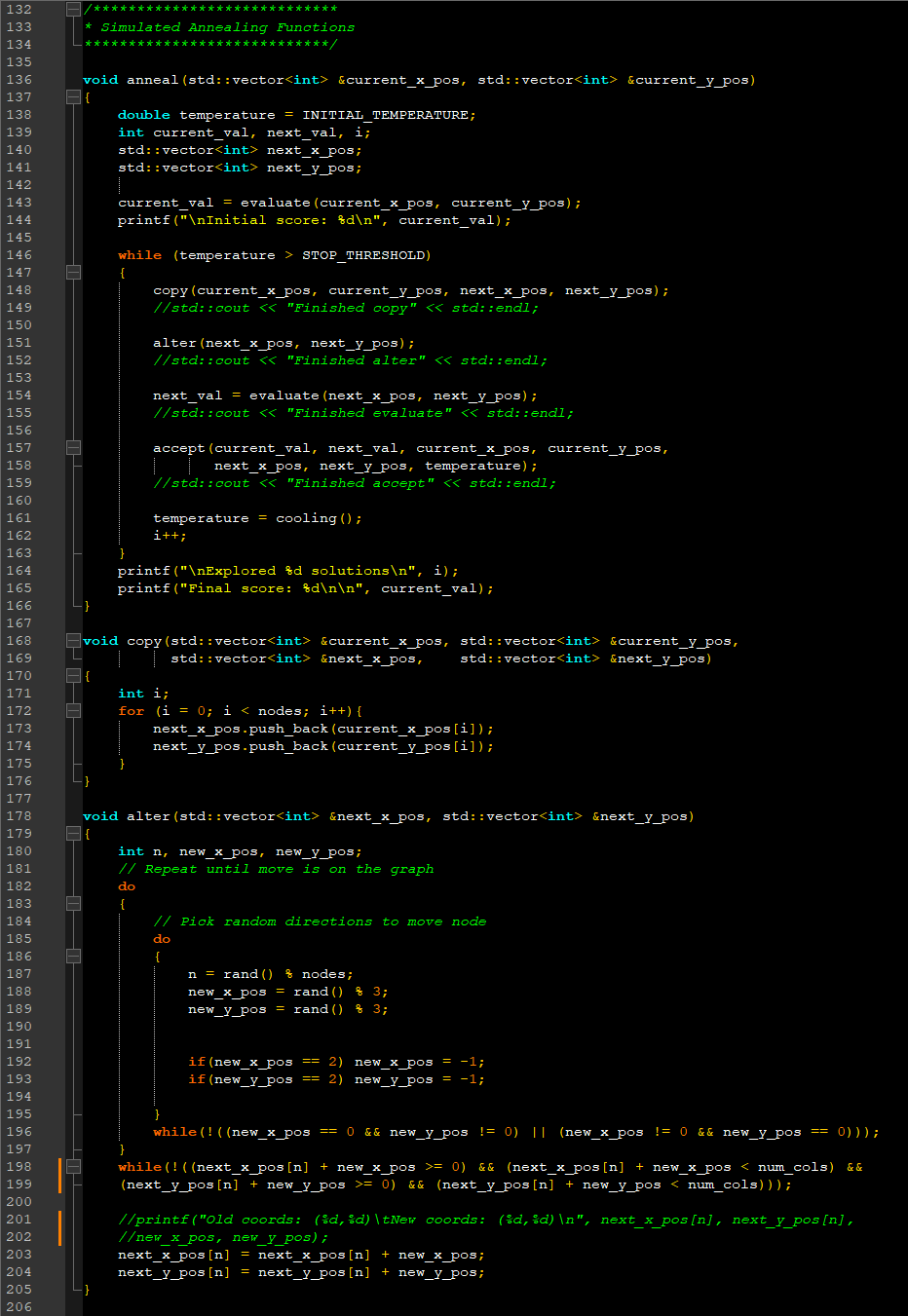


Figure 3: Annealing Functions Pt. 1 in Place.cpp

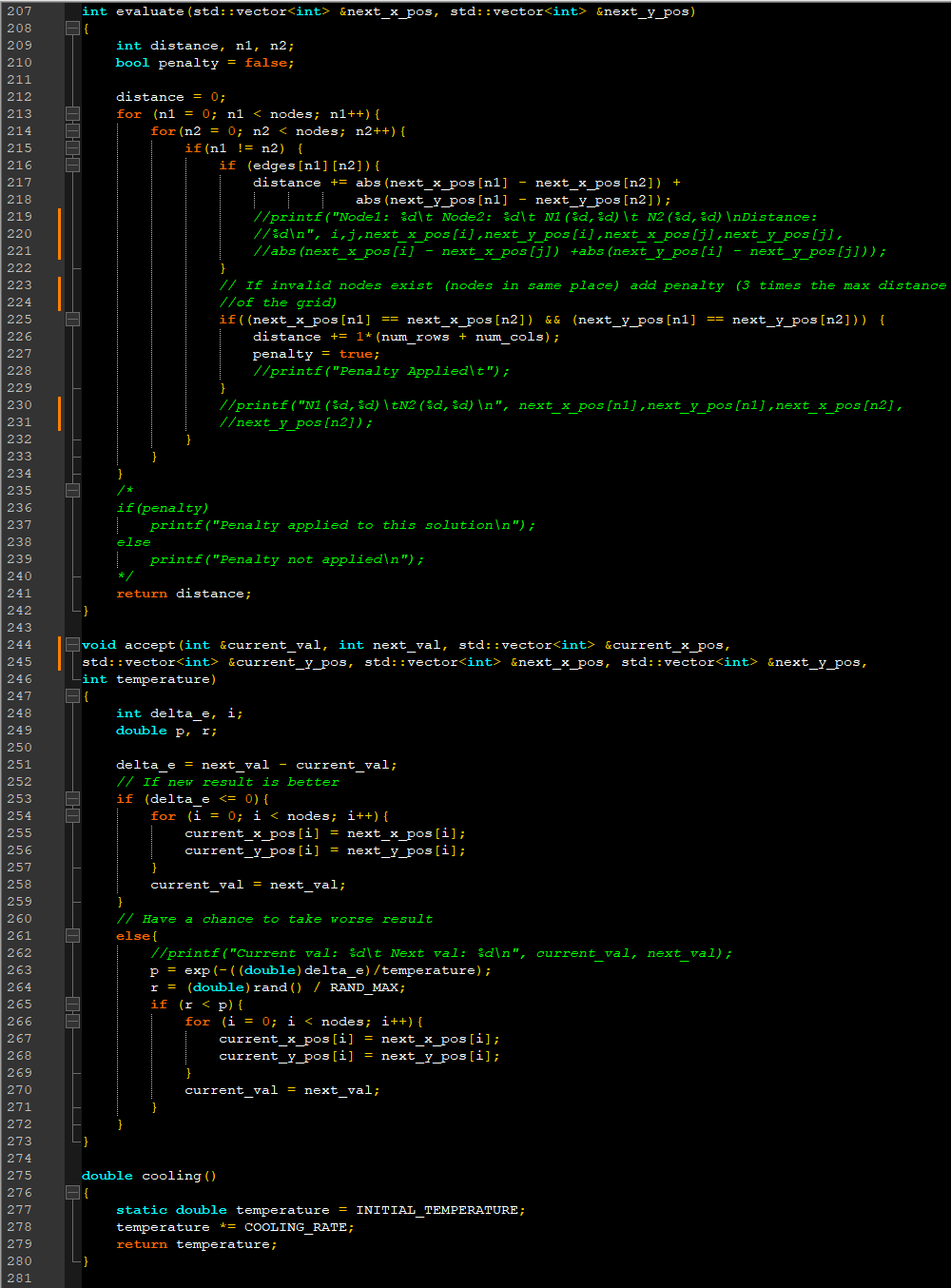


Figure 4: Annealing Functions Pt. 2 in Place.cpp

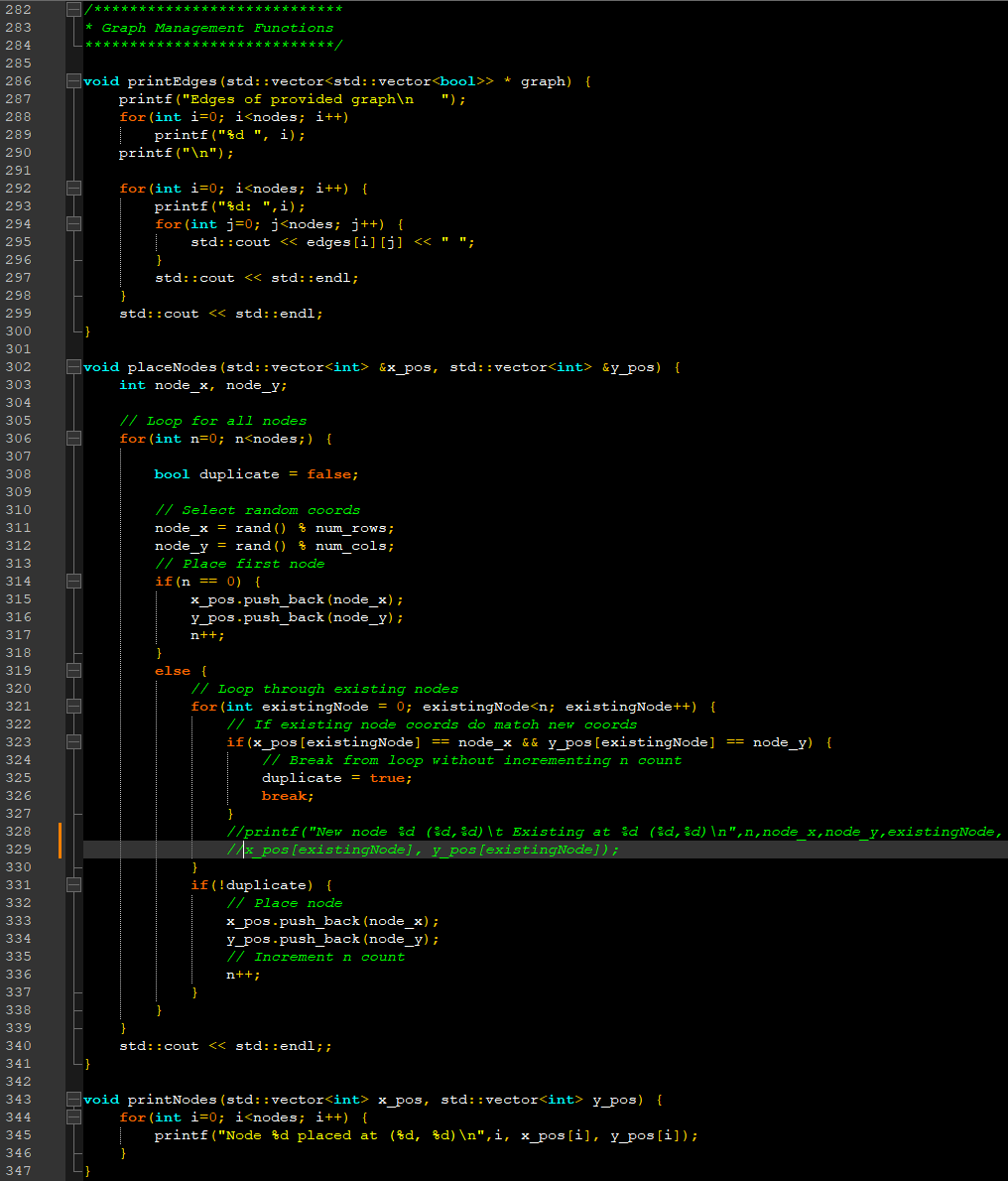


Figure 5: Grid Management functions in Place.cpp

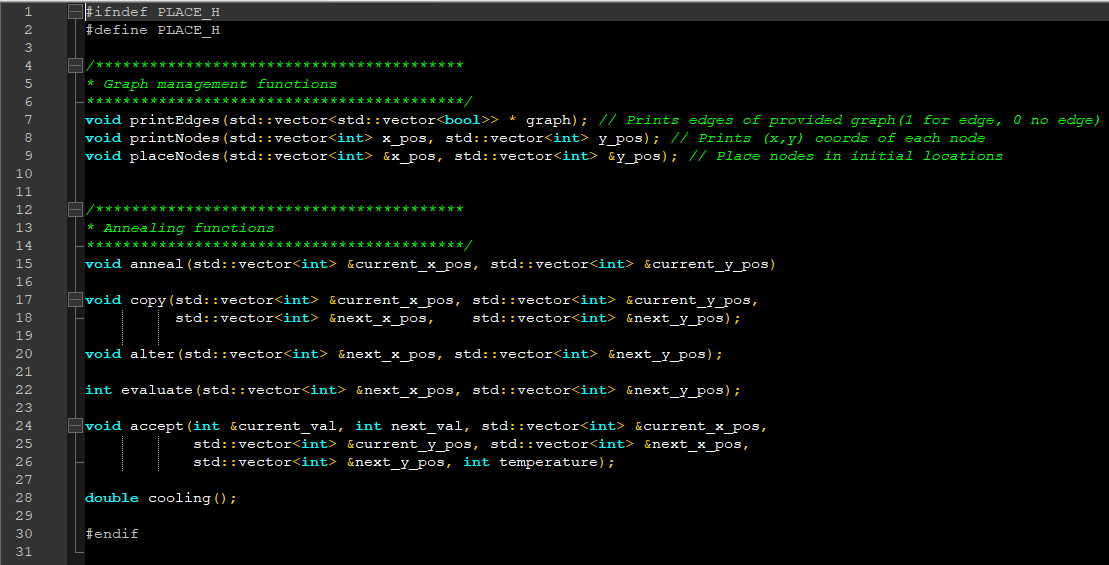
****

Figure 5: Place.h header file

**Conclusion**

In conclusion, we were able to implement a program to take an input file with parameters for a grid, vertices, and edges of nodes for an FPGA. The program places the nodes at random locations on the grid, and sends them through an annealing process. The nodes are moved randomly, one space away from their current location. Each new node location is then given a score, and assessed if it is better than the previous placement. The program will continue to do this until the cooling process is complete, and the results are printed to an output file. For a 6x6 graph, the best score the program was able to achieve was seven.