**TUGAS HEAP**

**ALGORITMA DAN STRUKTUR DATA**



**Nama : Khulika Malkan**

**NIM : 2311110057**

**Kelas : S1SD04-B**

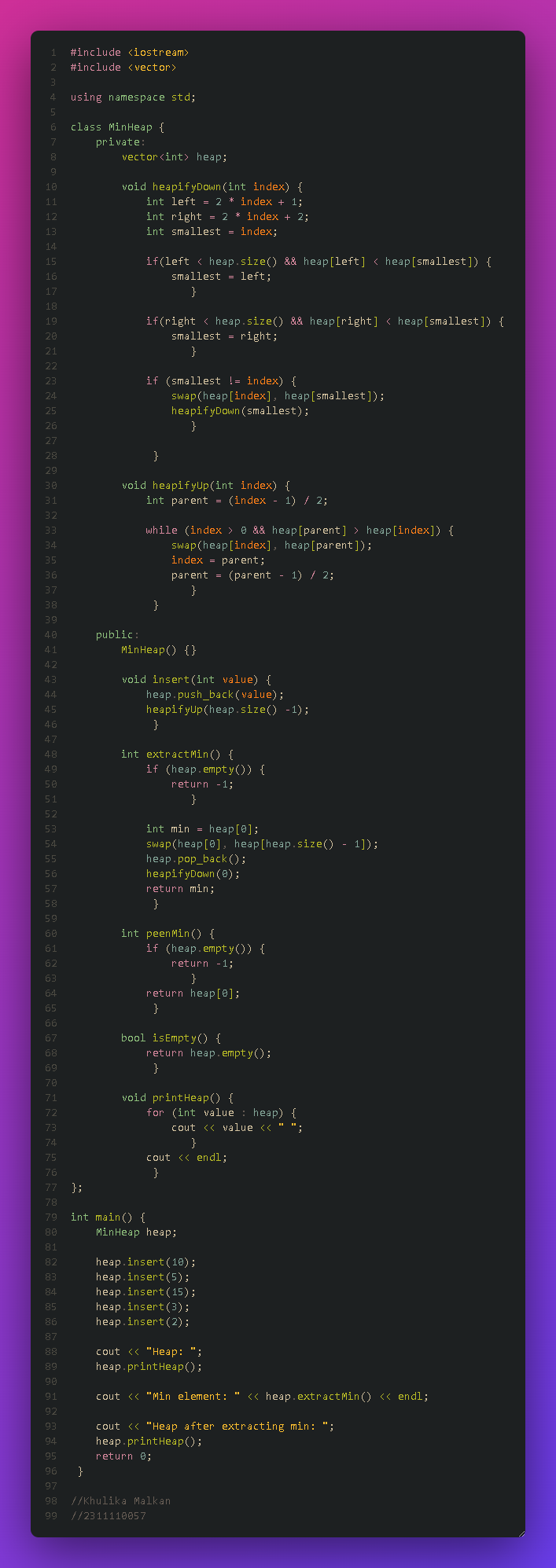
**PROGRAM STUDI SAINS DATA**

**FAKULTAS INFORMATIKA**

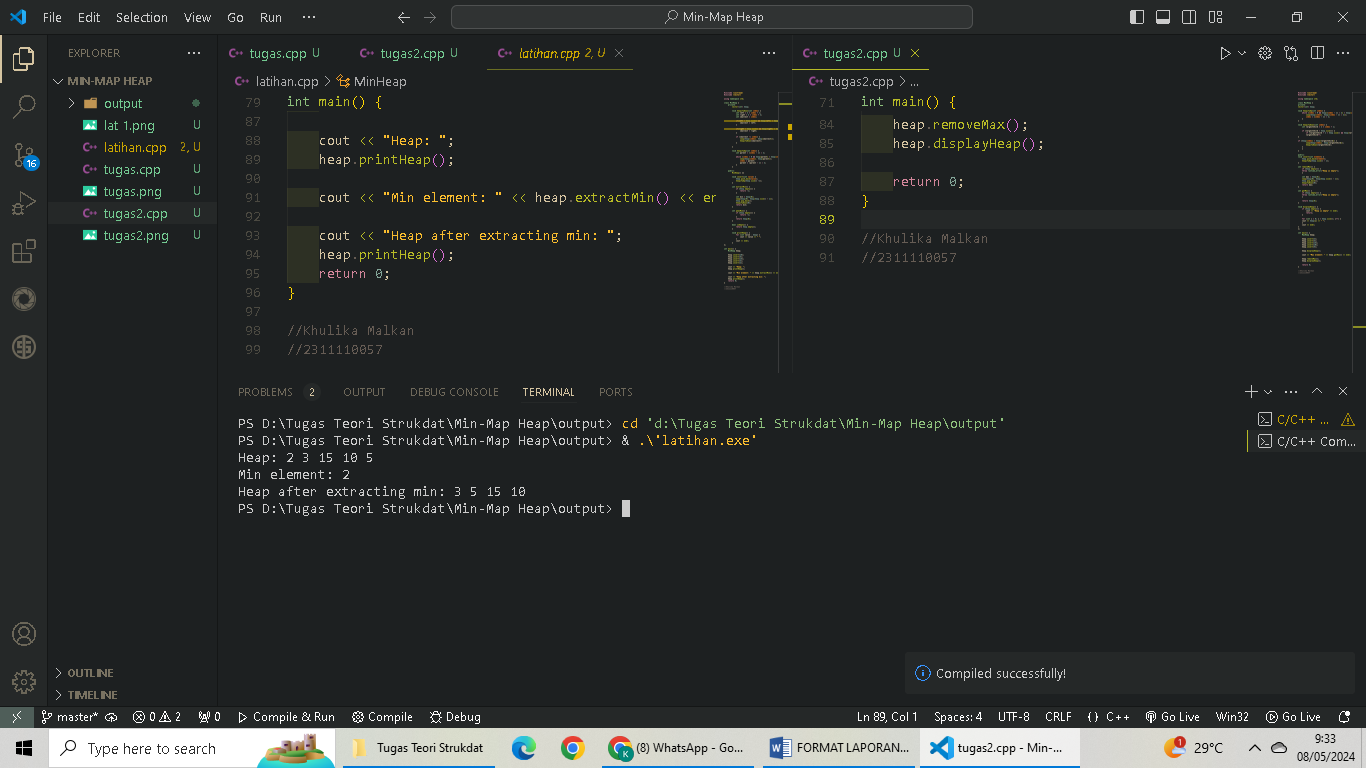
**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2023**

1. **Latihan**
   1. **Kode Program**
2. #include <iostream>
3. #include <vector>
4. using namespace std;
5. class MinHeap {
6. private:
7. vector<int> heap;
8. void heapifyDown(int index) {
9. int left = 2 \* index + 1;
10. int right = 2 \* index + 2;
11. int smallest = index;
12. if(left < heap.size() && heap[left] < heap[smallest]) {
13. smallest = left;
14. }
15. if(right < heap.size() && heap[right] < heap[smallest]) {
16. smallest = right;
17. }
18. if (smallest != index) {
19. swap(heap[index], heap[smallest]);
20. heapifyDown(smallest);
21. }
23. }
24. void heapifyUp(int index) {
25. int parent = (index - 1) / 2;
26. while (index > 0 && heap[parent] > heap[index]) {
27. swap(heap[index], heap[parent]);
28. index = parent;
29. parent = (parent - 1) / 2;
30. }
31. }
32. public:
33. MinHeap() {}
34. void insert(int value) {
35. heap.push\_back(value);
36. heapifyUp(heap.size() -1);
37. }
38. int extractMin() {
39. if (heap.empty()) {
40. return -1;
41. }
42. int min = heap[0];
43. swap(heap[0], heap[heap.size() - 1]);
44. heap.pop\_back();
45. heapifyDown(0);
46. return min;
47. }
48. int peenMin() {
49. if (heap.empty()) {
50. return -1;
51. }
52. return heap[0];
53. }
54. bool isEmpty() {
55. return heap.empty();
56. }
57. void printHeap() {
58. for (int value : heap) {
59. cout << value << " ";
60. }
61. cout << endl;
62. }
63. };
64. int main() {
65. MinHeap heap;
66. heap.insert(10);
67. heap.insert(5);
68. heap.insert(15);
69. heap.insert(3);
70. heap.insert(2);
71. cout << "Heap: ";
72. heap.printHeap();
73. cout << "Min element: " << heap.extractMin() << endl;
74. cout << "Heap after extracting min: ";
75. heap.printHeap();
76. return 0;
77. }
78. //Khulika Malkan
79. //2311110057
    1. **SS full code**

****

* 1. **Output**



1. **Tugas**
   1. **Kode Program**
2. #include <iostream>
3. #include <vector>
4. using namespace std;
5. class MaxHeap {
6. private:
7. vector<int> heap;
8. void heapifyUp(int index) {
9. while (index > 0 && heap[(index - 1) / 2] < heap[index]) {
10. swap(heap[index], heap[(index - 1) / 2]);
11. index = (index - 1) / 2;
12. }
13. }
14. void heapifyDown(int index) {
15. int largestChild = 2 \* index + 1;
16. if (largestChild < heap.size()) {
17. if (largestChild + 1 < heap.size() && heap[largestChild] < heap[largestChild + 1]) {
18. largestChild++;
19. }
20. if (heap[index] < heap[largestChild]) {
21. swap(heap[index], heap[largestChild]);
22. heapifyDown(largestChild);
23. }
24. }
25. }
26. public:
27. void insert(int element) {
28. heap.push\_back(element);
29. heapifyUp(heap.size() - 1);
30. }
31. int removeMax() {
32. if (heap.empty()) {
33. throw runtime\_error("Heap is empty");
34. }
35. int max = heap[0];
36. swap(heap[0], heap[heap.size() - 1]);
37. heap.pop\_back();
38. heapifyDown(0);
39. return max;
40. }
41. int getMax() {
42. if (heap.empty()) {
43. throw runtime\_error("Heap is empty");
44. }
45. return heap[0];
46. }
47. void displayHeap() {
48. if (heap.empty()) {
49. cout << "Heap is empty" << endl;
50. return;
51. }
52. for (int i = 0; i < heap.size(); i++) {
53. cout << heap[i] << " ";
54. }
55. cout << endl;
56. }
57. };
58. int main() {
59. MaxHeap heap;
60. heap.insert(2);
61. heap.insert(3);
62. heap.insert(15);
63. heap.insert(10);
64. heap.insert(5);
65. heap.displayHeap();
67. cout << "Max element: " << heap.getMax() << endl;
68. heap.removeMax();
69. heap.displayHeap();
70. return 0;
71. }
    1. **SS full code**

****

* 1. **Output**

