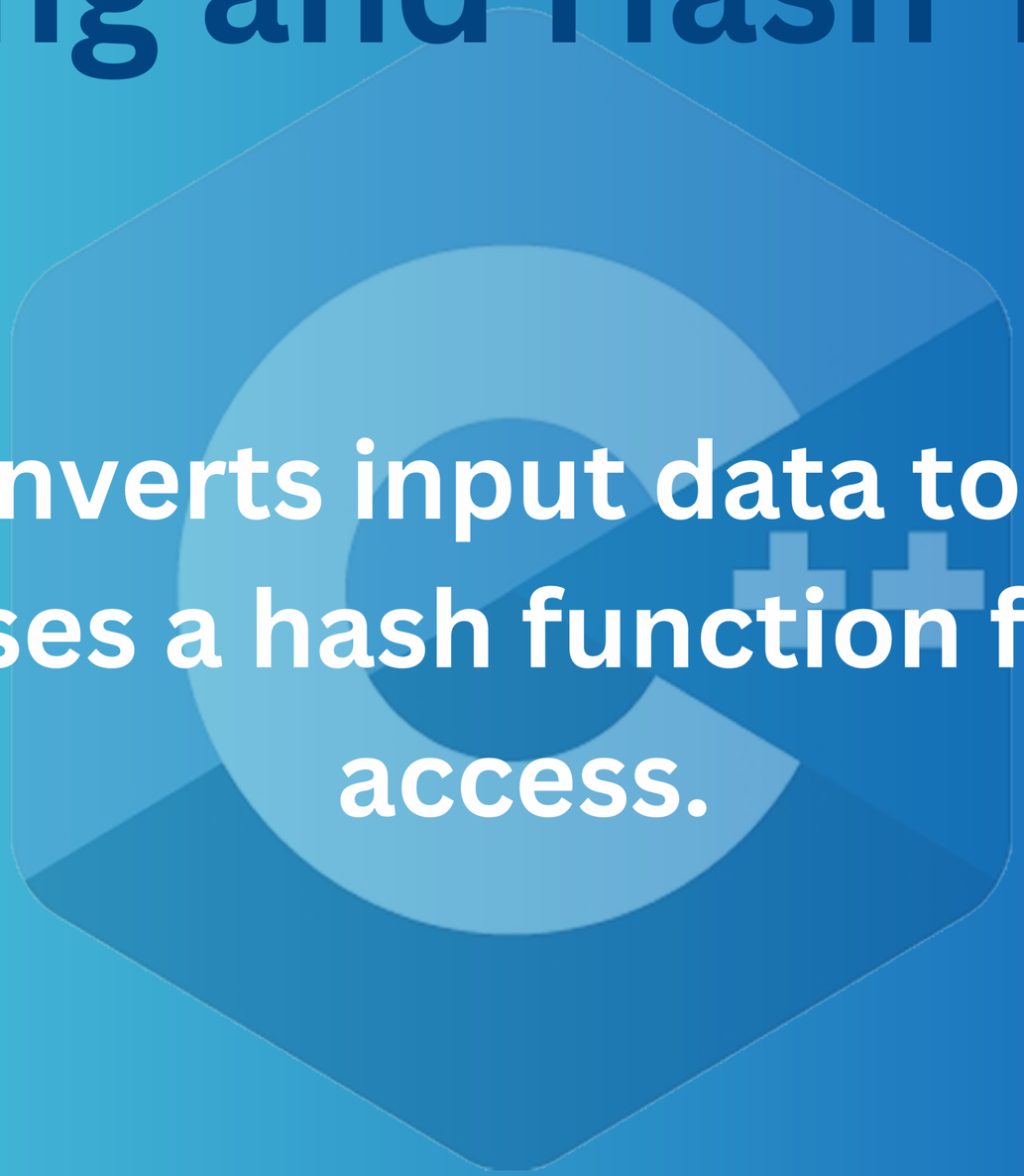
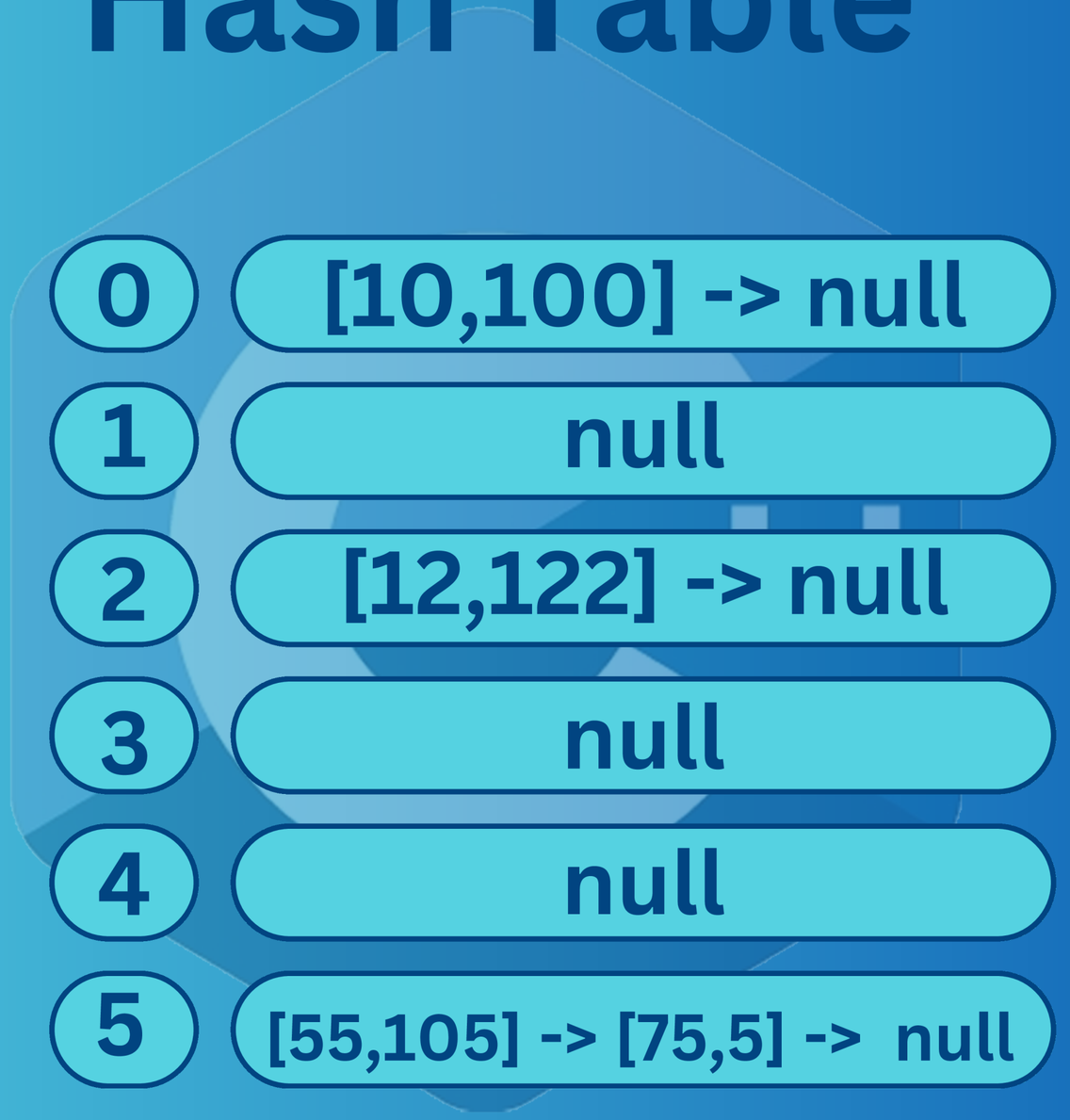
Hashing and Hash Tables



Hashing converts input data to a fixed-size keys and uses a hash function for fast data access.

Hash Table



hash.cpp U

code > hash.cpp > main()

```
53          }
54      }
55  };
56
57
58  int main (){
59      Hash table;
60
61      table.add(10, 100);
62      table.add(20, 200);
63      table.add(30, 300);
64      table.add(25, 250);
65      table.add(27, 77);
66
67
68      table.print();
69  }
70
```

PROBLEMS DEBUG CONSOLE OUTPUT PORTS SERIAL MONITOR COMMENTS TERMINAL

powershell - code

```
0: [10: 100] -> [20: 200] -> [30: 300] -> null
1: null
2: null
3: null
4: null
5: [25: 250] -> null
6: null
7: [27: 77] -> null
8: null
9: null
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code>
```

```cpp
// hash is a data structure that maps keys to values for efficient data retrieval using hash functions.
#include <iostream>
using namespace std;

struct Node{
    int key;
    int value;
    Node* next;

    // constructor
    Node(int k, int v): key(k), value(v), next(nullptr) {}
};
```

```
14  struct Hash{
15      static const int S = 10;
16      Node* table[S];
17
18      Hash(){
19          for (int i =0; i < S; i++){
20              table[i] = nullptr;
21          }
22      }
23
24      // hash fun
25      int hashFunction(int key){
26          return key % S;
27      }
```

```
29      void add( int key , int value ){
30          int index = hashFunction(key); //index for the key
31          Node* n = new Node(key, value);
32
33          if (table[index] == nullptr){
34              table[index] = n;
35          } else {
36              Node* t = table[index];
37              while (t -> next != nullptr){
38                  t = t -> next;
39              }
40              t -> next = n;
41          }
42      }
```

```
44      void print(){
45          for (int i = 0; i < S; i++){
46              cout << i << ": ";
47              Node* t = table[i];
48              while(t != nullptr){
49                  cout << "[" << t-> key << ": " << t-> value << "] -> ";
50                  t = t -> next;
51              }
52              cout << "null" << endl;
53          }
54      }
55  };
```

```
58  int main (){
59      Hash table;
60
61      table.add(10, 100);
62      table.add(20, 200);
63      table.add(30, 300);
64      table.add(25, 250);
65      table.add(27, 77);
66
67
68      table.print();
69  }
```

```
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code> ./hash.exe
0: [10: 100] -> [20: 200] -> [30: 300] -> null
1: null
2: null
3: null
4: null
5: [25: 250] -> null
6: null
7: [27: 77] -> null
8: null
9: null
PS D:\Documents\GitHub\Introduction-to-Data-Structure\code>
```

Last Video

Heaps and Priority Queues