|  |  |
| --- | --- |
| **Swap nods in pairs in C++** | |
| #include <iostream>  struct Node {  int val;  Node\* next;  Node(int x) {  val = x;  next = nullptr;  }  };  class SwapNodesInPairs {  public:  Node\* swapPairs(Node\* head) {  Node dummy(0);  dummy.next = head;  Node\* current = &dummy;  while (current->next != nullptr && current->next->next != nullptr) {  Node\* first = current->next;  Node\* second = current->next->next;  first->next = second->next;  second->next = first;  current->next = second;  current = first;  }  return dummy.next;  }  static void printList(Node\* head) {  while (head != nullptr) {  std::cout << head->val << " -> ";  head = head->next;  }  std::cout << "null" << std::endl;  }  };  int main() {  SwapNodesInPairs solution;  Node\* head = new Node(1);  head->next = new Node(2);  head->next->next = new Node(3);  head->next->next->next = new Node(4);  Node\* result = solution.swapPairs(head);  SwapNodesInPairs::printList(result);  // Free the allocated memory  Node\* curr = result;  while (curr != nullptr) {  Node\* temp = curr;  curr = curr->next;  delete temp;  }  return 0;  } | for input:  1 -> 2 -> 3 -> 4  The goal is to swap every two adjacent nodes. So, the expected output is:  2 -> 1 -> 4 -> 3  **🧠 Key Pointers:**   * dummy is a placeholder node that simplifies head manipulation. * current starts at dummy. * first and second are the two nodes to be swapped. * The loop continues as long as there are at least 2 nodes ahead of current.   **🔁 Dry Run Table:**   | **Iteration** | **current Points To** | **first** | **second** | **Operation** | **List After Swap** | | --- | --- | --- | --- | --- | --- | | 1 | dummy (0) → 1 | 1 | 2 | Swap 1 and 2 | 2 → 1 → 3 → 4 | |  |  |  |  | first->next = 3 |  | |  |  |  |  | second->next = 1, current->next = 2 |  | |  |  |  |  | current = first → moves to node 1 |  | | 2 | current → 1 | 3 | 4 | Swap 3 and 4 | 2 → 1 → 4 → 3 | |  |  |  |  | first->next = nullptr |  | |  |  |  |  | second->next = 3, current->next = 4 |  | |  |  |  |  | current = first → moves to node 3 |  |   **✅ Final Output:**  2 -> 1 -> 4 -> 3 -> null |
| Output:- 2 -> 1 -> 4 -> 3 -> null | |