|  |  |
| --- | --- |
| **All palindromic partition in C++** | |
| #include <iostream>  #include <string>  using namespace std;  class AllPalindromicPartition {  public:      static void main() {          string str = "abba";          sol(str, "");      }      static void sol(string str, string asf) {          if (str.length() == 0) {              cout << asf << endl;              return;          }          for (int i = 0; i < str.length(); i++) {              string prefix = str.substr(0, i + 1);              string ros = str.substr(i + 1);              if (isPalin(prefix)) {                  sol(ros, asf + "(" + prefix + ")");              }          }      }      static bool isPalin(string s) {          int li = 0;          int ri = s.length() - 1;          while (li < ri) {              if (s[li] != s[ri]) {                  return false;              }              li++;              ri--;          }          return true;      }  };  int main() {      AllPalindromicPartition::main();      return 0;  } | Dry Run for Input "abba" We will track the recursive calls with:   * str: Remaining string to process * prefix: Currently selected prefix * ros: Remaining string after prefix * asf: Accumulated string so far * Action: What's happening  | **Step** | **str** | **prefix** | **ros** | **asf** | **Action / Reason** | | --- | --- | --- | --- | --- | --- | | 1 | abba | a | bba | (a) | 'a' is palindrome → recurse | | 2 | bba | b | ba | (a)(b) | 'b' is palindrome → recurse | | 3 | ba | b | a | (a)(b)(b) | 'b' is palindrome → recurse | | 4 | a | a | "" | (a)(b)(b)(a) | ✅ 'a' is palindrome → print | | 5 | ba | ba | — | — | not palindrome ✖️ | | 6 | bba | bb | a | (a)(bb) | ✅ 'bb' is palindrome → recurse | | 7 | a | a | "" | (a)(bb)(a) | ✅ 'a' is palindrome → print | | 8 | bba | bba | — | — | not palindrome ✖️ | | 9 | abba | ab | — | — | not palindrome ✖️ | | 10 | abba | abb | — | — | not palindrome ✖️ | | 11 | abba | abba | "" | (abba) | ✅ 'abba' is palindrome → print |  ✅ Final Output (a)(b)(b)(a)  (a)(bb)(a)  (abba) |
| Output:-  (a)(b)(b)(a)  (a)(bb)(a)  (abba) | |