Practical C++ Code Examples

Constructing a String Using Character Array

```
char name[] = "James";

char name[] = {'J', 'a', 'm', 'e', 's', '\0'};

char name[6] = "James";

char name[6] = {'J', 'a', 'm', 'e', 's', '\0'};
```

Checking if Two Strings are Anagrams

```
#include <algorithm>

#include <string>

#include <iostream>

using namespace std;


bool IsAnagram(string str1, string str2) {

    transform(str1.begin(), str1.end(), str1.begin(), ::toupper);

    transform(str2.begin(), str2.end(), str2.begin(), ::toupper);


    str1.erase(remove(str1.begin(), str1.end(), ' '), str1.end());

    str2.erase(remove(str2.begin(), str2.end(), ' '), str2.end());


    sort(str1.begin(), str1.end());

    sort(str2.begin(), str2.end());


    return str1 == str2;

}
```

```
int main() {
    cout << "Anagram Checker\n";

    string string1, string2;
    cout << "Input string 1 -> ";
    getline(cin, string1);
    cout << "Input string 2 -> ";
    getline(cin, string2);

    if (IsAnagram(string1, string2)) {
        cout << "The strings are anagrams.\n";
    } else {
        cout << "The strings are NOT anagrams.\n";
    }
    return 0;
}
```

Checking if a String is a Palindrome

```
#include <algorithm>
#include <string>
#include <iostream>
using namespace std;

bool IsPalindrome(string str) {
    transform(str.begin(), str.end(), str.begin(), ::toupper);
```

```
    str.erase(remove(str.begin(), str.end(), ' '), str.end());

    return equal(str.begin(), str.begin() + str.size() / 2, str.rbegin());
}

int main() {
    cout << "Palindrome Checker\n";

    string str;
    cout << "Input string -> ";
    getline(cin, str);

    if (IsPalindrome(str)) {
        cout << "The string is a palindrome.\n";
    } else {
        cout << "The string is NOT a palindrome.\n";
    }
    return 0;
}
```

Converting Decimal to Binary String

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
```

```
string DecimalToBinaryString(int decimalNumber) {
    string binaryString = "";
    if (decimalNumber > 0) {
        div_t dv{};
        dv.quot = decimalNumber;
        do {
            dv = div(dv.quot, 2);
            binaryString = to_string(dv.rem) + binaryString;
        } while (dv.quot);
    }
    return binaryString.empty() ? "0" : binaryString;
}

int main() {
    cout << "Decimal to Binary Converter\n";

    int decNum;
    cout << "Input decimal number -> ";
    cin >> decNum;

    cout << "Binary representation: " << DecimalToBinaryString(decNum) << endl;
    return 0;
}
```

Converting Binary String to Decimal

```
#include <iostream>
```

```
#include <string>
#include <cmath>
using namespace std;

int BinaryStringToDecimal(string binaryString) {
    int decimal = 0, power = 0;
    for (auto it = binaryString.rbegin(); it != binaryString.rend(); ++it, ++power) {
        if (*it == '1') {
            decimal += pow(2, power);
        }
    }
    return decimal;
}

int main() {
    cout << "Binary to Decimal Converter\n";

    string binaryString;
    cout << "Input binary string -> ";
    cin >> binaryString;

    cout << "Decimal representation: " << BinaryStringToDecimal(binaryString) << endl;
    return 0;
}
```

Generating Subsequences of a String

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>
using namespace std;

vector<string> GenerateSubsequences(string str) {
    vector<string> subsequences;
    int n = str.size();
    int total = pow(2, n);

    for (int i = 1; i < total; ++i) {
        string subsequence = "";
        for (int j = 0; j < n; ++j) {
            if (i & (1 << j)) {
                subsequence += str[j];
            }
        }
        subsequences.push_back(subsequence);
    }
    return subsequences;
}

int main() {
    cout << "Subsequence Generator\n";

    string str;
```

```
    cout << "Input string -> ";
    cin >> str;

    auto subsequences = GenerateSubsequences(str);
    for (const auto& sub : subsequences) {
        cout << sub << endl;
    }
    return 0;
}
```

Checking if a String is a Subsequence of Another

```
#include <iostream>
#include <string>
using namespace std;

bool IsSubSequence(string str1, string str2, int x, int y) {
    if (x == 0) return true;
    if (y == 0) return false;

    if (str1[x - 1] == str2[y - 1]) {
        return IsSubSequence(str1, str2, x - 1, y - 1);
    }
    return IsSubSequence(str1, str2, x, y - 1);
}

int main() {
```

```
    cout << "Subsequence Checker\n";


    string string1, string2;
    cout << "Input 1st string -> ";
    getline(cin, string1);
    cout << "Input 2nd string -> ";
    getline(cin, string2);


    if (IsSubSequence(string1, string2, string1.size(), string2.size())) {
        cout << "The first string is a subsequence of the second.\n";
    } else {
        cout << "The first string is NOT a subsequence of the second.\n";
    }
    return 0;
}
```

Pattern Searching in a String

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;


vector<int> SearchPattern(string text, string pattern) {
    vector<int> indices;
    int n = text.size(), m = pattern.size();
```

```
    if (m <= n) {
        for (int i = 0; i <= n - m; ++i) {
            int j;
            for (j = 0; j < m; ++j) {
                if (text[i + j] != pattern[j]) break;
            }
            if (j == m) indices.push_back(i);
        }
    }
    return indices;
}

int main() {
    cout << "Pattern Searching\n";

    string text, pattern;
    cout << "Input text -> ";
    getline(cin, text);
    cout << "Input pattern -> ";
    getline(cin, pattern);

    auto indices = SearchPattern(text, pattern);
    if (!indices.empty()) {
        cout << "Pattern found at indices: ";
        for (auto idx : indices) {
            cout << idx << " ";
        }
```

```
        cout << endl;
    } else {
        cout << "Pattern not found.\n";
    }
    return 0;
}
```