МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Построение и анализ алгоритмов»

Тема: Алгоритм Кнута-Морриса-Пратта

Студентка гр. 9382 _____ Пя С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2021

Цель работы.

Разработать программу, реализующую поиск подстроки в строке с помощью алгоритма Кнута-Морриса-Пратта. Также разработать программу, определяющую, является ли первая строка циклическим сдвигом второй строки.

Задание 1

Реализуйте алгоритм КМП и с его помощью для заданных шаблона P ($|P| \leq 15000$) и текста T ($|T| \leq 5000000$) найдите все вхождения P в T.

Вход:

Первая строка - P

Вторая строка - T

Выход:

индексы начал вхождений P в T, разделенных запятой, если P не входит в T, то вывести -1

Sample Input:

ab

abab

Sample Output:

0,2

Задание 2

Заданы две строки A ($|A| \leq 5000000$) и B ($|B| \leq 5000000$).

Определить, является ли A циклическим сдвигом B (это значит, что A и B имеют одинаковую длину и A состоит из суффикса B, склеенного с префиксом B). Например, defabc является циклическим сдвигом abcdef.

Вход:

Первая строка - A

Вторая строка - B

Выход:

Если A вляется циклическим сдвигом B, индекс начала строки B в A, иначе вывести −1. Если возможно несколько сдвигов вывести первый индекс.

Sample Input:

defabc

abcdef

Sample Output:

3

Описание алгоритма

На вход подаются подстрока, которую нужно найти в строке, и строка. Алгоритм начинается с вычисления префикс функции для подстроки. Для этого создается список для длин префикс-функций, по величине равный количеству символов в подстроке. Префикс-функция для i-го символа образа – максимальная длина совпадающих префикса и суффикса подстроки в образе, которая заканчивается i-м символом. Для первого элемента префикс-функция будет равна нулю, так как подстрока будет равна единице.

В первом задании необходимо найти все вхождения подстроки в строке. Сравниваются символ из подстроки и символ из строки.

В случае их равенства мы начинаем сравнивать следующий символ подстроки и следующий символ строки, если к этому моменту символы подстроки были полностью рассмотрены, значит подстрока была найдена в строке, в ответ записывается индекс начала вхождения подстроки в строку, а рассматривать дальше мы будем сравнивать символ подстроки под индексом префикс-функции предыдущего символа, так как подстрока может сливаться с подстрокой при поиске в строке.

В случае неравенства символов, если рассматриваемый символ является первым в подстроке, берем для сравнения следующий символ строки.

В случае неравенства символов, если рассматриваемый символ не является первым в подстроке, следующим рассматриваемым символом в подстроке станет символ под индексом префикс-функции предыдущего символа, чтобы мы учли то, что в рассмотренных символах могла быть часть подстроки.

Алгоритм заканчивается, когда мы сравним все символы строки.

Во втором задании необходимо определить, является ли первая строка циклическим сдвигом второй. Для этого достаточно воспользоваться алгоритмом из предыдущего задания, выполняя поиск второй строки в удвоенной первой. Так как при сложении строк первая будет содержать в себе вторую строку, если первая строка является циклическим сдвигом второй. Алгоритм заканчивается при нахождении первого вхождения второй строки в модифицированной первой либо при сравнении всех символов модифицированной первой строки и не нахождении вхождения. Если данные строки изначально не равны, то по определению первая строка не является циклическим сдвигом второй.

Оценка сложности по памяти.

В обоих программах проходимся по двум строкам и используем вектор со значениями префикс-функции для подстроки. В худшем случае мы не найдем подстроку в строке, тогда сложность составляет O(2*N + M) для первой программы, где N – длина первой строки или размер вектора префикс-функции, M – длина второй строки, и сложность для второй программы составляет O(N + M*2*2), так как в худшем случае мы пройдем по всей удвоенной строке.

Оценка сложности по времени.

Значение префикс-функции вычисляется за O(N) сравнений, где N – длина подстроки, так как необходимо обойти всю строку, чтобы определить префикс-функцию.

Поиск подстроки в строке с помощью алгоритма КМП будет занимать O(M), где M – длина строки, так как вся строка будет пройдена один раз. Каждый символ строки будет рассмотрен один раз благодаря префикс-функции.

Итоговая оценка – O(N + M).

Для второй программы оценка сложности по времени вычисляется так же.

Тестирование.

Тестирование первой программы.

Нумерация	Входные данные	Выходные данные
1	abbaabbab abbaabbaabbaabbababbaabbab	Хотите считать данные из файла или ввести самостоятельно?(1/2) 1 Начинаем подсчет префикс функции i - индекс первого символа для сравнения, j - индекс второго символа для сравнения Символы не одинаковы, смещаем i "a""b"baabbab 0 0 Символы не одинаковы, смещаем i "a"b"b"aabbab 0 0 0 Символы одинаковы, смещаем j и i "a"bb"a"abbab 0 0 0 1 j не равен нулю, и символы не одинаковы

		Присвоим j значение префикса предыдущего символа, на который указывала j a"b"ba"a"bbab j = 0 Символы одинаковы, смещаем j и i "a"bba"a"bbab 0 0 0 1 1 Символы одинаковы, смещаем j и i a"b"baa"b"bab 0 0 0 1 1 2 Символы одинаковы, смещаем j и i ab"b"aab"b"ab 0 0 0 1 1 2 3 Символы одинаковы, смещаем j и i abb"a"abb"a"b 0 0 0 1 1 2 3 4 j не равен нулю, и символы не одинаковы Присвоим j значение префикса предыдущего символа, на который указывала j abba"a"bba"b" j = 1 Символы одинаковы, смещаем j и i a"b"baabba"b" 0 0 0 1 1 2 3 4 2 Вывод префикс функции a b b a a b b a b 0 0 0 1 1 2 3 4 2 Начинаем поиск подстоки в строке k - индекс символа строки для сравнения, l - индекс символа подстроки для сравнения

		Символы одинаковы, смещаем k и l "a"bbaabbaabbaabbababbaabbab "a"bbaabbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l a"b"baabbaabbaabbababbaabbab a"b"baabbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l ab"b"aabbaabbaabbababbaabbab ab"b"aabbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abb"a"abbaabbaabbababbaabbab abb"a"abbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abba"a"bbaabbaabbababbaabbab abba"a"bbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaa"b"baabbaabbababbaabbab abbaa"b"bab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaab"b"aabbaabbababbaabbab abbaab"b"ab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabb"a"abbaabbababbaabbab abbaabb"a"b 0 0 0 1 1 2 3 4 2 Символы не одинаковы

		Присваиваем l значение префикса предыдущего символа, на который указывала l abbaabba"a"bbaabbababbaabbab abbaabba"b" 0 0 0 1 1 2 3 4 2 l = 4 Символы одинаковы, смещаем k и l abbaabba"a"bbaabbababbaabbab abba"a"bbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaa"b"baabbababbaabbab abbaa"b"bab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaab"b"aabbababbaabbab abbaab"b"ab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabb"a"abbababbaabbab abbaabb"a"b 0 0 0 1 1 2 3 4 2 Символы не одинаковы Присваиваем l значение префикса предыдущего символа, на который указывала l abbaabbaabba"a"bbababbaabbab abbaabba"b" 0 0 0 1 1 2 3 4 2 l = 4 Символы одинаковы, смещаем k и l abbaabbaabba"a"bbababbaabbab

		abba"a"bbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaa"b"bababbaabbab abbaa"b"bab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaab"b"ababbaabbab abbaab"b"ab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabb"a"babbaabbab abbaabb"a"b 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabba"b"abbaabbab abbaabba"b" 0 0 0 1 1 2 3 4 2 Подстрока найдена! Индекс начала вхождения подстроки в строке 8 abbaabbaabbaabbab"a"bbaabbab abbaabbab 0 0 0 1 1 2 3 4 2 Присваиваем l значение префикс-функции предыдущего символа, на который указывала l l = 2 Символы не одинаковы Присваиваем l значение префикса предыдущего символа, на который указывала l abbaabbaabbaabbab"a"bbaabbab

		ab"b"aabbab 0 0 0 1 1 2 3 4 2 l = 0 Символы одинаковы, смещаем k и l abbaabbaabbaabbab"a"bbaabbab "a"bbaabbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbaba"b"baabbab a"b"baabbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbabab"b"aabbab ab"b"aabbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababb"a"abbab abb"a"abbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababba"a"bbab abba"a"bbab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababbaa"b"bab abbaa"b"bab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababbaab"b"ab abbaab"b"ab 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababbaabb"a"b

		abbaabb"a"b 0 0 0 1 1 2 3 4 2 Символы одинаковы, смещаем k и l abbaabbaabbaabbababbaabba"b" abbaabba"b" 0 0 0 1 1 2 3 4 2 Подстрока найдена! Индекс начала вхождения подстроки в строке 17 abbaabbaabbaabbababbaabbab abbaabbab 0 0 0 1 1 2 3 4 2 Присваиваем l значение префикс-функции предыдущего символа, на который указывала l l = 2 Подстрока входит в строку Индексы начала вхождения подстроки в строку 8,17 Хотите продолжить?(y/n) n
2	abcasda sadfasf	Подстрока не входит в строку -1
3	asddsad asd	Подстрока не входит в строку -1
4	qwe qwerty	Подстрока не входит в строку 0
5	love lovelovelovelovelovelovelveovevloe	Индексы начала вхождения подстроки в строку 0,4,8,12,16,20

Тестирование второй программы.

Нумерация	Входные данные	Выходные данные
1	abbaabbab abbababba	Хотите считать данные из файла или ввести самостоятельно?(1/2) Для того чтобы определить, является ли A циклическим сдвигом B, воспользуемся алгоритмом поиска подстроки в строке B нашем случае будем искать строку B в модифицированной A строке Модифицируем строку A, сложив ее с собой Начинаем подсчет префикс функции i - индекс первого символа для сравнения, j - индекс второго символа для сравнения Символы не одинаковы, смещаем i "a""b"bababba 0 0 Символы не одинаковы, смещаем i "a"b"b"ababba 0 0 0 Символы одинаковы, смещаем j и i "a"bb"a"babba 0 0 0 1 Символы одинаковы, смещаем j и i a"b"ba"b"abba 0 0 0 1 2 j не равен нулю, и символы не одинаковы Присвоим j значение префикса предыдущего символа, на который указывала j ab"b"ab"a"bba j = 0

		Символы одинаковы, смещаем j и i "a"bbab"a"bba 0 0 0 1 2 1 Символы одинаковы, смещаем j и i a"b"baba"b"ba 0 0 0 1 2 1 2 Символы одинаковы, смещаем j и i ab"b"abab"b"a 0 0 0 1 2 1 2 3 Символы одинаковы, смещаем j и i abb"a"babb"a" 0 0 0 1 2 1 2 3 4 Вывод префикс функции a b b a b a b b a 0 0 0 1 2 1 2 3 4 Определяем, является ли A циклическим сдвигом B k - индекс символа строки A, l - индекс символа строки B Символы одинаковы, смещаем k и l "a"bbaabbababbaabbab "a"bbababba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l a"b"baabbababbaabbab a"b"bababba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l ab"b"aabbababbaabbab ab"b"ababba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abb"a"abbababbaabbab abb"a"babba 0 0 0 1 2 1 2 3 4

		Символы не одинаковы Присваиваем l значение префикса предыдущего символа, на который указывала l abba"a"bbababbaabbab abba"b"abba 0 0 0 1 2 1 2 3 4 l = 1 Символы не одинаковы Присваиваем l значение префикса предыдущего символа, на который указывала l abba"a"bbababbaabbab a"b"bababba 0 0 0 1 2 1 2 3 4 l = 0 Символы одинаковы, смещаем k и l abba"a"bbababbaabbab "a"bbababba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaa"b"bababbaabbab a"b"bababba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaab"b"ababbaabbab ab"b"ababba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaabb"a"babbaabbab abb"a"babba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaabba"b"abbaabbab abba"b"abba

		0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaabbab"a"bbaabbab abbab"a"bba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaabbaba"b"baabbab abbaba"b"ba 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaabbabab"b"aabbab abbabab"b"a 0 0 0 1 2 1 2 3 4 Символы одинаковы, смещаем k и l abbaabbababb"a"abbab abbababb"a" 0 0 0 1 2 1 2 3 4 B в модифицированной A найдена! Индекс начала строки B в A 4 abbaabbababba"a"bbab abbababba 0 0 0 1 2 1 2 3 4 B является циклическим сдвигом A Индекс начала B в A 4 Хотите продолжить?(y/n) n
2	asd adff	B не является циклическим сдвигом A -1
3	iloveyou youlovei	B не является циклическим сдвигом A -1
4	iloveyou	Индекс начала B в A

	youilove	5
5	abcdef fabcde	Индекс начала B в A 6

Выводы.

Была разработана программа, занимающаяся поиском подстроки в строке и находящая все ее вхождения, и был изучен алгоритм Кнута-Морриса-Пратта. Также была реализована программа, определяющая, является ли строка циклическим сдвигом другой строки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab4_1.cpp

```
#include <iostream>
#include <vector>
#include <fstream>

class KMP {
private:
    std::string P, T;
    std::vector<int> answer;
    std::vector<size_t> prefix;
public:
    void readData(std::istream& fin){
        fin >> P >> T;
    }

    void writePrefix(bool withP, int n) {
        if (withP) {
            std::cout << "Вывод префикс функции\n";
            for (auto i : P) {
                std::cout << i << " ";
            }
            std::cout << "\n";
        }
        for (int i = 0; i < n;i++) {
            std::cout << prefix[i] << " ";
        }
        std::cout << "\n";
    }

    void writeStepForPorT(int j, int i, bool isP) {
        std::string temp = P;
        if (!isP)
            temp = T;
        for (int l = 0; l < temp.length(); l++) {
            if (l == j || l == i)
                std::cout << "\"" << temp[l] << "\"";
            else
                std::cout << temp[l];
        }
        std::cout << "\n";
    }

    void calculatePrefix() {
        std::cout << "Начинаем подсчет префикс функции\n";
        std::cout << "i - индекс первого символа для сравнения, j - индекс
второго символа для сравнения\n";
        size_t n = P.length();
        prefix = std::vector<size_t>(n);
        for (size_t i = 1; i < n; ++i) {
            size_t j = prefix[i - 1];
            while ((j > 0) && (P[i] != P[j])) {
                std::cout << "j не равен нулю, и символы не одинаковы\n";
                std::cout << "Присвоим j значение префикса предыдущего
символа, на который указывала j\n";
                writeStepForPorT(j, i, true);
                j = prefix[j - 1];
```

```
                std::cout << "j = " << j << "\n";
            }
            if (P[i] == P[j]) {
                std::cout << "Символы одинаковы, смещаем j и i\n";
                writeStepForPorT(j, i, true);
                ++j;
            } else {
                std::cout << "Символы не одинаковы, смещаем i\n";
                writeStepForPorT(j, i, true);
            }
            prefix[i] = j;
            writePrefix(false, i + 1);
        }
        writePrefix(true, n);
    }

    void writeVisualisation(int k, int l) {
        writeStepForPorT(k, k, false);
        writeStepForPorT(l, l, true);
        writePrefix(false, prefix.size());
    }
    void doAlgoritm() {
        int k = 0, l = 0;
        size_t n = P.length(), m = T.length();
        calculatePrefix();
        std::cout << "Начинаем поиск подстоки в строке\n";
        std::cout << "k - индекс символа строки для сравнения, l - индекс
символа подстроки для сравнения\n";
        while (k < m) {
            if (T[k] == P[l]) {
                std::cout << "Символы одинаковы, смещаем k и l\n";
                writeVisualisation(k, l);
                k++;
                l++;
                if (l == n) {
                    std::cout << "Подстрока найдена!\nИндекс начала вхождения
подстроки в строке " << k - l << "\n";
                    writeVisualisation(k, l);
                    answer.push_back(k - l);
                    std::cout << "Присваиваем l значение префикс-функции
предыдущего символа, на который указывала l\n";
                    l = prefix[l - 1];
                    std::cout << "l = " << l << "\n";
                }
            } else if (l == 0) {
                std::cout << "Символы не одинаковы, l = 0, смещаем k\n";
                writeVisualisation(k, l);
                k++;
            } else {
                std::cout << "Символы не одинаковы\n";
                std::cout << "Присваиваем l значение префикса предыдущего
символа, на который указывала l\n";
                writeVisualisation(k, l);
                l = prefix[l - 1];
                std::cout << "l = " << l << "\n";
            }
        }
    }

    void writeAnswer() {
        if (answer.empty()) {
```

```
            std::cout << "Подстрока не входит в строку\n";
            std::cout << "-1";
            return;
        }
        std::cout << "Подстрока входит в строку\nИндексы начала вхождения
подстроки в строку\n";
        for (auto i : answer) {
            if (i == answer.front())
                std::cout << i;
            else
                std::cout << "," << i;
        }
        std::cout << "\n";
    }
};


void startProgram() {
    char answ = 'y';
    while (answ == 'y') {
        std::cout << "Хотите считать данные из файла или ввести
самостоятельно?(1/2)\n";
        std::cin >> answ;
        KMP *kmp = new KMP();
        if (answ == '2') {
            std::cout << "Введите подстроку и строку\n";
            kmp->readData(std::cin);
        } else {
            std::ifstream fin("test1.txt");
            kmp->readData(fin);
            fin.close();
        }
        kmp->doAlgoritm();
        kmp->writeAnswer();
        std::cout << "Хотите продолжить?(y/n)\n";
        std::cin >> answ;
    }
}

int main() {
    startProgram();
    return 0;
}
```

Название файла: lab4_2.cpp

```
#include <iostream>
#include <vector>
#include <fstream>
class KMP {
private:
    std::string B, A;
    int answer;
    std::vector<size_t> prefix;
public:
    void readData(std::istream& fin){
        fin >> A >> B;
        std::cout << "Для того чтобы определить, является ли A циклическим
сдвигом B, воспользуемся алгоритмом поиска подстроки в строке\n";
        std::cout << "В нашем случае будем искать строку B в модифицированной
A строке\n";
        std::cout << "Модифицируем строку A, сложив ее с собой\n";
        A = A + A;
    }

    void writePrefix(bool withP, int n) {
        if (withP) {
            std::cout << "Вывод префикс функции\n";
            for (auto i : B) {
                std::cout << i << " ";
            }
            std::cout << "\n";
        }
        for (int i = 0; i < n;i++) {
            std::cout << prefix[i] << " ";
        }
        std::cout << "\n";
    }

    void writeStepForPorT(int j, int i, bool isP) {
        std::string temp = B;
        if (!isP)
            temp = A;
        for (int l = 0; l < temp.length(); l++) {
            if (l == j || l == i)
                std::cout << "\"" << temp[l] << "\"";
            else
                std::cout << temp[l];
        }
        std::cout << "\n";
    }

    void calculatePrefix() {
        std::cout << "Начинаем подсчет префикс функции\n";
        std::cout << "i - индекс первого символа для сравнения, j - индекс
второго символа для сравнения\n";
        size_t n = B.length();
        prefix = std::vector<size_t>(n);
        for (size_t i = 1; i < n; ++i) {
            size_t j = prefix[i - 1];
            while ((j > 0) && (B[i] != B[j])) {
                std::cout << "j не равен нулю, и символы не одинаковы\n";
                std::cout << "Присвоим j значение префикса предыдущего
символа, на который указывала j\n";
                writeStepForPorT(j, i, true);
```

```
                j = prefix[j - 1];
                std::cout << "j = " << j << "\n";
            }
            if (B[i] == B[j]) {
                std::cout << "Символы одинаковы, смещаем j и i\n";
                writeStepForPorT(j, i, true);
                ++j;
            } else {
                std::cout << "Символы не одинаковы, смещаем i\n";
                writeStepForPorT(j, i, true);
            }
            prefix[i] = j;
            writePrefix(false, i + 1);
        }
        writePrefix(true, n);
    }

    void writeVisualisation(int k, int l) {
        writeStepForPorT(k, k, false);
        writeStepForPorT(l, l, true);
        writePrefix(false, prefix.size());
    }
    void doAlgoritm() {
        int k = 0, l = 0;
        answer = -1;
        size_t n = B.length(), m = A.length();
        if (n != m / 2) {
            std::cout << "B и A не имеют одинаковую длину\n";
            return;
        }
        calculatePrefix();
        std::cout << "Определяем, является ли A циклическим сдвигом B\n";
        std::cout << "k - индекс символа строки A, l - индекс символа строки
B\n";
        while (k < m) {
            if (A[k] == B[l]) {
                std::cout << "Символы одинаковы, смещаем k и l\n";
                writeVisualisation(k, l);
                k++;
                l++;
                if (l == n) {
                    std::cout << "B в модифицированной A найдена!\nИндекс
начала строки B в A " << k - l << "\n";
                    writeVisualisation(k, l);
                    answer = k - l;
                    return;
                }
            } else if (l == 0) {
                std::cout << "Символы не одинаковы, l = 0, смещаем k\n";
                writeVisualisation(k, l);
                k++;
            } else {
                std::cout << "Символы не одинаковы\n";
                std::cout << "Присваиваем l значение префикса предыдущего
символа, на который указывала l\n";
                writeVisualisation(k, l);
                l = prefix[l - 1];
                std::cout << "l = " << l << "\n";
            }
        }
    }
```

```
    void writeAnswer() {
        if (answer == -1) {
            std::cout << "B не является циклическим сдвигом A\n";
        } else {
            std::cout << "B является циклическим сдвигом A\nИндекс начала B в
A\n";
        }
        std::cout << answer << "\n";
    }
};

void startProgram() {
    char answ = 'y';
    while (answ == 'y') {
        std::cout << "Хотите считать данные из файла или ввести
самостоятельно?(1/2)\n";
        std::cin >> answ;
        KMP *kmp = new KMP();
        if (answ == '2') {
            std::cout << "Введите A и B\n";
            kmp->readData(std::cin);
        } else {
            std::ifstream fin("test2.txt");
            kmp->readData(fin);
            fin.close();
        }
        kmp->doAlgoritm();
        kmp->writeAnswer();
        std::cout << "Хотите продолжить?(y/n)\n";
        std::cin >> answ;
    }
}

int main() {
    startProgram();
    return 0;
}
```