Министерство общего и профессионального

образования Российской Федерации

Пермский государственный технический университет

Лабораторная работа

"Методы поиска"

Выполнила студентка группы РИС 23-3б:

Федорова О.И.

Проверила

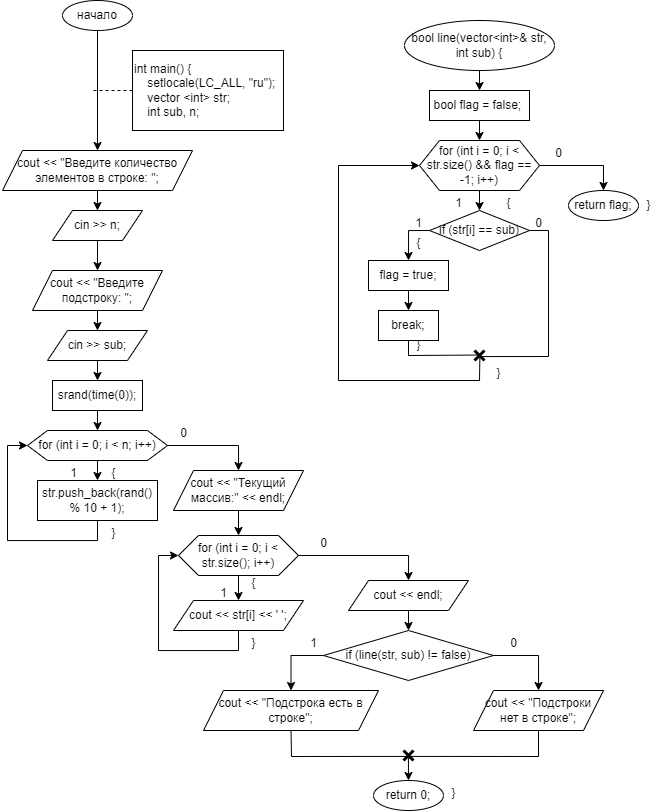
Доцент кафедры ИТАС

Полякова О.А.

Пермь 2024

Постановка задачи: реализовать поиск подстроки в строке с помощью методов: линейного, бинарного, интерполяционного и прямого поисков.

**Линейный поиск**

1. Анализ задачи:
2. Создать функцию, которая будет сравнивать каждый элемент с ключевым и реализовать в потоке.
3. При нахождении подстроки в строке, цикл будет прерываться, и функция возвратит значение “true”.
4. В функции main пользователь вводит требующиеся данные, после чего массив заполняется рандомными числами.
5. После вызова функции, с помощью проверки результата работы функции, в консоль выведется ответ.
6. Блок-схема:
7. Код программы:

#include <iostream>

#include <string>

#include <vector>

using namespace std;

bool line(vector<int>& str, int sub) { // линейный поиск

bool flag = false;

for (int i = 0; i < str.size() && flag == -1; i++) {

if (str[i] == sub) { // если подстрока найдена

flag = true; // смена флага для вывода ответа

break; // выход из цикла

}

}

return flag;

}

int main() {

setlocale(LC\_ALL, "ru");

vector <int> str;

int sub, n;

cout << "Введите количество элементов в строке: ";

cin >> n;

cout << "Введите подстроку: ";

cin >> sub;

srand(time(0));

for (int i = 0; i < n; i++) {

str.push\_back(rand() % 10 + 1); // заполнение массива рандомными числами

}

cout << "Текущий массив:" << endl;

for (int i = 0; i < str.size(); i++) { // вывод массива

cout << str[i] << ' ';

}

cout << endl;

if (line(str, sub) != false) {

cout << "Подстрока есть в строке";

}

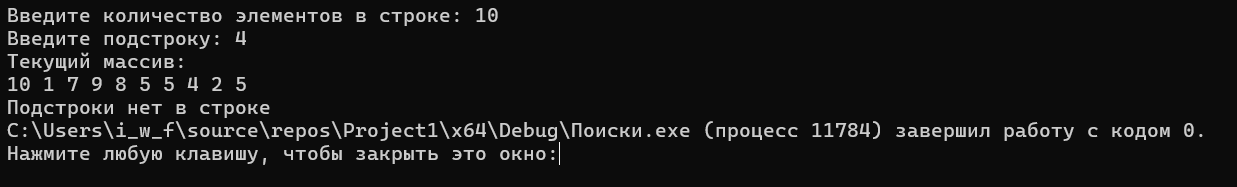
else {

cout << "Подстроки нет в строке";

}

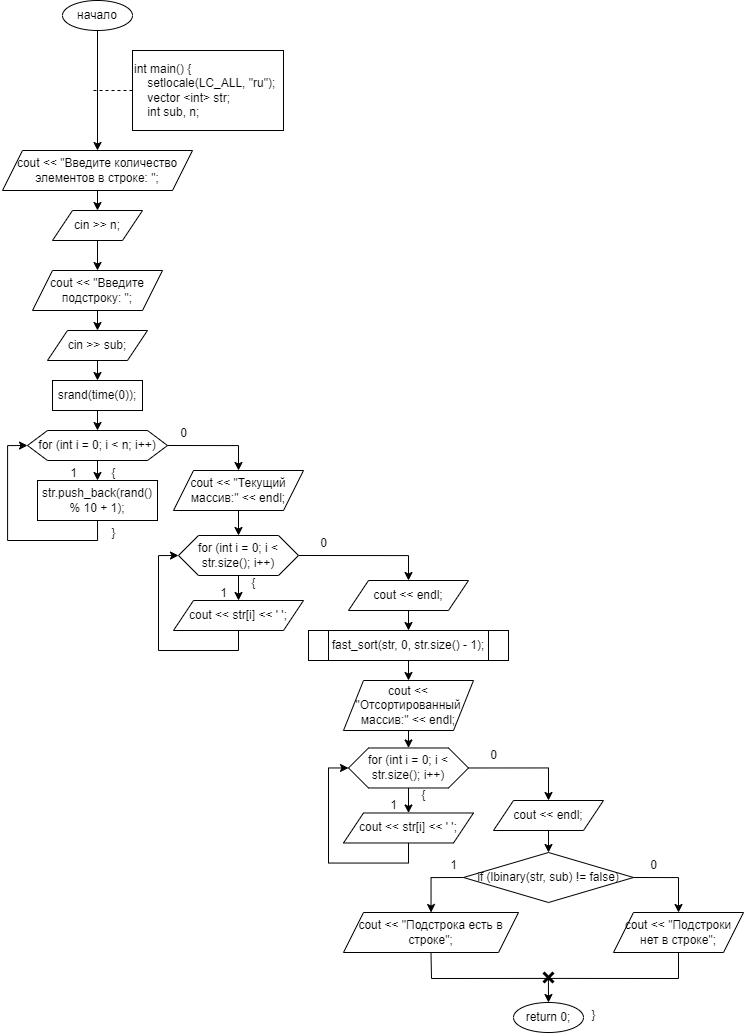
return 0;

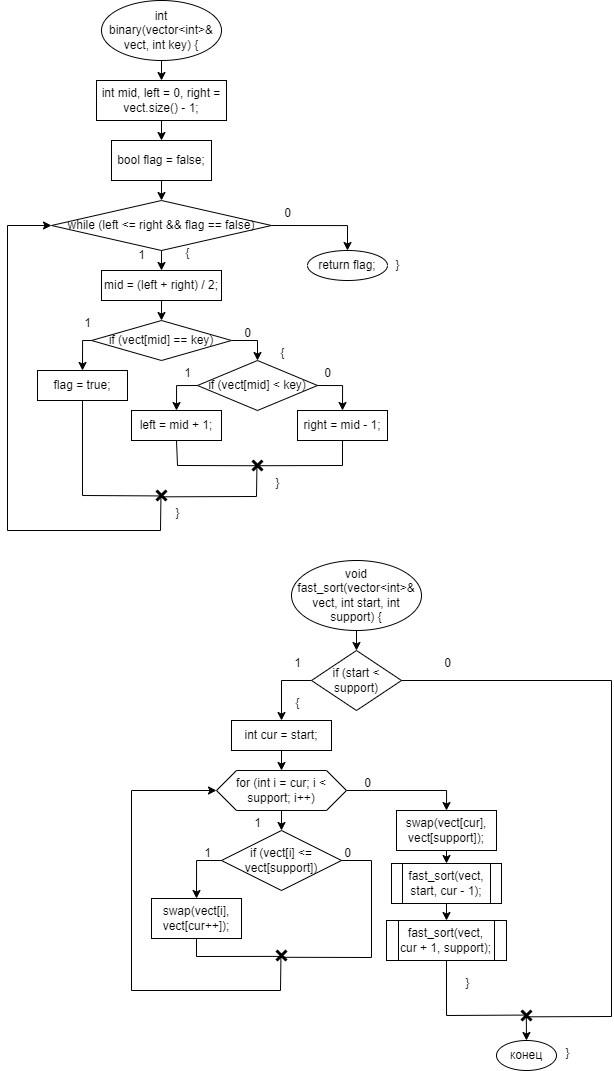
}

1. Результат работы программы:

**Бинарный поиск**

1. Анализ задачи:
2. Так как бинарный поиск работает только с отсортированными строками (массивами), то понадобится функция сортировки. Сортировка будет происходить методом быстрой сортировки.
3. Создать функцию для бинарного поиска. Функция будет искать середину строки, а после сравнивать ее с подстрокой. Пока строка не будет найдена, границы поиска будут меняться, а соответственно и середина строки тоже. При нахождении подстроки в строке, цикл будет прерываться, и функция возвратит значение “true”.
4. В функции main пользователь вводит требующиеся данные, после чего массив заполняется рандомными числами.
5. После вызова функции сортировки массива и функции поиска, с помощью проверки результата работы функции, в консоль выведется ответ.
6. Блок-схема:





1. Код программы:

#include <iostream>

#include <string>

#include <vector>

using namespace std;

void fast\_sort(vector<int>& vect, int start, int support) { //быстрая сортировка

if (start < support) {

int cur = start;

for (int i = cur; i < support; i++) {

if (vect[i] <= vect[support]) {

swap(vect[i], vect[cur++]); // замена элементов

}

}

swap(vect[cur], vect[support]);

fast\_sort(vect, start, cur - 1); // вызов функции с другими границами и ключом в левой стороне

fast\_sort(vect, cur + 1, support); // вызов функции с другими границами и ключом в правой стороне

}

}

int binary(vector<int>& vect, int key) { // бинарный поиск

int mid, left = 0, right = vect.size() - 1;

bool flag = false; // флаг для ответа

while (left <= right && flag == false) {

mid = (left + right) / 2; // рассчет среднего элемента

if (vect[mid] == key) { // если подстрока найдена

flag = true;

}

else {

if (vect[mid] < key) {

left = mid + 1; // изменение левой границы

}

else {

right = mid - 1; // изменение правой границы

}

}

}

return flag;

}

int main() {

system("chcp 1251");

int n, sub;

cout << "Введите количество элементов в строке: ";

cin >> n;

cout << "Введите подстроку: ";

cin >> sub;

vector <int> str;

srand(time(0));

for (int i = 0; i < n; i++) {

str.push\_back(rand() % 100 + 1); // заполнение массива рандомными числами

}

cout << "Текущий массив:" << endl;

for (int i = 0; i < str.size(); i++) { // вывод массива

cout << str[i] << ' ';

}

cout << endl;

fast\_sort(str, 0, str.size() - 1); // сортировка массива

cout << "Отсортированный массив:" << endl;

for (int i = 0; i < str.size(); i++) { // вывод отсортированного массива

cout << str[i] << ' ';

}

cout << endl;

if (binary(str, sub) != false)

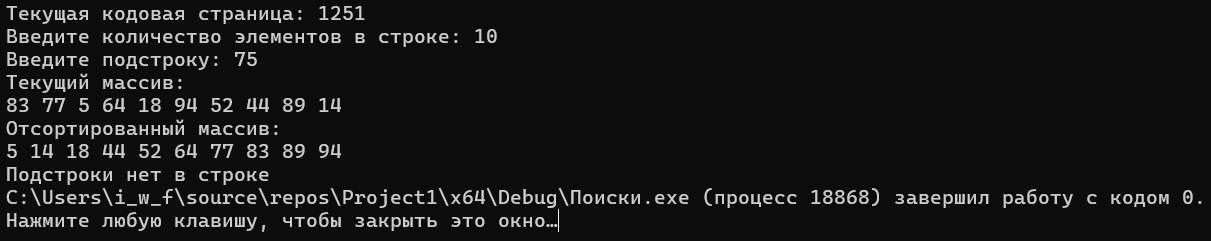
cout << "Подстрока есть в строке";

else

cout << "Подстроки нет в строке";

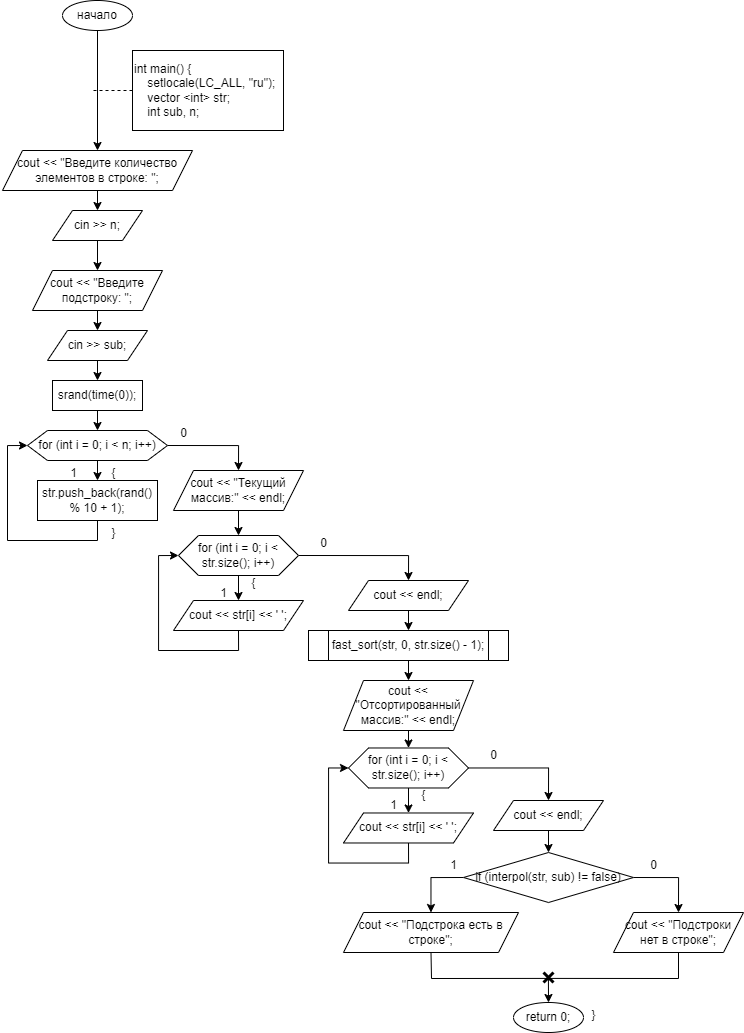
return 0;

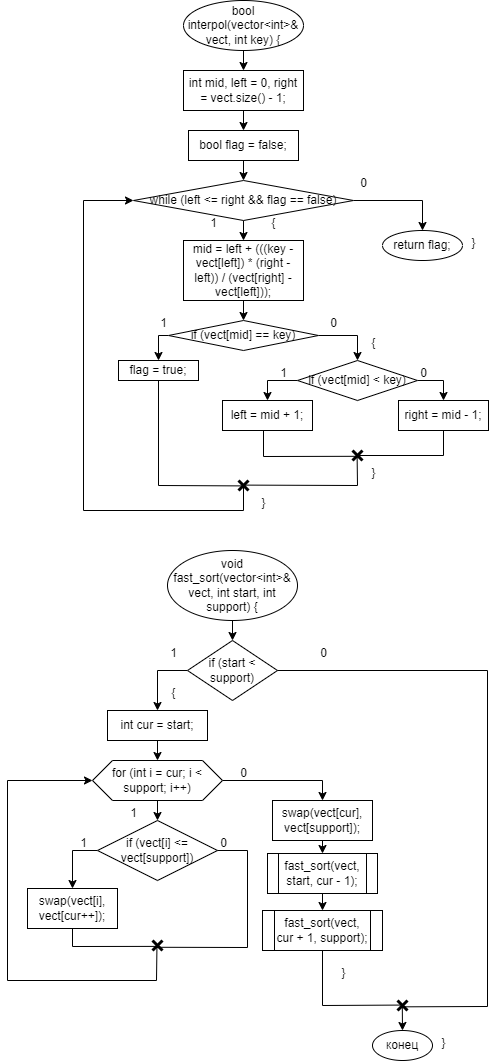
}

1. Результат работы программы:

**Интерполяционный поиск**

1. Анализ задачи:
2. Так как интерполяционный поиск работает только с отсортированными строками (массивами), то понадобится функция сортировки. Сортировка будет происходить методом быстрой сортировки.
3. Создать функцию для интерполяционного поиска. Функция будет искать ключевой элемент, который рассчитывается по формуле, а после сравнивать ее с подстрокой. Пока строка не будет найдена, границы поиска будут меняться, а соответственно и середина строки тоже. При нахождении подстроки в строке, цикл будет прерываться, и функция возвратит значение “true”.
4. В функции main пользователь вводит требующиеся данные, после чего массив заполняется рандомными числами.
5. После вызова функции сортировки массива и функции поиска, с помощью проверки результата работы функции, в консоль выведется ответ.
6. Блок-схема:





1. Код программы:

#include <iostream>

#include <vector>

using namespace std;

void fast\_sort(vector<int>& vect, int start, int support) { //быстрая сортировка

if (start < support) {

int cur = start;

for (int i = cur; i < support; i++) {

if (vect[i] <= vect[support]) {

swap(vect[i], vect[cur++]); // замена элементов

}

}

swap(vect[cur], vect[support]);

fast\_sort(vect, start, cur - 1); // вызов функции с другими границами и ключом в левой стороне

fast\_sort(vect, cur + 1, support); // вызов функции с другими границами и ключом в правой стороне

}

}

bool interpol(vector<int>& vect, int key) { // интерполяционный поиск

int mid, left = 0, right = vect.size() - 1;

bool flag = false; // флаг для выводв ответа

while (left <= right && flag == false) {

mid = left + (((key - vect[left]) \* (right - left)) / (vect[right] - vect[left])); // рассчет среднего элемента

if (vect[mid] == key) { // если подстрока найдена

flag = true; // изменение значения для выхода из цикла

}

else {

if (vect[mid] < key) {

left = mid + 1; // изменение левой границы

}

else {

right = mid - 1; // изменение правой границы

}

}

}

return flag;

}

int main() {

system("chcp 1251");

int n, sub;

cout << "Введите количество элементов в строке: ";

cin >> n;

cout << "Введите подстроку: ";

cin >> sub;

vector <int> str;

srand(time(0));

for (int i = 0; i < n; i++) {

str.push\_back(rand() % 100 + 1); // заполнение массива рандомными числами

}

cout << "Текущий массив:" << endl;

for (int i = 0; i < str.size(); i++) { // вывод массива

cout << str[i] << ' ';

}

cout << endl;

fast\_sort(str, 0, str.size() - 1); // сортировка массива

cout << "Отсортированный массив:" << endl;

for (int i = 0; i < str.size(); i++) { // вывод отсортированного массива

cout << str[i] << ' ';

}

cout << endl;

if (interpol(str, sub) != false)

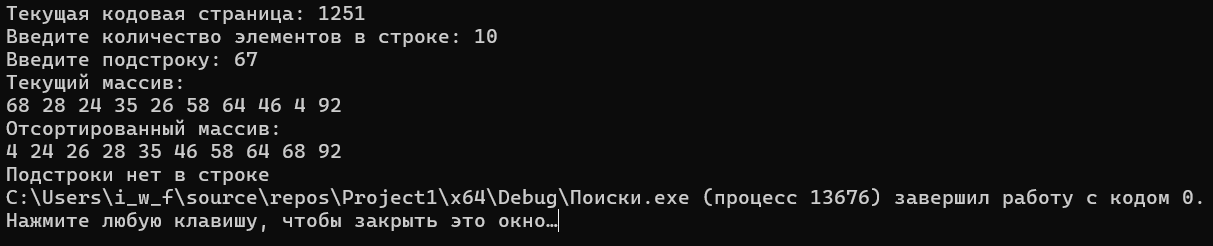
cout << "Подстрока есть в строке";

else

cout << "Подстроки нет в строке";

return 0;

}

1. Результат работы программы:

**Прямой поиск**

1. Анализ задачи:
2. Создать функцию для прямого поиска. В функции запустится цикл, в каждой итерации которого проверяется, совпадает ли текущий символ строки с первым символом подстроки. Если такие символы находятся, то начнется новый поиск, который будет проверять оставшиеся символы.
3. При нахождении подстроки в строке, цикл будет прерываться, и функция возвратит значение “true”.
4. В функции main пользователь вводит требующиеся данные и после вызова функции поиска, с помощью проверки результата работы функции, в консоль выведется ответ.
5. Блок-схема:
6. Код программы:

#include <iostream>

#include <string>

#include <string>

using namespace std;

int direct(string str, string sub) { //прямой поиск

int sub\_size = sub.size(); // размер подстроки

bool flag = false; // флаг для ответа

for (int i = 0; i < str.size() - sub\_size && flag == false; i++) {

bool search = true;

int j = 0;

while (j < sub\_size && search == true) {

search = (str[i + j] == sub[j]); // совпадение символов

j++; // увеличение длины совпадающих символов

}

if (search == true) { // если подстрока нашлась

flag = true; // смена флага для выхода из цикла и выводв ответа

}

}

return flag;

}

int main() {

system("chcp 1251");

string str, sub;

cout << "Введите строку: ";

cin >> str;

cout << "Введите подстроку: ";

cin >> sub;

if (direct(str, sub) != false)

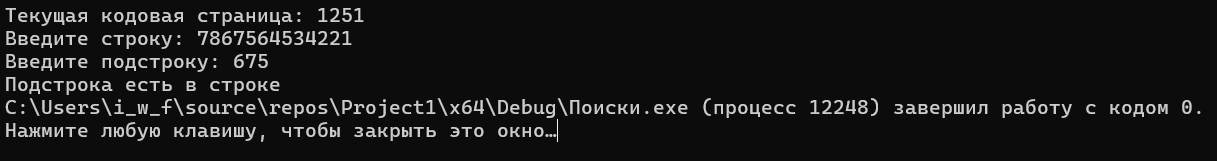
cout << "Подстрока есть в строке";

else

cout << "Подстроки нет в строке";

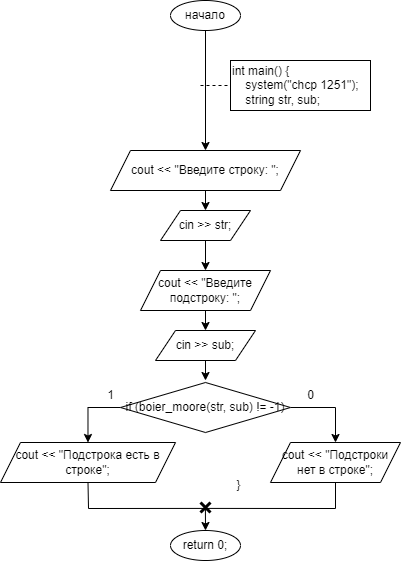
return 0;

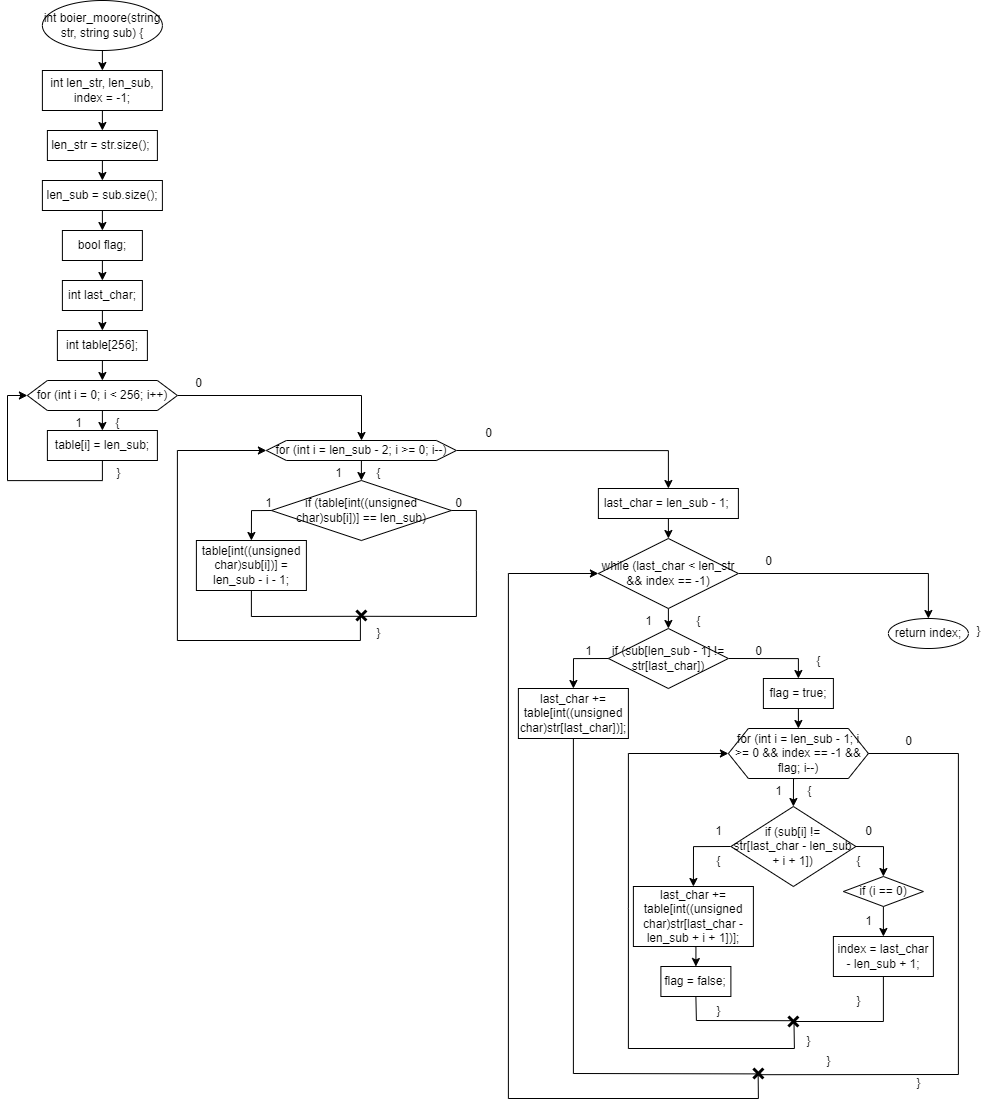
}

1. Результат работы программы:

**Поиск Бойера-Мура**

1. Анализ задачи:
2. Создать функцию для поиска. Функция будет содержать таблицу смещений для более быстрого поиска.
3. Затем для каждого элемента, который присутствует в подстроке, обновляется информация в таблице смещений.
4. Далее запускается цикл, который проверяет, совпадает ли последний символ подстроки с символом строки. Если это так, то запускается еще один цикл, который начинается с правого края. Если символы не совпадают, это фиксируется в таблице смещений.
5. При нахождении подстроки в строке, цикл будет прерываться, и функция возвратит значение “true”.
6. В функции main пользователь вводит требующиеся данные и после вызова функции поиска, с помощью проверки результата работы функции, в консоль выведется ответ.
7. Блок-схема:



****

1. Код программы:

#include <iostream>

#include <string>

using namespace std;

int boier\_moore(string str, string sub) { // поиск Бойера-Мура

int len\_str, len\_sub, index = -1;

len\_str = str.size();

len\_sub = sub.size();

bool flag;

int last\_char; // положение последнего символа подстроки в строке

int table[256];// таблица смещений

for (int i = 0; i < 256; i++) { // заполнение таблицы

table[i] = len\_sub;

}

for (int i = len\_sub - 2; i >= 0; i--) { // создание таблицы смещения элементов для подстроки

if (table[int((unsigned char)sub[i])] == len\_sub) { // если смещение элемента подстроки равно длине подстроки

table[int((unsigned char)sub[i])] = len\_sub - i - 1; // вычисление шага смещения

}

}

last\_char = len\_sub - 1;

while (last\_char < len\_str && index == -1) {

if (sub[len\_sub - 1] != str[last\_char]) {

last\_char += table[int((unsigned char)str[last\_char])]; // смещение подстроки

}

else {

flag = true;

for (int i = len\_sub - 1; i >= 0 && index == -1 && flag; i--) {

if (sub[i] != str[last\_char - len\_sub + i + 1]) { // смещение по таблице последнего символа, если он не равен элементу

last\_char += table[int((unsigned char)str[last\_char - len\_sub + i + 1])]; // смещение по строке

flag = false; // смена флага

}

else if (i == 0) { // если подстрока найдена

index = last\_char - len\_sub + 1; // вычисление индекса

}

}

}

}

return index; // возврат индекса

}

int main() {

system("chcp 1251");

string str, sub;

cout << "Введите строку: ";

cin >> str;

cout << "Введите подстроку: ";

cin >> sub;

if (boier\_moore(str, sub) != -1)

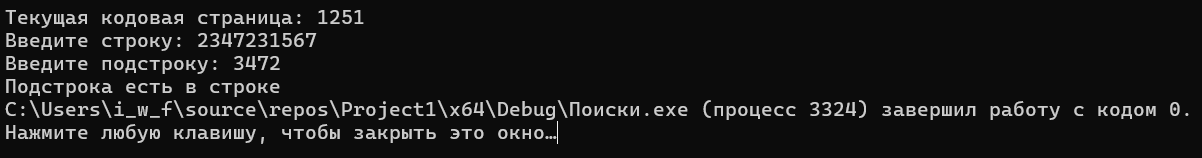
cout << "Подстрока есть в строке";

else

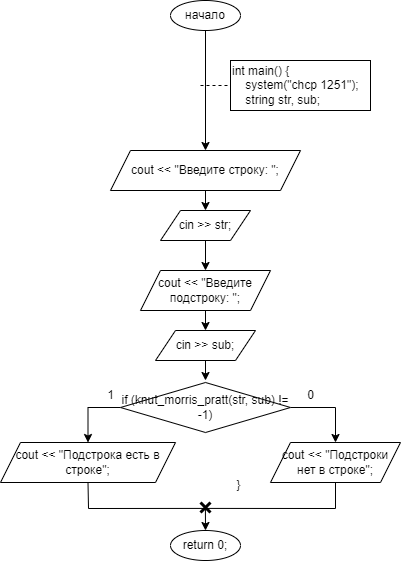
cout << "Подстроки нет в строке";

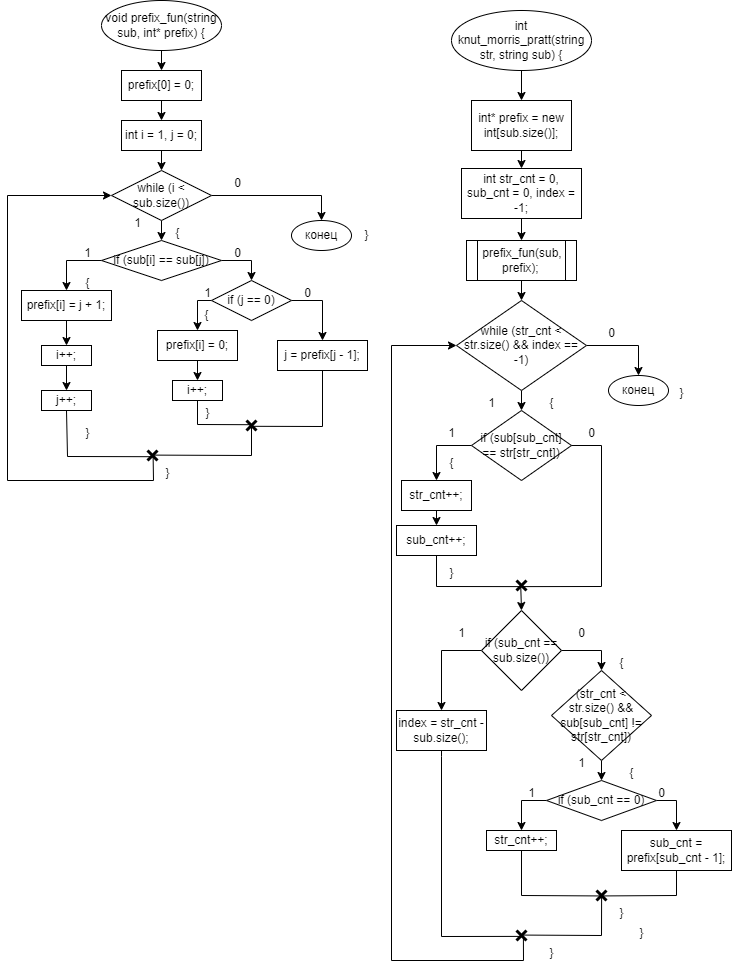
return 0;

}

1. Результат работы программы:

**Поиск Кнута-Морриса-Пратта**

1. Анализ задачи:
2. Перед тем как создать функцию для поиска, понадобится функция для префиксов. Функция создаст и будет хранить значения префиксов для подстроки.
3. В функции поиска начинается цикл, который проверяет, совпадает ли символ подстроки с предыдущим символом подстроки, и в таком случае значение префикс функции увеличивается на 1, а если не совпадает, тогда в префикс функции устанавливается 0.
4. Используя префикс функцию, происходит сравнение текущего символа строки и текущего символа подстроки. Если символы совпадают, то счетчики увеличиваются.
5. При нахождении подстроки в строке функция возвратит индекс первого символа подстроки в строке.
6. В функции main пользователь вводит требующиеся данные и после вызова функции поиска, с помощью проверки результата работы функции, в консоль выведется ответ.
7. Блок-схема:



1. Код программы:

#include <iostream>

#include <string>

using namespace std;

void prefix\_fun(string sub, int\* prefix) { // префикс функция

prefix[0] = 0; // изначально префикс нулевой

int i = 1, j = 0;

while (i < sub.size()) {

if (sub[i] == sub[j]) { // если символы совпадают

prefix[i] = j + 1; // изменение префикса

i++;

j++;

}

else {

if (j == 0) { // если начало подстроки

prefix[i] = 0;

i++;

}

else {

j = prefix[j - 1];

}

}

}

}

int knut\_morris\_pratt(string str, string sub) { // поиск Кнута-Морриса-Прата

int\* prefix = new int[sub.size()]; // массив для префикса

int str\_cnt = 0, sub\_cnt = 0, index = -1;

prefix\_fun(sub, prefix);

while (str\_cnt < str.size() && index == -1) {

if (sub[sub\_cnt] == str[str\_cnt]) { // если символы совпали

str\_cnt++; // увеличение счетчика

sub\_cnt++; // увеличение счетчика

}

if (sub\_cnt == sub.size()) { // если найдена вся подстрока

index = str\_cnt - sub.size();

}

else if (str\_cnt < str.size() && sub[sub\_cnt] != str[str\_cnt]) { // если найдена не вся подстрока

if (sub\_cnt == 0) { // если начало подстроки

str\_cnt++; // увеличение счетчика

}

else {

sub\_cnt = prefix[sub\_cnt - 1];

}

}

}

delete[] prefix; // освобождение памяти

return index;

}

int main() {

system("chcp 1251");

string str, sub;

cout << "Введите строку: ";

cin >> str;

cout << "Введите подстроку: ";

cin >> sub;

if (knut\_morris\_pratt(str, sub) != -1)

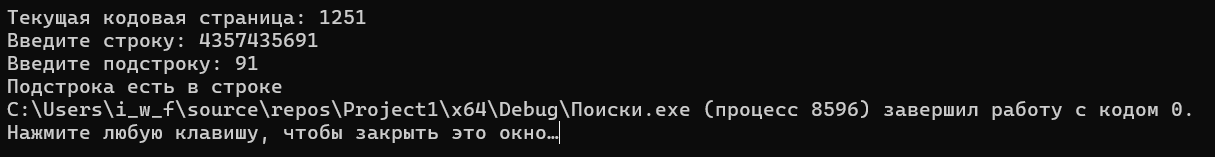
cout << "Подстрока есть в строке";

else

cout << "Подстроки нет в строке";

return 0;

}

1. Результат работы программы: