Министерство образования и науки Российской Федерации

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

*Кафедра МО ЭВМ*

Дисциплина « Алгоритмы и структуры данных»

**Отчет по лабораторной работе №3**

Факультет: КТИ

Группа: 1381

Выполнил: Козицкий Д.Н.

Проверил: Казаков Б.Б.

Санкт-Петербург

2013

**Поставленная задача**

Требуется по заданной строке определить минимальное количество символов, которые необходимо вставить в строку для преобразования ее в палиндром.

**Вход**: строка длиной n (3<=n<=5000), состоящая из прописных и строчных букв латинского алфавита и цифр от 0 до 9. Прописные и строчные буквы различаются.

**Выход**: целое число - минимальное количество вставляемых символов.

**Основные теоретические сведения**

Палиндром — это симметричная строка, т.е. она одинаково читается как слева направо, так и справа налево.

Например, вставкой двух символов строка "Ab3bd" может быть преобразована в палиндром ("dAb3bAd" или "Adb3bdA"), а вставкой менее двух символов палиндром в этом примере получить нельзя.

**Алгоритм решения**

Формируется матрица NxN, каждый элемент которой M[x][y] - минимальное количество букв, которые необходимо вставить в строку S(i,j) длины N с i-го символа по j-й символ строки S для того, чтобы получить из неё палиндром.

Искомый результат – M[0][N-1].

Главная диагональ матрицы изначально заполнена нулями, как и всё пространство под ней.

Заполняется оставшаяся часть по принципу:

* если S[i] = S[j], то M[i][j] = M[i+1][j-1];
* если же S[i] != S[j], то M[I][j] = Min( M[i+1][j], M[i][j-1] ) +1.

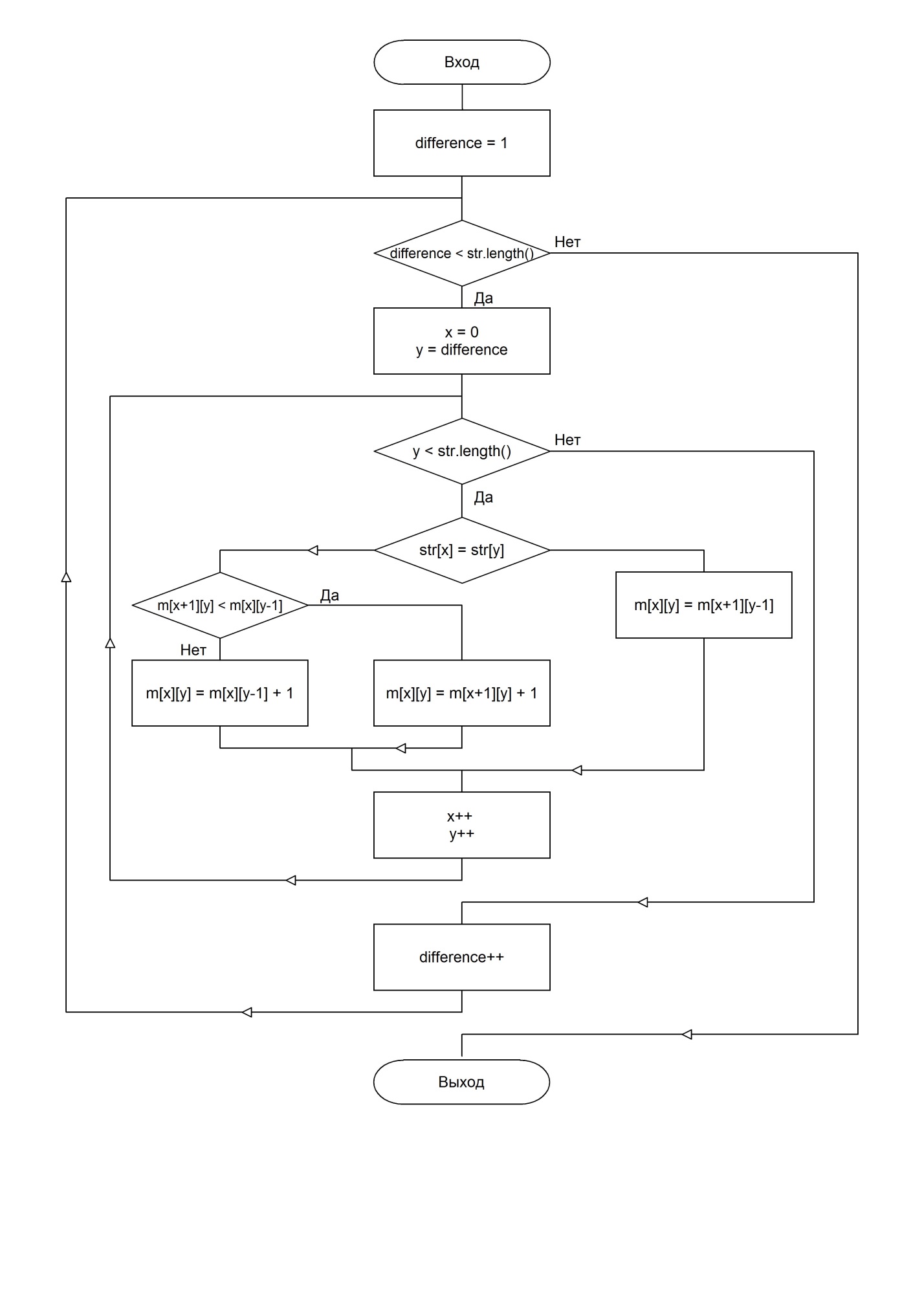
Матрица для S = 'Ab3bd':

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **1** | **2** |
| **-** | **0** | **1** | **0** | **1** |
| **-** | **-** | **0** | **1** | **3** |
| **-** | **-** | **-** | **0** | **1** |
| **-** | **-** | **-** | **-** | **0** |

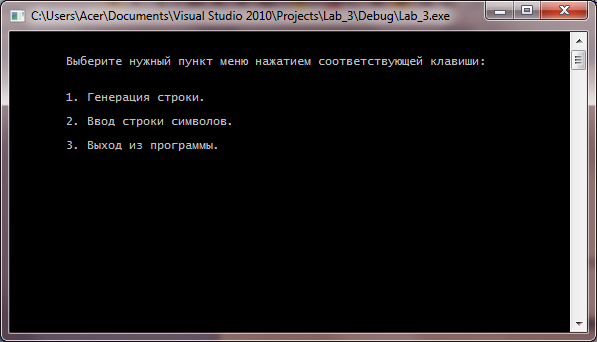
Матрица для S = 'abbbcda':

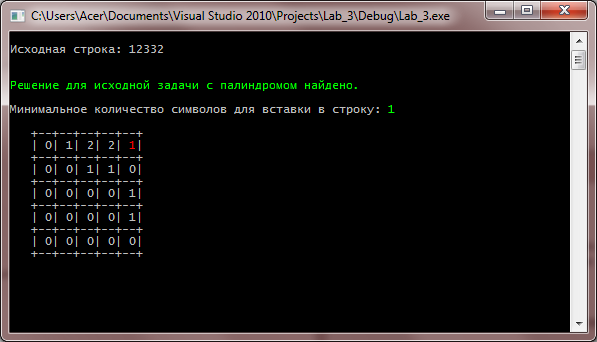
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **1** | **1** | **2** | **3** | **2** |
| **-** | **0** | **0** | **0** | **1** | **2** | **3** |
| **-** | **-** | **0** | **0** | **1** | **2** | **3** |
| **-** | **-** | **-** | **0** | **1** | **2** | **3** |
| **-** | **-** | **-** | **-** | **0** | **1** | **2** |
| **-** | **-** | **-** | **-** | **-** | **0** | **1** |
| **-** | **-** | **-** | **-** | **-** | **-** | **0** |

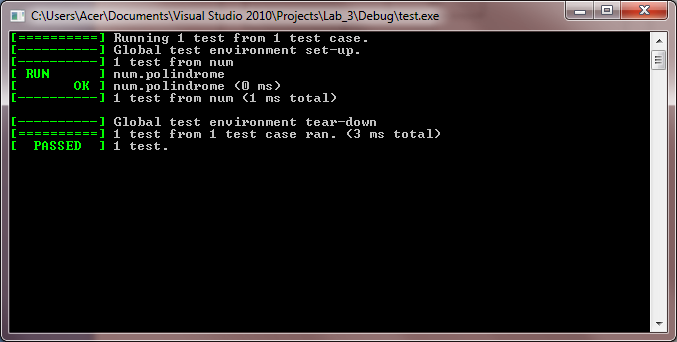
**Блок-схема алгоритма поиска количества вставок**

****

**Пример**  **работы программы**

****





**Исходный текст программы**

//Lab\_3.cpp

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <string>

#include <conio.h>

#include <stdio.h>

#include <ctime>

#include <math.h>

#include "polindrome.h"

using namespace std;

#define endl "\n\t"

int main ()

{

system("chcp 1251");

char sw;

bool menu\_view = TRUE;

bool string\_generation;

bool detailed\_mode;

Polindrome polindrome;

do{

system("cls");

cout << endl << endl << "Выберите нужный пункт меню нажатием соответствующей клавиши:";

cout << endl << endl << endl << "1. Генерация строки.";

cout << endl << endl << "2. Ввод строки символов.";

cout << endl << endl << "3. Выход из программы.";

fflush(stdin);

sw=\_getch();

switch (sw) {

case '1': string\_generation = TRUE; menu\_view = FALSE; break;

case '2': string\_generation = FALSE; menu\_view = FALSE; break;

case '3': return 1;

default : menu\_view = TRUE;

}

}while(menu\_view);

if( !string\_generation ) {

system("cls");

while( !polindrome.CheckOnCreated() ){

cout << endl << endl << "Введите исходную строку:";

cout << endl;

string \_str;

getline(cin, \_str);

cout << endl << \_str;

if ( !polindrome.AddString(\_str) ) {

cout << endl;

system("pause");

}

system("cls");

}

}

else{

char count\_symbols[20] = "";

while( atoi(count\_symbols) < 3 || atoi(count\_symbols) > 5000 ){

system("cls");

cout << endl << endl << "Введите количество символов в генерируемой строке:";

cout << endl;

cin >> count\_symbols;

if ( atoi(count\_symbols) < 3 || atoi(count\_symbols) > 5000 ) {

cout << endl << "Ошибка." << endl << "Количество символов "

<< "в строке не соответствует условию (3 - 5000)." << endl;

system("pause");

}

}

string \_str;

int temp;

srand(time(0));

for(int \_count\_symbols = atoi(count\_symbols); \_str.length() < \_count\_symbols;){

// А-Я: -64,-33; а-я: -32,-1; ё: -72 ; Ё: -88; A-Z: 65,90; a-z: 97,122; 0-9: 48,57

temp = fmod(rand(),127.0);

if(temp >= 0 && temp < 64)

\_str += temp - 64;

if(temp >= 64 && temp < 89)

\_str += temp + 1;

if(temp >= 89 && temp < 115)

\_str += temp + 8;

if(temp > 116)

\_str += temp -69;

if(temp == 115)

\_str += -72;

if(temp == 116)

\_str += -88;

}

polindrome.AddString(\_str);

cout << endl << "Сгенерирована строка:"

<< endl << \_str << endl << endl;

system("pause");

}

system("cls");

if(polindrome.Size() < 19){

do{

system("cls");

cout << endl << endl << "Выберите нужный пункт меню нажатием соответствующей клавиши:";

cout << endl << endl << endl << "1. Поиск решения"

<< " в автоматическом режиме" << endl

<< " с выводом конечного варианта на экран.";

cout << endl << endl << "2. Поиск решения"

<< " в ручном режиме"<<endl

<< " с выводом подробных действий на экран.";

fflush(stdin);

sw=\_getch();

switch (sw) {

case '1': detailed\_mode = FALSE; menu\_view = FALSE; break;

case '2': detailed\_mode = TRUE; menu\_view = FALSE; break;

default : menu\_view = TRUE;

}

}while(menu\_view);

}

else

detailed\_mode = FALSE;

polindrome.FindSolution(detailed\_mode);

cout << endl << "Программа успешно выполнена. ";

system("pause");

return 0;

}

//palindrome.h

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <string>

#include <conio.h>

#include <stdio.h>

#include <ctime>

#include <math.h>

enum Color

{

Black = 0,

Grey = FOREGROUND\_INTENSITY,

LightGrey = FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE,

White = FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY,

Blue = FOREGROUND\_BLUE,

Green = FOREGROUND\_GREEN,

Cyan = FOREGROUND\_GREEN | FOREGROUND\_BLUE,

Red = FOREGROUND\_RED,

Purple = FOREGROUND\_RED | FOREGROUND\_BLUE,

LightBlue = FOREGROUND\_BLUE | FOREGROUND\_INTENSITY,

LightGreen = FOREGROUND\_GREEN | FOREGROUND\_INTENSITY,

LightCyan = FOREGROUND\_GREEN | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY,

LightRed = FOREGROUND\_RED | FOREGROUND\_INTENSITY,

LightPurple = FOREGROUND\_RED | FOREGROUND\_BLUE | FOREGROUND\_INTENSITY,

Orange = FOREGROUND\_RED | FOREGROUND\_GREEN,

Yellow = FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_INTENSITY,

};

using namespace std;

class Polindrome{

public:

Polindrome(): Width(0), solution(-1), m(NULL) {}

public:

bool AddString (string \_str) {

if (\_str.length() < 3 || \_str.length() > 5000){

cout << endl << "Ошибка." << endl << "Количество символов "

<< "в строке не соответствует условию (3 - 5000).";

return FALSE;

}

str = \_str;

CreateNewMassive(\_str.length());

return TRUE;

}

private:

void CreateNewMassive(int n){

Width = n;

m = new int\*[n];

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

m[i] = new int[n];

for(int j = 0; j < n; j++)

m[i][j] = 0;

}

}

private:

int Width;

int \*\*m;

int solution;

string str;

public:

bool CheckOnCreated(){

if (m) return TRUE;

return FALSE;

}

public:

void Print( int x = 0, int y = 0, int flag = 0) {

if(Width < 19){

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

cout << endl << " +";

for(int k = 0; k < Width; k++)

cout << "--+";

cout << endl << " |";

for(int j = 0; j < Width; j++){

if((x || y) && i == x + 1 && j == y - 1 && flag == 1)

SetColor(LightGreen);

if((x || y) && ((i == x + 1 && j == y)

|| (i == x && j == y - 1)) && flag == 2)

SetColor(LightGreen);

if((x || y) && i == x && j == y) SetColor(LightRed);

printf("%2d", m[i][j]);

if((x || y) && i == x && j == y || flag) SetColor(LightGrey);

cout << "|";

}

}

cout << endl << " +";

for(int k = 0; k < Width; k++)

cout << "--+";

cout << endl;

}

else{

cout << endl << endl << "Слишком большая матрица для вывода." << endl;

}

}

private:

int PrintString(int x = 0, int y = 0){

cout << endl << "Исходная строка: ";

if(!x && !y){

cout << str;

return 1;

}

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

if(i == x || i == y) SetColor(LightRed);

cout << str[i];

if(i == x || i == y) SetColor(LightGrey);

}

return 1;

}

private:

void SetColor(Color color){

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE),color);

}

public:

int Size(){

return Width;

}

public:

void FindSolution (bool detailed\_mode) {

int x, y, difference;

for(difference = 1; difference < str.length() ; difference++){

for(x = 0, y = difference; y < str.length(); x++, y++){

if(str[x] == str[y]){

m[x][y] = m[x+1][y-1];

if(detailed\_mode){

system("cls");

PrintString(x,y);

cout << endl << endl << "Выделенные символы в строке ";

SetColor(LightGreen); cout << "равны"; SetColor(LightGrey);

cout << "." << endl

<< endl << "Значение выделенного красным элемента матрицы равно зеленому."

<< endl << endl;

Print(x,y,1);

cout << endl << endl;

system("pause");

}

}

else{

if(m[x+1][y] < m[x][y-1])

m[x][y] = m[x+1][y] + 1;

else

m[x][y] = m[x][y-1] + 1;

if(detailed\_mode){

system("cls");

PrintString(x,y);

cout << endl << endl << "Выделенные символы в строке ";

SetColor(LightRed); cout << "не равны"; SetColor(LightGrey);

cout << "." << endl

<< endl << "Значение выделенного красным элемента матрицы равно"

<< " минимальному " << endl << "из зеленых и увеличено на единицу."

<< endl;

Print(x,y,2);

cout << endl << endl;

system("pause");

}

}

}

}

system("cls");

PrintString();

SetColor(LightGreen);

cout << endl << endl << endl << "Решение для исходной задачи с палиндромом найдено. ";

SetColor(LightGrey);

cout << endl << endl << "Минимальное количество символов для вставки в строку: ";

SetColor(LightGreen);

cout << m[0][Width-1] << endl;

SetColor(LightGrey);

Print(0,Width-1);

}

int FindSolution\_2 (string str,int n) {

int x, y, difference;

int \*\*m2;

Width = n;

m2 = new int\*[n];

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

m2[i] = new int[n];

for(int j = 0; j < n; j++)

m2[i][j] = 0;

}

for(difference = 1; difference < n ; difference++){

for(x = 0, y = difference; y < n; x++, y++){

if(str[x] == str[y]){

m2[x][y] = m2[x+1][y-1];

}

else

{

if(m2[x+1][y] < m2[x][y-1])

m2[x][y] = m2[x+1][y] + 1;

else

m2[x][y] = m2[x][y-1] + 1;

}

}

}

int h=m2[0][Width-1];

return h;

}

};

//test.cpp

#include "stdafx.h"

#include "polindrome.h"

#include <conio.h>

#include "gtest\gtest.h"

#include <iostream>

#include <string.h>

class num : public ::testing::Test {

};

void Mas(int result, int n)

{

ASSERT\_EQ(result, n);

}

TEST\_F(num,polindrome)

{

//test 1

int result;

string str1="1221";

Polindrome polindrome;

result=polindrome.FindSolution\_2(str1,4);

Mas(result,0);

//test 1 end

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

::testing::InitGoogleTest(&argc, argv);

RUN\_ALL\_TESTS();

\_getch();

return 1;

}

**Вывод**

Была реализована программа по методу динамического программирования, позволяющая находить минимальное количество символов, которые нужно вставить в исходную (введенную или сгенерированную) строку для преобразования её в палиндром.