**Лабораторная работа 4**

**ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

**Отчёт выполнил : Алексей Шимко**

**Группа: 6**

**Подгруппа : 2**

## **Задание 1**

Код для создание случайной строки:

char\* GenerateRandomString(int size)

{

char\* str = (char\*)malloc(sizeof(char) \* size);

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

str[i] = rand() % 26 + 'a'; // 26 букв в алфавите

}

str[size] = '\0';

return str;

}

## **Задание 2**

Вычисление расстояние Левенштейна двумя способами:

clock\_t t1 = 0, t2 = 0, t3 = 0, t4 = 0;

int lx = sizeof(s1);

int ly = sizeof(s2);

int s1\_size[]{ FIRST\_LEN / 25, FIRST\_LEN / 20, FIRST\_LEN / 15, FIRST\_LEN / 10, FIRST\_LEN / 5, FIRST\_LEN / 2, FIRST\_LEN };

int s2\_size[]{ SECOND\_LEN / 25, SECOND\_LEN / 20, SECOND\_LEN / 15, SECOND\_LEN / 10, SECOND\_LEN / 5, SECOND\_LEN / 2, SECOND\_LEN };

cout << "\n\n-- расстояние Левенштейна -----";

cout << "\n\n--длина --- рекурсия -- дин.програм. ---\n";

for (int i = 0; i < min(lx, ly); i++)

{

t1 = clock();

levenshtein\_r(s1\_size[i], s1, s2\_size[i], s2);

t2 = clock();

t3 = clock();

levenshtein(s1\_size[i], s1, s2\_size[i], s2);

t4 = clock();

cout << right << setw(2) << s1\_size[i] << "/" << setw(2) << s2\_size[i]

<< " " << left << setw(10) << (t2 - t1)

<< " " << setw(10) << (t4 - t3) << endl;

}

## **Задание 3**

Результат работы:

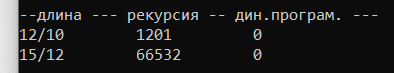
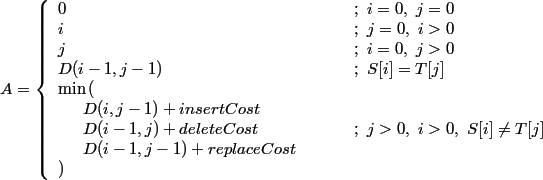


График:

# **Задание 4**

Слова: Раб , Барка

**Решение:**



1. L(«Раб», «Барка») = min
2. L(«Ра», «Барка») = min
3. L(«Раб», «Барк») = min
4. L(«Ра» , «Барк») = min
5. L(«Р» , «Барка») = min

L(«» , «Барка») = 5

L(«» , «Барк») = 4

1. L(«Р» , «Барк») = min

L(«» , «Бар») = 3

1. L(«Раб» , «Бар») = min
2. L(«Ра» , «Бар») = min
3. L(«Р» , «Бар») = min

L(«» , «Ба») = 2

1. L(«Раб» , «Ба») = min
2. L(«Ра» , «Ба») = min
3. L(«Р» , «Ба») = min

L(«», «Б») = 1

1. L(«Раб» , «Б») = min

L(«Раб» , «») = 3;

1. L(«Ра» , «Б») = min

L(«Ра» , «») = 2;

L(«Р» , «») = 1;

1. L(«Р» , «Б») = min

16. L(«Ра», «Б») = min (2, 3, 2) = 2

17. L(«Раб», «Б») = min (3, 4, 3) = 3

18. L(«Р», «Ба») = min (3, 2, 3) = 2

19. L(«Ра», «Ба») = min (3, 3, 2) = 2

20. L(«Раб», «Ба») = min (2, 4 , 3) = 2

21. L(«Р», «Бар») = min (4, 3, 3) = 3

22. L(«Ра», «Бар») = min (4, 2, 3) = 2

23. L(«Раб», «Бар») = min (3, 3, 2) = 2

24. L(«Р», «Барк») = min (5, 4, 4) = 4

25. L(«Р», «Барка») = min (6, 5, 5) = 5

26. L(«Ра», «Барк») = min (6 3, 4) = 3

27. L(«Раб», «Барк») = min (4, 2, 3) = 2

28. L(«Ра», «Барка») = min (6, 4, 4) = 4

29. L(«Вол», «Колун») = min (4, 3, 4) = 3

5.

int lcs(int lenx, const char x[],

int leny, const char y[])

{

int rc = 0;

if (lenx > 0 && leny > 0)

{

if (x[lenx - 1] == y[leny - 1]) rc = 1 + lcs(lenx - 1, x, leny - 1, y);

else rc = std::max(lcs(lenx, x, leny - 1, y), lcs(lenx - 1, x, leny, y));

}

return rc; //длина LCS

}

enum Dart { TOP, LEFT, LEFTTOP };

void getLCScontent(int lenx, int leny, const char x[],

const Dart\* B,

int n, int i, int j, char z[])

{

if ((i > 0 && j > 0 && n > 0))

{

if (LCS\_B(i, j) == LEFTTOP)

{

getLCScontent(lenx, leny, x, B, n - 1, i - 1, j - 1, z);

LCS\_Z(n) = LCS\_X(i);

LCS\_Z(n + 1) = 0;

}

else if (LCS\_B(i, j) == TOP)

getLCScontent(lenx, leny, x, B, n, i - 1, j, z);

else getLCScontent(lenx, leny, x, B, n, i, j - 1, z);

}

};

int lcsd(const char x[], const char y[], char z[])

{

int n;

int lenx = strlen(x), leny = strlen(x),

\* C = new int[(lenx + 1) \* (leny + 1)];

Dart\* B = new Dart[(lenx + 1) \* (leny + 1)];

memset(C, 0, sizeof(int) \* (lenx + 1) \* (leny + 1));

for (int i = 1; i <= lenx; i++)

for (int j = 1; j <= leny; j++)

if (LCS\_X(i) == LCS\_Y(j)) {

LCS\_C(i, j) = LCS\_C(i - 1, j - 1) + 1;

LCS\_B(i, j) = LEFTTOP;

}

else if (LCS\_C(i - 1, j) >= LCS\_C(i, j - 1)){

LCS\_C(i, j) = LCS\_C(i - 1, j);

LCS\_B(i, j) = TOP;

}

else

{

LCS\_C(i, j) = LCS\_C(i, j - 1);

LCS\_B(i, j) = LEFT;

}

getLCScontent(lenx, leny, x, B, LCS\_C(lenx, leny), lenx, leny, z);

return LCS\_C(lenx, leny);

}

Результат выполнения программы:

