

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 3**

|  |  |
| --- | --- |
| **Тема:** Построение и программная реализация алгоритма сплайн-интерполяции табличных функций.  **Студент:** Елгин И. Ю.  **Группа:** ИУ7-44Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель:** Градов В. М. |  |

Москва.

2021 г.

**Цель работы**: Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

**1 Исходные данные.**

1. Таблица функции с количеством узлов 11. Заданная с помощью формулы в диапазоне [0..10] с шагом 1.

|  |  |  |
| --- | --- | --- |
| № | X | Y |
| 1 | 0 | 0 |
| 2 | 1 | 1 |
| 3 | 2 | 4 |
| 4 | 3 | 9 |
| 5 | 4 | 16 |
| 6 | 5 | 25 |
| 7 | 6 | 36 |
| 8 | 7 | 49 |
| 9 | 8 | 64 |
| 10 | 9 | 81 |
| 11 | 10 | 100 |

1. Значение x = 0.5 (у края), x=5.5 (по середине)

Программа считывает таблицу из файла “*input.txt*”, находящегося в директории программы. В первой строке находятся количество строк в остальных данные таблицы.

Значение x вводятся с клавиатуры

**2 Код на c99.**

Main.c

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <math.h>
5. typedef struct table
6. {
7. int size;
8. double \*x;
9. double \*y;
10. }
11. Table;
13. typedef struct Koef\_table
14. {
15. int size;
16. double \*a;
17. double \*b;
18. double \*c;
19. double \*d;
20. double \*eps;
21. double \*nu;
22. double \*f;
23. double \*h;
24. }
25. Koef\_table;
27. int closest\_point\_num(Table \*TAB, double x)
28. {
29. int result;
30. for(int i = 0; i < TAB->size; i++)
31. if (x - TAB->x[i] < 0)
32. {
33. result = i;
34. break;
35. }
36. return result;
37. }
39. void Read(Table \*TAB, char \*f\_name)
40. {
41. FILE \*f = fopen(f\_name, "r");
42. fscanf(f, "%d", &(TAB->size));
43. TAB->x = malloc(TAB->size \* sizeof(double));
44. TAB->y = malloc(TAB->size \* sizeof(double));
45. for (int i = 0; i < TAB->size; i++)
46. {
47. fscanf(f, "%lf", TAB->x + i);
48. fscanf(f, "%lf", TAB->y + i);
49. }
50. fclose(f);
51. }
53. void Init\_koef\_table(Koef\_table \*KOEF, int size)
54. {
55. KOEF->size = size;
56. KOEF->a = calloc(size, sizeof(double));
57. KOEF->b = calloc(size, sizeof(double));
58. KOEF->c = calloc(size + 1, sizeof(double));
59. KOEF->d = calloc(size, sizeof(double));
60. KOEF->eps = calloc(size + 1, sizeof(double));
61. KOEF->nu = calloc(size + 1, sizeof(double));
62. KOEF->f = calloc(size, sizeof(double));
63. KOEF->h = calloc(size, sizeof(double));
64. }
66. void calculate\_koef(Koef\_table \*KOEF, Table \*TAB)
67. {
68. Init\_koef\_table(KOEF, TAB->size - 1);
69. for (int i = 0; i < KOEF->size; i++)
70. KOEF->h[i] = TAB->x[i+1] - TAB->x[i];
71. for (int i = 1; i < KOEF->size; i++)
72. KOEF->f[i] = 3 \* ((TAB->y[i+1] - TAB->y[i])/KOEF->h[i] - (TAB->y[i] - TAB->y[i-1])/KOEF->h[i-1]);
73. for (int i = 2; i < KOEF->size + 1; i++)
74. {
75. KOEF->eps[i] = -KOEF->h[i-1]/(KOEF->h[i-2] \* KOEF->eps[i-1] + 2 \* (KOEF->h[i-2] + KOEF->h[i-1]));
76. KOEF->nu[i] = (KOEF->f[i-1] - KOEF->h[i-2] \* KOEF->nu[i-1])/(KOEF->h[i-2] \* KOEF->eps[i-1] + 2 \* (KOEF->h[i-2] + KOEF->h[i-1]));
77. }
78. for (int i = KOEF->size; i >= 1; i--)
79. KOEF->c[i] = KOEF->eps[i + 1] \* KOEF->c[i + 1] + KOEF->nu[i + 1];
80. for (int i = 0; i < KOEF->size; i++)
81. {
82. KOEF->a[i] = TAB->y[i];
83. KOEF->b[i] = (TAB->y[i + 1] - TAB->y[i])/KOEF->h[i] - KOEF->h[i]\*(KOEF->c[i+1] + 2 \* KOEF->c[i])/3;
84. KOEF->d[i] = (KOEF->c[i + 1] - KOEF->c[i])/(3 \* KOEF->h[i]);
85. printf("\na=%lf,b=%lf,c=%lf,d=%lf,eps=%lf,nu=%lf,f=%lf,h=%lf\n",KOEF->a[i],KOEF->b[i],KOEF->c[i],KOEF->d[i],KOEF->eps[i],KOEF->nu[i],KOEF->f[i],KOEF->h[i]);
86. }
87. }
89. double calk\_result(Koef\_table \*KOEF, Table \*TAB, double x)
90. {
91. int i = closest\_point\_num(TAB, x);
92. return KOEF->a[i - 1] + KOEF->b[i - 1] \* (x - TAB->x[i - 1]) + KOEF->c[i - 1] \* pow(x - TAB->x[i - 1], 2) + KOEF->d[i - 1] \* pow(x - TAB->x[i - 1], 3);
93. }
95. int main()
96. {
97. setbuf(stdout, NULL);
98. Table TAB;
99. Koef\_table KOEF;
100. double x;
101. printf("Введите x: ");
102. scanf("%lf", &x);
103. Read(&TAB, "input.txt");
104. calculate\_koef(&KOEF, &TAB);
105. printf("\n\nРезультат при x=%lf y=%lf", x, calk\_result(&KOEF, &TAB, x));
106. return 0;
107. }

При работе программы строиться таблица коэффициентов для сплайнов.

Сначала вычисляются h, f, eps и nu (eps и nu вычисляются с 3 значения)

Затем обратным ходом вычисляется c

Затем вычисляются оставшиеся коэффициенты a, b, d

После этого находится интервал интерполирования для x и вычисляется y

**3 Результаты работы.**

1. Значения вычисленные с помощью сплайнов y(0.5)= 0.341506 , y(5.5)=30.250345
2. Полином Ньютона 3 степени y(0.5)=0.25000 , y(5.5)= 30.25000
3. Истинные значения y(0.5)=0.25 , y(5.5)=30.25

Интерполяция сплайном значительно менее точна на концах таблицы.

Также интерполяция сплайном менее точна чем интерполяция полиномом Ньютона.

**4 Вопросы при защите лабораторной работы.**

1. Получить выражения для коэффициентов кубического сплайна, построенного на двух точках.

0

1. Выписать все условия для определения коэффициентов сплайна, построенного на 3-х точках
2. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо C1=C2.

4. Написать формулу для определения последнего коэффициента сплайна , чтобы

можно было выполнить обратный ход метода прогонки, если в качестве граничного

условия задано , где k,m и p - заданные числа.