

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

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

**Лабораторная работа № 2**

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

Москва.

2021 г.

**Цель работы**: Получение навыков построения алгоритма интерполяции таблично заданных функций двух переменных.

**1 Исходные данные.**

1. Таблица функции и её производных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| y/x | 0 | 1 | 2 | 3 | 4 |
| 0 | 0 | 1 | 4 | 9 | 16 |
| 1 | 1 | 2 | 5 | 10 | 17 |
| 2 | 4 | 5 | 8 | 13 | 20 |
| 3 | 9 | 10 | 13 | 18 | 25 |
| 4 | 16 | 17 | 20 | 25 | 32 |

2. Степень аппроксимирующего полинома – nx и ny.

3. Значение аргументов x и y, для которого выполняется интерполяция.

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

Значения nx, ny, x, y вводятся с клавиатуры

**2 Код на c99.**

struct.h

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | #include <stdio.h>  #include <stdlib.h>  #define EPS 0.00001  **typedef** **struct** data\_table  {  **double** \*\*table;// таблица значений  **int** size\_x, size\_y;// размер таблицы  }data\_table;  // чтение таблицы из файла в структуру  **void** **input\_from\_file**(data\_table \*d\_table, **char** \*fname);  // вычисление полинома Ньютона  **double** **newton\_polinom**(data\_table \*d\_table, **int** p, **int** n, **double** x, **int** ind);  // поиск индекса начального значения для апроксимации  **int** **select\_first\_number\_x**(data\_table \*d\_table, **double** x, **int** nx);  // поиск индекса начального значения для апроксимации  **int** **select\_first\_number\_y**(data\_table \*d\_table, **double** y, **int** ny);  // отчистка данных таблицы  **void** **free\_table**(**double** \*\*table, **int** n);  //формирует таблицу для апроксимации по x  data\_table \***form\_table**(data\_table \*d\_table, **int** ny, **int** nx, **int** px, **int** py, **double** x, **double** y); |

functions.c

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129 | #include "struct.h"  **double** **my\_abs**(**double** d)  {  **return** (d < **0**) ? -d : d;  }  **void** **free\_table**(**double** \*\*table, **int** n)  {  **for** (**int** i = **0**; i < n; i++)  free(table[i]);  free(table);  }  //создаёт и высчитывает таблицу разделённых разностей  **double** \*\***creat\_div\_diff\_table**(data\_table \*d\_table, **int** p, **int** n, **int** ind)  {  **double** \*\*div\_diff\_table = malloc(n \* **sizeof**(**double** \*));  **for** (**int** i = **0**; i < n; i++)  {  div\_diff\_table[i] = malloc((n - i) \* **sizeof**(**double**));  // заполняет начало таблицы разделённых разностей корнями  div\_diff\_table[i][**0**] = d\_table->table[p + i][ind];  }  **for** (**int** i = **1**; i < n; i++)  **for** (**int** j = **0** ; j < n - i; j++)  {  div\_diff\_table[j][i] = (div\_diff\_table[j][i - **1**] - div\_diff\_table[j + **1**][i - **1**]) /\  (d\_table->table[p + j][**0**] - d\_table->table[p + i + j][**0**]);  }  **return** div\_diff\_table;  }  // ищет значение приближённое к x  **int** **find\_medium\_x**(**double** \*\*arr, **int** size, **double** x)  {  **int** result = **1**;  **double** delta = my\_abs(arr[**0**][**1**] - x);  **for** (**int** i = **2**; i < size; i++)  {  **if** (my\_abs(arr[**0**][i] - x) < delta)  {  delta = my\_abs(arr[**0**][i] - x);  result = i;  }  }  **return** result;  }  // ищет значение приближённое к y  **int** **find\_medium\_y**(**double** \*\*arr, **int** size, **double** y)  {  **int** result = **1**;  **double** delta = my\_abs(arr[**1**][**0**] - y);  **for** (**int** i = **2**; i < size; i++)  {  **if** (my\_abs(arr[i][**0**] - y) < delta)  {  delta = my\_abs(arr[i][**0**] - y);  result = i;  }  }  **return** result;  }  **int** **select\_first\_number\_x**(data\_table \*d\_table, **double** x, **int** nx)  {  **int** m = find\_medium\_x(d\_table->table, d\_table->size\_x, x);  **if** (m <= nx / **2**)  **return** **1**;  **else** **if** (d\_table->size\_x - m - **1** < (nx / **2** + nx % **2**))  **return** d\_table->size\_x - nx;  **else**  **return** m + **1** - nx / **2**;  }  **int** **select\_first\_number\_y**(data\_table \*d\_table, **double** y, **int** ny)  {  **int** m = find\_medium\_y(d\_table->table, d\_table->size\_y, y);  **if** (m <= ny / **2**)  **return** **1**;  **else** **if** (d\_table->size\_y - m - **1** < (ny / **2** + ny % **2**))  **return** d\_table->size\_y - ny;  **else**  **return** m + **1** - ny / **2**;  }  **double** **newton\_polinom**(data\_table \*d\_table, **int** p, **int** n, **double** x, **int** ind)  {  **double** k = **1**, result = d\_table->table[p][ind];  **double** \*\*div\_diff\_table = creat\_div\_diff\_table(d\_table, p, n + **1**, ind);  **for** (**int** i = **0**; i < n; i++)  {  k \*= (x - d\_table->table[p + i][**0**]);  result += k \* div\_diff\_table[**0**][i + **1**];  }  free\_table(div\_diff\_table, n + **1**);  **return** result;  }  **void** **input\_from\_file**(data\_table \*d\_table, **char** \*fname)  {  **FILE** \*f = fopen(fname, "r");  fscanf(f, "%d %d", &(d\_table->size\_y), &(d\_table->size\_x));  d\_table->table = malloc(d\_table->size\_y \* **sizeof**(**double** \*));  **for** (**int** i = **0**; i < d\_table->size\_y; i++)  {  d\_table->table[i] = malloc(d\_table->size\_x \* **sizeof**(**double**));  **for** (**int** j = **0**; j < d\_table->size\_x; j++)  fscanf(f, "%lf", d\_table->table[i] + j);  }  }  //формирует таблицу для апроксимации по x  data\_table \***form\_table**(data\_table \*d\_table, **int** ny, **int** nx, **int** px, **int** py, **double** x, **double** y)  {  data\_table \*result = malloc(**sizeof**(data\_table));  result->table = malloc(nx \* **sizeof**(**double** \*));  result->size\_y = nx;  result->size\_x = **2**;  **for** (**int** i = **0**; i < nx; i++)  {  result->table[i] = malloc(**2** \* **sizeof**(**double**));  result->table[i][**0**] = d\_table->table[**0**][px + i];  result->table[i][**1**] = newton\_polinom(d\_table, py, ny - **1**, y, px + i);  }  **return** result;  } |

main.c

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | #include "struct.h"  **int** **main**()  {  setbuf(stdout, NULL);  data\_table d\_table;  **int** ny, nx;  **double** x, y;  // чтение таблицы из файла в структуру  input\_from\_file(&d\_table, "input.txt");  // ввод степени интерполяции и значения аргументов x и y  printf("Введите степени полинома nx, ny и значения x, y: ");  scanf("%d%d%lf%lf", &nx, &ny, &x, &y);  // поиск индекса начальных значений для апроксимации  **int** px = select\_first\_number\_x(&d\_table, x, (nx + **1**));  **int** py = select\_first\_number\_y(&d\_table, y, (ny + **1**));  //вычисляем значения x  data\_table \*x\_table = form\_table(&d\_table, ny + **1**, nx + **1**, px, py, x, y);    //for (int i = 0; i < nx + 1; i++)  // printf("%lf %lf\n", x\_table->table[i][0], x\_table->table[i][1]);  // вычисление полинома Ньютона вывод результата  printf("**\n**Апроксимированное значение в точке: %lf**\n**",\  newton\_polinom(x\_table, **0**, nx, x, **1**));  free\_table(d\_table.table, d\_table.size\_y);  free\_table(x\_table->table, x\_table->size\_y);  free(x\_table);  **return** **0**;  } |

**3 Результаты работы.**

Программа вычисляет участки интерполяции x и у создаёт таблицу из nx + 1 аппроксимированных значений x для заданного y и по ней вычисляет аппроксимированное значение для заданного x используя полином Ньютона.

Формула для полинома Ньютона



1. Таблица аппроксимированных значений при nx, ny = (1, 2, 3) и x = y = 1.5 .

|  |  |  |  |
| --- | --- | --- | --- |
| Nx/ny | 1 | 2 | 3 |
| 1 | 5 | 4.75 | 4.75 |
| 2 | 4.75 | 4.5 | 4.5 |
| 3 | 4.75 | 4.5 | 4.5 |

**4 Вопросы при защите лабораторной работы.**

1. Пусть производящая функция таблицы суть z(x,y)=x 2 +y 2 . Область определения по x и y 0-5 и 0-5. Шаги по переменным равны 1. Степени nx = ny =1, x=y=1.5. Приведите по шагам те. значения функции, которые получаются в ходе последовательных интерполяций. по строкам и столбцу.

*Строка 1 аппроксимированное значение 3.500000*

*Строка 2 аппроксимированное значение 6.500000*

*Столбец аппроксимированное значение 5.000000*

2.Какова минимальная степень двумерного полинома, построенного на четырех узлах? На шести узлах?

*По 4 узлам можно построить двумерный полином от 0 до 3 степени*

*По 6 узлам можно построить двумерный полином от 0 до 5 степени*

3.Предложите алгоритм двумерной интерполяции при хаотичном расположении узлов, т.е. когда таблицы функции на регулярной сетке нет, и метод последовательной интерполяции не работает. Какие имеются ограничения на расположение узлов при разных степенях полинома?

*При интерполяции на хаотичном наборе точек необходимо выбрать 3 точки, лежащие возле интерполируемой и по ним вычислить коэффициенты уравнения z = ay + bx + c (Полином 1 степени). Коэффициенты можно вычислить методом Гауса.*

*Можно сделать тоже самое для полинома 2 степени и 6 точек.*

*Точки построения полинома не должны лежать на одной прямой.*

4. Пусть на каком-либо языке программирования написана функция, выполняющая интерполяцию по двум переменным. Опишите алгоритм использования этой функции для интерполяции по трем переменным.

*Выберем nz+1 точек, лежащих в одной плоскости с искомой (невдалеке от неё , но не на одной прямой)*

*Используем функцию двумерной интерполяции по перпендикулярным плоскостям для нахождения значений выбранных точек.*

*Проводим еще одну двумерную интерполяцию на полученных точках, для искомого значения.*

5. Можно ли при последовательной интерполяции по разным направлениям использовать полиномы несовпадающих степеней или даже разные методы одномерной интерполяции, например, полином Ньютона и сплайн?

*Можно, при двумерной интерполяции необходимо только составить таблицу значений для конечного полинома, метод вычисления этих значений роли не играет.*

6. Опишите алгоритм двумерной интерполяции на треугольной конфигурации узлов.

*Вычисление значения двумерной интерполяции на треугольной конфигурации узлов производится по формуле*