Вариант 4.

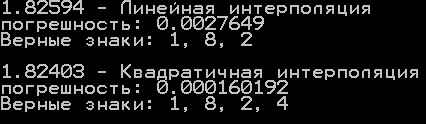
Задание: найти приближенное значение функции при данном значении аргумента с помощью интерполяционного многочлена Лагранжа, если функция задана: 1) в неравноотстоящих узлах таблицы; 2) в равноотстоящих узлах таблицы.

К заданию 1:

x = 0,616

|  |  |
| --- | --- |
| x | y |
| 0,41 | 2,57418 |
| 0,46 | 2,32513 |
| 0,52 | 2,09336 |
| 0,60 | 1,86203 |
| 0,65 | 1,74926 |
| 0,72 | 1,62098 |

Результаты работы программы 1:

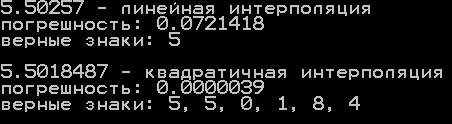


К заданию 2:

x = 0,1838

|  |  |
| --- | --- |
| x | y |
| 0,180 | 5,61543 |
| 0,185 | 5,46693 |
| 0,190 | 5,32634 |
| 0,195 | 5,19304 |
| 0,200 | 5,06649 |
| 0,205 | 4,94619 |

Результаты работы программы 2:



Программа 1

#include <iostream>

#include <vector>

#include <string>

#include <math.h>

#include <iomanip>

using namespace std;

void vern\_znaki(double pogr, double res)

{

int count = 0;

while (((int)(pogr \* 10)) == 0)

{

count++;

pogr \*= 10;

}

if (((int)(pogr \* 10)) > 5)

count--;

res \*= pow(10, count);

int r = (int)res;

int \*mas = new int[1000];

int it = 0;

while (r != 0)

{

mas[it++] = r % 10;

r /= 10;

}

cout << "Верные знаки: ";

for (int i = it - 1; i >= 0; i--)

{

if (i > 0)

cout << mas[i] << ", ";

else

cout << mas[i];

}

cout << endl << endl;

}

int main()

{

setlocale(LC\_ALL, "Russian");

double x = 0.616f, x1, x2, x3, x4, x5, x6, y1, y2, y3, y4, y5, y6, res1, res2, raz11, raz12, raz13, razz, raz21, raz22, razzz, pogr;

vector<double> X(6);

vector<double> Y(6);

X[0] = 0.41f;

X[1] = 0.46f;

X[2] = 0.52f;

X[3] = 0.60f;

X[4] = 0.65f;

X[5] = 0.72f;

Y[0] = 2.57418f;

Y[1] = 2.32513f;

Y[2] = 2.09336f;

Y[3] = 1.86203f;

Y[4] = 1.74926f;

Y[5] = 1.62098f;

double min = abs(x - X[0]);

int ind;

for (int i = 0; i < 6; i++)

{

if (abs(x - X[i]) < min)

{

min = abs(x - X[i]);

ind = i;

}

}

double X1, X2, X3, Y1, Y2, Y3;

if (x - X[ind] > 0)

{

if (ind < 4)

{

X1 = X[ind];

Y1 = Y[ind];

X2 = X[ind + 1];

Y2 = Y[ind + 1];

X3 = X[ind + 2];

Y3 = Y[ind + 2];

}

else

{

X1 = X[ind];

Y1 = X[ind];

X2 = X[ind + 1];

Y2 = Y[ind + 1];

X3 = X[ind - 1];

Y3 = Y[ind - 1];

}

}

else

{

if (ind < 5)

{

X1 = X[ind - 1];

Y1 = Y[ind - 1];

X2 = X[ind];

Y2 = Y[ind];

X3 = X[ind + 1];

Y3 = Y[ind + 1];

}

else

{

X1 = X[ind - 1];

Y1 = Y[ind - 1];

X2 = X[ind];

Y2 = Y[ind];

X3 = X[ind - 2];

Y3 = Y[ind - 2];

}

}

res1 = ((Y1 \* (x - X2)) / (X1 - X2)) + ((Y2 \* (x - X1)) / (X2 - X1));

cout << res1 << " - Линейная интерполяция" << endl;

raz11 = ((Y[1] - Y[0]) / (X[1] - X[0]));

raz12 = ((Y[2] - Y[1]) / (X[2] - X[1]));

razz = ((raz12 - raz11) / (X[2] - X[0]));

int factorial = 2;

pogr = (abs(razz) / factorial) \* abs((x - X1) \* (x - X2));

cout << "погрешность: " << pogr << endl;

vern\_znaki(abs(pogr), res1);

res2 = ((Y1 \* (x - X2) \* (x - X3)) / ((X1 - X2) \* (X1 - X3))) + ((Y2 \* (x - X1) \* (x - X3)) / ((X2 - X1) \* (X2 - X3))) + ((Y3 \* (x - X1) \* (x - X2)) / ((X3 - X1) \* (X3 - X2)));

cout << res2 << " - Квадратичная интерполяция" << endl;

raz13 = ((Y[3] - Y[2]) / (X[3] - X[2]));

raz21 = ((raz12 - raz11) / (X[2] - X[0]));

raz22 = ((raz13 - raz12) / (X[3] - X[1]));

razzz = ((raz22 - raz21) / (X[3] - X[0]));

factorial \*= (factorial+1);

pogr = (abs(razzz) / factorial) \* abs((x - X1) \* (x - X2) \* (x - X3));

cout << "погрешность: " << pogr << endl;

vern\_znaki(abs(pogr), res2);

system("pause");

}

Программа 2

#include <iostream>

#include <vector>

#include <string>

#include <math.h>

#include <iomanip>

using namespace std;

void vern\_znaki(double pogr, double res)

{

int count = 0;

while (((int)(pogr \* 10)) == 0)

{

count++;

pogr \*= 10;

}

if (((int)(pogr \* 10)) > 5)

count--;

res \*= pow(10, count);

int r = (int)res;

int\* mas = new int[1000];

int it = 0;

while (r != 0)

{

mas[it++] = r % 10;

r /= 10;

}

cout << "верные знаки: ";

for (int i = it - 1; i >= 0; i--)

{

if (i > 0)

cout << mas[i] << ", ";

else

cout << mas[i];

}

cout << endl << endl;

}

int main()

{

setlocale(LC\_ALL, "Russian");

double x = 0.1838f, x1, x2, x3, x4, x5, x6, y1, y2, y3, y4, y5, y6, res1, res2, raz11, raz12, raz13, razz, raz21, raz22, razzz, pogr;

vector<double> X(6);

vector<double> Y(6);

X[0] = 0.180f;

X[1] = 0.185f;

X[2] = 0.190f;

X[3] = 0.195f;

X[4] = 0.200f;

X[5] = 0.205f;

Y[0] = 5.61543f;

Y[1] = 5.46693f;

Y[2] = 5.32634f;

Y[3] = 5.19304f;

Y[4] = 5.06649f;

Y[5] = 4.94619f;

double min = abs(x - X[0]);

int ind;

for (int i = 0; i < 6; i++)

{

if (abs(x - X[i]) < min)

{

min = abs(x - X[i]);

ind = i;

}

}

double X1, X2, X3, Y1, Y2, Y3;

if (x - X[ind] > 0)

{

if (ind < 4)

{

X1 = X[ind];

Y1 = Y[ind];

X2 = X[ind + 1];

Y2 = Y[ind + 1];

X3 = X[ind + 2];

Y3 = Y[ind + 2];

}

else

{

X1 = X[ind];

Y1 = X[ind];

X2 = X[ind + 1];

Y2 = Y[ind + 1];

X3 = X[ind - 1];

Y3 = Y[ind - 1];

}

}

else

{

if (ind < 5)

{

X1 = X[ind - 1];

Y1 = Y[ind - 1];

X2 = X[ind];

Y2 = Y[ind];

X3 = X[ind + 1];

Y3 = Y[ind + 1];

}

else

{

X1 = X[ind - 1];

Y1 = Y[ind - 1];

X2 = X[ind];

Y2 = Y[ind];

X3 = X[ind - 2];

Y3 = Y[ind - 2];

}

}

double h = X[1] - X[0];

res1 = ((Y1 \* (x - X2)) / (-h)) + ((Y2 \* (x - X1)) / h);

cout << res1 << " - линейная интерполяция" << fixed << setprecision(7) << endl;

raz11 = ((Y[1] - Y[0]) / h);

raz12 = ((Y[2] - Y[1]) / h);

razz = ((raz12 - raz11) / (2 \* h \* h));

int factorial = 2;

pogr = (abs(razz) / factorial) \* abs((x - X1) \* (x - X2));

cout << "погрешность: " << fixed << setprecision(7) << pogr << endl;

vern\_znaki(abs(pogr), res1);

res2 = ((Y1 \* (x - X2) \* (x - X3)) / ((-h) \* (-2 \* h))) + ((Y2 \* (x - X1) \* (x - X3)) / ((h) \* (-h))) + ((Y3 \* (x - X1) \* (x - X2)) / ((2 \* h) \* (h)));

cout << res2 << " - квадратичная интерполяция" << fixed << setprecision(7) << endl;

raz13 = ((Y[3] - Y[2]) / (h));

raz21 = ((raz12 - raz11) / (2 \* h));

raz22 = ((raz13 - raz12) / (2 \* h));

razzz = ((raz22 - raz21) / (3 \* h));

factorial \*= (factorial + 1);

pogr = (abs(razzz) / factorial) \* abs((x - X1) \* (x - X2) \* (x - X3));

cout << "погрешность: " << fixed << setprecision(7) << pogr << endl;

vern\_znaki(abs(pogr), res2);

system("pause");

}