Варыянт 3

Заданне 4. В соответствии со своим вариантом отделить корни двух уравнений и вычислить их методом дихотомии для ис-ходных данных из таблицы, приведенной ниже. Точность вычислений принять равной e = 0,001 для всех вариантов.

Операторы метода вычисления корня оформить в виде функции пользователя, уравнения записать также в виде функций пользователя.

В главной функции предусмотреть ввод исходных данных, обращения к функции, реализующей метод дихотомии для двух уравнений. В процессе выполнения программы определить корни двух уравнений. Использовать указатель на функцию.



#include <stdio.h>

#include <math.h>

#include "Windows.h"

// Ураўненне 1: x^3 + 2x - 3 у пункце х

double equation1(double x) {

return pow(x, 3) + 2 \* x - 3;

}

// Ураўненне 2: e^x - 2 у пункце х

double equation2(double x) {

return exp(x) - 2;

}

// Метад дыхатаміі, прымаем паказальнік на функцыю, межы і значэнне e

double dichotomyMethod(double (\*equation)(double), double a, double b, double epsilon) {

double fa, fb, fx, x;

while (fabs(b - a) > epsilon) {

// Вылічэнне сярэдзіну інтэрвалу

x = (a + b) / 2;

// Вылічэнне значэнні ураўнення ў пунктах a, b і x

fa = equation(a);

fb = equation(b);

fx = equation(x);

// Калі знакі функцыі ў кропках a і x розныя, корань знаходзіцца паміж a і x

if (fa \* fx < 0) {

b = x; // Значэнне b устанаўліваецца роўным x

}

else {

a = x; // Інакш, корань знаходзіцца паміж b і x, значэнне a устанаўліваецца роўным x

}

}

return x;

}

int main() {

double a, b, e = 0.001;

SetConsoleOutputCP(1251);

// Увод інтэрвалу

printf("Увядзіце інтэрвал [a, b]: ");

// Функцыя scanf\_s вяртае колькасць паспяхова супастаўленых і прачытаных элементаў канкрэтнага тыпу дадзеных

// Таму

if (scanf\_s("%lf %lf", &a, &b) != 2) { // Калі не 2, то не былі прачытаны абедзве межы

printf("Памылка ўводу.\n");

return 1; // Вяртаем памылку

}

// Вылічэнне каранёў ураўненняў метадам дыхатаміі

double root1 = dichotomyMethod(equation1, a, b, e);

double root2 = dichotomyMethod(equation2, a, b, e);

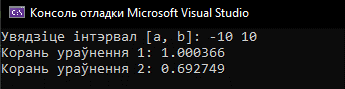
// Выснова вынікаў

printf("Корань ураўнення 1: %lf\n", root1);

printf("Корань ураўнення 2: %lf\n", root2);

return 0;

}



Заданне 5. В соответствии со своим вариантом написать программы по условиям задач из таблицы ниже. Программа должна содержать функцию пользователя с переменным числом параметров и не менее трех обращений к ней с различным ко-личеством параметров.



#include <iostream>

template <typename T> // Выкарыстоўваю шаблон для магчымасці выкарыстоўваць функцыю з тыпам int і double. T – назва тыпу дадзеных. замест можна было б выкарыстоўваць перагрузку і не толькі

T fmax(int ​​count, T first, ...) { // count - цэлы лік, колькасць элементаў. T first - першы элемент са спісу параметраў. дзякуючы шаблону з яго дапамогай будзем прадвызначаць тып дадзеных усіх наступных параметраў

T minNum = first; // За мінімальны бяру самы першы і буду параўноўваць усе лікі з ім

T\* ptr = &first; // Паказальнік на first

for (int i = 1; i < count; i++) { // паколькі count - лік параметраў, а fisrt захоўваецца ў асобную зменную, то ітэрацый цыклу трэба на 1 менш

if (\*++ptr < minNum) { // першапачаткова павялічваем паказальнік, бо ён на першай ітэрацыі цыклу паказвае на fisrt - першы параметр, а патрэбны з другога па апошнi

minNum = \*ptr;

}

}

return minNum;

}

int main() {

// Толькі чаму ва ўмове fmax калі шукаем мінімальны?

std::cout << "Min: " << fmax(3, 1.2, 3.4, 0.0) << std::endl;

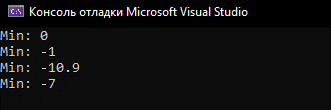
std::cout << "Min: " << fmax(5, 7, 2, 9, -1, 5) << std::endl;

std::cout << "Min: " << fmax(6, -10.9, -5.342, -8.423, -2.802472390, -1.1, -7.1117) << std::endl;

std::cout << "Min: " << fmax(8, 10, -5, 8, -2, 1, -7, 3, 0) << std::endl;

return 0;

}



Варыянт 1

Заданне 4



#include <stdio.h>

#include <math.h>

#include "Windows.h"

// Ураўненне 1: x^2 + 4x - 2 у пункце х

double equation1(double x) {

return pow(x, 2) + 4 \* x - 2;

}

// Ураўненне 2: sin(x) + 0.1 у пункце х

double equation2(double x) {

return sin(x) + 0.1;

}

// Метад дыхатаміі, прымаем паказальнік на функцыю, межы і значэнне e

double dichotomyMethod(double (\*equation)(double), double a, double b, double epsilon) {

double fa, fb, fx, x;

while (fabs(b - a) > epsilon) {

// Вылічэнне сярэдзіну інтэрвалу

x = (a + b) / 2;

// Вылічэнне значэнні ураўнення ў пунктах a, b і x

fa = equation(a);

fb = equation(b);

fx = equation(x);

// Калі знакі функцыі ў кропках a і x розныя, корань знаходзіцца паміж a і x

if (fa \* fx < 0) {

b = x; // Значэнне b устанаўліваецца роўным x

}

else {

a = x; // Інакш, корань знаходзіцца паміж b і x, значэнне a устанаўліваецца роўным x

}

}

return x;

}

int main() {

double a, b, e = 0.001;

SetConsoleOutputCP(1251);

// Увод інтэрвалу

printf("Увядзіце інтэрвал [a, b]: ");

// Функцыя scanf\_s вяртае колькасць паспяхова супастаўленых і прачытаных элементаў канкрэтнага тыпу дадзеных

// Таму

if (scanf\_s("%lf %lf", &a, &b) != 2) { // Калі не 2, то не былі прачытаны абедзве межы

printf("Памылка ўводу.\n");

return 1; // Вяртаем памылку

}

// Вылічэнне каранёў ураўненняў метадам дыхатаміі

double root1 = dichotomyMethod(equation1, a, b, e);

double root2 = dichotomyMethod(equation2, a, b, e);

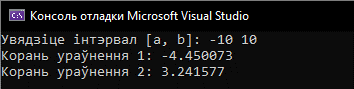
// Выснова вынікаў

printf("Корань ураўнення 1: %lf\n", root1);

printf("Корань ураўнення 2: %lf\n", root2);

return 0;

}



Заданне 5



#include <iostream>

int fmin(int count, int first, ...) { // count - цэлы лік, колькасць элементаў. first - першы элемент са спісу параметраў

int minNum = first; // За мінімальны бяру самы першы і буду параўноўваць усе лікі з ім

int\* ptr = &first; // Паказальнік на first

for (int i = 1; i < count; i++) { // паколькі count - лік параметраў, а fisrt захоўваецца ў асобную зменную, то ітэрацый цыклу трэба на 1 менш

if (\*++ptr < minNum) { // першапачаткова павялічваем паказальнік, бо ён на першай ітэрацыі цыклу паказвае на fisrt - першы параметр, а патрэбны з другога па апошнi

minNum = \*ptr;

}

}

return minNum;

}

int main() {

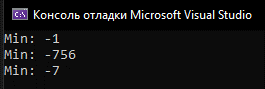
std::cout << "Min: " << fmin(5, 7, 2, 9, -1, 5) << std::endl;

std::cout << "Min: " << fmin(8, 13, -5, 534, -645, 1, -756, 867, 0) << std::endl;

std::cout << "Min: " << fmin(8, 10, -5, 8, -2, 1, -7, 3, 0) << std::endl;

return 0;

}



Варыянт 2

Заданне 4



#include <stdio.h>

#include <math.h>

#include "Windows.h"

// Ураўненне 1: x^3 + x - 3 у пункце х

double equation1(double x) {

return pow(x, 3) + x - 3;

}

// Ураўненне 2: cos^3(x) у пункце х

double equation2(double x) {

return pow(cos(x), 3);

}

// Метад дыхатаміі, прымаем паказальнік на функцыю, межы і значэнне e

double dichotomyMethod(double (\*equation)(double), double a, double b, double epsilon) {

double fa, fb, fx, x;

while (fabs(b - a) > epsilon) {

// Вылічэнне сярэдзіну інтэрвалу

x = (a + b) / 2;

// Вылічэнне значэнні ураўнення ў пунктах a, b і x

fa = equation(a);

fb = equation(b);

fx = equation(x);

// Калі знакі функцыі ў кропках a і x розныя, корань знаходзіцца паміж a і x

if (fa \* fx < 0) {

b = x; // Значэнне b устанаўліваецца роўным x

}

else {

a = x; // Інакш, корань знаходзіцца паміж b і x, значэнне a устанаўліваецца роўным x

}

}

return x;

}

int main() {

double a, b, e = 0.001;

SetConsoleOutputCP(1251);

// Увод інтэрвалу

printf("Увядзіце інтэрвал [a, b]: ");

// Функцыя scanf\_s вяртае колькасць паспяхова супастаўленых і прачытаных элементаў канкрэтнага тыпу дадзеных

// Таму

if (scanf\_s("%lf %lf", &a, &b) != 2) { // Калі не 2, то не былі прачытаныя абедзве межы

printf("Памылка ўводу.\n");

return 1; // Вяртаем памылку

}

// Вылічэнне каранёў ураўненняў метадам дыхатаміі

double root1 = dichotomyMethod(equation1, a, b, e);

double root2 = dichotomyMethod(equation2, a, b, e);

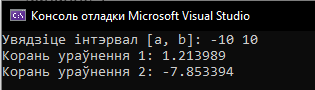
// Выснова вынікаў

printf("Корань ураўнення 1: %lf\n", root1);

printf("Корань ураўнення 2: %lf\n", root2);

return 0;

}



Заданне 5



#include <iostream>

int fmin(int count, int first, ...) { // count - цэлы лік, колькасць элементаў. first - першы элемент са спісу параметраў

int minNum = 0; // Сума

int\* ptr = &first; // Паказальнік на first

for (int i = 0; i < count-1; i++) { // count - 1 паколькі у індэксах выкарыстоўваю + 1 да бягучага, как не было памылак

minNum += \*ptr++ \* \*(ptr + 1); // Знаходзім здабытак і дабаўляем да зменнай

}

return minNum;

}

int main() {

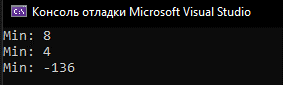
std::cout << "Min: " << fmin(3, 1, 2, 3) << std::endl;

std::cout << "Min: " << fmin(4, 1, -2, 3, 4) << std::endl;

std::cout << "Min: " << fmin(8, 10, -5, 8, -2, 1, -7, 3, 0) << std::endl;

return 0;

}



Варыянт 4

Заданне 4



#include <stdio.h>

#include <math.h>

#include "Windows.h"

// Ураўненне 1: x^3 + 2x - 1 у пункце х

double equation1(double x) {

return pow(x, 3) + 2 \* x - 1;

}

// Ураўненне 2: e^x - 2 у пункце х

double equation2(double x) {

return exp(x) - 2;

}

// Метад дыхатаміі, прымаем паказальнік на функцыю, межы і значэнне e

double dichotomyMethod(double (\*equation)(double), double a, double b, double epsilon) {

double fa, fb, fx, x;

while (fabs(b - a) > epsilon) {

// Вылічэнне сярэдзіну інтэрвалу

x = (a + b) / 2;

// Вылічэнне значэнні ураўнення ў пунктах a, b і x

fa = equation(a);

fb = equation(b);

fx = equation(x);

// Калі знакі функцыі ў кропках a і x розныя, корань знаходзіцца паміж a і x

if (fa \* fx < 0) {

b = x; // Значэнне b устанаўліваецца роўным x

}

else {

a = x; // Інакш, корань знаходзіцца паміж b і x, значэнне a устанаўліваецца роўным x

}

}

return x;

}

int main() {

double a, b, e = 0.001;

SetConsoleOutputCP(1251);

// Увод інтэрвалу

printf("Увядзіце інтэрвал [a, b]: ");

// Функцыя scanf\_s вяртае колькасць паспяхова супастаўленых і прачытаных элементаў канкрэтнага тыпу дадзеных

// Таму

if (scanf\_s("%lf %lf", &a, &b) != 2) { // Калі не 2, то не былі прачытаныя абедзве межы

printf("Памылка ўводу.\n");

return 1; // Вяртаем памылку

}

// Вылічэнне каранёў ураўненняў метадам дыхатаміі

double root1 = dichotomyMethod(equation1, a, b, e);

double root2 = dichotomyMethod(equation2, a, b, e);

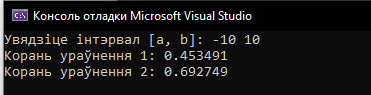
// Выснова вынікаў

printf("Корань ураўнення 1: %lf\n", root1);

printf("Корань ураўнення 2: %lf\n", root2);

return 0;

}



Заданне 5



#include <iostream>

int fmin(int count, int first, ...) { // count - цэлы лік, колькасць элементаў. first - першы элемент са спісу параметраў

int maxNum = first; // За мінімальны бяру самы першы і буду параўноўваць усе лікі з ім

int\* ptr = &first; // Паказальнік на first

for (int i = 1; i < count; i++) { // паколькі count - лік параметраў, а fisrt захоўваецца ў асобную зменную, то ітэрацый цыклу трэба на 1 менш

if (\*++ptr > maxNum) { // першапачаткова павялічваем паказальнік, бо ён на першай ітэрацыі цыклу паказвае на fisrt - першы параметр, а патрэбны з другога па апошнi

maxNum = \*ptr;

}

}

return maxNum;

}

int main() {

std::cout << "Max: " << fmin(5, 7, 2, 9, -1, 5) << std::endl;

std::cout << "Max: " << fmin(8, 13, -5, 534, -645, 1, -756, 867, 0) << std::endl;

std::cout << "Max: " << fmin(8, 10, -5, 8, -2, 1, -7, 3, 0) << std::endl;

return 0;

}

