Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

**Лабораторная работа №4**

**По дисциплине ЯП за 2 семестр**  
**Тема: «Подключение внешних модулей и применение параллельного**

**подхода»**

**Выполнил:**

Студент группы ПО-6(1)  
 1-го курса

Мартынович Даниил

**Проверила:**

Анфилец С.В

Брест, 2021

Задание.

1. Выделить в отдельный файл функции: расчёта функционала, пересчёта новой позиции

точки (вспомогательные, если есть).

2. Реализовать их в виде подключаемых библиотек (создать заголовочный файл \*.h и пр. )

3. Подключить вашу библиотеку к новому проекту( #include).

4. Создать в новом проекте популяцию из 1000 элементов (A[1000][3]).

5. Реализовать популяционный алгоритм поиска экстремума. Реализация должна

использовать метод параллельных потоков. Распараллеливание алгоритма предлагается

сделать в месте пересчёта новых значений точек (т.к. их количество 1000 и изменение каждой

из них не зависит от других).

Алгоритм поиска экстремума, используя популяцию:

1. Инициализировать: Популяцию, Массив, хранящий текущее значение функции для этой

точки. Задать стартовое значение шага и параметр его затухания.

2. Для каждого элемента пытаемся передвинуть точку в новое положение (см. пред. Лаб.

работу), т.е. если значение функции будет улучшено происходит изменение положения точки,

иначе возвращается предыдущее значение точки (вызов библиотечной функции // можно

вспомнить: передача массива в/из функций).

3. После движения всех точек изменить шаг (см. пред. Лаб.работу).

4. Проверка точки Останова, если выполняется, завершаем алгоритм, иначе переходим к п.2.

Условие Останова: значение шаг стало меньше минимального. Решением задачи будет: точка

из популяции, которой соответствует наименьшее значение функции.

#include <iostream>

#include <math.h>

#include <ctime>

#include "Header.h"

#include<omp.h>

using namespace std;

int main()

{

srand(time(0));

setlocale(LC\_ALL, "RU");

int a, b;

double array[1000];

int num = 0;

double step, x0;

string username;

cout << "Введите фамилию: ";

cin >> username;

a = username.length();

cout << "Введите значение в диапозоне [2,10]: ";

cin >> b;

if (b < 2 || b>10)

{

cout << "Ошибка ввода диавозона" << endl;

system("pause");

return 0;

}

cout << "Введите x0= ";

cin >> x0;

cout << "Введите шаг: ";

cin >> step;

cout << "x\t" << "\ty" << endl;

#pragma omp parallel for

for (size\_t i = 0; i < 1000; i++)

{

if (step < 0.000001)

{

break;

}

array[i] = function(x0, a, b);

cout << fixed << x0 << "\t" << array[i] << endl;

x0 = x0 + step \* Rand();

step = step \* Rand2();

}

system("pause");

}

#pragma once

double function(double x, int a, int b);

double Rand();

double Rand2();

#include <math.h>

#include<iostream>

double function(double x, int a, int b)

{

double y = cos(x) + (cos(a \* x + 1) / b) + (cos(pow(a, 2) \* x + 2) / pow(b, 2)) + (cos(pow(a, 3) \* x + 3) / pow(b, 3)) + (cos(pow(a, 4) \* x + 4) / pow(b, 4));

return y;

}

double Rand()

{

double massive[10] = { -1, -0.75, -0.50, -0.25, 0, 0.25, 0.50, 0.75, 1 };

int n = rand() % 10;

double num = massive[n];

return num;

}

double Rand2()

{

double massive[6] = { 0.75,0.80,0.85,0.90,0.95,1 };

int n = rand() % 6;

double num = massive[n];

return num;

}



