Образец оформления отчёта по Лабораторной работе.

|  |  |
| --- | --- |
| **К Г Э У** | МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |

**Кафедра Информатики и информационных управляющих систем**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9**

**ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ. РАБОТА СО ВЛОЖЕННЫМИ ЦИКЛАМИ.**

|  |  |
| --- | --- |
| **Исполнитель:** | Банницин Дмитрий |
| **Группа:** | ТРП-3-22 |
| **Вариант:** | 35 |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Казань -2022

**Задания для самостоятельной работы**

**Задача №1**

Составить функцию series(x, N) для получения суммы конечного ряда Л/р №5 (Задание 1). Построить таблицу значений суммы ряда для разных значений N = {5, 10, 15, 20, 25} при фиксированном значении х и оценить, как изменяется значение суммы от величины N. Построить из этих сумм массив. В отдельной функции определить, при каком значении N сумма перестает изменяться более чем на 10% от предыдущего значения, или, если этого не происходит, в другой функции оценить, при каком N достигается максимальное значение суммы.

**Контрольный пример**

****

**Код**

#include <iostream>

using namespace std;

float\* getSum(float x);

long double factorial(int i);

bool getN(float\* array);

int getMaxN(float\* array);

int main()

{

float x;

cout << "x = "; cin >> x;

float \* results;

results = getSum(x);

if (!getN(results)) {

cout << "maxN = " << getMaxN(results) << endl;

}

system("pause");

return 0;

}

float\* getSum(float x) {

float pi = acos(-1);

float U;

float S;

int i;

long double cosix, fact;

float \*result = new float[5];

for (size\_t N = 5; N <= 25; N +=5)

{

U = -1;

S = 0;

for (i = 1, S = 0; i <= N; i++)

{

U = -U;

cosix = cos(i \* x / 180 \* pi);

fact = factorial(i \* 2);

S += U \* cosix / fact;

}

cout << "\n S(" << x << ',' << N << ")= " << S << endl;

result[N / 5 - 1] = S;

}

cout << endl;

return &result[0];

}

long double factorial(int i)

{

if (i == 0) return 1;

else return i \* factorial(i - 1);

}

bool getN(float\* array) {

int maxN = 0;

bool result = false;

for (size\_t i = 1; i < 5; i++)

{

if (abs(array[i] - array[i - 1]) > 0.1 \* array[i - 1])

{

maxN = i + 1 \* 5;

result = true;

}

}

if (maxN != 0)

cout << "N = " << maxN << endl;

return result;

}

int getMaxN(float\* array) {

int N[]{5, 10, 15, 20, 25};

for (size\_t i = 0; i < 4; i++)

{

for (size\_t j = 0; j < 4 - i; j++)

{

if (array[j] > array[j+1])

{

swap(array[j], array[j + 1]);

swap(N[j], N[j+1]);

}

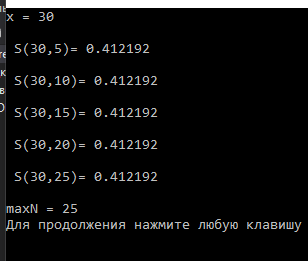
}

}

return N[4];

}

**Решение**

****

**Задача №2**

На плоскости задаётся множество 3 < n < 7 точек с целочисленными координатами ∈[−2;+3]; ∈[−3;+2] i i x y . Определить, образуют ли эти точки выпуклый многоугольник. Если образуют, найти самую длинную диагональ. Указание: можно использовать следующее свойство выпуклого многоугольника: сумма углов выпуклого n-угольника равна 180° (n – 2). Для определения угла использовать свойство скалярного произведения векторов.

**Код**

#include <iostream>

#include <time.h>

using namespace std;

const double PI = acos(-1);

void getCoords(pair<int, int>\* coords, int size);

void getVectors(pair<int, int>\* coords, int size, pair<int, int>\* vectors);

bool isConvex(pair<int, int>\* vectors, int size);

int main()

{

srand(time(0));

int n;

n = rand() % 3 + 4;

pair<int, int>\* coords = new pair<int, int>[n];

getCoords(coords, n);

cout << "points:\n";

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

{

cout << "(" << coords[i].first << "; " << coords[i].second << ")" << endl;

}

pair<int, int>\* vectors = new pair<int, int>[n];

getVectors(coords, n, vectors);

cout << "------------------\n";

cout << "vectors:\n";

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

{

cout << "(" << vectors[i].first << "; " << vectors[i].second << ")" << endl;

}

if (isConvex)

cout << "\nIs not convex\n";

else

cout << "\nIs not convex\n";

}

void getCoords(pair<int, int>\* coords, int size) {

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

{

coords[i].first = rand() % 6 - 2;

coords[i].second = rand() % 6 - 3;

}

}

void getVectors(pair<int, int>\* coords, int size, pair<int, int>\* vectors) {

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

{

if (i == size - 1) {

vectors[i].first = coords[0].first - coords[i].first;

vectors[i].second = coords[0].second - coords[i].second;

return;

}

vectors[i].first = coords[i+1].first - coords[i].first;

vectors[i].second = coords[i + 1].second - coords[i].second;

}

}

bool isConvex(pair<int, int>\* vectors, int size) {

float anglesSum = 0;

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

{

if (i = size - 1)

{

double vectorsMultiply = vectors[i].first \* vectors[0].first + vectors[i].second \* vectors[0].second;

double vector1Length = sqrt(pow(vectors[i].first, 2) + pow(vectors[i].second, 2));

double vector2Length = sqrt(pow(vectors[0].first, 2) + pow(vectors[0].second, 2));

double cosa = vectorsMultiply / vector1Length / vector2Length;

anglesSum += (acos(cosa) \* 180 / PI);

break;

}

double vectorsMultiply = vectors[i].first \* vectors[i + 1].first + vectors[i].second \* vectors[i + 1].second;

double vector1Length = sqrt(pow(vectors[i].first, 2) + pow(vectors[i].second, 2));

double vector2Length = sqrt(pow(vectors[i+1].first, 2) + pow(vectors[i+1].second, 2));

double cosa = vectorsMultiply / vector1Length / vector2Length;

anglesSum += (acos(cosa) \* 180 / PI);

}

if (anglesSum == 180 \* (size - 2))

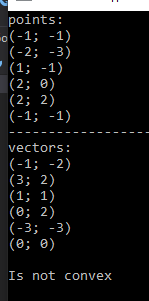
return true;

else

return false;

}

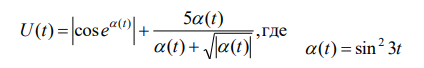
**Решение**

****

**Домашнее задание.**

**Задание №1.**

В течение 30 секунд проводится эксперимент. Каждую секунду снимается сигнал Uэ(t). Аналитический вид функции U(t) описывается выражением:



Составить программу, моделирующую данный эксперимент

**Код**

#include <iostream>

#include <ctime>

using namespace std;

float\* getValues();

float aT(int t);

int getMaxN(float\* array);

int getMinN(float\* array);

void getIntervals(float\* array);

int main()

{

float\* values = new float[30];

srand(time(0));

values = getValues();

for (size\_t i = 0; i < 30; i++)

{

cout << i << " " << values[i] << "\n";

}

cout << "\nmaxN = " << getMaxN(values) << endl;

cout << "minN = " << getMinN(values) << endl;

getIntervals(values);

}

float\* getValues() {

float\* values = new float[30];

int kd;

for (size\_t i = 1; i < 31; i++)

{

kd = rand() % 10;

float coseat = abs(cos(exp(aT(i))));

float chis = 5 \* aT(i);

float znam = aT(i) + sqrt(abs(aT(i)));

float Ut = coseat + chis / znam;

float d = kd \* Ut / 100;

if (rand() % 2)

values[i] = Ut + d;

else

values[i] = Ut - d;

}

return &values[0];

}

float aT(int t) {

return pow(sin(3\*t), 2);

}

int getMaxN(float\* array) {

float max = -9999999;

int maxN = 0;

for (size\_t i = 0; i < 30; i++)

{

if (array[i] > max)

{

max = array[i];

maxN = i;

}

}

return maxN;

}

int getMinN(float\* array) {

float min = 99999999;

int minN = 0;

for (size\_t i = 0; i < 30; i++)

{

if (array[i] < min)

{

min = array[i];

minN = i;

}

}

return minN;

}

void getIntervals(float\* array) {

pair<int, int>\* intervals = new pair<int, int>[30];

for (size\_t i = 0; i < 30; i++)

{

if (array[i] > 2)

{

intervals[i].first = i;

for (size\_t j = i; j < 30; j++)

{

if (j == 29 && array[j] > 2)

{

intervals[i].second = j;

cout << "(" << intervals[i].first << ";" << intervals[i].second << ")\n";

i = j;

}

else if (array[j] < 2)

{

intervals[i].second = j-1;

cout << "(" << intervals[i].first << ";" << intervals[i].second << ")\n";

i = j;

break;

}

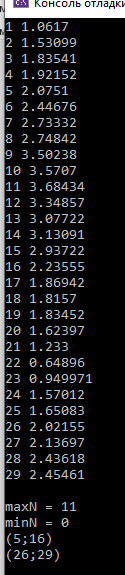
}

}

}

}

**Решение**

****