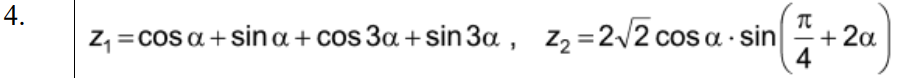
**30333-2 Калевич Сергей Михайлович**

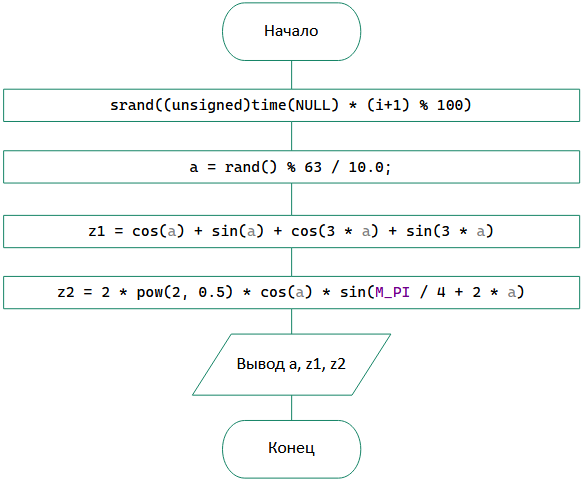
**Лабораторная работа №1 вариант 4**

**Задание 1. Линейный алгоритм**

Составить программу для расчета значений z1 и z2 (результаты должны совпадать).



**Блок-схема**



**Ход выполнения**

variant4\_task1.cpp

// Lab #1

// Task #1

// Variant #4

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <Windows.h>

#include <cmath>

using namespace std;

double absEpsilon = 1e-12;

double relEpsilon = 1e-8;

int testNum {};

double a{};

bool approximatelyEqualAbsRel(double a, double b, double absEpsilon, double relEpsilon);

void test(double z1, double z2);

double formulaZ1(double a)

{

return cos(a) + sin(a) + cos(3 \* a) + sin(3 \* a);

}

double formulaZ2(double a)

{

return 2 \* pow(2, 0.5) \* cos(a) \* sin(M\_PI / 4 + 2 \* a);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

for (int i = 0; i < 5; i++)

{

double z1 = formulaZ1(a);

double z2 = formulaZ2(a);

test(z1, z2);

srand((unsigned)time(NULL) \* (i+1) % 100);

a = rand() % 63 / 10.0; // Покрытие 360 градусов

}

system("pause");

return 0;

}

// Источник: https://ravesli.com/urok-42-operatory-sravneniya/

// Возвращаем true, если разница между a и b меньше absEpsilon или в пределах relEpsilon

bool approximatelyEqualAbsRel(double a, double b, double absEpsilon, double relEpsilon)

{

// Проверяем числа на равенство их друг другу - это нужно в тех случаях,

// когда сравниваемые числа являются нулевыми или "около нуля"

double diff = fabs(a - b);

if (diff <= absEpsilon)

return true;

// В противном случае возвращаемся к алгоритму Кнута

return diff <= ((fabs(a) < fabs(b) ? fabs(b) : fabs(a)) \* relEpsilon);

}

void test(double z1, double z2)

{

if (approximatelyEqualAbsRel(z1, z2, absEpsilon, relEpsilon))

{

testNum++;

cout << "Тест " << testNum << ":\na = " << a

<< ":\nz1 = " << z1 << "\nz2 = " << z2 << "\nOK\n\n";

return;

}

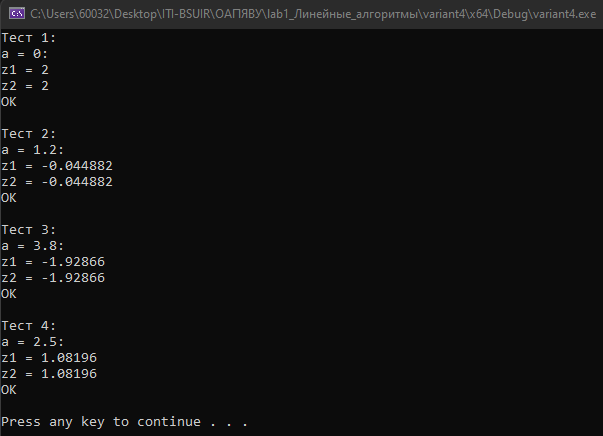
testNum++;

cout << "Тест " << testNum << ":\na = " << a

<< ":\nz1 = " << z1 << "\nz2 = " << z2 << "\nNOT OK\n\n";

}

**Результат**

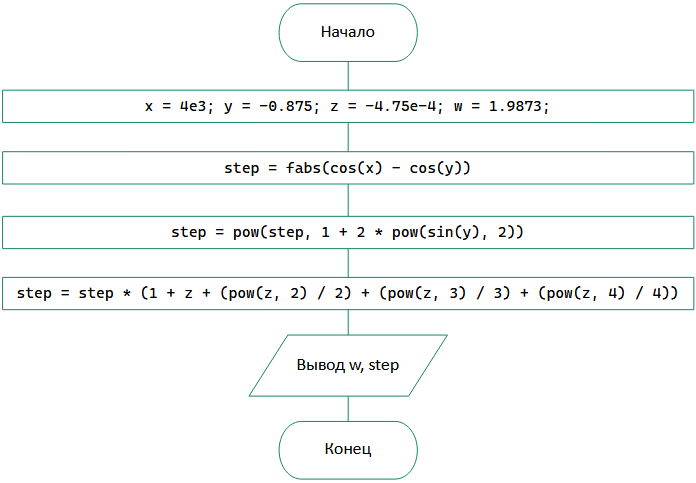


**Задание 2. Линейный алгоритм**

Составить программу для расчета значения следующей функции:

|  |  |
| --- | --- |
| **Условие** | **Контрольный пример** |
|  | |

**Блок-схема**



**Ход выполнения**

variant4\_task2.cpp

// Lab #1

// task #2

// Variant #4

#include <iostream>

#include <Windows.h>

#include <cmath>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

double x = 4e3;

double y = -0.875;

double z = -4.75e-4;

double w = 1.9873;

double step = fabs(cos(x) - cos(y));

step = pow(step, 1 + 2 \* pow(sin(y), 2));

step = step \* (1 + z + (pow(z, 2) / 2) + (pow(z, 3) / 3) + (pow(z, 4) / 4));

printf("%-22s %.4f \n", "Ожидаемый результат:", w);

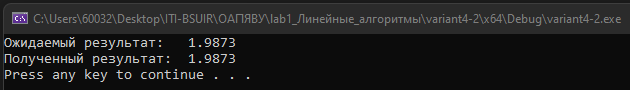
printf("%-22s %.4f \n", "Полученный результат:", step);

system("pause");

return 0;

}

**Результат**



Выводы

Научились: определять и подключать библиотеки, пользоваться вводом/выводом в консоль, определять вид переменных и присваивать значения. Так же проработаны встроенные функции библиотеки Math.h по тригонометрии и возведению в степень.

Так как константа Пи используется в программе – рекомендуется и корректно выбрать тип double, а не float, для расчетов, приближенных к математическим.