**TRANSPORT AND TELECOMMUNICATION INSTITUTE**



**ENGINEERING FACULTY**

**Laboratory work N1**

Course: **Programming**

Theme: **Linear programs. Working with mathematical functions.**

Student: Igors Oļeiņikovs

Student code: 93642

Group: 4501BTA

Riga

2025

**Contents**

[**1. Laboratory work task** 3](#_Toc1)

[**2. Individual task** 3](#_Toc2)

[**3. Algorithm** 4](#_Toc3)

[**4. Source code** 5](#_Toc4)

[**5. Running program example** 6](#_Toc5)

[**6. Testing** 6](#_Toc6)

[**7. Conclusions** 7](#_Toc7)

# **1. Laboratory work task**

## Create an algorithm that calculates values *z1* and *z2* by formulas given in individual task.

# **2. Individual task**

Individual task number is being defined by formula: **taskNr = studentCode % taskVarCount**

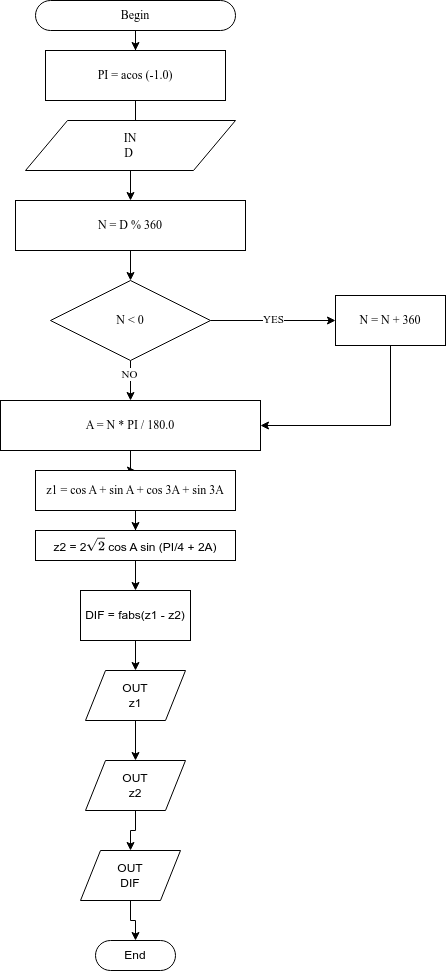
**93642 % 20 = 2**

Z1 = cos α + sin α + cos 3α + sin 3α

Z2 = 2 • cos α • sin ( + 2α)

Create a console application in C/C++ based on created algorithm that calculates *z1* and *z2* by given formulas*.* Output calculation results to a screen (if calculations are made correctly then *z1* and *z2* should be equal, difference is allowed in last digits after point).

# **3. Algorithm**

****Pic. 1. **Algorithm data flow**

# **4. Source code**

#**include** **<iostream>**

#**include** **<cmath>**

**int** main(**void**)

{

**int**         input\_degr;

**int**         normalized\_input;

**double**     alfa\_rad;

**double**     z1;

**double**     z2;

**double**     diff;

**const** **double** PI = acos(-1.0);

std::cout << **"Task Nr "** << 93642%20 << std::endl;

std::cout << **"Enter angle in degrees: "**;

std::cin >> input\_degr;

normalized\_input = input\_degr % 360;

**if** (normalized\_input < 0) normalized\_input += 360;

alfa\_rad = normalized\_input \* PI / 180.0;

z1 = cos(alfa\_rad) + sin(alfa\_rad) + cos(3 \* alfa\_rad) + sin(3 \* alfa\_rad);

z2 = 2 \* sqrt(2) \* cos(alfa\_rad) \* sin(PI / 4 + 2 \* alfa\_rad);

diff = fabs(z1 - z2);

std::cout << **"Z1 = "** << z1 << std::endl;

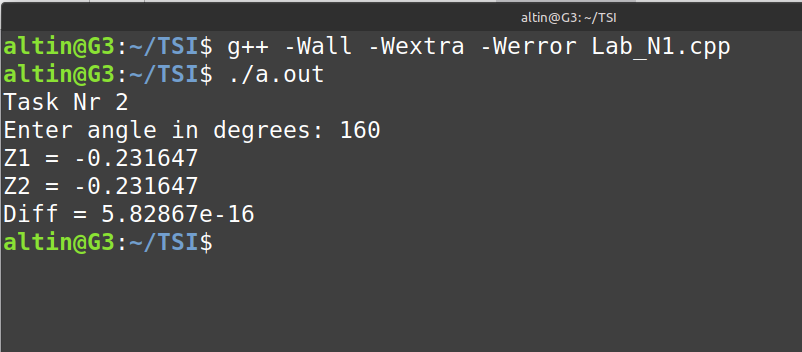
std::cout << **"Z2 = "** << z2 << std::endl;

std::cout << **"Diff = "** << diff << std::endl;

**return** (0);

}

# **5. Running program example**



Pic. 2. **Running program example**

# **6. Testing**

Table **1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Z1** | **Z2** | **Difference** |
| -360 | 2 | 2 | 0 |
| -90 | 3.33067e-16 | 3.67394e-16 | 3.43271e-17 |
| -45 | -1.41421 | -1.41421 | 1.33227e-15 |
| -705 | 2.63896 | 2.63896 | 4.44089e-16 |
| 0 | 2 | 2 | 0 |
| 45 | 1.41421 | 1.41421 | 4.44089e-16 |
| 60 | 0.366025 | 0.366025 | 1.66533e-16 |
| 90 | -2.22045e-16 | -1.22465e-16 | 9.95799e-17 |
| 180 | -2 | -2 | 0 |
| 200 | -2.64774 | -2.64774 | 4.44089e-16 |
| 1300 | -1.77486 | -1.77486 | 2.44249e-15 |

# **7. Conclusions**

Algorithmic flow:

* Normalize input\_degr into [0, 360) → normalized\_input.(Helps reduce rounding errors)
* Convert to radians: alfa = normalized\_input \* PI / 180.
* Compute two mathematically-equivalent expressions:
* z1 = cos(alfa) + sin(alfa) + cos(3alfa) + sin(3alfa)
* z2 = 2 \* sqrt(2) \* cos(alfa) \* sin(PI/4 + 2\*alfa)
* Compute diff = |z1 - z2|.
* Print z1, z2, diff.

Machine epsilon (double precision): eps ≈ 2.22e-16. Any single floating-point operation can introduce relative errors on the order of eps.

Each trigonometric evaluation (cos, sin) returns an approximate result with error roughly O(eps) relative to the true value. The subsequent arithmetic (adds, multiplies) accumulates and propagates those errors.

When the true result is 0 (for example at alfa = 90° both expressions are exactly 0 analytically), the computed terms (e.g., cos(pi/2)) are tiny non-zero values of ±O(eps). Summing several small terms with mixed signs can leave a residual of order a few × eps (e.g., -2.22e-16), and two different algebraic forms (z1 vs z2) do different sequences of operations => different rounding paths and slightly different residues (e.g., -2.22045e-16 vs -1.22465e-16). Their difference is about the same order as eps (≈ 1e-16).

Operation count matters: z1 computes 4 trigonometric calls + 3 additions; z2 uses fewer or different operations (including a multiplication by sqrt(2) and a single trig). Different operation trees produce different rounding errors.

Given expressions are matematically identical and can be mutated to one from other with trigonometric transformations.

**Conclusion**: A difference is expected and is numerical noise, not a correctness bug.