Berejnec Adrian-Daniel 1.1

Metoda Cordic

Pentru implementarea algoritmului am ales limbajul de programare C.

Mediul de lucru este CLion.

Implementare algoritm :

Prima parte :

- Functie pentru a calcula K
- Initializarea variabilelor cu valorile initiale respective

```c
#include <stdio.h>
#include <math.h>

double calculareK(int n) {
    int i;
    double k = 1;

    for(i = 0; i < n; i++)
        k = k * sqrt( _X: 1 + pow( _X: 2, _Y: -2*i));

    return k;
}
double CordicCos(double theta) {
    int n;
    printf( _Format: "Numar pasi:");
    scanf( _Format: "%d", &n);
    double x[n+1], y[n+1], z[n+1];
    x[0] = 1 / calculareK(n);
    y[0] = 0;
    z[0] = theta;

    int i;
```

A doua parte :

- Calculul valorilor cerute cu ajutorul unui “for”
- Afisarea corespunzatoare pentru fiecare iteratie

```c
    for(i = 0; i < n; i++)
    {
        printf( _Format: "x[%d] = %f, y[%d] = %f, z[%d] = %f\n",i,x[i],i,y[i],i,z[i]);
        double exponent = pow( _X: 2, -i);
        if(z[i] >= 0)
        {
            x[i+1] = x[i] - (exponent * y[i]);
            y[i+1] = y[i] + (exponent * x[i]);
            z[i+1] = z[i] - atan(exponent);
        }
        else
        {
            x[i+1] = x[i] + (exponent * y[i]);
            y[i+1] = y[i] - (exponent * x[i]);
            z[i+1] = z[i] + atan(exponent);
        }
    }
    return x[n];
}
```

A treia parte:

- Crearea unei functii de calcul pentru eroarea de aproximare

```
double eroareAproximare(double CosCordic, double CosStandard) { // functie pentru a calcula eroarea de aproximare
    return CosCordic - CosStandard;
}
```

Ultima parte:

- Crearea unui “main” corespunzator
- Apelarea functiilor create: CordicCos() si eroareAproximare()

```
int main() {
    double cordicCos = CordicCos( theta: M_PI / 6);
    printf( _Format: "K = %.8f\n", calculareK( n: 8));
    printf( _Format: "cos(pi/6) cu metoda Cordic : %.8f\n", cordicCos);  // afisam rezultatul cu 8 zecimale
    printf( _Format: "cos(pi/6) cu functia cos din libraria C : %.8f\n", cos( _X: M_PI / 6));
    printf( _Format: "Eroarea de aproximare este %.8f", eroareAproximare(cordicCos,  CosStandard: cos( _X: M_PI / 6))); // eroarea de aproximare
}
```

Tabel valori:

K = 1.64674351

Iteratie (i)	z(i)	y(i)	x(i)	alfa(i)
0	0.52359878	0.00000000	0.60725448	0.78539816
1	-0.26179939	0.60725448	0.60725448	0.46364761
2	0.20184822	0.30362724	0.91088172	0.24497866
3	-0.04313044	0.53134767	0.83497491	0.12435499
4	0.08122455	0.42697581	0.90139337	0.06241881
5	0.01880574	0.48331289	0.87470738	0.03123983
6	-0.01243409	0.51064750	0.85960385	0.01562373
7	0.00318964	0.49721619	0.86758272	0.00781234
8	-0.00462270	0.50399418	0.86369822	0.00390623

b) **Eroarea de aproximare:**

$\varepsilon = cos_{CORDIC} - cos_{STANDARD} = -0.00035846$

```
cos(pi/6) cu metoda Cordic : 0.86566694
cos(pi/6) cu functia cos din libraria C : 0.86602540
Eroarea de aproximare este -0.00035846
```