METODY NUMERYCZNE

ZADANIA

Z1C. Uruchom i przeanalizuj poniższy program.

```c
#include <stdio.h>
int main(void) {
	float a = 3e-1, b = 4e-1;
	printf("%g\n", 1e1 * sqrt(a * a + b * b));
	return 0;
}
```

Z1P. Uruchom i przeanalizuj poniższy program.

```python
import math
a= 3e-1
b= 4e-1
r= math.sqrt(a*a + b*b)
print(r)
```

Z2C. Uruchom i przeanalizuj poniższy program.

```c
#include <stdio.h>
int main(void)
{
    float a = 10.0 / 4;
    float b = 10 / 4;
    printf("%g %g\n", a, b);
}
```

Z2P. Uruchom i przeanalizuj poniższy program.

```python
a= 10.0 / 4
b= 10   / 4
print(a, b)
```

Z3L. Uruchom i przeanalizuj poniższy program.

```c
#include <stdio.h>
#include <math.h>
#define A 3e-200
#define B 4e-200
#define C 1e200
int main(void) {
	double m;
	m = C * sqrt(A * A + B * B);
	printf("m= %g\n", m);
	return(0);
}
```

Z3R. Uruchom i przeanalizuj poniższy program.

```c
#include <stdio.h>
#include <math.h>
#define A 3e-200
#define B 4e-200
#define C 1e200
int main(void) {
	double m;
	m = C * A * sqrt(1 + (B / A) * (B / A));
	printf("m= %g\n", m);
	return(0);
}
```

Z4C. Uruchom i przeanalizuj poniższy program.

```c
#include <stdio.h>
int main(void) {
  float  x;
  printf("F");
  for (x = 1.0; x > 0.0; x /= 10)
    ;
  printf("inished\n");
}
```

Z4P. Uruchom i przeanalizuj poniższy program.

```python
print("F", end="")
x= 1.0
while x > 0.0:
      x= x / 10
print("inished")
```

Z5A. Uruchom i przeanalizuj poniższy program.

```c
#include<stdio.h>
double Power(double b, int k) {
       double res = 1.0;
       int i;
       for (i = 1; i <= k; i++)
              res *= b;
       return res;
}
double p(double x, int n, double a[]) {
       double result = 0.0;
       int k;
       for (k = 0; k <= n; k++)
              result += a[k] * Power(x, k);
       return result;
}
void main(void) {
       double a[] = { 0, 0, 0, 0, 0, 0, 0, 0, 1e300 };
       printf("%g\n", p(1e-50, 8, a));
       return;
}
```

Z5B. Uruchom i przeanalizuj poniższy program.

```c
#include<stdio.h>
double p(double x, int n, double a[]) {
       if (n == 0) return a[0];
       else return p(x, n - 1, a)* x + a[n];
}
void main(void) {
       double a[] = { 1e300, 0, 0, 0, 0, 0, 0, 0, 0 };
       printf("%g\n", p(1e-50, 8, a));
       return;
}
```

Z6. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
#include <math.h>
#define MaxErr 0.03
#define ABS(A) (A<0? -(A) : (A))
float SqR(float a) {
	float X, NewX, Err;
	X = a >= 1 ? a : 1;
	NewX = 0.5 * (X + a / X);
	Err = ABS(NewX - X) / NewX;
	while (Err > MaxErr) {
		X = NewX;
		NewX = 0.5 * (X + a / X);
		Err = ABS(NewX - X) / NewX;
	}
	return NewX;
}
int main(void) {
	printf("2:%g\n", SqR(2));
	return 0;
}
```

Z7. Uruchom i przeanalizuj poniższy program.

```
#include <stdio.h>
#define N 7
double e(double x) {
	double Sum = 0, T;
	double num = 1.0;
	double den = 1.0;
	int i;
	for (i = 1; i <= N; i++) {
		T = num / den;
		Sum += T;
		num *= x;
		den *= i;
	}
	return Sum;
}
int main(void) {
	printf("%g \n", e(1));
	return(0);
}
```

Z8. Pobierz i przetestuj program przedstawiający reprezentację IEEE-754 liczb zmiennoprzecinkowych pojedynczej precyzji. Wprowadź liczbę 0.15625 na wejściu oraz przeanalizuj uzyskane rezultaty w kontekście informacji zaprezentowanych na wykładzie (slajdy 82-83).