НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
РАДІОТЕХНІЧНИЙ ФАКУЛЬТЕТ

**Звіт з лабораторної роботи № 2.2**

«Обчислення визначеного інтегралу»

з дисципліни: «Інформатика 1. Основи програмування та алгоритми»

Виконала:

Студентка групи РЕ-21

Кравецька В.М

Прийняв:

Катін П.Ю.

Київ 2022

Мета

Повторення основних методів обчислення інтегралів. Вивчення їх застосування для написання коду, за допомогою якого можна вирішувати інтеграл заданої функції.

Алгоритми кожного методу

1. Метод лівих прямокутників

$$\int_a^b f(x)dx \approx h\cdot\left(f(x_0)+f(x_1)+\ldots+f(x_{n-1})\right)=h\cdot\sum_{k=0}^{n-1} f(x_k)$$

2. Метод правих прямокутників

$$\int_a^b f(x)dx \approx h \cdot \left(f(x_1) + f(x_2) + \ldots + f(x_n)\right) = h \cdot \sum_{k=1}^{n} f(x_k)$$

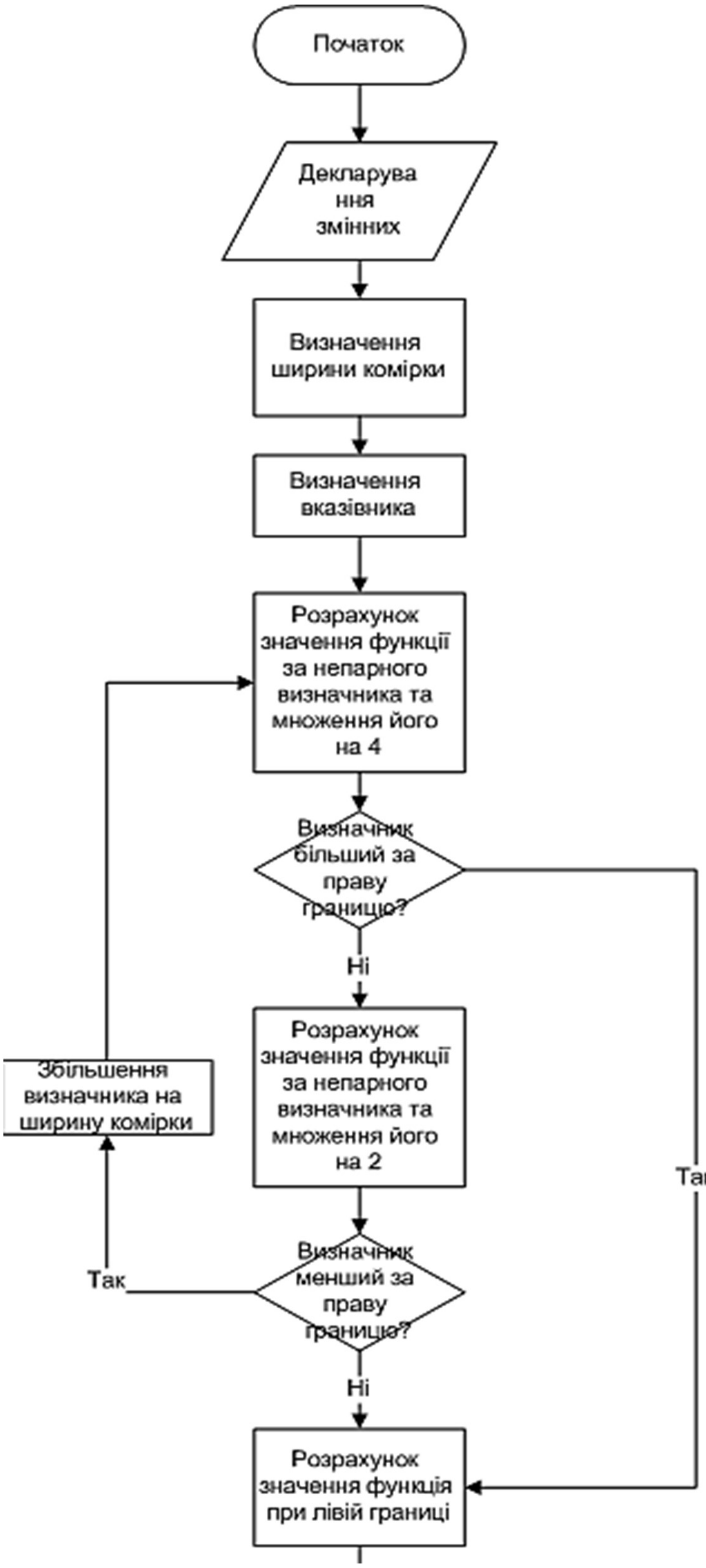
3. Метод трапецій

$$\int_a^b f(x)dx \approx h \cdot \left( \frac{f(x_0)+f(x_1)}{2} + \frac{f(x_1)+f(x_2)}{2} + \ldots + \frac{f(x_{n-1})+f(x_n)}{2} \right) =$$

$$= h \cdot \sum_{k=0}^{n-1} \frac{f(x_k)+f(x_{k+1})}{2} = h \cdot \left( \frac{f(x_0)}{2} + \sum_{k=1}^{n-1} f(x_k) + \frac{f(x_n)}{2} \right)$$

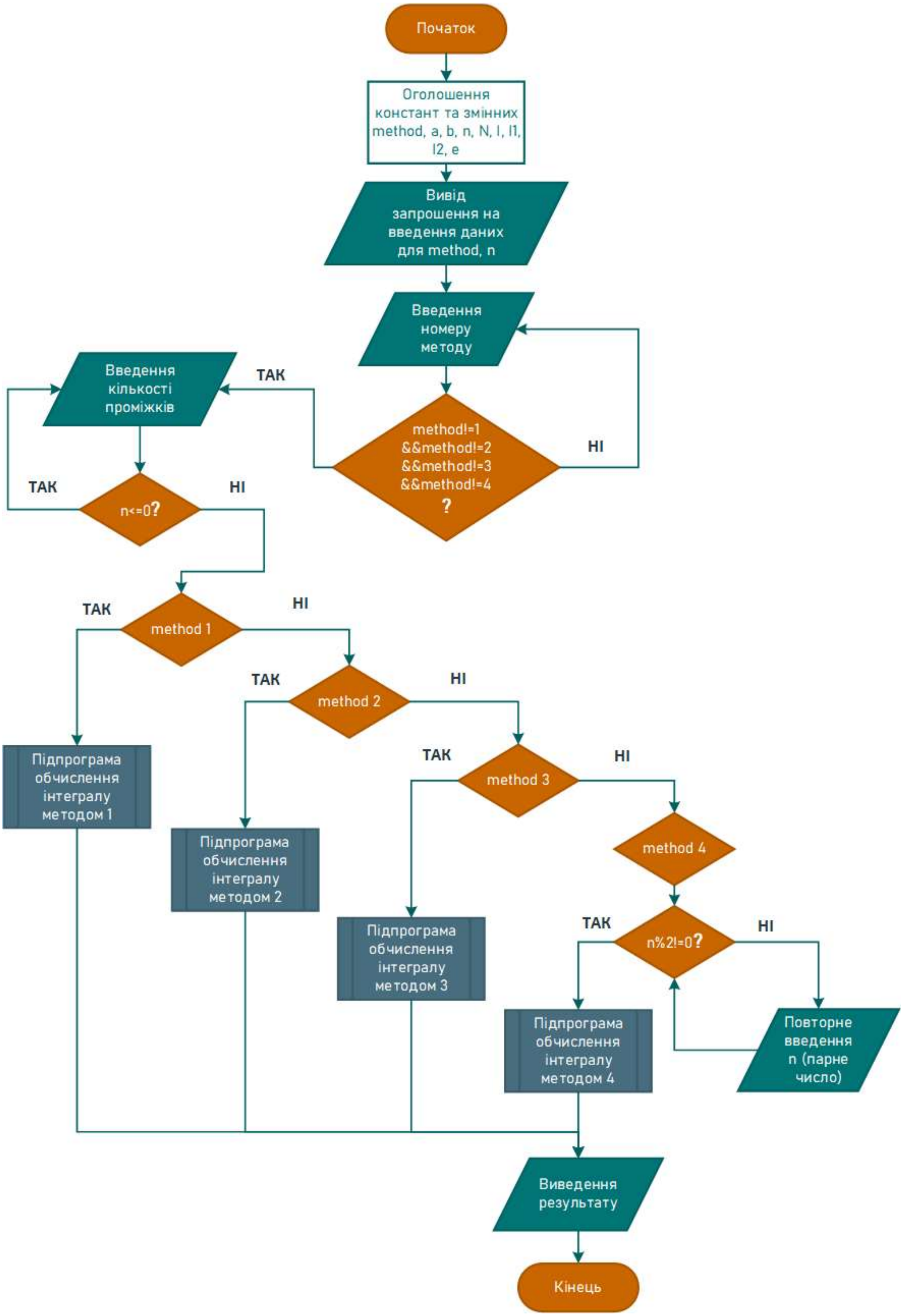
4. Метод парабол (Сімпсона)

$$\int_a^b f(x)dx \approx \frac{h}{3} \cdot \left( f(x_0) + 4 \sum_{k=1,3,5\ldots}^{n-1} f(x_k) + 2 \sum_{k=2,4,6,\ldots}^{n-2} f(x_k) + f(x_n) \right)$$




Розрахунок значення функція при правій границі
Сумування отриманих значень та множення їх на третину ширини комірки
Поверненя отриманого результату
Кінець

Основна блок-схема



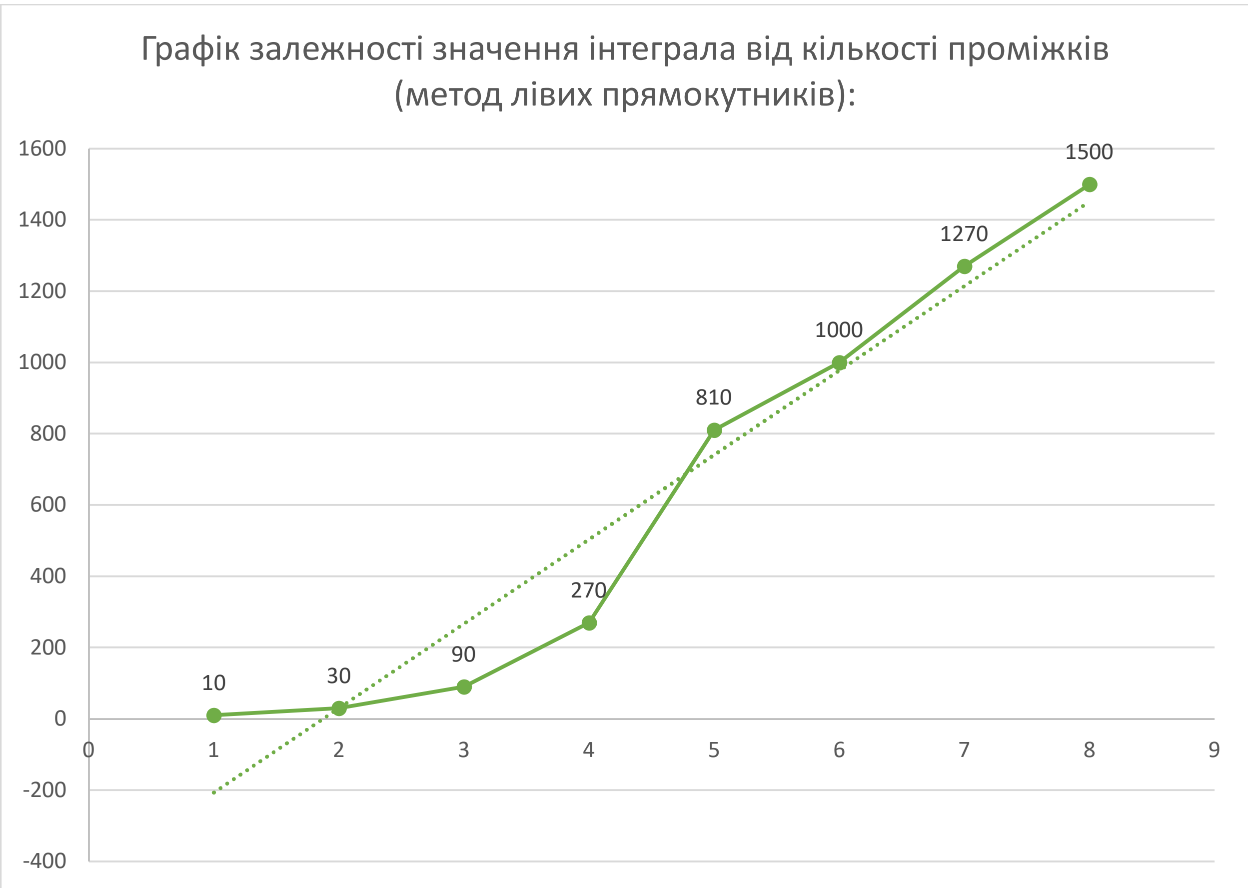
Функція (11 варіант): $\int_0^1 x^2 \cdot e^{2 \cdot x} dx$

Обрахований аналітично інтеграл в символьному виді: $\frac{e^2-1}{4}$

Чисельне значення аналітично обрахованого інтеграла: $\approx 1{,}59726$

Таблиця з результатами обрахунків заданого інтеграла всіма методами при 5 кількостях проміжків:

| Метод | n=10 | n=30 | n=90 | n=270 | n=810 |
|---|---|---|---|---|---|
| Лівих прямокутників | 1.25240 | 1.47685 | 1.55652 | 1.58361 | 1.59271 |
| Правих прямокутників | 1.99130 | 1.72315 | 1.63862 | 1.61098 | 1.60183 |
| Трапецій | 2.53671 | 1.86369 | 1.68153 | 1.62487 | 1.60641 |
| Парабол | 3.62939 | 2.56705 | 2.26941 | 2.17561 | 2.14492 |



Код

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

double f(double x);
void result (double a, double b, unsigned int n, double I);
double lrect(double a, double b, unsigned int n);
double rrect(double a, double b, unsigned int n);
double trapez(double a, double b, unsigned int n);
double parab(double a, double b, unsigned int n);

int main()
{
    unsigned int method;
    const double a=0;
    const double b=1;
    unsigned int n, N;
    double I, I1, I2;
    const double e=0.0001;

    printf(" The following methods of integration are available:");
    printf("\n\t1 - left rectangles");
    printf("\n\t2 - right rectangles");
    printf("\n\t3 - trapezoids");
    printf("\n\t4 - parabols");
    printf("\n\n Please choose the preferred one: ");
    scanf("%u", &method);

    while (method!=1&&method!=2&&method!=3&&method!=4)
    {
    printf("\n We only have 4 (four) options, please choose one of them (1, 2, 3, 4):\t");
```

```
scanf("%d", &method);
}

do{
        printf("\n Enter n (number of partition intervals):\t");
        scanf("%u", &n);
 }while(n<=0);

switch (method)
{
    case 1:
    {
        I=lrect(a, b, n);
        result(a, b, n, I);
        N=0;
        do {
            N = N+2;
            I1 = lrect(a, b, N);
            I2 = lrect(a, b, N+2);
           } while (fabs(I2-I1)>e);
        printf("\n\nN=%u, I1(N)=%.5lf\n", N, I1);
    }
    break;

    case 2:
    {
        I=rrect(a, b, n);
        result(a, b, n, I);
        N=0;
        do {
            N = N+2;
            I1 = rrect(a, b, N);
```

```
        I2 = rrect(a, b, N+2);
       } while (fabs(I2-I1)>e);
    printf("\n\nN=%u, I1(N)=%.5lf\n", N, I1);
}
break;

case 3:
{
    I=trapez(a, b, n);
    result(a, b, n, I);
    N=0;
    do {
        N = N+2;
        I1 = trapez(a, b, N);
        I2 = trapez(a, b, N+2);
       } while (fabs(I2-I1)>e);
    printf("\n\nN=%u, I1(N)=%.5lf\n", N, I1);
}
break;

case 4:
{
    if (n%2!=0)
    {
        printf("\n For the chosen method an even number of partition intervals is needed");
        printf("\n Please enter a different number: ");
        scanf("\t%d", &n);
    }

    I=parab(a, b, n);
    result(a, b, n, I);
    N=0;
```

```
            do {
                N = N+2;
                I1 = parab(a, b, N);
                I2 = parab(a, b, N+2);
               } while (fabs(I2-I1)>e);
            printf("\n\nN=%u, I1(N)=%.5lf\n", N, I1);
        }
        break;
    }
}

double f(double x)
{
    double y;
    y=pow(x,2)*exp(2*x);
    return y;
}

double lrect(double a, double b, unsigned int n)
{
    double h;
    unsigned int i;
    double x;
    double S=0;

    h = (b-a)/n;
    x = a;
    for (i=0; i<=n-1; i++)
    {
        S = S + f(x);
        x = x+h;
    }
```

```
    return S*h;
}

double rrect(double a, double b, unsigned int n)
{
    double h;
    unsigned int i;
    double x;
    double S=0;

    h = (b-a)/n;
    x = a+h;
    for (i=1; i<=n; i++)
    {
        S = S + f(x);
        x = x+h;
    }
    return S*h;
}

double trapez(double a, double b, unsigned int n)
{
    double h;
    unsigned int i;
    double x;
    double S=0;

    h = (b-a)/n;
    x = a+h;
    for (i=0; i<=n-1; i++)
    {
        S = S + (f(x)+f(x+h))/2;
```

```
        x = x+h;
    }
    return S*h;
}

double parab(double a, double b, unsigned int n)
{
    double h;
    double x;
    double I=0;

    h = (b-a)/n;
    x = a+h;
    while (x<b)
    {
        I += 4*f(x);
        x += h;
        if (x>=b)
        {
            I += 2*f(x);
            x += h;
        }
    }
    I = (h/3)*(I+f(a)+f(b));
    return I;
}

void result (double a, double b, unsigned int n, double I)
{
    system ("cls");
    printf("----------------------");
    printf("\n+~~~~~~~Result~~~~~~~+");
```

```
    printf("\n----------------------");
    printf("\na = %.1lf \nb = %.1lf \nn = %u \nIntegral = %.5lf", a, b, n, I);
}
```

Висновок

Для даної функції найкраще підійшов метод лівих прямокутників. Саме з ним результат швидше наближався до порахованого аналітичним шляхом при меншій кількості ітерацій. Відповідно, він виявився найточнішим.