***<u>Завдання</u>*: Розкласти функцію в ряд.**

Теоретична частина

Числовим рядом будемо називати вираз типу $u_1 + u_2 + \ldots + u_n + \ldots = \sum_{n=1}^{\infty} u_n$, де числа u_n – члени ряду і складають нескінченну числову послідовність.

Числовий ряд $\sum_{n=1}^{\infty} u_n$ є ***збіжним***, якщо є збіжною послідовність його часткових сум:
$\lim_{n\to\infty} S_n = \boldsymbol{S}$
В ***протилежному*** випадку числовий ряд є ***розбіжним***.

Функціональним рядом будемо називати вираз виду $\sum_{n=1}^{\infty} u_n(x) = u_1(x) + u_2(x) + \ldots + u_n(x) + \ldots$, де $u_n(x)$ – деякі функції відносно змінної x.

Сукупність всіх значень ***x***, при яких ряд є ***збіжним*** називається ***областю збіжності*** ряду. Відповідно ***решта x-ів*** складає ***область розбіжності***.

Сумою функціонального ряду будемо називати ***функцію S(x)***, визначену в області збіжності ряду як границю відповідних часткових сум.
S(x) = $\lim_{n\to\infty} S_n(x)$

Степеневий ряд

Функціональний ряд вигляду $\sum_{n=0}^{\infty} c_n x^n$ будемо називати ***степеневим*** рядом відносно змінної ***x***, де c_0, c_1, c_2, … = const.
Ряд вигляду $\sum_{n=0}^{\infty} c_n (x-x_0)^n$ також є степеневим, але дає розклад по степеням *(x - x_0).*

Розклад функції в степеневі ряди

Теорема:

Нехай функція ***f(x)*** має в деякому околі неперервні похідні ***n+1*** порядку. Нехай точка ***a*** знаходиться в цьому інтервалі. Тоді для будь-яких ***x-ів***, що належать інтервалу і околу точки ***a***, має місце ***формула Тейлора***:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_{n+1}(x)$$

де $R_{n+1}(x)$ – залишковий член.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \qquad (*)$$

Представлення функції f(x) у вигляді (*) називається розкладом функції f(x) у степеневий ряд функції f(x) у степеневий ***ряд Тейлора*** за степенями (x-a).
У випадку a = 0 :

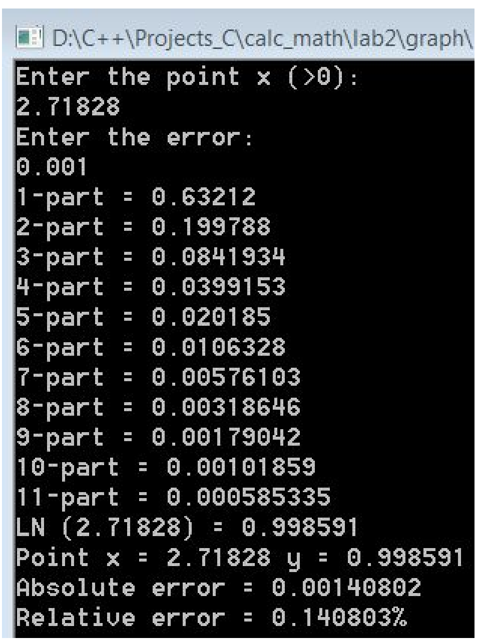
$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n \text{ - ряд Маклорена.}$$

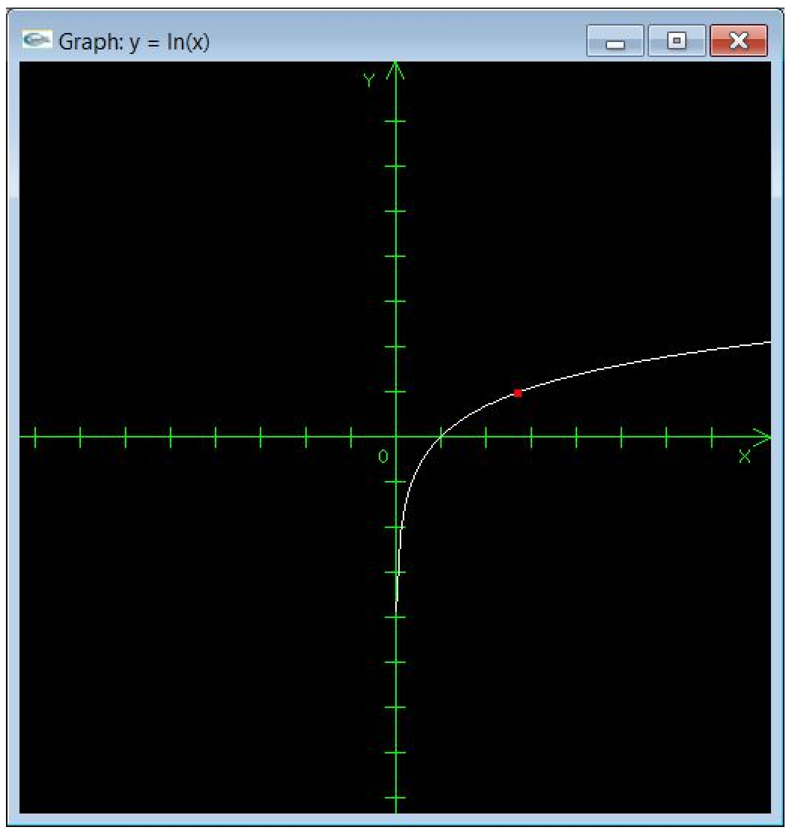
Якщо ми маємо ***знакопочерговий*** збіжний ряд і хочемо обчислити суму ряду з певною точністю ε, то при починаючи з певного ***n***, для якого виконується нерівність $\varepsilon \geq u_n(x)$, залишковий член можна не враховувати.

Приклад виконання програми:

Нехай маємо функцію Ln[x] Розкладемо її у степеневий ряд. Нехай шукаємо значення для Ln[x], при заданій точності e= *0,01*. Тоді за формулою розкладу у ряд Тейлора:

Тож сумою ряду буде: 0,998591.





Розраховані вручну дані співпадають із підрахунками програми.

На шостому елементі досягнута точність, що вимагалась. Аналогічно обчислюються значення для побудови графіку. Записуються значення Y для кожного X від початкового до кінцевого із заданною точністю.

Код програми
main.cpp

```
#include <GL/glut.h>
#include <cmath>
#include <iostream>
#include <cstring>
#include "draw.h"
#include "calc.h"
#include "const.h"
using namespace std;

float aValue = -1;
float eps = -1;

void Initialize (){
   glClearColor(0,0,0, 1.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-width/2, width/2, -height/2, height/2, -1, 1);
}

void display(){
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0, 1, 0);
   draw_net();      // Cartesian
   glColor3f(1,1,1);
   glBegin(GL_LINE_STRIP);
   for(float x = -width; x < width; x += 0.1) //func(x)
      glVertex2f(x*SCL, log(x)*SCL);

   glEnd();

   long double y = Taylor(aValue, eps);
   cout << "LN (" << aValue << ") = " << y << endl;

   glColor3f(1, 0, 0);
   glPointSize(5);
   glBegin(GL_POINTS); //draw Point
   glVertex2f(aValue * SCL, y * SCL);
   glEnd();
   cout << "Point x = " << aValue << " y = " << y <<endl;
   glutSwapBuffers();
}
```

```
void Keyboard(unsigned char key, int x, int y){
    switch (key){
    case 'e':
        exit(0);
        break;
    }
}

int main(int argc, char **argv){

    cout << "Enter the point x (>0):" << endl;
    while(aValue <= 0)
        cin >> aValue;

    cout << "Enter the error:" << endl;
    while(eps <= 0)
        cin >> eps;
    //Initialization
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(width, height);
    glutInitWindowPosition(x_position, y_position);
    glutCreateWindow("Graph: y = ln(x)");

    //Registration
    glutDisplayFunc(display);
    Initialize();
    glutKeyboardFunc(Keyboard);
    glutMainLoop();
    return 0;
}
```

Draw.cpp

```
#include "cstring"
#include "draw.h"
#include "const.h"
//window parameters
int width =  500;
int height = 500;
int x_position = 700;
int y_position = 0;
int pause  = 2;
int x = 0;
int y = 0;
```

```
//color marks
float r = 0;
float g = 0;
float b = 0;
int SCL = 30;
void renderBitmapString(float x, float y, void *font, char *str){
  char *c;
  int aLength = strlen(str);
  int i;
  for (i = 0; i < aLength; i++)
        x -= 4; //bias for words
  glRasterPos2f(x, y); //there
  for (c = str; *c != '\0'; c++) // draw string
        glutBitmapCharacter(font, *c);
}
void showSignature(){
    int  bias = 17;
    glColor3f(0, 1, 0);
    renderBitmapString(-bias/2        , -bias   , GLUT_BITMAP_8_BY_13, "0" );
    renderBitmapString(width/2 - bias , -bias  , GLUT_BITMAP_8_BY_13, "X" );
    renderBitmapString(-bias    , height/2-bias , GLUT_BITMAP_8_BY_13, "Y" );
}
void draw_net(void){
    showSignature(); //draw "aValue" , "Y", "0"
    int bias = 12; //for coordinate arrows
    glBegin(GL_LINES);
    glVertex2f(-width, 0); //axis "aValue"
    glVertex2f(width, 0);
    glVertex2f(0, -height); //axis "Y"
    glVertex2f(0, height);
    glEnd();
    int dist = 30;
    glBegin(GL_LINES); // draw lines on the axes
    for (int i = -(width/dist) * dist; i < width/2 * 0.9; i += dist ){ //axis "aValue"
        glVertex2f(i, dist/4);
        glVertex2f(i, -dist/4);}
    for (int i = -(height/dist) * dist; i < height/2 * 0.9; i += dist ){ //axis "Y"
        glVertex2f(dist/4, i);
        glVertex2f(-dist/4, i);}
    glEnd();
    glBegin(GL_LINE_STRIP); // arrow for "Y"
    glVertex2f(-bias/2, height/2 - bias);
    glVertex2f(0 , height/2);
    glVertex2f(bias/2, height/2 - bias);
    glEnd();
```

```
    glBegin(GL_LINE_STRIP); // arrow for "aValue"
    glVertex2f(width/2 - bias, bias/2);
    glVertex2f(width/2, 0);
    glVertex2f(width/2 - bias, -bias/2);
    glEnd();
}
```

calc.cpp

```
#include "iostream"
#include "cmath"
#include "calc.h"
using namespace std;

long double Taylor(float x, float eps){
    long double res = 0, part;
    int n = 1;
    while (1){
        part = (double)pow(x-1, n) / (n*pow(x, n));
        cout << n << "-part = " << part << endl;
        if (abs(part) > eps){
            res += part;
            n++;}
        else
            break; // Leibniz's theorem
    }
    return res;}
long double fact(int num){ //Factorial of the number
    if (num < 0)
         return -1;
    if (num == 0)
        return 1;

    int i = 2;
    long double res = 1;
    while (i <= num)
      res *= i++;

    return res;
}
```

Висновок

Для обчислення функції Ln[x] було використано розклад в ряд Тейлора. Цей метод дозволяє нам с заданою точністю розрахувати значення функції в точці *x,* просумувавши скінченну к-ть елементів степенного ряду.

Для ряду я отримав значення абсолютної похибки: 0,00140802, відносна похибка — 0.14%.

Відмінності в розрахунках Ln[x] з математичної бібліотеки розкладу в ряд зумовлені похибкою: численним методом (точність ε визначає скінченну кількість членів ряду, що зумовлює точність представлення суми ряду).