ЦЕЛЬ

1. Изучение методики измерения времени работы подпрограммы.

2. Изучение приемов повышения точности измерения времени работы подпрограммы.

3. Изучение способов измерения времени работы подпрограммы.

4. Измерение времени работы подпрограммы в прикладной программе.

ЗАДАНИЕ

1. Написать программу на языке C или C++, которая реализует выбранный алгоритм из задания.

2. Проверить правильность работы программы на нескольких тестовых наборах входных данных.

3. Выбрать значение параметра N таким, чтобы время работы программы было порядка 15 секунд

4. По приведенной методике определить время работы программы с относительной погрешностью не более 1%.

1. Вариант задания

Для выполнения мной было выбрано задание №6: Алгоритм вычисления функции ln(1+x) с помощью разложения в ряд по первым N членам этого ряда.

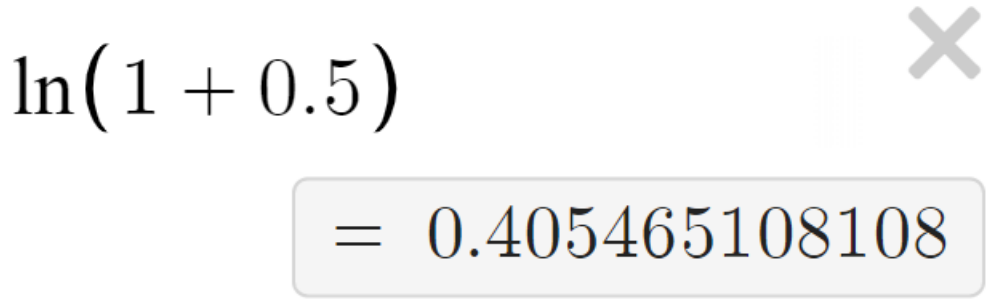
Реализация алгоритма:

```c
#include <stdio.h>

double Power(double number, double pow) {
    double result = 1;
    for (int i = 1; i <= pow; i++) {
        result *= number;
    }
    return result;
}

int main() {
    double n = 50000;
    double x = 0.5;
    double sum = 0;
    for (double i = 1; i <= n; i++) {
        sum += (Power(-1, i + 1) * Power(x, i) / i);
    }
    printf("%f\n", sum);
    return 0;
}
```

Правильность работы программы проверялась помощью калькулятора https://www.desmos.com/

1) ln(1+0.5)

Результат desmos:

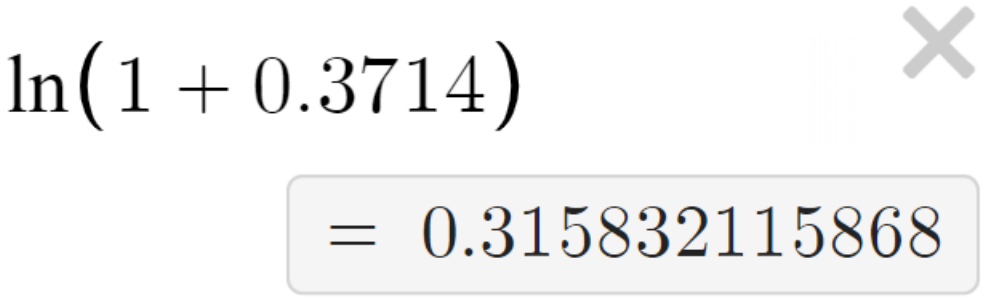


Результат работы программы:

```
x=0.500000
ln(1+x)=0.405465
```

2) ln(1+0.3714)

Результат desmos:



Результат работы программы:

```
x=0.371400
ln(1+x)=0.315832
```

3) ln(1+0)

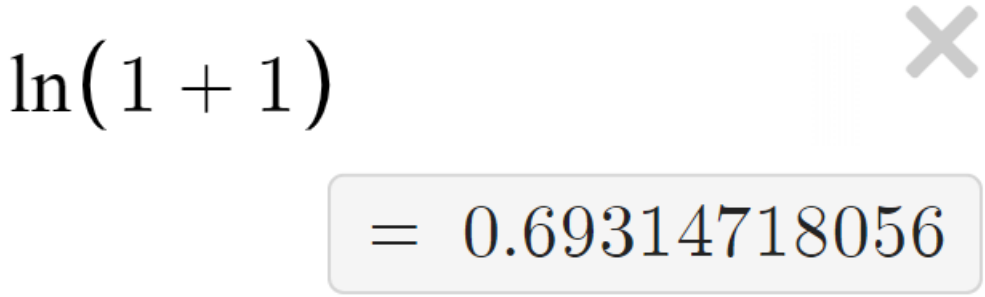
Результат desmos:

Результат работы программы:

```
x=0.000000
ln(1+x)=0.000000
```

4) ln(1+1)

Результат desmos:



Результат работы программы:

```
x=1.000000
ln(1+x)=0.693140
```

Перейдем к измерению времени работы программы.

3. Описание методики для определения времени работы программы

Для определения времени работы программы мной было выбрано использование функции clock_gettime. Для использования данной функции необходимо в коде обозначить начало и конец измерения времени, что может быть удобно в случае, если нам интересно время исполнения конкретного фрагмента кода, а не программы в целом, а затем после компиляции и исполнения программы узнать итоговое время. Точность данной функции не самая высокая, так же измеренный интервал включает в себя время работы других процессов, работающих на процессоре в измеряемый период. Однако в дальнейшем убедимся, что точность хоть и не высокая, но достаточная.

4. Результат измерения времени работы программы

```
evm24@comrade:~$ ./lab1_2.exe
0.405465
Time taken: 19.158051 sec.
evm24@comrade:~$ sync
evm24@comrade:~$ ./lab1_2.exe
0.405465
Time taken: 18.993224 sec.
evm24@comrade:~$ ./lab1_2.exe
0.405465
Time taken: 18.901864 sec.
```

```
evm24@comrade:~$ ./lab1_2.exe
0.405465
Time taken: 18.897255 sec.
evm24@comrade:~$ ./lab1_2.exe
0.405465
Time taken: 18.897386 sec.
evm24@comrade:~$ ./lab1_2.exe
0.405465
Time taken: 19.153284 sec.
```

Подсчет относительной погрешности:

1) Абсолютная погрешность таймера определяется его точностью:

0,000001 с.

2) Относительная погрешность:

0,000001 /18.897255 = $5{,}29*10^{-8}$

Значит, относительная погрешность точно меньше 1%, что удовлетворяет нашим требованиям.

Чтобы узнать приблизительное время работы программы, возьмем наименьший результат из полученных. Это 18.897255 сек.

5. Полный компилируемый листинг реализованной программы и команду для ее компиляции.

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

double Power(double number, double pow) {
    double result = 1;
    for (int i = 1; i <= pow; i++) {
        result *= number;
    }
    return result;
}

int main() {
    struct timespec start, end;
    double n = 150;
    double x = 0.5;
    double sum = 0;
    clock_gettime(CLOCK_MONOTONIC_RAW, &start);
    for (double i = 1; i <= n; i++) {
        sum += (Power(-1, i + 1) * Power(x, i) / i);
    }
    clock_gettime(CLOCK_MONOTONIC_RAW, &end);
    printf("Time taken: %lf sec.\n",
    end.tv_sec-start.tv_sec
    + 0.000000001*(end.tv_nsec-start.tv_nsec));
    printf("%f\n", sum);


    return 0;
}
```

Команда:

```
gcc lab1_clock_gettime.c -o lab1_2.exe -lrt
./lab1_2.exe
```

6. Вывод по результатам лабораторной работы

Мной были изучены методика и способы измерения работы программы. Я научилась самостоятельно измерять время работы программы разными способами и смогла сама убедиться в точности одного из этих способов.