Цель работы:
Изучить методы разработки консольных приложений, способы их запуска и обработки кодов возврата.

Ход работы:
Реализуем функцию вычисления степени экспоненты, путем разложения в ряд Тейлора. Используем данную формулу

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad x \in (-\infty, +\infty)$$

Чтобы написать e^x, необходимы функции факториала и возведения в степень, а также функция модуля, которая понадобится для проверки. Запишем их

```
/*Функция возведения числа в степень*/
double RaiseToPow(double x, int power)
{
	double res;
	int i;
	res = 1.0;
	if (power == 0) {
		return 1;
	}
	else if (power == 1) {
		return x;
	}

	else
		for (i = 1; i <= power; i++)
		{
			res = res * x;
		}
	return(res);
}
/* Функция нахождения факториала числа */
double fact(int k) {
	if (k < 2)
		return 1;
	return k * fact(k - 1);
}
/* Функция нахождения модуля числа */
double fabs(double x) {
	if (x > 0)
		return x;
	else return x * -1;
}
```

С использованием этих функций напишем функцию синуса, вычисляющую сумму первых n членов ряда Тейлора

```
/* Функция нахождения экспоненты разложением в ряд Тейлора */
double exponent(double x) {
	int n;
	double exp;
	exp = 0.0;
	for (n = 0; n <= 20; n++)
	{
		exp = exp + (RaiseToPow(x, (n)) / fact(n));
```

```
	}
	return(exp);
}
```

Для проверки этой функции составим набор параметров с помощью калькулятора, с учетом выбранной точности (0.0001).

$e^0=1$

$e^1=2.7183$

$e^{-2}=0.1353$

$e^5=148.4132$

$e^{0.5}=1.6487$

Далее напишем функцию проверяющую нашу функцию вычисления экспоненты на данных значениях, которая возвращает 0, если функция e^x возвращает значение в допустимой области согласно заданной точности, и 1, если результат не удовлетворяет условиям

```
/*Функция для проверки точности вычисления степени экспоненты*/
	int test_exp() {
		int r;
		r = 0;
		r = r || (fabs(exponent(0) - 1.0) >= 0.0001);
		r = r || (fabs(exponent(1) - 2.7183) >= 0.0001);
		r = r || (fabs(exponent(-2) - 0.1353) >= 0.0001);
		r = r || (fabs(exponent(5) - 148.4132) >= 0.0001);
		r = r || (fabs(exponent(0.5) - 1.6487) >= 0.0001);
		return r;
	}
```

При проверке пришлось увеличить количество членов в ряду Тейлора до 20. После выполнения программа выводит 0. При изменении значений или знака, программа выдает 1.

Вывод:

Я написал функцию вычисляющую синус, с помощью разложения в ряд Тейлора, тем самым изучил методы разработки консольных приложений, способы их запуска и обработки кодов возврата.