1. **Cosine power series expansion.** Evaluate the cosine function at an arbitrary point x using a truncated power series expansion

$$\cos x \approx \cos^{(N)} x = \sum_{k=0}^{N} \frac{(-1)^k}{(2k)!} x^{2k}. \tag{1}$$

For large x the series will converge too slowly. Still, you must somehow make the code applicable even in that case. Compare the values of your approximation to the correct values at $x = 0, \pi/5, \ldots, 100\pi$.

Implement the code as a separate `C` subroutine that takes as its arguments the value of x and the order of the expansion, n.

2. **Polynomials.** Evaluate an arbitrary polynomial

$$P_n(x) = \sum_{k=1}^{n} a_k x^{n-k} \tag{2}$$

and its derivative $P'(x)$ using the Horner's method. Your program should read the coefficients a_k from an external file and print it's value and derivative at points $x = 0, 1, \ldots, 200$. A sample file with the 250 first coefficients of the power series expansion of $\exp(x)$ is provided at `http://cc.oulu.fi/~tf/tiedostot/pub/atkIV/harjoitukset/Ex003/expc.txt`.

Again, the the polynomial and the derivative are to be computed in a separate subroutine.

3. **Polynomial interpolation.** For any given four points there exists a unique polynomial that passes through each of these points. Evaluate this polynomial for points (1,1), (2,3), (3,6), (4,8) at $x = 2.5$. Do this by implenting the Neville's algorithm by hand and then by using the Numerical Recipes library function `polint`.

4. **Chebyshev polynomial expansion.** A function can be approximated by the Chebyshev polynomial expansion formula

$$f(x) \approx -\frac{1}{2}c_0 + \sum_{k=0}^{N-1} c_k T_k(x), \tag{3}$$

where coefficients c_j are

$$c_j = \frac{2}{N} \sum_{k=1}^{N} f\left(\cos \frac{\pi(k-1/2)}{N}\right) \cos \frac{\pi j(k-1/2)}{N}. \tag{4}$$

Calculate the approximation for function $\tanh(x)$, where $-2 \leq x \leq 2$ for $n = 3, 5, 10$. Plot the approximations and compare them to the original function. Use Numerical Recipes routines `chebft` and `chebev`.

`void polint(float xa[], float ya[], int n, float x, float *y, float *dy)`

Description

Evaluates the polynomial interpolating given points at a point

Arguments

`float`	`xa[]`	*input*	Array of x–coordinates
`float`	`ya[]`	*input*	Array of y–coordinates
`int`	`n`	*input*	Number of points
`float`	`x`	*input*	Point where to evaluate the polynomial
`float`	`*y`	*output*	Pointer to the value of the polynomial
`float`	`*dy`	*output*	Pointer to an error estimate

`void chebft(float a, float b, float c[], int n, float (*func)(float))`

Description

Generates the Chebyshev expansion coefficients of function `func`.

Arguments

`float`	`a`	*input*	Lower limit of the fit interval
`float`	`b`	*input*	Upper limit of the fit interval
`float`	`c[]`	*output*	Array of expansion coefficients c_k
`int`	`n`	*input*	Maximum degree
`float, function`	`*func`	*input*	Pointer to the function to be fitted

`float chebev(float a, float b,float c[], int m, float x)`

Description

Evaluates an Chebyshev expansion

Arguments

`float`	`a`	*input*	Lower limit of the fit interval
`float`	`b`	*input*	Upper limit of the fit interval
`float`	`c[]`	*input*	Array of expansion coefficients c_k, computed by `chebft`.
`int`	`m`	*input*	Number of coefficients
`float`	`x`	*input*	Point where expansion is evaluated