Notes. 9/5/97

General problem: Given a function y = f(x) which is analytic on the interval [a,b], use Taylor's theorem with remainder to find a polynomial y = p(x) that approximates f to within a specified error on the interval. Then write a procedure to implement the polynomial function you have found.

For example, suppose the function is exp(x)*cos(2*x) and the interval is [-1,1]. Suppose the error that will be tolerated is 10^(-3)

```
> restart;
```

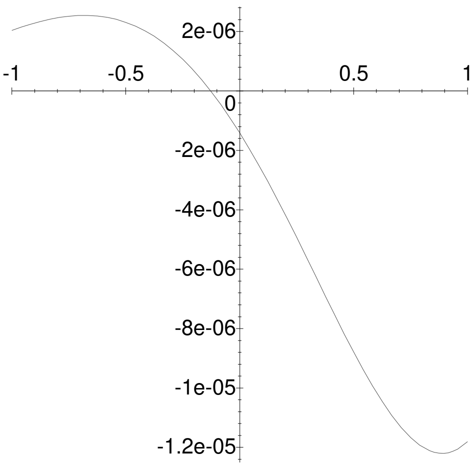
Define the function.

```
> f := x -> exp(x)*cos(2*x);
```

$$f := x \rightarrow \mathbf{e}^x \cos(2\,x)$$

By taylor's theorem, the error in using the nth taylor polynomial is no more than $\frac{M}{(n+1)!}$, where M is an upper bound on the n+1st derivative and 1 is the radius of the interval. We can bound the error in using the nth taylor polynomial by plotting the function D@@(n+1)/(n+1)! over the interval [-1..1] for various n until we find one with a small enough error.

```
> plot({0,(D@@13)(f)/13!},-1..1);
```



By experimenting, the 11th taylor polynomial about 0 is almost good enough, but is off in the 3rd decimal place near -1. The 12th taylor polynomial is close enough everywhere.

```
> p := taylor(f(x),x=0,13);
```

$$p := 1 + x - \frac{3}{2}x^2 - \frac{11}{6}x^3 - \frac{7}{24}x^4 + \frac{41}{120}x^5 + \frac{13}{80}x^6 + \frac{29}{5040}x^7 - \frac{527}{40320}x^8 - \frac{1199}{362880}x^9 + \frac{79}{1209600}x^{10} + \frac{6469}{39916800}x^{11} + \frac{1679}{68428800}x^{12} + \mathrm{O}(x^{13})$$

```
> p :=convert(evalf(p),polynom);
```

$$p := 1. + 1.\,x - 1.500000000\,x^2 - 1.833333333\,x^3 - .2916666667\,x^4 + .3416666667\,x^5 + .1625000000\,x^6 + .005753968254\,x^7 - .01307043651\,x^8 - .003304122575\,x^9 + .00006531084656\,x^{10} + .0001620620891\,x^{11} + .00002453645249\,x^{12}$$

We should convert this to the nested for to make evaluation more efficient.

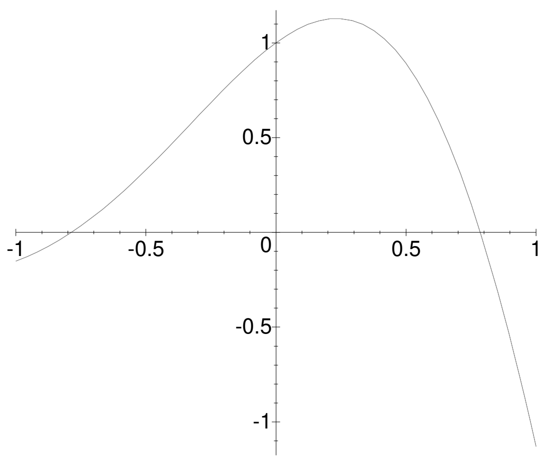
```
> p := convert(p,horner);
```

$$p := 1. + (1. + (-1.500000000 + (-1.833333333 + (-.2916666667 + (.3416666667 + (.1625000000 + (.005753968254 + (-.01307043651 + (-.003304122575 + (.00006531084656 + (.0001620620891 + .00002453645249\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x$$

```
> p := unapply(p,x);
```

$$p := x \to 1. + (1. + (-1.500000000 + (-1.833333333 + (-.2916666667 + (.3416666667 + (.1625000000 + (.005753968254 + (-.01307043651 + (-.003304122575 + (.00006531084656 + (.0001620620891 + .00002453645249\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x)\,x$$

```
> plot({f,p},-1..1);
```




```
> for i from -10 to 10 do
    printf(`%3.2e %15.3e  %15.3e \n`,
  evalf(i/10.),evalf(f(i/10.)),evalf(p(i/10.))) od;
-1.00e+00      -1.530e-01       -1.530e-01
-9.00e-01      -9.237e-02       -9.237e-02
-8.00e-01      -1.312e-02       -1.312e-02
-7.00e-01       8.440e-02        8.440e-02
-6.00e-01       1.988e-01        1.988e-01
-5.00e-01       3.277e-01        3.277e-01
-4.00e-01       4.670e-01        4.670e-01
-3.00e-01       6.114e-01        6.114e-01
-2.00e-01       7.541e-01        7.541e-01
-1.00e-01       8.868e-01        8.868e-01
```

```
0.00e-01       1.000e+00        1.000e+00
1.00e-01       1.083e+00        1.083e+00
2.00e-01       1.124e+00        1.124e+00
3.00e-01       1.114e+00        1.114e+00
4.00e-01       1.039e+00        1.039e+00
5.00e-01       8.908e-01        8.908e-01
6.00e-01       6.602e-01        6.602e-01
7.00e-01       3.422e-01        3.422e-01
8.00e-01      -6.498e-02       -6.498e-02
9.00e-01      -5.588e-01       -5.588e-01
1.00e+00      -1.131e+00       -1.131e+00
```

Exporting to C

Here's how to take the function developed in Maple and translate it to C

Read in the translator.

```
> readlib(C);
```

proc() ... **end**

Define a function procedure. Note: I colored and copied the nested polynomial down into the definition below. Never retype anthing you can handle this way.

```
> f := proc(x)
  1.+(1.+(-1.500000000+(-1.833333333+(-.2916666667+(.3416666667+(.16
  25000000+(.5753968254e-2+(-.1307043651e-1+(-.3304122575e-2+(.65310
  84656e-4+(.1620620891e-3+.2453645249e-4*x)*x)*x)*x)*x)*x)*x)*x)*x)
  *x)*x)*x end;
```

$f :=$ **proc**(x)

$$1. + (1. + (-1.500000000 + (-1.833333333 + (-.2916666667 + (.3416666667 + (.1625000000 + (.005753968254 + (-.01307043651 + (-.003304122575 + (.00006531084656 + (.0001620620891 + .00002453645249*x)*x)*x)*x)*x)*x)*x)*x)*x)*x)*x)*x$$

end

Translate it.

```
> C(f,ansi);
double f(double x)
{
  return(1.0+(1.0+(-0.15E1+(-0.1833333333E1+(-0.2916666667+(0.3416666667+(
0.1625+(0.5753968254E-2+(-0.1307043651E-1+(-0.3304122575E-2+(0.6531084656E-4+(
0.1620620891E-3+0.2453645249E-4*x)*x)*x)*x)*x)*x)*x)*x)*x)*x)*x)*x);
}
```

Define a main procedure to test it.

```
> main := proc()
  local i;
  for i from 0 to 20 do
   printf(`%6.2e  %15.3e  %15.3e\n`,
  -1.+.1*i,f(-1.+.1*i),exp(-1.+.1*i)*cos(2*(-1.+.1*i))) od;
  NULL;
  end;
>
```

main := **proc**()

local *i*;

for *i* **from** 0 **to** 20 **do** printf(*'%6.2e %15.3e %15.3e\n'*, $-1. + .1*i$, f($-1. + .1*i$), exp($-1. + .1*i$)*cos($-2. + .2*i$))

od;

NULL

end

```
> main();
-1.00e+00       -1.530e-01       -1.530e-01
-9.00e-01       -9.237e-02       -9.237e-02
-8.00e-01       -1.312e-02       -1.312e-02
-7.00e-01        8.440e-02        8.440e-02
-6.00e-01        1.988e-01        1.988e-01
-5.00e-01        3.277e-01        3.277e-01
-4.00e-01        4.670e-01        4.670e-01
-3.00e-01        6.114e-01        6.114e-01
-2.00e-01        7.541e-01        7.541e-01
-1.00e-01        8.868e-01        8.868e-01
0.00e-01        1.000e+00        1.000e+00
1.00e-01        1.083e+00        1.083e+00
2.00e-01        1.124e+00        1.124e+00
3.00e-01        1.114e+00        1.114e+00
4.00e-01        1.039e+00        1.039e+00
5.00e-01        8.908e-01        8.908e-01
6.00e-01        6.602e-01        6.602e-01
7.00e-01        3.422e-01        3.422e-01
8.00e-01       -6.498e-02       -6.498e-02
9.00e-01       -5.588e-01       -5.588e-01
1.00e+00       -1.131e+00       -1.131e+00
```

```
> C(main,ansi);
double main(void)
{
  int i;

  for(i = 0;(i <= 20);i++)
    printf( "%6.2e  %15.3e  %15.3e
" ,-1.0+0.1*i,f(-1.0+0.1*i),exp(-1.0+0.1*i)*
cos(-2.0+0.2*i));
  return(NULL);
}
```

Now color and paste these into your cpp text editor, add some include directives, compile and execute. See below.

```
//  polynomial approximating exp(x)*cos(2*x) on [-1,1]

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

```
double f(double x)
{
  return(1.0+(1.0+(-0.15E1+(-0.1833333333E1+(-0.2916666667+(0.3416666667+(
0.1625+(0.5753968254E-2+(-0.1307043651E-1+(-0.3304122575E-2+(0.6531084656E-4+(
0.1620620891E-3+0.2453645249E-4*x)*x)*x)*x)*x)*x)*x)*x)*x)*x)*x)*x);
}

void main(void)
{
  int i;
  for(i = 0;(i <= 20);i++)
    printf( "%6.2e  %15.3e  %15.3e" ,-1.0+0.1*i,f(-1.0+0.1*i),exp(-1.0+0.1*i)*
cos(-2.0+0.2*i));
  return(NULL);
}
```

This is the output obtained from a compile and execute in Microsoft Visual C++

```
 -1.00e+000    -1.531e-001    -1.531e-001
-9.00e-001    -9.237e-002    -9.237e-002
-8.00e-001    -1.312e-002    -1.312e-002
-7.00e-001     8.440e-002     8.440e-002
-6.00e-001     1.989e-001     1.989e-001
-5.00e-001     3.277e-001     3.277e-001
-4.00e-001     4.670e-001     4.670e-001
-3.00e-001     6.114e-001     6.114e-001
-2.00e-001     7.541e-001     7.541e-001
-1.00e-001     8.868e-001     8.868e-001
0.00e+000     1.000e+000     1.000e+000
1.00e-001     1.083e+000     1.083e+000
2.00e-001     1.125e+000     1.125e+000
3.00e-001     1.114e+000     1.114e+000
4.00e-001     1.039e+000     1.039e+000
5.00e-001     8.908e-001     8.908e-001
6.00e-001     6.603e-001     6.603e-001
7.00e-001     3.423e-001     3.423e-001
8.00e-001    -6.498e-002    -6.498e-002
9.00e-001    -5.588e-001    -5.588e-001
1.00e+000    -1.131e+000    -1.131e+000
Press any key to continue
```