**נספח**

**טורי טיילור**

התוכנית הבאה מחשבת את e0.5 ע"י נוסחת מקלורן, ומשווה לערך המתקבל מפונקצית הספרייה. כפי שאנחנו רואים, הם שווים.

/\* exp.c - compute e(x) \*/

#include <stdio.h>

#include <math.h<

long double my\_exp(long double x, long double eps)

{

 long double n, nf, xp, result, term;

 result = 1;

 xp = x;

 nf = 1.0;

 n = 1.0;

 do {

 term = xp/nf;

 result += term;

 xp = xp\*x;

 n = n + 1.0;

 nf = nf \* n;

 { while (term > eps);

 return result;

 } /\* my\_exp \*/

int main()

}

 long double sqr\_root\_e, lib\_value;

 sqr\_root\_e = my\_exp(0.5, 1.0/100000000000000000);

 lib\_value = expl(0.5);

 printf("Our value = %22.19Lf, Lib value = %22.19Lf\n", sqr\_root\_e,

lib\_value);

} /\* main \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\> exp.exe

Our value = 1.6487212707001281469, Lib value = 1.6487212707001281469

C:\>

בתוכנית הבאה מחשבת את e3.7 ע"י נוסחת טיילור (2), ומשווה לערך המתקבל מנפונקצית הספריה. כפי שאנחנו רואים, הם שווים.

/\* exp2.c - compute e(x) \*/

#include <stdio.h>

#include <math.h>

long double compute\_e()

}

 long double n, nf, result, term, eps;

 result = 1;

 nf = 1.0;

 n = 1.0;

 eps = 0.000000000000000001;

 do }

 term = 1.0/nf;

 result += term;

 n = n + 1.0;

 nf = nf \* n;

 { while (term > eps);

 return result;

/\* compute\_e()\*/

long double my\_exp(long double x, long double eps)

}

 long double n, nf, xp, result, term, ek, e;

 unsigned int k;

 e = compute\_e();

 printf("e = %22.18Lf\n", e);

 k = x;

 x = x - k;

 ek = 1.0;

 for (; k > 0; k--)

 ek = ek \* e;

 printf("ek = %Lf, x = %Lf\n", ek, x);

 result = 1.0;

 xp = x;

 nf = 1.0;

 n = 1.0;

 do{

 term = xp/nf;

 result += term;

 xp = xp\*x;

 n = n + 1.0;

 nf = nf \* n;

 { while (term > eps);

 result = ek\* result;

 return result;

/\* my\_exp()\*/

int main()

{

 long double sqr\_root\_e, lib\_value, e;

 printf("lib e = %22.18Lf\n", expl(1));

 sqr\_root\_e = my\_exp(3.7, 0.000000000000000001);

 lib\_value = expl(3.7);

 printf("Our value = %25.19Lf, Lib value = %25.19Lf\n", sqr\_root\_e,

lib\_value);

} /\* main\*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\> exp2.exe

lib e = 2.718281828459045235

e = 2.718281828459045235

ek = 20.085537, x = 0.700000

Our value = 40.4473043600673977140, Lib value = 40.4473043600673977140

C:\>

התוכנית הבאה מחשבת sinx בעבור x כלשהוא (גם שלילי) תוך הקטנת x בכדי שהטור יתכנס יותר מהר. אנחנו גם נשווה עם ערך של פונקצית ספריה.

/\* sin2.c - compute sin(x) in radians, using the taylor series \*/

#include <stdio.h>

#include <math.h>

/\* compute sin(x) within epsilon \*/

long double my\_abs(long double x(

{

 if ( x >= 0(

 return x;

 else

 return -x;

}/\* my\_abs \*/

long double pi;

long double my\_sin(long double x, long double eps)

{

 long double sinx, xx, nf, R, xp, n, sign, flag;

 unsigned long int k;

 flag = 1.0;

 if (x < 0)

 }

 flag = -1.0;

 x = 0 - x;

 { /\* if \*/

 if ( x> 2\*pi(

}

 k = x/(2\*pi);

 x = x - k\*2\*pi;

 if (x > pi(

 {

 x = x - pi;

 flag = -1.0 \*flag;

 } /\* if \*/

 if (x > pi/2.0(

 x = pi - x;

 } /\* if \*/

 xx = x\*x;

 nf = 1.0;

 sinx = 0.0;

 xp = x;

 n = 1.0;

 sign = 1.0;

 do {

 R = my\_abs(xp / nf);

 sinx = sinx + sign \* R;

 sign = - sign;

 xp = xp \* xx;

 nf = nf \* ((n+1.0)\*(n+2.0));

 n = n + 2.0;

 { while( R >= eps);

 return flag \* sinx;

} /\* my\_sin \*/

 void main()

}

 long double x, eps, y;

 pi = 3.141592653589793238;

 x=-(7\*180-53)\*pi/180.0;

 eps = 0.0000000000001;

 y = my\_sin(x, eps);

 printf("\nsin(%22.18Lf), within %22.18Lf, = %22.18Lf\n",

 x, eps, y);

 printf("\nlib value sin(%22.18Lf) = %22.18lf\n", x, sin(x;((

}/\* main \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C:\> sin2.exe

sin(-21.066124071571557173), within 0.000000000000100000, =

-0.798635510047292826

lib value sin(-21.066124071571557173) = -0.798635510047292940

C:\>

התוכנית הבאה מחשבת 0x sintdt ∫בעבור x חיובי כלשהוא:

/\* sinx\_x1.c \*/

#include <stdio.h>

#include <math.h>

/\* Compute sinx\_x(x) within epsilon \*/

float my\_abs(float x)

{

 if ( x >= 0)

 return x;

 else

 return -x;

} /\* my\_abs \*/

float pi;

float my\_Isinx\_x(float x, float eps)

{

 float sinx\_xx, xx, nf, R, xp, n, sign;

 xx = x\*x;

 nf = 1.0;

 sinx\_xx = 0.0;

 xp = x;

 n = 1.0;

 sign = 1.0;

 do

 {

 R = xp / ((2\*n+1)\*nf) ;

 sinx\_xx = sinx\_xx + sign \* R;

 sign = - sign;

 xp = xp \* xx;

 nf = nf \* ((2\*n)\*(2\*n+1.0));

 n = n + 1.0;

 R = my\_abs(R);

 } while( R >= eps );

 return sinx\_xx;

} /\* my\_sinx\_x \*/

 int main()

{

 return sinx\_xx;

} /\* my\_sinx\_x \*/

 int main()

{

 float x, eps, y;

 pi = 3.141592653589793238;

 x=30\*pi/180.0;

 printf("x = %f, deg(x)=%f\n", x, x\*180/pi);

 eps = 0.000001;

 y = my\_Isinx\_x(x, eps);

 printf("\nIsinx\_x(%12.8f), within %12.8f, = %12.8f\n",

 x, eps, y);

 return 0;

} /\* main \*/

פלט ריצה:

C:\>sinx\_x1.exe

x = 0.523599, deg(x)=30.000000

Isinx\_x( 0.52359879), within 0.00000100, = 0.16979462

C:\>