Michael Tagg

ENCM369 - B04

Laboratory 1

21/01/2020

**Exercise G**

**exG Source File:**

**// exG.c**

**// ENCM 369 Winter 2020 Lab 1 Exercise G**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**#define MAX\_ABS\_F (5.0e-10)**

**#define POLY\_DEGREE 4**

**double polyval(const double \*a, int n, double x);**

**// Return a[0] + a[1] \* x + ... + a[n] \* pow(x, n).**

**int main(void)**

**{**

**double f[ ] = {12.0, 8.0, -7.5, -2.0, 1.0};**

**double dfdx[POLY\_DEGREE];**

**double guess;**

**int max\_updates;**

**int update\_count;**

**int scan\_count;**

**int i;**

**int close\_enough = 0;**

**double current\_x, current\_f, current\_dfdx;**

**printf("This program demonstrates use of Newton's Method to find\n"**

**"approximate roots of the polynomial\nf(x) = ");**

**printf("%.2f", f[0]);**

**i = 1;**

**start\_loop0:**

**if (!(i <= POLY\_DEGREE)) goto end\_loop0;**

**if (!(f[i] >= 0)) goto else\_line0;**

**printf(" + %.2f\*pow(x,%d)", f[i], i);**

**i++;**

**goto start\_loop0;**

**else\_line0:**

**printf(" - %.2f\*pow(x,%d)", -f[i], i);**

**i++;**

**goto start\_loop0;**

**end\_loop0:**

**printf("\nPlease enter a guess at a root, and a maximum number of\n"**

**"updates to do, separated by a space.\n");**

**scan\_count = scanf("%lf%d", &guess, &max\_updates);**

**if (!(scan\_count != 2)) goto end0;**

**printf("Sorry, I couldn't understand the input.\n");**

**exit(1);**

**end0:**

**if (!(max\_updates < 0)) goto end1;**

**printf("Sorry, a negative limit on updates does not make sense.\n");**

**exit(1);**

**end1:**

**printf("Running with initial guess %f.\n", guess);**

**i = POLY\_DEGREE-1;**

**start\_loop1:**

**if (!(i >= 0)) goto end\_loop1;**

**dfdx[i] = (i + 1) \* f[i + 1]; // Calculus!**

**i--;**

**goto start\_loop1;**

**end\_loop1:**

**current\_x = guess;**

**update\_count = 0;**

**start\_loop2:**

**current\_f = polyval(f, POLY\_DEGREE, current\_x);**

**printf("%d update(s) done; x is %.15f; f(x) is %.15e\n",**

**update\_count, current\_x, current\_f);**

**close\_enough = fabs(current\_f) < MAX\_ABS\_F;**

**if (close\_enough) goto end\_loop2;**

**if (update\_count == max\_updates) goto end\_loop2;**

**current\_dfdx = polyval(dfdx, POLY\_DEGREE - 1, current\_x);**

**current\_x -= current\_f / current\_dfdx;**

**update\_count++;**

**goto start\_loop2;**

**end\_loop2:**

**if (!(close\_enough)) goto else\_line1;**

**printf("Stopped with approximate solution of %.12f.\n",**

**current\_x);**

**goto end2;**

**else\_line1:**

**printf("%d updates performed, |f(x)| still >= %g.\n",**

**update\_count, MAX\_ABS\_F);**

**end2:**

**;**

**return 0;**

**}**

**double polyval(const double \*a, int n, double x)**

**{**

**double result = a[n];**

**int i = n - 1;**

**start\_loop3:**

**if (!(i >= 0)) goto end\_loop3;**

**result = x \* result + a[i];**

**i--;**

**goto start\_loop3;**

**end\_loop3:**

**;**

**return result;**

**}**

**exG Test Runs:**

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

6.0 x

Sorry, I couldn't understand the input.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

6.0 -1

Sorry, a negative limit on updates does not make sense.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

6.0 7

Running with initial guess 6.000000.

0 update(s) done; x is 6.000000000000000; f(x) is 6.540000000000000e+02

1 update(s) done; x is 4.844522968197880; f(x) is 1.981524083898128e+02

2 update(s) done; x is 4.049713244711731; f(x) is 5.753016233189771e+01

3 update(s) done; x is 3.547342312231244; f(x) is 1.507245027828954e+01

4 update(s) done; x is 3.286762260541826; f(x) is 2.961161605424200e+00

5 update(s) done; x is 3.204294159629291; f(x) is 2.494859925491522e-01

6 update(s) done; x is 3.195958776264439; f(x) is 2.417101680794786e-03

7 update(s) done; x is 3.195876426758148; f(x) is 2.346939780295543e-07

7 updates performed, |f(x)| still >= 5e-10.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

6.0 8

Running with initial guess 6.000000.

0 update(s) done; x is 6.000000000000000; f(x) is 6.540000000000000e+02

1 update(s) done; x is 4.844522968197880; f(x) is 1.981524083898128e+02

2 update(s) done; x is 4.049713244711731; f(x) is 5.753016233189771e+01

3 update(s) done; x is 3.547342312231244; f(x) is 1.507245027828954e+01

4 update(s) done; x is 3.286762260541826; f(x) is 2.961161605424200e+00

5 update(s) done; x is 3.204294159629291; f(x) is 2.494859925491522e-01

6 update(s) done; x is 3.195958776264439; f(x) is 2.417101680794786e-03

7 update(s) done; x is 3.195876426758148; f(x) is 2.346939780295543e-07

8 update(s) done; x is 3.195876418760682; f(x) is 3.552713678800501e-15

Stopped with approximate solution of 3.195876418761.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

1.9 3

Running with initial guess 1.900000.

0 update(s) done; x is 1.900000000000000; f(x) is -5.609000000000002e-01

1 update(s) done; x is 1.861905732138006; f(x) is 3.697768316357042e-03

2 update(s) done; x is 1.862153736035794; f(x) is 1.310122179631890e-07

3 update(s) done; x is 1.862153744823214; f(x) is 0.000000000000000e+00

Stopped with approximate solution of 1.862153744823.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

-2.0 4

Running with initial guess -2.000000.

0 update(s) done; x is -2.000000000000000; f(x) is -2.000000000000000e+00

1 update(s) done; x is -2.111111111111111; f(x) is 3.657216887669499e-01

2 update(s) done; x is -2.096310072355151; f(x) is 6.956161761438651e-03

3 update(s) done; x is -2.096017446438796; f(x) is 2.692369063694855e-06

4 update(s) done; x is -2.096017333090758; f(x) is 3.979039320256561e-13

Stopped with approximate solution of -2.096017333091.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

-1.0 3

Running with initial guess -1.000000.

0 update(s) done; x is -1.000000000000000; f(x) is -5.000000000000000e-01

1 update(s) done; x is -0.961538461538462; f(x) is 6.317618430726313e-03

2 update(s) done; x is -0.962012765968632; f(x) is 8.592169002952232e-07

3 update(s) done; x is -0.962012830493154; f(x) is 1.776356839400250e-14

Stopped with approximate solution of -0.962012830493.

mmmta@LAPTOP-35G9NI35 /cygdrive/e/encm369/lab01/exg

$ ./a

This program demonstrates use of Newton's Method to find

approximate roots of the polynomial

f(x) = 12.00 + 8.00\*pow(x,1) - 7.50\*pow(x,2) - 2.00\*pow(x,3) + 1.00\*pow(x,4)

Please enter a guess at a root, and a maximum number of

updates to do, separated by a space.

-1.0 2

Running with initial guess -1.000000.

0 update(s) done; x is -1.000000000000000; f(x) is -5.000000000000000e-01

1 update(s) done; x is -0.961538461538462; f(x) is 6.317618430726313e-03

2 update(s) done; x is -0.962012765968632; f(x) is 8.592169002952232e-07

2 updates performed, |f(x)| still >= 5e-10.

Exercise H

**exG Source File:**

// exH.c

// ENCM 369 Winter 2020 Lab 1 Exercise H

#include <stdio.h>

void print\_array(const char \*str, const int \*a, int n);

// Prints the string given by str on stdout, then

// prints a[0], a[1], ..., a[n - 1] on stdout on a single line.

void sort\_descending(int \*a, int n);

// Sorts a[0], a[1], ..., a[n - 1] from largest to smallest.

int main(void)

{

int test\_array[] = {440, 220, 330, 550, 330, 660, 110, 330, 440};

print\_array("before sorting ...", test\_array, 9);

sort\_descending(test\_array, 9);

print\_array("after sorting ...", test\_array, 9);

return 0;

}

void print\_array(const char \*str, const int \*a, int n)

{

int i = 0;

puts(str);

start\_loop0:

if (!(i < n)) goto end\_loop1;

printf(" %d", a[i]);

i++;

goto start\_loop0;

end\_loop1:

printf("\n");

}

void sort\_descending(int \*a, int n)

{

// This is an implementation of an algorithm called insertion sort.

int j = 1, k, v;

start\_outter\_loop:

if (!(j < n)) goto end\_outter\_loop;

v = a[j];

k = j;

start\_inner\_loop:

if (!(k >= 1)) goto end\_inner\_loop;

if (!(a[k - 1] < v)) goto end\_inner\_loop;

a[k] = a[k - 1];

k--;

goto start\_inner\_loop;

end\_inner\_loop:

a[k] = v;

j++;

goto start\_outter\_loop;

end\_outter\_loop:

;

}