```cpp
// NewtonRaphsonRoots.cpp : Defines the entry point for the application.
// Copyright (c) Monday, January 20, 2025 by James Pate Williams, Jr.
// Finds the nth root of a real number using an iterative method

#include "framework.h"
#include "NewtonRaphsonRoots.h"
#include <float.h>
#include <math.h>
#include <cwchar>
#include <chrono>

#define MAX_LOADSTRING 100

#define IDC_EDIT1 201
#define IDC_EDIT2 202
#define IDC_EDIT3 203
#define IDC_EDIT4 204
#define IDC_EDIT5 205
#define IDC_EDIT6 206

#define IDC_STATIC1 301
#define IDC_STATIC2 302
#define IDC_STATIC3 303
#define IDC_STATIC4 304
#define IDC_STATIC5 305

#define IDC_BUTTON_COMPUTE 401
#define IDC_BUTTON_CLEAR 402

// Global Variables:
HINSTANCE hInst;                                // current instance
WCHAR szTitle[MAX_LOADSTRING];                  // The title bar text
WCHAR szWindowClass[MAX_LOADSTRING];            // the main window class name
WCHAR buffer[16384];

// Forward declarations of functions included in this code module:
ATOM                MyRegisterClass(HINSTANCE hInstance);
BOOL                InitInstance(HINSTANCE, int);
LRESULT CALLBACK    WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK    About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
                      _In_opt_ HINSTANCE hPrevInstance,
                      _In_ LPWSTR    lpCmdLine,
                      _In_ int       nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Place code here.

    // Initialize global strings
```

```
53      LoadStringW(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
54      LoadStringW(hInstance, IDC_NEWTONRAPHSONROOTS, szWindowClass,
          MAX_LOADSTRING);
55      MyRegisterClass(hInstance);
56
57      // Perform application initialization:
58      if (!InitInstance (hInstance, nCmdShow))
59      {
60          return FALSE;
61      }
62
63      HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE
          (IDC_NEWTONRAPHSONROOTS));
64
65      MSG msg;
66
67      // Main message loop:
68      while (GetMessage(&msg, nullptr, 0, 0))
69      {
70          if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
71          {
72              TranslateMessage(&msg);
73              DispatchMessage(&msg);
74          }
75      }
76
77      return (int) msg.wParam;
78  }
79  //
80  //  FUNCTION: MyRegisterClass()
81  //
82  //  PURPOSE: Registers the window class.
83  //
84  ATOM MyRegisterClass(HINSTANCE hInstance)
85  {
86      WNDCLASSEXW wcex;
87
88      wcex.cbSize = sizeof(WNDCLASSEX);
89
90      wcex.style          = CS_HREDRAW | CS_VREDRAW;
91      wcex.lpfnWndProc    = WndProc;
92      wcex.cbClsExtra     = 0;
93      wcex.cbWndExtra     = 0;
94      wcex.hInstance      = hInstance;
95      wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE
          (IDI_NEWTONRAPHSONROOTS));
96      wcex.hCursor        = LoadCursor(nullptr, IDC_ARROW);
97      wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
98      wcex.lpszMenuName   = MAKEINTRESOURCEW(IDC_NEWTONRAPHSONROOTS);
99      wcex.lpszClassName  = szWindowClass;
100     wcex.hIconSm        = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));
101
```

```
102     return RegisterClassExW(&wcex);
103 }
104 //
105 //   FUNCTION: InitInstance(HINSTANCE, int)
106 //
107 //   PURPOSE: Saves instance handle and creates main window
108 //
109 //   COMMENTS:
110 //
111 //        In this function, we save the instance handle in a global variable and
112 //        create and display the main program window.
113 //
114 BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
115 {
116    hInst = hInstance; // Store instance handle in our global variable
117
118    HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
119       CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);
120
121    if (!hWnd)
122    {
123       return FALSE;
124    }
125
126    ShowWindow(hWnd, nCmdShow);
127    UpdateWindow(hWnd);
128
129    return TRUE;
130 }
131
132 // nth root function
133
134 double fx(double x, int n)
135 {
136     return pow(x, n) - n;
137 }
138
139 // nth root derivative
140
141 double fd(double x, int n)
142 {
143     return n * pow(x, n - 1);
144 }
145
146 void iterativeFunction(
147     double x0, double& delta, double& x1, int n)
148 {
149     delta = fx(x0, n) / fd(x0, n);
150     x1 = x0 - delta;
151 }
152
153 //
```

```
154 //  FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
155 //
156 //  PURPOSE: Processes messages for the main window.
157 //
158 //  WM_COMMAND  - process the application menu
159 //  WM_PAINT    - Paint the main window
160 //  WM_DESTROY  - post a quit message and return
161 //
162 //
163 LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
164 {
165     static HWND hwndEdit1 = 0, hwndEdit2 = 0, hwndEdit3 = 0;
166     static HWND hwndEdit4 = 0, hwndEdit5 = 0, hwndEdit6 = 0;
167
168     switch (message)
169     {
170     case WM_CREATE:
171     {
172         CreateWindowEx(0, L"STATIC", L"n:", WS_CHILD | WS_VISIBLE,
173             10, 10, 80, 20, hWnd, (HMENU)IDC_STATIC1, GetModuleHandle(NULL),
                NULL);
174         CreateWindowEx(0, L"STATIC", L"x0:", WS_CHILD | WS_VISIBLE,
175             10, 40, 80, 20, hWnd, (HMENU)IDC_STATIC2, GetModuleHandle(NULL),
                NULL);
176         CreateWindowEx(0, L"STATIC", L"Tolerance:", WS_CHILD | WS_VISIBLE,
177             10, 70, 80, 20, hWnd, (HMENU)IDC_STATIC3, GetModuleHandle(NULL),
                NULL);
178         CreateWindowEx(0, L"STATIC", L"Max Its:", WS_CHILD | WS_VISIBLE,
179             10, 100, 80, 20, hWnd, (HMENU)IDC_STATIC4, GetModuleHandle(NULL),
                NULL);
180
181         hwndEdit1 = CreateWindowEx(0, L"EDIT", NULL, WS_CHILD | WS_VISIBLE |
              WS_BORDER,
182             100, 10, 200, 20, hWnd, (HMENU)IDC_EDIT1, GetModuleHandle(NULL),
                NULL);
183         hwndEdit2 = CreateWindowEx(0, L"EDIT", NULL, WS_CHILD | WS_VISIBLE |
              WS_BORDER,
184             100, 40, 200, 20, hWnd, (HMENU)IDC_EDIT2, GetModuleHandle(NULL),
                NULL);
185         hwndEdit3 = CreateWindowEx(0, L"EDIT", NULL, WS_CHILD | WS_VISIBLE |
              WS_BORDER,
186             100, 70, 200, 20, hWnd, (HMENU)IDC_EDIT3, GetModuleHandle(NULL),
                NULL);
187         hwndEdit4 = CreateWindowEx(0, L"EDIT", NULL, WS_CHILD | WS_VISIBLE |
              WS_BORDER,
188             100, 100, 200, 20, hWnd, (HMENU)IDC_EDIT4, GetModuleHandle(NULL),
                NULL);
189         hwndEdit5 = CreateWindowEx(0, L"EDIT", NULL, WS_CHILD | WS_VISIBLE |
              WS_BORDER
190             | ES_MULTILINE, 100, 190, 300, 300, hWnd, (HMENU)IDC_EDIT6,
191             GetModuleHandle(NULL), NULL);
192
```

```
193         CreateWindowEx(0, L"BUTTON", L"Compute", WS_CHILD | WS_VISIBLE |
              BS_PUSHBUTTON,
194             10, 130, 80, 30, hWnd, (HMENU)IDC_BUTTON_COMPUTE, GetModuleHandle
                (NULL), NULL);
195         CreateWindowEx(0, L"BUTTON", L"Clear", WS_CHILD | WS_VISIBLE |
              BS_PUSHBUTTON,
196             220, 130, 80, 30, hWnd, (HMENU)IDC_BUTTON_CLEAR, GetModuleHandle
                (NULL), NULL);
197     }
198     break;
199     case WM_COMMAND:
200         {
201             int wmId = LOWORD(wParam);
202             // Parse the menu selections:
203             switch (wmId)
204             {
205             case IDC_BUTTON_COMPUTE:
206             {
207                 WCHAR* end, line[128] = { };
208                 double tolFound = 0.0, tolerance = 0.0, x0 = 0.0;
209                 double delta = DBL_MAX, x1 = 0.0;
210                 int its = 0, maximumIts = 0, n = 2;
211
212                 n = GetDlgItemInt(hWnd, IDC_EDIT1, FALSE, FALSE);
213
214                 if (n < 2)
215                 {
216                     MessageBox(hWnd, L"n must be >= 2", L"Warning",
217                         MB_OK | MB_ICONWARNING);
218                     return 0;
219                 }
220
221                 maximumIts = GetDlgItemInt(hWnd, IDC_EDIT4, FALSE, FALSE);
222
223                 if (maximumIts < 1 || maximumIts > 1000)
224                 {
225                     MessageBox(hWnd, L"Maximum Its must be >= 2 and <= 1000",
226                         L"Warning", MB_OK | MB_ICONWARNING);
227                     return 0;
228                 }
229
230                 GetWindowText(hwndEdit2, line, 127);
231                 x0 = wcstod(line, &end);
232
233                 GetWindowText(hwndEdit3, line, 127);
234                 tolerance = wcstod(line, &end);
235
236                 auto start = std::chrono::high_resolution_clock::now();
237
238                 while (
239                     fabs(delta) > tolerance &&
240                     its < maximumIts)
```

```
241                 {
242                     iterativeFunction(x0, delta, x1, n);
243                     delta = x1 - x0;
244                     tolFound = fabs(delta);
245                     its++;
246                     x0 = x1;
247                 }
248 
249                 auto endPt = std::chrono::high_resolution_clock::now();
250                 auto duration = std::chrono::duration_cast<
251                     std::chrono::microseconds>(endPt - start);
252 
253                 long long runtime = duration.count();
254                 swprintf_s(line, L"x0 = %13.10lf\r\n", x0);
255                 wcscat_s(buffer, 16383, line);
256                 swprintf_s(line, L"x1 = %13.10lf\r\n", x1);
257                 wcscat_s(buffer, 16383, line);
258                 swprintf_s(line, L"Tolerance = %13.10lf\r\n", tolerance);
259                 wcscat_s(buffer, 16383, line);
260                 swprintf_s(line, L"Tol Found = %13.10lf\r\n", tolFound);
261                 wcscat_s(buffer, 16383, line);
262                 swprintf_s(line, L"n = %ld\r\n", n);
263                 wcscat_s(buffer, 16383, line);
264                 swprintf_s(line, L"Iterations = %ld\r\n", its);
265                 wcscat_s(buffer, 16383, line);
266                 swprintf_s(line, L"Runtime = %lld Microseconds\r\n",
267                     runtime);
268                 wcscat_s(buffer, 16383, line);
269                 SetDlgItemText(hWnd, IDC_EDIT6, buffer);
270                 return 0;
271             }
272             break;
273             case IDC_BUTTON_CLEAR:
274             {
275                 wcscpy_s(buffer, 16383, L"\0");
276                 SetDlgItemText(hWnd, IDC_EDIT6, buffer);
277                 return 0;
278             }
279             break;
280             case IDM_ABOUT:
281                 DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
282                 break;
283             case IDM_EXIT:
284                 DestroyWindow(hWnd);
285                 break;
286             default:
287                 return DefWindowProc(hWnd, message, wParam, lParam);
288             }
289         }
290         break;
291     case WM_PAINT:
292         {
```

```
293             PAINTSTRUCT ps;
294             HDC hdc = BeginPaint(hWnd, &ps);
295             // TODO: Add any drawing code that uses hdc here...
296             EndPaint(hWnd, &ps);
297         }
298         break;
299     case WM_DESTROY:
300         PostQuitMessage(0);
301         break;
302     default:
303         return DefWindowProc(hWnd, message, wParam, lParam);
304     }
305     return 0;
306 }
307
308 // Message handler for about box.
309 INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
310 {
311     UNREFERENCED_PARAMETER(lParam);
312     switch (message)
313     {
314     case WM_INITDIALOG:
315         return (INT_PTR)TRUE;
316
317     case WM_COMMAND:
318         if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
319         {
320             EndDialog(hDlg, LOWORD(wParam));
321             return (INT_PTR)TRUE;
322         }
323         break;
324     }
325     return (INT_PTR)FALSE;
326 }
```