```
/*
my methods:
integerToText (preserves Integer f, produces Text& x)
doHandleLButtonDown (produces Boolean& s)
doArrangeSet (arrayN& a, Boolean& s)
doLoadData(HWND hwnd, arrayN& a)
doPaintData(HDC hdc, arrayN& a, Boolean& s)
doPaintRectangles (preserves HDC hdc)
scheduleWmPaint (HWND hWnd)
*/
#include "stdafx.h"
#include "Lab7.h"
#include "ArrayContainer.hpp"
#include "WinRect.h"
#include "ComFileDialog\ComFileDialog.hpp"

#define MAX_LOADSTRING 100

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

arrayN arrayOfC; // array of containers (sets, and value)
Boolean state = false; // boolean for left click

// 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 _tWinMain(_In_ HINSTANCE hInstance,
                       _In_opt_ HINSTANCE hPrevInstance,
                       _In_ LPTSTR    lpCmdLine,
                       _In_ int       nCmdShow)
{
        UNREFERENCED_PARAMETER(hPrevInstance);
        UNREFERENCED_PARAMETER(lpCmdLine);

        // TODO: Place code here.
        MSG msg;
        HACCEL hAccelTable;

        // Initialize global strings
        LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
        LoadString(hInstance, IDC_LAB7, szWindowClass, MAX_LOADSTRING);
        MyRegisterClass(hInstance);

        // Perform application initialization:
        if (!InitInstance (hInstance, nCmdShow))
        {
                return FALSE;
        }

        hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_LAB7));

        // Main message loop:
```

```
	while (GetMessage(&msg, NULL, 0, 0))
	{
		if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
		{
			TranslateMessage(&msg);
			DispatchMessage(&msg);
		}
	}

	return (int) msg.wParam;
}



//
//  FUNCTION: MyRegisterClass()
//
//  PURPOSE: Registers the window class.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
	WNDCLASSEX wcex;

	wcex.cbSize = sizeof(WNDCLASSEX);

	wcex.style			= CS_HREDRAW | CS_VREDRAW;
	wcex.lpfnWndProc	= WndProc;
	wcex.cbClsExtra		= 0;
	wcex.cbWndExtra		= 0;
```

```
        wcex.hInstance          = hInstance;
        wcex.hIcon                      = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_LAB7));
        wcex.hCursor            = LoadCursor(NULL, IDC_ARROW);
        wcex.hbrBackground      = (HBRUSH)(COLOR_WINDOW+1);
        wcex.lpszMenuName       = MAKEINTRESOURCE(IDC_LAB7);
        wcex.lpszClassName      = szWindowClass;
        wcex.hIconSm            = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

        return RegisterClassEx(&wcex);
}

//
//   FUNCTION: InitInstance(HINSTANCE, int)
//
//   PURPOSE: Saves instance handle and creates main window
//
//   COMMENTS:
//
//        In this function, we save the instance handle in a global variable and
//        create and display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
  HWND hWnd;

  hInst = hInstance; // Store instance handle in our global variable

  hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);
```

```
  if (!hWnd)
  {
    return FALSE;
  }

  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);

  return TRUE;
}

//////////////                    ////////////////////////////////////////////////
//////////////  MY METHODS  ////////////////////////////////////////////////
//////////////                    ////////////////////////////////////////////////
//////////////                    ////////////////////////////////////////////////

void integerToText (preserves Integer f, produces Text& x) {
  const int bufferSize = 25;
  char cString[bufferSize];

  sprintf_s (cString, bufferSize, "%d", (int)f);
  x = cString;
}

void doHandleRButtonDown (produces Boolean& s){

        if(s == false){
                s = true;
```

```
        }else{
                s = false;
        }
}

void doArrangeSet (arrayN& a, Boolean& s){
        if(s){
                Integer x,y,j,t1, t2, t3;
                for (int i = 0; i < 81; i++){
                        if(a[i].value == 0){
                                x = i % 9;
                                y = 9*(i / 9);
                                j = x;
                                while(j < 81){
                                        if(a[i].set.contains(a[j].value)){
                                                a[i].set.remove(a[j].value, t1);
                                        }
                                        j = j + 9;
                                }
                                j = y;
                                t1 = j + 9;
                                while(j < t1){
                                        if(a[i].set.contains(a[j].value)){
                                                a[i].set.remove(a[j].value, t2);
                                        }
                                        j++;
                                }
                                x = x % 3;
                                y = y / 9;
```

```
                                        y = y % 3;
                                        j = (i - x);
                                        j = j - (y*9);
                                        for (int co = 0; co < 9; co++){
                                                if(co % 3 == 0 && co != 0){
                                                        j = j + 6;
                                                }
                                                if(a[i].set.contains(a[j].value)){
                                                        a[i].set.remove(a[j].value, t3);
                                                }
                                                j++;
                                        }
                                }
                        }
                }
}

void doLoadData(HWND hwnd, arrayN& a){

        Boolean successful;
        Integer k;
        Text fileToRead, fullPath;
        File input;
        CommonFileDialog fileDialog;

        fileDialog.getOpenFilename (hwnd, fileToRead, fullPath, successful);
        if (!successful) {
                MessageBox (NULL, "File not found", "Error: Open File", MB_OK);
        }else{
```

```
                input.openFile (fileToRead, File::openForRead, successful);
                for (int i = 0; i < 81; i++) {
                        input.read(a[i].value, successful);
                        a[i].set.clear();
                        if(a[i].value == 0){
                                for(int j = 1; j < 10; j++){
                                        k = j;
                                        a[i].set.add(k);
                                }
                        }
                }
                input.closeFile (successful);
        }
}

void doPaintData(HDC hdc, arrayN& a, Boolean& s){

        Integer x, y, x2, y2, count, t, size;
        Text k;
        x = 70;
        y = 68;
        for (int i = 0; i < 81; i++){
                count++;
                size = a[i].set.size();
                if(a[i].value != 0){
                        integerToText(a[i].value, k);
                        TextOut (hdc, x, y, k, k.size());
                }else if(s){
                        numContainer temp;
```

```
                    TextOut (hdc, x2 = x-16, y2 = y-16, "{", 1);
                    x2 = x2 + 1;
                    for (int j = 0; j < size; j++){
                        if(j % 3 == 0 && j != 0){
                            x2 = x-15;
                            y2 = y2 + 15;
                        }
                        a[i].set.removeAny(t);
                        if(t != 0){
                            integerToText(t, k);
                            TextOut (hdc, x2 = x2 + 4, y2, k, 1);
                            temp.add(t);
                            if (j < (size-1)) {
                                TextOut (hdc, x2 = x2 + 7, y2, ",", 1);
                            }
                        }
                    } // end for
                    TextOut (hdc, x2 + 8, y2, "}", 1);
                    a[i].set.transferFrom(temp);
                }
                x = x + 50;
                if(count % 9 == 0){
                    x = 70;
                    y = y + 50;
                }

            }
        }
```

```
void doPaintRectangles (preserves HDC hdc){

        RECT r;
        Integer l = 50,t = 50,ri = 100,b = 100,count;
        // l,t = top left point rect. ri,b = bottom left point rect.
        for (int i = 0; i < 81; i++){
                count++;
                if(count % 9 == 1 || i == 0){
                        SetRect (&r, l, t, ri, b);
                        Rectangle (hdc, r.left, r.top, r.right, r.bottom);
                }else{
                        SetRect (&r, (l = l + 50), t, (ri = ri + 50), b);
                        Rectangle (hdc, r.left, r.top, r.right, r.bottom);
                }
                if(count % 9 == 0){
                        l = l - 400;
                        ri = ri - 400;
                        t = t + 50;
                        b = b + 50;
                }
        }
        MoveToEx (hdc, 198, 50, NULL);
        LineTo (hdc, 198, 500);
        MoveToEx (hdc, 201, 50, NULL);
        LineTo (hdc, 201, 500);
        MoveToEx (hdc, 348, 50, NULL);
        LineTo (hdc, 348, 500);
        MoveToEx (hdc, 351, 50, NULL);
        LineTo (hdc, 351, 500);
```

```
        MoveToEx (hdc, 50, 198, NULL);
        LineTo (hdc, 500, 198);
        MoveToEx (hdc, 50, 201, NULL);
        LineTo (hdc, 500, 201);
        MoveToEx (hdc, 50, 348, NULL);
        LineTo (hdc, 500, 348);
        MoveToEx (hdc, 50, 351, NULL);
        LineTo (hdc, 500, 351);
} // doPaintRectangle

void scheduleWmPaint (HWND hWnd){
        RECT clientRect;
        GetClientRect(hWnd, &clientRect);
        InvalidateRect(hWnd, &clientRect, true);
}

//////////////                    ////////////////////////////////////////////////////
//////////////    END OF    ////////////////////////////////////////////////////
//////////////     MY METHODS   ////////////////////////////////////////////////////
//////////////                    ////////////////////////////////////////////////////

//
//  FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
//
//  PURPOSE:  Processes messages for the main window.
//
//  WM_COMMAND      - process the application menu
//  WM_PAINT   - Paint the main window
//  WM_DESTROY          - post a quit message and return
```

```
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
	int wmId, wmEvent;
	PAINTSTRUCT ps;
	HDC hdc;

	switch (message)
	{
	case WM_COMMAND:
		wmId    = LOWORD(wParam);
		wmEvent = HIWORD(wParam);
		// Parse the menu selections:
		switch (wmId)
		{
		case ID_FILE_OPEN:
			doLoadData(hWnd, arrayOfC);
			scheduleWmPaint(hWnd);
			break;
		case IDM_ABOUT:
			DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
			break;
		case IDM_EXIT:
			DestroyWindow(hWnd);
			break;
		default:
			return DefWindowProc(hWnd, message, wParam, lParam);
		}
```

```
                break;
        case WM_PAINT:
                hdc = BeginPaint(hWnd, &ps);
                doPaintRectangles(hdc);
                doArrangeSet(arrayOfC, state);
                doPaintData(hdc, arrayOfC, state);
                EndPaint(hWnd, &ps);
                break;
        case WM_RBUTTONDOWN:
      doHandleRButtonDown(state);
                 scheduleWmPaint(hWnd);
    break;
        case WM_DESTROY:
                PostQuitMessage(0);
                break;
        default:
                return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}

// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
        UNREFERENCED_PARAMETER(lParam);
        switch (message)
        {
        case WM_INITDIALOG:
                return (INT_PTR)TRUE;
```

```
        case WM_COMMAND:
                if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
                {
                        EndDialog(hDlg, LOWORD(wParam));
                        return (INT_PTR)TRUE;
                }
                break;
        }
        return (INT_PTR)FALSE;
}
```