```
#include <windows.h>
#include <string.h>
#include <stdlib.h>

#include "TextLine.h"

BOOLEAN inputStage = TRUE;

HFONT hFont;

PWSTR Append(PWSTR, PWSTR);
void SelectionSort(PDWORD, DWORD);
PWSTR Calculate();
void DrawTextLine(HWND);
void LineToInt(PDWORD, const WORD);

const WORD WIDTH = 1500;
const WORD HEIGHT = 300;
```

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch (msg){
        case WM_PAINT:
            DrawTextLine(hWnd);
            break;

        case WM_KEYDOWN:
            if (inputStage)
            {
                switch (wParam)
                {
                case VK_RETURN:
                    inputStage = FALSE;
                    InvalidateRect(hWnd, NULL, NULL);
                    break;
                case VK_BACK:
                    TextLine.PopNum();
                    InvalidateRect(hWnd, NULL, TRUE);
                    break;
                default:
                    TextLine.PushNum((WCHAR)wParam);
                    InvalidateRect(hWnd, NULL, TRUE);
                }
            }
            break;

        case WM_DESTROY:
            PostQuitMessage(NULL);
            return 0;
    }

    return DefWindowProc(hWnd, msg, wParam, lParam);
}
```

```
55  int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR pCmdLine, int nCmdShow)
56  {
57      PCWSTR wndclName = L"first";
58      hFont = CreateFont(70, 30, 0, 0, 0, 0, 0, 0, DEFAULT_CHARSET, 0, 0, 0, 0, L"Arial Bold");
59
60      WNDCLASSEX wndcl;
61      wndcl.cbSize = sizeof(WNDCLASSEX);
62      wndcl.style = CS_HREDRAW | CS_VREDRAW;
63      wndcl.lpfnWndProc = WndProc;
64      wndcl.cbClsExtra = NULL;
65      wndcl.cbWndExtra = NULL;
66      wndcl.hInstance = hInstance;
67      wndcl.hIcon = LoadIcon(NULL, IDI_WINLOGO);
68      wndcl.hIconSm = NULL;
69      wndcl.hCursor = LoadCursor(NULL, IDC_ARROW);
70      wndcl.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
71      wndcl.lpszMenuName = NULL;
72      wndcl.lpszClassName = wndclName;
73
74      if (!RegisterClassEx(&wndcl))
75          return GetLastError();
76
77      HWND hWindow = CreateWindow(
78          wndclName,
79          L"Window",
80          WS_CAPTION | WS_VISIBLE | WS_OVERLAPPED | WS_SYSMENU,
81          CW_USEDEFAULT,
82          CW_USEDEFAULT,
83          WIDTH,
84          HEIGHT,
85          NULL,
86          NULL,
87          hInstance,
88          NULL
89      );
```

```
90
91      if (!hWindow)
92          return GetLastError();
93
94      MSG msg;
95      while (GetMessage(&msg, NULL, NULL, NULL))
96      {
97          TranslateMessage(&msg);
98          DispatchMessage(&msg);
99      }
100
101     return msg.wParam;
102 }
```

```
104  void DrawTextLine(HWND hWnd)
105  {
106      PAINTSTRUCT paintStruct;
107      RECT rectPlace;
108
109      HDC drawer = BeginPaint(hWnd, &paintStruct);
110          SelectObject(drawer, hFont);
111          SetTextColor(drawer, RGB(0, 0, 0));
112          GetClientRect(hWnd, &rectPlace);
113          if (inputStage)
114          {
115              DrawText(drawer, (LPCWSTR)TextLine.txtln, TextLine.cur_id, &rectPlace, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
116          }
117          else
118          {
119              LPCWSTR res = Calculate();
120              FillRect(drawer, &rectPlace, (HBRUSH)GetStockObject(WHITE_BRUSH));
121              DrawText(drawer, res, wcslen(res), &rectPlace, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
122          }
123      EndPaint(hWnd, &paintStruct);
124  }
```

```
126  void SelectionSort(PDWORD num, DWORD size)
127  {
128      DWORD min, temp;
129      for (DWORD i = 0; i < size - 1; i++)
130      {
131          min = i;
132
133          for (DWORD j = i + 1; j < size; j++)
134          {
135              if (num[j] < num[min])
136                  min = j;
137          }
138          temp = num[i];
139          num[i] = num[min];
140          num[min] = temp;
141      }
142  }
```

```
144  PWSTR Append(PWSTR dest, PWSTR source)
145  {
146      DWORD destSz = wcslen(dest);
147      DWORD srcSz = wcslen(source);
148      PWSTR nstr = (PWSTR)calloc(destSz + srcSz + 2, sizeof(WCHAR));
149
150      if (destSz)
151      {
152          wcscpy_s(nstr, destSz + srcSz + 2, dest);
153          nstr[destSz] = L' ';
154          nstr[destSz + 1] = L'\0';
155      }
156      wcscat_s(nstr, srcSz + destSz + 2, source);
157
158      return nstr;
159  }
160
161  void LineToInt(PDWORD buf, const WORD numCount)
162  {
163      PDWORD arr_curptr = buf;
164
165      PWCHAR text_curptr = TextLine.txtln;
166      PWCHAR text_prevptr = TextLine.txtln;
167
168      TextLine.txtln[TextLine.cur_id] = '\0';
169      while (text_curptr != &TextLine.txtln[TextLine.cur_id + 1])
170      {
171          if (*text_curptr == L' ' || *text_curptr == L'\0')
172          {
173              *text_curptr++ = L'\0';
174              *arr_curptr++ = _wtoi(text_prevptr);
175              text_prevptr = text_curptr;
176          }
177          else ++text_curptr;
178      }
179  }
```

```
181  PWSTR Calculate()
182  {
183      DWORD arr[10] = { 0 };
184
185      LineToInt(arr, 10);
186      SelectionSort(arr, 10);
187
188      PWSTR res = (PWSTR)calloc(1, sizeof(WCHAR));
189      for (int i = 0; i < 10; ++i)
190      {
191          if (arr[i] % 2 != 0)
192          {
193              PWSTR buf = (PWSTR)calloc(20, sizeof(WCHAR));
194              _itow_s(arr[i], buf, 20, 10);
195              res = Append(res, buf);
196          }
197      }
198
199      return res;
200  }
```

```
#pragma once

void pushnum(WCHAR);
void popnum();

struct wtextline {
    const WORD MAX_TEXT_LEN = 40;

    DWORD cur_id = 0;
    WORD numOfSpaces = 0;

    LPWSTR txtln = (LPWSTR)calloc(MAX_TEXT_LEN + 1, sizeof(WCHAR));
    void (*PushNum)(WCHAR) = pushnum;
    void (*PopNum)() = popnum;
};

struct wtextline TextLine;
```

```
19  void pushnum(WCHAR key)
20  {
21      if (TextLine.cur_id < TextLine.MAX_TEXT_LEN)
22      {
23          if ((key >= L'0' && key <= L'9') || (key == L' ' && TextLine.txtln[TextLine.cur_id - 1] != L' ' && TextLine.numOfSpaces < 9))
24          {
25              if (key == L' ')
26                  TextLine.numOfSpaces++;
27              TextLine.txtln[TextLine.cur_id++] = key;
28          }
29      }
30  }
31
32  void popnum()
33  {
34      if (TextLine.cur_id > 0)
35      {
36          if (TextLine.txtln[TextLine.cur_id] == L' ')
37              TextLine.numOfSpaces--;
38          --TextLine.cur_id;
39      }
40  }
```