**Бруцкая Анастасия 251001**

**Лабораторная работа №2**

**Задание №1**

#define MIN\_ROWS 1  
#define MIN\_COLS 1  
#define MAX\_ROWS 20  
#define MAX\_COLS 20  
  
int nRows = 2;  
int nCols = 3;  
  
HWND hEdits[MAX\_ROWS][MAX\_COLS];  
  
int GetFontSize(HWND hWndEdit) {  
 RECT rect;  
 GetClientRect(hWndEdit, &rect);  
  
 int width = rect.right - rect.left;  
 int height = rect.bottom - rect.top;  
  
 int textLength = GetWindowTextLength(hWndEdit);  
  
 return sqrt(width\*height/(0.5\*textLength + 1))\*0.75;  
}  
  
  
void CreateTable(HWND hwnd) {  
 RECT rect;  
 GetClientRect(hwnd, &rect);  
 int width = rect.right - rect.left;  
 int height = rect.bottom - rect.top;  
  
 int colWidth = width / nCols;  
 int rowHeight = height / nRows;  
  
 for (int row = 0; row < MAX\_ROWS; ++row) {  
 for (int col = 0; col < MAX\_COLS; ++col) {  
 if (row < nRows && col < nCols) {  
 if (hEdits[row][col] == nullptr) {  
 hEdits[row][col] = CreateWindowEx(WS\_EX\_CLIENTEDGE, "EDIT", nullptr,  
 WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_MULTILINE | ES\_AUTOVSCROLL,  
 col \* colWidth, row \* rowHeight, colWidth, rowHeight,  
 hwnd, (HMENU) (row \* MAX\_COLS + col), GetModuleHandle(nullptr),  
 nullptr);  
 }  
 } else {  
 if (hEdits[row][col] != nullptr) {  
 DestroyWindow(hEdits[row][col]);  
 hEdits[row][col] = nullptr;  
 }  
 }  
 }  
 }  
}  
  
void ResizeTable(HWND hwnd) {  
 RECT rect;  
 GetClientRect(hwnd, &rect);  
 int width = rect.right - rect.left;  
 int height = rect.bottom - rect.top;  
  
 int colWidth = width / nCols;  
 int rowHeight = height / nRows;  
  
 for (int row = 0; row < nRows; ++row) {  
 for (int col = 0; col < nCols; ++col) {  
 SetWindowPos(hEdits[row][col], nullptr, col \* colWidth, row \* rowHeight,  
 colWidth, rowHeight, SWP\_NOZORDER);  
  
 int fontSize = GetFontSize(hEdits[row][col]);  
  
 HFONT newFont = CreateFont(  
 fontSize,  
 0,  
 0,  
 0,  
 FW\_NORMAL,  
 FALSE,  
 FALSE,  
 FALSE,  
 DEFAULT\_CHARSET,  
 OUT\_DEFAULT\_PRECIS,  
 CLIP\_DEFAULT\_PRECIS,  
 DEFAULT\_QUALITY,  
 DEFAULT\_PITCH | FF\_SWISS,  
 "Courier New"  
 );  
 SendMessage(hEdits[row][col], WM\_SETFONT, (WPARAM) newFont, TRUE);  
 }  
 }  
}  
  
LRESULT CALLBACK WndProc(HWND hWnd, UINT message,  
 WPARAM wParam, LPARAM lParam) {  
 switch (message) {  
 case WM\_CREATE: {  
 CreateTable(hWnd);  
 break;  
 }  
  
 case WM\_SIZE: {  
 ResizeTable(hWnd);  
 }  
 break;  
  
 case WM\_COMMAND: {  
 if (HIWORD(wParam) == EN\_UPDATE) {  
 int editId = LOWORD(wParam);  
 int row = editId / MAX\_COLS;  
 int col = editId % MAX\_COLS;  
  
 int fontSize = GetFontSize(hEdits[row][col]);  
  
 HFONT newFont = CreateFont(  
 fontSize,  
 0,  
 0,  
 0,  
 FW\_NORMAL,  
 FALSE,  
 FALSE,  
 FALSE,  
 DEFAULT\_CHARSET,  
 OUT\_DEFAULT\_PRECIS,  
 CLIP\_DEFAULT\_PRECIS,  
 DEFAULT\_QUALITY,  
 DEFAULT\_PITCH | FF\_SWISS,  
 "Courier New"  
 );  
 SendMessage(hEdits[row][col], WM\_SETFONT, (WPARAM) newFont, TRUE);  
 }  
 }  
 break;  
  
 case WM\_KEYDOWN: {  
 switch (wParam) {  
 case VK\_UP:  
 if (nRows < MAX\_ROWS) {  
 nRows++;  
 CreateTable(hWnd);  
 ResizeTable(hWnd);  
 }  
 break;  
 case VK\_DOWN:  
 if (nRows > MIN\_ROWS) {  
 nRows--;  
 CreateTable(hWnd);  
 ResizeTable(hWnd);  
 }  
 break;  
 case VK\_LEFT:  
 if (nCols > MIN\_COLS) {  
 nCols--;  
 CreateTable(hWnd);  
 ResizeTable(hWnd);  
 }  
 break;  
 case VK\_RIGHT:  
 if (nCols < MAX\_COLS) {  
 nCols++;  
 CreateTable(hWnd);  
 ResizeTable(hWnd);  
 }  
 break;  
 }  
 }  
 break;  
  
 case WM\_CLOSE:  
 DestroyWindow(hWnd);  
 break;  
  
 case WM\_DESTROY:  
 PostQuitMessage(0);  
 break;  
  
 default:  
 return DefWindowProc(hWnd, message, wParam, lParam);  
 }  
 return 0;  
}

**Задание №2**

LRESULT CALLBACK WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)  
{  
 static const int TEXT\_BUFFER\_SIZE = 256;  
 static char text[TEXT\_BUFFER\_SIZE];  
 static int textLength = 0;  
 static int charWidth = 25;  
  
 switch (uMsg)  
 {  
 case WM\_CREATE:  
 {  
 CreateWindow("EDIT", "", WS\_CHILD | WS\_VISIBLE | WS\_BORDER | ES\_AUTOHSCROLL,  
 50, 50, 200, 20, hwnd, (HMENU)1, nullptr, nullptr);  
 CreateWindow("BUTTON", "Set Text", WS\_CHILD | WS\_VISIBLE,  
 260, 50, 80, 25, hwnd, (HMENU)2, nullptr, nullptr);  
 return 0;  
 }  
  
 case WM\_COMMAND:  
 {  
 if (LOWORD(wParam) == 2)  
 {  
 GetDlgItemText(hwnd, 1, text, TEXT\_BUFFER\_SIZE);  
 textLength = strlen(text);  
 InvalidateRect(hwnd, nullptr, TRUE); // Перерисовать окно  
 }  
 return 0;  
 }  
  
 case WM\_PAINT:  
 {  
 PAINTSTRUCT ps;  
 HDC hdc = BeginPaint(hwnd, &ps);  
  
 RECT rect;  
 GetClientRect(hwnd, &rect);  
  
 int radius = textLength \* charWidth / (2 \* M\_PI);  
 int centerX = (rect.right - rect.left)/2;  
 int centerY = (rect.bottom - rect.top)/2;  
  
 if (textLength > 0)  
 {  
 double angleStep = 2 \* M\_PI / textLength;  
  
 SetTextAlign(hdc, TA\_CENTER | TA\_BASELINE);  
  
 for (int i = 0; i < textLength; ++i)  
 {  
 double angle = i \* angleStep;  
  
 int x = centerX + static\_cast<int>(radius \* cos(angle));  
 int y = centerY + static\_cast<int>(radius \* sin(angle));  
  
 TextOutA(hdc, x, y, &text[i], 1);  
 }  
 }  
  
 EndPaint(hwnd, &ps);  
 }  
 return 0;  
  
 case WM\_SIZE:  
 {  
 InvalidateRect(hwnd, nullptr, TRUE);  
 return 0;  
 }  
  
 case WM\_DESTROY:  
 PostQuitMessage(0);  
 return 0;  
 }  
 return DefWindowProc(hwnd, uMsg, wParam, lParam);  
}