МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

НР

НАУКА В РЕГИОНЫ

Учебное пособие к курсам
дополнительного общего образования
для первого–четвёртого годов обучения

Документация к библиотеке TX Library

Версия: 00173a, Ревизия: 168

Справочное руководство

Часть 2

Фонд развития
Физтех-лицея и Физтех-школ

Иннопрактика

МФТИ
Долгопрудный, 2021

Автор:
Дединский Илья Рудольфович
Старший преподаватель
кафедры информатики МФТИ

СОДЕРЖАНИЕ

2.8.48. bool txAlphaBlend (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource = 0, double ySource = 0, double alpha = 1.0)

Копирует изображение с одного холста (контекста рисования, DC) на другой с учётом полупрозрачности.

Аргументы:

destImage	Контекст назначения (куда копировать). Для копирования в окно **TXLib** укажите txDC().
xDest	X-координата верхнего левого угла копируемого изображения.
yDest	Y-координата верхнего левого угла копируемого изображения.
width	Ширина копируемой области. Если равна **0**, то равна ширине изображения-источника.
height	Ширина копируемой области. Если равна **0**, то равна ширине изображения-источника.
sourceImage	Контекст источника (откуда копировать). Должен иметь 32-битовый формат и альфа-канал (см. ниже).
xSource	X-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то **0**.
ySource	Y-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то **0**.
alpha	Общая прозрачность изображения, в дополнение к альфа-каналу (**0** — все прозрачно, **1** — использовать только альфа-канал). Необязательна. Если не указана, то 1.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Предупреждения:

- *Если контекст источника равен* `NULL`*, то он не существует и копирование вызовет ошибку. Наиболее частая причина — ошибка при загрузке файла изображения и отсутствие проверки на эту ошибку. Пример с проверкой на правильность загрузки см. ниже.*
- *Изображение-источник и изображение-приемник не могут налагаться друг на друга.*
- *Если прямоугольник копируемой области не полностью лежит внутри изображения-источника, то функция работать не будет. Такое бывает, если* `xSource` *или* `ySource` *отрицательны, или величина* `(xSource + width)` *больше ширины изображения-источника, или величина* `(ySource + height)` *больше высоты изображения-источника.*

Заметки:

- Изображение-источник должно быть загружено с помощью `txLoadImage()` и иметь 32-битовый RGBA-формат. Дополнительный канал (A: альфа-канал) 32-битового формата отвечает за попиксельную прозрачность участков изображения: для каждого пикселя в этом канале значение `0` означает его полную прозрачность, значение `255` — полную непрозрачность, промежуточные значения — полупрозрачность.

- Пустое 32-битовое изображение-источник также может быть создано с помощью функции `txCreateDIBSection()`, а 24-битовое — функцией `txCreateCompatibleDC()`. После создания на этом изображении можно рисовать, как на основном окне **TXLib**, используя параметр `dc` функций рисования.

Предупреждения:

- Если изображение имеет 32-битовый RGBA-формат файла, то его альфа-канал не должен быть полностью чёрным, иначе `txAlphaBlend()` посчитает изображение полностью прозрачным и не выведет на экран ничего.
- Если изображение-источник создано с помощью `txCreateDIBSection()` или `txLoadImage()`, то `txAlphaBlend()` может использовать как общую прозрачность, задачаемую параметром `alpha`, так и попиксельную прозрачность, задаваемую альфа-каналом.
- Если изображение имеет 24-битовый RGB-формат файлов (`TrueColor`), то это подразумевает, что попиксельной прозрачности нет и все пиксели изображения имеют равную прозрачность, задающуюся параметром `alpha`.
- Имейте в виду, что многие графические редакторы сохраняют изображение в 32-битовом формате, даже если вы явно указываете 24-битовый формат или формат `TrueColor`. Если при этом в изображении нет альфа-канала, то `txAlphaBlend` не выведет ничего. Так происходит, например, если сохранять картинку в **Microsoft Paint**, который не умеет создавать альфа-каналы и при этом сохраняет всегда в 32-битовом формате. Для таких случаев надо использовать `txTransparentBlt`.

- Для BMP-файла альфа-канал можно сделать, например, в **Adobe Photoshop**, командой "Новый канал (New Channel)" в палитре каналов (Channels). Чёрный цвет в альфа-канале соответствует полной прозрачности, белый — полной непрозрачности. При этом в прозрачных областях само изображение (в каналах R, G, B) должно быть чёрным, и чем прозрачнее, тем чернее. Такой формат цвета называется Premultiplied Alpha. *См. изображение* с альфа-каналом в примере `TX\Examples\Tennis\ Tennis.cpp` (файл с теннисной ракеткой: `TX\Examples\Tennis\ Resources\ Images\Racket.bmp`).

Иначе говоря, при пересчёте в формат Premultiplied Alpha надо домножить цвета в каналах R, G, B на значения альфа-канале `A: R, G, B *= A / 255.0`. Получится вот что:

- Если значение альфа-канала для некоторого пикселя равно 0 (полная прозрачность), тогда значения каналов R, G, B для этого пикселя также станут `0` (это чёрный цвет).
- Если значение альфа-канала для некоторого пикселя равно `255` (полная непрозрачность), тогда значения каналов R, G, B для этого пикселя не изменятся.

Для других значений альфа-канала, пиксели изображения станут темнее.

В редакторе ***Adobe Photoshop*** это можно сделать командой ***Image — Apply Image*** с параметрами:

Source:	Имя файла с картинкой
Layer:	Background
Channel:	Alpha 1
Blending:	Multiply
Opacity:	`100%`

Если изображение с альфа-каналом не находится в формате Premultiplied Alpha ~~и вам лень читать что, что написано выше~~, то для перевода в этот формат можно использовать функцию `txUseAlpha()`. Однако не надо вызывать `txUseAlpha()` несколько раз для одного и того же уже загруженного изображения, иначе оно может становиться темнее и темнее.

Заметки:

Стандартная функция `AlphaBlend` из **Win32 API** может масштабировать изображение. В `txAlphaBlend` это убрано для упрощения использования. If you still need image scaling, use original function `AlphaBlend` and don't mess with stupid TX-based tools. (See implementation of `txAlphaBlend` in **TX Library** source: open `TXLib.h` file in your editor and search for t`xAlphaBlend` function definition.)

См. также:

`txBitBlt()`, `txTransparentBlt()`, `txLoadImage()`, `txCreateCompatibleDC()`, `txSaveImage()`, `txGetExtent()`, `txCreateDIBSection()`, `txUseAlpha()`

Примеры использования:

Пример использования *см. в файле* `TX\Examples\Tennis\Tennis.cpp`.

```
HDC batman_CopiedFromHelp = txLoadImage ("Resources\\Images\\Batman.bmp");

if (!batman_CopiedFromHelp)
      txMessageBox ("Call to Batman failed because I copied it from TXLib Help);

txUseAlpha (batman_CopiedFromHelp); // If image colors are not premultiplied

// См. важный комментарий в примере к функции txLoadImage!
```

```
    txAlphaBlend (txDC(), 0, 0, 800, 600, batman_CopiedFromHelp);
...
    txDeleteDC (batman_CopiedFromHelp);   // Don't worry, batman will
...
    return batman_CopiedFromHelp;  //...and there he comes
```

2.8.49. bool txAlphaBlend (double xDest, double yDest, HDC sourceImage, double xSource = 0, double ySource = 0, double alpha = 1.0)

Копирует изображение на экран с учётом полупрозрачности.

Аргументы:

xDest	X-координата верхнего левого угла копируемого изображения.
yDest	Y-координата верхнего левого угла копируемого изображения.
sourceImage	Копируемое изображение.
xSource	X-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то **0**.
ySource	Y-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то **0**.
alpha	Общая прозрачность изображения, в дополнение к альфа-каналу (**0** — все прозрачно, **1** — использовать только альфа-канал). Необязательна. Если не указана, то **1**.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

См. описание в функции `txAlphaBlend()` выше.

HDC `txUseAlpha` (HDC image)

Пересчитывает цвета пикселей с учётом прозрачности (переводит цвета в формат Premultiplied Alpha).

Аргументы:

image	Дескриптор холста, изображение которого пересчитывается.

Возвращает:

- Если операция была успешна, возвращается исходный `HDC`, иначе — `NULL`.
- Пересчёт цветов в каналах R, G, B в формат Premultiplied Alpha с учётом значения в альфа-канале A идет по формуле `R, G, B *= A / 255.0`.

Получается вот что:

- Если значение альфа-канала для некоторого пикселя равно `0` (полная прозрачность), тогда значения каналов R, G, B для этого пикселя также станут 0 (это чёрный цвет).
- Если значение альфа-канала для некоторого пикселя равно `255` (полная непрозрачность), тогда значения каналов R, G, B для этого пикселя не изменятся.
- Для других значений альфа-канала, пиксели изображения станут темнее.

- Пересчёт цветов пикселей с учётом их прозрачности в формат Premultiplied Alpha необходим:
- В случае ручного изменения цветов, пример *см. в функции* `txCreateDIBSection()`.
- После загрузки картинок из файла с помощью `txLoadImage()`, если цвета изображения в нём не были заранее домножены на альфа-канал в **Adobe Photoshop** или аналогичной программе (см. замечания к функции `txAlphaBlend`).

См. также замечания к функции `txAlphaBlend()`.

См. также:

`txCreateCompatibleDC()`, `txCreateDIBSection()`, `txLoadImage()`, `txDeleteDC()`, `txAlphaBlend()`

Примеры использования:

См. в функции `txCreateDIBSection()`.

2.8.50. `bool txSaveImage (const char filename[], HDC dc = txDC())`

Сохраняет в файл изображение в формате BMP.

Аргументы:

filename	Имя файла с расширением "BMP", куда будет записано изображение в формате BMP.
dc	Дескриптор холста, изображение которого сохраняется в файл. Необязателен. Если `txDC()`, сохраняется изображение окна **TXLib**.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

См. также:

`txCreateCompatibleDC()`, `txLoadImage()`, `txDeleteDC()`, `txBitBlt()`, `txAlphaBlend()`, `txTransparentBlt()`

Примеры использования:x

```
txDrawMan (50, 110, 100, 100, TX_YELLOW, 0.3, -0.5, -0.4, 0, 0.8, 1, -1, 0.5, 1, 1);

HDC dc = txCreateCompatibleDC (100, 110);
txBitBlt (dc, 0, 0, 100, 110, txDC());
txSaveImage ("TXLibMan.bmp", dc);

txBitBlt           (dc,     0,    0,    100,   110,   txDC());
txBitBlt           (txDC(), 200,  200,  0,     0,     dc);
txAlphaBlend       (txDC(), 100,  100,  0,     0,     dc, 0, 0, 0.25, 0);
txDeleteDC (dc);

txSaveImage ("ScreenShot.bmp");
```

2.8.51. double txSleep (double time = 0)

Задерживает выполнение программы на определённое время.

Аргументы:

time	Задержка в миллисекундах. Необязательна. Если не указана, то используется минимальная возможная задержка.

Возвращает:

Реальное время задержки в миллисекундах.

Заметки:

Перед началом задержки изображение в окне обязательно обновится, даже если рисование заблокировано через `txBegin()`.

Более полную информацию об обновлении изображения окна *см. в функции* `txBegin()`.

См. также:

`txBegin()`, `txEnd()`, `txRedrawWindow()`, `txUpdateWindow()`, `txDC()`

Примеры использования:

```
txSleep (500); // ПП: Поспать Полсекунды
```

2.8.52. int txBegin ()

Блокирует обновление изображения окна, во избежание мигания.

Для снятия блокировки используется функция `txEnd()`.

TXLib реализует двойную буферизацию. Все рисовательные действия происходят со скрытым HDC, находящемся в памяти, и его содержимое периодически автоматически копируется на экран. Это иногда приводит к мерцанию. Автоматическое копирование можно выключить функцией `txBegin()` и обратно включить функцией `txEnd()`, в этом случае содержимое окна можно перерисовать функциями `txRedrawWindow()` или `txSleep()`.

Если в программе требуется задержка, то используйте функции `txRedrawWindow()` или `txSleep()`, так как они автоматически обновляют изображение, независимо от состояния блокировки.

Предупреждения:

Избегайте блокирования на долгое время. Это может привести к дефектам изображения в окне.

Заметки:

Если нажата клавиша [Alt]+[PrintScreen], то блокировка временно отменяется.

Возвращает:

Значение счётчика блокировки (если 0, то рисование разблокировано).

См. также:

`txEnd()`, `txSleep()`, `txUpdateWindow()`, `txDC()`, `txVideoMemory()`, `txTextCursor()`

Примеры использования:

```
    txBegin();                // Здесь изображение "замёрзнет"
    txSetFillColor (TX_WHITE);
    txClear();                // Это вызвало бы мигание без txBegin()
    txSetFillColor (TX_RED);
    txRectangle (100, 100, 200, 200);
    txEnd();                  // Здесь мы сразу увидим окончательный рисунок
//
// Как правильно использовать блокировку обновления окна в циклах:
//
```

```
int x = 0, y = 0;
txBegin();                              // Отключаем автоматическое обновление окна
while (!GetAsyncKeyState (VK_ESCAPE)) // Цикл, пока не нажата клавиша ESCAPE
      {
      txSetFillColor (TX_BLACK);
      txClear();                        // Очищаем окно

      txSetFillColor (TX_DARKGRAY);
      txCircle (x, y, 50);
      txSetFillColor (TX_LIGHTGRAY);
      txCircle (x, y, 30);
      txSetFillColor (TX_WHITE);
      txCircle (x, y, 10);

      x += 5;                           // Изменяем координаты объекта
      y += 5;
      txSleep (50);                     // Делаем задержку
      }
txEnd();                                // Включаем автоматическое обновление окна
//
// Как *НЕ* правильно использовать блокировку:
//
```

```
txBegin();
while (!GetAsyncKeyState (VK_ESCAPE))
      {
      txSetFillColor (TX_BLACK);      // Очищаем окно
      txClear();
      txSleep (50);                   // Первая ошибка

      txSetFillColor (TX_DARKGRAY);   // Рисуем объект, но он не будет виден.
      txCircle (x, y, 50);
      txSetFillColor (TX_LIGHTGRAY);
      txCircle (x, y, 30);
      txSetFillColor (TX_WHITE);
      txCircle (x, y, 10);
x += 5;
y += 5;
txEnd();                              // Вторая ошибка
}                                     // блокировка обновления
```

2.8.53. int txEnd ()

Разблокирует обновление окна, заблокированное функцией txBegin().

Предупреждения:

Если txBegin() *вызывалась несколько раз, то для снятия блокировки нужно столько же раз вызвать* txEnd().

Возвращает:

Значение счётчика блокировки (если 0, то рисование разблокировано).

Более полную информацию об автоматическом обновлении см. в функции txBegin().

См. также txRedrawWindow(), txUpdateWindow(), txSleep().

Заметки:

Если нажата клавиша [Alt]+[PrintScreen], то блокировка временно отменяется.

См. также:

txBegin(), txSleep(), txUpdateWindow(), txDC(), txVideoMemory(), txTextCursor()

Примеры использования:

```
txBegin();                      // Здесь изображение "замёрзнет"
txSetFillColor (TX_WHITE);
txClear();                      // Это вызвало бы мигание без txBegin()
txSetFillColor (TX_RED);
txRectangle (100, 100, 200, 200);
txEnd();                        // Здесь мы сразу увидим окончательный рисунок
```

2.8.54. void txRedrawWindow ()

Обновляет изображение в окне **TXLib** вручную.

Более полную информацию об автоматическом обновлении см. в функции `txBegin()`.
См. также `txEnd()`, `txUpdateWindow()`, `txSleep()`.

Предупреждения:

Реализация этой функции неэффективна: она вызывает `txSleep (0)`*, что вызывает задержку. Чтобы сделать более эффективную реализацию, посмотрите исходный текст функций* `txRedrawWindow()` *и* `txSleep()` *и сделайте лучше. Либо напишите свою, более быструю библиотеку на основе* ***Win32 GDI, GDI Plus, OpenGL, DirectX, SDL*** *или* ***SFML****.*

См. также:

`txWindow()`, `txBegin()`, `txEnd()`, `txLock()`, `txUnlock()`, `txGDI()`

Примеры использования:

```
txBegin();                // Выключаем автоматическое обновление экрана
txCircle (100, 100, 50);  // Рисуем – изображение не появляется
Sleep (3000);             // не txSleep()! txSleep() сама обновит изображение
txRedrawWindow();         // Обновляем экран вручную – изображение появляется
```

2.8.55. bool txDestroyWindow (HWND wnd = txWindow())

Уничтожает окно.

Аргументы:

wnd	Дескриптор окна для уничтожения. Необязательно. Если не указан, уничтожается окно **TXLib**.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Предупреждения:

Если уничтожается окно **TXLib**, функция `main()` принудительно прерывается и программа завершается. При этом не гарантируется правильное завершение программы, поэтому так делать не рекомендуется.

См. также:

`txCreateWindow()`

Примеры использования:

```
txDestroyWindow();  // Farewell to the world
```

2.8.56. double txQueryPerformance ()

Оценивает скорость работы компьютера.

Возвращает:

Скорость работы (графических операций) в условных единицах.

См. также:

`txSleep()`, `txGetFPS()`

Примеры использования:

```
if (txQueryPerformance() < 1) printf ("Хочется новый компьютер");
```

2.8.57. double txGetFPS (int minFrames = txFramesToAverage)

Выдает количество кадров (вызовов этой функции) в секунду.

Аргументы:

minFrames	Количество вызовов, после которых FPS начинает усредняться по последним `txFramesToAverage` кадрам. Необязательно. Если не указано, используется `txFramesToAverage`.

Возвращает:

FPS (Frames per Second), т.е. количество кадров (вызовов этой функции) в секунду.

Заметки:

Когда количество вызовов этой функции превысит `minFrames`, FPS начинает усредняться по последним `txFramesToAverage` кадрам. Максимальный интервал усреднения — `txFramesToAverage` кадров.

См. также:

`txSleep()`, `txQueryPerformance()`

Примеры использования:

См. в функции `txCreateDIBSection()`.

2.8.58. unsigned txSetConsoleAttr (unsigned colors = 0x07)

Устанавливает цветовые атрибуты консоли.

Аргументы:

colors	Цветовые атрибуты консоли. Необязательны. Если не указаны, то цвет — светло-серый.

Возвращает:

Предыдущие цветовые атрибуты консоли.

Атрибуты — это цвет текста (`colorText`) и цвет фона (`colorBackground`), объединённые вместе:

```
colors = colorText + colorBackground * 16
```

либо

```
colors = colorText | (colorBackground << 4)
```

Цвета атрибутов не имеют никакого отношения к цветам рисования, задаваемыми `TX_`... константами, `RGB()`, `txSetColor()`, `txSetFillColor()` и т.д. Значения цветов см. ниже.

Значения цветов атрибутов

Dec	Hex	
0 =	0x0	= BLACK (Чёрный)
1 =	0x1	= BLUE (Синий)
2 =	0x2	= GREEN (Зелёный)

3 =	0x3	= CYAN (Сине-зелёный)
4 =	0x4	= RED (Красный)
5 =	0x5	= MAGENTA (Малиновый)
6 =	0x6	= DARKYELLOW (Тёмно-жёлтый)
7 =	0x7	= LIGHTGRAY (Светло-серый)
8 =	0x8	= DARKGRAY (Тёмно-серый)
9 =	0x9	= LIGHTBLUE (Светло-синий)
10 =	0xA	= LIGHTGREEN (Светло-зелёный)
11 =	0xB	= LIGHTCYAN (Светло-сине-зелёный)
12 =	0xC	= LIGHTRED (Светло-красный)
13 =	0xD	= LIGHTMAGENTA (Светло-малиновый)
14 =	0xE	= YELLOW (Жёлтый)
15 =	0xF	= WHITE (Белый)

Цвета букв образуются соединением слов `FOREGROUND_`<*название цвета*>, например, `FOREGROUND_WHITE`.

Цвета фона образуются соединением слов `BACKGROUND_`<*название цвета*>, например, `BACKGROUND_CYAN`.

В шестнадцатеричной системе счисления атрибуты задавать так: если нужен, скажем, жёлтый цвет на синем фоне, то его код будет **`0x1e`** (старшая цифра — старшие 4 бита — это цвет фона, младшая цифра — младшие 4 бита — это цвет текста).

См. также:

txTextCursor(), txGetConsoleAttr(), txSetConsoleCursorPos(), txGetConsoleCursorPos(), txGetConsoleExtent(), txGetConsoleFontSize(), txClearConsole()

Примеры использования:

```
txSetConsoleAttr (FOREGROUND_YELLOW | BACKGROUND_BLUE);
printf ("А в небе 0x1 есть город 0xE");  // (с) Б. Гребенщиков
```

2.8.59. unsigned txGetConsoleAttr ()

Возвращает текущие цветовые атрибуты консоли.

Возвращает:

Текущие цветовые атрибуты консоли. *См.* txSetConsoleAttr().

См. также:

txTextCursor(), txSetConsoleAttr(), txSetConsoleCursorPos(), txGetConsoleCursorPos(), txGetConsoleExtent(), txGetConsoleFontSize(), txClearConsole()

Примеры использования:

```
unsigned attr = txGetConsoleAttr();
```

2.8.60. bool txClearConsole ()

Стирает текст консоли.

Возвращает:

Если операция была успешна — true, иначе — false.

При стирании используются текущие атрибуты (цвета текста и фона) консоли.

См. также:

txTextCursor(), txSetConsoleAttr(), txGetConsoleAttr(), txGetConsoleCursorPos(), txGetConsoleExtent(), txGetConsoleFontSize(), txClearConsole()

Примеры использования:

```
txClearConsole(); // Ну вот и все, дружок
```

2.8.61. POINT txSetConsoleCursorPos (double x, double y)

Устанавливает позицию мигающего курсора консоли.

Аргументы:

x	X-координата курсора в пикселях.
y	Y-координата курсора в пикселях.

Возвращает:

Предыдущее положение мигающего курсора в пикселях, в структуре POINT.

Заметки:

Нельзя установить совсем любую позицию. Текст в консоли расположен по прямоугольной сетке, размер которой зависит от размеров шрифта консоли. Устанавливаемая позиция округляется, чтобы курсор попал в ячейку сетки. *См. пример* к функции txGetConsoleFontSize().

См. также:

txTextCursor(), txSetConsoleAttr(), txGetConsoleAttr(), txGetConsoleCursorPos(), txGetConsoleExtent(), txClearConsole()

Примеры использования:

```
txSetConsoleCursorPos (txGetExtentX(), txGetExtentY());   // Центр Вселенной
```

2.8.62. POINT txGetConsoleCursorPos ()

Возвращает позицию мигающего курсора консоли.

Возвращает:

Положение мигающего курсора в пикселях, в структуре POINT.

См. также:

txTextCursor(), txSetConsoleAttr(), txGetConsoleAttr(), txSetConsoleCursorPos(), txGetConsoleExtent(), txClearConsole()

Примеры использования:

```
POINT pos = txGetConsoleCursorPos();
```

2.8.63. POINT txGetConsoleExtent ()

Возвращает размер консоли.

Возвращает:

Положение мигающего курсора в символах, в структуре POINT.

См. также:

txTextCursor(), txSetConsoleAttr(), txGetConsoleAttr(), txSetConsoleCursorPos(), txGetConsoleExtent(), txClearConsole()

Примеры использования:

```
POINT size = txGetConsoleExtent();
```

2.8.64. POINT txGetConsoleFontSize ()

Возвращает размеры шрифта консоли.

Возвращает:

Размеры шрифта консоли в пикселях, в структуре `POINT`.

См. также:

`txTextCursor()`, `txSetConsoleAttr()`, `txGetConsoleAttr()`, `txSetConsoleCursorPos()`, `txGetConsoleExtent()`, `txGetConsoleFontSize()`, `txClearConsole()`

Примеры использования:

```
POINT size = txGetConsoleFontSize();
txSetConsoleCursorPos (5 * size.x, 10 * size.y);    // А теперь мигай вот там
```

2.8.65. bool txTextCursor (bool blink = true)

Запрещает или разрешает рисование мигающего курсора в окне.

Аргументы:

blink	false — запретить мигающий курсор. Если не указано, то мигание разрешается.

Возвращает:

Предыдущее значение состояния курсора.

См. также:

txSetConsoleAttr(), txGetConsoleAttr(), txSetConsoleCursorPos(), txGetConsoleCursorPos(), txGetConsoleFontSize(), txClearConsole(), txCreateWindow(), txUpdateWindow(), txLock(), txUnlock(), txGDI()

Примеры использования:

```
    txTextCursor (false);
...
    txTextCursor();
```

2.8.66. const char* txVersion ()

Возвращает строку с информацией о текущей версии библиотеки.

Возвращает:

Строка с информацией о текущей версии библиотеки.

Примеры использования:

```
printf ("I personally love %s\n", txVersion());
```

2.8.67. unsigned txVersionNumber ()

Возвращает номер версии библиотеки.

Возвращает:

Номер версии библиотеки.

Примеры использования:

```
printf ("My magic number is %x\n", txVersionNumber());
```

2.8.68. const char* txGetModuleFileName (bool fileNameOnly = true)

Возвращает имя исполняемого файла или изначальный заголовок окна **TXLib**.

Аргументы:

fileNameOnly	Возвратить только полное имя исполняемого файла, полученного через **Win32** функцию `GetFileModuleName (NULL, ...)`. Необязательно. Если не указано, то возвращается полное имя исполняемого файла.

Возвращает:

`fileNameOnly = true:` Имя исполняемого файла.

`fileNameOnly = false:` Изначальный заголовок окна **TXLib**.

Заметки:

Возвращается статическая строка.

См. также:

`txWindow()`, `txVersion()`, `txVersionNumber()`

Примеры использования:

```
printf ("Смотрите на заголовок окна!");

for (int done = 0; done <= 100; done++)
      {
      char title [1024] = "";
      sprintf (title, "%s - [%-10.*s] %d%%", txGetModuleFileName (false), done/10);
```

```
        SetWindowText (txWindow(), title);
        SetWindowText (Win32::GetConsoleWindow(), title);
        txSleep (50);
        }
txMessageBox ("Вот такой вот progress bar", "TXLib forever)");
```

2.8.69. int txUpdateWindow (int update = true)

Разрешает или запрещает автоматическое обновление изображения в окне.

Аргументы:

update	Режим обновления (`true` — разрешить, `false` — запретить). Необязателен. Если не указан, то "разрешить".

Возвращает:

- Предыдущее состояние режима обновления.
- В отличие от `txBegin()` и `txEnd()`, которые поддерживают вложенные вызовы и работают как "скобки для замерзания картинки", `txUpdateWindow()` позволяет явно установить или снять блокировку автоматического обновления экрана.
- Более полную информацию об автоматическом обновлении *см. в функции* `txBegin()`. *См. также* `txEnd()`, `txRedrawWindow()`, `txSleep()`.

См. также:

`txBegin()`, `txEnd()`, `txSleep()`, `txUpdateWindow()`, `txDC()`, `txVideoMemory()`, `txTextCursor()`, `txLock()`, `txUnlock()`, `txGDI()`

Примеры использования:

```
    txUpdateWindow (false);
...
    txUpdateWindow();
```

2.8.70. bool txSelectObject (HGDIOBJ obj, HDC dc = txDC())

Устанавливает текущий активный объект GDI.

Аргументы:

obj	Дескриптор объекта GDI.
dc	Холст (контекст рисования), в котором устанавливается текущий активный объект GDI. Необязателен.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Заметки:

Предыдущий выбранный объект того же типа (PEN/BRUSH/BITMAP и т.п.), как и `obj`, уничтожается.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txSelectFont()`

Примеры использования:

```
HPEN pen = CreatePen (PS_DASH, 1, RGB (255, 128, 0));
txSelectObject (pen);
```

2.8.71. bool txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture ()

Делает нечто иногда удобное. См. название функции.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

У этой функции есть синоним с простым понятным названием, поищите его в файле библиотеки, около определения этой функции. Или можно скопировать набрать это километровое имя и посмотреть, что получится.

См. также:

txCreateWindow(), txSleep()

Примеры использования:

```
int main()
    {
    txCreateWindow (800, 600);
    txSetTextAlign (TA_CENTER);
    txTextOut (txGetExtentX()/2, txGetExtentY()/2, "Press any key to exit!");
    txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture();
    return 0;
    }
```

2.9. Перечисления

2.9.1. enum txColors

Названия предопределённых цветов.

См. `TX_BLACK, TX_BLUE` и другие цвета в списке выше.

Если кому-то эти цвета не нравятся (что неудивительно), всегда можно сделать свои с помощью `RGB()`. См. пример ниже.

См. также:

`txSetColor(), txSetFillColor(), txGetColor(), txGetFillColor(), txGetPixel(), RGB()`

Примеры использования:

```
    #include "TXLib.h"

    const COLORREF MY_DEEP_ROMANTIC_BLUE = RGB (  0,   0, 129),
                   MY_SHINING_MOONLIGHT  = RGB (128, 255,  64);
...
    txSetColor         (TX_YELLOW);       // Устанавливаем жёлтый цвет линий
    txSetFillColor     (TX_NULL);         // Заливка фигур будет прозрачная

    txSetFillColor     (MY_DEEP_ROMANTIC_BLUE);  // А.И. Куинджи, "Лунная ночь"
    txSetColor         (MY_SHINING_MOONLIGHT
```

Элементы перечислений:

TX_BLACK	Чёрный цвет.
TX_BLUE	Тёмно-синий цвет. Плохо виден.
TX_GREEN	Зелёный цвет.
TX_CYAN	Бирюзовый цвет.
TX_RED	Тёмно-красный цвет. Слишком тёмный.
TX_MAGENTA	Тёмно-малиновый цвет.
TX_BROWN	Коричневый цвет. Некрасивый. Можно сделать свои с помощью `RGB()`
TX_ORANGE	Оранжевый цвет.
TX_GRAY	Серый цвет.
TX_DARKGRAY	Тёмно-серый цвет.
TX_LIGHTGRAY	Светло-серый цвет.
TX_LIGHTBLUE	Светло-синий цвет.
TX_LIGHTGREEN	Светло-зелёный цвет.
TX_LIGHTCYAN	Светло-бирюзовый цвет.
TX_LIGHTRED	Светло-красный цвет. Не самого лучшего оттенка.
TX_LIGHTMAGENTA	Светло-малиновый цвет. Ещё менее лучшего оттенка.
TX_PINK	Розовый гламурный :)
TX_YELLOW	Жёлтый цвет.
TX_WHITE	Белый цвет.

TX_TRANSPARENT	Прозрачный цвет. Отключает рисование.
TX_NULL	Прозрачный цвет. Отключает рисование.
TX_HUE	Цветовой тон цвета в модели HSL.
TX_SATURATION	Насыщенность цвета в модели HSL.
TX_LIGHTNESS	Светлота цвета в модели HSL.

3. ПОДДЕРЖКА МЫШИ!

3.1. Работа с Мышью!

- `POINT txMousePos ()`

  Возвращает позицию Мыши!

- `double txMouseX ()`

  Возвращает X-Координату Мыши!

- `double txMouseY ()`

  Возвращает Y-Координату Мыши!

- `unsigned txMouseButtons ()`

  Возвращает состояние Кнопок Мыши!

- `Mouse & txCatchMouse (bool shouldEat=true)`

  Ловит Мышь!

3.1.1. Функции

3.1.2. POINT txMousePos ()

Возвращает позицию Мыши!

Возвращает:

Позиция Мыши! как структура `POINT`, отсчитывающаяся от левого верхнего угла окна.

Заметки:

Если окна **TXLib** нет (какой ужас!), то координаты отсчитываются от верхнего левого угла экрана.

См. также:

`txMouseX(), txMouseY(), txMousePos(), txMouseButtons()`

Примеры использования:

```
RECT area = { 100, 100, 110, 110 };
while (txMouseButtons() != 1)
      {
      if (In (txMousePos(), area)) txTextOut(100, 100, "http://vk.com/TiShtoTamDelaesh?!");
      txSleep (0);
      }
```

3.1.3. double txMouseX ()

Возвращает X-Координату Мыши!

Возвращает:

X-координата Мыши!, отсчитывающаяся от левого верхнего угла окна.

Заметки:

Если окна **TXLib** нет (какой ужас!), то координаты отсчитываются от верхнего левого угла экрана.

См. также:

`txMouseX(), txMouseY(), txMousePos(), txMouseButtons()`

Примеры использования:

```
while (txMouseButtons() != 1)
      {
      txCircle (txMouseX(), txMouseY(), 20);
      txSleep (0);
      }
```

3.1.4. double txMouseY ()

Возвращает Y-Координату Мыши!

Возвращает:

Y-координата Мыши!, отсчитывающаяся от левого верхнего угла окна.

Заметки:

Если окна **TXLib** нет (какой ужас!), то координаты отсчитываются от верхнего левого угла экрана.

См. также:

`txMouseX(), txMouseY(), txMousePos(), txMouseButtons()`

Примеры использования:

```
while (txMouseButtons() != 1)
      {
      txCircle (txMouseX(), txMouseY(), 20);
      txSleep (0);
      }
```

3.1.5. unsigned txMouseButtons ()

Возвращает состояние Кнопок Мыши!

Возвращает:

Состояние Кнопок Мыши!

В возвращаемом значении выставленный в единицу 1-й (младший) бит означает нажатую левую Кнопку Мыши!, 2-й — правую Кнопку.

Например, возвращённое число 3 (двоичное 11) означает одновременное нажатие левой и правой Кнопок.

См. также:

`txMouseX(), txMouseY(), txMousePos(), txMouseButtons()`

Примеры использования:

```
while (txMouseButtons() != 3)
      {
      if (txMouseButtons() & 1) txCircle (txMouseX(), txMouseY(), 20);
      if (txMouseButtons() & 2) txLine   (txMouseX(), txMouseY(), 0, 0);
      txSleep (0);
      }
```

3.1.6. Mouse& txCatchMouse (bool shouldEat = true)

Ловит Мышь!

Аргументы:

shouldEat	To eat, or not to eat: that is the question.

Возвращает:

Пойманная Мышь!

Заметки:

Эту функцию можно применять, только если Вы — Кот. =^..^=

Поэтому в текущей версии она не реализована :)

См. также:

`txMouseX()`, `txMouseY()`, `txMousePos()`, `txMouseButtons()`

Примеры использования:

```
void CatsLife()
      {
      try
            {
            while (true)
                  {
                  Mouse mouse = txCatchMouse();
                  Eat (mouse);
                  txSleep();
                  }
            }
      catch (Mouse& mouse)
            {
            Eat (mouse);  // Just do eat (R). Anyway
            }
      }
```

4. РАЗНОЕ

4.1. Функции

4.1.1. void* operator new (size_t size, int)

Выделяет блок динамической памяти через new с обнулением его содержимого перед вызовом конструктора.

Аргументы:

size	Размер элемента/элементов

Возвращает:

Выделенная память

Примеры использования:

```
   int* i = new(0) int;
...
   delete i;
```

4.1.2. void* operator new (size_t size, size_t items, int)

Выделяет блок динамической памяти через new[] с обнулением содержимого перед вызовом конструкторов.

Аргументы:

size	Размер элементов
items	Количество элементов массива

Возвращает:

Выделенная память

Примеры использования:

```
   char* str = new(0) char [100];
...
   delete[] str;
```

4.1.3. int txUpdateWindow (intupdate = true)

Разрешает или запрещает автоматическое обновление изображения в окне.

Аргументы:

update	Режим обновления (`true` — разрешить, `false` — запретить). Необязателен. Если не указан, то "разрешить".

Возвращает:

Предыдущее состояние режима обновления.

В отличие от `txBegin()` и `txEnd()`, которые поддерживают вложенные вызовы и работают как "скобки для замерзания картинки", `txUpdateWindow()` позволяет явно установить или снять блокировку автоматического обновления экрана.

Более полную информацию об автоматическом обновлении см. в функции `txBegin()`.
См. также `txEnd()`, `txRedrawWindow()`, `txSleep()`.

См. также:

`txBegin()`, `txEnd()`, `txSleep()`, `txUpdateWindow()`, `txDC()`, `txVideoMemory()`, `txTextCursor()`, `txLock()`, `txUnlock()`, `txGDI()`

Примеры использования:

```
    txUpdateWindow (false);
...
    txUpdateWindow();
```

4.1.4. bool txSelectObject (HGDIOBJ obj, HDC dc = txDC())

Устанавливает текущий активный объект GDI.

Аргументы:

obj	Дескриптор объекта GDI.
dc	Холст (контекст рисования), в котором устанавливается текущий активный объект GDI. Необязателен.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Заметки:

Предыдущий выбранный объект того же типа (`PEN/BRUSH/BITMAP` и т.п.), как и obj, уничтожается.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txSelectFont()`

Примеры использования:

```
HPEN pen = CreatePen (PS_DASH, 1, RGB (255, 128, 0));
txSelectObject (pen);
```

4.1.5. bool txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture ()

Делает нечто иногда удобное. *См. название функции.*

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

У этой функции есть синоним с простым понятным названием, поищите его в файле библиотеки, около определения этой функции. Или можно скопировать набрать это километровое имя и посмотреть, что получится.

См. также:

`txCreateWindow(), txSleep()`

Примеры использования:

```
int main()
      {
      txCreateWindow (800, 600);
      txSetTextAlign (TA_CENTER);

      txTextOut (txGetExtentX()/2, txGetExtentY()/2, "Press any key to exit!");
txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture();
      return 0;
      }
```

4.1.6. bool txPlaySound (const char filename[] = NULL, DWORD mode = SND_ASYNC)

Воспроизводит звуковой файл.

Аргументы:

filename	Имя звукового файла, включая расширение. Если не указано или NULL, то останавливает звук.
mode	Режим воспроизведения. Необязательно. Если не указан, то SND_ASYNC (см. ниже).

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Режимы воспроизведения:

SND_ASYNC	Звук проигрывается одновременно с работой программы. Чтобы отменить звучание, вызовите `txPlaySound (NULL)`.
SND_SYNC	Выполнение программы приостанавливается до окончания воспроизведения звука.
SND_LOOP	Зацикливать звук при воспроизведении. Чтобы отменить звучание, вызовите `txPlaySound (NULL)`.
SND_NODEFAULT	Не использовать звук по умолчанию, если нельзя проиграть указанный звуковой файл.
SND_NOSTOP	Если какой-либо звук уже проигрывается, не останавливать его для запуска указанного звука.
SND_APPLICATION	Проигрывать звук, используя программу, зарегистрированную для данного типа звуковых файлов.

Заметки:

Поддерживаются только файлы в формате WAV. Остальные форматы (MP3 и др.) надо перекодировать в WAV. Переименование со сменой расширения не поможет, как и в случае с форматом картинок в `txLoadImage()`.

См. также:

`txSpeak()`, `txPlayVideo()`

Примеры использования:

```
txPlaySound ("tada.wav"); // So happy that this always exists
```

4.1.7. int txSpeak (const char *text, ...)

Читает ~~мысли~~ текст вслух.

Аргументы:

text	Текст для внеклассного чтения, как для функции printf().

Возвращает:

Время чтения текста в миллисекундах (если не указана опция `\a`). Если -1, то текст не прочитался. :(

Для использования этой функции укажите перед включением файла `TXLib.h` определите имя `TX_USE_SPEAK` командой `#define TX_USE_SPEAK` до строки с `#include "TXLib.h"` для того, чтобы **TXLib** задействовал библиотеку `Microsoft Speech API (SAPI.h)`.

См. пример использования ниже.

Текст читается голосом, установленным по умолчанию в **Панели управления Windows.** Если голос поддерживает русский язык (начиная с **Windows 8.1**), то можно использовать русские фразы. До **Windows 8.1** они не проговариваются. Если голос читает плохо, или с акцентом, то смените его в Панели Управления **Windows**.

Заметки:

- Если текст начинается с символа `\v`, то он ещё и печатается на экране (сам символ `\v` не печатается).

- Если текст начинается с символа `\a`, то он проговаривается асинхронно: `txSpeak()` возвращается сразу после вызова, и чтение идёт параллельно с работой программы. Без этой опции `txSpeak()` делает паузу в программе, ожидая конца чтения, и только потом возвращается.
- Если текст начинается с символа `<`, то он трактуется как формат `SSML` (*см. ниже*).
- Если строка с текстом пустая (не считая `\v` и `\a`), то `txSpeak()` лучше скажет что-то более важное, чем просто молчание. :)

Предупреждения:

Для работы этой функции ***TXLib*** *требует наличия файла* ***SAPI.h*** *из стандартной библиотеки* ***Windows****. Не на всех платформах он есть. Если этот файл не найден, то будет ошибка компиляции.*

Этот файл точно есть в ***Visual Studio*** *или в этой сборке платформы* `MinGW:` `https://nuwen.net/mingw.html`.

Заметки:

Вы можете настраивать синтез речи, используя язык разметки SSML (Speech Synthesis Markup Language). Его описание см. здесь: `https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-synthesis-markup`. Признаком наличия разметки SSML является наличие открывающей угловой скобки `<` в начале сообщения (после опций `\v` и `\a`). В этом случае будьте осторожны с амперсандами `&`, кавычками и знаками *"меньше"* `<` и *"больше"* `>`, а также другими спецсимволами XML. Если вы не знаете, как задавать их в XML, то не используйте. Формату SSML нужно строго соответствовать, иначе программа упадёт, ~~как это принято у Microsoft.~~

Если вам не нравится, что надо каждый раз указывать заголовок SSML, то вам это правильно не нравится. Сделайте функцию-обёртку над txSpeak, заодно познакомитесь, как делать функции с переменным числом параметров. Не забудьте о vsnprintf @ (vsprintf), эти функции сэкономят вам немало сил.

См. также:

txPlaySound(), txPlayVideo(), txMessageBox(), txOutputDebugPrintf()

Примеры использования:

```
#define TX_USE_SPEAK
#include "TXLib.h"

int main()
      {
      txSpeak ("TX Library is cool!");

      txSpeak ("\vДобрый вечер, котики. Вы все мыши. Goog evening, kittens. You are all mices.");

      txSpeak ("<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='EN'>"
// SSML format
            "Goog evening, kittens. You are all mices."
            "</speak>");
      }
```

4.1.8. intptr_t txPlayVideo(int x, int y, int width, int height, const char fileName[], double zoom=0, double gain = 1, HWND wnd = txWindow())

Проигрывает видео.

Аргументы:

x	X-координата верхнего левого угла видео. *См. предупреждение ниже.*
y	Y-координата верхнего левого угла видео. *См. предупреждение ниже.*
width	Ширина видео. Если `0`, то равно ширине окна. *Также см. предупреждение ниже.*
height	Высота видео. Если 0, то равно высоте окна. *Также см. предупреждение ниже.*
fileName	Имя видеофайла (включая расширение) или любого другого источника, с которым работает **VideoLAN** (rtsp://, http:// и т.д.). Если имя — пустая строка (`""`), то проводится только проверка на наличие видеопроигрывателя.
zoom	Масштаб изображения. Необязателен. Если не указан, то равен `0` (Автомасштабирование).
gain	Громкость звука, от `0` до `8` (`800%`). Необязательна. Если не указана, то равна `1` (`100%`).
wnd	Окно, в котором воспроизводится видео. Необязательно. Если не указано, то используется окно **TXLib**.

Возвращает:

Время воспроизведения в миллисекундах (если не указана опция `\a`). Если меньше или равно `0`, то видео не запустилось. :(*См.* коды ошибок ниже.

Если указана опция `\a`, то возвращаемое значение — дескриптор (`HWND`) окна видеопотока *(см. "Асинхронное воспроизведение").*

Возвращаемые значения в случае ошибки:

INT_MIN	Не найден видеопроигрыватель.
INT_MIN+1	Не найден видеофайл.
INT_MIN+2	Внутренняя ошибка регистрации класса окна видеопотока.
INT_MIN+3	Внутренняя ошибка создания окна видеопотока.
INT_MIN+4	Внутренняя ошибка запуска VideoLAN.
Другие отрицательные значения	Код завершения процесса VideoLAN с обратным знаком.

Асинхронное воспроизведение видео

Если в параметре `fileName` самым первым символом поставить символ `\a`, то файл будет воспроизводиться асинхронно: `txPlayVideo()` возвратится сразу после вызова, а воспроизведение видео пойдет параллельно с работой программы. Само имя файла на диске менять не надо. Воспроизведение завершится, когда закончится видеофайл, или когда будет закрыто окно **TXLib**. *См. пример использования.*

Без этой опции `txPlayVideo()` делает паузу в программе, ожидая конца воспроизведения, и только потом возвращается.

В случае асинхронного воспроизведения возвращаемым значением функции `txPlayVideo()` будет не время её работы, а дескриптор окна видеопотока (HWND). С его помощью можно контролировать это окно, его положение на экране, размеры, видимость и другие свойства. *См. функции* `MoveWindow()`, `ShowWindow()` и многие другие в **MSDN**. Гуглите. *Также см. пример использования.*

Досрочно прервать асинхронное воспроизведение можно, уничтожив окно видеопотока с помощью функции `txDestroyWindow()`.

Принудительно завершить асинхронное воспроизведение всех видео можно, сделав вызов `txPlayVideo (NULL)`.

Предупреждения:

Если в последнем параметре `(wnd)` *указано не окно* ***TXLib****, а какое-либо другое, то координаты видео x, y и его размеры* `width, height` *игнорируются. В этом случае видеопоток будет занимать всю клиентскую область окна, указанного параметром wnd. См. пример использования.*

Заметки:

Воспроизведение видео требует установки внешней программы **VideoLAN (vlc.exe)** версии **3.0** или новее. Её можно скачать с официального сайта VideoLAN.org. Без установки **VideoLAN** видео воспроизводиться не будет и выведется сообщение об ошибке.

Если после установки **VideoLAN** программа (`vlc.exe`) все равно не найдена и выводится сообщение об этой ошибке, то установите не 64-битную версию **VideoLAN**, а 32-битную (x86).

См. также:

`Кот Мару!, txPlaySound(), txSpeak(), txMessageBox(), txOutputDebugPrintf()`

Примеры использования:

```
// Кот Мару, 24 000 000+ просмотров. А ты чего добился в жизни? :)

                                        // Cached from www.youtube.com/watch/z_AbfPXTKms
int main()                              // Because Google prevents direct playing from Youtube
{
      txCreateWindow (800, 500);
      txSetFillColor (TX_BLUE);
      txClear();

      txDrawText (0, 0, txGetExtentX(), txGetExtentY(), "Press ESC to stop!");
      txSleep();

      if (txPlayVideo ("") <= 0) return TX_ERROR ("VideoLAN не найден..."), 1;

      txPlayVideo ("\a" MARU_ON_YOUTUBE);

      txPlayVideo (580, 330, 200, 150, MARU_ON_YOUTUBE, 0, 0);

      // Для тех, кто [добился] дождался...

      system ("cmd.exe /c start /min notepad.exe");
      double dt = 0.04 / (txQueryPerformance() + 1);

      HWND notepad = FindWindow ("Notepad", NULL);
      if (!notepad) return 1;

      txPlayVideo ("\a" MARU_ON_YOUTUBE, 0, 1, notepad);

      HWND wnd = (HWND) txPlayVideo (0, 330, 200, 150, "\a" MARU_ON_YOUTUBE, 0, 0);

      for (double t = 0; !GetAsyncKeyState (VK_ESCAPE); t += dt)
```

```
        {
        static const POINT cent = { GetSystemMetrics (SM_CXSCREEN) / 2,
                                    GetSystemMetrics (SM_CYSCREEN) / 2 };
        static const POINT size = { 500, 400 };

        MoveWindow (notepad, cent.x + ROUND (cent.y/2 * cos (t)) - size.x/2,
                     cent.y - ROUND (cent.y/2 * sin (t))-size.y/2, size.x, size.y);

        ShowWindow (notepad, SW_RESTORE);

        MoveWindow (wnd, ROUND (t*100) % 1000 - 200, 330, 200, 150, false);

        Sleep (20);
        }
    }
```

4.1.9. intptr_t txPlayVideo (const char fileName[], double zoom = 0, double gain = 1, HWNDwnd = txWindow())

Проигрывает видео.

Аргументы:

fileName	Имя видеофайла (включая расширение) или любого другого источника, с которым работает **VideoLAN** (rtsp://, http:// и т.д.). Если имя — пустая строка (""), то проводится проверка на наличие видеопроигрывателя.
zoom	Масштаб изображения. Необязателен. Если не указан, то равен **0** (Автомасштабирование).
gain	Громкость звука, от **0** до **8** (**800%**). Необязательна. Если не указана, то равна **1** (**100%**).

wnd	Окно, в котором воспроизводится видео. Необязательно. Если не указано, то используется окно **TXLib**.

Возвращает:

См. в функции `txPlayVideo()` выше.

См. описание в функции `txPlayVideo()` выше.

4.1.10. `int txMessageBox(const char text[] = "Муахахаха! :)", const char header[] = "TXLib сообщает", unsigned flags = MB_ICONINFORMATION|MB_OKCANCEL)`

Выводит сообщение в окне с помощью функции MessageBox.

Аргументы:

text	Текст сообщения. В принципе, необязательно, но зачем вы тогда меня вызывали?
header	Заголовок сообщения. Необязательно
flags	Флаги отображения сообщения. Необязательно.

Возвращает:

Значение, возвращаемое функцией MessageBox.

Предупреждения:

Текст не должен превышать `_TX_BIGBUFSIZE` *символов, а заголовок —* `_TX_BIGBUFSIZE` *символов, иначе они обрезаются.*

Заметки:

Вместо `txMessageBox (text, header, flags)` можно использовать стандартную функцию **Win32** `MessageBox (txWindow(), text, header, flags)`. Отличия `txMessageBox` в том, что она автоматически подставляет окно-родитель, и в том, что при выводе в окно строчки переводятся в формат UNICODE. Это важно лишь в том случае, когда в региональных настройках контрольной панели **Windows** неверно установлена кодовая страница для программ, не поддерживающих UNICODE. В остальных случаях нужды в `txMessageBox` нет.

См. также:

`TX_ERROR()`, `TX_DEBUG_ERROR()`, `txOutputDebugPrintf()`, `txNotifyIcon()`, `txSpeak()`, `txStackBackTrace()`

Примеры использования:

```
if (txMessageBox ("Получилось?", "Прочти меня", MB_YESNO) == IDYES)
      {
      MessageBox (txWindow(), "Хватит и обычного MessageBox()", "Win32 сообщает", 0);
      }
else
      txMessageBox ("Спасаем от кракозябл вместо русских букв, без регистрации и СМС.");
```

4.1.11. `bool txGetAsyncKeyState (int key)`

Проверяет, нажата ли указанная клавиша.

Аргументы:

key	Код (номер) клавиши, как правило, заданный константой (`VK_SPACE`, `VK_LEFT`, `'W'` и т.п.)

Возвращает:

`true` — если указанная клавиша нажата, `false` — если не нажата.

Заметки:

В отличие от оригинальной функции `GetAsyncKeyState()`, возвращает `false`, если окно **TXLib** не активно.

Примеры использования:

```
void PlayBall();

int main()
      {
      txCreateWindow (800, 600);

      PlayBall();
      }
void PlayBall()
      {
      int x  = 100, y = 100;
      int vx = 5,   vy = 7;
      int ax = 0,   ay = 1;
      int dt = 1;

      txSetColor   (TX_LIGHTGREEN);
      txSetFillColor (TX_GREEN);

      while (!txGetAsyncKeyState (VK_ESCAPE)) // Погуглите любой код VK_...,
            {                                 // и найдёте их все
            txCircle (x, y, 20);
```

```
        vx += ax * dt;                          // First velocity, then position
        vy += ay * dt

         x += vx * dt;
         y += vy * dt;

        if (x > 800) { vx = -vx; x = 800; }  // = 800 is not the precise solution
        if (x <   0) { vx = -vx; x =   0; }
        if (y > 600) { vy = -vy; y = 600; }
        if (y <   0) { vy = -vy; y =   0; }

        if (txGetAsyncKeyState (VK_LEFT))  vx--;
        if (txGetAsyncKeyState (VK_RIGHT)) vx++;
        if (txGetAsyncKeyState (VK_UP))    vy--;
        if (txGetAsyncKeyState (VK_DOWN))  vy++;
        if (txGetAsyncKeyState (VK_SPACE))  vx = vy = 0;
        if (txGetAsyncKeyState ('M'))   printf ("Meow ");

        txSleep (20);
        }
    }
```

4.1.12. bool txNotifyIcon (unsigned flags, const char title[], const char format[], ...)

Выводит всплывающее сообщение в системном трее.

Аргументы:

flags	Флаги сообщения.
title	Заголовок сообщения.

format	Строка для печати, как в printf().

Флаги сообщения:

NIIF_INFO	Информация
NIIF_WARNING	Предупреждение
NIIF_ERROR	Сообщение об ошибке

Возвращает:

Удалось ли отобразить сообщение.

Функция формирует сообщение по правилам printf() и выводит во всплывающем окне.

Предупреждения:

- *Эта функция требует, чтобы при компиляции константа версии **Internet Explorer** (_WIN32_IE) была задана не ниже 0x0500. Для этого надо либо включить **TXLib.h** вместо windows.h или перед ним. Либо надо самостоятельно определить (#define) эту константу.*
- *С версией **Internet Explorer** это связано потому, что при его установке в **Windows** обновляются многие компоненты (например, shell32.dll и comctl32.dll), которые влияют на функциональность системы независимо от использования браузера). Сам **Internet Explorer** в отображении сообщения не участвует.*
- *Сообщение не должно превышать _TX_BUFSIZE символов, иначе оно обрезается.*

См. также:

TX_ERROR(), TX_DEBUG_ERROR(), txOutputDebugPrintf(), txMessageBox(), txSpeak()

Примеры использования:

```
   int hours = 3, minutes = 10;
   const char station[] = "Юму";
...
   txNotifyIcon (NIIF_INFO, "Уважаемые пассажиры",
                 "Поезд на %s отправляется в %d:%d.", station, hours, minutes);
```

4.1.13. int txOutputDebugPrintf (const char format[], ...)

Выводит сообщение в отладчике.

Аргументы:

format	Строка для печати, как в `printf()`.

Возвращает:

Количество напечатанных символов.

Функция формирует сообщение по правилам `printf()` и передаёт его в `OutputDebugString()`. Её вывод можно перехватить отладчиком или утилитами-логгерами, например, **DbgView**. Если этого не сделать, и не задать первый символ `'\a'` *(см. ниже)*, то о сообщении никто не узнает. :(

Заметки:

Если первый символ в строке `'\a'`, то сообщение также дублируется `txMessageBox()`.

Если первый или второй символ в строке `'\f'`, то сообщение также дублируется `printf()`.

Предупреждения:

Сообщение не должно превышать `_TX_BIGBUFSIZE` символов, иначе оно обрезается.

См. также:

`TX_ERROR()`, `TX_DEBUG_ERROR()`, `txPrintf()`, `txNotifyIcon()`, `txMessageBox()`, `txStackBackTrace()`

Примеры использования:

```
  int x = 42;
...
  txOutputDebugPrintf ("Никто не узнает, что %d.\n", x);
```

4.1.14. int txPrintf (const char *format, ArgsT...args)

Добрый дядюшка Принтф. Теперь шаболонный.

Аргументы:

format	Строка для печати, все как в вашем любимом printf().
args	Значения для печати.

Возвращает:

Количество напечатанных символов.

Функция работает аналогично `printf()`. Однако, в силу применения сноубордических вариадических шаблонов, она типобезопасна — сама определяет тип печатаемых аргументов и подставляет нужные символы преобразования типов. Допускается применение универсального символа типа % (или % ?) вместо других символов типов, таких как `d`, `i`, `c`, `s`, `p`, `g` и других.

`txPrintf()` позволяет распечатывать типы, не встроенные в язык **C++**: строки **C++** (`std::string`), некоторые типы **Win32** (например, **POINT**) и ваши собственные типы. Для того, чтобы ваши типы могли печататься, надо определить пользовательский оператор вывода в поток STL. Вот, например, как определен оператор вывода для структуры **POINT**:

```
std::ostream& operator << (std::ostream& stream, const POINT& point)
      {
      stream << "{ x: " << point.x << ", y: " << point.y << " }";
      return stream;
      }
```

Вы можете определить свой оператор вывода аналогично.

На самом деле функция `txPrintf()` — это обёртка над операторами потокового вывода STL: `operator <<`. Чудес не бывает, и в любом случае вывод пользовательских типов и автоматический подбор преобразований так или иначе был бы основан на семействе перегруженных функций. Вместо того, чтобы создавать такое семейство с нуля, **TXLib** использует уже существующий оператор `<<`, уже перегруженный для всех стандартных типов, и про который все хорошо знают.

Заметки:

Если нужно передать спецификаторы ширины или точности в виде переменных (для форматов `*`, `%.*` или `*.*`), их нужно оборачивать в конструкции `width()` и `precision()` с помощью соответствующих функций. *См. пример ниже.*

`width_t width (int)` – Функция, оборачивающая спецификатор ширины в тип `enum width_t`.

`precision_t precision (int)` – Функция, оборачивающая спецификатор точности в тип `enum precision_t`.

Предупреждения:

Функция довольно медленная и жадная на ресурсы: во-первых, в силу применения весьма неоптимальных потоков вывода STL (`std::ostream`), во-вторых, вывод сначала идёт в строку (через поток `std::ostringstream`), и только потом печатается.

Если хочется быстрее и экономнее – юзайте обычный принтф, он относительно быстрый. Или, как это принято в **TXLib** – делайте сами.

Идея реализации взята у признанного демона современного **C++** (Alexandrescu A., "Variadic Templates are Funadic", Going Native 2012, Redmond, WA, USA, https://www.youtube.com/watch?v=dD57tJjkumE, краткий обзор докладов см. https://habr.com/ru/post/139064).

Есть также макрос `TX_PRINTF (format, ...)`, который проверяет аргументы согласно спецификации стандартной функции `printf()` и потом вызывает `txPrintf()`. Но он отвергает с `% (%?)`, требует точного соответствия символов преобразований типов и не позволяет передавать типы, не встроенные изначально в язык **C++** (структурные и классовые типы). Используйте его, если в дальнейшем вы планируете перейти на обычный `printf()`.

Предупреждения:

*Эта функция доступна только если ваш компилятор поддерживает стандарт **C++11** (g++ с опцией -std=c++11 или выше, **Microsoft Visual Studio 2013** или выше). Используйте макрос `_TX_CPP11`, который определен, если компилятор поддерживает стандарт **C++11**.*

См. также:

`txFormat()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txOutputDebugPrintf()`, `txNotifyIcon()`, `txMessageBox()`

Примеры использования:

```
//           +--+----+-----+-----+----+------+----+----+
//           |2|  3 |  4  |  5  |  6 |   7  |  8 |    |
//           +--+----+-----+-----+----+------+----+----+
txPrintf    ("|% | %s | %5s | %*d |%.*f | %*.* | %* | %% | \n",       // Arg#1
//           |  |    |     |     |    |      |    |    |
//           +--+----+-----+-----+----+------+----+----+
                                                          12,              // Arg#2
                                                       "Cats power!",          // Arg#3
                                                       std::string ("meow"),   // Arg#4
width (10),                                            56,                     // Arg#5
precision (3),                                         3.141592653,            // Arg#6
width (15),  precision (6),                            2.718281828,            // Arg#7
width (20),                                            POINT { 10, 20 }) ;     // Arg#8
```

4.1.15. `int txPrintf (std::ostringstream &stream, const char *format, ArgsT...args)`

Печатает в строковый поток вывода, как sprintf().

Аргументы:

format	Строка, как в `sprintf()`.
stream	Строковый поток вывода.
args	Значения для печати.

Возвращает:

Количество напечатанных символов.

Предупреждения:

Эта функция доступна только если ваш компилятор поддерживает стандарт ***C++11*** *(g++ с опцией -std=c++11 или выше,* **MSVC 2013** *или выше). Используйте макрос* `_TX_CPP11`*, который определён, если компилятор поддерживает стандарт* ***C++11****.*

См. также:

`txPrintf()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txPrintf()`, `txNotifyIcon()`, `txMessageBox()`, `txStackBackTrace()`

Примеры использования:

См. пример в функции `txPrintf()` *выше.*

4.1.16. `int txPrintf (charbuffer[], size_t size, const char *format, ArgsT...args)`

Печатает в строковый буфер, как sprintf().

Аргументы:

format	Строка, как в `sprintf()`.
buffer	Буфер для вывода.
size	Длина буфера вывода.
args	Значения для печати.

Возвращает:

Количество напечатанных символов.

Предупреждения:

Эта функция доступна только если ваш компилятор поддерживает стандарт ***C++11*** *(g++ с опцией -std=c++11 или выше,* ***MSVC 2013*** *или выше). Используйте макрос* `_TX_CPP11`*, который определён, если компилятор поддерживает стандарт* ***C++11****.*

См. также:

`txPrintf()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txPrintf()`, `txNotifyIcon()`, `txMessageBox()`, `txStackBackTrace()`

Примеры использования:

См. пример в функции `txPrintf()` выше.

4.1.17. `std::string txFormat (const char *format, ArgsT...args)`

Форматирует строку, как `sprintf()`.

Аргументы:

format	Строка, как в `sprintf()`.
args	Значения для печати.

Возвращает:

Отформатированная строка в виде `std::string`.

Предупреждения:

Эта функция доступна только если ваш компилятор поддерживает стандарт ***C++11*** *(g++ с опцией -std=c++11 или выше,* ***MSVC 2013*** *или выше). Используйте макрос* `_TX_CPP11`*, который определён, если компилятор поддерживает стандарт* ***C++11.***

См. также:

`txPrintf()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txPrintf()`, `txNotifyIcon()`, `txMessageBox()`, `txStackBackTrace()`

Примеры использования:

См. пример в функции `txPrintf()` выше.

4.1.18. int random (int range)

Генератор случайных чисел

Аргументы:

rang	Правая граница диапазона (не включая саму границу).

Возвращает:

Случайное целое число в диапазоне [0; range).

Вы ещё помните, что означают разные скобочки в обозначении интервалов? :)

Предупреждения:

Эта функция может мяукать. Just because it can. Потому что она не часть стандарта ***C++*** *или* ***Windows****, а зависит от* ***TXLib'a.*** *Если это вам не нравится, вы можете написать её сами, с помощью стандартной функции* `rand()` *и операции остатка от деления %. Подсказка:* `rand() % range`.

Примеры использования:

```
    char message[100] = "Maybe...";
    sprintf (message, "You SUDDENLY got %d bucks now. But note that tax rate is %d.",
random (100), 100);
    txMessageBox (message, "Lottery");
```

4.1.19. double random (double left, double right)

Генератор случайных чисел

Аргументы:

left	Левая граница диапазона (включая саму границу).
right	Правая граница диапазона (включая саму границу).

Возвращает:

Случайное целое число в диапазоне `[left; right]`.

Вы всё ещё помните, что означают разные скобочки в обозначении интервалов? :)

Предупреждения:

Эта функция может мяукать. Just in case. Потому что она не часть стандарта ***C++*** *или* ***Windows****, а зависит от* ***TXLib'a****. Если это вам не нравится, вы можете написать её сами, с помощью стандартной функции* `rand()` *и небольшой уличной магии с делением на* `RAND_MAX` *и таинственной операцией вычитания.*

Примеры использования:

```
int money = random (-100, +100);
if (money < 0)
      {
      char message[100] = "Maybe...";
      sprintf ("Проиграли в лотерею? Отдайте долг в %d рублей", -money);
      txMessageBox (message, "Быстро!");
      }
```

4.1.20. bool In (Tx x, Ta a, Tb b)

Проверка, находится ли параметр x внутри замкнутого интервала [a; b].

Аргументы:

x	Проверяемый параметр.
a	Левая граница (включительно).
b	Правая граница (включительно).

Возвращает:

Если a <= x && x <= b, то истина, если нет — ложь.

Предупреждения:

Эта функция может мяукать. Because cats are power! Потому что она не часть стандарта ***C++*** *или* ***Windows****, а зависит от* ***TXLib'a.*** *Если это вам не нравится, вы можете написать её сами, используя неравенства.*

Примеры использования:

```
while (txMouseButtons() != 1)
      {
      if (In (txMouseX(), 110, 120)) txTextOut (100, 100, "Meet the wall!");
      txSleep (0);
      }
```

4.1.21. bool In (const POINT &pt, const RECT &rect)

Проверка, находится ли точка pt внутри прямоугольника rect.

Аргументы:

pt	Проверяемая точка в виде `POINT {x, y}`.
rect	Прямоугольник в виде `RECT {left, top, right, bottom}`.

Возвращает:

Результат проверки.

Удобно для реализации экранных кнопок, нажимаемых курсором мыши.

Предупреждения:

Эта функция может мяукать. Because cats are always right. Потому что она не часть стандарта ***C++*** *или* ***Windows****, а зависит от* ***TXLib'a****. Если это вам не нравится, вы можете написать её сами.*

Примеры использования:

```
RECT button = { 100, 100, 150, 120 };

txSetFillColor (TX_LIGHTGRAY);
txRectangle (button.left, button.top, button.right, button.bottom);

txSetTextAlign();
txSetFillColor (TX_WHITE);
txTextOut (125, 115, "Cookie");

for (;;)
      {
      if (In (txMousePos(), button))
```

```
        {
        txSetFillColor (TX_TRANSPARENT);
        txRectangle (button.left, button.top, button.right, button.bottom);

        if (txMouseButtons())
              {
              txSetFillColor (TX_DARKGRAY);
              txRectangle (button.left, button.top, button.right, button.bottom);

              txSetFillColor (TX_WHITE);
              txTextOut (125, 115, "You got cookie");

              break;
              }
        }

    txSleep (0);
    }
```

4.1.22. void tx_fpreset ()

Переинициализирует математический сопроцессор

Сбрасывает состояние математического сопроцессора, вызывая `_fpreset()` и разрешая генерацию исключений сопроцессора для неверного результата (`_EM_INVALID`), денормализации (`_EM_DENORMAL`), деления на ноль (`_EM_ZERODIVIDE`), переполнения (`_EM_OVERFLOW`) и антипереполнения (`_EM_UNDERFLOW`). Обычный вызов `_fpreset()` эти исключения маскирует, в результате чего вычислительные ошибки могут оказаться незамеченными.

Если вы хотите замаскировать эти исключения, вызывайте обычный `_fpreset()`, и затем проверяйте результат вычислений на достоверность хотя бы с помощью `std::isfinite (x)`.

Заметки:

Поведение **TXLib** по умолчанию — генерация этих исключений и их перехват TXLib'ом в виде ошибки.

См. [1] пример работы с этими исключениями, [2] статью о них.

Примеры использования:

```
tx_fpreset();
```

4.1.23. double txSqr (double x)

Очень удобное возведение числа в квадрат.

Аргументы:

x	Число для возведения в него. (Кого? (Who?))

Возвращает:

Квадрат, полученный путём возведения в него числа, заданного для возведения в квадрат.

Заметки:

Это пример, как не надо писать код: `txSqr()` — функция с *"медвежьей услугой"*. Иногда встречаются те, кто любит печатать в функции результат её вычислений (не данные для отладки, а именно результат), вместо того, чтобы просто возвра-

щать его туда, где эту функцию вызывали. Пусть эти люди воспользуются приведённой `txSqr()` для какого-нибудь нужного дела, особенно в цикле. Пример, конечно, несколько преувеличен. *См. в* исходном тексте код этой навязчивой радости.

Примеры использования:

```
printf ("\n" "Радиус\t\t" "Площадь круга\n\n");

for (double r = 100; r > 0; r--)
{
printf ("%6.2lf...", r);

double square = M_PI * txSqr (r); // Надолго запомним эту площадь!
printf ("\b\b\b   \t");

printf ("%-.2lf\n", square);
}
```

4.1.24. void txDump (const void *address, const charname[] = "_txDump()", bool pause = true)

Распечатывает дамп области памяти в консоли.

Аргументы:

address	Адрес начала распечатки.
pause	Делать ли паузу в конце распечатки. Необязательно.
name	Название распечатки (усекается до 8 символов). Необязательно.

Заметки:

Распечатывается область памяти размером 256 байт.

См. также:

`txStackBackTrace()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`

Примеры использования:

```
const char text[] = "Каждому лектору – в портфель по вектору";
txDump (text);

SetConsoleOutputCP (437); // Будет отображаться псевдографика, но НЕ русские буквы
txDump (text);
SetConsoleOutputCP (1251);
```

4.1.25. std::string txDemangle (const char *mangledName)

Преобразует декорированное имя **C++** в название типа.

Аргументы:

mangledName	Декорированное (mangled) имя.

Возвращает:

Строка с полным названием типа.

Что такое декорирование имён (name mangling) см. здесь.

Предупреждения:

Если используется форма функции, возвращающая `char*`, вы должны сами освободить память, занимаемую строкой, с помощью вызова `free()`, иначе будет утечка памяти.

См. также:

`txDump()`, `txStackBackTrace()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`

Примеры использования:

```
auto type = txDemangle (typeid (std::string).name());
std::cout << "The real type of std::string is: " << type << ", muahhaha! :)\n";

std::cout << "Call is shorter, but result is the same: " << txTypename (std::string)
<< ", muahhaha, too.\n";
```

4.1.26. `int txRegQuery (const char *keyName, const char *valueName, void *value, size_t szValue)`

Читает информацию из реестра **Windows**.

Аргументы:

keyName	Имя ключа реестра (см. ниже)
valueName	Имя параметра ключа
value	Буфер, в который записывается значение
szValue	Размер буфера

Возвращает:

Количество байт, записанных в буфер со смыслом со значением `value`

Заметки:

Имя ключа реестра `keyName` обязательно должно начинаться с имени или обозначения одного из разделов:

Обозначение	Ключ реестра
HKLM	HKEY_LOCAL_MACHINE
HKCU	HKEY_CURRENT_USER
HKCR	HKEY_CLASSES_ROOT
HKU	HKEY_USERS
HKCC	HKEY_CURRENT_CONFIG

Примеры использования:

```
char path[MAX_PATH] = "(not installed)";
txRegQuery ("HKCU\\Software\\TX Library", "ProductDir", path, sizeof (path));

printf ("TX Library is installed in: \"%s\"\n", path);
```

4.1.27. WNDPROC txSetWindowsHook (WNDPROCwndProc = NULL)

Устанавливает альтернативную функцию обработки оконных сообщений **Windows** (оконную функцию) для окна **TXLib**.

Аргументы:

wndProc	Новая оконная функция окна **TXLib**.

Если не указана или `NULL`, то текущая альтернативная оконная функция удаляется и устанавливается стандартная.

Возвращает:

Адрес предыдущей оконной функции для окна **TXLib**.

Используйте эту функцию, чтобы самому обрабатывать сообщения **Windows**, например, сообщения от колесика Мыши. *См. примеры* такой обработки в примере использования ниже.

Заданная оконная функция `wndProc` будет вызываться до обработки события средствами **TXLib**. Она должна быть функцией со следующим прототипом:

```
LRESULT CALLBACK NewWndProc (HWND window, UINT message, WPARAM wParam, LPARAM lParam);
```

Предупреждения:

Оконная функция будет вызываться из вспомогательного (второго) потока, создаваемого `txCreateWindow()`. *Это не тот же самый поток, в котором выполняется* `main()`. *В связи с этим будьте внимательны при работе с глобальными переменными или их аналогами, т.к. может возникнуть "гонка потоков" (race condition).*

Если оконная функция вернёт значение, не равное 0, то стандартная обработка сообщений средствами ***TXLib*** *не будет произведена. Из-за этого, возможно, окно даже не сможет нормально закрыться. Придётся завершать программу с помощью Alt-Ctrl-Del из диспетчера задач, или из более продвинутого диспетчера* ***Process Explorer****. Если Вы берёте на себя обработку оконных сообщений, делайте её по правилам* **Win32** *(см. MSDN), включая вызов* `DefWindowProc()`.

Заметки:

Полностью поменять оконную функцию можно с помощью функций `SetWindowLong` или `SetWindowLongPtr`:

```
WNDPROC OldWndProc = (WNDPROC) SetWindowLongPtr (txWindow(), GWL_WNDPROC, (LONG_PTR)
NewWndProc);
```

При этом надо обязательно всегда вызывать старую оконную функцию с помощью `CallWindowProc`, (*см.* MSDN), иначе последствия будут такими же плачевными, как описаны выше.

См. также:

`txCreateWindow(), txDialog, txInputBox()`

Примеры использования:

```
LRESULT CALLBACK MyWndProc (HWND wnd, UINT msg, WPARAM wParam, LPARAM lParam);
void SetColors  (COLORREF lineColor, COLORREF fillColor, int op);
void DrawCursor (POINT pos, int size);
void DrawShoot  (POINT pos, int size);

int main()
      {
      _txWindowStyle |= WS_THICKFRAME;
      txCreateWindow (GetSystemMetrics (SM_CXSCREEN) / 4, GetSystemMetrics (SM_CYSCREEN) / 4);

      txSetWindowsHook (MyWndProc);

      POINT sz = txGetExtent();

      txSelectFont ("Lucida Console", 30, 15);
      txDrawText   (0,    0,  sz.x,   sz.y, "MOV txWindow, eax [please]");

      txSelectFont ("Lucida Console", 15, 7.5);
      txDrawTex    (0, sz.y/2, sz.x,   sz.y, "(Info for the cats: NO MOUSE HERE)");

return 0;
}
```

```
LRESULT CALLBACK MyWndProc (HWND wnd, UINT msg, WPARAM wParam, LPARAM lParam)
      {
      static bool wheelMsg     = false;
      static POINT pos         = {-10, -10};
      const int size           = 20;

      if (message == WM_MOUSEMOVE)
            {
            DrawCursor (pos,                                        size);
            DrawCursor (pos = {LOWORD (lParam), HIWORD (lParam)}, size);

            txUpdateWindow();
            }

      if (message == WM_LBUTTONDOWN)
            {
            DrawCursor (pos, size);
            DrawShoot  (pos, size);
            DrawCursor (pos, size);

            txUpdateWindow();
      txPlaySound ("C:\\Windows\\Media\\ir_begin.wav");
      }

if (message == WM_MOUSEWHEEL)
      {
      int wheel = (short) HIWORD (wParam) / WHEEL_DELTA;

      RECT rect = {}; GetWindowRect (window, &rect);
```

```
      POINT sz = { rect.right - rect.left, rect.bottom - rect.top };

      wheelMsg = true;
      MoveWindow (window, rect.left - wheel, rect.top - wheel,
                  sz.x + wheel*2, sz.y + wheel*2, true);
      wheelMsg = false;
      }

if (message == WM_MOVE && !wheelMsg)
      {
      txMessageBox (" I like to MOVe it, MOVe it", "TXLib 2 Real",
                              MB_ICONINFORMATION);
      wheelMsg = false;
      }

if (message == WM_SETCURSOR && LOWORD (lParam) == HTCLIENT)
      {
      SetCursor (NULL);  // Hide the mouse cursor
      return true;       // Не продолжать обработку сообщения средствами TXLib
      }

static int i = 0;
if (i++ % 15 == 0)
{
char str[2] = {"-\\|/" [i/15 % 4]};     // Прропппелллллерррр
SetWindowText (txWindow(), str);
}

return false;      // Продолжить обработку сообщения средствами TXLib
}
```

```
void DrawCursor (POINT pos, int size)
      {
      SetColors (TX_YELLOW, TX_TRANSPARENT, R2_XORPEN);

      txCircle (pos.x,      pos.y,      size/2);
      txLine (pos.x-size, pos.y,        pos.x+size, pos.y);
      txLine (pos.x,        pos.y-size, pos.x,      pos.y+size);

      Win32::SetROP2 (txDC(), R2_COPYPEN);
      }
void DrawShoot (POINT pos, int size)
      {
      SetColors (RGB (255, 64, 0), RGB (255, 64, 0), R2_COPYPEN);

      for (int i = 0; i < 5; i++)
            txCircle   (pos.x + rand() % (size*2) - size,
                        pos.y + rand() % (size*2) - size, rand() % (size/2) + size/2);
      }
void SetColors (COLORREF lineColor, COLORREF fillColor, int op)
      {
      txSetColor     (lineColor);
      txSetFillColor  (fillColor);
      Win32::SetROP2  (txDC(), op);
}
```

4.1.28. bool txLock (bool wait = true)

Блокировка холста (контекста рисования).

Аргументы:

wait	Ожидать конца перерисовки окна вспомогательным потоком. Если не указано, то "ожидать".

Возвращает:

Состояние блокировки.

Перед вызовом любых функций **Win32** GDI необходимо заблокировать холст функцией `txLock()` и затем разблокировать с помощью `txUnlock()`. Это связано с тем, что обновление содержимого окна (для тех, кто знает **Win32**: обработка сообщения `WM_PAINT`) в библиотеке **TXLib** происходит в отдельном вспомогательном потоке. Надолго блокировать его нельзя — при заблокированном потоке окно не обновляется.

`txLock()` использует `EnterCriticalSection()`, и физически приостанавливает поток, обновляющий окно, так что надолго блокировать нельзя. Иначе тормозится обработка оконных сообщений, окно перестаёт реагировать на действия пользователя и перерисовываться. Нельзя также вызывать `txSleep()` или `Sleep()` при заблокированном потоке.

`txLock()` / `txUnlock()` — это низкоуровневый механизм. Он отличается от более простого высокоуровневого механизма `txBegin()` / `txEnd()` / `txUpdateWindow()`, который не приостанавливает поток, а просто отключает принудительное постоянное обновление окна.

См. также:

`txDC()`, `txVideoMemory()`, `txLock()`, `txUnlock()`, `txGDI()`

Примеры использования:

См. исходный текст функций `_txCanvas_OnPAINT()` и `_txConsole_Draw()` в **TXLib.h**.

4.1.29. bool txUnlock ()

Разблокировка холста

Возвращает:

Состояние блокировки (всегда `false`).

Более подробно см. в описании `txLock()`.

См. также:

`txDC()`, `txVideoMemory()`, `txLock()`, `txGDI()`

Примеры использования:

См. исходный текст функций `_txCanvas_OnPAINT()` и `_txConsole_Draw()` в **TXLib.h**.

4.1.30. int txSetLocale (int codepage = _TX_CODEPAGE, const char locale[] = _TX_LOCALE, const wchar_t wLocale[] = _TX_WLOCALE)

Смена кодовой страницы консоли и локали стандартной библиотеки **C++**.

Аргументы:

codepage	Номер новой кодовой страницы консоли. Если не указано, то 1251.
locale	Новая локаль (информация о языке) стандартной библиотеки **C++**. Если не указано, то "Russian" или "ru_RU.CP1251".
wLocale	Новая локаль стандартной библиотеки **C++** для wide character funstions. Если не указано, то L"`Russian_Russia.ACP`".

Возвращает:

Номер старой кодовой страницы консоли.

Список кодовых страниц консоли

Список имен локалей **C++**

Предупреждения:

Если устанавливается кодовая страница не 1251, возможны нарушения в работе строковых функций **C++** и вывода текста в консоль.

См. также:

`txClearConsole()`, `txSetConsoleAttr()` и другие консольные функции, `txReopenStdio()`

Примеры использования:

```
txSetLocale (866, "Russian_Russia.866", L"Russian_Russia.866");
```

4.2. Макросы

4.2.1. #define CALLOC(type, size)

Выделяет блок динамической памяти через calloc с автоматическим преобразованием типа указателя.

Аргументы:

type	Тип данных, размещаемый в памяти
size	Количество элементов типа type

Возвращает:

Выделенная память

Примеры использования:

```
   char* str = CALLOC (char, 100);
 ...
   FREE (str);
```

4.2.2. #define FREE(ptr)

Освобождает динамическую память и обнуляет указатель на неё.

Аргументы:

ptr	Указатель на освобождаемую память. После освобождения памяти он обнуляется.

Примеры использования:

```
   char* str = CALLOC (char, 100);
...
   FREE (str);
```

4.2.3. #define __TX_FUNCTION__

Имя текущей функции

Предупреждения:

Если определение имени функции не поддерживается компилятором, то `__TX_FUNCTION__` *раскрывается в имя исходного файла и номер строки.*

4.2.4. #define sizearr(arr)

Вычисление размера массива в элементах

Аргументы:

arr	Имя массива.

Возвращает:

Размер массива в элементах (не в байтах).

Макрос `sizearr()` вычисляет размер массива в элементах, проверяя, можно ли его правильно вычислить при компиляции.

Макрос `SIZEARR()` просто делит размер всего массива в байтах на размер его элемента, получается размер массива в элементах.

Он не проверяет, можно ли его правильно вычислить, и при неправильном использовании может выдать неверный размер.

Предупреждения:

`SIZEARR()` *выдаёт неверный размер, если определение массива вместе с его размером, известным при компиляции, недоступно в месте использования* `SIZEARR()`. *См. пример ниже.*

Примеры использования:

```
void test()
      {
      // Размер этого массива, хоть и не указан, но может быть автоматически определён
      // компилятором при компиляции программы. Он равен 4 (четыре структуры POINT).
```

```
      POINT coord[] = { {110, 110}, {120, 120}, {130, 110}, {140, 120} };

      // Здесь размер массива известен при компиляции, т.к. он был определён тут.

      for (int i = 0; i < sizearr (coord) - 1; i++)
            txLine (coord[i].x, coord[i].y, coord[i+1].x, coord[i+1].y);

      DrawLines1 (coord);                    // Попытка передать массив без размера.
      DrawLines2 (coord, sizearr (coord));   // Правильная передача размера массива.

      DrawLines3 (coord);                    // В принципе, можно и так.
      }

// Функции DrawLines1 и DrawLines2 определены так:

void DrawLines1 (const POINT coord[])
      {
      // Массивы в Си передаются как указатели на начало массива. Поэтому:
      // 1) sizearr здесь выдаст ошибку компиляции, и ее легко будет найти.
      // 2) SIZEARR здесь неверно посчитает размер, что намного хуже
      //     т.к. он будет равен sizeof (POINT*) / sizeof (POINT) == 4/8 == 0
      //     что станет трудноуловимой логической ошибкой времени исполнения.

      for (int i = 0; i < sizearr (coord) - 1; i++)
            txLine (coord[i].x, coord[i].y, coord[i+1].x, coord[i+1].y);
      }

void DrawLines2 (const POINT coord[], int n)
      {
      // Здесь размер приходит как параметр n, так что всё просто. Функция
      // test) правильно вычисляет его и передаёт функции DrawLines2
```

```
      for (int i = 0; i < n - 1; i++)
      txLine (coord[i].x, coord[i].y, coord[i+1].x, coord[i+1].y);
      }

// HIC SVNT TEMPLATES

template <int size>
void DrawLines3 (const POINT (&coord) [size])
      {
      for (int i = 0; i < size - 1; i++)
      txLine (coord[i].x, coord[i].y, coord[i+1].x, coord[i+1].y);
      }
```

4.2.5. #define MAX(a, b)

Возвращает максимальное из двух чисел

Аргументы:

a	Одно из чисел :)
b	Другое из чисел :)

Возвращает:

Максимальное из двух чисел *a* и *b*.

Предупреждения:

Это макрос, и аргументы *a* и *b* могут вычисляться в нем два раза. Поэтому не используйте в нём побочных действий ++, --, += и т.п. Например, не пишите так: `int m = MAX (++x, y += 2);`

См. также:

`MIN()`

Примеры использования:

```
if (MAX (3, 7) != 7) printf ("Your CPU is broken, throw it away.");
```

4.2.6. #define MIN(a, b)

Возвращает минимальное из двух чисел

Аргументы:

a	Одно из чисел :)
b	Другое из чисел :)

Возвращает:

Минимальное из двух чисел *a* и *b*.

Предупреждения:

Это макрос, и аргументы *a* и *b* могут вычисляться в нём два раза. Поэтому не используйте в нём побочных действий ++, --, += и т.п. Например, не пишите так: `int m = MIN (x--, y /= 2);`

См. также:

`MAX()`

Примеры использования:

```
if (MIN (3, 7) != 3) printf ("Your CPU is still broken, throw it away twice.");
```

4.2.7. #define ROUND(x)

Округляет число до целого

Аргументы:

x	Число.

Возвращает:

Округлённое число, преобразованное в тип `long`.

Примеры использования:

```
double weight = 5.5;            // 5.5 kilos is the weight of Maru the Cat in 2012.
long Maru = ROUND (weight);     // Should use ROUND() because Maru is so round
```

4.2.8. #define _TX_DESTROY_3D

Ну просто очень удобный макрос.

Предупреждения:

Это ещё один пример, как не надо писать код.

Макрос определён так:

```
#define z 0
```

Замечательный макрос! Теперь на одну переменную в программе стало меньше. :((

Заметки:

(Используйте `#undef`. С <http://www.google.ru/search?q=%23undef> `#undef` ваша программа станет мягкой и шелковистой.)

Примеры использования:

```
#define _TX_DESTROY_3D
#include "TXLib.h"

// А теперь попробуйте объявить переменную z для расчета 3-D координат x, y, z:

int z = 0; // Да! TXLib уничтожает трехмерность! Очень круто. :=/
```

4.2.9. #define please

Ещё парочка макросов.

`please` увеличивает вероятность успешного выполнения кода*.

`meow` —...просто мяу :)

Заметки:

Это шутка :)

Примеры использования:

```
#include "TXLib.h"

int x = rand() meow // Как же без котиков?
int y = rand() meow
if (x+y > RAMD_MAX/2) please x = y = 0;
```

4.2.10. #define ZERO(type)

Обнулитель типов, не имеющих конструкторов

Аргументы:

type	Имя типа.

Возвращает:

Значение типа `type`, покомпонентно инициализированное по умолчанию (для встроенных типов — нулем).

Примеры использования:

```
  void f (POINT p);
...
  POINT z = {}; f (z);      // Так без ZERO
  f (ZERO (POINT));         // Так с ZERO
```

4.2.11. #define tx_auto_func(func)

Автоматический вызов функции при завершении другой функции (аналог __finally)

Аргументы:

func	Тело автоматически вызываемой функции (фигурные скобки не обязательны).

Заметки:

Все переменные вызываемой функции связываются с переменными внешней функции по ссылке.

Их названия и типы не указываются. Указывается только тело вызываемой функции.

Синоним: `tx_finally`

См. также:

`txAutoLock`

Примеры использования:

```
void some_func()
      {
      int x = 1;
      tx_auto_func ((x));                    // Will be printed at return

      FILE* f = fopen (__FILE__".o.txt", "w");
      tx_auto_func (fclose (f));             // Will be closed at return

      fprintf (f, "start: x = %d\n", x);     // Do some job
      x = 2;                                 // Do some job
      }
```

4.2.12. #define TX_ASSERT(cond)

Замена стандартного макроса **assert()**, с выдачей сообщения через **txMessageBox()**, консоль и **OutputDebugString()**.

Аргументы:

cond	Условие для проверки.

Возвращает:

Не определено.

Если условие, проверяемое **TX_ASSERT()**, истинно, то макрос ничего не делает.

Если условие оказывается ложно, то выводится диагностическое сообщение и программа аварийно завершается.

Предупреждения:

При компиляции в режиме `Release` (или если определён `NDEBUG`) `TX_ASSERT` превращается в пустой оператор.

Не надо помещать в `TX_ASSERT()` или в `assert()` действия, которые важны для работы алгоритма!

Заметки:

Если условие cond может быть вычислено уже во время компиляции как ложное, компилятор может предупредить об этом (как о делении на `0`).

See: "JPL Institutional Coding Standard for the C Programming Language", Jet Propulsion Laboratory, California Institute of Technology, JPL DOCID D-60411, Ver. 1.0, March 3, 2009, page 15.

Стандартный макрос `assert()` переопределяется так, что становится эквивалентен `TX_ASSERT()`.

Предупреждения:

Кто не юзает `assert()`*, тот ест баги на десерт!*

См. также:

`asserted`, `verified`, `verify()`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txOutputDebugPrintf()`, `txMessageBox()`, `txNotifyIcon()`, `txStackBackTrace()`, `__TX_FILELINE__`, `__TX_FUNCTION__`

Примеры использования:

```
TX_ASSERT   (0 <= i && i < ARRAY_SIZE);
assert      (0 <= i && i < ARRAY_SIZE);     // То же самое, что и TX_ASSERT

FILE* input = fopen ("a.txt", "r");
TX_ASSERT (input);
assert      (input);                        // То же самое, что и TX_ASSERT

// Этот вызов fgets() НЕ будет выполнен в режиме Release:
assert (fgets (str, sizeof (str) - 1, input));

// Здесь все будет правильно:
bool ok = (fclose (input) == 0);
TX_ASSERT   (ok);
assert      (ok);                           // То же самое, что и TX_ASSERT
```

4.2.13. #define asserted

Выводит диагностическое сообщение в случае нулевого или ложного результата.

4.2.14. Возвращает:

Всегда **0**.

Суффиксная форма макроса **assert()**, не теряющая в режиме **Release** исполнения предиката.

4.2.15. Заметки:

Предполагается, что операция в случае неуспеха возвращает **0** или **false**.

При компиляции в режиме **Release** (или если определён **NDEBUG**) asserted превращается в пустое место.

См. также:

`assert()`, `verify()`, `verified`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txOutputDebugPrintf()`, `txMessageBox()`, `txNotifyIcon()`, `txStackBackTrace()`, `__TX_FILELINE__`, `__TX_FUNCTION__`

Примеры использования:

```
FILE* input = fopen ("a.txt", "r"); assert (input);

                                    // Этот вызов fgets() будет выполнен в любом случае:
fgets (str, sizeof (str) - 1, input) asserted;

                                    // Этот вызов fgets() НЕ будет выполнен в режиме Release:
assert (fgets (str, sizeof (str) - 1, input));

(fclose (input) != 0) asserted;
```

4.2.16. #define verify

Выполняет команду (вычисляет выражение) и проверяет результат.

Аргументы:

expr	Команда (выражение)

Возвращает:

1, если выражение `expr` истинно, иначе `0`.

Если условие, проверяемое `verify()`, истинно, то макрос ничего не делает.

Если условие оказывается ложно, то выводится диагностическое сообщение и программа аварийно завершается.

Заметки:

Действие макроса аналогично `assert()`, но при компиляции в режиме `Release` (или если определён `NDEBUG`) `verify` не превращается в пустой оператор.

См. также:

`verified`, `assert()`, `asserted`, `TX_ERROR()`, `TX_DEBUG_ERROR()`, `txOutputDebugPrintf()`, `txMessageBox()`, `txNotifyIcon()`, `txStackBackTrace()`, `__TX_FILELINE__`, `__TX_FUNCTION__`

Примеры использования:

```
FILE* input = verify (fopen ("a.txt", "r"));

// Этот вызов fgets() БУДЕТ выполнен в режиме Release:
verify (fgets (str, sizeof (str) - 1, input));

// Здесь все тоже будет правильно:
verify (fclose (input) == 0);
```

4.2.17. `#define TX_ERROR(msg)`

Выводит развернутое диагностическое сообщение.

Аргументы:

msg	Сообщение с произвольным количеством параметров в стиле функции `printf()`.

Заметки:

Этот макрос может распечатывать стек вызовов функций в консоли. По этому поводу *см. замечания* к функции `txStackBackTrace()` и раздел Установка библиотеки, п.4.

Возвращает:

Всегда `false`.

См. также:

`_`, `TX_COMMA`, `assert()`, `asserted`, `verify()`, `verified`, `TX_DEBUG_ERROR()`, `txOutputDebugPrintf()`, `txMessageBox()`, `txNotifyIcon()`, `txStackBackTrace()`, `__TX_FILELINE__`, `__TX_FUNCTION__`

Примеры использования:

```
TX_ERROR ("Не смог прочитать 'Войну и мир'. Отмазка %d: не нашел '%s'", reasonNum, fileName);
```

4.2.18. #define TX_DEBUG_ERROR(...)

Выводит развернутое диагностическое сообщение в отладочном режиме.

Описание см. в `TX_ERROR`.

Заметки:

В режиме `Release` этот макрос не выводит ничего.

См. также:

`_`, `TX_COMMA`, `assert()`, `asserted`, `verify()`, `verified`, `TX_ERROR()`, `txOutputDebugPrintf()`, `txMessageBox()`, `txNotifyIcon()`, `txStackBackTrace()`, `__TX_FILELINE__`, `__TX_FUNCTION__`

Примеры использования:

```
TX_DEBUG_ERROR ("Так и не смог прочитать 'Войну и мир'."
                "Отмазка %d: потерял '%s'", reasonNum, fileName);
```

4.2.19. #define txStackBackTrace()

Распечатывает текущий стек вызовов функций в консоли.

Предупреждения:

Для корректной работы этой функкции требуются модули, которые нужно установить (скопировать) в папку **Windows**. *См. раздел* Установка библиотеки, п.4.

Заметки:

Для наиболее корректной работы этой функции полностью отключайте оптимизацию при компиляции. Например, для компилятора **GCC g++** — с помощью ключа командной строки **-O0**. Разные среды программирования позволяют задать эти ключи по-разному, например, в **CodeBlocks** через ***Главное меню — Settings — Compiler — (Global Compiler Settings) — (Compiler Settings) — Other Options***.

См. также:

`txDump(), TX_ERROR(), TX_DEBUG_ERROR()`

Примеры использования:

```
void Recursion()
      {
      txStackBackTrace();

      printf ("Press any key...\n");
      _getch();

      Recursion();
}
```

4.2.20. #define _ ,

Макрос, позволяющий передать переменное число параметров в какой-либо другой макрос.

Заметки:

Символ подчеркивания и символ `TX_COMMA` просто переопределяются в запятую.

См. также:

`TX_ERROR(), TX_DEBUG_ERROR()`

Примеры использования:

```
TX_ERROR ("Слишком умный абзац: роман 'Война и мир', файл '%s', строка %d" _ fileName _ lineNum);
```

4.2.21. #define txGDI(command, dc)

Вызов функции **Win32 GDI** с автоматической блокировкой и разблокировкой.

Аргументы:

command	Функция GDI (возможно, возвращающая значение).
dc	Дескриптор контекста рисования (холста), использующийся в вызове функции GDI (*см. параметр* `command`).

Возвращает:

Значение, возвращаемое вызываемой функцией **GDI**.

Заметки:

Если параметр `dc` соответствует основному холсту **TXLib** (совпадает с возвращаемым значением `txDC()`), то на время выполнения функции **GDI** поток, обновляющий окно **TXLib**, блокируется.

Если в вызове функции GDI используются запятые, то используйте двойные скобки, чтобы получился один параметр, так как `txGDI()` это всё же макрос.

См. также:

`txDC()`, `txVideoMemory()`, `txLock()`, `txUnlock()`

Примеры использования:

```
txGDI (( Rectangle (txDC(), x1, y1, x2, y2) )); // Не забудьте про две скобки
```

4.2.22. #define TX_BEGIN_MESSAGE_MAP()

Заголовок карты сообщений (`Message Map`).

Раскрывается в

```
virtual int dialogProc (HWND _wnd, UINT _msg, WPARAM _wParam, LPARAM _lParam) override
      {
      int _result = txDialog::dialogProc (_wnd, _msg, _wParam, _lParam);

      switch (_msg)
            {
```

См. также:

`TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_HANDLE()`, `TX_COMMAND_MAP`, `txDialog::dialogProc()`, `txDialog`

Примеры использования:

См. реализацию функции `txInputBox()`.

`#define TX_HANDLE(id)`

Заголовок обработчика сообщения (Message handler) карты сообщений.

Аргументы:

id	Идентификатор сообщения.

Раскрывается в

```
break;
case (id):
```

См. также:

`TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_HANDLE()`, `TX_COMMAND_MAP`, `txDialog::dialogProc()`, `txDialog`

Примеры использования:

См. реализацию функции `txInputBox()`.

4.2.23. #define TX_COMMAND_MAP

Начало карты команд (`Command map`) в карте сообщений.

Раскрывается в

```
      } // Конец switch (_msg)

if (_msg == WM_COMMAND)
      switch (LOWORD (_wParam))
            {
```

См. также:

`TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_HANDLE()`, `TX_COMMAND_MAP`, `txDialog::dialogProc()`, `txDialog`

Примеры использования:

См. реализацию функции **txInputBox()**.

4.2.24. #define TX_END_MESSAGE_MAP

Завершитель карты сообщений.

Раскрывается в

```
      } // Конец switch (_msg) или switch (LOWORD (_wParam))

return FALSE;
}
```

См. также:

TX_BEGIN_MESSAGE_MAP(), TX_END_MESSAGE_MAP, TX_HANDLE(), TX_COMMAND_MAP, txDialog::dialogProc(), txDialog

Примеры использования:

См. реализацию функции txInputBox().

4.2.25. #define __TX_DEBUG_MACROS ("Группа отладочных -макросов")

Отладочная печать переменной во время вычисления выражения или участка кода во время его выполнения.

Сделай приятными твои круглые сутки отладки!

Предупреждения:

Эти макросы могут измениться в будущих версиях. Чтобы вам повеселее жилось.

Назначение:

(var, [name])	Печать имени и значения переменной или выражения **var**. [**name**] — необязательное примечание.
_(var, [name])	То же, что и, но без новой строки. [**name**] — необязательное примечание.

x (var, [name])	Печать имени и значения переменной или выражения **var** в 16-ричной системе счисления. [**name**] — необязательное примечание.
x_(var, [name])	То же, что и **x(var)**, но без новой строки. [**name**] — необязательное примечание.

v (var, cond, [name])	То же, что и, но различным цветом в зависимости от условия **cond**. [**name**] — необязательное примечание.
v_(var, cond, [name])	То же, что и **v(var)**, но без новой строки. [**name**] — необязательное примечание.

v (var, cond, [name])	То же, что и, но различным цветом в зависимости от условия **cond**. [**name**] — необязательное примечание.
v_(var, cond, [name])	То же, что и v(var), но без новой строки. [**name**] — необязательное примечание.

do(code)	Печать строки кода, затем выполнение этого кода.
DO(code)	То же, что и **do(code)**, но с паузой (нажмите любую клавишу).
Do(code)	То же, что и **do(code)**, но с паузой (**txMessageBox**).

(expr)	Печать выражения, его вычисление, печать и возврат значения. Если выражение содержит оператор "запятая", не взятый в скобки, необходимо окружать `expr` ещё одной парой скобок.
_(expr)	То же, что и, но вторая печать идёт без новой строки.

(code)	То же, что и, но для операторов или блоков кода (без возврата значения).
_(code)	То же, что и, но вторая печать идёт без новой строки.

	Печать местоположения в коде.
_	Печать местоположения в коде (только имя функции и номер строки).
meow (msg)	То же, что и, но ещё и печатает сообщение `msg`.

test (cond)	Печать результата теста различным цветом в зависимости от условия `cond`.
unittest (code, expected)	Печать результата юнит-теста `code` с ожидаемым результатом `expected`.

n	Перевод строки (печать '\n').
nn	Пустая строка (печать '\n\n')
t	Табуляция (печать '\t').

Установка атрибутов символов консоли:

d	Светло-серый цвет
b	Светло-синий цвет
g	Светло-зелёный цвет
c	Светло-бирюзовый цвет
r	Светло-красный цвет
m	Светло-малиновый цвет
y	Жёлтый цвет
h	Белый цвет

D	Темно-серый цвет
B	Темно-синий цвет
G	Темно-зелёный цвет
C	Темно-бирюзовый цвет
R	Темно-красный цвет
M	Темно-малиновый цвет
Y	Темно-жёлтый цвет
H	Прозрачный цвет

o	OK	Светло-зелёный на зелёном
i	Information	Светло-синий на синем
w	Warning	Светло-малиновый на малиновом

e	Error	Светло-красный на красном
f	Fatal	Чёрный на светло-красном
l	Location	Чёрный на темно-сером

O	OK bold	Жёлтый на зелёном
I	Information bold	Жёлтый на синем
W	Warning bold	Жёлтый на малиновом
E	Error bold	Жёлтый на красном
F	Fatal bold	Малиновый на светло-красном
L	Location bold	Светло-серый на тёмно-сером

T (cond)	Светло-зелёный или светло-красный, в зависимости от условия `cond`.

s	Запомнить атрибуты. При выходе из { блока кода } атрибуты восстанавливаются.
s*	Запомнить атрибуты и установить цвет (замените звёздочку кодом цвета, см. выше). Пример: `sg` — запомнить атрибуты и установить светло-зелёный цвет.

Что такое юнит-тест? А вот что: www.google.com/search?q=Юнит- тестирование+C++.
Это когда ошибки ищутся сами.

См. также:

`assert()`, `asserted`, `__TX_FILELINE__`, `__TX_FUNCTION__`, `TX_ERROR`

Заметки:

Есть еще другие (недокументированные) подобные макросы. Загляните в исходный текст **TXLib.h**, ищите `__TX_DEBUG_MACROS` и *см. ниже.*

Примеры использования:

```
g / green
int x = 5;
int y = (x) + 1;
( x = (y) + 2 );

int xx[] = { 10, 20, 30 };
(xx);

r // red
double xy = ( pow (x, y) );

meow ("Computing hypothenuse...")
double h  = (( (x) = x*x, y = y*y, sqrt ((x+y)) ));

P;
( txCreateWindow (800, 600) );

d // default color
( if ((xy) < (h)) { sE return (h); } );       // Save color, print h in error color

T (h <  10); (h);                 // Print h again, but in success color (h < 10)...
T (h <= 10); (h);                 //...or error color (otherwize)
n                                 // New line
```

```
Do (bool isPositive = (h > 0));
test (isPositive); n                                    // Print a test result

#ifndef _MSC_VER
bool ok = ( unittest (strlen ("abc"), 3) );             // Checks in unit-test style
#endif

unittest (strlen ("abc"), 3);                           // Checks in unit-test style, Microsoft
                                                        // No return result from unittest here
p;
```

4.3. Переменные

4.3.1. const double txPI

Число Пи

Примеры использования:

```
if (txPI == 1)
      xMessageBox ("Вы попали в другую Вселенную.", "Поздравляем", MB_ICONSTOP);
```