HarrixQtLibrary v.3.28

А. Б. Сергиенко

17 июня 2014 г.

Аннотация

Библиотека HarrixQtLibrary — сборник функций для Qt.

Содержание

1 Введение

Библиотека HarrixQtLibrary — это сборник функций для Qt.

Последнюю версию документа можно найти по адресу:

https://github.com/Harrix/HarrixQtLibrary

Об установке библиотеки можно прочитать тут:

http://blog.harrix.org/?p=1186

С автором можно связаться по адресу sergienkoanton@mail.ru или http://vk.com/harrix.

Сайт автора, где публикуются последние новости: http://blog.harrix.org/, а проекты располагаются по адресу http://harrix.org/.

2 Список функций

Обработка данных, считанных их файла

1. **HQt_CountOfColsFromQStringList** — Функция подсчитывает сколько столбцов в текстовом файле, который скопировали в QStringListFromFile.
2. **HQt_CountOfRowsFromQStringList** — Функция подсчитывает сколько строк в текстовом файле, который скопировали в QStringListFromFile. Функция подсчитывает сколько строк в k столбце из текстового файла, который скопировали в QStringListFromFile. Функция подсчитывает сколько строк в каждом столбце из текстового файла с матрицей, который скопировали в QStringListFromFile.
3. **THQt_ReadColFromQStringList** — Функция считывает данные какого-то k столбца с датами из QStringList в виде матрицы. Функция считывает данные какого-то k столбца с датами из QStringList в виде матрицы. Для строк.
4. **THQt_ReadColFromQStringList2** — Функция считывает данные какого-то k столбца из QStringList в виде матрицы.
5. **THQt_ReadMatrixFromQStringList** — Функция считывает данные из QStringList в матрицу.
6. **THQt_ReadTwoVectorFromQStringList** — Функция считывает данные из QStringList в два вектора.
7. **THQt_ReadVectorFromQStringList** — Функция считывает данные из QStringList в вектор.

Получение строк-выводов по данным

1. **HQt_BoolToWord** — Функция переводит переменную, принимающую значения "0"и "1"в слова.
2. **HQt_RandomString** — Функция генерирует случайную строку из английских больших и малых букв.
3. **HQt_TryingReduceString** — Функция старается сократить строку длиной больше MaxSize символов, сокращая слова. Функция старается сократить строку длиной больше 40 символов, сокращая слова.
4. **HQt_UniqueName** — Функция возвращает уникальную строку, которую можно использовать как некий идентификатор. Собирается из "HQt_"+ текущее время или из BeginString+"_"+ текущее время.
5. **HQt_UniqueNameOnlyNumbers** — Функция возвращает уникальную строку, которую можно использовать как некий идентификатор. В строке только цифры. Собирается из текущего времени.
6. **HQt_WriteTime** — Функция переводит миллисекунды в строку с описанием сколько это минут, секунд и др.

Работа с алфавитом и переносами

1. **HQt_BreakdownOfTextWithWordWrap** — Функция разбивает текст на строки длиной не более length. Если может, то ставит переносы.

2. **HQt_CheckIntolerablePunctuation** — Является ли символом знаком пунктуации, который нельзя переносить.

3. **HQt_CheckLetterFromWord** — Является ли буква символом из слова. Считается, что это или латинские буквы, или русские, или цифры или нижнее подчеркивание. Плюс некоторые знаки препинания, так как их нельзя переносить.

4. **HQt_CheckRus** — Функция проверяет наличие русских букв в строке.

5. **HQt_CutToWords** — Функция разбивает строку на слова и те, части, между которыми они находятся. Слова с дефисом считаются за несколько слов.

6. **HQt_CutToWordsWithWordWrap** — Функция разбивает строку на слова и те, части, между которыми они находятся. А русские и английские слова разделяет по переносам. Слова с дефисом считаются за несколько слов.

7. **HQt_GetTypeCharEng** — Функция выдает тип вводимого QString (считая, что это буква английская). Нужно для алгоритма переноса строк П.Христова в модификации Дымченко и Варсанофьева.

8. **HQt_GetTypeCharRus** — Функция выдает тип вводимого QString (считая, что это буква). Нужно для алгоритма переноса строк П.Христова в модификации Дымченко и Варсанофьева.

Работа с датами

1. **HQt_DaysBetweenDates** — Функция определяет сколько дней между двумя датами.

Работа с файлами и папками

1. **HQt_CopyFile** — Функция копирует файл filename в папку dir. Функция копирует файл filename в папку dir, с возможностью перезаписи (в функции-перегрузке).

2. **HQt_DirDelete** — Функция удаляет директорию и всё ее содержимое.

3. **HQt_DirExists** — Функция проверяет существование директории.

4. **HQt_DirMake** — Функция проверяет существование директории, и если ее нет, то создает.

5. **HQt_FileExists** — Функция проверяет существование файла.

6. **HQt_GetExpFromFilename** — Функция получает расширение файла по его имени.

7. **HQt_GetFilenameFromFullFilename** — Функция получает имя файла по полному пути.

8. **HQt_GetNameFromFilename** — Функция получает имя файла без расширения по его имени.

9. **HQt_ListDirsInDir** — Функция считывает список директорий в директории в QString.

10. **HQt_ListDirsInDirQStringList** — Функция считывает список директорий в директории в QStringList..

11. **HQt_ListFilesInDir** — Функция считывает список файлов (включая скрытые) в директории в QString.

12. **HQt_ListFilesInDirQStringList** — Функция считывает список файлов (включая скрытые) в директории в QStringList.
13. **HQt_ReadFile** — Функция считывает текстовой файл в QString.
14. **HQt_ReadFileToQStringList** — Функция считывает текстовой файл в QStringList.
15. **HQt_RenameFile** — Функция переименовывает файл filename в newfilename.
16. **HQt_SaveFile** — Функция сохраняет QString в текстовой файл.

Работа с цветом

1. **THQt_AlphaBlendingColorToColor** — Функция накладывает сверху на цвет другой цвет с определенной прозрачностью.
2. **THQt_ColorFromGradient** — Функция выдает код RGB из градиента от одного цвета FirstRGB к другому цвету SecondRGB согласно позиции от 0 до 1.
3. **THQt_GiveRainbowColorRGB** — Функция выдает код RGB из градиента радуги для любой позиции от 0 до 1 из этого градиента.
4. **THQt_RGBStringToThreeNumbers** — Функция переводит строку RGB типа #25ffb5 в три числа от 0 до 255, которые кодируют цвета.
5. **THQt_ThreeNumbersToRGBString** — Функция переводит три числа в строку RGB типа #25ffb5, как в Photoshop или HTML.

Работа со строками и списками строк

1. **HQt_AddUniqueQStringInQStringList** — Функция добавляет в QStringList строку QString. Но если такая строка уже присутствует, то добавление не происходит.
2. **HQt_IsNumeric** — Функция проверяет - является ли строка числом.
3. **HQt_MaxCountOfQStringList** — Функция выдает длину макимальной по длине строки в QStringList.
4. **HQt_NaturalCompareTwoQStrings** — Функция сравнивает две строки и определяет какая строчка идет первой. Служебная функция для сортировки строк в обычном стиле, когда строки: z1, z10, z2 сортируются как z1, z2, z10.
5. **HQt_NaturalSortingQStringList** — Функция сортировки строк в сортировки строк QStringList в натуральном виде, например, строки: z1, z10, z2 сортируются как z1, z2, z10.
6. **HQt_QStringListToQString** — Функция переводит QStringList в QString.
7. **HQt_QStringToNumber** — Функция выводит строку x в число.
8. **HQt_QStringToQStringList** — Функция переводит QString в QStringList.
9. **HQt_SearchQStringInQStringList** — Функция ищет в QStringList строку QString (номер первого вхождения).
10. **HQt_StringForLaTeX** — Функция обрабатывает строку String так, чтобы она подходила для публикации в LaTeX.

11. **HQt_StringFromLaTeX** — Функция обрабатывает строку String из переделки функции HQt_StringForLaTeX в нормальную строку. Еще удаляются знаки $, которые обрамляют формулы.

12. **HQt_StringToLabelForLaTeX** — Функция обрабатывает строку String так, чтобы она подходила для публикации в LaTeX в виде label.

13. **HQt_TextAfterEqualSign** — Функция возвращает текст строки после первого знака =.

14. **HQt_TextBeforeEqualSign** — Функция возвращает текст строки до первого знака =.

15. **THQt_VectorToQStringList** — Функция переводит вектор чисел в QStringList.

Служебные функции

1. **HQt_Delay** — Функция делает задержку в MSecs миллисекунд.

3 Функции

3.1 Обработка данных, считанных их файла

3.1.1 HQt_CountOfColsFromQStringList

Функция подсчитывает сколько столбцов в текстовом файле, который скопировали в QStringListFromFile.

Код 1. Синтаксис

```
int HQt_CountOfColsFromQStringList(QStringList QStringListFromFile);
```

Входные параметры:

QStringListFromFile - непосредственно сам файл (его содержимое).

Возвращаемое значение:

Число столбцов (по первой строке).

Пример содержимого QStringListFromFile:

1 2.2

2.8 9

Считается, что файл правильный, ошибки не проверяются. В строке числа разделяются через табуляцию, а десятичные числа используют точку, а не запятую. Во всех столбцах должно быть одинаковое число элементов. Поэтому, если в одном столбце больше элементов, чем в других, то в остальные столбцы на место недостающих элементов ставится знак «-».

3.1.2 HQt_CountOfRowsFromQStringList

Функция подсчитывает сколько строк в текстовом файле, который скопировали в QStringListFromFile. Функция подсчитывает сколько строк в k столбце из текстового файла, который скопировали в QStringListFromFile. Функция подсчитывает сколько строк в каждом столбце из текстового файла с матрицей, который скопировали в QStringListFromFile.

Код 2. Синтаксис

```
int HQt_CountOfRowsFromQStringList(QStringList QStringListFromFile);
int HQt_CountOfRowsFromQStringList(QStringList QStringListFromFile, int k);
int HQt_CountOfRowsFromQStringList(QStringList QStringListFromFile, int *
    VMHL_ResultVector);
```

Входные параметры:

QStringListFromFile — непосредственно сам файл (его содержимое).

k — номер столбца (необзяательно).

VMHL_ResultVector — сюда количество строк заносится (необязательно).

Возвращаемое значение:

Число строк.

Пример содержимого QStringListFromFile:

1 2.2

2.8 9

Считается, что файл правильный, ошибки не проверяются. В строке числа разделяются через табуляцию, а десятичные числа используют точку, а не запятую. Во всех столбцах должно быть одинаковое число элементов. Поэтому, если в одном столбце больше элементов, чем в других, то в остальные столбцы на место недостающих элементов ставится знак «-».

3.1.3 THQt_ReadColFromQStringList

Функция считывает данные какого-то k столбца с датами из QStringList в виде матрицы. Функция считывает данные какого-то k столбца с датами из QStringList в виде матрицы. Для строк.

Код 3. Синтаксис

```
void THQt_ReadColFromQStringList(QStringList QStringListFromFile, int k, QDate *
    VMHL_VectorResult);
void THQt_ReadColFromQStringList(QStringList QStringListFromFile, int k, QString *
    VMHL_VectorResult);
```

Входные параметры:

QStringListFromFile — отсюда берем информацию;

k — номер столбца, начиная с нуля, который считываем;

VMHL_VectorResult — сюда будем записывать результат считывания столбца из матрицы.

Входные параметры 2:

QStringListFromFile — отсюда берем информацию;

k — номер столбца, начиная с нуля, который считываем;

VMHL_VectorResult — сюда будем записывать результат считывания столбца из матрицы.

Возвращаемое значение:

Число строк.

Пример содержимого QStringListFromFile:

1 2013.04.05 6

52 2013.02.25 96

6.4 2013.01.15 4

Считается, что файл правильный, ошибки не проверяются. В строке числа разделяются через табуляцию, а десятичные числа используют точку, а не запятую. Во всех столбцах должно быть одинаковое число элементов. Поэтому, если в одном столбце больше элементов, чем в других, то в остальные столбцы на место недостающих элементов ставится знак «-».

3.1.4 THQt_ReadColFromQStringList2

Функция считывает данные какого-то k столбца из QStringList в виде матрицы.

Код 4. Синтаксис

```
template <class T> void THQt_ReadColFromQStringList(QStringList QStringListFromFile,
    int k, T *VMHL_VectorResult);
```

Входные параметры:

QStringListFromFile — отсюда берем информацию;

k — номер столбца, начиная с нуля, который считываем;

VMHL_VectorResult — сюда будем записывать результат считывания столбца из матрицы.

Возвращаемое значение:

Количество элементов в столбце. Как только встречает вместо числа символ «-», то функция считает, что вектор закончился.

Пример содержимого QStringListFromFile:

1 2 6

52 3 96

6.4 7 4

Второй пример содержимого VMHL_VectorResult.

1 2 6 5

52 3 96 5

- - 4 2

Считается, что файл правильный, ошибки не проверяются. В строке числа разделяются через табуляцию, а десятичные числа используют точку, а не запятую. Во всех столбцах должно быть одинаковое число элементов. Поэтому, если в одном столбце больше элементов, чем в других, то в остальные столбцы на место недостающих элементов ставится знак «-».

Код 5. Пример использования

```
QString DS=QDir::separator();
QString path=QGuiApplication::applicationDirPath()+DS;//путь к папке

QStringList List = HQt_ReadFileToQStringList(path+"5.txt");
int N;
N=HQt_CountOfRowsFromQStringList(List,k);

double *X;
X=new double[N];

int k=2;//номер столбца

THQt_ReadColFromQStringList(List, k, X);

delete [] X;
```

3.1.5 THQt_ReadMatrixFromQStringList

Функция считывает данные из QStringList в матрицу.

Код 6. Синтаксис

```
template <class T> void THQt_ReadMatrixFromQStringList(QStringList
   QStringListFromFile, T **VMHL_MatrixResult);
```

Входные параметры:

QStringListFromFile — отсюда берем информацию;

VMHL_MatrixResult — сюда будем записывать результат считывания матрицы.

Возвращаемое значение:

Отсутствует.

Пример содержимого:

1 2 6

52 3 96

6.4 7 4

Второй пример содержимого:

1 2 6 5

52 3 96 5

- - 4 2

Десятичные числа должны разделяться точкой.

Код 7. Пример использования

```
QString DS=QDir::separator();
QString path=QGuiApplication::applicationDirPath()+DS;//путь к папке
QStringList List = HQt_ReadFileToQStringList(path+"5.txt");

int N,M;
M=HQt_CountOfColsFromQStringList(List);
N=HQt_CountOfRowsFromQStringList(List);

double **X;
X=new double*[N];
for (int i=0;i<N;i++) X[i]=new double[M];

THQt_ReadMatrixFromQStringList(List, X);

for (int i=0;i<N;i++) delete [] X[i];
delete [] X;
```

3.1.6 THQt_ReadTwoVectorFromQStringList

Функция считывает данные из QStringList в два вектора.

Код 8. Синтаксис

```
template <class T> void THQt_ReadTwoVectorFromQStringList(QStringList
    QStringListFromFile, T *VMHL_VectorResult1, T *VMHL_VectorResult2);
template <class T> void THQt_ReadTwoVectorFromQStringList(QStringList
    QStringListFromFile, T *VMHL_VectorResult1, QDate *VMHL_VectorResult2);
```

Входные параметры:

QStringListFromFile — отсюда берем информацию;

VMHL_VectorResult1 — сюда будем записывать результат первого вектора;

VMHL_VectorResult2 — сюда будем записывать результат второго вектора.

Возвращаемое значение:

Отсутствует.

Пример содержимого:

1 2

52 3

6.4 7

Десятичные числа должны разделяться точкой.

Код 9. Пример использования

```
QString DS=QDir::separator();
QString path=QGuiApplication::applicationDirPath()+DS;//путь к папке
int N;
double *x,*y;
QStringList List = HQt_ReadFileToQStringList(path+"2.txt");
N=HQt_CountOfRowsFromQStringList(List);
x=new double [N];
y=new double [N];

THQt_ReadTwoVectorFromQStringList(List,x,y);

delete [] y;
delete [] x;
```

3.1.7 THQt_ReadVectorFromQStringList

Функция считывает данные из QStringList в вектор.

Код 10. Синтаксис

```
template <class T> void THQt_ReadVectorFromQStringList(QStringList
    QStringListFromFile, T *VMHL_VectorResult);
```

Входные параметры:

QStringListFromFile — отсюда берем информацию;

VMHL_VectorResult — сюда будем записывать результат.

Возвращаемое значение:

Отсутствует.

Пример содержимого:

1

52

6.45

Десятичные числа должны разделяться точкой.

Код 11. Пример использования

```
QString DS=QDir::separator();
QString path=QGuiApplication::applicationDirPath()+DS;//путь к папке
int N;
double *y;
QStringList List = HQt_ReadFileToQStringList(path+"1.txt");
N=HQt_CountOfRowsFromQStringList(List);
y=new double [N];

THQt_ReadVectorFromQStringList(List,y);//считываем

delete [] y;
```

3.2 Получение строк-выводов по данным

3.2.1 HQt_BoolToWord

Функция переводит переменную, принимающую значения "0"и "1"в слова.

Код 12. Синтаксис

```
QString HQt_BoolToWord(QString Bool);
QString HQt_BoolToWord(QString Bool, QString No, QString Yes);
QString HQt_BoolToWord(bool Bool);
QString HQt_BoolToWord(bool Bool, QString No, QString Yes);
QString HQt_BoolToWord(int Bool);
QString HQt_BoolToWord(int Bool, QString No, QString Yes);
```

Входные параметры:

Bool — исходная переменная.

No — слово, которое означает 0 (необязательно).

Yes — слово, которое означает 1 (необязательно).

Возвращаемое значение:

Слово, которое характеризует переменную.

3.2.2 HQt_RandomString

Функция генерирует случайную строку из английских больших и малых букв.

Код 13. Синтаксис

```
QString HQt_RandomString(int Length);
```

Входные параметры:

Length — длина строки, которую надо сгенерировать.

Возвращаемое значение:

Случайная строка.

Примечание:

Используются случайные числа, так что рекомендуется вызвать в программе инициализатор случайных чисел qsrand.

Рекомендую так: qsrand(QDateTime::currentMSecsSinceEpoch ()

3.2.3 HQt_TryingReduceString

Функция старается сократить строку длиной больше MaxSize символов, сокращая слова. Функция старается сократить строку длиной больше 40 символов, сокращая слова.

Код 14. Синтаксис

```
QString HQt_TryingReduceString(QString text, int MaxSize);
QString HQt_TryingReduceString(QString text);
```

Входные параметры:

text — сокращаемая строка;

MaxSize — с какого количества символов сокращаем строку (необязательно).

Возвращаемое значение:

Сокращенная строка. Обратите внмиание, что строка сокращенная может быть длиннее MaxSize.

3.2.4 HQt_UniqueName

Функция возвращает уникальную строку, которую можно использовать как некий идентификатор. Собирается из "HQt_"+ текущее время или из BeginString+"_"+ текущее время.

Код 15. Синтаксис

```
QString HQt_UniqueName ();
QString HQt_UniqueName (QString BeginString);
```

Входные параметры:

BeginString — Приставка вначале строки (необязательно).

Возвращаемое значение:

Уникальная строка.

3.2.5 HQt_UniqueNameOnlyNumbers

Функция возвращает уникальную строку, которую можно использовать как некий идентификатор. В строке только цифры. Собирается из текущего времени.

Код 16. Синтаксис

```
QString HQt_UniqueNameOnlyNumbers ();
```

Входные параметры:

Отсутствуют.

Возвращаемое значение:

Уникальная строка.

3.2.6 HQt_WriteTime

Функция переводит миллисекунды в строку с описанием сколько это минут, секунд и др.

Код 17. Синтаксис

```
QString HQt_WriteTime(int t);
QString HQt_WriteTime(qint64 t);
```

Входные параметры:

t — миллисекунды.

Возвращаемое значение:

Строка в виде текста — сколько секунд, минут и так далее было.

3.3 Работа с алфавитом и переносами

3.3.1 HQt_BreakdownOfTextWithWordWrap

Функция разбивает текст на строки длиной не более length. Если может, то ставит переносы.

Код 18. Синтаксис

```
QStringList HQt_BreakdownOfTextWithWordWrap(QString S, int length);
```

Входные параметры:

S — проверяемая строка,

length — длина строки.

Возвращаемое значение:

Список строк, на которые разбивается текст.

Примечание:

Перевод слов производится по алгоритму П.Христова в модификации Дымченко и Варсанофьева.

3.3.2 HQt_CheckIntolerablePunctuation

Является ли символом знаком пунктуации, который нельзя переносить.

Код 19. Синтаксис

```
bool HQt_CheckIntolerablePunctuation(QString x);
```

Входные параметры:

x — некий символ.

Возвращаемое значение:

true — символ есть непереносимый символ;

false — не из слова.

3.3.3 HQt_CheckLetterFromWord

Является ли буква символом из слова. Считается, что это или латинские буквы, или русские, или цифры или нижнее подчеркивание. Плюс некоторые знаки препинания, так как их нельзя переносить.

Код 20. Синтаксис

```
bool HQt_CheckLetterFromWord(QString x);
```

Входные параметры:

x — некая буква.

Возвращаемое значение:

true — буква из слова;

false — не из слова.

3.3.4 HQt_CheckRus

Функция проверяет наличие русских букв в строке.

Код 21. Синтаксис

```
bool HQt_CheckRus(QString S);
```

Входные параметры:

S — проверяемая строка.

Возвращаемое значение:

true — ectm буквы русские;

false — нет букв русских.

3.3.5 HQt_CutToWords

Функция разбивает строку на слова и те, части, между которыми они находятся. Слова с дефисом считаются за несколько слов.

Код 22. Синтаксис

```
QStringList HQt_CutToWords(QString S);
```

Входные параметры:

S - разбиваемая строка.

Возвращаемое значение:

Список подстрок, на которые можно разбить слово.

Примечание:

Если кроме русского и английского языка у вас могут слова других языков, то дополните функцию HQt_CheckLetterFromWord.

3.3.6 HQt_CutToWordsWithWordWrap

Функция разбивает строку на слова и те, части, между которыми они находятся. А русские и английские слова разделяет по переносам. Слова с дефисом считаются за несколько слов.

Код 23. Синтаксис

```
QStringList HQt_CutToWordsWithWordWrap(QString S);
```

Входные параметры:

S - разбиваемая строка.

Возвращаемое значение:

Список подстрок, на которые можно разбить слово.

Примечание:

Если кроме русского и английского языка у вас могут слова других языков, то дополните функцию HQt_CheckLetterFromWord.

Перевод слов производится по алгоритму П. Христова в модификации Дымченко и Варсанофьева.

3.3.7 HQt_GetTypeCharEng

Функция выдает тип вводимого QString (считая, что это буква английская). Нужно для алгоритма переноса строк П.Христова в модификации Дымченко и Варсанофьева.

Код 24. Синтаксис

```
int HQt_GetTypeCharEng(QString x);
```

Входные параметры:

x — некая буква.

Возвращаемое значение:

1 — гласная;

2 — согласная;

0 — иначе.

3.3.8 HQt_GetTypeCharRus

Функция выдает тип вводимого QString (считая, что это буква). Нужно для алгоритма переноса строк П.Христова в модификации Дымченко и Варсанофьева.

Код 25. Синтаксис

```
int HQt_GetTypeCharRus(QString x);
```

Входные параметры:

x — некая буква.

Возвращаемое значение:

1 — гласная;

2 — согласная;

3 — буква из множества ьъй;

0 — иначе (английские или какие—то иные).

3.4 Работа с датами

3.4.1 HQt_DaysBetweenDates

Функция определяет сколько дней между двумя датами.

Код 26. Синтаксис

```
int HQt_DaysBetweenDates(QDate BeginDate, QDate EndDate);
int HQt_DaysBetweenDates(QString BeginDate, QString EndDate);
```

Входные параметры:

BeginDate — первая дата.

EndDate — вторая дата.

Возвращаемое значение:

Число дней между датами.

3.5 Работа с файлами и папками

3.5.1 HQt_CopyFile

Функция копирует файл filename в папку dir. Функция копирует файл filename в папку dir, с возможностью перезаписи (в функции-перегрузке).

Код 27. Синтаксис

```
bool HQt_CopyFile(QString filename, QString dir);
bool HQt_CopyFile(QString filename, QString dir, bool overwrite);
bool HQt_CopyFile(QString filename, QString dir, bool overwrite, bool dirmake);
```

Входные параметры:

filename — имя файла (с полным путем);

dir — путь к папке, куда нужно скопировать файл;

overwrite — если true, то перезаписывать, если false, то не перезаписывать (необязательный параметр);

dirmake — если true, то если нет директории, то она создается (необязательный параметр).

Возвращаемое значение:

true — если скопировалось удачно,

false — если скопировалось неудачно.

3.5.2 HQt_DirDelete

Функция удаляет директорию и всё ее содержимое.

Код 28. Синтаксис

```
bool HQt_DirDelete(QString path);
```

Входные параметры:

path — полный путь к папке.

Возвращаемое значение:

true — если удаление прошло нормально.

false — если удаление прошло не нормально.

3.5.3 HQt_DirExists

Функция проверяет существование директории.

Код 29. Синтаксис

```
bool HQt_DirExists(QString path);
```

Входные параметры:

path — полный путь к папке.

Возвращаемое значение:

false — если папка не существует;

true — если папка существует.

3.5.4 HQt_DirMake

Функция проверяет существование директории, и если ее нет, то создает.

Код 30. Синтаксис

```
void HQt_DirMake(QString path);
```

Входные параметры:

path — полный путь к папке.

Возвращаемое значение:

Отсутствует.

3.5.5 HQt_FileExists

Функция проверяет существование файла.

Код 31. Синтаксис

```
bool HQt_FileExists(QString filename);
```

Входные параметры:

filename — имя файла.

Возвращаемое значение:

false — если файл не существует;

true — если файл существует.

3.5.6 HQt_GetExpFromFilename

Функция получает расширение файла по его имени.

Код 32. Синтаксис

```
QString HQt_GetExpFromFilename(QString filename);
```

Входные параметры:

filename — имя файла.

Возвращаемое значение:

Строка значением расширения файла в нижнем регистре.

3.5.7 HQt_GetFilenameFromFullFilename

Функция получает имя файла по полному пути.

Код 33. Синтаксис

```
QString HQt_GetFilenameFromFullFilename(QString filename);
```

Входные параметры:

filename — имя файла с путем.

Возвращаемое значение:

Строка с именем файла.

3.5.8 HQt_GetNameFromFilename

Функция получает имя файла без расширения по его имени.

Код 34. Синтаксис

```
QString HQt_GetNameFromFilename(QString filename);
```

Входные параметры:

filename — имя файла (можно и с полным путем).

Возвращаемое значение:

Строка с именем файла без расширения.

3.5.9 HQt_ListDirsInDir

Функция считывает список директорий в директории в QString.

Код 35. Синтаксис

```
QString HQt_ListDirsInDir(QString path);
```

Входные параметры:

path — путь к папке.

Возвращаемое значение:

Строка со списком директорий в директории, разделенные
n в алфавитном порядке.

3.5.10 HQt_ListDirsInDirQStringList

Функция считывает список директорий в директории в QStringList..

Код 36. Синтаксис

```
QStringList HQt_ListDirsInDirQStringList(QString path);
```

Входные параметры:

path — путь к папке.

Возвращаемое значение:

Список строк со списком директорий в директории в алфавитном порядке.

3.5.11 HQt_ListFilesInDir

Функция считывает список файлов (включая скрытые) в директории в QString.

Код 37. Синтаксис

```
QString HQt_ListFilesInDir(QString path);
```

Входные параметры:

path — путь к папке.

Возвращаемое значение:

Строка со списком файлов в директории, разделенные
n в алфавитном порядке.

3.5.12 HQt_ListFilesInDirQStringList

Функция считывает список файлов (включая скрытые) в директории в QStringList.

Код 38. Синтаксис

```
QStringList HQt_ListFilesInDirQStringList(QString path);
```

Входные параметры:

path — путь к папке.

Возвращаемое значение:

Список строк файлов в директории в алфавитном порядке.

3.5.13 HQt_ReadFile

Функция считывает текстовой файл в QString.

Код 39. Синтаксис

```
QString HQt_ReadFile(QString filename);
```

Входные параметры:

filename — имя файла.

Возвращаемое значение:

Строка со всем содержимым текстового файла.

3.5.14 HQt_ReadFileToQStringList

Функция считывает текстовой файл в QStringList.

Код 40. Синтаксис

```
QStringList HQt_ReadFileToQStringList(QString filename);
```

Входные параметры:

filename — имя файла.

Возвращаемое значение:

QStringList со всем содержимым текстового файла.

3.5.15 HQt_RenameFile

Функция переименовывает файл filename в newfilename.

Код 41. Синтаксис

```
bool HQt_RenameFile(QString filename, QString newfilename);
```

Входные параметры:

filename — имя файла (с полным путем),

newfilename — новое имя файла (без полного пути).

Возвращаемое значение:

true — если переименовалось удачно,

false — если переименовалось неудачно.

3.5.16 HQt_SaveFile

Функция сохраняет QString в текстовой файл.

Код 42. Синтаксис

```
void HQt_SaveFile(QString line, QString filename);
```

Входные параметры:

line — содержимое, которое нужно сохранить;

filename — имя файла.

Возвращаемое значение:

Отсутствует.

3.6 Работа с цветом

3.6.1 THQt_AlphaBlendingColorToColor

Функция накладывает сверху на цвет другой цвет с определенной прозрачностью.

Код 43. Синтаксис

```
QString THQt_AlphaBlendingColorToColor(double alpha, QString FirstRGB, QString
    SecondRGB);
```

Входные параметры:

alpha — прозрачность второго накладываемого цвета из интервала [0;1];

FirstRGB — строка RGB кода первого цвета градиента, например: #63ddb2;

SecondRGB — строка RGB кода последнего цвета градиента, например: #5845da.

Возвращаемое значение:

Строка содержащая код цвета, например: #25ffb5.

3.6.2 THQt_ColorFromGradient

Функция выдает код RGB из градиента от одного цвета FirstRGB к другому цвету SecondRGB согласно позиции от 0 до 1.

Код 44. Синтаксис

```
QString THQt_ColorFromGradient(double position, QString FirstRGB, QString SecondRGB);
```

Входные параметры:

position — позиция из интервала [0;1], которая говорит какой цвет выдать из градиента;

FirstRGB — строка RGB кода первого цвета градиента, например: #63ddb2;

SecondRGB — строка RGB кода последнего цвета градиента, например: #5845da.

Возвращаемое значение:

Строка содержащая код цвета, например: #25ffb5.

3.6.3 THQt_GiveRainbowColorRGB

Функция выдает код RGB из градиента радуги для любой позиции от 0 до 1 из этого градиента.

Код 45. Синтаксис

```
QString THQt_GiveRainbowColorRGB(double position);
```

Входные параметры:

position — позиция из интервала [0;1], которая говорит какой цвет выдать из радуги.

Возвращаемое значение:

Строка содержащая код цвета, например: #25ffb5.

3.6.4 THQt_RGBStringToThreeNumbers

Функция переводит строку RGB типа #25ffb5 в три числа от 0 до 255, которые кодируют цвета.

Код 46. Синтаксис

```
void THQt_RGBStringToThreeNumbers(QString RGB, int *R, int *G, int *B);
```

Входные параметры:

RGB — строка, которая содержит код RGB цвета вида: #25ffb5.

R — число от 0 до 255 включительно означающее красный цвет;

G — число от 0 до 255 включительно означающее зеленый цвет;

B — число от 0 до 255 включительно означающее синий цвет.

Возвращаемое значение:

Отсутствует.

3.6.5 THQt_ThreeNumbersToRGBString

Функция переводит три числа в строку RGB типа #25ffb5, как в Photoshop или HTML.

Код 47. Синтаксис

```
QString THQt_ThreeNumbersToRGBString(int R, int G, int B);
```

Входные параметры:

int R — число от 0 до 255 включительно означающее красный цвет;

int G — число от 0 до 255 включительно означающее зеленый цвет;

int B — число от 0 до 255 включительно означающее синий цвет.

Возвращаемое значение:

Строка содержащая код цвета, например: #25ffb5.

3.7 Работа со строками и списками строк

3.7.1 HQt_AddUniqueQStringInQStringList

Функция добавляет в QStringList строку QString. Но если такая строка уже присутствует, то добавление не происходит.

Код 48. Синтаксис

```
QStringList HQt_AddUniqueQStringInQStringList (QStringList StringList, QString String
    );
```

Входные параметры:

StringList — QStringList, в который мы добавляем строку (добавление в возвращаемом элементе);

String — добавляемая строка.

Возвращаемое значение:

Список строк с добавленной строкой.

3.7.2 HQt_IsNumeric

Функция проверяет - является ли строка числом.

Код 49. Синтаксис

```
bool HQt_IsNumeric(QString x);
```

Входные параметры:

x — проверяемая строка.

Возвращаемое значение:

true — является числом;

false — не является числом.

3.7.3 HQt_MaxCountOfQStringList

Функция выдает длину макимальной по длине строки в QStringList.

Код 50. Синтаксис

```
int HQt_MaxCountOfQStringList(QStringList x);
```

Входные параметры:

x — список строк.

Возвращаемое значение:

Длина макимальной по длине строки.

3.7.4 HQt_NaturalCompareTwoQStrings

Функция сравнивает две строки и определяет какая строчка идет первой. Служебная функция для сортировки строк в обычном стиле, когда строки: z1, z10, z2 сортируются как z1, z2, z10.

Код 51. Синтаксис

```
bool HQt_NaturalCompareTwoQStrings(const QString& s1,const QString& s2);
```

Входные параметры:

s1 — первая строка;

s2 — вторая строка.

Возвращаемое значение:

false — когда вторая строка должна быть первой;

true — когда первая строка должна быть первой.

3.7.5 HQt_NaturalSortingQStringList

Функция сортировки строк в сортировки строк QStringList в натуральном виде, например, строки: z1, z10, z2 сортируются как z1, z2, z10.

Код 52. Синтаксис

```
QStringList HQt_NaturalSortingQStringList (QStringList StringList);
```

Входные параметры:

StringList — сортируемый список строк.

Возвращаемое значение:

Отсортированный список строк.

3.7.6 HQt_QStringListToQString

Функция переводит QStringList в QString.

Код 53. Синтаксис

```
QString HQt_QStringListToQString(QStringList lines);
```

Входные параметры:

lines — список строк.

Возвращаемое значение:

Строка с разделениями
n.

3.7.7 HQt_QStringToNumber

Функция выводит строку x в число.

Код 54. Синтаксис

```
double HQt_QStringToNumber (QString x);
double HQt_QStringToNumber (QString x, bool checkcomma);
```

Входные параметры:

x — строка.

checkcomma — проверять наличие запятой (необязательно).

Возвращаемое значение:

Число из строки.

3.7.8 HQt_QStringToQStringList

Функция переводит QString в QStringList.

Код 55. Синтаксис

```
QStringList HQt_QStringToQStringList(QString line);
```

Входные параметры:

line — строка.

Возвращаемое значение:

Список строк.

3.7.9 HQt_SearchQStringInQStringList

Функция ищет в QStringList строку QString (номер первого вхождения).

Код 56. Синтаксис

```
int HQt_SearchQStringInQStringList (QStringList StringList, QString String);
```

Входные параметры:

StringList — QStringList, в который мы ищем строку;

String - добавляемая строка.

Возвращаемое значение:

Номер найденной строки. Если не найдено, то возвращается -1.

3.7.10 HQt_StringForLaTeX

Функция обрабатывает строку String так, чтобы она подходила для публикации в LaTeX.

Код 57. Синтаксис

```
QString HQt_StringForLaTeX (QString String);
```

Входные параметры:

String — обрабатываемая строка.

Возвращаемое значение:

Обработанная строка.

3.7.11 HQt_StringFromLaTeX

Функция обрабатывает строку String из переделки функции HQt_StringForLaTeX в нормальную строку. Еще удаляются знаки $, которые обрамляют формулы.

Код 58. Синтаксис

```
QString HQt_StringFromLaTeX (QString String);
```

Входные параметры:

String — обрабатываемая строка.

Возвращаемое значение:

Обработанная строка.

3.7.12 HQt_StringToLabelForLaTeX

Функция обрабатывает строку String так, чтобы она подходила для публикации в LaTeX в виде label.

Код 59. Синтаксис

```
QString HQt_StringToLabelForLaTeX (QString String);
```

Входные параметры:

String — обрабатываемая строка.

Возвращаемое значение:

Обработанная строка.

3.7.13 HQt_TextAfterEqualSign

Функция возвращает текст строки после первого знака =.

Код 60. Синтаксис

```
QString HQt_TextAfterEqualSign (QString String);
```

Входные параметры:

String — строка вида: Title = Пример

Возвращаемое значение:

Текст строки после первого знака =.

3.7.14 HQt_TextBeforeEqualSign

Функция возвращает текст строки до первого знака =.

Код 61. Синтаксис

```
QString HQt_TextBeforeEqualSign (QString String);
```

Входные параметры:

String — строка вида: Title = Пример

Возвращаемое значение:

Текст строки до первого знака =.

3.7.15 THQt_VectorToQStringList

Функция переводит вектор чисел в QStringList.

Код 62. Синтаксис

```
template <class T> void THQt_VectorToQStringList(T *x, int N);
```

Входные параметры:

x — переводимый массив.

N — Количество элементов в массиве.

Возвращаемое значение:

Список строк.

3.8 Служебные функции

3.8.1 HQt_Delay

Функция делает задержку в MSecs миллисекунд.

Код 63. Синтаксис

```
void HQt_Delay(int MSecs);
```

Входные параметры:

MSecs — миллисекунды, сколько надо подержать работу Qt. Не меньше пяти миллисекунд должно быть.

Возвращаемое значение:

Отсутствуют.