Работа с файлами в библиотеке Qt

Типовая последовательность работы с файлом

Стандартный диалог выбора файла

Класс файла

Потоки данных

Использование перегруженных операций для работы с потоками

Типовая последовательность работы с файлом

узнать имя файла используя стандартный диалог (`QFileDialog`)

создать объект файла (`QFile`)

открыть файл (`QFile`)

создать поток для ввода/вывода (`QDataStream` или `QTextStream`)

считать/записать данные с использованием потока

закрыть файл — выполняется автоматически при уничтожении объекта (`QFile`)

Стандартный диалог выбора файла

Стандартный диалог выбора файла предназначен для задания пользователем имени файла (каталога), который будет открыт или сохранен.

В библиотеке Qt стандартный диалог выбора файла реализуется классом `QFileDialog`.

Работа со стандартным диалогом выбора файла

В большинстве случаев при работе с классом `QFileDialog` используется одна из следующих статических функций:

- `getOpenFileName()`
- `getSaveFileName()`
- `getOpenFileNames()`
- `getExistingDirectory()`

Все параметры этих функций имеют значения по умолчанию, поэтому могут явно не указываться

Диалог открытия файла

Диалог открытия файла

QString — имя выбранного файла, пустая строка в случае отмены

```
QFileDialog::getOpenFileName(
QWidget * - указатель на окно-родитель
const QString & - строка заголовка
const QString & - начальный каталог (пустая строка если использовать текущий)
const QString & - фильтр файлов по расширению (см. ниже)
QString * - изначально выбранный фильтр, обычно 0
Options — опции настройки, обычно 0
 )
```

Фильтрация файлов по расширению

Диалог выбора файлов может отбирать файлы по указанным расширениям.

При этом пользователь может выбрать один из предложенных вариантов фильтрации.

Фильтрация файлов по расширению

Строка фильтров состоит из описаний фильтров, разделенных двойным знаком точки с запятой.

Описание фильтра состоит из имени и перечня шаблонов для имен файлов в круглых скобках

Пример строки фильтров: `"Open Office document (*.odt, *.odp);;Portable document format (*.pdf)"`

Задание

Вызовите стандартный диалог открытия файла с заголовком «Открыть файл», в текущем каталоге, с двумя фильтрами: текстовые файлы (расширения txt, bat) и все файлы.

Имя открытого файла сохранить в переменную **`FileName`**

Диалог открытия файла

```
FileName =
QFileDialog::getOpenFileName(
this,
QString("Открыть файл"),
QString(),
QString("Текстовые файлы
  (*.txt,*.bat);;Все файлы (*.*)"));
```

Класс файла

Для представления файла в программе используется класс `QFile`.

Создание объекта файла

Один из конструкторов **QFile** принимает имя файла в качестве параметра:

```
QFile::QFile(
const QString & - имя файла
)
```

Созданный объект привязан к файлу, но файл при этом не открывается — он должен быть открыт методом **open()**.

Открытие файла

`bool` — успешно ли открытие

`QFile::open(`

`OpenMode` — режим доступа.

`)`

Возможные режимы доступа:

`QIODevice::ReadOnly` — только для чтения;

`QIODevice::WriteOnly` — только для записи, имеющиеся данные затираются;

`QIODevice::ReadWrite` — для чтения и записи, новые данные добавляются к уже существующим.

Открытие файла

bool — успешно ли открытие
QFile::open(
OpenMode — режим:
QIODevice::ReadOnly,
QIODevice::WriteOnly или
QIODevice::ReadWrite — права, с которыми открывается файл
дополнительно может указываться **QIODevice::Text** для взаимодействия с файлом в текстовом режиме (через операцию побитового ИЛИ)
)

Потоки ввода/вывода

Поток — это объект, предназначенный для ввода/вывода даных с использованием объекта с последовательным доступом (например файла).

В библиотеке Qt потоки ввода/вывода представлены классами `QDataStream` и `QTextStream`.

В конструктор потока передается указатель на объект, в который осуществляется вывод (ввод) данных, например файл.

Задание

Создайте объект файла `MyFl` с именем `FileName.dat`.

Откройте этот файл в режиме только записи данных.

В случае успешного открытия создайте объект потока `outp` для вывода в файл.

Работа с файлами

```
// Создаем файл
QFile MyFl("FileName.dat");

// Создаем поток данных на основе
// файла
if(MyFl.open(QIODevice::WriteOnly))
{
  QDataStream outp(&MyFl);
}
```

Работа с потоками ввода/вывода

Чтение (ввод) из потока осуществляется через перегруженную операцию >>:

```
поток >> переменная для ввода;
```

Несколько операций ввода могут быть записаны в цепочку:

```
поток >> переменная1 >> переменная2;
```

Работа с потоками ввода/вывода

Запись (вывод) в поток осуществляется аналогичным образом с использованием операции <<

Операции ввода/вывода в поток сами определяют тип вводимых/выводимых данных и действуют соответственно.

Задание

Запишите в поток `outp` строку `str` (массив символов), число `N` и константу 103, если известны следующие операции:

```
QDataStream & operator<< (quint16 i)
QDataStream & operator<< (qint16 i)
QDataStream & operator<< (qint32 i)
QDataStream & operator<< (quint32 i)
QDataStream & operator<< (float f)
QDataStream & operator<< (double f)
QDataStream & operator<< (const
                          char * s)
```

Вывод в поток

```
outp << str << N << (qint32)103;
```

Задание

Считайте из потока **inp** строку **str** (массив символов), целое 16-разрядное число **N** и целое 32-разрядное число **M**, если известны следующие операции:

```
QDataStream & operator>> (quint16 &i)
QDataStream & operator>> (qint16 &i)
QDataStream & operator>> (qint32 &i)
QDataStream & operator>> (quint32 &i)
QDataStream & operator>> (float &f)
QDataStream & operator>> (double &f)
QDataStream & operator>> (const
                          char *& s)
```

Вывод в поток

```
char str[81];
qint16 N;
qint32 M;

inp >> str >> N >> M;
```

Операции ввода/вывода

В классах `QDataStream` и `TextStream` определены операции ввода/вывода для стандартных типов данных (числа, массивы символов, логические значения).

В классах `QString`, `QDate`, `QDateTime` и `QTime` определены собственные операции ввода/вывода.

В контейнерных классах также определены операции ввода/вывода. Однако для хранимых значений и ключей должны быть определены операции ввода/вывода.

Задание

Имеется словарь городов. В словаре хранится название города и кол-во его жителей

```
QMap <QString,int> cities;
```

Запишите в поток `outp` содержимое контейнера, если для контенера определена следующая операция:

```
QDataStream & operator<< ( QDataStream &
    out, const QMap<Key, T> & map )
```

Пример записи контейнера в поток данных

```
// Так тип ключа (QString) и тип значения
// (int) поддерживают операции ввода-вывода,
// то запись контейнера в поток выполняется
// одной операцией
outp << cities;
```

Перегрузка операций ввода/вывода в поток

Возможно создать (перегрузить) операции ввода-вывода для собственных классов.

Операции потокового ввода/вывода являются бинарными: левым операндом всегда является ссылка на поток, правым — ссылка на объект класса. Для операции вывода обычно передается ссылка на константный объект.

Для того чтобы операцию ввода-вывода можно было выполнять последовательно, она возвращает ссылку на поток, с которым работаем.

Перегрузка операций ввода/вывода в поток

Поскольку невозможно модифицировать код класса потока, то операции ввода-вывода могут быть перегружены только как свободные функции (не принадлежащие ни одному классу).

Ввод/вывод класса осуществляется, как правило, последовательным вводом (выводом) всех его полей.

Синтаксис операций ввода/вывода в поток

Синтаксис операции вывода данных

```
QDataStream & operator<< ( QDataStream & stream, const <класс> & data )
```

Синтаксис операции ввода данных

```
QDataStream & operator>> ( QDataStream & stream, <класс> & data )
```

Задание

Дан класс:

```
class FIO
{
  QString FirstName;
  QString LastName;
  QString SurName;
};
```

Определите для него операцию вывода в поток **QDataStream**

Перегрузка операции вывода в поток

```
QDataStream &operator<< (
QDataStream &ostrm, const FIO &data)
{
  ostrm << data.FirstName <<
    data.LastName << data.SurName;

  return ostrm;
}
```

Особенности работы с текстовыми потоками

Операции ввода и вывода для потока данных (класса `QDataStream`) являются обратимыми, т.е. если последовательность операций ввода совпадает с последовательностью операций вывода, то данные будут считаны верно.

Для текстового потока (класса `QTextStream`) операции ввода и вывода не обратимы. Например, сохранив в поток три строки, при чтении мы получим одну общую строку.

Особенности работы с текстовыми потоками

Для того чтобы операции ввода-вывода с текстовым потоком были обратимы, необходимо записывать строки в поток через разделители.

В качестве разделителей обычно используются пробелы, знаки табуляции и переводы строк.

Для вывода в текстовый поток перевода строки используется псевдопеременная **`endl`**.

Задание

Для класса **FIO**, приведенного выше, перегрузите оператор вывода в поток **QTextStream**, разделяя поля символами перевода строки.

Перегрузите оператор ввода класса **FIO** из потока **QTextStream**.

Вывод в текстовый поток

```
// Записываем FIO в поток
QTextStream &operator<<
(QTextStream &ostrm, const FIO &data)
{
  ostrm << data.FirstName << endl
        << data.LastName  << endl
        << data.SurName   << endl;

  return ostrm;
}
```

Ввод из текстового потока

```
// Считываем FIO из потока
QTextStream &operator>>
(QTextStream &ostrm, FIO &data)
{
  ostrm >> data.FirstName
        >> data.LastName
        >> data.SurName;

  return ostrm;
}
```

Потоковый ввод/вывод с консоли

Поток `QTextStream` можно использовать для взаимодействия с консолью (текстовым экран), если программа консольная. Для этого используются переменные-псевдофайлы `stdin` (ввод) и `stdout` (вывод).

Пример содания потока для ввода с консоли:

```
QTextStream conin(stdin);
```

Пример содания потока для вывода на консоль:

```
QTextStream conout(stdout);
```