Глава 2

Консольное приложение:

проект DISKINFO

Проект DISKINFO знакомит с приемами разработки консольных приложений. Описывается структура консольного приложения и подключаемые к нему пространства имен. Рассматривается форматированный вывод, в частности, использование управляющих последовательностей. Описываются классы QStorageInfo, QTextCodec и QStringList. Кроме того, описываются приемы работы с параметрами командной строки.

2.1. Создание консольного приложения

При создании нового проекта — консольного приложения следует указать тип проекта **Консольное приложение Qt** в окне **Новый проект**. Далее требуется выбрать расположение проекта, систему сборки (по умолчанию qmake) и комплект.

Созданный проект будет содержать файл main.cpp, который сразу загрузится в редактор (листинг 2.1).

**Листинг 2.1. Исходный текст файла Program.cs для консольного приложения**

#include <QCoreApplication>

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

return a.exec();

}

Метод main является стартовой точкой выполнения программы. Его параметр argv позволяет получить информацию о параметрах командной строки, указанных при запуске данной программы, а argc – их количество (использование параметров argc и argv демонстрируется в разд. 2.3).

Добавьте в метод main новые операторы (листинг 2.2).

**Листинг 2.2. Добавление к методу main**

#include <QCoreApplication>

**#include <QTextCodec>**

**#include <iostream>**

**using namespace std;**

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

**#ifdef Q\_OS\_WIN32**

**QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));**

**#else**

**QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));**

**#endif**

**std::cout << QString("Программа DISKINFO\n").toLocal8Bit().data() << std::endl;**

**std::cout << QString("\nДля завершения программы нажмите <Enter>...")**

**.toLocal8Bit().data() << std::endl;**

~~return a.exec();~~

}

**Результат.** При запуске программы на экране появляется особое *консольное окно*, используемое для ввода/вывода данных в текстовом режиме. Это окно обладает всеми свойствами окна приложения DOS; в частности, оно имеет расширенное системное меню, вызываемое комбинацией клавиш <Alt>+<Пробел>.При выполнении программы можно переключаться в полноэкранный режимкомбинацией клавиш <Alt>+<Enter> (эта же комбинация восстанавливаетоконный режим). Консольное окно будет содержать текст, приведенный в листинге 2.3.

После завершения программы консольное окно немедленно закрывается. Чтобы можно было ознакомиться с содержимым окна, в программу был удален вызов оператора return (см. листинг 2.2, последний оператор).

**Листинг 2.3. Содержимое консольного окна при выполнении программы**

Программа DISKINFO

Для завершения программы нажмите <Enter>...

Комментарии

1. Для корректного отображения русского текста используются кодировки IBM 866 и UTF-8. Директива #ifdef позволяет компилировать внесенный в данный блок только для определенных операционных систем: кодировка IBM 866 будет для систем семейства Windows, для остальных (#else) используется UTF-8. Изменение кодировки производится с помощью статического метода setCodecForLocale, параметром которого является результат выполнения другой статической функции этого класса codecForName (ее единственный параметр – название кодировки).
2. Вывод информации производится в стандартный поток cout, однако класс QString нельзя без конвертирования вывести в данный поток. Класс имеет метод, переводящий строку QString в строку std::string, однако при этом вновь возникнут проблемы с кодировкой. Поэтому необходимо использовать комбинацию методов toLocal8Bit и data, позволяющие интегрировать строку сначала в массив байтов (QByteArray), а затем в строку типа const char\*. Такой подход позволяет и вывести строку QString в поток cout, и сохранить требуемую кодировку.
3. Комбинация \n, использованная в строковых константах (см. листинг 2.2), является одной из управляющих последовательностей (*escape-последовательностей*), которые можно указывать в строковых выражениях. Эта комбинация обозначает символ с кодом 10 (переход на новую строку). Среди других управляющих последовательностей отметим \\ (символ \), \" (двойная кавычка), \' (одинарная кавычка), \0 (символ с кодом 0), \r (возврат каретки — символ с кодом 13), \b (символ с кодом 8, генерируемый клавишей <Backspace>), \t (табуляция — символ с кодом 9) и \uN (символ Unicode с шестнадцатеричным кодом N). В управляющей последовательности \uN код N должен состоять из 4 цифр и может иметь значение от 0000 до FFFF; например, символ \u0041 обозначает латинскую букву A (десятичный код 65).

2.2. Получение информации

о текущем диске

В начале файла main.cpp подключите заголовочные файлы <QStorageInfo> и <QStringList>, а также блок:

**#ifdef Q\_OS\_WIN32**

**#include <windows.h>**

**#endif**

Добавьте новый метод DInfo (листинг 2.4) и измените метод main (листинг 2.5).

**Листинг 2.4. Метод DInfo**

**void DInfo(QString path)**

**{**

**QString none("---");**

**QString d = path.mid(0, 3);**

**QStorageInfo \*si = new QStorageInfo(d);**

**QString s = QString(" %1").arg(d[0], -4);**

**QStringList dt = { "UNKNOWN", "NO\_ROOT\_DIR", "REMOVABLE", "FIXED", "REMOTE", "CDROM", "RAMDISK" };**

**if (si->isValid())**

**{**

**#ifdef Q\_OS\_WIN32**

**s.append(QString(" %1")**

**.arg(dt.at(GetDriveType((wchar\_t\*)d.utf16())), -12));**

**#else**

**s.append(QString(" %1").arg(none, -4));**

**#endif**

**if (si->isReady())**

**s.append(QString("%L1 %L2").arg(si->bytesTotal() / 1024, 12)**

**.arg(si->bytesFree() / 1024, 12));**

**else**

**s.append(QString("%1 %2").arg(none, 12).arg(none, 12));**

**}**

**else**

**s.append(QString(" %1%2 %3").arg(none, -12).arg(none, 12).arg(none, 12));**

**cout << s.toLocal8Bit().data() << endl;**

**}**

**Листинг 2.5. Новый вариант метода main**

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

#ifdef Q\_OS\_WIN32

QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));

#else

QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));

#endif

cout << QString("Программа DISKINFO\n").toLocal8Bit().data() << endl;

**cout << QString(" Disk Type Size (K) Free (K)\n").toStdString();**

**cout << QString(43, '=').toStdString() << endl;**

**DInfo(a.applicationFilePath());**

cout << QString("\nДля завершения программы нажмите <Enter>...")

.toLocal8Bit().data() << endl;

}

**Результат.** При запуске программы в консольном окне отображается информация о текущем диске (листинг 2.6). Если используется операционная система Windows, выводится также тип диска.

**Листинг 2.6. Содержимое консольного окна при выполнении программы**

Программа DISKINFO

Disk Type Size (K) Free (K)

===========================================

C FIXED 831 842 300 639 503 268

Для завершения программы нажмите <Enter>...

Комментарии

1. Начиная с версии 5.5, в Qt доступен класс QStorageInfo. позволяющий получить «кросс-платформенную» информацию о диске. В её число не входит получение типа диска, он вычисляется только для операционной системы Windows с помощью WinAPI. Словесные описания типов легче всего указать явно в списке строк QStringList.
2. Для доступного диска (свойство isValid) можно определить его размер в байтах (свойство bytesTotal типа qint64), размер свободного пространства в байтах (свойство bytesFree типа qint64). Тип диска вычисляется в функции GetDriveType, для ее вызова необходимо преобразовать строку в тип wchar\*.