УНИВЕРСИТЕТ СИНЕРГИЯ | SYNERGY ACADEMY

МНОГОФАЙЛОВЫЕ ПРОГРАММЫ

МНОГОФАЙЛОВЫЕ ПРОГРАММЫ

В прошлом модуле мы мельком узнали о том, что такое многофайловые программы. Перед тем, как перейти к работе с Qt, давай немного освежим память о том, что это такое.

Многофайловая программа - это программа, которая состоит из нескольких заголовочных и/или ресурсных файлов.

Заголовочный файл - это файл, который содержит в себе объявления функций, без их непосредственной реализации. В C++ такие файлы имеют расширение .h

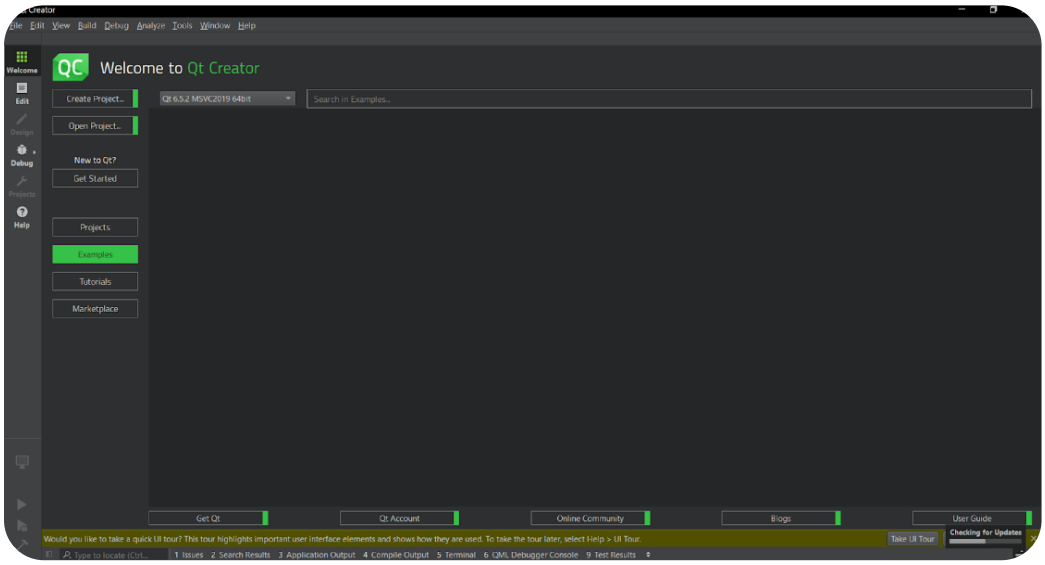
Ресурсные файлы содержат в себе реализацию функций.

Мы можем собирать многофайловую программу вручную, но гораздо более действенным способом будет использование систем сборки. К ним относятся, например, CMake, с которым мы уже знакомы, и qmake, с которым мы познакомимся сегодня.

Система сборки - это набор скриптов, содержащих команды компилятору о том, как собирать наши таргеты (исполняемые файлы, библиотеки и т.д.). Мы пишем только короткие скрипты, которые система сборки уже преобразует в полные команды компилятору.

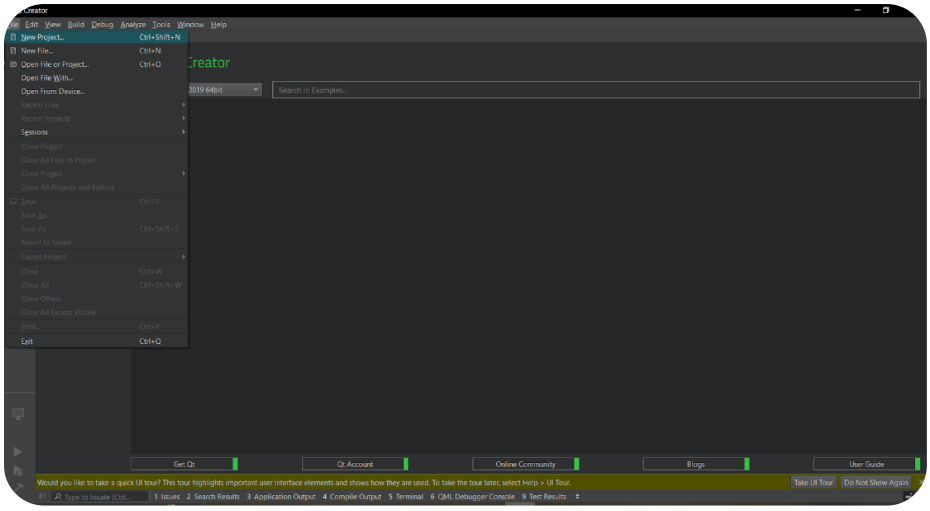
Создание проекта в QT Creator

1. Запускай Qt Creator

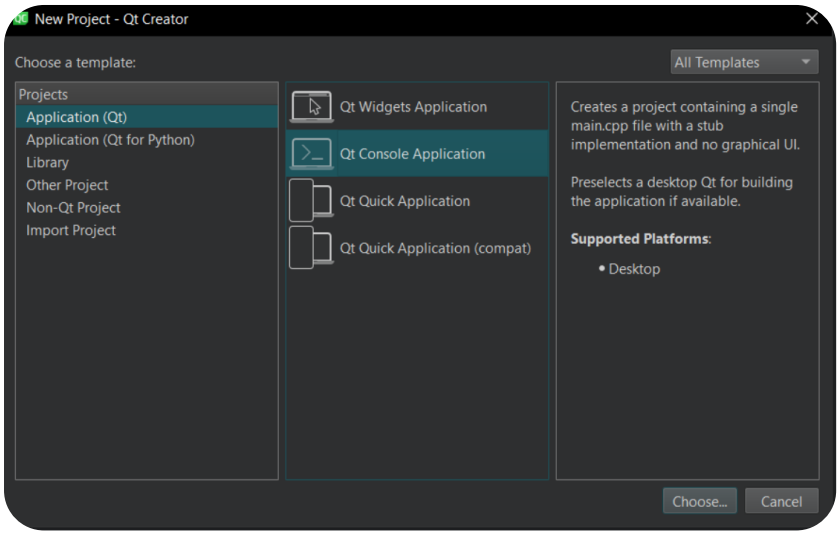


1. Запускай Qt Creator

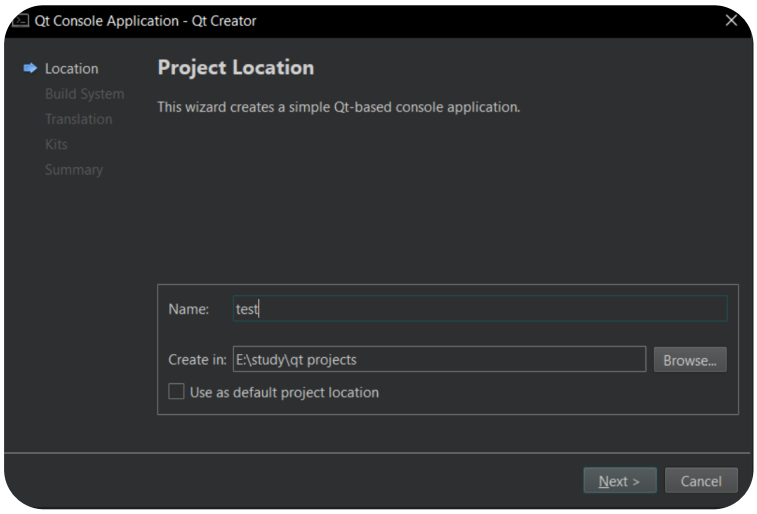
2. Нажми на сочетание клавиш Ctrl+Shift+N или выбери файл -> новый проект.



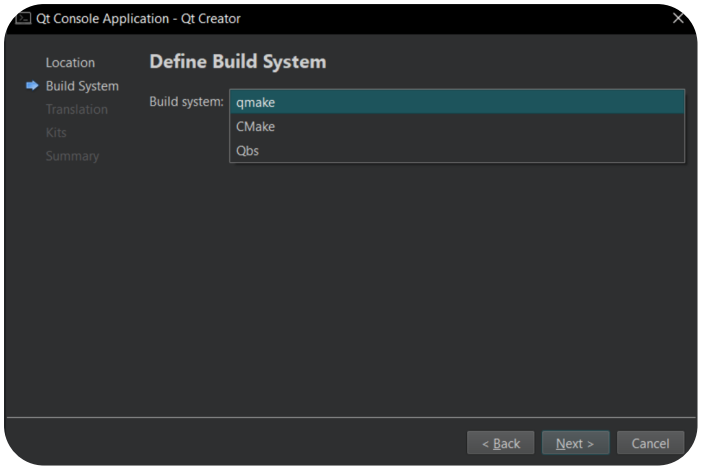
3. Выбери приложение -> консольное приложение QT. Обрати внимание, что у тебя должна быть указана поддерживаемая платформа Desktop. Если графа поддерживаемые платформы пустая, значит, что-то не так. Надо перепроверить, установлен у тебя Qt или только Qt Creator и стоит у тебя MSVC или MinGW. Если всё хорошо, нажми выбрать:



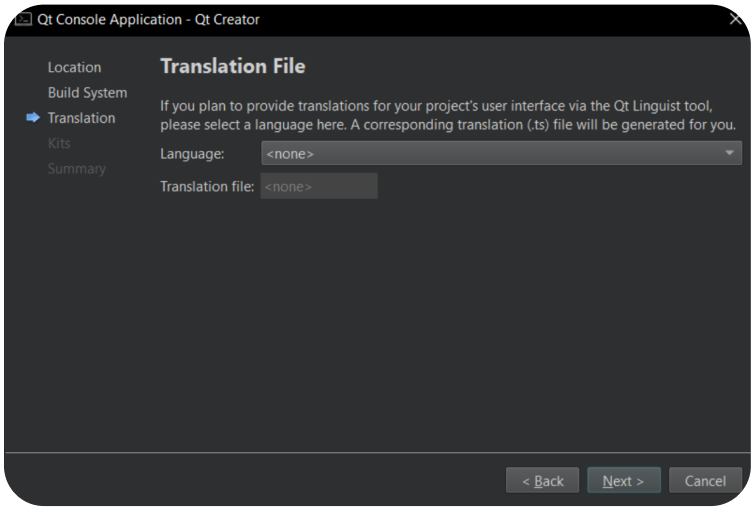
4. Укажи имя программы и папку, где она будет располагаться. Нажми Далее:



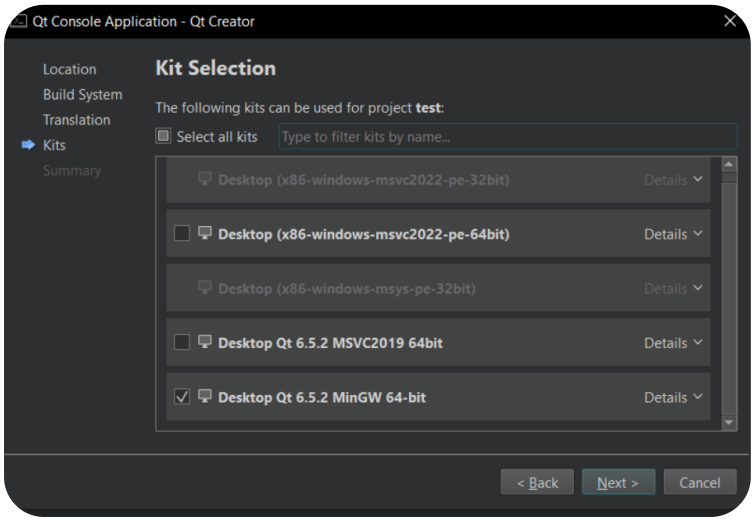
5. Выбери систему сборки qmake. Нажми Далее:

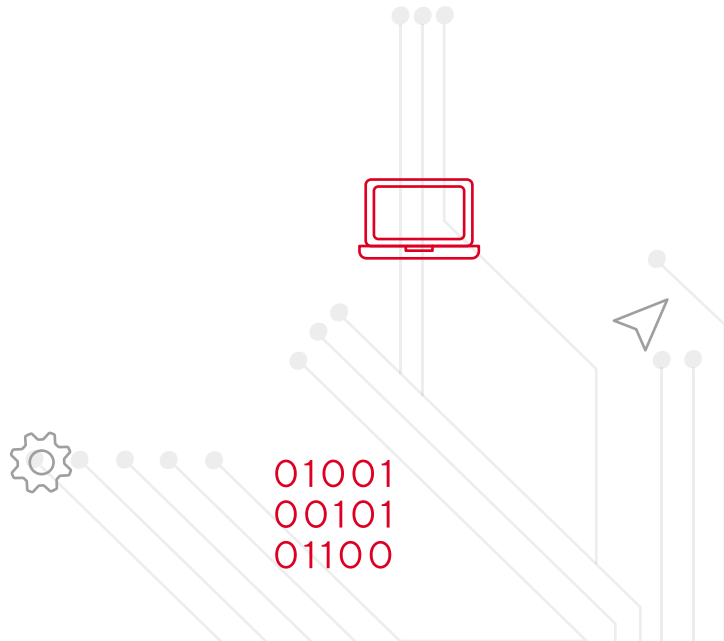


6. Дальше тебе будет предложено добавить перевод для твоей программы с помощью Qt Translation. Нам это пока не нужно (но при желании можешь потыкать и посмотреть что это такое:)), поэтому нажимай Далее:

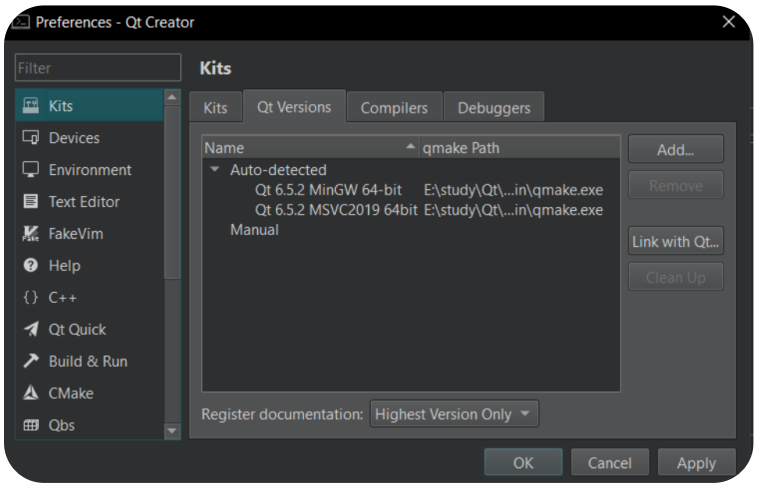


7. Выбери компиляторы. Нам потребуется только один, с остальных стоит снять отметки. Если всё хорошо, то нажми далее. Если программа не даёт тебе возможности выбрать компилятор, посмотри, какие комплекты у тебя есть.

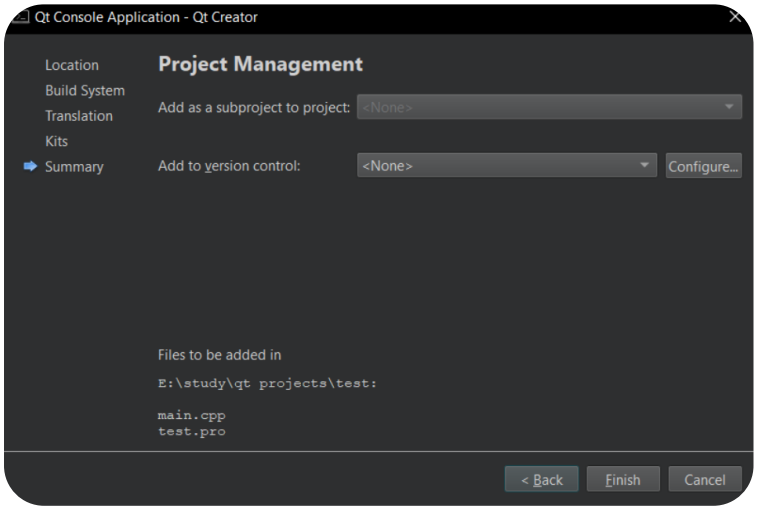




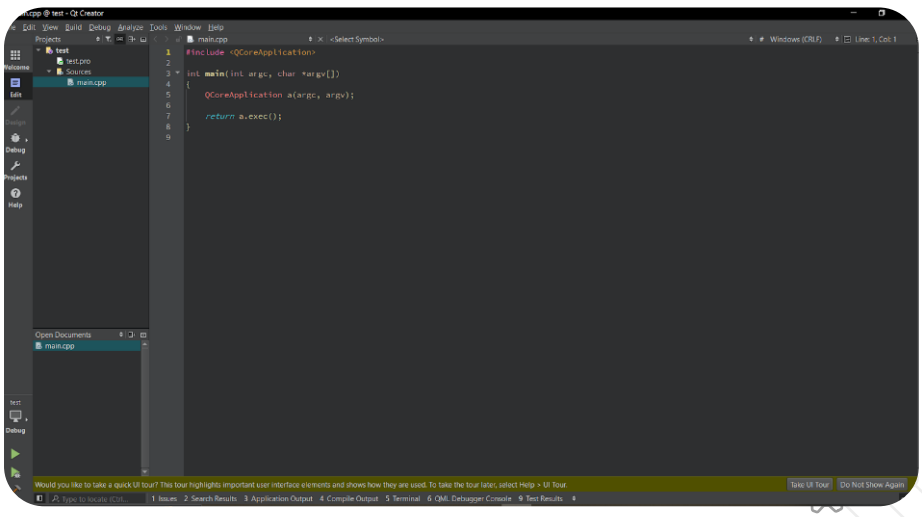
Если что-то не работает, то посмотри в настройках, есть ли у тебя файл qmake.exe. Если qmake нет - что-то было установлено неправильно, можно попытаться доустановить саму систему qt через инсталлятор. Вот здесь должен быть qmake:

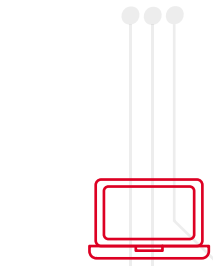


8. Выбери завершить:



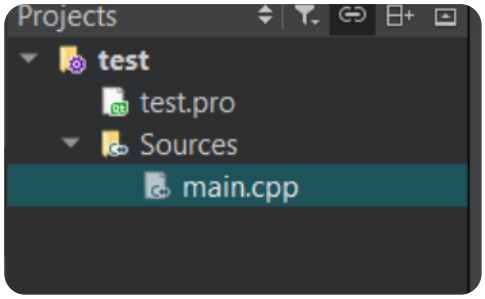
После этих действий у тебя создаётся проект консольного приложения:





Обзор проекта

Слева у нас есть список файлов проекта. Файл название_проекта.pro - это файл с настройками проекта. В папке “Исходники” находится файл main.cpp, в котором содержится исходный код нашей программы:



Внутри main.cpp мы сразу видим автоматически сгенерированный код:

main.cpp <Select Symbol>

```
1  #include <QCoreApplication>
2
3  int main(int argc, char *argv[])
4  {
5      QCoreApplication a(argc, argv);
6
7      return a.exec();
8  }
9
```

QCoreApplication - это специальный класс, с помощью которого мы можем обрабатывать сообщения для консольных приложений QT. В этом классе находится цикл, в котором обрабатываются все сообщения операционной системы и других источников. Этот цикл начинается с вызова метода exec().

exec() - метод, который запускает основной цикл обработки сообщений и ждёт вызова exit(). Данную функцию необходимо вызвать для начала обработки сообщений. Основной цикл обработки сообщений получает сообщения от оконной системы и перенаправляет их виджетам приложения.

exit() - метод, который посылает сообщение о выходе с определённым возвращаемым кодом (returnCode). Обычно, если возвращается 0, то это означает, что программа успешно завершилась. Если возникли какие-либо проблемы или ошибки, возвращается другое число. После вызова этой функции обработчик событий останавливается.

argc и argv - это системные переменные, которые определяют передачу аргументов из командной строки в функцию **main(). argc** (argument count) означает количество строк, которые хранятся в argv (argument vector).

Модуль Qt Core

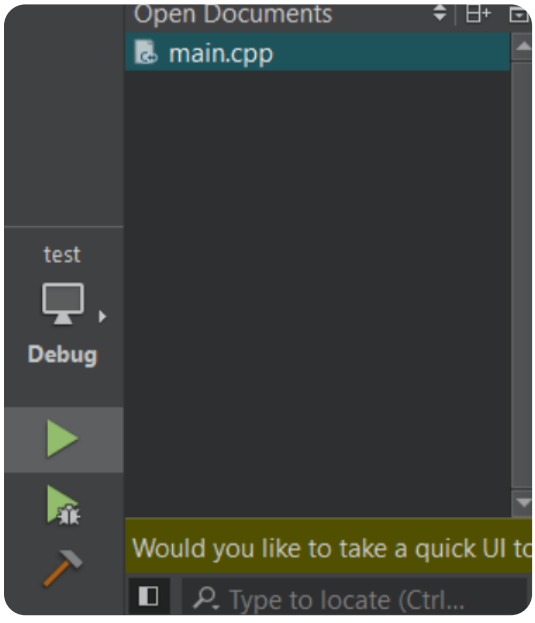
Модуль QtCore содержит ядро не-GUI-функциональности. Все другие модули Qt опираются на этот модуль. Проще говоря, это библиотека, которая содержит очень много нужных и полезных классов, таких как QChar, QString, QEvent, QCoreApplication и другие. О некоторых из этих классов мы поговорим на занятиях, о других ты сможешь прочитать самостоятельно в интернете. Для включения определений классов этого модуля, нужно использовать директиву #include <QtCore>.

Вывод текста на экран

Пока мы можем воспользоваться привычной нам библиотекой iostream и написать первую программу в Qt Creator. Вот он наш любимый HelloWorld:

```
#include <QCoreApplication>
#include<iostream>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    std::cout << "Hello World!" << std::endl;
    return a.exec();
}
```

Чтобы запустить программу, ты можешь нажать на ctrl + s (чтобы сохранить и применить изменения) и ctrl + r (чтобы именно запустить) или нажать на значок запуска слева внизу:



Когда программа скомпилируется, внизу мы увидим вывод в консоль:

```
21:12:21: Starting E:\study\qt projects\build-test-Desktop_Qt_6_5_2_MinGW_64_bit-Debug\debug\test.exe...
Hello World!
```

QString

QString - это строчный тип данных. Вернее, это класс, который позволяет нам создать и обработать Unicode строку из значений типа **QChar** (как ты понимаешь, это символы в Qt).

Инициализация объекта. Есть несколько способов, чтобы создать объект класса QString:

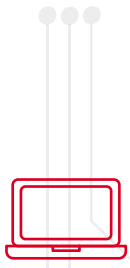
```
// способ 1. Обычное объявление
 QString str = “world”;

// способ 2. Объявление в виде объекта
 QString str2(“hi”);

// способ 3. Преобразование обычной строки C++ в строку C с
последующим её присвоением строке QString
 std::string s1 = “hello”;
 QString str3 = s1.c_str();

// способ 4. Преобразование обычной строки C++ в строку
QString с последующим её присвоением строке QString. data -
указатель на массив символов, size - размер строки
 QString str4 = QString::fromLatin1(s1.data(), s1.size());

// способ 5. Создание объекта QString с помощью конструктора,
который принимает на вход классическую строку C
char s2[] = "string";
QString str5(s2);
```

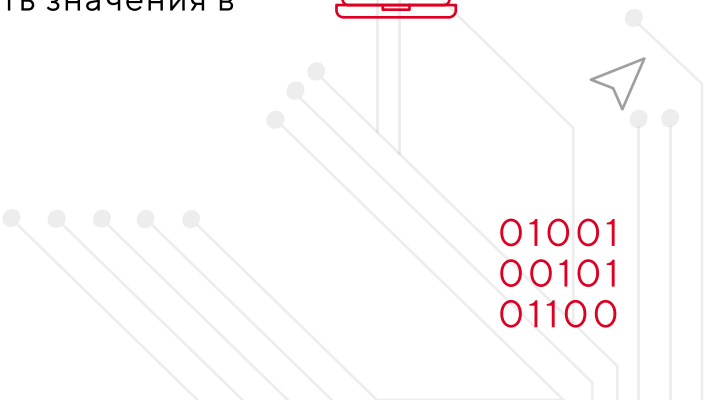
Доступ к элементам строки. Чтобы получить доступ к конкретному элементу, мы можем воспользоваться методом at() или квадратными скобками []:

```
QChar q = str.at(3);
 q = str[1];
```

Метод append(). Используется, чтобы добавить значения в конец строки:

```
str.append(“!!!”);
```





Метод prepend(). Используется, чтобы добавить значения в начало строки:

```
str.prepend(“Hello ”);
```

Метод remove(). Используется, чтобы удалить n символов, начиная с m-го символа строки:

```
str.remove(m, n);
```

Метод chop(). Используется, чтобы удалить n последних символов из строки:

```
str.chop(n);
```

Метод replace(). Используется, чтобы заменить n символов, начиная с m, символами str2:

```
str.replace(m, n, str2);
```

Метод clear(). Используется, чтобы очистить строку:

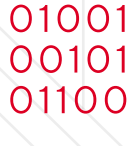
```
str.clear();
```

Метод toUpper(). Возвращает копию строки, где все буквы заменены на заглавные.

Метод toLower(). Возвращает копию строки, где все буквы заменены на строчные.

Методы определения длины строки. Их существует 3: length(), count(), size().

Метод isEmpty(). Возвращает true, если строка пустая.



Форматирование (построение) строк. Мы можем выбрать место для определённого значения в строке и подставить само значение позже с помощью маркера вставки значения % и функции arg():

```
QString str = “%1, %2 – это значения”;
 int a = 10;
 char b = ‘a’;
 QString str2 = str.arg(a).arg(b);
```

Метод left(). Возвращает подстроку длиной в указанное количество символов с начала:

```
QString str = “Hello world!”;
 QString str2 = str.left(3);
```

Метод right(). Возвращает подстроку длиной в указанное количество символов с конца:

```
QString str = “Hello world!”;
 QString str2 = str.right(5);
```

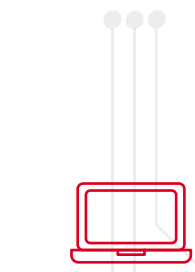
Метод mid(). Возвращает подстроку длиной в указанное количество символов, начиная от определённой позиции:

```
QString str = “Hello world!”;
 QString str2 = str.mid(6, 3);
```

Метод compare(). Сравнивает две строки и возвращает:

- 0, если строки совпадают
- Значение меньше 0, если первая строка меньше второй
- Значение больше 0, если первая строка больше второй

Существует ещё множество других методов. Их ты можешь изучить самостоятельно.



QTextStream

Если ты помнишь, как работать с файлами, то здесь у тебя проблем не возникнет.

QTextStream - это класс, с помощью которого мы можем считывать текст и выводить его на экран. Чтобы работать с этим классом, нужно создать объект этого класса и связать его со стандартным выводом:

```
QTextStream out(stdout);
 out << “Hi”;
```

Или со стандартным вводом:

```
QTextStream in(stdin);
QString line, line2;
line = in.readLine();
in >> line2;
```

Метод flush(). Нужен, чтобы “вытолкнуть” данные, которые должны быть записаны, из буфера.