ФГБОУ ВПО «Воронежский государственный технический университет»
Кафедра компьютерных интеллектуальных технологий проектирования

286-2014

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 1-3 по дисциплине “Объектно-ориентированное программирование” для студентов направления 230100.62 «Информатика и вычислительная техника» (профиль «Системы автоматизированного проектирования в машиностроении») очной формы обучения

Воронеж 2014

Составители: канд. техн. наук А.Н. Юров,
канд. техн. наук М.В. Паринов,
ст. преп. В.А. Рыжков,
ст. преп. А.А. Филимонова

УДК 004.9

Методические указания к лабораторным работам № 1-3 по дисциплине “Объектно-ориентированное программмирование” для студентов направления 230100.62 «Информатика и вычислительная техника» очной формы обучения / ФГБОУ ВПО «Воронежский государственный технический университет»; сост. А.Н. Юров, М.В. Паринов, В.А. Рыжков, А.А. Филимонова. Воронеж, 2014. 23 с.

Методические указания содержат материал по созданию кроссплатформенных приложений в среде QT, а также практические задачи и перечень заданий для выполнения лабораторных работ по дисциплине «Объектно-ориентированное программирование».

Предназначены для студентов 1,2 курсов.

Методические указания подготовлены в электронном виде в тектовом редакторе MS Word 2010 и содержатся в файле МУ 2014_1.docx.

Табл. 3. Ил. 11. Библиогр.: 10 назв.

Рецензент канд. физ.-мат. наук, доц. Н.А. Тюкачев.

Ответственный за выпуск зав. кафедрой д-р техн. наук, проф. М.И. Чижов

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ВВЕДЕНИЕ

Создание “универсальных” приложений, которые с успехом выполняются на персональных ЭВМ и мобильных устройствах является весьма перспективным направлением в области разработки программного обеспечения. При этом нет необходимости тратить время на локализацию и адаптацию приложения к тем или иным тонкостям работы в операционных системах, которые значительно разняться между собой как из-за исторических предпосылок создания и развития, так и особенностей аппаратной части вычислительных систем, где они выполняются. Инструментальных средств разработки программных средств, исходя из вышесказанного, на данный момент немного, однако это направление активно развивается, в том числе и на некоммерческой основе. Одним из таких решений является кроссплатформенный инструментарий QT, в котором присутствует не только набор из множества компонентов, позволяющий создать практически любой пользовательский интерфейс, но и мощный редактор с развитой структурой управления проектами (QT Creator), инструменты по созданию интерфейса, управления и связями элементов в пределах приложения (QT Designer). Кроме того, реализована развитая система помощи по документации, а также огромное количество готовых примеров, проектов сборок под различные вычислительные платформы.

В методических указаниях представлен материал по созданию приложений в среде QT как для разработки программ в консольном режиме, так и с графическим интерфейсом с использованием QT SDK. Реализация проектов позволит усвоить концепции объектно-ориентированного программирования на практике. Все примеры могут быть апробированы на известных операционных системах: Windows, Linux, Mac OS X, Android и ряда других.

ЛАБОРАТОРНАЯ РАБОТА № 1
ИНСТРУМЕНТАЛЬНАЯ СРЕДА РАЗРАБОТКИ QT CREATOR, СОЗДАНИЕ КОНСОЛЬНОГО ПРОЕКТА

Цель работы: разработать консольное приложение в среде QT согласно заданию.

Задачи и требования к выполнению:

1.Изучить среду разработки QT Creator, знать особенности установки QT SDK.

2.Изучить структуру консольного проекта, подготовленного средствами IDE QT.

3.Собрать консольный проект в QT Creator, ознакомиться с отладочным режимом работы приложения.

Теоретические сведения

Инструментальные средства QT (читается, как кьют) имеют следующие отличительные преимущества перед иными решениями по созданию программных средств (ПС):

-кроссплатформенность;

-быстрота разработки графического интерфейса пользователя (GUI);

-логичная структура построения классов;

-собственная среда проектирования (возможность встраивания);

-подробное документирование и обширная база примеров;

-поддержка набора компиляторов, сборщиков, а также отладочных средств сторонних производителей программного обеспечения (ПО);

-активная техническая поддержка.

На рис. 1 представлены оконные интерфейсы среды разработки QT.

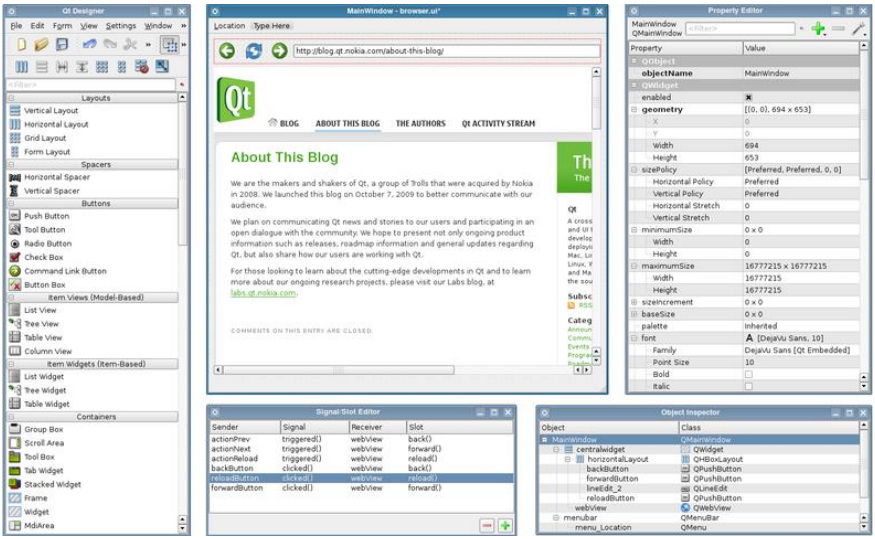

Рис. 1. Оконные интерфейсы среды QT

Перед началом использования QT SDK, необходимо выполнить установку необходимого инструментария на вычислительной системе пользователя. Установку требуемого программного обеспечения можно выполнить со следующего адреса сети Интернет, а именно www.qt-project.org/downloads (рис. 2).

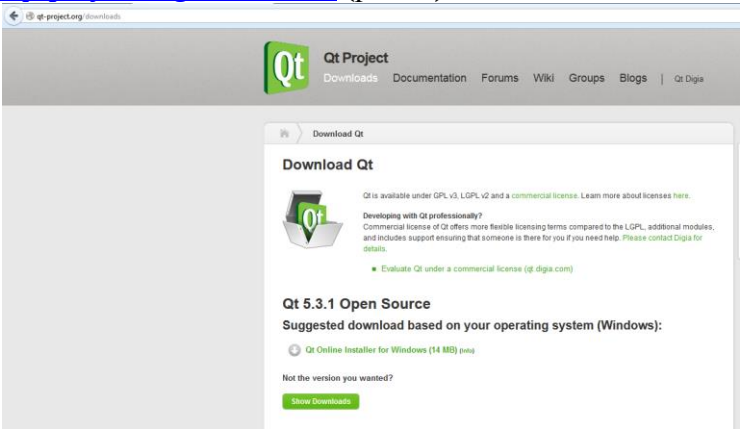

Рис. 2. Ресурс для получения инсталлятора QT

Если по каким-либо причинам указанный сайт доступен не будет, рекомендуется воспользоваться поиском в любом браузере, который присутствует в операционной системе (ОС) учащегося. По указанному адресу будут доступны ссылки на инсталляторы QT под ряд ОС. Необходимый инсталлятор QT под Windows имеет онлайн-установку в системе, однако есть установщики, которые не обращаются в сеть интернет и имеют определенный профиль под использование того или иного компилятора (например, MinGW), а также надстройки к сторонним средам разработки (например, Visual Studio 2013). В онлайн-установщике для ОС Windows необходимо выполнить следующие действия:

-указать путь установки инструментальных средств разработки (рис. 3);

-выбрать профили сборки и компиляторы, как показано на рис. 4;

-указать тип используемых лицензий (рис. 5);

-приступить к установке QT SDK (рис. 6).

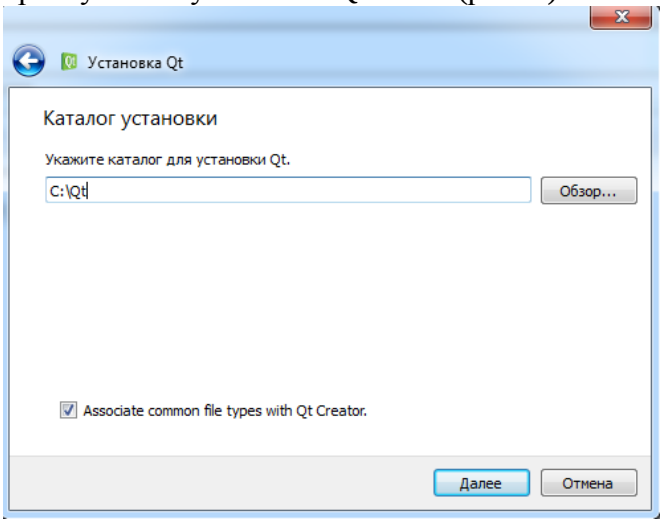

Рис. 3. Определение путей установки IDE QT

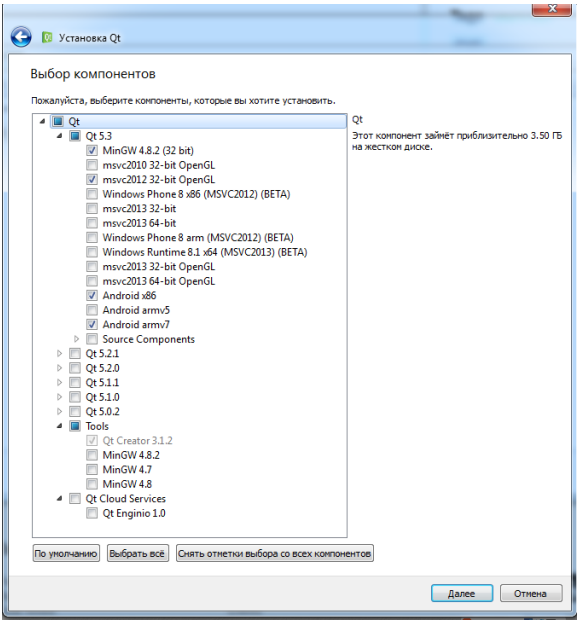

Рис. 4. Требуемые инструменты QT для разработки приложений

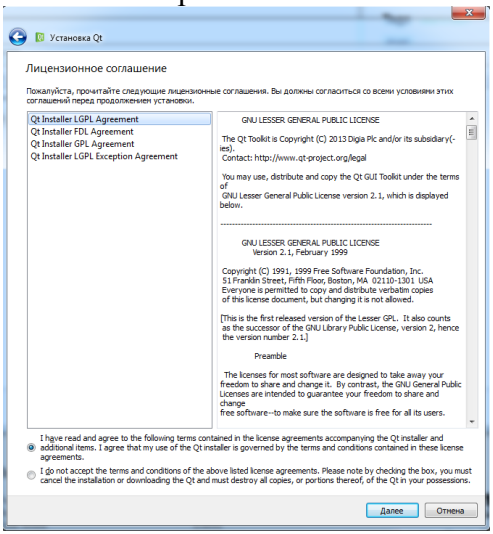

Рис. 5. Выбор лицензионного соглашения на использование программных средств разработки QT

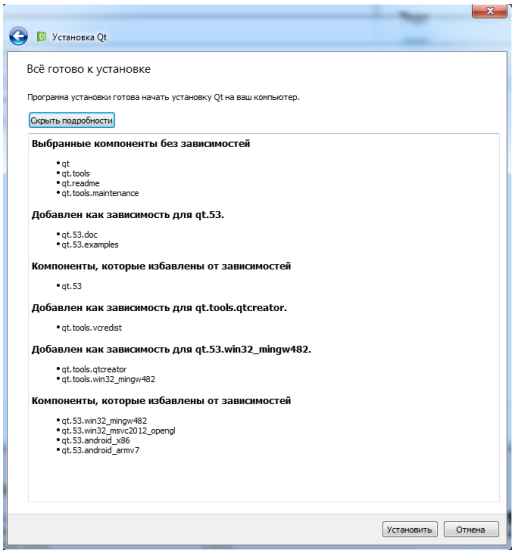

Рис. 6. Переход к автоматическому развёртыванию среды QT

После завершения установки необходимо запустить среду QT Creator, в которой и будет производиться разработка программ с использованием классов QT. Стартовый экран QT Creator показан на рис. 7, где приведен базовый интерфейс управления, который необходим разработчику.

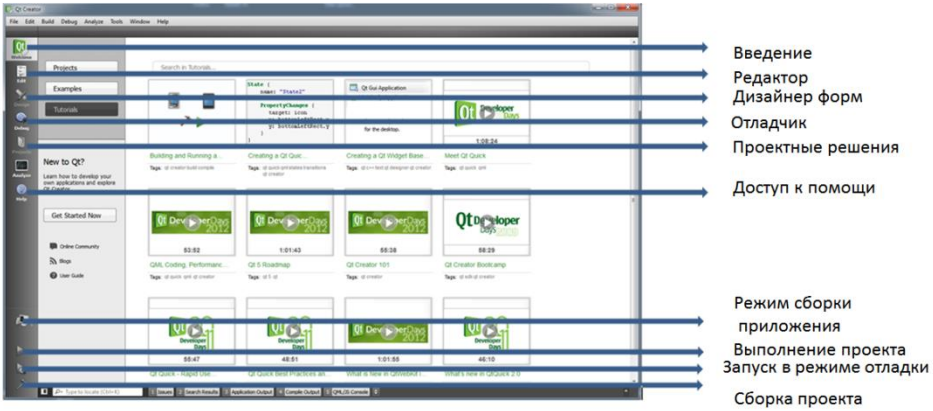

Рис. 7. Элементы управления QT Creator

Кроме того, возможно придется выполнить ряд настроек среды по сборке и запуску проектов QT- для этих целей требуется обратиться к документации или форум по конфигурированию системы. Настройки и установленные компоненты представлены рис. 8.

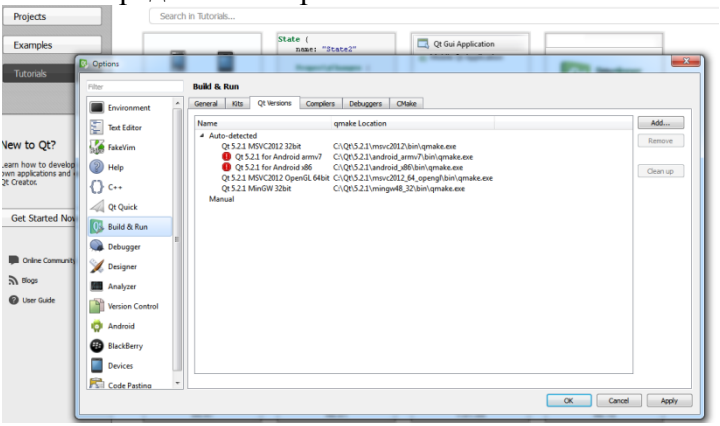

Рис. 8. Настройка QT Creator

Синтаксис классов QT построен на языке C++, при этом расширяет его возможности при разработке интерфейса, а также взаимодействии элементов управления между собой.

Перечень ключевых слов, конструкций C++, а также директив предпроцессора, необходимых для выполнения работы:

- cin>> потоковый ввод;
- cout<< потоковый вывод;
- if (), switch()/case, ? условные конструкции;
- for(), while, do…while операторы циклов;
- int my_array[10], double randValue []={1.1,2.2} объявления массивов;
- && логическое И, || логическое ИЛИ;
- #include <> подключение заголовочного файла using namespace использование пространства имен

Листинг 1. Консольное приложение на QT

Файл проекта:

```
QT      += core
QT      -= gui //графический интерфейс не используется
TARGET = firstgui
CONFIG   += console
CONFIG   -= app_bundle
TEMPLATE = app
SOURCES += main.cpp
```

Файл с программой:

```
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    return a.exec();
}
```

Для каждого Qt-класса имеется заголовочный файл с тем же именем (с учетом регистра), содержащий определение этого класса. Объект QApplication нужен для управления всеми ресурсами приложения. Для конструктора QApplication необходимо указывать параметры argc и argv, поскольку Qt сама обрабатывает некоторые из аргументов командной строки.

Последняя строка (return…) обеспечивает передачу управления приложением Qt. В этом месте программа переходит в цикл обработки событий, то есть в своего рода режим «простоя», ожидая со стороны пользователя таких действий, как щелчок мыши или нажатие клавиши на клавиатуре.

Задания на самостоятельную работу:

1. Разработать консольный проект, в котором выполнить вычисление: скорости, времени, расстояния.

2. Разработать консольный проект, в котором по формуле Герона ($S = \frac{\sqrt{(-1)(a^4+b^4+c^4)+2(a^2b^2+b^2c^2+a^2c^2)}}{4}$) вычислить площадь треугольника.

3.Написать приложение по вычислению корней квадратного уравнения.

ЛАБОРАТОРНАЯ РАБОТА № 2
НАСЛЕДОВАНИЕ, ВИРТУАЛЬНЫЕ ФУНКЦИИ, ОТЛАДОЧНЫЕ СРЕДСТВА СРЕДЫ РАЗРАБОТКИ

Цель работы: разработать консольное приложение в среде QT согласно заданию.

Задачи и требования к выполнению:

1.Изучить особенности наследования классов C++.

2.Изучить приемы работы с виртуальными функциями.

3.Изучить имеющиеся в среде отладочные возможности проектов.

4.Ознакомиться со структурой файла .pro

Теоретические сведения

Наследование-это подход по разработке нового класса на основе уже существующего. В языке C++ имеется возможность в качестве базовых задать несколько классов. В таком случае производный класс наследует методы и атрибуты всех его родителей. На рис. 9 показана схема, в которой показаны взаимосвязи между классами, а на рис. 10-подготовленное в QT Creator приложение с использованием наследования.

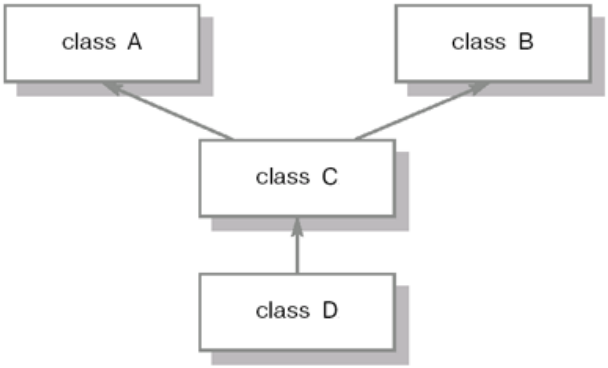

Рис. 9. Схема взаимосвязей между классами

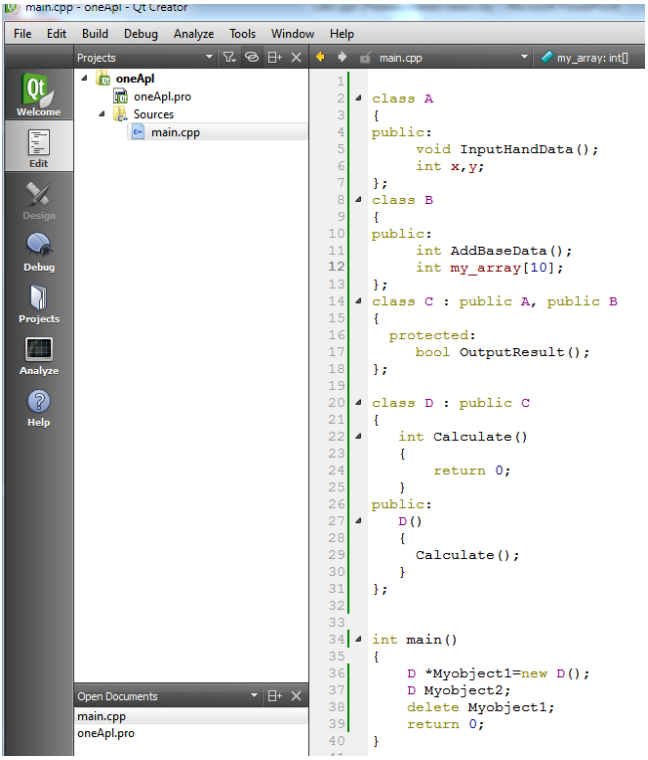

Рис. 10. Программная реализация механизма наследования

Виртуальная функция – это функция, объявленная с ключевым словом virtual в базовом классе и переопределенная в одном или в нескольких производных классах. Виртуальные функции являются особыми функциями, потому что при вызове объекта производного класса с помощью указателя или ссылки на него C++ определяет во время исполнения программы, какую функцию вызвать, основываясь на типе объекта. Для разных объектов вызываются разные версии одной и той же виртуальной функции. Пример по использованию виртуальных функций представлен листингом 2.

Листинг 2. Консольное приложение на QT.

```
#include <iostream>
using namespace std;
class Base {
public:
virtual void who()
{ // определение виртуальной функции
cout << "Base\n";
}
};
class first: public Base {
public:
void who() {
// определение who() применительно к first_d
cout << "First derivation\n";
}
};
class second: public Base {
public:
void who() {
// определение who() применительно к second_d
cout << "Second derivation\n";
}
};
int main()
{
Base base_obj;
```

```
Base *p;
first first_obj;
second second_obj;
p = &base_obj;
p->who(); // доступ к who класса Base
p = &first_obj;
p->who(); // доступ к who класса first_d
p = &second_obj;
p->who(); // доступ к who класса second_d
cin.get();
return 0;
}
```

Отладочные средства в среде QT Creator

Qt Creator не имеет собственного отладчика. Вместо этого он предоставляет графический интерфейс к различным отладчикам. Для Windows это GNU Symbolic Debugger (gdb).

Для начала процесса отладки требуется установить точку останова на строке, где требуется посмотреть промежуточные результаты работы ПО, щёлкнув между номером строки и границей окна (рисунок 11). Затем надо выбрать “Начать отладку” из меню “Отладка” или нажать **F5.**

Чтобы закончить отладку надо нажать **Shift+F5**. Строка кода может быть исполнена как одно целое с **F10**; чтобы выполнить функцию или подфункцию используйте **F11.** Можно продолжить выполнение программы с помощью **F5.** Допускается выполнение программы до окончания текущей функции или перепрыгнуть на произвольную позицию в текущей функции.

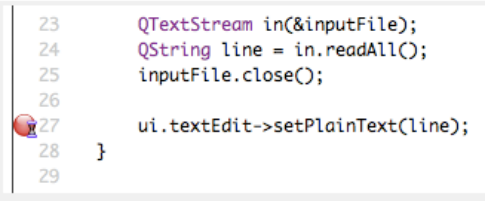

Рис. 11. Точка остановки в листинге приложения

Проектный файл в приложении .pro QT

Основные блоки, отвечающие за настройку проекта и включения директив компилятора, приведены в табл. 1

Элементы-блоки в проектном файле QT Таблица 1

Переменная	Содержимое
CONFIG	Настройки общей конфигурации проекта
DESTDIR	Директория, в которую будет помещен исполняемый или двоичный файл
FORMS	Список файлов форм (.ui), обрабатываемых uic
HEADERS	Список имен заголовочных файлов (.h), используемых для постройки проекта
QT	Qt-специфичные настройки конфигурации
RESOURCES	Список файлов ресурсов (.rc), которые должны быть включены в проект
SOURCES	Список файлов реализации, используемых для постройки проекта
TEMPLATE	Шаблон, используемый для постройки проекта. Определяет, будет ли построенный проект приложением, библиотекой или плагином

В качестве примера можно привести поддержку условных структур, которая выполнена с помощью

специальных инструкций, аналогично работающими подобно условиям if в языках программирования:

win32 { SOURCES += paintwidget_win.cpp }

Общее конфигурирование проекта

Режимы конфигурирования представлены в табл. 2

Режимы сборки проекта Таблица 2

Опция	Описание
release	Проект строится в режиме полноценного окончательного варианта. Игнорируется, если также определена опция debug
debug	Проект строится в режиме отладки.
warn_on	Компилятор должен вывести, насколько возможно, много сообщений. Игнорируется, если также определена опция warn_off
warn_off	Компилятор должен вывести, насколько возможно, мало сообщений

Возможный режим сборки может быть указан следующим образом: CONFIG(debug, debug|release) { message(Building in debug mode.) } else { message(Building in release mode. Ignoring debug if it is available.) }

Декларация библиотек Qt

Внесение компонентов Qt в проектный файл может быть выполнено добавлением опций посредством операции “+=”.

В табл. 3 показаны опции добавления с именем подключаемого модуля. Особенности модуля по содержанию классов можно узнать из документации по QT SDK.

Таблица 3

Объявление библиотек перед их использованием

Опция	Особенности
core (включается по умолчанию)	модуль QtCore
gui (включается по умолчанию)	модуль QtGui
network	модуль QtNetwork
opengl	модуль QtOpenGL
sql	модуль QtSql
svg	модуль QtSvg
xml	модуль QtXml
qt3support	модуль Qt3Support

Возможная конфигурация по подключению объявлений выполняется следующим образом:

```
CONFIG += qt
 QT += network xml
 QT -= gui # Используется только модуль ядра.
QMAKE_CXXFLAGS+=-std=c++0x #подключение
```

Стандарта С++11 в про файле

Задания на самостоятельную работу:

1. Разработать консольный проект, в котором выполнить набор текста, следом определить в нем количество цифр. Решение задачи построить с использованием механизма наследования.

2.Используя механизм виртуальных функций подготовить реализацию по расчету площадей плоских геометрических фигур (взять выпуклый многоугольник, трапецию, круг).

ЛАБОРАТОРНАЯ РАБОТА № 3
ШАБЛОННЫЕ ФУНКЦИИ И КЛАССЫ

Цель работы: разработать консольное приложение в среде QT согласно заданию.

Задачи и требования к выполнению:

1.Изучить работу шаблонных функций.

2.Изучить приемы использования шаблонных классов.

Теоретические сведения

Функция-шаблон определяет общий набор операций, который будет применен к данным различных типов. Используя этот механизм, можно применять некоторые общие алгоритмы к широкому кругу данных.

После этого компилятор автоматически генерирует корректный код для того типа данных, для которого создается данная конкретная реализация функции на этапе компиляции.

По существу, когда создается функция- шаблон, создается функция, которая может автоматически перегружать сама себя.

Функция-шаблон может быть определена следующим образом:

template <class имя> возвращаемый_тип имя_функции (список параметров) {// тело функции}

Реализация прототипа функции выглядит так:

template <class X> void changeValue(X &var1, X &var2);

Пример по использованию функции-шаблона представлен листингом 3, которая реализует обмен двух переменных (значений в них), представленных целочисленным, вещественным и символьным типами данных.

Листинг 3. Консольное приложение на QT.

```
#include <QCoreApplication>
#include <iostream>
using namespace std;

template <class X> void changeValue(X &var1, X &var2);
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    int i=10, j = 20;
    float x=10.1,y=23.3;
    char b1='x', b2='z';
    cout << "Original i, j: " << i << ' ' << j <<
endl;
    cout << "Original x, y: " << x << ' ' << y <<
endl;
    cout << "Original a, b: " << b1 << ' ' << b2 <<
endl;
    changeValue(i, j); // обмен целых
    changeValue(x, y); // обмен вещественных значений
    changeValue(b1, b2); // обмен символов
    cout << "Swapped i, j : " << i << ' ' << j <<
endl;
    cout << "Swapped x, y: " << x << ' ' << y << endl;
    cout << "Swapped a, b: " << b1 << ' ' << b2 <<
endl;
    return a.exec();
}
template <class X> void changeValue(X &var1, X &var2)
{    X temp;
    temp = var1;
    var1 = var2;
    var2 = temp;
    }
```

Допускается определение функций-шаблонов с использованием нескольких типов данных в инструкции template, используя список с запятыми в качестве разделителя. Например, следующая программа создает функцию-шаблон, имеющую два типа-шаблона (листинг 4).

Листинг 4. Консольное приложение на QT.

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

using namespace std;

template <class type1, class type2> void myfunc(type1
x, type2 y)
{
 cout << x << ' ' << y << endl;
}
int main()
{
myfunc(10, "hi");
myfunc(0.23, 10L);
return 0;}
```

В примере типы-шаблоны type1 и type2 заменяются компилятором на типы int, char*, double и long соответственно, причем компилятор создает два различных экземпляра функции myfunc() в функции main().

Допускается явное переопределение функий-шаблонов. В листинге 5 показана данная реализация:

Листинг 5. Консольное приложение на QT.

```
// переопределение функции-шаблона
template <class X> void swap(X &a, X &b)
{
X temp;
temp = a;
a = b;
b = temp;
```

```
}
// обобщенная версия swap()
void swap(int &af int &b)
{
int temp;
temp = a;
a = b;
b = temp;
cout << "Inside overloaded swap(int &, int
&).\n";
}
```

Имеются некоторые ограничения по использованию функций-шаблонов. Функции-шаблоны сходны с перегруженными функциями, за исключением того, что они имеют некоторые ограничения. Для перегруженных функций можно выполнять различные действия в теле каждой функции. В отличие от этого, для функции-шаблона необходимо выполнять одни и те же общие действия, и только тип данных может быть различным. Другим ограничением на функции-шаблоны является то, что виртуальная функция не может быть функцией-шаблоном.

Кроме функций-шаблонов можно также определить классы-шаблоны. Для этого следует создать класс, определяющий все алгоритмы, но фактический тип данных является параметром, определяющимся при создании класса.

Классы-шаблоны полезны тогда, когда класс содержит логику, допускающую значительные обобщения. Компилятор автоматически создаст корректный код, основываясь на типе данных, указанном перед компиляцией. Общая форма объявления класса-шаблона показана ниже:

```
template < class  Тип> class имя_класса
{
//...
}
```

На листинге 6 приведен пример приложения, в котором показана работа со стеком.

Листинг 6. Консольное приложение на QT.

```
#include <QCoreApplication>
#include <iostream>
using namespace std;
template <class Type> class stack
{
private:
  int top;
  Type s[10];
public:
  stack ()//: top(0)
  {top=0;}
void push(Type var)
  {
    top++;
    s[top] = var;
  }
  Type pop();
};
template <class Type> Type  stack <Type>::pop()
{
  Type var = s[top];
  top--;
  return var;
}
int main()
{
    stack<int> s1;
    stack<float> s2;
    s1.push(3);
     cout<<s1.pop()<<endl;
    s1.push(2);
     cout<<s1.pop()<<endl;
    s2.push(0.5);
     cout<<s2.pop()<<endl;
    return 0;
}
```

Задания на самостоятельную работу:

1. Написать приложение с использованием класса-шаблона, позволяющее конвертировать валюты ряда государств.
2. Подготовить функцию-шаблон, в которой осуществляется сортировка массива данных по возрастанию, представленных типом double и int.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Прата С. Язык программирования С++. Лекции и упражнения / С. Прата. 5-е изд. – М.: ООО ”И.Д. Вильямс”, 2007. – 1184 с.
2. Страуструп Б Язык программирования С++ / Б. Страуструп. - М.: Бином, 2011. – 1136 с.
3. Шилдт Г. С++ Базовый курс / Г. Шилдт. 3-е изд. – М.: ООО ”И.Д. Вильямс”, 2010. – 624 с.
4. Бланшет Ж. QT4: программирование GUI на С++ /Ж. Бланшет, М. Саммерфилд. - М.: Кудиц-Пресс, 2007. - 641 с.
5. Иванова Г.С. Создание пользовательских интерфейсов в программах на С++ с использованием библиотеки QT: учеб. пособие Г. С. Иванова. - М. : МГТУ имени Н.Э. Баумана, 2011. - 52 с.
6. Шлее М. Qt 4.8 / Профессиональное программирование на С++/ М. Шлее. -СПб.: БХВ-Петербург, 2012. - 912 с.
7. Roberge J. A laboratory course in C++ structures. 2ed / J. Roberge, S. Brandl, D. Whittington. Jones and Bartlett, 2003. - 411 p.
8. London J. Modeling Derivatives in C++ / London J. Wiley, 2005. - 841p.
9. Документация библиотеки Qt [Электронный ресурс]. – Режим доступа: http://qt-project.org/doc/
10. Документация библиотеки Qt [Электронный ресурс]. – Режим доступа: http://qt-doc.ru/

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 1-3 по дисциплине
“ Объектно-ориентированное программирование ” для студентов направления 230100.62 «Информатика и вычислительная техника» очной формы обучения

Составители:
Юров Алексей Николаевич
Паринов Максим Викторович
Рыжков Владимир Анатольевич
Филимонова Анастасия Анатольевна

В авторской редакции

Компьютерный набор А.Н. Юрова

Подписано к изданию 21.11.2014.
Уч.-изд. л. 1,4. «С»

ФГБОУ ВПО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14