

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Р.Е.АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий

Кафедра «Информационные радиосистемы»

## **Приобретение практических навыков работы в интегрированной среде разработки**

Методические указания к лабораторной работе  
по дисциплине «Информационные технологии» для студентов  
направления подготовки бакалавра 210400 «Радиотехника»  
дневной формы обучения

Нижний Новгород  
2011

Составители: Е.Н.Приблудова, С.Б.Сидоров

УДК 519.6

Приобретение практических навыков работы в интегрированной среде разработки: метод. указания к лаб. работам по дисциплине «Информационные технологии» для студентов направления подготовки бакалавра 210400 «Радиотехника» дневной формы обучения / НГТУ; сост.: Е.Н.Приблудова, С.Б. Сидоров. Н. Новгород, 2011.– 16 с.

Изложены краткие сведения о работе в среде KDevelop под управлением операционной системой GNU/Linux. Рассмотрены вопросы ее использования для разработки программ на языке программирования С. Сформулированы задания и порядок их выполнения. Приведен перечень сообщений о синтаксических ошибках времени компиляции.

Редактор Э.Б.Абросимова

Подп. к печ. 22.07.2011. Формат 60x84<sup>1</sup>/<sub>16</sub>. Бумага офсетная.

Печать офсетная. Усл. печ. л. 1. Уч.-изд. л. 0,8. Тираж 300 экз. Заказ .

---

Нижегородский государственный технический университет им. Р.Е. Алексеева.  
Типография НГТУ.

Адрес университета и полиграфического предприятия:  
603950, г. Нижний Новгород, ул. Минина, 24.

© Нижегородский государственный технический университет, 2011

© Приблудова Е.Н., Сидоров С.Б., 2011

## 1. Цель работы

Приобретение начальных навыков работы в среде **KDevelop** для разработки программ на языке программирования C. Практическое изучение методики разработки и отладки простых программ.

## 2. Краткие сведения

### 2.1. Общая схема запуска среды разработки **KDevelop**

Выполнение лабораторных работ производится с использованием среды разработки программ **KDevelop**. Данная среда разработки выполняется под управлением операционной системы GNU/Linux в графической оболочке KDE. Запуск среды осуществляется в несколько этапов:

- загрузка операционной системы GNU/Linux;
- регистрация пользователя в операционной системе;
- запуск графической оболочки KDE;
- запуск программы **KDevelop**.

### 2.2. Загрузка GNU/Linux и регистрация пользователя

Процесс загрузки операционной системы инициируется специальным загрузчиком. После включения компьютера, в ответ на приглашение выбрать операционную систему для загрузки необходимо выбрать пункт меню *Linux* и нажать клавишу «*Enter*». При этом происходит загрузка образа ядра операционной системы в оперативную память, после чего ему передается управление и выполняется загрузка остальных компонент операционной системы *GNU/Linux*.

Особенностью операционной системы GNU/Linux является возможность одновременной работы в ней нескольких пользователей. Поэтому каждый раз в начале работы с системой пользователь должен в ней **зарегистрироваться**. Процесс регистрации позволяет системе идентифицировать пользователя. В дальнейшем система будет иметь возможность взаимодействовать с пользователем отдельно, предоставлять ему по запросу ресурсы в соответствии с правами доступа данного пользователя, ограничивать по желанию доступ к его данным для других пользователей и т.д.

По завершении загрузки системы появится диалоговое окно с запросами `login` (имя пользователя) и `password` (пароль пользователя). Имя и пароль пользователя необходимо оставить по умолчанию, нажать кнопку **ОК**.

### 2.3. Работа в графической оболочке KDE

KDE – графическая оболочка, предназначенная для использования на рабочих станциях Unix-систем. С точки зрения пользователя, KDE представляет

собой совокупность рабочих столов (desktop). Типичный рабочий стол KDE состоит из трех областей:

- панель запуска, которая по умолчанию располагается внизу экрана и предназначена для запуска приложений и переключения между рабочими столами;
- панель задач, по умолчанию встроенная в панель запуска, используется для переключения между приложениями и управления ими. Щелчок левой кнопкой «мыши» на названии приложения в панели задач позволяет перейти к этому приложению;
- непосредственно сам рабочий стол, на который можно поместить часто используемые файлы и папки.

Запуск приложения производится выбором соответствующего элемента из основного меню, расположенного в крайнем левом нижнем углу.

Каждое приложение в KDE владеет своим собственным окном, переключение между которыми в пределах одного рабочего стола выполняется через комбинацию клавиш «**Alt-Tab**».

Для доступа к файлам можно использовать менеджер файлов, запуск к которому выполняется **Компьютер/Домашняя папка**. Он позволяет осуществлять навигацию по файловой системе, производить выбор документов и выполнение над ними операций.

Для выполнения лабораторных работ каждому пользователю отводится свой домашний каталог. Все файлы, которые пользователь создает при работе с системой, размещаются им только в этом каталоге. Кроме этого, во избежание путаницы в созданных Вами файлах, необходимо в начале выполнения цикла лабораторных работ создать в домашнем каталоге специальный каталог лабораторных работ для размещения проектов.

Запуск интегрированной среды разработки **KDevelop** производится выбором пункта меню **Приложения/Разработка/Интегрированная среда разработки KDevelop 4**.

По завершении работы пользователь обязан выйти из системы. Завершение работы с оболочкой может быть выполнено через основное меню выбором соответствующего элемента из появившегося меню: **Выход/Завершить сеанс**.

## 2.4. Структура среды KDevelop

Интегрированная среда разработки **KDevelop** предназначена для разработки программных систем на многих языках программирования, в том числе и на C. Она предоставляет в распоряжение разработчику следующие основные средства:

- доступ ко всем стандартным инструментам, необходимым для программирования на C, таким как компилятор, компоновщик и т. д.;
- мастер создания проектов для генерации готовых к запуску примеров приложений;

- набор удобных инструментов для редактирования текстов программ, включая форматирование, дополнение кода, быструю навигацию по проекту, подсказки по аргументам, закладки.
- средства управления файлами исходных текстов, заголовочных файлов, документации и т.д., включенных в текущий проект;
- создание руководств пользователя для разрабатываемых программных систем;
- отладка приложений;
- система помощи, включающая описание языка Си и библиотечных функций.

В **KDevelop** интерфейс пользователя определяется следующими компонентами (рис. 1): основное окно, боковые панели вкладок, нижняя панель вкладок, средства управления, включающие меню и командные кнопки.

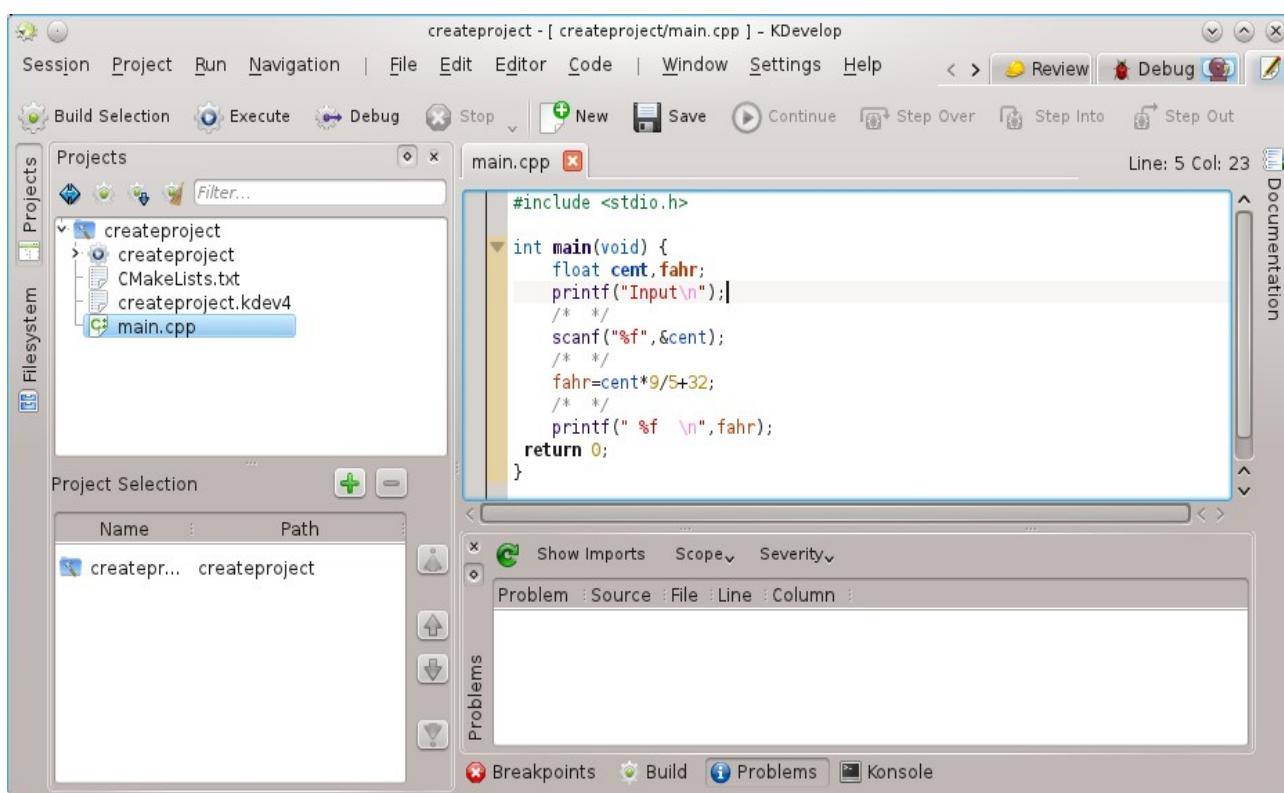





Рис. 1. Внешний вид окна **KDevelop**


**Основное окно** используется для отображения или редактирования элементов проекта: заголовочных файлов, исходных текстов программ и документации, а также для просмотра справочной информации. Для каждого элемента, открытого в основном окне, в верхней части окна создается своя вкладка. Одновременно в окне отображается содержимое только одного элемента проекта. Доступ к другим открытым элементам производится выбором его соответствующей вкладки. Использование этого механизма позволяет удобно и оперативно переключаться между различными элементами проекта и справочной информацией.

**Боковые панели вкладок** используются для навигации между различными элементами проекта и имеет следующие основные вкладки:


 *Filesystem (Файловая система)* – отображает все файлы внутри каталога проекта в том виде, как они размещаются в файловой системе.


 *Projects (Проекты)* — отображает файлы проекта с представлением структуры каталогов проекта и позволяет выполнять стандартные операции над ними через контекстное меню;


 *Variables (Переменные)* – отображает список и текущие значения всех переменных текущей выполняемой функции и пользовательских выражений. Предоставляет доступ к ряду инструментов отладки программ. Данная вкладка «работает» только при выполнении программы в режиме отладки и рассматривается подробнее ниже в соответствующем разделе;


 *Documentation (Документация)* – отображает список доступной документации, каждый элемент которого представляет HTML-книгу или ссылку на внешний источник информации. Позволяет осуществлять навигацию, поиск и установку закладок;

**Нижняя панель вкладок** используется для доступа к различным информационным окнам и окнам дополнительных средств и имеет следующие основные вкладки:

 *Build (Сборка)* – используется для отображения информации, выдаваемой компилятором. Выбор в этом окне строки, содержащей сообщение об ошибке, приводит к отображению соответствующей строки программы в основном окне. Через контекстное меню можно управлять детализацией отображения вывода;

 *Problems (Проблемы)* – отображает список «проблемных» мест в исходном коде программы. В указанных местах предположительно находятся ошибки в записи программы, или же они помечены для доработки или исправления.

 *Konsole* – предоставляет в распоряжение разработчику окно с командной строкой, в котором можно вводить, редактировать и выполнять команды как при работе с пользовательской оболочкой *bash*;

 *Breakpoints (Точки останова)* – позволяет управлять точками останова при отладке программы. Доступ к операциям осуществляется через командные кнопки, расположенные в левой части окна.

## 2.5. Описание меню

В таблице 1 приведены группы команд и их назначение, доступные из основного меню **Kdevelop**.

Многие из приведенных групп содержат большое число различных команд. Наиболее важные из них рассматриваются ниже. Описание остальных команд, не рассмотренных в данных методических указаниях, можно получить с помощью интерактивного средства самой среды **KDevelop** «*Что это такое?*». Оно доступно через меню **Справка/Что это? Shift+F1**. После выбора этого средства необходимо указателем «мыши» выбрать желаемый пункт меню, по-

сле чего высветится всплывающее окно с описанием пункта.

Таблица 1

<i>Название группы</i>	<i>Назначение группы команд</i>
<b>Session (Сеанс)</b>	Команды для запуска нового приложения, закрытия приложения
<b>Project (Проект)</b>	Команды работы с проектами
<b>Run (Запуск)</b>	Команды для отладки и запуска программ
<b>Navigation (Навигация)</b>	Команды для просмотра стандартных функций, файлов
<b>File (Файл)</b>	Команды работы с файлами
<b>Edit (Правка)</b>	Команды редактирования
<b>Editor (Редактор)</b>	Команды управления отображением, добавление/удаление закладок, редактирование
<b>Code (Код)</b>	Команды работы с кодом
<b>Window (Окно)</b>	Команды работы с окнами
<b>Settings (Настройка)</b>	Команды установки параметров работы с системой
<b>Help (Справка)</b>	Команды вызова системы помощи

#### Меню Сеанс

**Запустить новый сеанс ...** Запуск нового приложения.

**Переименовать сеанс ...** Переименовать созданное приложение.

**Выход Ctrl+Q** Закрыть приложение.

#### Меню Проект

**Создать из шаблона ...** Вызов мастера создания нового приложения.

**Открыть проект ...** Открытие существующего проекта.

**Закрыть проект** Закрытие текущего проекта.

#### Меню Запуск

**Настроить конфигурации запуска ...** Выбор конфигурации проекта.

**Запустить программу Shift+F9** Запуск главной программы проекта.

**Начать отладку F9** Запуск главной программы на выполнение под отладчиком.

**Завершить задачи** Остановка выполнения программы и выход из отладки.

**Выполнить строку F10** Выполнить одну строку программы без захода внутрь функции.

**Step Into F11** Выполнить одну строку программы с заходом внутрь пользовательской функции при ее вызове.

**Step Out F12** Выполнить до конца текущую функцию.

**Выполнить до курсора** Продолжить выполнение программы до достижения

строки программы, содержащей курсор.

**Поставить точку останова** Установить/снять точку останова в текущей строке редактора.

### Меню **Правка**

**Отменить действие Ctrl+Z** Отмена последней выполненной операции.

**Повторить Ctrl+Shift+Z** Выполнение отмененной операции.

**Вырезать Ctrl+X** Вырезать выделенный текст и поместить его в буфер обмена.

**Копировать Ctrl+C** Копирование выделенного текста в буфер обмена.

**Вставить Ctrl+V** Вставка текста из буфера обмена.

**Найти Ctrl+F** Поиск строки в текущем файле.

### Меню **Редактор**

#### Команды меню **Редактор**

#### **Вид**

**Номера строк** Отображение/скрытие номеров строк файла по левой стороне окна.

#### **Сервис**

**Кодировка** Выбор кодировки для отображения текста в окне редактора.

**Вставить отступ Ctrl+I** Сдвиг строк выделенного фрагмента текста вправо.

**Уменьшить отступ Ctrl+Shift+I** Сдвиг строк выделенного фрагмента текста влево.

**Закомментировать Ctrl+D** Оформить текущую строку или выделенный фрагмент текста в виде комментария.

**Раскомментировать Ctrl+Shift+D** Убрать символ комментариев для текущей строки или выделенного фрагмента текста.

### Меню **Настройка**

**Панели инструментов...** Настройка панелей инструментов.

**Configure Editor...** Настройка режимов работы в редакторе.

**Настроить KDevelop...** Настройка параметров и режимов работы интегрированной среды.

### Меню **Справка**

**Что это? Shift+F1** Интерактивное средство «Что это такое?».

## 2.6. Создание нового проекта

Разработка программы начинается с четкого уяснения требований к программе посредством анализа постановки задачи. Только по завершении этого этапа программа непосредственно записывается на языке программирования с привлечением среды разработки **Kdevelop**.

Разработка программной системы в среде начинается с **создания нового**



проекта. Создание проекта выполняется при помощи «Мастера приложений», доступного через меню **Project (Проект)/New from Template... (Создать из шаблона...)**. Это средство представлено диалоговым окном (рис. 2), в котором осуществляется задание параметров (настройка) проекта. Весь процесс создания нового проекта разбит на несколько шагов, каждый из которых выполняется на своей странице мастера. Переход между страницами выполняется с помощью кнопок *Back (Назад)* (к предыдущей странице) и *Next (Далее)* (к следующей), расположенными внизу окна мастера.

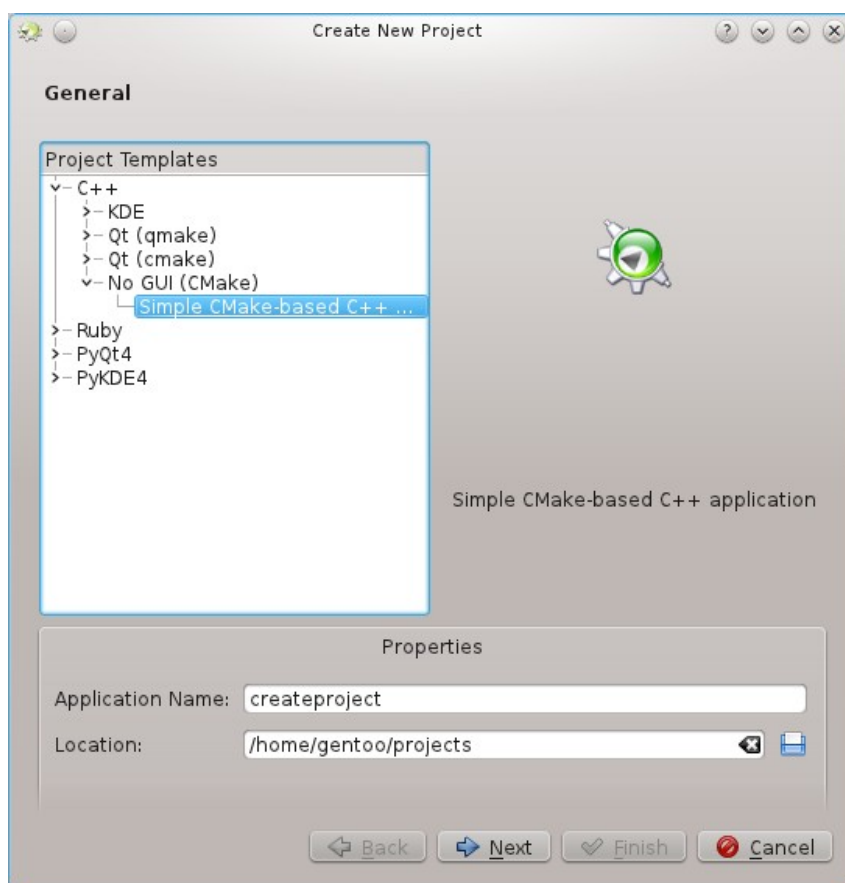


Рис. 2. Диалоговое окно мастера нового проекта

Прежде всего на первой странице *General (Основное)* из списка вкладки *Project Templates (Шаблоны проекта)* выбирается тип разрабатываемого приложения. Для целей лабораторных работ в качестве типа приложения следует выбирать консольное приложение C++ (элемент списка *C++/No GUI (CMake)/Simple Cmake-based C++...(Простое приложение на C...)*). В группе *Properties (Свойства)* в поле *Application Name: (Название приложения:)* указывается название проекта (латинскими буквами), а в поле *Location (Адрес)* указывается каталог, внутри которого будет создан каталог для хранения всех файлов проекта. **Категорически запрещено в названии проекта и в расположении использовать русские символы, пробелы и специальные символы \, /, \$.** При выполнении лабораторных работ в качестве такого каталога следует указывать созданный Вами ранее специальный каталог проектов. Для выбор ката-

лога **настоятельно рекомендуется** использовать диалоговое окно, вызываемое при нажатии на кнопку, расположенную справа от поля задания пути каталога.

Для выполнения лабораторных работ все остальные поля на этой и других страницах можно оставить без изменения. После задания всех необходимых параметров нажатием на кнопку *Finish (Готово)*, расположенную внизу на последней странице Мастера, выполняется генерация проекта со всеми необходимыми файлами. По завершении генерации появляется страница *Configure a build directory for ... - Kdevelop (Настройка каталога сборки для...)*, на которой необходимо нажать кнопку **ОК**.

Результатом создания проекта является шаблон программной системы, который будет готов к сборке и запуску (раздел 2.8) после настройки параметров проекта (раздел 2.7).

## 2.7. Настройка параметров проекта

Для созданного проекта следует **выполнить настройку его параметров**. Настройка осуществляется через меню **Run (Запуск)/Configure Launches... (Настроить конфигурации запуска...)**, выбор которого приводит к вызову окна (рис. 3). После подсветки названия проекта (createproject) в левой половине окна и нажатия кнопки **+** вверху окна, в правой половине окна появится возможность установки параметров. Для успешного выполнения лабораторных работ рекомендуется выполнить следующую настройку параметров (рис. 3):

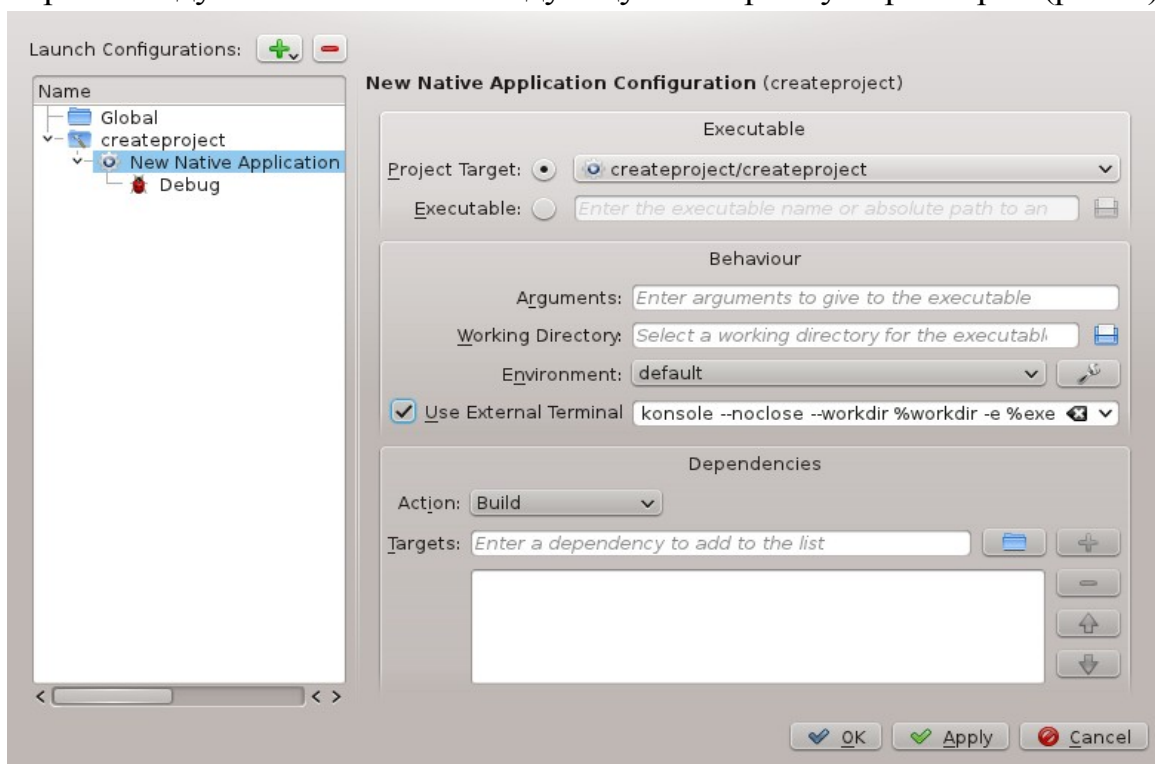



Рис. 3. Диалог настройки параметров компилятора

- выбрать *Project Target (Цель проекта)*;

- включить параметр *Use External Terminal (Использовать внешний терминал)*;
- выбрать *Action (Действие): Build (Сборка)*.

По завершении настройки параметров следует нажать на кнопку **ОК**.


## 2.8. Сборка и запуск проекта

По завершении ввода и редактирования текста программы выполняется его компиляция и сборка. Доступ к этой операции осуществляется либо через меню интегрированной среды **Project (Проект)/Build Selection (Сборка)**, либо нажатием кнопки . Во время компиляции в окне *Build (Сборка)* выводятся диагностические сообщения о процессе компиляции и сборки. По завершении компиляции и сборки в последней строке будет сообщено об успешности этой операции.

Обнаруженные в процессе компиляции синтаксические ошибки в тексте программы перечисляются в окне сообщений. При этом строки с описанием ошибки, для отделения от прочей информации, выделяются специальным цветом. Ошибка или предупреждение указывается в следующем виде:

*имя\_файла : номер\_строки : описание\_ошибки*

При наличии в тексте программы синтаксических ошибок исполняемая программа не будет получена до их полного устранения. Процесс устранения ошибок состоит в следующем. Во вкладке *Build (Сборка)* щелчком левой кнопки мыши на строке с описанием ошибки активизируется окно редактирования. При этом в нем открывается файл, содержащий ошибку, и курсор устанавливается на соответствующую строку программы. Средствами редактирования в тексте программы необходимо устранить ошибку, после чего заново провести компиляцию программы. Хотя во время компиляции может быть выдано сразу несколько сообщений об ошибках, настоятельно рекомендуется исправлять только одну ошибку и именно первую, а не пытаться разобраться во всех сразу. Это связано с тем, что последующие ошибки, выданные компилятором, могут быть следствием первой, и, очевидно, пропадут после ее устранения. В конце методического указания приведен список наиболее часто встречаемых ошибок с рекомендациями по их устранению.

После успешно проведенной сборки проекта и настройки его параметров получена исполняемая программа, которая может быть выполнена через меню **Run (Запуск)/Execute Launch (Запустить программу)** или с помощью кнопки , расположенной на панели инструментов. При этом диалог программы с пользователем осуществляется в отдельном консольном окне приложения. После завершения диалога необходимо закрыть окно приложения пользователя.

Полученная программа, скорее всего, содержит семантические ошибки, которые не могли быть выявлены компилятором, например, ошибки в алгоритме. Поэтому следующим этапом является поиск и устранение таких ошибок,


используя средства отладки интегрированной среды **KDevelop**.


## 2.9. Средства отладки. Методика отладки программ

Во время отладки разработчиком решаются следующие задачи:

- пошаговая проверка хода выполнения отдельных компонент программы. При этом исследуется последовательность выполнения операторов, какие операторы выполняются и сколько раз. Реальное поведение сопоставляется с прогнозируемым поведением на тестовых данных;
- трассировка значений используемых переменных. В этом случае текущие значения переменных сопоставляются с теми, которые должны быть при правильном выполнении программы. Обнаруженное несоответствие свидетельствует об ошибке в программе;



Для проведения отладки используется заранее подготовленный набор *тестов*, то есть конкретных значений входных данных программы, для которых поведение программы и результат ее работы известны. Работа программы должна проверяться на нескольких различных тестах с тем, чтобы проверить все возможные пути ее прохождения.


Среда программирования **KDevelop** предоставляет в распоряжение разработчику удобные средства отладки программ. Отладка программ выполняется на уровне исходных текстов. Это означает, что в процессе отладки Вы работаете с исходным текстом программы. Одним из центральных понятий при отладке является *точка выполнения* программы. Это та строка программы, которая должна быть выполнена в данный момент. При проведении отладки она отмечается в окне редактирования в серой вертикальной полосе в левой части окна значком . Ниже рассматриваются наиболее полезные средства отладки:

Установка *точки останова* на нужной строке программы. В левой вертикальной полосе окна редактирования она помечается специальным значком . Когда во время выполнения программы эта строка станет точкой выполнения, исполнение программы будет приостановлено с передачей управления среде программирования. Установка и снятие точки останова производится щелчком левой кнопкой «мыши» на серой вертикальной полосе окна редактирования. Одновременно может быть установлено несколько точек останова. Для отображения списка точек останова необходимо выбрать вкладку *Точки останова* на нижней панели вкладок. Щелчком левой кнопкой «мыши» на элементе списка выполняется отображение строки программы с точкой останова, а щелчком правой кнопкой вызывается контекстное меню для выполнения операций с точками останова.

Для запуска программы *на выполнение под отладчиком* необходимо выбрать пункт меню **Run (Запуск)/Debug Launch (Запуск отладчика)**. **Обязательно для успешного запуска отладчика необходимо сначала осуществлять сборку проекта.**

При *пошаговом выполнении* программы исполняются операторы строки,

которая соответствует текущей точке выполнения. Различают два способа выполнения шага программы – с заходом внутрь пользовательских функций и без захода. Они отличаются только в том случае, если в точке выполнения программы осуществляется вызов некоторой функции, определенной в программе. В первом случае, меню **Run (Запуск) / Step Into... (Пройти к следующей строке ...)** или кнопка , будет выполнен заход в функцию с возможностью ее пошагового выполнения. Во втором случае, меню **Run (Запуск) / Выполнить строку** или кнопка , выполнение функции производится за один шаг.

*Принудительное завершение* отладки программы. В ряде случаев возникает необходимость завершить отладку, не дожидаясь завершения программы. Причинами для этого решения могут послужить найденная во время отладки ошибка, вход программы в бесконечный цикл и т.д. Это действие выполняется через меню **Run (Запуск) / Stop Jobs (Завершить задачи)** или кнопку  .

*Трассировка значений переменных.* В процессе отладки в боковой панели вкладок становится доступным вкладка *Переменные*, в которой отображаются значения всех локальных переменных, доступных в текущей точке выполнения программы. При этом, если переменная имеет сложную структуру, то нажатие кнопки «мыши» на знак > слева от имени приводит к отображению значений полей этой переменной. Для отображения значения глобальной переменной ее имя необходимо ввести в поле, расположенное внизу раздела, после чего нажать кнопку *Enter*. Если во время отладки значения переменных изменятся, то эти изменения сразу же будут отражены в панели просмотра. Таким образом, пользователь может отслеживать динамику изменения значений интересующих его переменных по ходу выполнения программы. Вызов по отображаемой переменной контекстного меню позволяет получить доступ к дополнительным операциям над ней.

В ходе выполнения лабораторных работ предлагается использовать следующую последовательность действий для отладки программы:

- установите точку останова на требуемой строке программы. При первоначальной отладке в большинстве случаев это должен быть заголовок функции *main()*;
- запустите программу на выполнение под отладчиком;
- по достижении точки останова выполняйте пошаговое исполнение программы с анализом последовательности выполнения операторов и наблюдением за значениями переменных после каждого шага на вкладке *Переменные*;
- при обнаружении несоответствия реального поведения программы с ожидаемым поведением, проанализируйте сложившуюся ситуацию с целью выявления причин несоответствия;
- после выявления причины ошибочного выполнения программы завершите ее отладку и внесите изменения в текст программы;
- выполните сборку проекта;
- повторите процесс отладки измененной программы.

### 3. Задания и порядок их выполнения

1. Выполните загрузку операционной системы GNU/Linux и зарегистрируйтесь в системе, как это описано в разделах 2.1 и 2.2.
2. В соответствии с разделом 2.3 изучите основные приемы работы с менеджером файлов **Konqueror**. В домашнем каталоге создайте каталог проектов (например, .../11P1/Petrov, указав номер группы и свою фамилию).
3. Запустите среду **Kdevelop** (раздел 2.3) и ознакомьтесь с её структурой, используя информацию из разделов 2.4 и 2.5 данных методических указаний.
4. Для задачи, указанной преподавателем, создайте новый проект, как это описано в разделе 2.6. Сразу после создания первого проекта выполните настройку параметров проекта в соответствии с рекомендациями раздела 2.7.
5. В соответствии с разделом 2.8 выполните сборку и запуск проекта.
6. Используя средства отладки (инструменты, рассмотренные в разделе 2.9) и заранее подготовленный набор тестов, проверьте правильность работы созданной программы и покажите ее работу преподавателю.
7. Доработайте программу с учётом изменений, предложенных преподавателем и продемонстрируйте её работу преподавателю.

### 4. Контрольные вопросы

1. Перечислите действия, необходимые для запуска **KDevelop**.
2. Укажите основные этапы разработки программ в среде.
3. Опишите структуру программной системы **KDevelop**.
4. Как осуществлять перемещение по компонентам проекта?
5. Опишите назначение ключевых вкладок боковых и нижней панелей.
6. Как выполнить сборку проекта и запустить программу на выполнение?
7. Перечислите основные средства отладки программ в интегрированной среде.

### 5. Список ошибок времени компиляции

*assignment of read-only variable 'имя'*. Попытка изменить значение постоянной переменной.

*assignment to 'mun1' from 'mun2' lacks a cast*. Попытка присвоить значение *mun2* переменной, имеющей *mun1*.

*base operand of '->' has non-pointer type 'mun'*. Слева от оператора стрелка может стоять только указатель на структуру.

*'имя' cannot be used as a function*. Указанное имя не является функцией. Обычно это является следствием применения операции вызова функции для переменной.

*conflicting types for 'переменная'*. Указанная переменная объявлена более

одного раза с разными типами.

***control reaches end of non-void function 'функция'***. Текущая функция должна возвращать значение некоторого типа, отличного от ***void***, но компилятор обнаружил оператор ***return*** без возвращаемого значения, либо вообще его не нашел.

***declaration does not declare anything***. В объявлении пропущено имя объявляемой переменной.

***excess elements in aggregate initializer***. Компилятор встретил больше инициализаторов, чем нужно для данного элемента.

***implicit declaration of function 'function'***. Либо отсутствует прототип указанной функции, либо неверно указано имя функции при вызове или в прототипе.

***initialization to 'mun1' from 'mun2' lacks a cast***. Попытка инициализировать переменную *mun1* значением *mun2*.

***no newline at end of file***. В указанном файле не обнаружена в конце пустая строка. Компилятор требует ее наличия.

***non-lvalue in assignment***. Слева от оператора присваивания должно быть адресуемое выражение.

***'имя': No such file or directory***. Нет такого файла или каталога. Ошибка обычно вызвана попыткой подключения несуществующего заголовочного файла или неверным указанием имени файла.

***parse error at end of input***. Компилятор обнаружил конец файла с текстом программы, но для текущей функции не нашел закрывающей фигурной скобки. Наиболее часто ошибка вызвана несогласованным количеством скобок «{», «}».

***parse error before 'элемент'***. Синтаксическая ошибка перед указанным элементом. Как правило вызвана либо пропуском символа, либо лишним символом в строке. Иногда имеет смысл проверить предыдущую строку программы.

***parse error before character 'символ'***. В тексте программы встретился недопустимый в этом месте символ.

***passing 'mun' to argument n of 'функция' lacks a cast***. При вызове функции ей передается параметр с номером *n* недопустимого типа *mun*.

***redefinition of 'функция'***. Указанная функция уже была определена ранее.

***return type for 'main' changed to 'int'*** Тип значения, возвращаемого функцией *main*, должен быть целым.

***statement with no effect***. Результат действия в строке программы никак не используется (является бессмысленным).

***'struct имя' has no member named 'поле'***. Попытка использования *поле* в качестве члена структуры типа *имя*, но такого поля в данном типе нет. Проверьте определение типа.

***suggest parentheses around assignment used as truth value***. Ошибка вызвана неправильной записью операции сравнения на равенство при задании усло-

вия.

*too few arguments to function 'function'*. При вызове функции указано недостаточное количество параметров. Сопоставьте способ вызова функции с ее прототипом.

*too many arguments to function 'функция'*. При вызове функции с указанным именем ей передаются лишние параметры. Сопоставьте способ вызова функции с ее прототипом.

*'имя' undeclared (first use this function)*. Указанное имя используется, но нигде не объявлено.

*undefined reference to 'имя'*. Неопределённая ссылка на указанное имя. Ошибка возникает на этапе компоновки и обычно вызвана тем, что некоторая компонента с указанным именем объявлена, используется, но нигде в программе не определена.

*unterminated string or character constant*. Компилятор не обнаружил завершающих кавычек для строки или апострофа для символа.

*unused variable 'имя'*. Переменная с указанным именем объявлена, но нигде не используется.

*'имя' was not declared in this scope*. Указанное имя в текущей области видимости никак не определено. Обычно следует проверить правильность его написания или подключить нужный заголовочный файл с объявлением типа.

## 6. Список литературы

1. Шилдт, Г. Справочник программиста по C / C++: [пер. с англ.] / Г.Шилдт – 2-е изд.– М.: Вильямс, 2001.
2. Керниган, Б. Язык программирования Си: [пер. с англ.] / Б.Керниган, Д.Ритчи – 2-е изд.- М.: Вильямс, 2006.
3. Павловская, Т.А. C/C++. Программирование на языке высокого уровня: учебник для вузов / Т.А.Павловская.- СПб.: Питер, 2005.
4. Костюкова, Н. Язык Си и особенности работы с ним / Н.Костюкова, Н.Калинина — Интернет-университет информационных технологий — ИНТУИТ.ру, 2006.