Московский государственный технический университет

имени Н.Э. Баумана

Факультет Информатика и системы управления

Кафедра Компьютерные системы и сети

Г.С. Иванова, Т.Н. Ничушкина

**Консольные приложения на языке C++**

**в среде Qt Creater 4.12 Community**

Учебно-методическое пособие

Москва 2021

**Иванова Г.С., Ничушкина Т.Н.**

Консольные приложения на языке C++ в среде Qt Creater 4.12 Community, 2021. 32 с.

Издание содержит методический материал для выполнения лабораторной работы по дисциплинам «Информатики» и «Объектно-ориентированное программирование». Работа посвящена первичному знакомству с со средой программирования программированием Qt Creater 4.12 Community и формированию умений и навыков создания программ на C и C++.

В пособии определены цели лабораторной работы, пояснены сложные моменты, приведены задания, предложен порядок их выполнения и сформулированы требования к отчету..

Для студентов МГТУ имени Н.Э. Баумана, обучающихся по программам бакалавриата направлений «Математика и компьютерные науки», «Информатика и вычислительная техника» и «Прикладная информатика».

*Рекомендовано учебно-методической комиссией факультета «Информатика и системы управления» МГТУ им. Н.Э. Баумана*

*Учебное издание*

**Иванова Галина Сергеевна**

**Ничушкина Татьяна Николаевна**

**Консольные приложения на языке C++ в среде Qt Creater 4.12 Community**

**.**

© 2021 МГТУ имени Н.Э. Баумана

**Оглавление**

[Предисловие 4](#_Toc80781804)

[Введение 5](#_Toc80781805)

[1 Описание используемого программного обеспечения 5](#_Toc80781806)

[1.1 Создание заготовки консольного приложения 5](#_Toc80781807)

[1.2 Ввод программы 8](#_Toc80781808)

[1.3 Запуск программы на выполнение 8](#_Toc80781809)

[1.4 Модульное программирование. Файлы заголовков 8](#_Toc80781810)

[1.5 Отладка консольных приложений 8](#_Toc80781811)

[1.6 Просмотр значений переменных в режиме отладки 8](#_Toc80781812)

[1.7 Установка и отмена точек останова в программе 8](#_Toc80781813)

[2 Порядок выполнения работы 8](#_Toc80781814)

[3 Требования к отчету 8](#_Toc80781815)

[4 Требования к защите 8](#_Toc80781816)

[5 Контрольные вопросы 8](#_Toc80781817)

[Список литературы 8](#_Toc80781818)

[Приложение А. Пример титульного листа. 8](#_Toc80781819)

# Предисловие

Учебно-методическое пособие составлено в соответствии с самостоятельно устанавливаемыми образовательными стандартами (СУОС), основными образовательными программами по направлениям подготовки бакалавров 09.03.01 «Информатика и вычислительная техника», 09.03.03 «Прикладная информатика» и «Математика и компьютерные науки», а также, программой дисциплин «Информатика» и «Объектно-ориентированное программирование». Издание предназначено для изучения среды программирование для ее использования при выполнении других лабораторных работ и домашних заданий всех модулей указанных дисциплин.

Цель учебно-методического пособия – закрепление теоретических знаний и формирование практических навыков, необходимых для разработки программ на языках программирования C и C++, формирование умений создания, тестирования и отладки созданных программ.

В результате изучения материала, представленного в пособии, студент будет:

- знать особенности создания приложений в среде Qt Creater 4.12 Community,

- уметь настроить приложение для его корректного применения,

- знать особенности формирования многофаловых проектов,

- знать средства отладки среды Qt Creater 4.12 Community,

- уметь проводить отладку приложения с применением имеющихся отладочных средств.

Для успешного освоения материала необходимо знания школьного курса «Информатика».

Задача издания – предоставить студентам материал, который будет способствовать более качественному усвоению материала всех модулей дисциплин «Информатики» и «Объектно-ориентированное программирование». Для успешного усвоения материала необходимо внимательно прочитать издание и выполнить все предлагаемые задания. Ответы на вопросы позволят обучающемуся оценить степень понимания и усвоения теоретических положений.

# Введение

Интегрированная среда программирования ***Qt\_Creater 4-12 Community*** предназначена для создания программ для 32х или 64х разрядных процессоров, в том числе и консольных приложений, написанных на языках *С* и *С++*. Может функционировать на различных операционных системах: *Linux*, *MAC Os*, *Windows*. Является свободно распространяемым программным продуктом.

Лабораторная работа состоит из трех логических частей: создание однофайлового приложения в *Qt\_Creater 4-12 Community*,создание многофайлового приложения и модульное программирование, изучение средств отладки среды *Qt\_Creater 4-12 Community*. Объем работы 2 часа.

# Описание используемого программного обеспечения

## Создание заготовки консольного приложения

Главное окно среды при запуске имеет вид, представленный на рисунке 1. Окно включает основное меню (сверху), боковую панель инструментов (слева на черном фоне), навигатор со стартовой страницей (среднее окно) и список уже созданных проектов в основном окне (справа).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Вид главного окна среды при отсутствии открытого проекта

Создание заготовки консольного приложения выполняется либо нажатием кнопки **Создать проект...** стартовой страницы (справа вверху), либо с использованием пункта меню **Файл/Создать/Проект...**.

В появившемся окне **Создание проекта** выбираем тип проекта **Проект без Qt** ишаблон **Приложение на языке С** (рисунок 2).

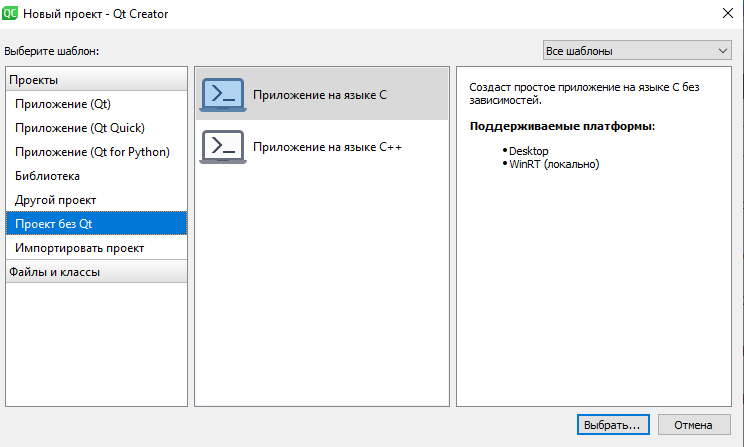


Рисунок 2 – Вид окна выбора типа проекта и задания его имени и местоположения

После нажатия кнопки **Выбрать**, появляется окно выбора имени и места размещения проекта (рисунок 3). Вводим имя проекта *Prnod* в поле **Название.** В поле **Создать в** можно указать отличное от стандартного местонахождение папки будущего проекта в позиции.

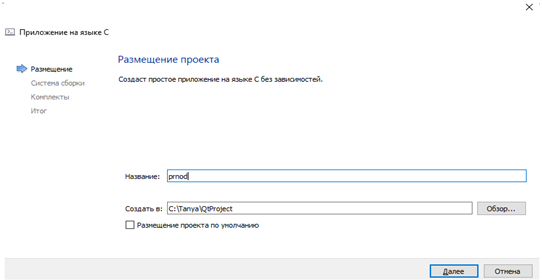


Рисунок 3 – Вид окна выбора имени проекта и задания его местоположения

*Важно!* Среда *Qt\_Creator* критична к именам проектов и папок. Имена не должны содержать пробелов и символов кириллицы!!!

Аналогично можно выбрать **Приложение на языке *С*++.** Тогда проект будет содержать шаблон программы на C++ с подключением соответствующих библиотек.

После нажатия кнопки **Далее** необходимо выбрать комплект, который должен быть использован при создании программы. Комплект включает компилятор и другие вспомогательные средства. В качестве примера на скриншоте выбран комплект ***Desktop Qr 5.12.2 MinGW32-bit*** (рисунок 4). Это комплект с компилятором ***MinGW***, используемый для создания 32-х разрядных консольных приложений.

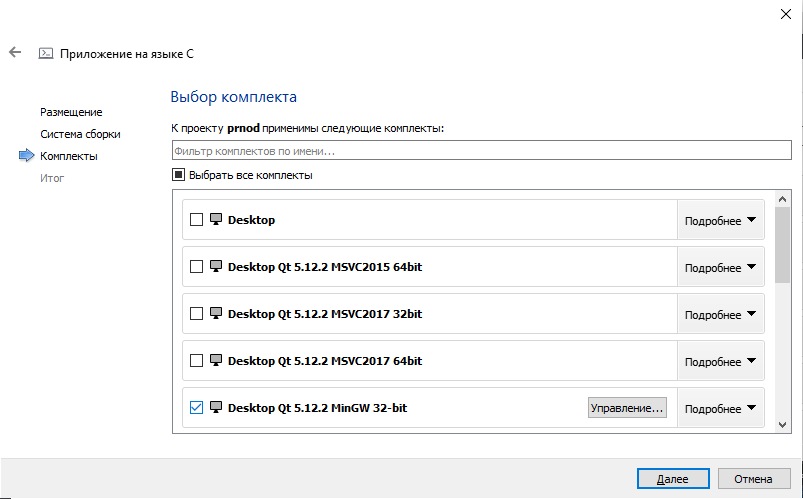


Рисунок 4 – Вид окна выбора комплектов

После нажатия кнопки **Далее** среда предлагает выбрать систему сборки (рисунок 5). Оставляем сборку по умолчанию *qmake* и нажимаем кнопку **Далее**.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Вид окна выбора системы сборки

В результате будет создан проект с шаблоном программы, которая выводится в окне редактора среды (справа). В окне навигатора (посередине) высвечивается список исходных файлов проекта, который раскрывается и свертывается при нажатии стрелок «>» и «v» соответственно. Двойной щелчок мыши на имени файла приводит к визуализации файла в окне редактора.

Полученную программу можно запустить на выполнение. Для этого достаточно нажать кнопку запуска (зеленый треугольник слева внизу экрана на рисунке 6).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Вид окна с готовым шаблоном программы на С

В зависимости от настроек среды результат работы программы – сообщение «Hello, World!» – будет выведен либо в окне **3 Вывод приложения** (см. рисунок 6), либо в раскрывшемся окне консоли.

Если результаты появились в окне 3 Вывод приложения, то следует изменить настройку среды. Для этого необходимо выбрать пункт **Проекты** панели инструментов, расположенной слева на черном фоне, в появившемся окне в подменю, раскрывшемся в поле навигатора, выбрать подпункт **Запуск**. В появившемся окне **Настройка запуска** следует найти поле **Запуск** и установить флаг **Запускать в терминале** (рисунок 7). После этого можно вернуться в редактор, используя левую панель инструментов. После этого ввод и вывод информации для программы будет выполняться через консоль.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 7** – Настойка запуска проекта

Консоль с результатами работы программы представлена на рисунке 8.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 8**– Вывод результатов работы программы шаблона на консоль

Теперь программу можно корректировать для выполнения задания лабораторной работы.

## Ввод программы

В качестве примера введем в окно редактора программу вычисления наибольшего общего делителя двух целых чисел. Причем само вычисление выделим в функцию *nod*, оставив в основной программе ввод данных и вывод результата.

**#include <stdio.h>** // подключение процедур ввода вывода

**int nod(int x, int y)**

**{**

**while (x!=y)**

**if (x>y) x=x-y;**

**else y=y-x;**

**return y;**

**}**

**int main()**

**{**

**int a,b;**

**puts("Enter two integer value:");**

**scanf\_s("%d %d",&a,&b);**

**printf("Nod %d и %d = %d.\n",a,b,nod(a,b));**

**return 0;**

**}**

Программа настроена на использование при выводе английского языка.

Если вы обратили внимание на дерево проекта, то должны были увидеть еще один файл – *prnod.pro*. Это и есть один из главных файлов, создаваемых средой – файл проекта, который обеспечивает компиляцию, сборку и запуск на выполнение программы в среде *Qt Creator*. Именно в нем указаны все главные настройки.

Ниже приведен файла prnod.pro, созданный средой.

**TEMPLATE = app**

**CONFIG += console**

**CONFIG -= app\_bundle**

**CONFIG -= qt**

**SOURCES += main.c**

Знак «+=» означает подключение соответствующего ресурса среды, знак «-=» –отключение указанного ресурса.

При необходимости файл проекта тоже можно корректировать, добавляя или удаляя ресурсы.

## Запуск программы на выполнение

Для запуска программы обычно используют инструмент **Запуск**, который находится в панели инструментов слева и имеет вид зеленого треугольника (рисунок 9).

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

**Рисунок 9** –Запуск на программы выполнение

Если компиляция и компоновка программы прошли без ошибок, то на экране появляется окно консоли, в которое выводится запрос на ввод данных и в котором отображаются вводимые данные и строка результата (рисунок 10). Окно закрывается по нажатию любой клавиши или кнопки закрытия окна «**×**» в правом верхнем углу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10– Окно консоли с выводом результата (изменены цвета фона и символов)

При обнаружении ошибок среда высвечивает сообщения в соответствующих вкладках, расположенных в нижней части окна среды. Для просмотра сообщений следует выбрать необходимые вкладки, например **1 Проблемы** или **4 Вывод сборки**.

Если это ошибка в синтаксисе программы, то в окне редактора соответствующие строки помечены редактором.

Внесем ошибку в программу, изменив имя подпрограммы *puts*. Пример окон с диагностикой ошибок приведен на рисунках 11 и12.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 11** –Вывод сообщений в закладке **1-** **Проблемы**

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 12** –Вывод сообщений этапа сборки программы (закладка **4 - Вывод сборки)**

После исправления ошибок программу необходимо вновь скомпилировать и скомпоновать.

Если ошибка обнаружена уже при выполнении программы, а именно:

* программа выдает сообщение об ошибке выполнения,
* результаты программы отличны от ожидаемых,
* программа «зациклилась»,

то для поиска ошибки следует использовать средства отладки среды *Qt Creater* (см. разделы 1.5 и 1.6).

*Примечание*. При «зацикливании» программы ее выполнение можно завершить, используя кнопку закрытия окна «**×**» в правом верхнем углу окна результата.

## Модульное программирование. Файлы заголовков

Среда *Qt Creater*позволяет создавать и отлаживать программы, использующие не только стандартные, но и пользовательские библиотеки (модули).

Модуль языка *C++* в отличие от модуля языка *Free Pascal* по рекомендации включает два файла: заголовочный файл с расширением **.h** и файл реализации с расширением **.cpp**.

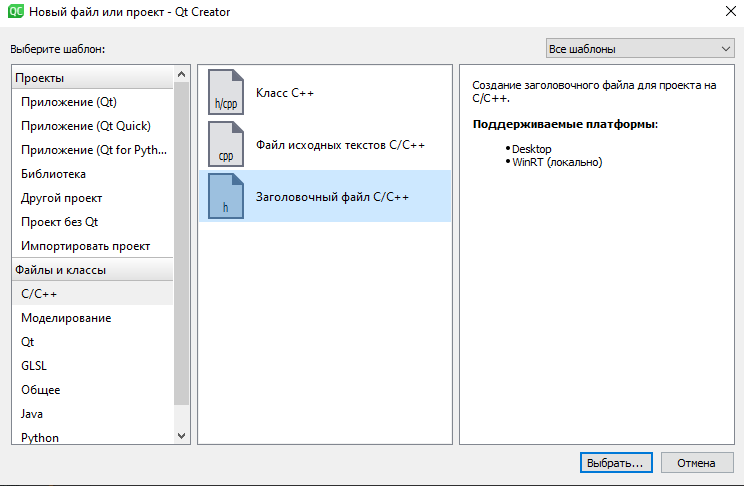
*Важно!!!.* Среда как при выборе проекта на языке *С***,** так и при выборе проекта на языке *С++*создает файл реализации с расширением ***.cpp****.* Поэтому, при выборе проекта на языке *C*, расширение файла **обязательно** (!) нужно поменять на ***.с***, иначе при сборке программы будут диагностированы ошибки.

Заголовочный файл играет роль своеобразной интерфейсной секции модуля. В него помещают объявление экспортируемых ресурсов модуля: прототипы (заголовки) процедур и функций, объявления переменных, типов и констант. Этот файл подключают директивой **#include** "**<Имя модуля>.h**"к файлам реализации модуля, программы или другого модуля, если они используют ресурсы описываемого модуля.

Файл реализации представляет собой аналог секции реализации модуля языка *Pascal*. Он должен содержать директивы подключения заголовочных файлов используемых модулей, описания экспортируемых процедур и функций, а также объявления и описания внутренних ресурсов модуля.

Для создания файлов заголовка и реализации модуля и *добавления их к проекту* используют пункт меню **Файл\ Создать файл**.

В появившемся окне в меню слева необходимо выбрать раздел **Файлы и классы,** а в появившемся левее меню выбрать **Заголовочный файл С/С++** (рисунок 13)



**Рисунок 13** – Выбор типа создаваемого файла модуля (заголовочного)

После нажатия кнопки **Выбрать** появится форма выбора имени создаваемого файла, в которую следует ввести его имя (рисунок 14).

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 14** – Ввод имени создаваемого файла модуля (заголовочного).

После ввода имени и нажатия кнопки **Далее**, появляется окно завершения процеса создания файла и его атрибуты (рисунок 15).

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 15** – Вид окна подтверждения создания файла модуля (заголовочного).

После нажатия кнопки **Завершить** справа, в окне редактора появится заготовка текста файла. Для конкретного задания этот шаблон корректируется. В заголовочном файле *Nod.h* добавим описание заголовка функции (рисунок 16):

**int nod(int x,int y)**

Добавленное определение функции в окне редактора выделяется слева красной линией. Обратите внимание на инструкции, выделенные синим цветом. Это команды препроцессора, которые добавляются автоматически средой, и предназначены для исключения повторного подключения этого модуля к проекту.

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

**Рисунок 16** – Вид текста заголовочного файла после создания и корректировки.

По описанной схеме создадим второй файл – файл реализации модуля *Nod.c* или файл *Nod.cpp* в зависимости от выбранной заготовки проекта.

С учетом того, что по умолчанию среда создает файл реализации **всегда** с расширением ***.cpp***, при работе с проектом на языке *С* следует менять расширение имени исходного файла на ***.c***. Для изменения назначенного средой расширения в навигаторе **Проекты**, где приведено дерево всех файлов проекта, следует выделить правой кнопкой мыши файл *Nod.cpp* и переименовать его в *Nod.c* (рисунки 17–18).

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 17**- Корректировка имени файла реализации

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 18** - После корректировки имени файла реализации

Затем в редакторе перенесем в этот файл текст функцию *nod*, вырезав его из файла программы, и добавим строку подключения заголовочного файла модуля. В результате файл *Nod.c* должен содержать следующий текст:

**#include "Nod.h"**

**int nod(int x, int y)**

**{**

**while (x!=y)**

**if (x>y) x=x-y;**

**else y=y-x;**

**return y;**

**}**

Вид окна с тектом файла реализации приведен на рисунке 19.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 19** – Вид окна приложения после корректировки файла реализации

Весь скорректированный текст отмечен в редакторе слева красной линией, а сам файл в поле открытые документы – \*.

Вместо функции *nod()* в файл основной программыдобавим ссылку на заголовочный файл *Nod.h*:

**#include "Nod.h"**

Вид текста основной программы *main* после корректировки приведен на рисунке 20.

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 20** – Вид окна приложения после корректировки main

После запуска скорректированной программы с теми же данными, что и для проекта без модуля, на экране консоли появляется тот же самый результат (рисунок 21).

**Изображение выглядит как текст

Автоматически созданное описание**

**Рисунок 21** – Вид окна приложения после запуска многофайлового проекта

Следует отметить, что подключение файлов модуля получает свое отражение и в файле проекта. В нем появляются имена созданных файлов и их назначение. Так видно, что к проекту подключаются файл реализации *Nod.c* и заголовочный файл *Nod.h*. Содержимое файла проекта приведено ниже.

**TEMPLATE = app**

**CONFIG += console**

**CONFIG -= app\_bundle**

**CONFIG -= qt**

**SOURCES += Nod.c main.c**

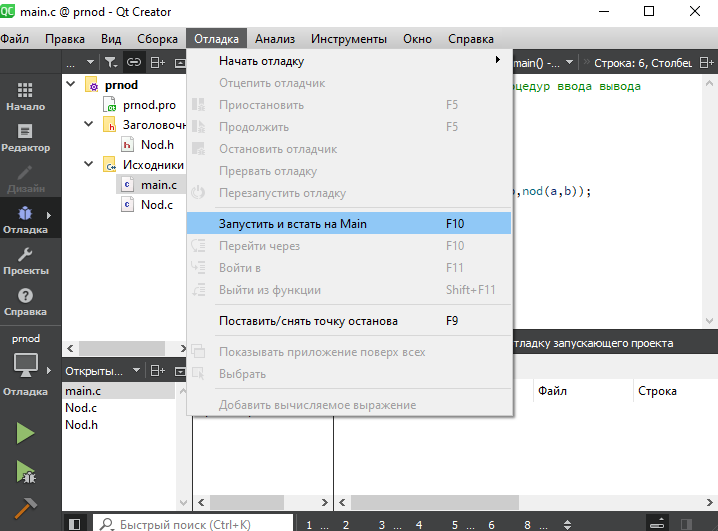
**HEADERS += Nod.h**

*Примечание*. Модуль не всегда включает два файла. Возможно создание модулей, состоящих из одного файла заголовка или файла реализации. Если файл реализации пуст, то он не создается. В программе это никак не отражается. Если не используется файл заголовка, то в проекте или других модулях, обращающихся к ресурсам этого модуля, объявленным в файле реализации, указывается непосредственно подключение файла реализации. Однако в последнем случае нарушается принцип инкапсуляции модулей, что *нетехнологично*.

Для удаления файла *из проекта* необходимо выделить этот файл на вкладке **Решение** обозревателя и нажать на клавиатуре **Delete**.

## Отладка консольных приложений

Для входа в режим отладки необходимо воспользоваться пунктом основного меню **Отладка** (рисунок 21). Чтобы пройти программу в пошаговом режиме, следует выбрать подпункт меню **Запустить и встать на main**.



**Рисунок 21** – Вид окна с выбором пошагового режима отладки

После вызова пошагового режима отладки приложение выведет окно отладки и перейдет в ожидание выполнения процесса отладки с помощью команд (рисунок 22). Окно отладки включает текст основной программы с установленной желтой стрелкой на первой исполняемой команде.

Слева от редактора расположены два информационных окна отладчика. Верхнее предназначено для вывода текущих значений всех доступных в данный момент переменных программы. В начале работы программы в переменных находятся случайные значения, так как переменные неинициализированы.

Во втором окне можно вывести значения каких-либо выражений или интересующих нас переменных. Пример такого заказного вывода будет приведен далее.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 22** – Вид окна после выбора режима пошаговой отладки

При входе в режим отладки содержимое пункта меню **Отладка** (см. рисунок 21) меняется: в нем появляются команды управления отладкой. Измененное меню пункта **Отладка** приведено на рисунке 23.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 23** – Вид меню Отладка

В этом пункте появляются следующие основные команды управления режимом:

**Продолжить (F5);**

**Перейти через (F10)** – если на данном шаге есть вызов подпрограммы, то не заходить в нее;

**Войти в (F11) -** если на данном шаге осуществляется вызов процедуры или функции, то зайти в них;

**Выйти из функции** **(Shift+F11)**  – если на данном шаге вы зашли в подпрограмм, то можно выйти из нее;

**Остановить отладчик;**

**Прервать отладку;**

**Перезапустить отладку;**

**Остановить отладку** **(****Shift+F5)**;

**Выполнить до строки** **(Ctrl+ F10)**.

После выхода из режима отладки меню среды принимает исходный вид.

Отладку программы можно выполнять в пошаговом режиме, просматривая изменяющиеся значения переменных программы. Для этого используем клавиши *F*10 и *F*11, в зависимости от того, хотим ли мы заходить в подпрограмму. На рисунках 24 и 25 показана работа в отладке с использованием этих клавиш.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 24** – Использование клавиши *F*10 для пошагового выполнения main

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 25** – Использование клавиши *F*11 для тестирования подпрограммы *nod*

*Примечание*. В процессе пошагового выполнения многофайловых проектов переключение между файлами, содержащими исходные тексты программы и модулей, происходит автоматически.

## Просмотр значений переменных в режиме отладки

Для просмотра значений переменных в процессе отладки открывают вкладки окна (рисунок 26). В верхнем правом окошке высвечиваютсязначения всех переменных выполняемой функции. По мере выполнения операторов программы значения этих переменных меняются. Однако локальных переменных в сложных функциях много, отслеживать их трудно, поэтому обычно используют второе окошко, расположенное ниже первого.

В этом окошке высвечиваются значения только переменных или выражений, указанных программистом. Добавить переменную в окно можно с помощью пункта меню **Отладка/****Добавить вычисляемое выражение** (см. рисунок 23). После выбора этого пункта меню появляется окно, в котором нужно ввести имя нужной переменной или выражение (рисунок 26). Еще проще добавить просматриваемое выражение или переменную кликнув правой клавишей мыши по области, где они должны отобразиться. В этом случае появится подменю (см. рисунок 27), в котором необходимо выбрать тоже пункт **Добавить вычисляемое выражение.** После выбора этого пункта меню появляется окно, в котором нужно ввести имя нужной переменной или выражение (рисунок 26).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 26 – Окно добавления переменной или выражения

После этого в соответствующем поле окна появится выражение или переменная, текущее значение которых в процессе отладки будет выводиться (см. рисунок 30).

Удалить добавленные выражения и переменные, можно также непосредственно кликом на удаляемое выражение. Для этого необходимо использовать появившееся после клика подменю. Можно удалить одно выражение или один идентификатор отслеживаемой переменной или сразу все выражения и переменные (рисунок 27).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27 – Удаление переменной или выражения

## Установка и отмена точек останова в программе

Часто при отладке программу целесообразно выполнять пошагово не с самого начала, а с некоторого места, подозрительного с точки зрения наличия ошибки. Для этого используют точки останова.

Задать точку останова в программе можно, щелкнув мышкой по серому полю текстового редактора перед интересующим нас оператором. Это можно сделать в процессе пошагового выполнения программы или перед ее запуском. При таком щелчке на сером поле появляется красный кружок, отмечающий точки останова (рисунок 28).

Изображение выглядит как текст, снимок экрана, монитор

Автоматически созданное описание

**Рисунок 28 –** Отметка точки останова в программе кликом мыши

Теперь, если программу запустить на выполнение, то она автоматически остановится перед выполнением отмеченного оператора. Повторный щелчок отменяет точку останова. Красный кружок при этом с серого поля исчезает.

Кроме того, точку останова можно назначить используя пункт меню **Отладка/Поставить точку останова** (рисунок 29).

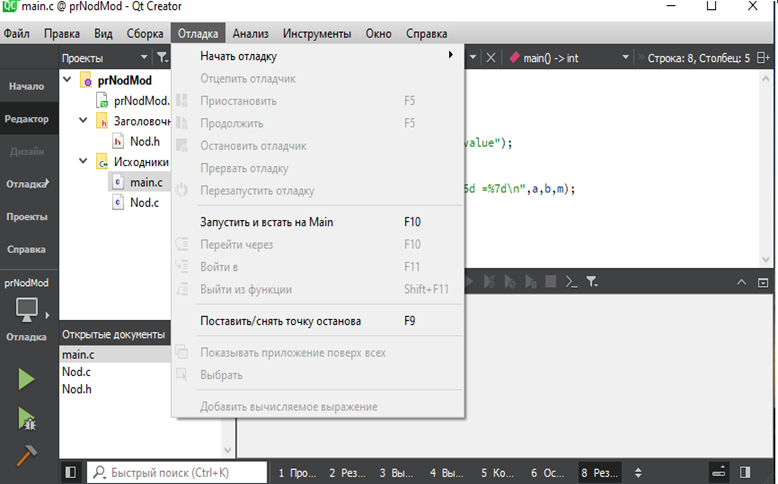


Рисунок 29 – Задание точки останова через пункт меню Отладка

Однако следует знать, что для фиксирования точки останова в этом случае, в редакторе в тексте программы нужно установить курсор на ту строку программы, на которую хотите установить точку останова.

Виды окон с использованием точки останова и клавиш *F*10 и *F*11 приведены на рисунках 30 и 31.

Изображение выглядит как текст

Автоматически созданное описание

**Рисунок 30** – Окно при отладке основной программы с помощью клавиш

Изображение выглядит как текст, снимок экрана, компьютер, внутренний

Автоматически созданное описание

Рисунок 31 – Вид окна среды при отладке функции с добавленными выражениями или переменной

# Порядок выполнения работы

2.1. Прочитать методические указания, выполняя описанные действия.

2.2. Продемонстрировать работу программы преподавателю.

2.3. Составить отчет по лабораторной работе.

2.4. Защитить лабораторную работу преподавателю.

# Требования к отчету

Отчет должен оформляться в соответствие с требованиями кафедры к отчетам. Все записи в отчете должны быть либо напечатаны на принтере, либо разборчиво выполнены от руки синей или черной ручкой (карандаш – не допускается).

Отчет должен иметь титульный лист, на котором указывается:

а) наименование факультета, кафедры и специальности;

б) название дисциплины;

в) номер и тема лабораторной работы;

г) фамилия преподавателя, ведущего занятия;

д) фамилия, имя и номер группы студента.

Кроме того отчет по лабораторной работе должен содержать:

1) тексты программ для каждого из этапов, выполняемые в среде;

2) описание последовательности действий при подготовке программы к выполнению и отладке.

3) выводы.

# Требования к защите

На защиту допускаются только студенты, выполнившие лабораторную. На защиту необходимо представить отчет, оформленный в соответствие с требованиями кафедры. Для успешной защиты студент должен ответить на вопросы преподавателя и, при необходимости, продемонстрировать разработанные программы и умение пользоваться средствами отладки.

# Контрольные вопросы

1. Как создается проект приложения в *Qt Creater*?
2. Какие действия нужно выполнить по настройки проекта для создания программы на языке *С* с вводом и выводом в консоль?
3. Какие параметры при этом необходимо указать?

4. Как запустить программу на выполнение?

5. Как добавить файлы модуля к проекту?

6. Какие средства отладки встроены в среду программирования?

7. Как создать точку останова и посмотреть значения переменных в момент останова?

# Список литературы

1. Г.С. Иванова, Т.Н. Ничушкина, Р.С. Самарев. Средства процедурного программирования Microsoft Visual С++ 2008. Учебное пособие. – М.: Издательство МГТУ им. Н.Э. Баумана, 2011.
2. В.В. Подбельский. Язык С++: Учеб. пособие. – М.: Финансы и статистика, 2006.
3. Г.С. Иванова. Программирование. Учебник для ВУЗов. – М.: Кнорус, 2013.- 432 с.
4. Г.С. Иванова. Технология программирования. Учебник для ВУЗов. – М.: Кнорус, 2013. – 336 с.

# Приложение А. Пример титульного листа.

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное учреждение  высшего образования  «Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»  (МГТУ им. Н.Э. Баумана) |

ФАКУЛЬТЕТ **ФУНДАМЕНТАЛЬНЫЕ НАУКИ**

КАФЕДРА **ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И МАТЕМАТИЧЕСКАЯ ФИЗИКА (фн11)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **02.03.01 МАТЕМАТИКА И КОМПЬЮТЕРНЫЕ НАУКИ**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 10 |

**Название: Вычисления. Погрешности вычислений.**

**Дисциплина: Информатика**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ФН11-13Б |  |  | В.П.Малышев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Т.Н.Ничушкина |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021