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 Visual C++**

**в среде Microsoft Studio 2017 Community**

Методические указания по выполнению лабораторных работ

Москва 2021

**Оглавление**

[Введение 3](#_Toc536381394)

[1 Создание заготовки консольного приложения 3](#_Toc536381395)

[2 Ввод программы 7](#_Toc536381396)

[3 Запуск программы на выполнение 8](#_Toc536381397)

[4 Модульное программирование. Файлы заголовков 10](#_Toc536381398)

[5 Отладка консольных приложений 12](#_Toc536381399)

[6 Просмотр значений переменных в режиме отладки 14](#_Toc536381400)

[7 Установка и отмена точек останова в программе 15](#_Toc536381401)

# Введение

Интегрированная среда программирования **Microsoft Visual Studio 2017 Community** предназначена для создания программ для 32х или 64х разрядных процессоров, в том числе и консольных приложений, написанных на С++.

# Создание заготовки консольного приложения

Главное окно среды при запуске имеет вид, представленный на рисунке 1. Окно включает меню, панели инструментов, текстовый редактор со стартовой страницей и окно навигатора **Обозреватель решений**. Причем последнее включает несколько вкладок, позволяющих просматривать различную информацию.

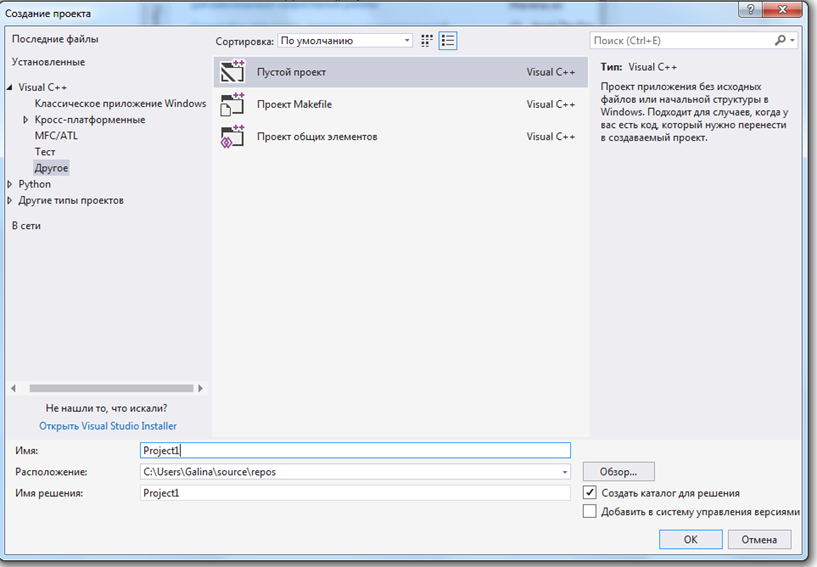


Рисунок 1 – Вид главного окна среды при отсутствии открытого проекта

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

В появившемся окне **Создание проекта** выбираем тип проекта **Visual C++** ишаблон **Пустой проект**. Вводим имя проекта **Project1** в поле **Имя** (см. рисунок 2).

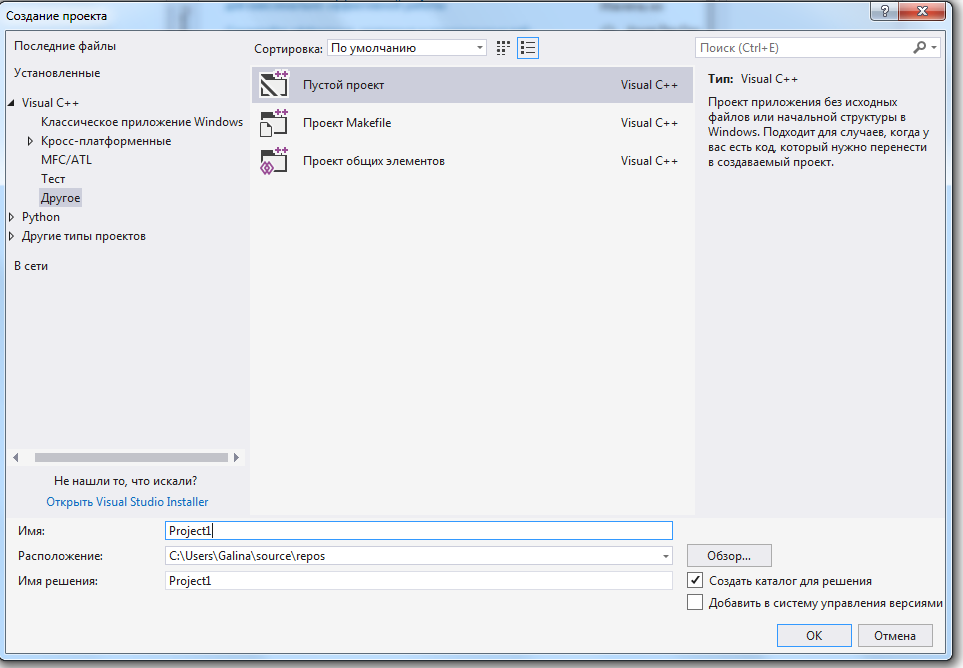


Рисунок 2 – Вид окна выбора типа проекта и задания его имени и местоположения

При желании можно указать отличное от стандартного местонахождение папки будущего проекта в позиции **Расположение** и отличное от имени проекта имя программы **Имя решения**, но особой необходимости в этом нет.

Также можно выбрать и **Классическое приложение Windows,** и шаблон **Консольное приложение**. В этом случае создается приложение «Hello, Windows», которое надо исправлять под свою программу.

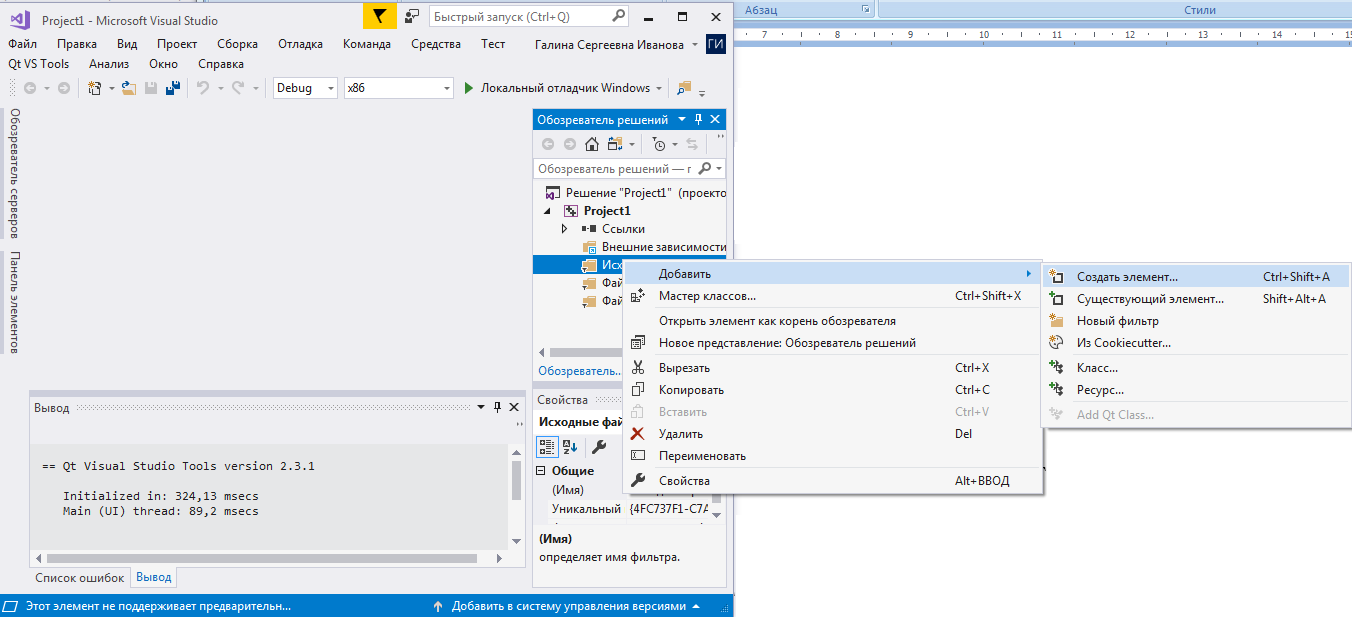
По умолчанию Visual Studio вместе с файлами проекта создает файл . pdb. Этот файл содержит информацию для работы отладчика. Например, он содержит имя файла и номер строки оператора, которые включаются в сообщение об ошибке при возникновении исключений. Формирование этого файла можно исключить, для этого следует сбросить опцию:

**Проект->Свойства->Компоновщик->Отладка->Создавать отладочную информацию: Нет.**

Для предотвращения закрытия окна консоли вместо ввода в конце программы, как это было в Delphi, устанавливаем опцию:

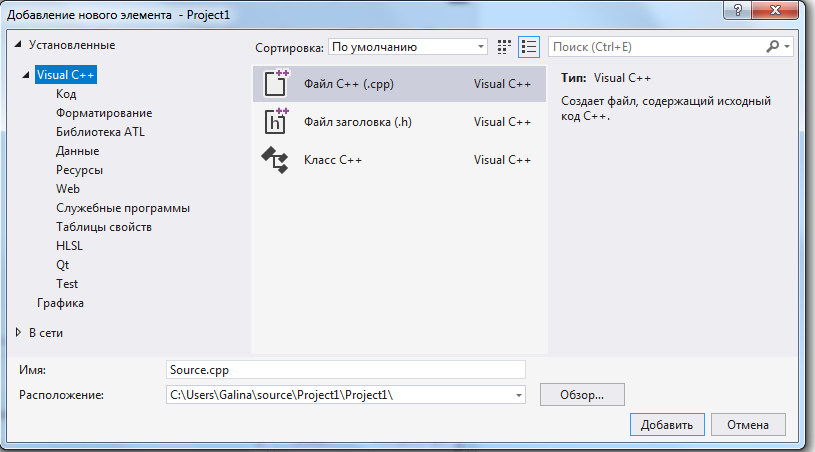
**Проект->Свойства->Компоновщик->Система->Подсистема: Консоль.**

**Пустой проект.** Если вы создали пустой проект, то к нему необходимо добавить файлы, содержащие исходный код программы. Для этого можно использовать пункт меню **Проект**/**Добавить новый элемент** или один раз щелкнуть правой кнопкой мышки по пункту Исходные файлы в Обозревателе решений (рисунок 4).

****

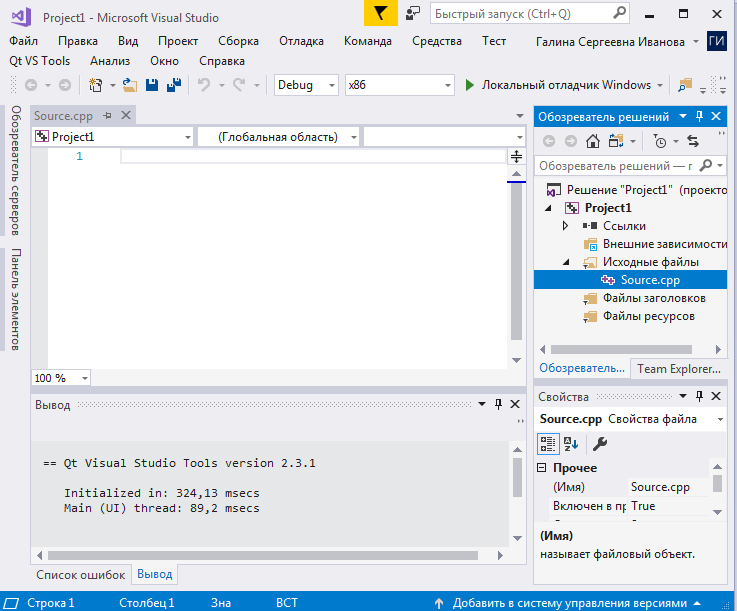
**Рисунок 4** – Добавление файлов к проекту

Тип добавляемого файла выбирается в специальном окне (рисунок 5). Первым добавляем к пустому проекту файл типа .cpp для основной программы.



**Рисунок 5** – Выбор типа добавляемого файла

Чтобы ввести текст программы необходимо открыть соответствующий файл в окне текстового редактора. Для этого переходим на вкладку **Обозреватель решений** и щелкаем правой кнопкой мыши по добавленному файлу. Если пункт Исходные файлы «закрыт», то его предварительно надо «раскрыть», щелкнув по треугольнику слева от пункта, раскрывая список исходных файлов проекта (рисунок 5).



**Рисунок 5** – Переход в окно текстового редактора для ввода программы

**Консольное приложение Windows.** При выборе консольного приложения Visual Studio высвечивает в окне редактора приложение «Hello, Windows» и некоторую дополнительную информацию:

// ConsoleApplication1.cpp: Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include "pch.h"

#include <iostream>

int main()

{

std::cout << "Hello World!\n";

}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"

// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

// Советы по началу работы

// 1. В окне обозревателя решений можно добавлять файлы и управлять ими.

// 2. В окне Team Explorer можно подключиться к системе управления версиями.

// 3. В окне "Выходные данные" можно просматривать выходные данные сборки и другие сообщения.

// 4. В окне "Список ошибок" можно просматривать ошибки.

// 5. Последовательно выберите пункты меню "Проект" > "Добавить новый элемент", чтобы создать файлы кода, или "Проект" > "Добавить существующий элемент", чтобы добавить в проект существующие файлы кода.

// 6. Чтобы снова открыть этот проект позже, выберите пункты меню "Файл" > "Открыть" > "Проект" и выберите SLN-файл.

Файл pch.h содержит предварительно откомпилированные заголовки (англ. *pre-compiled headers*). Это способ ускорить компиляцию программ за счёт предварительной обработки заголовочных файлов (h-файлов), которые содержат интерфейсы модулей и подключаются к программе путём прямой вставки их текстов в тело основной программы с помощью директивы **#include**. Важно помнить, что директива **#include pch.h** должна быть включена *во все*(!) исходные файлы проекта с расширением cpp.

Однако использование **pch**-файлов не обязательно. Его можно отменить, сбросив опцию:

**Проект->Свойства->Си/С++->Предварительно откомпилированные заголовки->Предварительно откомпилированный заголовок: Не использовать.**

Visual C++ 2017 имеет еще одну особенность: по умолчанию компилятор настроен на проверку безопасности функций, т.е. он разрешает использовать только безопасные варианты функций (с постфиксом «**\_s»** или др.), использование которых уменьшает вероятность появления той или иной ошибки. Для отмены проверки можно отключить опции контроля:

а) **Проект->Свойства->Си/С++->Создание кода->Проверка безопасности: Отключить проверку безопасности (/GS-).**

б) **Проект->Свойства->Си/С++->Общие->Проверка SDL: Нет (/sdl-).**

# 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 "pch.h"** // только при создании проекта с заголовочными файлами (!)

**#include <locale.h>** // для подключения русского языка

**#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;**

**setlocale(0,"russian");** // подключение русского языка

**puts("Введите два натуральных числа:");**

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

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

**return 0;**

**}**

Программа настроена на использование при выводе русского языка. Для этого перед программой подключена библиотека **locale.h**, а в самой программе вызвана процедура назначения необходимой кодировки. Консольная программа с такими настройками может выводить (но не вводить!) русские тексты.

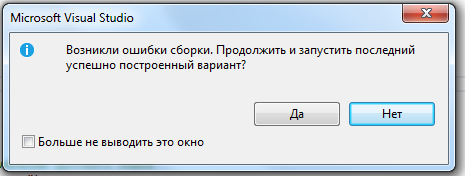
# Запуск программы на выполнение

Для запуска программы на выполнение используют команды подменю пункта главного меню **Отладка** или клавиши «быстрого доступа», показанные в скобках:

* **Начать отладку (F5)** –компиляция, компоновка и выполнение с отладкой **–** позволяет в случае выдачи сообщения об ошибке выполнения просмотреть содержимое интересующих программиста переменных (см. раздел 6)**;**
* **Запуск без отладки (Ctrl +F5)** – компиляция, компоновка и выполнение программы без отладки**.**

В обоих случаях, если готовая программа не существует, среда выдаст соответствующих запрос на выполнение компиляции и компоновки программы перед запуском.

Если при компиляции или компоновке обнаружены ошибки, то среда запрашивает, продолжить или нет создание программы (рисунок 6).



**Рисунок 6** – Запрос на запуск на выполнение программы при наличии ошибок

При отрицательном ответе на запрос среда высвечивает сообщения об ошибках в окне **Вывод**, которое появляется в нижней части окна текстового редактора. При положительном – пытается запустить программу с ошибками, что в большинстве случаев обречено на провал.

Для перехода на строку, при обработке которой зафиксирована ошибка, необходимо дважды щелкнуть мышью по соответствующему сообщению. После исправления ошибок программу необходимо вновь скомпилировать и скомпоновать.

Если компиляция и компоновка программы прошли без ошибок, то на экране появляется функционирующее в режиме консоли окно результатов, в которое выводятся сообщения программы и «эхо» вводимых символов (рисунок 7). Окно результата закрывается по нажатию любой клавиши или кнопки закрытия окна «**×**» в правом верхнем углу.

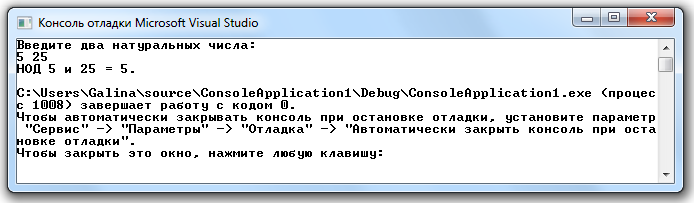


Рисунок 7 – Окно результата (изменены цвета фона и символов)

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 Visual C++ (см. разделы 5 и 6).

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

# Модульное программирование. Файлы заголовков

Среда Microsoft Visual Studio позволяет создавать и отлаживать программы, использующие не только стандартные, но и пользовательские библиотеки (модули).

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

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 файлу реализации модуля, программы или другого модуля, если они используют ресурсы описываемого модуля.

Файл реализации представляет собой аналог секции реализации модуля Pascal. Он должен содержать директивы подключения заголовочных файлов используемых модулей, описания экспортируемых процедур и функций, а также объявления и описания внутренних ресурсов модуля. Если используется проект с прекомпилируемым заголовком, то в начало каждого файла реализации необходимо поместить оператор подключения заголовочного файла pch.h: **#include** "**pch.h**". Этот файл осуществляет подсоединение библиотек среды, и при его отсутствии компилятор выдает ошибку «Не найден конец файла».

Для создания файлов заголовка и реализации модуля и *добавления их к проекту* используют пункт меню **Проект/Добавить новый элемент**. Выполнение этого пункта также как и добавление файла через вызов контекстного меню щелчком правой кнопки мыши на пункте **Файлы заголовков** Обозревателя решений вызовет открытие окна добавления файлов различных типов к проекту(рисунок 5).

В этом окне необходимо выбрать и тип подключаемого файла и ввести его имя. По данной схеме создадим два файла, образующих модуль:

* файл заголовка Nod.h – **Файл заголовка (.h)**;
* файл реализации Nod.cpp – **Файл С++ (.cpp)**.

Если используется проект с прекомпилированным файлом, то в начало файла Nod.cpp добавим строку подключения файла **pch.h:**

**#include** "**pch.h**" .

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

**#include "pch.h"** // только для проекта с прекомпилированным файлом

**#include "Nod.h"**

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

**{**

**while (x!=y)**

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

**else y=y-x;**

**return y;**

**}**

В заголовочном файле **Nod.h** объявим внешний ресурс – функцию **nod()**:

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

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

**#include ″Nod.h″**

Для навигации по многофайловому проекту используют вкладку **Решение** обозревателя, на которой высвечен список всех исходных файлов проекта.

*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*.

Если необходимо подключить модуль, файлы которого размещены в других каталогах и не могут быть скопированы в каталог текущего проекта, то в операторе **include** указывают полное имя файла, например:

**#include** "**C:\primer.cpp\con2\filemod.h**"

Файл реализации модуля с расширением **.cpp** при этом обязательно должен быть скопирован в папку проекта.

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 пошагового выполнения используют команды **Отладка/Шаг с заходом (F11)** или **Отладка/Шаг с обходом (F10)** (рисунок 8).

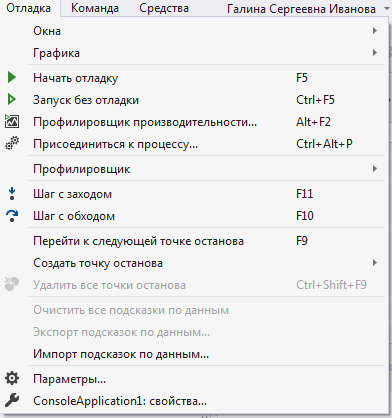


Рисунок 8 – Подменю отладки

После запуска программы в режиме отладки на экране появляется окно с текстом основной программы, в котором следующий выполняемый оператор отмечен специальной стрелкой слева (рисунок 9).

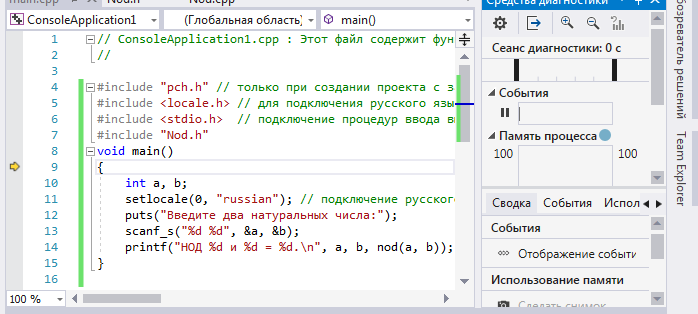
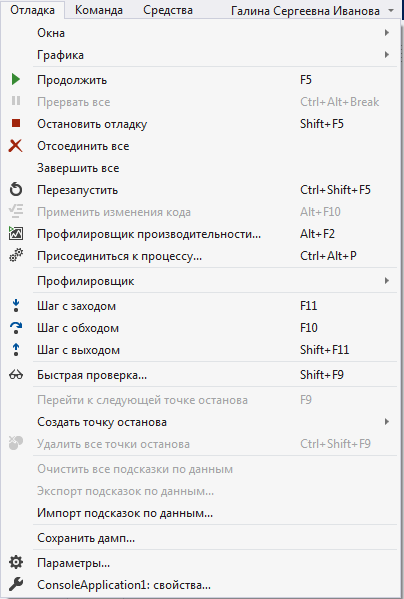


Рисунок 9 – Окно основной программы при запуске в режиме отладки

При входе в режим отладки содержимое пункта меню **Debug** изменяется (рисунок 10). В нем появляются следующие основные команды управления режимом:

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

**Перезапустить** **(Ctrl+Shift+F5)**;

**Шаг с заходом** **(F11) –** если на данном шаге осуществляется вызов процедуры или функции, то зайти в них;

Рисунок 13 – Подменю Debug, доступное в режиме отладки

**Шаг с обходом** **(F10)** – если на данном шаге есть вызов подпрограммы, то не заходить в нее;

**Шаг с выходом** **(Shift+F10)** – выполнить процедуру или функцию до конца и вернуться в вызывающую функцию.

**Рисунок 13** – Подпункты меню в режиме отладки

Рисунок 10 – Вид меню во время отладки

После выхода из режима отладки меню среды принимает исходный вид.

Отладку программы можно выполнять в пошаговом режиме, просматривая изменяющиеся значения переменных, или устанавливая точки останова в ключевых точках программы.

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 **Вывод**, появляющегося в нижней части окна среды при запуске программы на выполнение: вкладки **Видимые**, **Локальные,** **Контрольные значения 1**.

На вкладке **Локальные** (рисунок 11) 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 вкладку **Контрольное значение 1**.

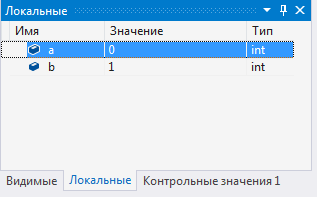


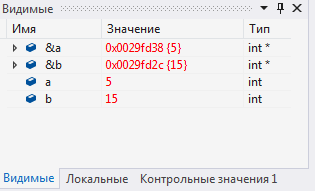
Рисунок 11 – Окно Locals

На вкладке **Контрольное значение 1** высвечиваются значения только переменных, указанных программистом (рисунок 12). Добавить переменную в окно **Контрольное значение 1** можно, введя ее имя в поле **Имя**. Также непосредственно в окне, можно удалить идентификатор отслеживаемой переменной.

# Рис 12.png

Рисунок 12 – Окно Контрольное значение 1

Очень удобно при пошаговом выполнении также окно **Видимые** (рисунок 13). В нем высвечиваются значения переменных текущего и следующего оператора программы. Особенно это удобно, если отлаживаемый фрагмент сложен и использует большое количество переменных.

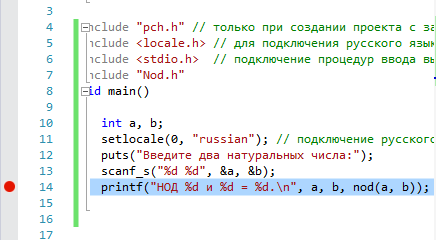


**Рисунок 13** – Окно Видимые

# 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 останова программы в этом месте используют точки останова.

Задать точку останова в программе проще всего, 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 14).



**Рисунок 14 –** Отметка точки останова в программе

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.