

Ravesli [Ravesli](#)

- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegExr](#)
- [Ассемблер](#)
- [Купить .PDF](#)

Урок №20. Многофайловые программы

Юрий |

- [Уроки C++](#)

Обновл. 2 Сен 2020 |

73803

42

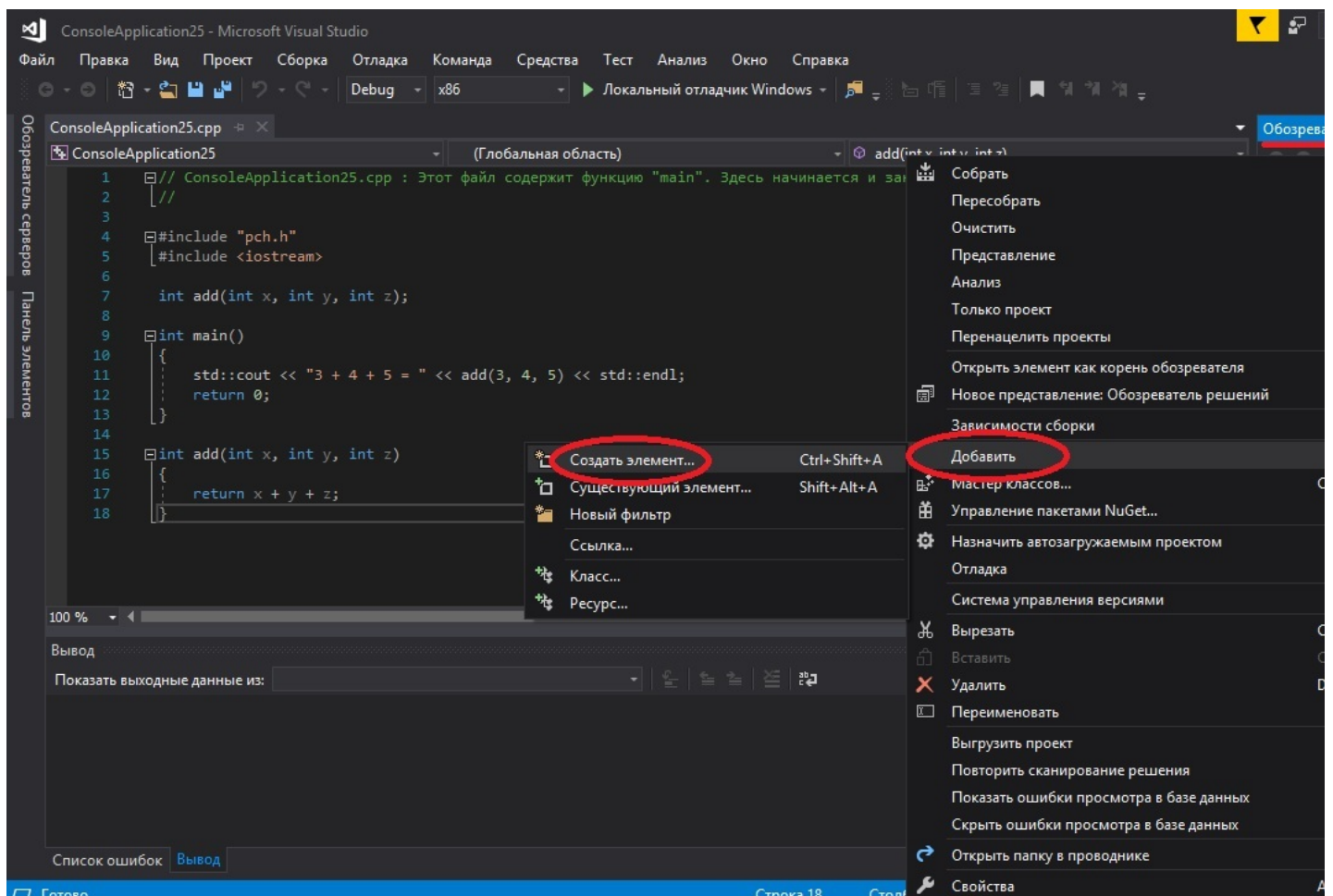
Как только программы становятся больше, их следует разбивать на несколько файлов (в целях удобства и улучшения функциональности). Одним из преимуществ использования [IDE](#) является легкость в работе с n-ным количеством файлов. Мы уже знаем, как создавать и компилировать однофайловые проекты, добавление новых файлов не составит труда.

Оглавление:

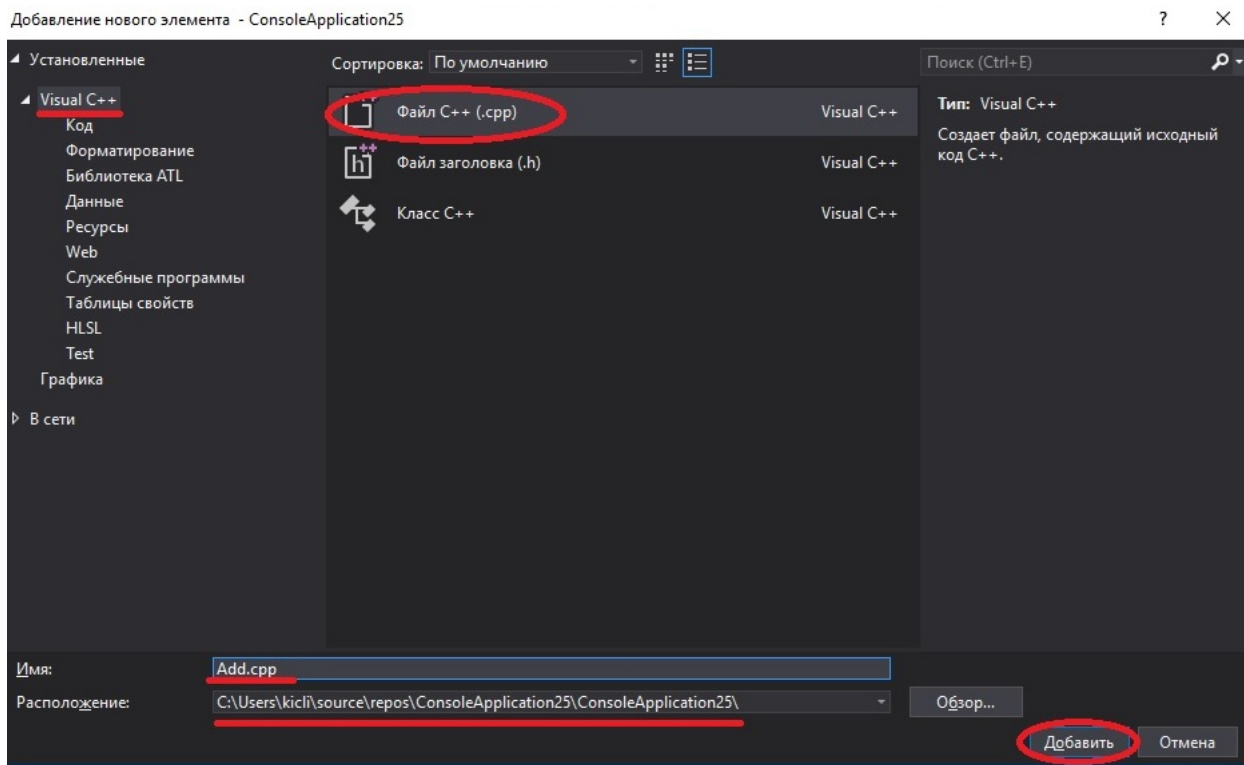
1. [Многофайловые проекты в Visual Studio](#)
2. [Многофайловые проекты в Code::Blocks](#)
3. [Многофайловые проекты в GCC/G++](#)
4. [Пример многофайловой программы](#)
5. [Что-то пошло не так!](#)
6. [Тест](#)

Многофайловые проекты в Visual Studio

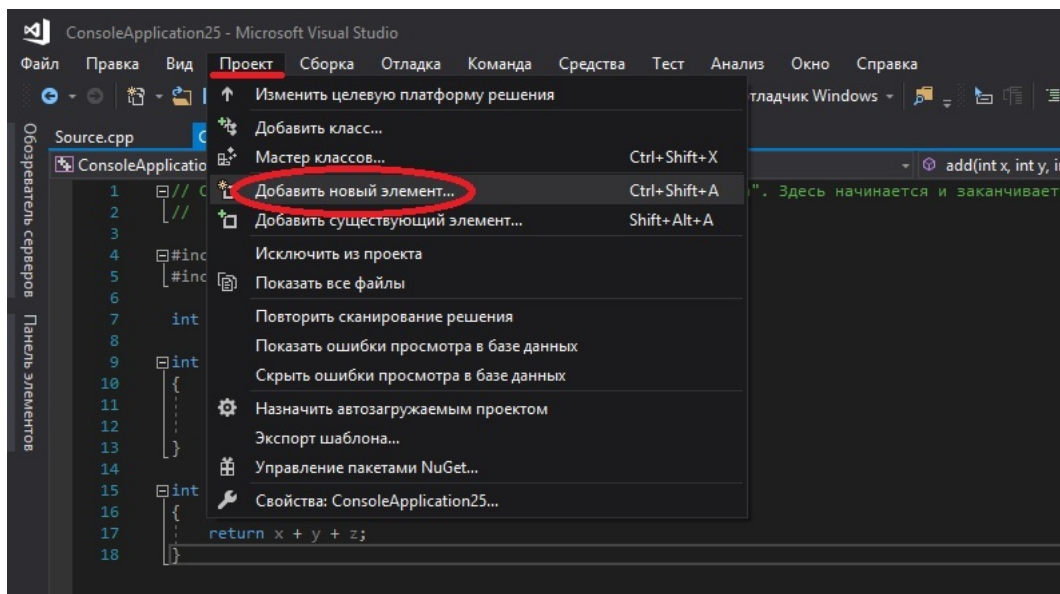
В Visual Studio щелкните правой кнопкой мыши по имени вашего проекта в "Обозревателе решений", затем "Добавить" > "Создать элемент...":



Во всплывающем диалоговом окне выберите тип файла, укажите его имя, расположение, а затем нажмите "Добавить":

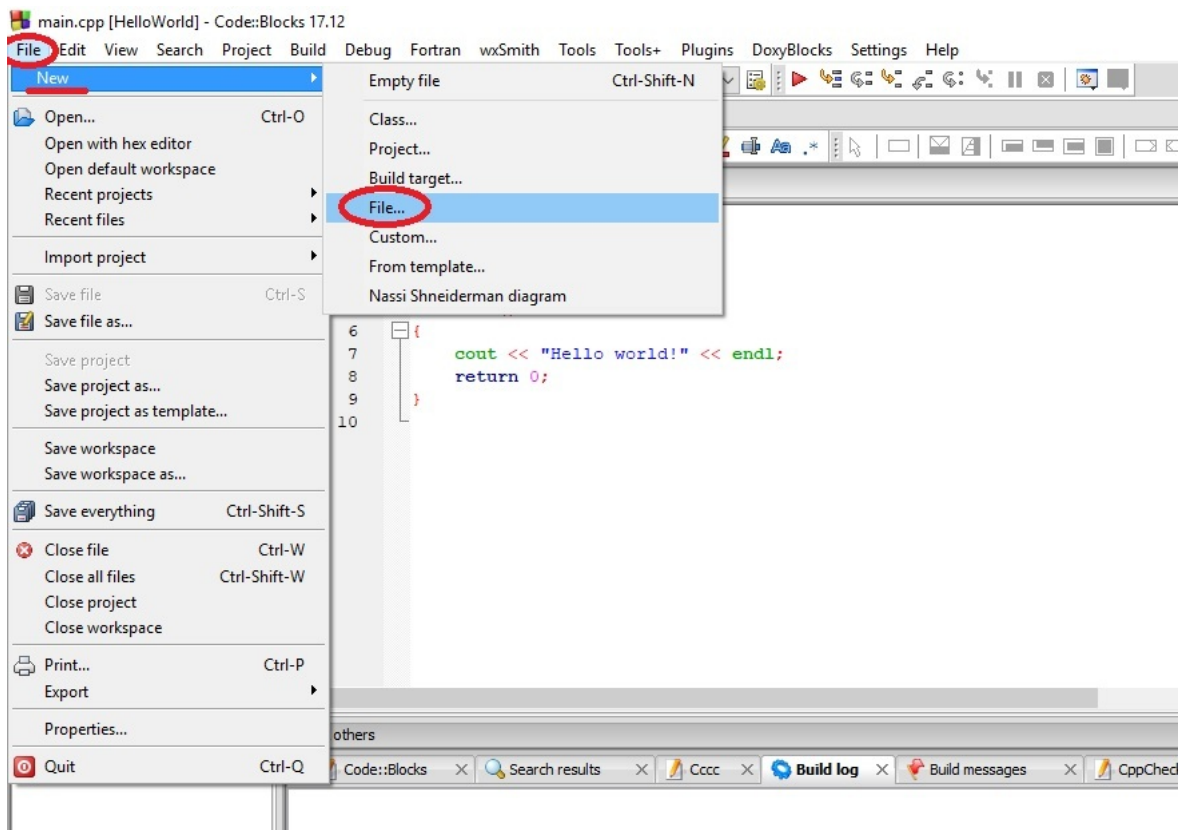


Также вы можете добавлять файлы к вашему проекту через "Проект" > "Добавить новый элемент...":

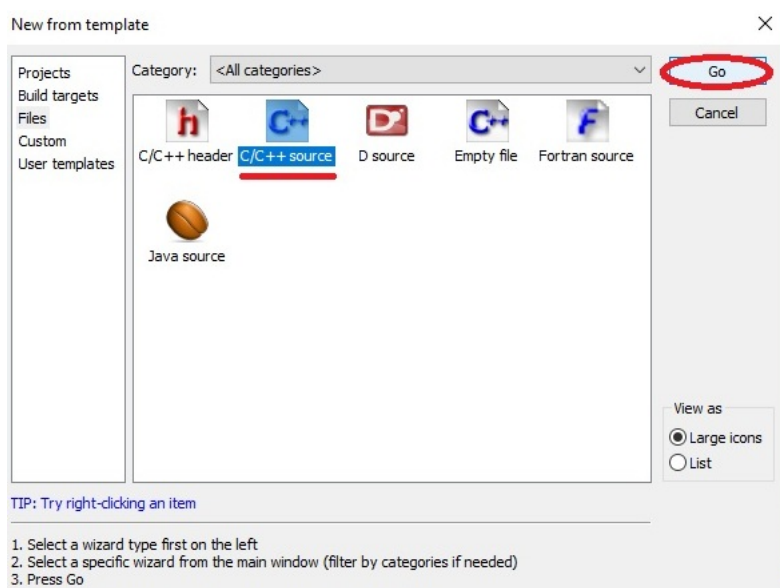


Многофайловые проекты в Code::Blocks

В Code::Blocks перейдите в "File" > "New" > "File...":



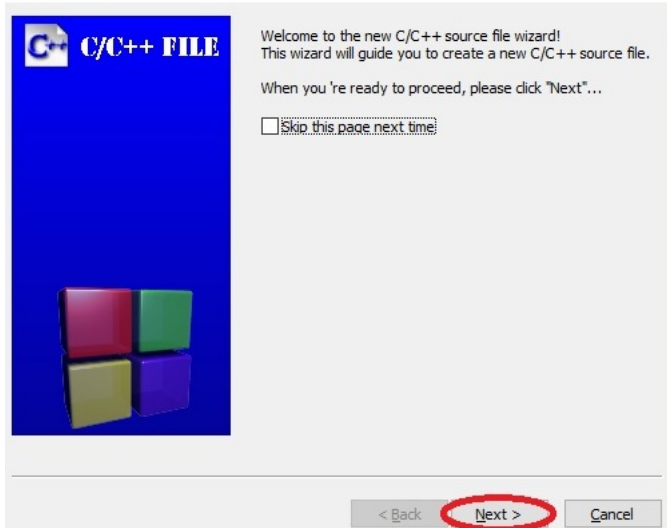
Затем выберите "C/C++ source" и нажмите "Go":



Затем "Next" (этого окна может и не быть):

C/C++ source

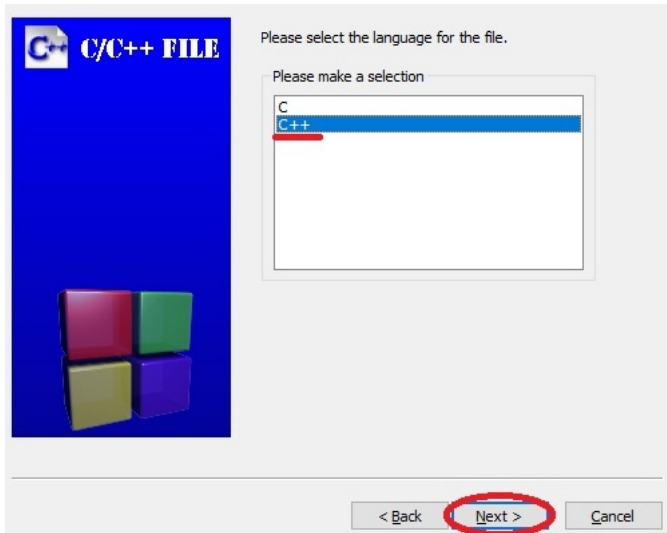
✕



Затем "C++" и опять "Next":

C/C++ source

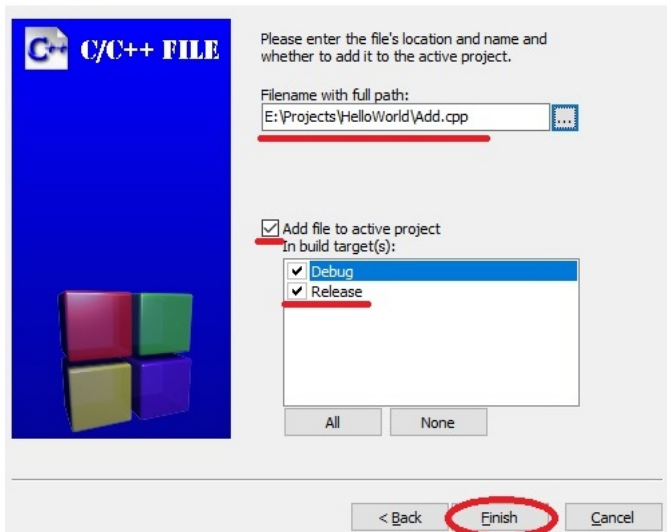
✕



Затем укажите имя нового файла (не забудьте расширение .cpp) и его расположение (нажмите на троечку и выберите путь). Убедитесь, что поставлены все три галочки (они отвечают за [конфигурации сборки](#)). Затем нажмите "Finish":

C/C++ source

✕



Готово! Файл добавлен.

Многофайловые проекты в GCC/G++

В командной строке вам нужно будет создать файл, указать его имя и подключить к компиляции, например:

```
g++ main.cpp add.cpp -o main
```

(где *main.cpp* и *add.cpp* — это имена файлов с кодом, а *main* — это имя файла-результата)

Пример многофайловой программы

Рассмотрим следующую программу, которая состоит из двух файлов.

add.cpp:

```
1 int add(int x, int y)
2 {
3     return x + y;
4 }
```

main.cpp:

```
1 #include <iostream>
2
3 int main()
4 {
5     std::cout << "The sum of 3 and 4 is: " << add(3, 4) << std::endl;
6     return 0;
7 }
```

Попробуйте запустить эту программу. Она не скомпилируется, вы получите следующую ошибку:

add: идентификатор не найден

При компиляции кода, компилятор не знает о существовании функций, которые находятся в других файлах. Это сделано специально, чтобы функции и переменные с одинаковыми именами, но в разных файлах, не вызывали конфликт имен.

Тем не менее, в данном случае, мы хотим, чтобы *main.cpp* знал (и использовал) функцию *add()*, которая находится в *add.cpp*. Для предоставления доступа *main.cpp* к функциям *add.cpp*, нам нужно использовать предварительное объявление:

```
1 #include <iostream>
2
3 int add(int x, int y); // это нужно для того, чтобы main.cpp знал, что функция add() определена в другом месте
4
5 int main()
6 {
7     std::cout << "The sum of 3 and 4 is: " << add(3, 4) << std::endl;
8     return 0;
9 }
```

Теперь, когда компилятор будет компилировать *main.cpp*, он будет знать, что такое *add()*. Попробуйте запустить эту программу еще раз.

Что-то пошло не так!

Есть много вещей, которые могут пойти не так, особенно, если вы это делаете в первый раз. Главное — не паниковать:

Пункт №1: Если вы получили ошибку от компилятора, что функция *add()* не определена в *main()*, то, скорее всего, вы забыли записать предварительное объявление функции *add()* в *main.cpp*.

Пункт №2: Если вы получили следующую ошибку от линкера:

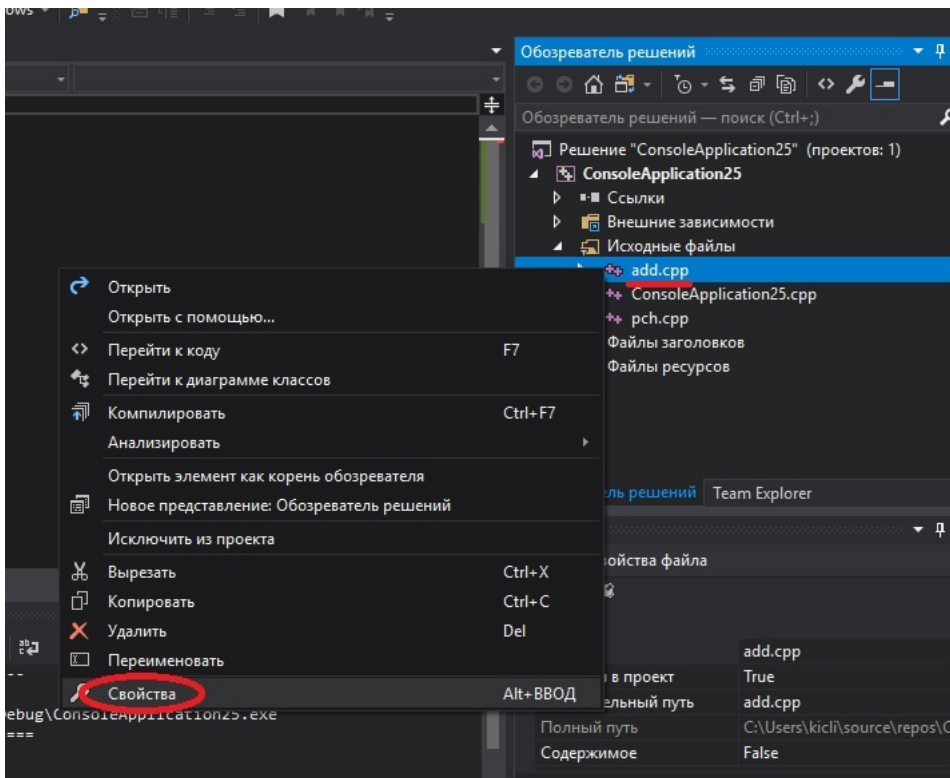
```
unresolved external symbol "int __cdecl add(int,int)" (?add@@YAHNN@Z) referenced in function _main
```

то возможных решений есть несколько:

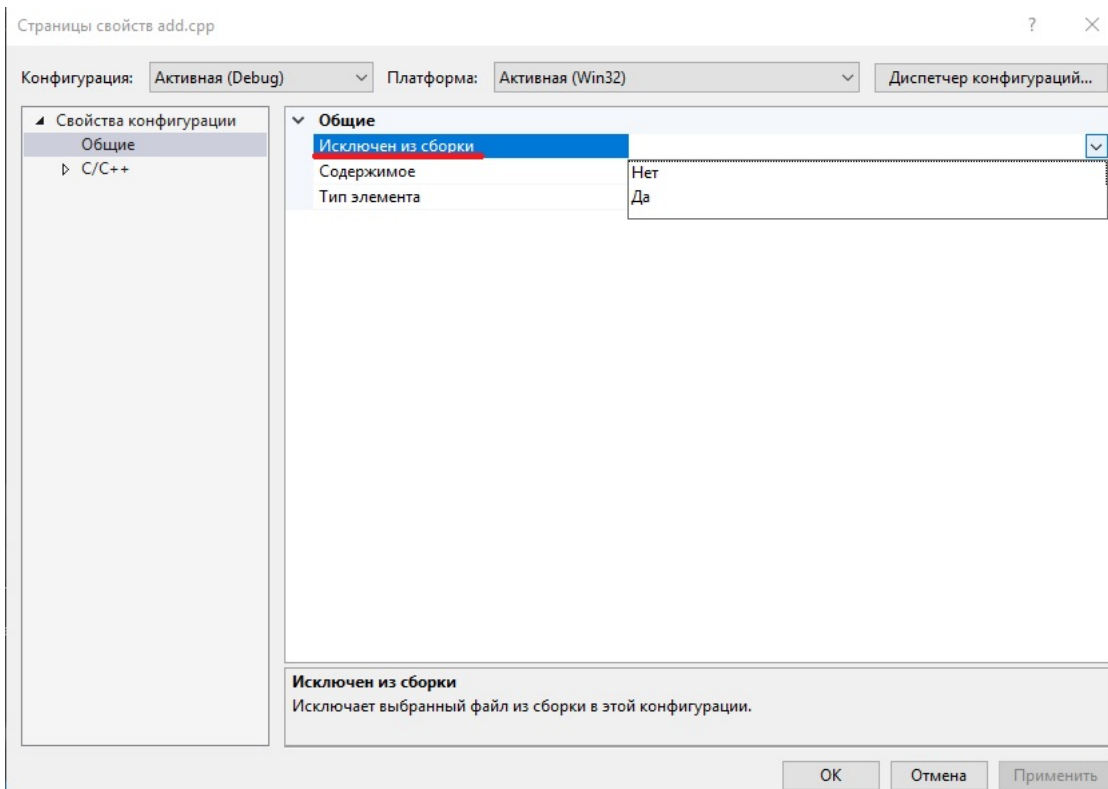
а) Скорее всего, *add.cpp* некорректно добавлен в ваш проект. Если вы используете Visual Studio или Code::Blocks, то вы должны увидеть *add.cpp* в "Обозревателе решений" в списке файлов вашего проекта или в панели проекта IDE. Если добавленного файла нет, то щелкните правой кнопкой мыши по вашему проекту и добавьте файл, как это показано выше, а затем повторите попытку компиляции вашего проекта.

б) Вполне возможно, что вы добавили *add.cpp* к другому проекту.

в) Вполне возможно, что добавленный файл не подключен к компиляции/линкингу. Щелкните правой кнопкой мыши по имени вашего добавленного файла и выберите "Свойства":



Убедитесь, что пункт "Исключен из сборки" оставлен пустым или выбрано значение "Нет":



Пункт №3: Не следует писать следующую строку в `main.cpp`:

```
1 #include "add.cpp"
```

Наличие этой строки приведет к тому, что компилятор вставит всё содержимое `add.cpp` непосредственно в `main.cpp` вместо того, чтобы рассматривать эти файлы как отдельные.

Тест

Разделите следующую программу на два файла (`main.cpp` и `input.cpp`): `main.cpp` должен содержать функцию `main()`, а `input.cpp` — функцию `getInteger()`.

Помните, что для функции `getInteger()` вам понадобится предварительное объявление в `main.cpp`.

```
1 #include <iostream>
```

```

2
3 int getInteger()
4 {
5     std::cout << "Enter an integer: ";
6     int x;
7     std::cin >> x;
8     return x;
9 }
10
11 int main()
12 {
13     int x = getInteger();
14     int y = getInteger();
15
16     std::cout << x << " + " << y << " is " << x + y << '\n';
17     return 0;
18 }

```

Ответ

input.cpp:

```

1 #include <iostream>
2
3 int getInteger()
4 {
5     std::cout << "Enter an integer: ";
6     int x;
7     std::cin >> x;
8     return x;
9 }

```

main.cpp:

```

1 #include <iostream>
2
3 int getInteger(); // предварительное объявление функции getInteger()
4
5 int main()
6 {
7     int x = getInteger();
8     int y = getInteger();
9
10    std::cout << x << " + " << y << " is " << x + y << '\n';
11    return 0;
12 }

```

Оценить статью:

 (604 оценок, среднее: 4,87 из 5)

[← Урок №19. Прототип функции и Предварительное объявление](#)

[Урок №21. Заголовочные файлы →](#)
Комментариев: 42

1. **Рахим Умурзаков:**
[22 июля 2020 в 00:20](#)

Спасибо, Юрий! Замечательные уроки! Получаю удовольствие.. Возникают вопросы.. ишу на сайтах.. нахожу.. Будут посерьёзнее, намерен обращаться, если позволите!

[Ответить](#)


1. **Юрий:**
[22 июля 2020 в 14:58](#)

Пожалуйста))

[Ответить](#)



2. *Никита:*
[7 июня 2020 в 12:22](#)

Мне кажется стоило также указать что хорошим тоном является написать функцию в <name>.cpp, сделать её объявление в <name>.h, после чего подключить <name.h> в основном файле.

Вместо того чтобы писать объявления функций напрямую.

[Ответить](#)



3. *Андрей:*
[6 мая 2020 в 18:31](#)

Первый файл с функцией int getInteger():

```
1 #include <iostream>
2 #include <string>
3 #include <cstdlib>
4 using namespace std;
5
6 int getInteger()
7 {
8     setlocale(LC_ALL, "ru");
9     cout << "Введи число: ";
10    int x;
11    cin >> x;
12    return x;
13 }
```

Второй файл(главный):

```
1 using namespace std;
2
3 int getInteger();
4
5 int main()
6 {
7     setlocale(LC_ALL, "ru");
8     int x = getInteger();
9     int y = getInteger();
10
11    cout << x << " + " << y << " is " << x + y << '\n';
12    return 0;
13 }
```

[Ответить](#)



4. *Фродо:*
[17 февраля 2020 в 23:55](#)

Может кто-нибудь знает как это все осуществить при помощи Xcode, или статью какую-нибудь по обучению использования этой среды???

[Ответить](#)



1. *Петр:*
[2 апреля 2020 в 09:52](#)

Скачай VisualStudio

[Ответить](#)



5. *Александр:*
[25 января 2020 в 07:13](#)

Здравствуйте Юрий! Благодарю вас за интересные и познавательные уроки! Стараюсь изучить все тонкости. Вы пожалуйста не бросайте этот сайт. Я думаю что развитие нужно направить в сторону 3D. Как вы на это смотрите?

[Ответить](#)



1. *Юрий:*
[25 января 2020 в 15:43](#)

Пожалуйста)

[Ответить](#)

6. *Somon:*[23 ноября 2019 в 22:03](#)

Здравствуйте я пишу в андроид с помощью Dcoder и хотел пройти тест в конце этого урока но у меня выдаёт ошибку.

```
/tmp/cc5DeaBV.o: In function main':
source.cpp:(.text.startup+0x19): undefined reference to `getInteger()'
source.cpp:(.text.startup+0x20): undefined reference to `getInteger()'
collect2: error: ld returned 1 exit status
```

А вот коды

```
1 #include <iostream>
2 int getInteger()
3 {
4     std::cout << "Enter an integer: ";
5     int x;
6     std::cin >> x;
7     return x;
8 }
9
10 #include <iostream>
11 int getInteger();
12 int main()
13 {
14     int x = getInteger();
15     int y = getInteger();
16
17     std::cout << x << " + " << y << " is " << x + y << '\n';
18     return 0;
19 }
```

[Ответить](#)7. *Tor:*[6 июля 2019 в 20:40](#)

В VS 2019 все компилируется и без #include "pch.h". А если как раз использую #include "pch.h" выдает ошибки "не удается открыть источник файл "pch.h" "и "не удается открыть файл включения: pch.h: No such file or directory" стоит ли пытаться это исправить или оставить как есть?

[Ответить](#)8. *Алексей:*[21 июня 2019 в 13:40](#)

Сначала разделил верно, потом думал насчет iostream в инпуте, вот только зря добавил в саму функцию инициализацию переменной. Взглянул в ответ — понял сразу ошибку.

Спасибо за уроки, хороший язык оказывается.

[Ответить](#)9. *Bohdan:*[17 апреля 2019 в 18:23](#)

Срочно нужна помощь.

Начал программировать на андроид с приложения Dcoder(c++: GCC compiler 6.3 (знаю, что это неудобно и т. д., но лучше так, чем никак).

С этим уроком получилась зиминка из-за моего не состояния найти способ, как связать несколько файлов именно в этом приложении. Если кто-то что с этим делать, тогда пишите (буду очень благодарен). Автору спасибо за уроки. Всё очень доходчиво написано.

[Ответить](#)10. *Дмитрий:*[7 апреля 2019 в 17:19](#)

Большое спасибо автору, читаю уроки — не могу оторваться (давно хотел изучить язык C++).

У меня такой вопрос: почему при определении (и описании соответственно) функции getInteger() мы оставили скобки пустыми, (ведь мы задействуем 1 переменную и возвращаем ее значение)?

[Ответить](#)1. *Константин:*[8 мая 2020 в 13:46](#)

Дык мы ж не собираемся в эту функцию никаких значений всовывать — вот и параметры в ней нече заявлять!

[Ответить](#)

11. *Андрей:*
[26 марта 2019 в 15:51](#)

Все трудности с ошибками прошел, разобрался, помогли подсказки. Осталось не понятным — '\n', что оно означает?

[Ответить](#)

1. *Павел:*
[28 марта 2019 в 13:55](#)

Это символ переноса строки, такая управляющая последовательность(и гуглите 😊)

[Ответить](#)

12. *Артем:*
[19 марта 2019 в 23:24](#)

Какая же чудесная находка для меня эти уроки) Автору огромное спасибо за проделанную работу за перевод и адаптацию

[Ответить](#)

1. *Юрий:*
[19 марта 2019 в 23:52](#)

Пожалуйста 😊

[Ответить](#)

13. *Иван:*
[1 февраля 2019 в 16:29](#)

Здравствуйте. у меня возникла небольшая заминка. Не могли бы вы помочь мне в её разрешении?

Я создал оба файла, как объяснялось в задании, добавил add.cpp (Добавить новый элемент —> файл.cpp). И в общем всё работает, но в файле add.cpp возникла ошибка: "IntelliSense: не удастся открыть источник файл "stdafx.h" ".

Помогите разобраться.

[Ответить](#)

1. *Юрий:*
[1 февраля 2019 в 16:41](#)

Если у вас Visual Studio 2017, то вам нужно `#include <pch.h>` вместо `stdafx.h`.

[Ответить](#)

14. *Станислав:*
[7 января 2019 в 19:26](#)

Здравствуйте. Как добавлять файлы в Visual Studio 2017 если он на русском. Просто не очень понятно, переводчик не помогает.

[Ответить](#)

15. *данила:*
[13 декабря 2018 в 15:29](#)

не получается разделить функцию `main()` на 2 файла (без возвратных функций)...это вообще возможно?)

[Ответить](#)

1. *Константин:*
[22 декабря 2018 в 22:27](#)

Данила, функция это тот же блок, т.е. некий цельный кусок, который из своих недр выдаёт на гора некое одно значение, например — превращает в число результат неких математических вычислений. Это значение можно впихнуть в другой блок, который, используя его произведёт также одно значение, но смысл последнего м.б. другим, например — выведется какая-нибудь надпись. Смыслы закладывает кодер (как и всё остальное). А функцию `main()` — вообще воспринимай как оглавление книги, в котором записаны вызовы всех функций, которые, собственно, и выполняют поставленные кодером задачи.

[Ответить](#)

16. *Александр:*
[5 октября 2018 в 03:22](#)

После быстрого просмотра десятка-другого уроков понял, что уроки-то неплохие (не считая некоторых грамматических ошибок), да вот возникло стойкое чувство déjà vu. И правда, как оказалось, этот курс — просто перевод с минимальными изменениями курса, представленного на сайте LearnCpp.com.

И хотя я искренне считаю, что людям, не владеющим английским языком, в программировании категорически делать нечего, все же ничего не имею против переводов хороших источников.

Но только если автор честно пишет, что это не его детище, а просто перевод. Я не смог найти никакого упоминания или ссылки на оригинал. Мало того, вы, Юрий, еще и пытаетесь денег на этом заработать, продавая книжку с этим добром. Если я ошибся, прошу прощения, но если нет, то это очень грустно.

[Ответить](#)



1. Юрий:

[5 октября 2018 в 13:31](#)

А где же, Александр, я писал, что это моё детище? Моё детище — это сайт Ravesli, а уроки C++ — это перевод. На странице "Уроки C++" пролистайте до конца — там источник уроков.

Насчет денег — да, пытаюсь. Только перевод, а не оригинал, да ещё и с "минимальными изменениями". Уроки по C++ останутся бесплатными на сайте до тех пор, пока я буду поддерживать этот сайт, а всё что дополнительно — то это уже на моё усмотрение.

[Ответить](#)



1. Александр:

[5 октября 2018 в 19:55](#)

Тогда я ошибся, и это чудесно => Прошу прощения. Я не заметил ссылки, привык их искать где-то в аннотации.

[Ответить](#)



1. Юрий:

[5 октября 2018 в 23:38](#)

Ничего, без проблем 😊



17. Леонид:

[29 июня 2018 в 08:22](#)

Добрый день! При попытке добавить в функцию, которая находится в отдельном файле, "stdafx.h" компилятор выдаёт ошибку: не удаётся открыть файл включения stdafx.h: No such file or directory. Без stdafx.h всё нормально работает. Visual Studio 2015.

[Ответить](#)



1. Юрий:

[2 июля 2018 в 22:37](#)

Скорее всего у вас отключены предварительно скомпилированные заголовки, коим и является stdafx.h. Работает без него — хорошо, работает с ним — ещё лучше.

[Ответить](#)



18. Марина:

[22 июня 2018 в 08:09](#)

Сегодня читала статью с телефона. Очень удобный сайт)

[Ответить](#)



1. Юрий:

[22 июня 2018 в 23:41](#)

Можно и с телефона, и с планшета, и с компьютера, и с ноутбука — лишь бы желание было, а возможность предоставим 😊

[Ответить](#)



19. Илья:

[7 июня 2018 в 15:45](#)

Когда пишу файл add.cpp постоянно не идёт, это ж линкер мешает по моему??? И что ему не так, когда твою программу скопировал, также всё:

Ошибка LNK2019 ссылка на неразрешенный внешний символ _main в функции "int __cdecl invoke_main(void)" (?invoke_main@@@YAHXZ) add.cpp 1

[Ответить](#)



1. Юрий:

[7 июня 2018 в 19:14](#)

Вы сделали всё как в уроке? Правильно добавили файл? К тому проекту, что нужно?

В уроке есть отдельный пункт, где рассказываются возможные ошибки — ваша ошибка — это та же, что и в уроке. Там есть возможные решения.

[Ответить](#)



20. [Максим\(\)](#):

[8 мая 2018 в 16:33](#)

Почему в VS нужно обязательно подключать библиотеку stdafx.h? Раньше, когда я учился по видеоурокам, я писал программы и без ее подключения. Но конечно есть одно но, я тогда немного изменил параметры создания проекта: поставил галочку на "Empty project" и удалил галочку на "Security Development Lifecycle (SDL) checks ". Может быть это как-то повлияло?

[Ответить](#)



1. [Юрий](#):

[10 мая 2018 в 17:46](#)

stdafx.h — это реализация механизма «предварительно скомпилированных заголовков», который используется для ускорения процесса сборки проектов. В этом файле содержатся вызовы других библиотек и заголовочных файлов, которые нужны для корректного выполнения ваших программ. Т.е. чтобы не прописывать дополнительно подключение определенных библиотек и заголовочных файлов — эти вызовы записали в одном файле и подключают только stdafx.h (1 строчка кода), а не прописывают дополнительно 20 строчек `#include`. Но от этого можно увидеть толк и почувствовать реальную пользу, когда ваши проекты состоят минимум из десятка файлов.

Вы можете отключить использование предварительно скомпилированных заголовков в Visual-е в настройках и тогда вам не нужно будет прописывать в каждом файле строчку с подключением stdafx.h. Если вы будете писать простенькие программы на один-два файла, то вам что с stdafx.h, что без него — разница небольшая. Если же будете писать сложные проекты с десятками файлов, то отключив использование stdafx.h, вам придется вручную дополнительно прописывать в каждом файле подключение требуемой библиотеки или заголовочного файла, вызовы которых до этого находились в файле stdafx.h.

Отключается stdafx.h в Visual Studio так: *Project (Проект) -> Properties (Свойства) -> C/C++ -> Precompiled Headers (Предварительно скомпилированные заголовки)* и выбираете пункт «Not using Precompiled Files» (сокр. «Не использовать»).

По поводу того, как повлияли галочки возле «Empty Project» и «Security Development Lifecycle checks» на отключение механизма «предварительно скомпилированных заголовков» я ничего не могу сказать.

[Ответить](#)



21. [Андрей Оганесян](#):

[2 апреля 2018 в 22:50](#)

Зачем писать названия в VS (например Source Files) на английском, если в VS есть русский язык?

[Ответить](#)



1. [Юрий](#):

[2 апреля 2018 в 23:06](#)

Это уже дело вкуса, кому как привычнее.

[Ответить](#)



1. [Андрей Оганесян](#):

[2 апреля 2018 в 23:21](#)

Я имел в виду, что кому то может быть непонятно что значит то или иное название. Мне, как новичку, например, пришлось прибегнуть к гугл-переводчику, чтобы узнать какие это файлы source

[Ответить](#)



1. [Юрий](#):

[2 апреля 2018 в 23:32](#)

Хотите вы этого или нет, но английский, даже самый базовый, знать вам придется. Любые команды, ключевые слова и прочее в коде — это обычные английские слова, которые имеют определенные значения. Писать транслитом (русские слова английской раскладкой) названия файлов, переменных, функций, классов — это, как минимум, некрасиво и неудобно. Что тут делать? Учить английский.



2. [Данила](#):

[12 декабря 2018 в 15:45](#)

+).. тоже в гугл лез)))



2. *Андрей Оганесян:*

[2 апреля 2018 в 23:34](#)

Согласен, спасибо

[Ответить](#)



1. *Юрий:*

[3 апреля 2018 в 00:27](#)

Понимаю, полным новичкам могут быть непонятны некоторые слова, но от этого никуда не деться. Там, где можно упростить — я упрощаю, что можно перевести — я перевожу, но всё перевести нельзя. Сам синтаксис любого языка программирования — это английский. И тут уже нужно подстраиваться.

Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Комментарий

☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию

☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.






[TELEGRAM](#)  [КАНАЛ](#)

Электронная почта



[ПАБЛИК](#) 

ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020