

Ravesli [Ravesli](#)

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





Урок №52. Связи, область видимости и продолжительность жизни

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 22 Сен 2020 |

 24479

[1](#)  [5](#)

Мы уже ранее рассматривали, что такое область видимости, продолжительность жизни, связи и то, какими они могут быть в языке C++. Давайте сейчас закрепим это всё.

Оглавление:

1. [Область видимости](#)
2. [Продолжительность жизни](#)
3. [Связи](#)
4. [Резюмируем](#)
5. [Предварительные объявления](#)

Область видимости

Область видимости идентификатора определяет, где он доступен для использования. К идентификатору, который находится вне области видимости, доступ закрыт.

■ Переменные с **локальной/блочной областью видимости** доступны только в пределах блока, в котором они объявлены. Это:

- ➔ локальные переменные;
- ➔ параметры функции.

- Переменные с **глобальной/файловой областью видимости** доступны в любом месте файла. Это:

- глобальные переменные.

Продолжительность жизни

Продолжительность жизни переменной определяет, где она создается и где уничтожается.

- Переменные с **автоматической продолжительностью жизни** создаются в точке определения и уничтожаются при выходе из блока, в котором определены. Это:

- обычные локальные переменные.

- Переменные со **статической продолжительностью жизни** создаются, когда программа запускается, и уничтожаются при её завершении. Это:

- глобальные переменные;
- статические локальные переменные.

- Переменные с **динамической продолжительностью жизни** создаются и уничтожаются по запросу программиста. Это:

- динамические переменные (о них мы поговорим на соответствующем уроке).

Связи

Связь идентификатора определяет, относятся ли несколько упоминаний одного идентификатора к одному и тому же идентификатору или нет.

- Идентификаторы **без связей** — это идентификаторы, которые ссылаются сами на себя. Это:

- обычные локальные переменные;
- пользовательские типы данных, такие как `enum`, `typedef` и классы, объявленные внутри блока (об этом детально поговорим на соответствующих уроках).

- Идентификаторы с **внутренней связью** доступны в любом месте файла, в котором они объявлены. Это:

- статические глобальные переменные (инициализированные или неинициализированные);
- константные глобальные переменные;
- статические функции (о них поговорим чуть позже).

- Идентификаторы с **внешней связью** доступны как в любом месте файла, в котором они объявлены, так и в других файлах (через предварительное объявление). Это:

- обычные функции;

- ➔ неконстантные глобальные переменные (инициализированные или неинициализированные);
- ➔ внешние константные глобальные переменные;
- ➔ определяемые пользователем типы данных, такие как `enum`, `typedef` и классы с глобальной областью видимости (о них мы поговорим чуть позже).

Идентификаторы с внешней связью могут вызвать ошибку дублирования определений, если определения скомпилированы в более чем одном файле `.cpp`.

Функции по умолчанию имеют внешнюю связь, что можно изменить с помощью ключевого слова `static` (на внутреннюю связь).

Внимательные читатели могут заметить, что глобальные типы данных имеют внешнюю связь, но их определения не вызывают ошибки линкера при использовании в нескольких файлах. Это связано с тем, что типы, шаблоны и внешние встроенные функции являются исключениями из правила, и это позволяет им быть определенными более чем в одном файле, при условии, что эти определения идентичны. В противном случае, они не были бы так полезны.

Резюмируем

Весь материал, изложенный выше:

Тип	Пример	Область видимости	Продолжительность жизни	Связь	Примечание
Локальная переменная	<code>int x;</code>	Локальная область видимости	Автоматическая продолжительность жизни	Нет связей	
Статическая локальная переменная	<code>static int s_x;</code>	Локальная область видимости	Статическая продолжительность жизни	Нет связей	
Динамическая переменная	<code>int *x = new int;</code>	Локальная область видимости	Динамическая продолжительность жизни	Нет связей	
Параметр функции	<code>void foo(int x)</code>	Локальная область видимости	Автоматическая продолжительность жизни	Нет связей	
Внешняя неконстантная глобальная переменная	<code>int g_x;</code>	Глобальная область видимости	Статическая продолжительность жизни	Внешняя связь	Инициализированная или неинициализированная
Внутренняя неконстантная глобальная переменная	<code>static int g_x;</code>	Глобальная область видимости	Статическая продолжительность жизни	Внутренняя связь	Инициализированная или неинициализированная
Внутренняя константная	<code>const int g_x(1);</code>	Глобальная область видимости	Статическая продолжительность жизни	Внутренняя связь	Должна быть инициализирована

глобальная
переменная

Внешняя
константная
глобальная
переменная

extern
const int
g_x(1);

Глобальная
область
видимости

Статическая
продолжительность
жизни

Внешняя
связь

Должна быть
инициализирована

Предварительные объявления

С помощью предварительного объявления мы можем получить доступ к функции или переменной из другого файла:

Тип	Пример	Примечание
Предварительное объявление функции	<code>void foo(int x);</code>	Только прототип, без тела функции
Предварительное объявление неконстантной глобальной переменной	<code>extern int g_x;</code>	Переменная должна быть инициализирована
Предварительное объявление константной глобальной переменной	<code>extern const int g_x;</code>	Переменная должна быть инициализирована

Оценить статью:



(259 оценок, среднее: 4,90 из 5)



[← Урок №51. Статические переменные](#)

[Урок №53. Пространства имен](#)



Комментариев: 5



1. Михаил:

[29 апреля 2020 в 02:33](#)

Внешняя неконстантная глобальная переменная может быть не инициализированной. Тогда вопрос, как компилятор понимает, что это определение переменной, а не предварительное объявление переменной определенной где-то во внешнем файле.

Например имеем код в начале файла:

```
1 | extern int g_y;
```

Это определение не инициализированной неконстантной глобальной переменной, которую можно будет использовать где-то вне данного файла или это предварительное объявление

(привязка на совести линкера)?

Понятно, что данный вопрос должен решаться на этапе компиляции, но как компилятор это поймет?

[Ответить](#)



1. *Влад:*

[29 июля 2020 в 16:51](#)

Ключевое слово `extern`, при ОПРЕДЕЛЕНИИ неконстантной глобальной переменной `int g_u`, в каком-нибудь внешнем файле, писать НЕ нужно. Как раз-таки когда мы хотим сделать ПРЕДВАРИТЕЛЬНОЕ ОБЪЯВЛЕНИЕ то пишем ключевое слово `extern` (мы как бы указываем компилятору с помощью слова `extern` что переменную `g_u` нужно использовать из другого файла)

[Ответить](#)



2. *Владимир:*

[1 декабря 2019 в 00:35](#)

Кажется это самый серьезный головняк из пройденного. Разнообразие, подобие и отличие всех этих вариантов переменных реальный треш. Вроде не сложно, но запомнить это — просто нечто...

[Ответить](#)



1. *winnie:*

[6 января 2020 в 23:55](#)

Да, последние несколько тем — выносят мозг. Это все, конечно, нужно знать, но, имхо, только многократное применение на практике может помочь это все запомнить и корректно использовать... Я эти темы даже не стараюсь зазубрить и полностью понять, пока просто читаю для общего развития 😊

[Ответить](#)



2. *Артемий:*

[20 февраля 2020 в 11:18](#)

Надо читать по настроению, не заучивать... Сложно, но понятно все.

[Ответить](#)

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](#)
[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 «SameGame» на C++/MFC](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

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