Содержание

[ОБЩАЯ ИНФОРМАЦИЯ 2](#_heading=h.gjdgxs)

[Тема занятия](#_heading=h.30j0zll) 2

[Цели и задачи 2](#_heading=h.1fob9te)

[Ожидаемый результат 2](#_heading=h.3znysh7)

[Структура занятия 3](#_heading=h.tyjcwt)

[ОПИСАНИЕ ЗАНЯТИЯ 4](#_heading=h.3dy6vkm)

# ОБЩАЯ ИНФОРМАЦИЯ

### Тема занятия

### Глобальная область видимости.

**Цели и задачи:**

* Познакомиться с понятием областей видимости
* Научиться работать с глобальными переменными

**По результатам занятия слушатель будет знать:**

* Что такое область видимости
* Параметры функции по умолчанию

**По результатам занятия слушатель будет уметь:**

* Работать с параметрами по умолчанию
* Понимать и разбираться областях видимости
* Глобальные переменные

### Структура занятия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тайминг занятия** | |  |  |  | Таблица 1 |
| № | Этапы | Что делает преподаватель | Что делает Слушатель | Время | Общее время |
| 1 | Приветственное слово преподавателя |  |  | 5 мин. | 5 мин. |
| 2 | Повторение пройденного |  |  | 5 мин. | 40 мин. |
| 3 | Теоретическая часть |  |  | 30 мин. |
| 4 | Вопросы по теоретической части |  |  | 5 мин. |
|  | *Перерыв* |  |  | 15 мин. | 15 мин. |
| 5 | Практическая часть |  |  | 40 мин. | 45 мин. |
| 6 | Рефлексия и вопросы |  |  | 5 мин. |

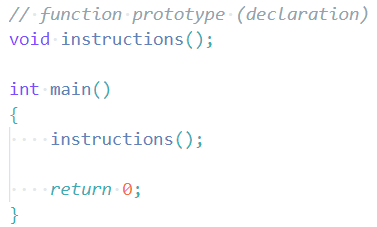
# ОПИСАНИЕ ЗАНЯТИЯ

**Теоретическая часть**

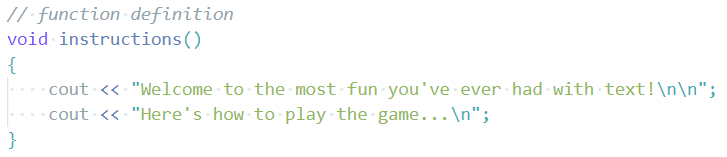
Когда программа достигает определенных размера и уровня сложности, с ней становится неудобно работать в рамках одной функции. К счастью, существуют способы разбивать программу на компактные, более удобоваримые фрагменты кода.

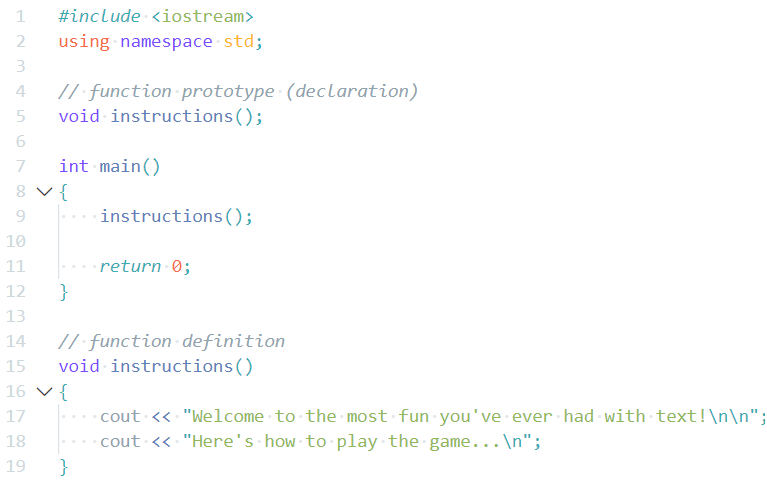
Функция приступает к выполнению задачи, справляется с ней, а затем передает управление вашей программе. Писать новые функции удобно как раз потому, что таким образом вы можете разбивать код на небольшие фрагменты, которыми легко управлять.

Прежде чем вы сможете вызвать написанную вами функцию, ее потребуется **объявить**. Один из способов сделать это - подготовить прототип функции, то есть код, который эту функцию описывает. Чтобы написать прототип функции, нужно указать ее возвращаемое значение, затем написать имя функции, затем - список принимаемых ею параметров, заключенный в круглые скобки. Параметры получают значения, передаваемые в виде аргументов при вызове функции.



**Определить функцию** - значит написать весь код, благодаря которому она будет работать. Определяя функцию, вы указываете ее возвращаемое значение (или void, если функция не возвращает значения), затем пишете имя функции, а после этого - список ее параметров, перечисленных в круглых скобках. Процесс очень напоминает создание прототипа функции (за тем исключением, что в конце такой строки кода не ставится точка с запятой). Так получается заголовок функции. Затем пишется заключенный в фигурные скобки блок кода, в котором перечисляются все инструкции, которые должны выполняться в ходе работы функции. Этот код называется телом функции.



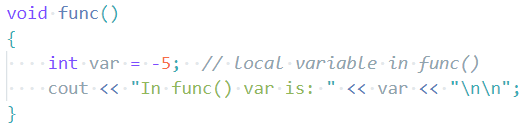
****

Работа по написанию и вызову функций называется абстрагированием. Благодаря абстрагированию можно держать в уме целостную картину происходящего, не вдаваясь в детали. Так, я могу просто использовать функцию instructions (), не задумываясь о том, как именно текст отображается на экране. Мне просто нужно вызвать функцию одной строкой кода, и работа будет выполнена**.**

**Работа с областями видимости**

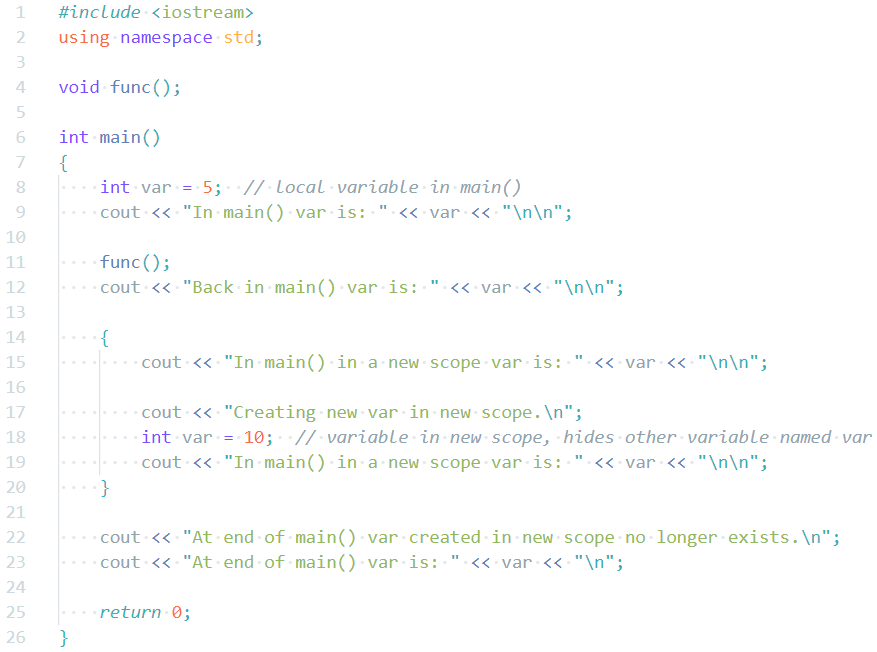
Область видимости переменной определяет, в каких частях вашей программы будет видна эта переменная. Применяя области видимости, можно ограничить доступ к переменным. С помощью областей видимости удобно отграничивать друг от друга различные компоненты вашей программы, например функции.

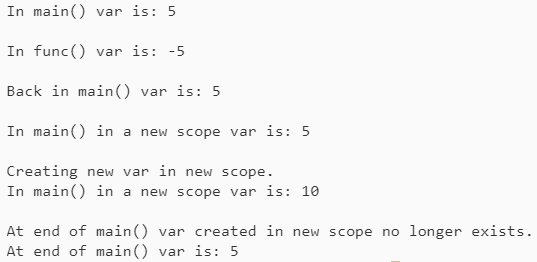
Программа Scoping демонстрирует обращение с областями видимости. В ней мы создадим три одноименные переменные, каждая из которых имеет свою область видимости. Программа отображает значения этих переменных, и вы можете убедиться, что, хотя эти переменные и имеют одинаковые имена, они независимы друг от друга.

****

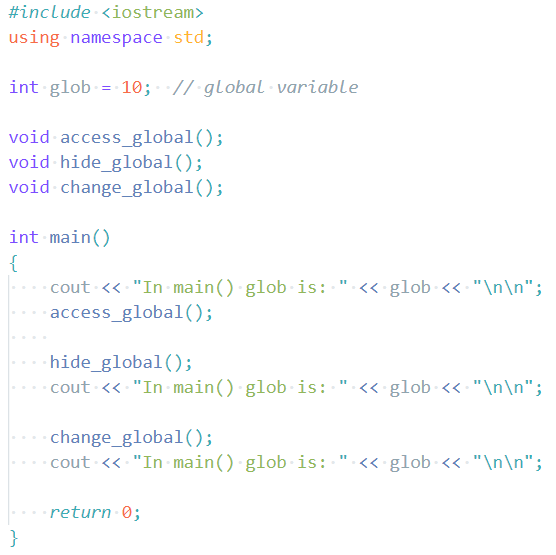
Работа с отдельными областями видимости

Всякий раз, создавая с помощью фигурных скобок блок кода, вы создаете новую область видимости. Переменные, объявляемые внутри этой области, невидимы за ее пределами. Таким образом, переменные, определенные в функции, невидимы за пределами этой функции. Переменные, определяемые внутри функции, называются локальными переменными, то есть они локализованы в своей функции.

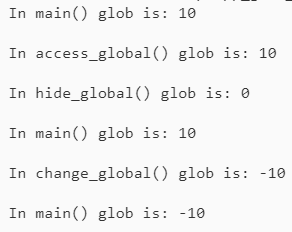
****

****

В программе Global Reach демонстрируется работа с глобальными переменными. Здесь будет показано, как обратиться к глобальной переменной из любой части вашей программы.

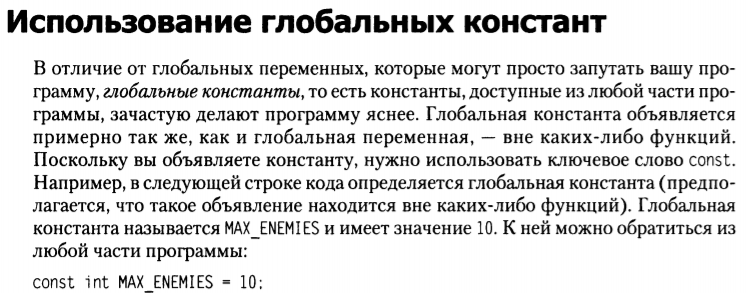
****

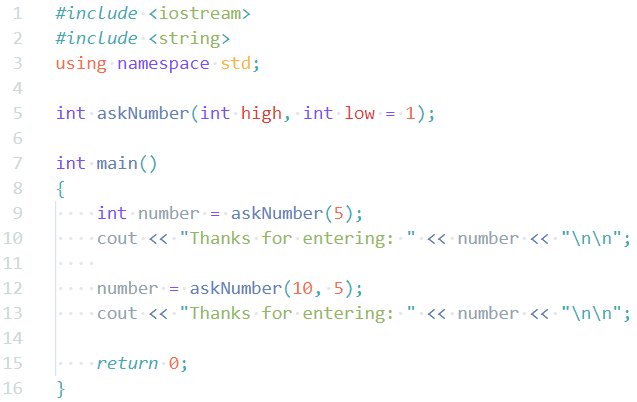
****

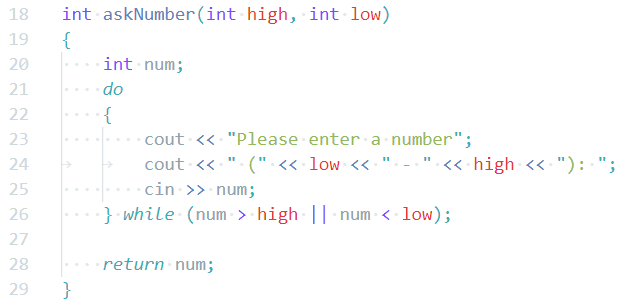
****

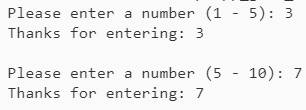
**Минимизация использования глобальных переменных**

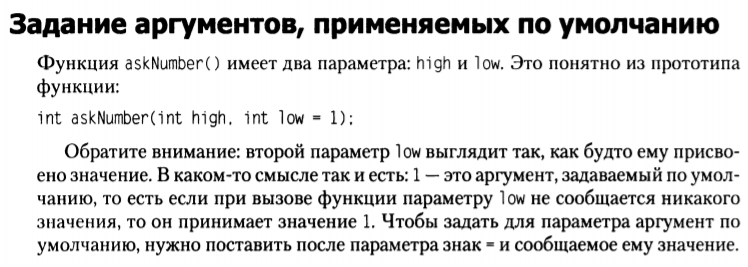
То, что можно что-то сделать, еще не означает, что так действительно стоит поступать. Это важное кредо программистов. Иногда технически операция возможна, но к ней лучше не прибегать. Классический пример такого случая - использование глобальных переменных. Вообще с глобальными переменными программа становится более запутанной, так как зачастую бывает сложно отслеживать все изменяющиеся в ней значения. Поэтому следует максимально ограничить любое использование глобальных переменных.

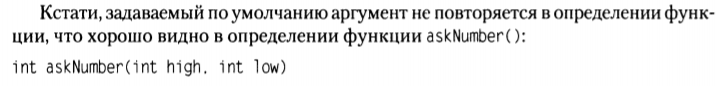
****

****

****

****

****

****

**Практическая часть**

