

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский ядерный университет «МИФИ»

Димитровградский инженерно-технологический институт –

филиал федерального государственного автономного образовательного учреждения высшего образования

«Национальный исследовательский ядерный университет «МИФИ»

(ДИТИ НИЯУ МИФИ)

Специальность 09.02.07 Информационные системы и программирование

МДК.01.01. Разработка программных модулей

РЕФЕРАТ НА ТЕМУ

«Область видимости (контекст) переменных и констант языка C#»

Выполнил студент 3 курса 331 группы

Данилин Алексей Андреевич

Преподаватель ДИТИ НИЯУ МИФИ

_____ А.В. Надеждина

Работа сдана 23.01.2023 г

Оценка _____

Димитровград

СОДЕРЖАНИЕ

1 ОБЛАСТЬ ВИДИМОСТИ ПЕРЕМЕННЫХ C#	3
2 КОНФЛИКТЫ ОБЛАСТЕЙ И ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ	5
3 КОНФЛИКТЫ ОБЛАСТЕЙ ВИДИМОСТИ ПОЛЕЙ И ЛОКАЛЬНЫХ ПЕРЕМЕННЫХ	7
ЗАКЛЮЧЕНИЕ	9
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	10

1 ОБЛАСТЬ ВИДИМОСТИ ПЕРЕМЕННЫХ C#

Локальная область – участок кода, внутри класса или блок, который ограничен фигурными скобками.

Область видимости, или контекст переменной — это часть кода, в пределах которой доступна данная переменная.

Переменная созданная внутри локальной области называется локальной переменной, область ее действия – от открывающей скобки локальной области (‘{’) до ее окончания (закрывающей скобки ‘}’) блока, включая все вложенные локальные области.

Глобальная область видимости — это самая внешняя область из всех. Когда мы «просто объявляем переменную», вне функций, вне модулей, эта переменная попадает в глобальную область видимости.

Переменная уровня класса называется глобальной переменной или полем.

В общем случае область видимости определяется описанными ниже правилами:

- Поле находится в области видимости до тех пор, пока в этой области находится содержащий поле класс.
- Локальная переменная находится в области видимости до тех пор, пока закрывающая фигурная скобка не укажет конец блока операторов или метода, в котором она объявлена.
- Локальная переменная, объявленная в операторах цикла for, while или подобных им, видима в пределах тела цикла.

Правила работы с областями видимости:

1. В коде можно создавать локальные области и, в двух разных локальных областях, хранить одноименные переменные.
2. Если в коде имеются локальные области, то запрещается хранить одноименные переменные за пределами локальных областей. И наоборот, если за пределами локальных областей уже созданы переменные с каким-то именем, то в локальных областях этого уровня запрещается создавать одноименные переменные.

3. В локальных областях можно обращаться к переменным их глобальных областей, но не наоборот.

Приведем пример областей видимости на более простых вещах. Рассмотрим рисунок 1.1.



Рисунок 1.1 – Пример областей видимости

Представим себе две комнаты (Локальная область 1 и Локальная область 2) в каждой из этих комнат находится человек (красный и зелёный соответственно) – ассоциируем их с переменными. Они не могут видеть друг друга, поскольку им мешают стены. Прийти к другому они не могут из-за следующего правила: люди (переменные) не могут покинуть пределы своих комнат (локальных областей). Соответственно красный и зелёный человек взаимодействовать между собой не могут.

Две комнаты между собой соединены через коридор (Глобальную область) в котором также находится человек (синий). Он может через двери увидеть как красного, так и зелёного человека (аналогично, они тоже могут его видеть), также они могут взаимодействовать между собой (ведь никто не запрещает синему человеку посетить комнату зелёного или красного: их локальные области находятся в его локальной области).

2 КОНФЛИКТЫ ОБЛАСТЕЙ И ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ

Использование в больших программах одних и тех же имен переменных в разных частях программы является обычной практикой. Это нормально до тех пор, пока области видимости этих переменных не перекрываются и находятся в совершенно разных частях программы, таким образом, исключая любую неоднозначность. Однако, следует иметь в виду, что локальные переменные с одним и тем же именем не могут быть объявлены дважды в одном и том же контексте, поэтому вы не сможете поступить так, как показано на листинге 2.1.

Листинг 2.1 – Объявление двух переменных в одной области видимости

```
static void Main(string[] args)
{
    int x = 20;
    // какой-то код
    int x = 30;
}
```

Рассмотрим следующий пример кода, представленный на листинге 2.2

Листинг 2.2 – Пример использования переменной в разных контекстах

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 0; i < 10; i++)
            {
                Console.Write(" {0}", i);
            }
        }
    }
}
```

```

    } // здесь i покидает область видимости

    Console.WriteLine();

    // мы можем вновь объявить i
    for (int i = 0; i > -10; i--)
    {
        Console.Write(" {0}", i);
    } // i снова покидает область видимости

    Console.ReadLine();

    var j = i * i; // данный код не выполнится, т.к i не
определена в текущем контексте
    }
}
}

```

Важно отметить, что переменная `i` объявляется в этом коде два раза в пределах одного и того же метода. Это можно делать, поскольку переменные `i` объявляются в двух отдельных циклах, поэтому каждая из них локальна в пределах собственного цикла.

Рассмотрим следующий пример, представленный на листинге 2.3.

Листинг 2.3 – Пример кода.

```

public static int Main()
{
    int j = 20;
    for (int i = 0; i < 10; i++)
    {
        int j = 30; // Так делать нельзя - j все еще в контексте
        Console.WriteLine (j + i) ;
    }
    return 0;
}

```

При попытке скомпилировать этот код, на экран будет выведено следующее сообщение об ошибке:

ScopeTest.cs (12,15) : error CS0136: A local variable named 'j' cannot be declared in this scope because it would give a different meaning to 'j', which is already used in a 'parent or current' scope to denote something else

Дело в том, что переменная `j`, которая определена перед началом цикла `for`, внутри цикла все еще находится в области видимости и не может из нее выйти до завершения метода `Main()`. Хотя вторая переменная `j` (недопустимая) объявлена в контексте цикла, этот контекст вложен в контекст метода `Main()`. Компилятор не может различить эти две переменных, поэтому не допустит объявления второй из них.

3 КОНФЛИКТЫ ОБЛАСТЕЙ ВИДИМОСТИ ПОЛЕЙ И ЛОКАЛЬНЫХ ПЕРЕМЕННЫХ

В некоторых случаях два идентификатора с одинаковыми именами (хотя и не совпадающими полностью уточненными именами) и одинаковой областью видимости можно различить, и тогда компилятор допускает объявление второй переменной. Причина в том, что `C#` делает принципиальное различие между переменными, объявленными на уровне типа (полями) и переменными, объявленными в методах (локальными). Рассмотрим следующий фрагмент кода, представленный на листинге 3.4.

Листинг 3.4 – Пример кода.

```
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int j = 20;

        public static void Main()
        {
            int j = 30;

            Console.WriteLine(j);
        }
    }
}
```

```
        return;  
    }  
}
```

Этот код компилируется, несмотря на то, что здесь в контексте метода `Main()` присутствуют две переменных с именем `j`: переменная `j`, определенная на уровне класса и существующая до тех пор, пока не будет уничтожен класс (когда завершится метод `Main()`, а вместе с ним и программа), и переменная `j`, определенная внутри `Main()`. В данном случае новая переменная с именем `j`, объявленная в методе `Main()`, скрывает переменную уровня класса с тем же именем. Поэтому, когда вы запустите этот код, на дисплее будет отображено число 30.

ЗАКЛЮЧЕНИЕ

В языке C# локальные переменные широко используются именно из-за *аппарата* разграничения области видимости. Понимание контекста и областей видимости переменных поможет предотвратить ошибки, которые могут быть вызваны использованием переменных вне их контекста, что значительно сократит время работы над программой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сайт о программировании: сайт. – 2012. – URL:
<https://metanit.com/sharp/tutorial/2.18.php> (дата обращения: 11.03.2023)
2. Сайт о программировании: сайт. – 2019. – URL:
<https://forum.itvdn.com/t/urok-3-oblasti-vidimosti/3030> (дата обращения: 12.03.2023)
3. Сайт о программировании: сайт. – 2011. – URL:
https://professorweb.ru/my/csharp/charp_theory/level3/3_5.php (дата обращения: 11.03.2023)