Окна уязвимости

№ урока: 20 **Курс:** Процедурное программирование на языке C#

Средства обучения: Visual Studio 2019 Community Edition

Обзор, цель и назначение урока

На данном уроке вы рассмотрите очень важное для чистоты и качества кода понятие – окна уязвимости. Понимание темы данного урока позволит вам совершать меньше ошибок при написании кода, а также писать код в красивом и понятном стиле.

Изучив материал данного занятия, учащийся сможет:

- Понимать основы понятия «окно уязвимости».
- Понимать предназначение безымянных блоков.
- Понимать понятие «время жизни переменных».
- Понимать принципы построения кода в процедурном стиле, обеспечивающие правильных подход к безопасному использованию переменных.

Содержание урока

1. Окна уязвимости
2. Окна уязвимости на практике
3. Время жизни переменной
4. Значения переменных по умолчанию
5. Структурное программирование

Резюме

- **Блок** или составной оператор — конструкция языка программирования, состоящая из нескольких команд (операторов) языка программирования, но участвующая в программе в качестве единого оператора.
  Блоки создаются с помощью добавления открывающей и закрывающей операторных скобок вокруг участка программного кода. Эти блоки предназначены для структурирования кода и являются основой использования структурной парадигмы в C#.
  Блок еще называют блоком кода, блоком команд, или блоком инструкций.
  Блоки — это основные «кирпичики», которые применяются, для построения наших программ. Блок считается одним оператором, несмотря на то что он занимает много строк.
  Именно блоки являются основными элементами структурного подхода в программировании, потому что в основе структурного подхода в программировании, лежит именно, идея использования *вложенности блоков*.

- **Область видимости переменной** – это тело блока, в котором переменная может быть использована, или, другими словами, мы можем к ней обратиться.

- **Структурный подход** (а именно – *идея использования блоков и их вложенности*) лежит не только в основе процедурного программирования, он также лежит в основе объектно-ориентированного и функционального программирования.

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Термин «**структурное программирование**» был введен в исторической статье «**Structured Programming**», представленной Эдсгером Дейкстрой на конференции НАТО по разработке Программного Обеспечения в 1969 году.

- Блоки, у которых нет имени, имеют только три предназначения:
  - Первое - ограничивать область видимости переменных.
  - Второе - ограничивать время жизни переменных.
  - Третье – группировать, схожий по смыслу выполняемой деятельности, программный код (то есть алгоритм).

- **Время жизни переменной** – это участок программы, в котором, можно воспользоваться этой переменной. По правилам программирования - жизнь переменной должна начинаться при ее создании, и жизнь переменной должна заканчиваться при последнем обращении к этой переменной.

- Как показывает практика, программисты редко контролируют время жизни переменных. В большинстве программ, все переменные остаются жить «вечно» после последнего обращения к ним, что не мешает в дальнейшем, по ошибке, этими (вовремя не умершими) переменными воспользоваться.

- Время жизни переменной регулируется при помощи локальных областей видимости (то есть блоков). В правильно написанном коде переменная должна жить (то есть быть доступной) только в своем блоке и умирать в конце блока.

- **Окно уязвимости** в программировании — это участок кода (часть текста программы), который располагается между строкой, на которой создана переменная, и строкой, на которой эта переменная используется.

- Величина окна уязвимости, или размер окна уязвимости, называется – **интервалом.**

- Чем больше интервал окна уязвимости, тем выше опасность, что в этом интервале может быть добавлен некий новый код, который изменяет или искажает значение этой переменной.

- Чтобы убрать окно уязвимости, нужно просто создание переменной переместить ближе к месту её использования.

- Нельзя допускать большого количества окон уязвимости, а также нельзя допускать больших интервалов в окнах уязвимости. Относительно небольшое количество таких окон в коде с небольшими интервалами – это неизбежное зло структуры нашей программы.

- Множественное объявление даёт эффект того, что мы должны воспринимать, весь этот (возможно многострочный) набор неструктурированных переменных, как одну строку.

- **Принцип близости** - инициализируйте каждую переменную там, где она используется в первый раз. Лучше инициализировать каждую переменную как можно ближе к месту первого обращения к ней. В идеальном случае сразу объявляйте и определяйте каждую переменную непосредственно перед первым обращением к ней.

- Практика инициализации каждой переменной при ее объявлении полезна для языка C++, но **крайне вредна, даже губительна, для языка C#**

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- **Значение по умолчанию** — это такое значение, которое автоматически, присваивается переменной, если вы, не присвоили это значение самостоятельно. В C#, автоматически присваиваются значения по умолчанию только переменным, которые мы создаем в блоках-классах. Такие переменные называются полями. Термин «значение по умолчанию» для переменной, это термин, больше из объектно-ориентированного программирования. Локальные переменные значений по умолчанию иметь не могут.

- Для всех числовых типов, как для целочисленных, так и для вещественных, значение по умолчанию – это ноль. Для булевых переменных – значение по умолчанию – это false. Для переменных типа стринг – значение по умолчанию – это «null».

Закрепление материала

- Что такое блок?
- Что такое область видимости переменной?
- Как можно убрать или уменьшить окно уязвимости переменной?
- В чем состоит принцип близости?
- Что такое значение по умолчанию для переменной? Каковы значения по умолчанию для известных вам типов пременных?

Самостоятельная деятельность учащегося

- Задание 1

Ознакомьтесь с дополнительными материалами к уроку.

- Задание 2

Перепишите данный код в свой проект:

```
static void Main(string[] args)
{
    int a = 10;
    int b = 20;
    int c = 30;

    int sum = a + b + c;

    int x = 50;
    int y = 150;
    int z = 300;

    int dif = z - y - x;

    double quotient = dif / sum;

    Console.WriteLine($"Значение частного равно {quotient}");
}
```

Максимально уменьшите количество окон уязвимостей для этого кода и их интервалы.

Рекомендуемые ресурсы

https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/default-values

https://ru.wikipedia.org/wiki/Структурное_программирование

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com