Множественное ветвление

№ урока:	25	Курс:	Процедурное программирование на языке C#

Средства обучения: Visual Studio 2019 Community Edition

Обзор, цель и назначение урока

Прохождение данного урока позволит вам понять важные особенности и нюансы использования условных конструкций в коде на практике.

Изучив материал данного занятия, учащийся сможет:

- Понимать возможность и смысл множественного ветвления в коде.
- Понимать смысл применения пустого оператора.
- Уметь работать с вложенными условными конструкциями.

Содержание урока

1. Условные конструкции и границы блоков кода
2. Сложность условных конструкций
3. Множественное ветвление условных конструкций на практике

Резюме

- Чтобы выровнять все конструкции в коде по своим местам, мы можем воспользоваться комбинацией клавиш – Ctrl + K + D.

- Условный оператор **if** реализует выполнение определённых команд при условии, что используемое логическое выражение в условии, принимает значение **true**.

- Если использовалась конструкция **if-else,** и результатом условия было значение **true**, то выполнится только тело оператора **if**, а тело блока **else** останется не выполненным.

- После выполнения оператора **if** управление передается следующему оператору.

- Оператор, выполняемый после проверки условия, может быть любого типа, включая другой оператор **if**, вложенный в оригинальный оператор **if**. Во вложенных операторах **if** предложение **else** принадлежит к последнему оператору **if**, у которого нет соответствующего **else**.

- Если тело блока **if** или **else** состоит из одного выражения, то операторные скобки можно опустить.

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

- Конструкция ветвления **if** – **else** может реализовывать разделение вариантов выполнения кода не только на два маршрута, но и на большее количество маршрутов (ветвей). Для этого используется оператор, состоящий сразу из двух ключевых слов «**else if**» и области условия, на которое выполняется проверка.

- Блок, обозначенный **else if**, должен располагаться между блоками **if** и **else.** Таких блоков **else if** можно добавлять больше одного.

- Блок, которому предшествует эта комбинация ключевых слов **else if**, стоящих вместе, открывает еще одну ветвь маршрута выполнения кода и этот блок будет выполняться, если условие после **else if** вернет **true** и только если не выполнился вышестоящий блок **if**.

- В условной конструкции, состоящей из многих блоков, выполниться может только один блок: или блок, обозначенный if, или один из блоков, обозначенных else if, или блок else. Если последний блок **else** не указан – может возникнуть ситуация, когда не выполнится ни один блок условной конструкции. Желательно всегда явно указывать последний блок **else**, даже если никакая логика в этом варианте не предусмотрена.

- **Пустой оператор** — это оператор, состоящий только из точки с запятой. Такой оператор может появиться в любом месте программы, где по правилам синтаксиса требуется оператор. Выполнение пустого оператора не меняет состояния программы, т.е не выполняет никакого действия.

- «Это выражение (имеется ввиду, одиночно стоящая, точка с запятой) не делает ничего, кроме бесспорного подтверждения того факта, что никакие действия предприниматься не должны. Это похоже на пометку пустых страниц документа, фразами: «Эта страница намеренно оставлена пустой». На самом деле страница не совсем пустая, но вы знаете, что ничего другого на ней быть не должно.» С. Макконнел. Т.е. **смысл применения пустого оператора** – явно указать, что по задумке разработчика в этом конкретном месте программы не нужно выполнять никаких действий.

- Стив Макконнелл говорит: Пустые операторы, не являются широко распространенными, поэтому сделайте их очевидными. Один из способов (сделать их очевидными) — это, выделить точке с запятой, представляющей собой пустой оператор, отдельную строку.

- Телефонные номера, почтовые индексы и прочие числовые сущности, которые не имеет смыла складывать друг с другом, рекомендуется представлять в виде строк.

Закрепление материала

- Что такое пустой оператор и для чего он используется?
- Как можно составить конструкцию, которая для пяти разных вариантов значения переменной выполнит пять разных вариантов вычислений?

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

- Стоит ли использовать операторные скобки для явного обозначения блоков условной конструкции? Если да - то зачем, если нет – то почему?

Самостоятельная деятельность учащегося

- Задание 1

Ознакомьтесь с дополнительными материалами к уроку.

- Задание 2

Напишите программу, в которой примите от пользователя значения двух длин сторон прямоугольника. Далее пользователь вводит строковую команду «площадь» или «периметр». Если пользователь ввел «площадь» - вывести на экран значение площади фигуры. Если пользователь ввел «периметр» - вывести на экран значение периметра. Если пользователь ввел какую-либо другую строку – выведите «Неверная команда!».
Если прямоугольник является квадратом – дополнительно выведите на экран фразу: «данный прямоугольник – квадрат».

- Задание 3

Пользователь вводит 4 числа. Найти наибольшее четное. Если такого нет – вывести «Not found». Если есть – вывести его. (Решите задачу с использованием условных конструкций)

Рекомендуемые ресурсы

https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/if-else

https://ru.wikipedia.org/wiki/Ветвление_(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