ФГАОУ ВО «КФУ им. В.И. Вернадского»
Физико-технический институт (структурное подразделение)

Кафедра компьютерной инженерии и моделирования

конспект (резюме) лекции №3

«Использование конструкций C#»

по дисциплине «Объектно-ориентированное программирование»

Выполнил:
студент 2 курса группы
ПИ-б-о-231(2)
Покидько Максим Сергеевич

Проверил:
Заведующий кафедрой
компьютерной инженерии и
моделирования Милюков В. В.

Симферополь, 2024

Основные концепции языка C#

---

1. Объявление переменных и присвоение значений

Для работы с переменными необходимо:

- **Объявить переменную**, указав её тип.
- **Присвоить значение** переменной.

Локальное поле видимости

- Поле блока
- Поле процедуры
- Поле класса
- Поле пространства имён

Использование суффиксов для числовых типов

Суффиксы используются для указания типов чисел:

- u — для uint
- l — для long
- ul — для ulong
- f — для float
- d — для double
- m — для decimal

---

2. Неявная типизация

Используется ключевое слово var, которое позволяет компилятору автоматически определить тип переменной на основе присваиваемого значения:

```
var number = 5; // int
```

---

3. Преобразование типов в C#

Неявные преобразования

Происходят автоматически, без указания специального синтаксиса, если преобразование безопасно. Примеры:

- Преобразование от меньшего к большему типу (например, int к long).
- Преобразование от производного класса к базовому.

Примеры неявных преобразований:

- sbyte -> short, int, long, float, double, decimal
- int -> long, float, double, decimal

Явные преобразования

Требуют явного указания типа, так как могут привести к потере данных. Пример явного преобразования:

double num = (double)25;

Преобразования с использованием классов

Класс System.Convert позволяет выполнять расширяющие и сужающие преобразования между типами данных.

Пользовательские преобразования

Ключевые слова implicit и explicit позволяют создавать собственные неявные и явные преобразования типов.

---

4. Операторы приведения типов

Оператор is

Используется для проверки, можно ли преобразовать объект к указанному типу:

if (obj is int) { // код }

Оператор as

Позволяет попытаться выполнить преобразование. В случае неудачи возвращает null:

MyClass obj = someObject as MyClass;

5. Обработка исключений при преобразованиях

Исключение OverflowException

Возникает при выходе значения за пределы допустимого диапазона для типа данных.

Безопасное приведение типов

Операторы is и as помогают избежать исключения InvalidCastException, проверяя возможность преобразования.

---

6. Работа с null и nullable-типы

Чтобы переменная могла принимать значение null, используйте символ ?:

int? nullableValue = null;

Оператор ??

Позволяет задать значение по умолчанию, если переменная равна null:

int? x = null; int y = x ?? -1; // y будет равно -1

Оператор ?.

Позволяет обращаться к вложенным свойствам объекта, проверяя его на null:

var length = myString?.Length;

---

7. Операторы и операнды

Операторы выполняют действия над операндами (переменные или значения). Операции могут быть:

- **Унарными** — операция над одним операндом (например, !).
- **Бинарными** — операция над двумя операндами (например, +, -).
- **Тернарными** — операция над тремя операндами (например, ?:).

Примеры операторов

- Арифметические: +, -, *, /, %
- Логические: &&, ||, !
- Поразрядные: &, |, ^, ~, <<, >>
- Операторы сравнения: ==, !=, <, >, <=, >=
- Условные операторы: ?:

---

8. Конструкции управления

Условные выражения

- Оператор if проверяет условие и выполняет код, если условие истинно.
- Оператор switch выбирает одну из нескольких ветвей кода на основе значения.

Циклы

Основные конструкции циклов в C#:

- for
- while
- do-while
- foreach

---

9. Статический импорт

С помощью ключевого слова using static можно подключить статические методы, свойства и константы класса без указания названия класса:

using static System.Math; double result = Sqrt(25); // вместо Math.Sqrt(25)
