

Домашняя работа

Тема: “Переменные”

Вопрос 1

Что выведет следующий код:

```
1  string name = "Tom";  
2  Console.WriteLine (Name);
```

Ответ: Tom

Вопрос 2

Что выведет на консоль следующий код:

```
1  string person = "Tom";  
2  person = "Sam";  
3  Console.WriteLine (person);
```

Варианты ответов

- Tom
- Sam
- person
- Программа завершит выполнение с ошибкой

Ответ: Sam

Вопрос 3

Какие из следующих вариантов представляют корректное определение переменных:

```
string person = "Tom";
```

```
person = "Tom";
```

```
string person;
```

```
string "Tom";
```

Ответ: `string person`;

Вопрос 4

Какие три основных компонента имеет переменная в языке C#?

- класс, имя, метод
- тип, размер, область видимости
- имя, индекс, значение
- Тип, имя, значение

Ответ: Тип, имя, значение

Вопрос 5

В чём заключается различие между определением переменной и её инициализацией в C#?

- определение создаёт новую переменную в памяти, а инициализация её удаляет.
- определение задаёт начальное значение, а инициализация устанавливает тип переменной.
- Определение устанавливает тип и имя переменной, а инициализация задаёт начальное значение.
- определение и инициализация — это одно и то же действие.

Ответ: Определение устанавливает тип и имя переменной, а инициализация задаёт начальное значение.

Вопрос 6

Почему важно учитывать регистрозависимость при работе с переменными в C#? Приведите пример.

- Регистр важен для типов данных, а не для имён переменных.
- C# регистрозависимый язык, поэтому `name` и `Name` — разные переменные.
- В C# регистр не имеет значения для имён переменных.
- Имена переменных в C# должны быть записаны только строчными буквами.

Ответ: C# регистрозависимый язык, поэтому `name` и `Name` — разные переменные

Вопрос 7

В чём состоит ключевое отличие константы от переменной в C# и как это отражается на их использовании в программе?

- Значение переменной фиксируется при определении и не может быть изменено.
- Константа может быть изменена в процессе работы программы, как и переменная.
- Переменные и константы в C# ничем не отличаются друг от друга.
- Константа инициализируется при определении и её значение нельзя изменить, в отличие от переменной.

Ответ: Константа инициализируется при определении и её значение нельзя изменить, в отличие от переменной.

Литералы

Вопрос 1

Какие виды литералов существуют и чем они отличаются друг от друга?

- Логические, целочисленные, вещественные, символьные, строковые и null.
- целые, дробные, текстовые, булевы и специальные.
- положительные, отрицательные, дробные, символьные и строковые.
- числовые, буквенные, логические, графические и пустые.

Ответ: Логические, целочисленные, вещественные, символьные, строковые и null.

Вопрос 2

В каких формах могут быть представлены вещественные литералы и как они интерпретируются?

- строковые литералы в двойных кавычках
- Вещественные числа с фиксированной запятой и в экспоненциальной форме MEp
- целые числа в десятичной, шестнадцатеричной и двоичной форме
- символьные литералы в одинарных кавычках

Ответ: Вещественные числа с фиксированной запятой и в экспоненциальной форме MEp

Базовые типы данных

Вопрос 1

Какие из нижеперечисленных НЕ являются встроенными типами языка C#?

- uint
- sbyte
- real
- int128
- object
- float64

Ответ: real, int128, float64

Вопрос 2

Какой тип данных языка C# будет представлять следующая переменная?

```
1 bool enabled = true;
```

Ответ: bool

Вопрос 3

Какой тип данных языка C# будет представлять следующая переменная?

```
1 var weight = 84.45f;
```

Ответ: float

Вопрос 4

Сколько байт занимает значение типа **uint**?

Ответ: 4 байт

Вопрос 5

Какие из следующих вариантов представляют корректное определение переменных:

```
1 string person = "Tom";
```

```
2 var person = "Tom";
```

3 `var person;`
4 `string person;`

Ответ: 1,2,4

Правильные варианты:

Вопрос 6

Какой системный тип соответствует базовому типу данных **int** в языке C# и сколько байт он занимает?

1. `System.Int32`, 4 байта
2. `System.Single`, 4 байта
3. `System.UInt32`, 8 байт
4. `System.Int16`, 2 байта

Ответ: `System.Int32`, 4 байта

Вопрос 7

Какие суффиксы используются в C# для явного указания типа данных `float` и `decimal` при присвоении значений?

1. `S/s` — для `float`, `D/d` — для `decimal`
2. `X/x` — для `float`, `Y/y` — для `decimal`
3. `F/f` — для `float`, `M/m` — для `decimal`
4. `L/l` — для `float`, `U/u` — для `decimal`

Ответ: `F/f` – для `float`, `M/m` – для `decimal`

Вопрос 8

Чем отличается объявление переменной с использованием `var` от явного указания типа данных, например, `int`?

1. `var` и `int` — это синонимы для объявления целочисленных переменных.
2. При использовании `var` тип переменной определяется автоматически на основе присвоенного значения.
3. `var` используется для объявления переменных с типом `string`.
4. `var` позволяет объявлять переменные без указания типа и инициализации.

Ответ: При использовании `var` тип переменной определяется автоматически на основе присвоенного значения.

Консольный ввод-вывод

Вопрос 1

Как вывести на консоль значения нескольких переменных в одной строке с помощью интерполяции?

- `Console.WriteLine("{name} {age} {height}");`
- `Console.WriteLine("Имя: " name " Возраст: " age " Рост: " height "м");`
- `Console.WriteLine("Имя: {name} Возраст: {age} Рост: {height}м");`
- `Console.Write(name, age, height);`

Ответ: `Console.WriteLine("Имя: {name} Возраст: {age} Рост: {height}м");`

Вопрос 2

Что такое плейсхолдеры в контексте вывода данных на консоль и как они используются?

- Плейсхолдеры — это числа в фигурных скобках, которые заменяются значениями при выводе на консоль
- плейсхолдеры используются для создания пустых строк в выводе
- плейсхолдеры — это имена переменных, которые выводятся на консоль без изменений
- плейсхолдеры — это специальные символы для форматирования строк

Ответ: Плейсхолдеры — это числа в фигурных скобках, которые заменяются значениями при выводе на консоль

Вопрос 3

В чём отличие метода `Console.Write()` от `Console.WriteLine()`?

1. `Console.Write()` используется для ввода данных, а `Console.WriteLine()` — для вывода.
2. `Console.Write()` выводит информацию в виде таблицы, а `Console.WriteLine()` — в виде списка.
3. `Console.Write()` не добавляет переход на следующую строку, а `Console.WriteLine()` добавляет.
4. `Console.Write()` может выводить только числа, а `Console.WriteLine()` — любые данные.

Ответ: 3.

Вопрос 4

Каким методом можно получить ввод с консоли и в каком виде он возвращается?

1. методом `Console.WriteLine()`, возвращается в виде числа.
2. методом `Console.Write()`, возвращается в виде массива.
3. методом `Convert.ToInt()`, возвращается в виде строки.
4. Методом `Console.ReadLine()`, возвращается в виде строки.

Ответ: `Console.ReadLine()`

Вопрос 5

Какие методы предоставляет платформа .NET для преобразования строковых значений в числовые типы данных?

1. `Convert.ToString()`, `Convert.ToInt()`, `Convert.ToChar()`
2. `Parse.ToInt()`, `Parse.ToFloat()`, `Parse.ToNumber()`
3. `Convert.ToInt()`, `Convert.ToDouble()`, `Convert.ToDecimal()`
4. `Console.WriteLine()`, `Console.Write()`, `Console.ReadLine()`

Ответ: `Convert.ToInt()`, `Convert.ToDouble()`, `Convert.ToDecimal()`

Операции

Вопрос 1

Есть следующий код:

```
1  int n1 = 2;
2  int n2 = 5;
3  int result = n2 * 3 + 20 / 2 * n1--;
```

Используя приоритеты операций, разложите выражение `int result = n2 * 3 + 20 / 2 * n1--` по шагам.

Ответ: 1. `n1--`
2. `n2 * 3`
3. `20 / 2`
4. `(20 / 2) * (n1--)`
5. `(n2*3) + ((20 / 2) * (n1--))`

Вопрос 2

Есть следующий код:

```
1  int num1 = 4;
2  int num2 = 5;
3  int num3 = 15;
4  int num4 = 10;
5  int num5 = 5;
6  int result = 12;
7
8  result += num1 * num2 + num3 % num4 / num5;
```

Используя приоритеты операций, разложите выражение `result += num1 * num2 + num3 % num4 / num5` по шагам.

Ответ: 1. Num1 * num2
2. num3 % num4
3. num3 % num4 / num5
4. (Num1 * num2) + (num3 % num4 / num5)

Вопрос 3

Чему будет равна переменная z после выполнения следующего кода и почему?

```
1  int x = 8;
2  int y = 9;
3  int z = x++ + ++y;
```

Ответ: 18

Практическое задание:

Задача 1

Ваша задача — создать простой калькулятор, который сможет выполнять базовые арифметические операции: сложение, вычитание, умножение и деление, остаток от деления, инкремент, декремент. Калькулятор должен предоставлять пользователю возможность вводить числа и вывод всех математических действий.

Условия выполнения:

1. Ввод данных:

- Пользователь должен вводить два числа (например, целые или дробные).

2. Операции:

- Реализуйте следующие арифметические операции:

- Сложение (+)
- Вычитание (-)
- Умножение (*)
- Деление (/)
- Остаток от деления (%)
- Инкремент (++)
- Декремент (--)

3. Вывод результата:

- После выполнения операции калькулятор должен выводить результат на экран.

```
using System;
```

```
class Program
{
    static void Main()
    {
```

```
Console.WriteLine("===КАЛЬКУЛЯТОР ===");

// Ввод первого числа
Console.Write("Введите первое число: ");
double num1 = Convert.ToDouble(Console.ReadLine());

// Ввод второго числа
Console.Write("Введите второе число: ");
double num2 = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("\n--- Результаты ---");

// Сложение
double sum = num1 + num2;
Console.WriteLine($"{num1} + {num2} = {sum}");

// Вычитание
double difference = num1 - num2;
Console.WriteLine($"{num1} - {num2} = {difference}");

// Умножение
double product = num1 * num2;
Console.WriteLine($"{num1} * {num2} = {product}");

// Деление
if (num2 != 0)
{
    double quotient = num1 / num2;
    Console.WriteLine($"{num1} / {num2} = {quotient:F4}");
}
else
{
    Console.WriteLine($"{num1} / {num2} = Ошибка: деление на ноль!");
}

// Остаток от деления
if (num2 != 0)
{
    double remainder = num1 % num2;
    Console.WriteLine($"{num1} % {num2} = {remainder}");
}
else
{
    Console.WriteLine($"{num1} % {num2} = Ошибка: деление на ноль!");
}

// Инкремент
double num1Inc = num1;
```

```
double num2Inc = num2;  
Console.WriteLine($"Инкремент {num1}++ = {++num1Inc} (префиксная форма)");  
Console.WriteLine($"Инкремент {num2}++ = {num2Inc++} (постфиксная форма,  
значение после: {num2Inc})");
```

```
// Декремент  
double num1Dec = num1;  
double num2Dec = num2;  
Console.WriteLine($"Декремент {num1}-- = {--num1Dec} (префиксная форма)");  
Console.WriteLine($"Декремент {num2}-- = {num2Dec--} (постфиксная форма,  
значение после: {num2Dec})");  
}  
}
```