Домашняя работа

Переменные

Вопрос 1

Что выведет следующий код:

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

Ответ Name не определена (C# регистрозависим: name ≠ Name)

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 = "Tom"; 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 задаёт тип и имя переменной, а инициализация задаёт её 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

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?

```
bool enabled = true;
```

Ответ bool

Вопрос 3

Какой тип данных языка C# будет представлять следующая переменная?

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

Ответ float (System.Single)

Вопрос 4

Сколько байт занимает значение типа **uint**?

Ответ 4 байта

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 string person = "Tom"; var person = "Tom"; string person;

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 decima

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 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 Console.Write() не добавляет переход на новую строку, Console.WriteLine() добавляет перевод строки

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(), возвращает строку

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

```
Output
Initial value: n1 = 2, n2 = 5
Step 1: n2 * 3 = 5 * 3 = 15
Step 2: 20 / 2 = 10
Step 3: 10 * 2 = 20 ('n1' after the calculation = 1)
Step 4: 15 + 20 = 35

Result: result = 35, n1 = 1

=== Code Execution Successful ===
```

Код

```
1  using System;
2  class Program
3  {
4      static void Main()
5      {
6          int n1 = 2;
7          int n2 = 5;
8          Console.WriteLine($"Initial value: n1 = {n1}, n2 = {n2}");
9          int step1 = n2 * 3;
10         Console.WriteLine($"Step 1: n2 * 3 = {n2} * 3 = {step1}");
11         int step2 = 20 / 2;
12         Console.WriteLine($"Step 2: 20 / 2 = {step2}");
13         int step3 = step2 * n1--;
14         Console.WriteLine($"Step 3: {step2} * 2 = {step3} ('n1' after the
               calculation = {n1})");
15         int result = step1 + step3;
16         Console.WriteLine($"Step 4: {step1} + {step3} = {result}");
17         Console.WriteLine($"\nResult: result = {result}, n1 = {n1}");
18     }
19 }
```

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

```
Initial values:
num1 = 4, num2 = 5, num3 = 15, num4 = 10, num5 = 5, result = 12

Step 1 (multiplication): num1 * num2 = 4 * 5 = 20
Step 2 (remainder): num3 % num4 = 15 % 10 = 5
Step 3 (division): 5 / 5 = 1  (integer division)
Step 4 (plus): 20 + 1 = 21
Step 5 (compound assignment): result (before) = 12; result += 21 => result (after)
    = 33

Result: result = 33
```

Код

```
1  using System;
2  class Program
3  {
4      static void Main()
5      {
6          int num1 = 4;
7          int num2 = 5;
8          int num3 = 15;
9          int num4 = 10;
10         int num5 = 5;
11         int result = 12;
12         Console.WriteLine("Initial values:");
13         Console.WriteLine($"num1 = {num1}, num2 = {num2}, num3 = {num3}, num4 = {num4}, num5 = {num5},
               result = {result}");
14         Console.WriteLine();
15         int step1 = num1 * num2; // 4 * 5 = 20
16         Console.WriteLine($"Step 1 (multiplication): num1 * num2 = {num1} * {num2} = {step1}");
17         int step2 = num3 % num4; // 15 % 10 = 5
18         Console.WriteLine($"Step 2 (remainder): num3 % num4 = {num3} % {num4} = {step2}");
19         int step3 = step2 / num5; // 5 / 5 = 1
20         Console.WriteLine($"Step 3 (division): {step2} / {num5} = {step3}  (integer division)");
21         int step4 = step1 + step3; // 20 + 1 = 21
22         Console.WriteLine($"Step 4 (plus): {step1} + {step3} = {step4}");
23         int before = result;
24         result += step4; // result = result + step4
25         Console.WriteLine($"Step 5 (compound assignment): result (before) = {before}; result += {step4} =>
               result (after) = {result}");
26         Console.WriteLine();
27         Console.WriteLine($"Result: result = {result}");
28     }
29 }
```

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

```
Initial values: x = 8, y = 9

After calculating the expression: z = x++ + ++y
value x++ (post-increment): is used 8, then 'x' increases to 9
value ++y (preincrement): first, 'y' increases to 10, and is used 10
in this way: z = 8 + 10 = 18

Result:
x = 9
y = 10
z = 18

=== Code Execution Successful ===
```

Код

```
1  using System;
2
3  class Program
4  {
5      static void Main()
6      {
7          int x = 8;
8          int y = 9;
9
10         Console.WriteLine($"Initial values: x = {x}, y = {y}");
11
12         int z = x++ + ++y;
13
14         Console.WriteLine($"\nAfter calculating the expression: z = x++ + ++y");
15         Console.WriteLine($"value x++ (post-increment): is used {8}, then 'x' increases to
               {x}");
16         Console.WriteLine($"value ++y (preincrement): first, 'y' increases to {y}, and is
               used {y}");
17         Console.WriteLine($"in this way: z = 8 + 10 = {z}");
18
19         Console.WriteLine($"\nResult:");
20         Console.WriteLine($"x = {x}");
21         Console.WriteLine($"y = {y}");
22         Console.WriteLine($"z = {z}");
23     }
24 }
```

Практическое задание:

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.

Скриншот результата