**Лабораторная работа № 1**

**Задание 1.** Для создания нового проекта выполните следующие действия (см. Приложение 1):

1. Создайте новый проект: File ► New Project, либо Сreate Project в

окне Start Page

2. В окне New project в левой части выберите Visual C# Projects, в пра-

вой – пункт Console Application

3. В поле Name введите имя проекта, в поле Location – место его сохранения на диске

4. Ознакомьтесь с основными окнами среды.

5. Рассмотрите каждую строку заготовки программы.

6. Наберите приведенный пример программы (Листинг 1).Вставьте свои значения соответствующих типов в пропущенных местах операторов.

using System;

namespace matemat1

{

class Program

{

public static void Main(string[] args)

{

// Объявление переменной x и присвоение ей значения 12

int x = 12;

// Объявление переменной y и присвоение ей значения 3

double y = 3;

// Вывод значения переменной x на консоль

Console.WriteLine("x: " + x);

// Объявление переменной b и присвоение ей значения 9

double b = 9;

// Вывод значения переменной y на консоль

Console.WriteLine("y: " + y);

}

}

}



**Задание 2.** Создайте новое консольное приложение для решения задачи.

Введите вещественные числа x, у, z из области допустимых значений

исходных данных. Для преобразования к числовой форме используйте

класс Convert и метод Parse. Вычислите а, b. Результаты выведите на

экран с использованием формата и шаблонов.



using System;

namespace matemat1

{

class Program

{

// Объявление переменной x и присвоение ей значения, введенного пользователем

static void Main()

{

Console.Write("Введите x: ");

double x = Convert.ToDouble(Console.ReadLine());

// Объявление переменной y и присвоение ей значения, введенного пользователем

Console.Write("Введите y: ");

double y = Convert.ToDouble(Console.ReadLine());

// Объявление переменной z и присвоение ей значения, введенного пользователем

Console.Write("Введите z: ");

double z = Convert.ToDouble(Console.ReadLine());

// Вычисление значения переменной a

double a = Math.Sqrt(y - Math.Abs(x)) \* (x - Math.Sin(x + y));

// Вычисление значения переменной b

double b = Math.Cos(z) + (x \* x / 4);

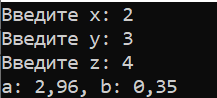
// Вывод результатов на консоль

Console.WriteLine($"a: {a:F2}, b: {b:F2}");

}

}

}



Контрольные вопросы:

1. Основные принципы технологии .NET.

* Многоязычность: .NET поддерживает несколько языков программирования, таких как C#, VB.NET, F#, и другие, что позволяет разработчикам выбирать наиболее подходящий язык для конкретной задачи.
* Управляемый код: .NET использует Common Language Runtime (CLR) для управления выполнением программ, обеспечивая безопасность, управление памятью и другие важные функции.
* Библиотека классов: .NET Framework предоставляет обширную библиотеку классов, которая содержит множество готовых компонентов для разработки приложений.
* Переносимость: Приложения, разработанные на платформе .NET, могут быть запущены на различных устройствах и операционных системах.

2. Что представляет собой платформа Visual Studio.NET?

Платформа Visual Studio .NET представляет собой интегрированную среду разработки (IDE) для создания приложений под управлением технологии .NET. Visual Studio предоставляет разработчикам широкий набор инструментов для написания, отладки, тестирования и развертывания приложений, а также интегрированные средства для работы с различными языками программирования, включая C#, VB.NET, F# и другие.

3. Как создать консольное приложение?

Для создания консольного приложения в Visual Studio .NET можно выполнить следующие шаги:

* + Открыть Visual Studio и создать новый проект.
  + Выбрать тип проекта "Console Application" (Консольное приложение) из доступных шаблонов.
  + Задать имя и местоположение проекта, а затем нажать "Create" (Создать).
  + После этого будет создано консольное приложение, где можно писать код для выполнения различных операций в командной строке.

4. Принципы объектно-ориентированного программирования.

* Инкапсуляция: скрытие внутренних данных объекта от внешнего доступа и предоставление доступа к ним только через методы объекта.
* Наследование: возможность создания новых классов на основе существующих, что позволяет повторно использовать код и создавать иерархии классов.
* Полиморфизм: способность объектов разных классов отвечать на одинаковые запросы с различными действиями, что обеспечивает гибкость и расширяемость программ.

5. Литералы. Как определяются типы литералов?

Литералы - это постоянные значения, встроенные непосредственно в текст программы. Типы литералов определяются на основе их формата и значения. Например, целочисленные литералы могут быть определены как целые числа, литералы с плавающей запятой - как числа с плавающей запятой, логические литералы - как значения true или false, и т. д.

6. Какие типы относятся к встроенным?

К встроенным типам относятся:

* + Целочисленные типы (int, long, short, byte и другие).
  + Типы с плавающей запятой (float, double).
  + Символьный тип (char).
  + Логический тип (bool).
  + Строковый тип (string).

7. Чем отличаются типы-значения и ссылочные типы?

Типы-значения (value types) хранят свои данные в стеке и содержат само значение, в то время как ссылочные типы (reference types) хранят ссылку на данные в куче. При работе с типами-значения происходит копирование значений, в то время как при работе с ссылочными типами копируется только ссылка на данные. Типы-значения в C# включают целочисленные типы, типы с плавающей запятой, логический тип, структуры и перечисления, в то время как классы, интерфейсы, делегаты и массивы относятся к ссылочным типам.

8. Какие типы числовых переменных имеются?

* Целочисленные типы: int, long, short, byte, sbyte, uint, ulong, ushort.
* Типы с плавающей запятой: float, double, decimal.
* Логический тип: bool.
* Символьный тип: char.

9. Что такое объявление и инициализация?

Объявление переменной - это процесс создания переменной с указанием её типа и имени. Инициализация переменной - это присвоение начального значения переменной в момент её создания или позже в программе.

10.Для чего используется упаковка и распаковка?

Упаковка (boxing) в C# используется для преобразования значимых типов (типов-значений) в ссылочные типы (object или другие классы). Распаковка (unboxing) - обратная операция, при которой значение упакованного объекта извлекается и преобразуется обратно в значимый тип.

11.Как в С# выполняется преобразование типа?

Преобразование типа в C# может быть явным (explicit) или неявным (implicit). Неявное преобразование выполняется автоматически компилятором, если нет потери данных, например, преобразование от int к long. Явное преобразование требует явного указания типа, например, при преобразовании от double к int.

12.Как осуществляется консольный ввод?

Для консольного ввода в C# можно использовать методы класса Console:

* Для чтения строки: Console.ReadLine().
* Для чтения символа: Console.ReadKey().
* Для чтения числа: считать строку с помощью Console.ReadLine() и затем преобразовать в нужный тип данных, например, с помощью метода int.Parse() для целых чисел.

13.Чем отличаются методы Read и ReadLine?

Методы Read() и ReadLine() в C# отличаются следующим образом:

* Read() считывает следующий символ из стандартного ввода (обычно консоли) и возвращает его в виде целочисленного значения (код символа).
* ReadLine() считывает следующую строку текста из стандартного ввода и возвращает её в виде строки.

14.Как обеспечить вывод данных на экран?

Для вывода данных на экран в C# можно использовать методы класса Console, например:

* Console.Write() для вывода текста без перехода на новую строку.
* Console.WriteLine() для вывода текста с переходом на новую строку.

15.Для чего предназначен и как используется форматный вывод данных?

Форматный вывод данных в C# предназначен для управления способом отображения данных при выводе. Для этого используется метод Console.WriteLine() с использованием специальных форматирующих строк, например:

* {0} - для подстановки первого аргумента.
* {1} - для подстановки второго аргумента.
* {0:N2} - для вывода числа с двумя знаками после запятой и т.д.

16.Каковы основные правила использования стандартных функций?

Основные правила использования стандартных функций включают:

* Понимание назначения и синтаксиса функции.
* Правильное указание аргументов функции в соответствии с её требованиями.
* Обработка возвращаемых значений функции, если они есть.
* Учет возможных исключений или ошибок, которые может генерировать функция.

17.Основные приемы работы в среде разработки Visual Studio.NET:

- Как создать консольное приложение?

- Как сохранить проект с заданным именем?

- Как загрузить проект?

- Как выполнить отладку программы?

- Как откомпилировать и выполнить программу?

- Как просмотреть результаты выполнения программы?

* Для создания консольного приложения:
  1. Откройте Visual Studio.
  2. Выберите "File" -> "New" -> "Project".
  3. В окне "New Project" выберите "Console App (.NET Core)".
  4. Укажите имя проекта и нажмите "Create".
* Для сохранения проекта с заданным именем:
  1. Выберите "File" -> "Save All".
  2. Укажите путь и имя файла проекта.
* Для загрузки проекта:
  1. Выберите "File" -> "Open" -> "Project/Solution".
  2. Укажите путь к файлу проекта и нажмите "Open".
* Для выполнения отладки программы:
  1. Установите точки останова (breakpoints) в нужных местах кода.
  2. Нажмите "Start Debugging" (F5) или выберите "Debug" -> "Start Debugging".
* Для компиляции и выполнения программы:
  1. Нажмите "Start" (зеленый треугольник) или выберите "Debug" -> "Start Without Debugging" (Ctrl + F5).
* Для просмотра результатов выполнения программы:
  1. Результаты вывода на консоль можно увидеть в окне Output.
  2. Если программа записывает результаты в файл, вы можете открыть этот файл для просмотра.