

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы трансляции

ОТЧЕТ
к лабораторной работе №1
на тему

**ОПРЕДЕЛЕНИЕ МОДЕЛИ ЯЗЫКА.
ВЫБОР ИНСТРУМЕНТАЛЬНОЙ ЯЗЫКОВОЙ СРЕДЫ**

Студент
Преподаватель

Д. С. Кончик
Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

1 Цель работы	3
2 Подмножество языка программирования.....	4
3 Инструментальная языковая среда.....	8
Заключение	9
Список использованных источников	10
Приложение А (обязательное) Листинг кода.....	11

1 ЦЕЛЬ РАБОТЫ

Цели лабораторной работы:

1 Определить подмножество выбранного языка программирования (типы констант, переменных, операторов и функций). В подмножество как минимум должны быть включены числовые и текстовые константы, 3-4 типа переменных, операторы цикла и условные операторы.

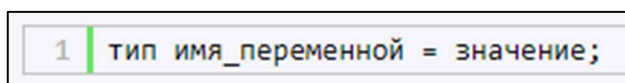
2 Определить инструментальную языковую среду, которая включает в себя язык программирования с указанием версии, на котором ведется разработка, операционная система, в которой выполняется разработка, и компьютер.

3 Предоставить тексты программ, включающих все элементы подмножества выбранного языка программирования.

2 ПОДМНОЖЕСТВО ЯЗЫКА ПРОГРАММИРОВАНИЯ

C# – высокоуровневый язык программирования, разработанный компанией *Microsoft*. Он объединяет простоту использования с мощностью и гибкостью, предоставляя разработчикам эффективные средства для создания высокопроизводительных приложений под платформу *.NET*. Язык поддерживает современные концепции программирования и обладает строгой типизацией, что способствует повышению надежности и безопасности кода. Он широко применяется для создания разнообразных приложений, включая веб-приложения, десктопные программы и игры [1].

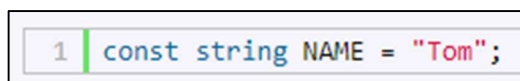
Для хранения данных в программе применяются переменные. Переменная представляет именнованную область памяти, в которой хранится значение определенного типа. Переменная имеет тип, имя и значение. Тип определяет, какого рода информацию может хранить переменная. Перед использованием любую переменную надо определить. Синтаксис определения переменной представлен на рисунке 1 [2].



```
1 | тип имя_переменной = значение;
```

Рисунок 1 – Определение переменной

Отличительной особенностью переменных является то, что можно изменить их значение в процессе работы программы. Но, кроме того, в *C#* есть константы. Константа должна быть обязательно инициализирована при определении, и после определения значение константы не может быть изменено. Для определения констант используется ключевое слово *const*, которое указывается перед типом константы (рисунок 2).



```
1 | const string NAME = "Tom";
```

Рисунок 2 – Определение константы

Как и во многих языках программирования, в *C#* есть своя система типов данных, которая используется для создания переменных. Тип данных определяет внутреннее представление данных, множество значений, которые может принимать объект, а также допустимые действия, которые можно применять над объектом [3].

В языке C# есть следующие типы данных:

- *bool*, хранит значение *true* или *false* (логические литералы);
- *byte*, хранит целое число от 0 до 255 и занимает 1 байт;
- *sbyte*, хранит целое число от -128 до 127 и занимает 1 байт;
- *short*, хранит целое число от -32768 до 32767 и занимает 2 байта;
- *ushort*, хранит целое число от 0 до 65535 и занимает 2 байта;
- *int*, хранит целое число от -2 147 483 648 до 2 147 483 647 и занимает 4 байта;
- *uint*, хранит целое число от 0 до 4 294 967 295 и занимает 4 байта;
- *long*, хранит целое число от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт;
- *ulong*, хранит целое число от 0 до 18 446 744 073 709 551 615 и занимает 8 байт;
- *float*, хранит число с плавающей точкой от $-3.4 \cdot 10^{38}$ до $3.4 \cdot 10^{38}$ и занимает 4 байта;
- *double*, хранит число с плавающей точкой от $\pm 5.0 \cdot 10^{-324}$ до $\pm 1.7 \cdot 10^{308}$ и занимает 8 байта;
- *decimal*, хранит десятичное дробное число от $\pm 1.0 \cdot 10^{-28}$ до $\pm 7.9228 \cdot 10^{28}$, может хранить 28 знаков после запятой и занимает 16 байт;
- *char*, хранит одиночный символ в кодировке *Unicode* и занимает 2 байта;
- *string*, хранит набор символов *Unicode*;
- *object*, может хранить значение любого типа данных и занимает 4 байта на 32-разрядной платформе и 8 байт на 64-разрядной платформе.

Литерал – запись в коде программы, которая имеет фиксированное (константное) значение. Литералами также называют представление некоторых типов данных в виде текстовых строк. С помощью литералов, в языке C#, переменным задаются значения (рисунок 3) [4].

```
250 - целочисленный литерал типа int
's' - литерал символьного типа char
'@' - литерал символьного типа char
11.235 - литерал с плавающей запятой типа double
"Text" - литерал строчного типа string
```

Рисунок 3 – Примеры литералов

Условные конструкции – один из базовых компонентов многих языков программирования, которые направляют работу программы по одному из путей в зависимости от определенных условий. Одной из таких конструкций в языке программирования *C#* является конструкция *if..else*. Конструкция *if/else* проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код (рисунок 4).

```
1 int num1 = 8;
2 int num2 = 6;
3 if(num1 > num2)
4 {
5     Console.WriteLine($"Число {num1} больше числа {num2}");
6 }
7 else if (num1 < num2)
8 {
9     Console.WriteLine($"Число {num1} меньше числа {num2}");
10 }
11 else
12 {
13     Console.WriteLine("Число num1 равно числу num2");
14 }
```

Рисунок 4 – Условная конструкция

Тернарная операция также позволяет проверить некоторое условие и в зависимости от его истинности выполнить некоторые действия (рисунок 5). Здесь сразу три операнда. В зависимости от условия тернарная операция возвращает второй или третий операнд: если условие равно *true*, то возвращается второй операнд; если условие равно *false*, то третий [5].

```
1 [первый операнд - условие] ? [второй операнд] : [третий операнд]
```

Рисунок 5 – Тернарная операция

Конструкция *switch/case* оценивает некоторое выражение и сравнивает его значение с набором значений. И при совпадении значений выполняет определенный код. После ключевого слова *switch* в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора *case*. И если совпадение будет найдено, то будет выполняться определенный блок *case* [6].

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз. В С# имеются следующие виды циклов [7]:

1 Цикл *for*. Объявление цикла состоит из трех частей. Первая часть объявления цикла – некоторые действия, которые выполняются один раз до выполнения цикла. Вторая часть – условие, при котором будет выполняться цикл. И третья часть – некоторые действия, которые выполняются после завершения блока цикла.

2 Цикл *do..while*. В цикле *do* сначала выполняется код цикла, а потом происходит проверка условия в инструкции *while*. И пока это условие истинно, цикл повторяется.

3 Цикл *while*. В отличие от цикла *do* цикл *while* сразу проверяет истинность некоторого условия, и если условие истинно, то код цикла выполняется.

4 Цикл *foreach* – предназначен для перебора набора или коллекции элементов. После оператора *foreach* в скобках сначала идет определение переменной. Затем ключевое слово *in* и далее коллекция, элементы которой надо перебрать.

Иногда возникает ситуация, когда требуется выйти из цикла, не дожидаясь его завершения. В этом случае можно воспользоваться оператором *break*. А если нужно, чтобы при проверке цикл не завершался, а просто пропускал текущую итерацию, можно воспользоваться оператором *continue*.

Если переменные хранят некоторые значения, то методы содержат собой набор инструкций, которые выполняют определенные действия. По сути метод – это именованный блок кода, который выполняет некоторые действия. Общее определение методов представлено на рисунке 6.

```
1 [модификаторы] тип_возвращаемого_значения название_метода ([параметры])
2 {
3     // тело метода
4 }
```

Рисунок 6 – Определение метода

Модификаторы и параметры необязательны. Перед названием метода идет возвращаемый тип данных. После названия метода в скобках идет перечисление параметров. После списка параметров в круглых скобках идет блок кода, который представляет набор выполняемых методом инструкций [8].

3 ИНСТРУМЕНТАЛЬНАЯ ЯЗЫКОВАЯ СРЕДА

В качестве языковой среды выбран язык программирования *Python 3.12*. Разработка основана на работе с операционной системой *Windows 10* на *PC*.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как *C* или *C++*.

Python является мультипарадигменным языком программирования. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Стандартная библиотека включает большой набор полезных переносимых функций, начиная с возможностей для работы с текстом и заканчивая средствами для написания сетевых приложений. Дополнительные возможности, такие как математическое моделирование, работа с оборудованием, написание веб-приложений или разработка игр, могут реализовываться посредством обширного количества сторонних библиотек, а также интеграцией библиотек, написанных на *C* или *C++*, при этом и сам интерпретатор *Python* может интегрироваться в проекты, написанные на этих языках. Существует и специализированный репозиторий программного обеспечения, написанного на *Python*, – *PyPI*. Данный репозиторий предоставляет средства для простой установки пакетов в операционную систему и стал стандартом де-факто для *Python* [9].

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы было определено подмножество языка программирования, включающее числовые и текстовые константы, различные типы переменных, циклы и условные операторы. Также было выбрано инструментальное окружение, состоящее из языка программирования и его версии, операционной системы и компьютера для разработки. В ходе работы были написаны программы, демонстрирующие использование всех элементов выбранного подмножества языка программирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Краткий обзор языка C# [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>.

[2] Переменные и константы [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/2.25.php>.

[3] Типы данных [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/2.1.php>.

[4] C#. Литералы. [Электронный ресурс]. – Режим доступа: <https://www.bestprog.net/ru/2016/10/14/литералы/>.

[5] Конструкция if..else и тернарная операция. [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/2.5.php>.

[6] Конструкция switch. [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/3.45.php>.

[7] Циклы. [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/2.6.php>.

[8] Методы. [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/2.8.php>.

[9] Python. [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Python>.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода

Листинг 1 – Первая программа

```
using System;

class Program
{
    static void Main()
    {
        // Использование переменных разных типов
        int age = 25;

        double height = 1.75;

        char gender = 'M';

        bool isStudent = true;

        // Условная конструкция if..else
        if (age >= 18 && gender == 'M')
        {
            Console.WriteLine("Вы совершеннолетний мужчина.");
        }
        else if (age >= 18 && gender == 'F')
        {
            Console.WriteLine("Вы совершеннолетняя женщина.");
        }
        else
        {
            Console.WriteLine("Вы несовершеннолетний.");
        }

        // Цикл for с использованием разных типов данных
        Console.WriteLine("Цикл for:");
        for (int i = 0; i < 5; i++)
        {
            if (i % 2 == 0)
            {
                Console.WriteLine($"Четное значение: {i}");
            }
            else
            {
                Console.WriteLine($"Нечетное значение: {i}");
            }
        }

        // Использование тернарной операции
        int number = 10;
        string result = (number % 2 == 0) ? "Четное" : "Нечетное";

        Console.WriteLine($"Число {number} - {result}");
    }
}
```

Листинг 2 – Вторая программа

```
using System;

class Program
{
    // Определение метода с разными типами параметров
    static void DisplayInfo(string name, int age, bool isStudent)
    {
        Console.WriteLine($"Имя: {name}");
        Console.WriteLine($"Возраст: {age}");
        Console.WriteLine($"Студент: {isStudent}");
    }

    // Определение метода для вычисления среднего значения
    static double CalculateAverage(int[] numbers)
    {
        int sum = 0;
        foreach (var number in numbers)
        {
            sum += number;
        }

        return (double)sum / numbers.Length;
    }

    static void Main()
    {
        // Использование массива разных типов данных
        object[] mixedArray = { "John", 30, true, 175.5 };

        // Цикл foreach для перебора элементов массива
        Console.WriteLine("Цикл foreach:");
        foreach (var item in mixedArray)
        {
            Console.WriteLine(item);
        }

        // Вызов метода с разными параметрами
        DisplayInfo("Alice", 22, false);

        // Использование массива целых чисел и вызов метода для вычисления
        // среднего значения
        int[] numbers = { 10, 20, 30, 40, 50 };
        double average = CalculateAverage(numbers);
        Console.WriteLine($"Среднее значение чисел: {average}");

        // Дополнительные операции
        for (int i = 0; i < 3; i++)
        {
            if (i % 2 == 0)
            {
                Console.WriteLine($"Дополнительная итерация {i}");
            }
            else
            {
                Console.WriteLine($"Еще одна дополнительная итерация {i}");
            }
        }
    }
}
```

Листинг 3 – Третья программа

```
using System;

class Program
{
    // Определение метода
    static int CalculateSum(int a, int b)
    {
        return a + b;
    }

    static void Main()
    {
        // Цикл while с операторами break и continue
        int counter = 0;
        Console.WriteLine("Цикл while с операторами break и continue:");
        while (counter < 10)
        {
            if (counter == 5)
            {
                Console.WriteLine("Достигнуто значение 5. Выход из цикла.");
                break;
            }

            if (counter % 2 == 0)
            {
                counter++;
                continue;
            }

            Console.WriteLine($"Текущее значение: {counter}");
            counter++;
        }

        // Использование switch/case
        Console.WriteLine("Выберите действие (1-3):");
        int choice = int.Parse(Console.ReadLine());

        // Использование switch/case для выбора действия
        switch (choice)
        {
            case 1:
                Console.WriteLine("Вы выбрали действие 1.");
                break;
            case 2:
                Console.WriteLine("Вы выбрали действие 2.");
                break;
            case 3:
                Console.WriteLine("Вы выбрали действие 3.");
                break;
            default:
                Console.WriteLine("Некорректный выбор.");
                break;
        }

        // Вызов метода и использование возвращаемого значения
        int result = CalculateSum(3, 7);
        Console.WriteLine($"Сумма чисел: {result}");
    }
}
```