Тема: «Разработка, отладка и испытание программ, основанных на операторах ветвления и циклических алгоритмах»

Цель: разрабатывать программы с применением ввода-вывода информации на экран, условного оператора, оператора множественного выбора и циклов на языке программирования C#.

Время выполнения: 4 часа.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Консольный вывод

Для вывода информации на консоль нам необходимо передать ее в метод `Console.WriteLine`:

```
string hello = "Привет мир";
Console.WriteLine(hello);
Console.WriteLine("Добро пожаловать в C#!");
Console.WriteLine("Пока мир...");
Console.WriteLine(24.5);
```

Console.Write

Кроме `Console.WriteLine()` можно также использовать метод `Console.Write()`, он работает точно так же за тем исключением, что не добавляет переход на следующую строку, то есть последующий консольный вывод будет выводиться на той же строке.

```
string name = "Tom";
int age = 34;
double height = 1.7;
Console.Write($"Имя: {name} Возраст: {age} Рост: {height}м");
```

Консольный ввод

Кроме вывода информации на консоль мы можем получать информацию с консоли. Для этого предназначен метод `Console.ReadLine()`. Он позволяет получить введенную строку.

```
Console.Write("Введите свое имя: ");
string? name = Console.ReadLine();
Console.WriteLine($"Привет {name}");
```

В данном случае все, что вводит пользователь, с помощью метода `Console.ReadLine()` передается в переменную `name`.

Конструкция if..else

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.

Ее простейшая форма состоит из блока if:

```
if(условие)
{
    выполняемые инструкции
}
```

После ключевого слова if ставится условие. Условие должно представлять значение типа bool. Это может быть непосредственно значение типа bool или результат условного выражения или другого выражения, которое возвращает значение типа bool. И если это условие истинно (равно true), то срабатывает код, который помещен далее после условия внутри фигурных скобок.

Выражение else

Но что, если мы захотим, чтобы при несоблюдении условия также выполнялись какие-либо действия? В этом случае мы можем добавить блок else:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2)
{
    Console.WriteLine($"Число {num1} больше числа {num2}");
}
else
{
    Console.WriteLine($"Число {num1} меньше числа {num2}");
}
```

Блок else выполняется, если условие после if ложно, то есть равно false. Если блок else содержит толко одну инструкцию, то оять же мы можем его сократить, убрав фигурные скобки:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2)
    Console.WriteLine($"Число {num1} больше числа {num2}");
```

```
else
    Console.WriteLine($"Число {num1} меньше числа {num2}");
```

Но в примере выше при сравнении чисел мы можем насчитать три состояния: первое число больше второго, первое число меньше второго и числа равны. Используя конструкцию else if, мы можем обрабатывать дополнительные условия:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2)
{
    Console.WriteLine($"Число {num1} больше числа {num2}");
}
else if (num1 < num2)
{
    Console.WriteLine($"Число {num1} меньше числа {num2}");
}
else
{
    Console.WriteLine("Число num1 равно числу num2");
}
```

При необходимости можно добавить несколько выражений else if:

```
string name = "Alex";

if (name == "Tom")
    Console.WriteLine("Вас зовут Tomas");
else if (name == "Bob")
    Console.WriteLine("Вас зовут Robert");
else if (name == "Mike")
    Console.WriteLine("Вас зовут Michael");
else
    Console.WriteLine("Неизвестное имя");
```

Оператор выбора switch

Оператор выбора switch обеспечивает многонаправленное ветвление в программе. Этот оператор позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программы.

В некоторых случаях оператор выбора switch может иметь более эффективное использование чем использование вложенных операторов if.

Общий вид оператора switch:

```
switch (выражение)
{
    case константа1:
        операторы1;
    break;
    case константа2:
        операторы2;
    break;

    ...

    case константаN:
        операторыN;
    break;

    ...

    default:
        операторы;
    break;
}
```

где:

выражение – заданное выражение целочисленного (char, byte, short, int) типа, перечисления или строчного (string) типа;

константа1, константа2, … константаN – константы выбора, тип которых может быть совместим с типом выражения. Среди констант выбора не должно быть двух с одинаковыми значениями;

оператор1, оператор2, …, операторN, оператор – последовательность операторов, которые выполняются в случае, если значение константы выбора совпадет со значением выражения.

Если ни одна из констант выбора не совпадает с заданным выражением, то выполняются операторы, которые следуют за словом default.

Операторы, указанные после слова default, выполняются в том случае, если значение выражения не совпадает ни с одной из констант. Блок (ветвь) default не является обязательной.

В случае, если блок default отсутствует в операторе switch и ни одна из констант не совпадает со значением выражения, то никаких действий не выполняется.

Тернарный оператор

Условный оператор ?:, также называемый тернарным, вычисляет логическое выражение и в зависимости от полученного значения true или false возвращает результат одного из двух соответствующих выражений, синтаксис условного оператора выглядит следующим образом:

```
condition ? consequent : alternative
```

Выражение condition должно принимать значение true или false. Если condition принимает значение true, вычисляется выражение consequent, а результат становится результатом операции. Если condition принимает значение false, вычисляется выражение alternative, а результат становится результатом операции. Вычисляется только выражение consequent или alternative.

Циклы

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз.

В C# имеются следующие виды циклов:

- for
- foreach
- while
- do...while

Цикл for

Цикл for имеет следующее формальное определение:

```
for ([действия_до_выполнения_цикла]; [условие];
[действия_после_выполнения])
{
    // действия
}
```

Объявление цикла for состоит из трех частей.

Первая часть объявления цикла - некоторые действия, которые выполняются один раз до выполнения цикла. Обычно здесь определяются переменные, которые будут использоваться в цикле.

Вторая часть - условие, при котором будет выполняться цикл. Пока условие равно true, будет выполняться цикл.

И третья часть - некоторые действия, которые выполняются после завершения блока цикла. Эти действия выполняются каждый раз при завершении блока цикла.

После объявления цикла в фигурных скобках помещаются сами действия цикла.

Рассмотрим стандартный цикл for:

```
for (int i = 1; i < 4; i++)
{
    Console.WriteLine(i);
}
```

Здесь первая часть объявления цикла - int i = 1 - создает и инициализирует переменную i.

Вторая часть - условие i < 4. То есть пока переменная i меньше 4, будет выполняться цикл.

И третья часть - действия, выполняемые после завершения действий из блока цикла - увеличение переменной i на единицу.

Весь процесс цикла можно представить следующим образом:

Определяется переменная int i = 1

Проверяется условие i < 4. Оно истинно (так как 1 меньше 4), поэтому выполняется блок цикла, а именно инструкция Console.WriteLine(i), которая выводит на консоль значение переменной i

Блок цикла закончил выполнение, поэтому выполняется треться часть объявления цикла - i++. После этого переменная i будет равна 2.

Снова проверяется условие i < 4. Оно истинно (так как 2 меньше 4), поэтому опять выполняется блок цикла - Console.WriteLine(i)

Блок цикла закончил выполнение, поэтому снова выполняется выражение i++. После этого переменная i будет равна 3.

Снова проверяется условие i < 4. Оно истинно (так как 3 меньше 4), поэтому опять выполняется блок цикла - Console.WriteLine(i)

Блок цикла закончил выполнение, поэтому снова выполняется выражение i++. После этого переменная i будет равна 4.

Снова проверяется условие i < 4. Теперь оно возвражает false, так как значение переменной i НЕ меньше 4, поэтому цикл завершает выполнение. Далее уже выполняется остальная часть программы, которая идет после цикла

В итоге блок цикла сработает 3 раза, пока значение i не станет равным 4. И каждый раз это значение будет увеличиваться на 1. Однократное

выполнение блока цикла называется итерацией. Таким образом, здесь цикл выполнит три итерации.

Результат работы программы:

1

2

3

Если блок цикла for содержит одну инструкцию, то мы можем его сократить, убрав фигурные скобки:

```
for (int i = 1; i < 4; i++)
    Console.WriteLine(i);

// или так
for (int i = 1; i < 4; i++) Console.WriteLine(i);
```

При этом необязательно именно в первой части цикла объявлять переменную, а в третий части изменять ее значение - это могут быть любые действия. Например:

```
var i = 1;

for (Console.WriteLine("Начало выполнения цикла"); i < 4; Console.WriteLine($"i = {i}"))
{
    i++;
}
```

Здесь опять же цикл срабатывает, пока переменная i меньше 4, только приращение переменной i происходит в блоке цикла. Консольный вывод данной программы:

Начало выполнения цикла

i = 2

i = 3

i = 4

Нам необязательно указывать все условия при объявлении цикла. Например, мы можем написать так:

```
int i = 1;
for (; ;)
{
    Console.WriteLine($"i = {i}");
    i++;
}
```

Формально определение цикла осталось тем же, только теперь блоки в определении у нас пустые: for (; ;). У нас нет инициализированной

переменной, нет условия, поэтому цикл будет работать вечно - бесконечный цикл.

Мы также можем опустить ряд блоков:

```
int i = 1;
for (; i<4;)
{
    Console.WriteLine($"i = {i}");
    i++;
}
```

Этот пример по сути эквивалентен первому примеру: у нас также есть переменная-счетчик, только определена она вне цикла. У нас есть условие выполнения цикла. И есть приращение переменной уже в самом блоке for.

Также стоит отметить, что можно определять несколько переменных в объявлении цикла:

```
for (int i = 1, j = 1; i < 10; i++, j++)
    Console.WriteLine($"{i * j}");
```

Здесь в первой части объявления цикла определяются две переменных: i и j. Цикл выполняется, пока i не будет равна 10. После каждой итерации переменые i и j увеличиваются на единицу. Консольный вывод программы:

```
1
4
9
16
25
36
49
64
81
```

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.

```
do
{
    действия цикла
```

```
}
while (условие)
Например:

int i = 6;
do
{
    Console.WriteLine(i);
    i--;
}
while (i > 0);
```

Здесь код цикла сработает 6 раз, пока i не станет равным нулю. Но важно отметить, что цикл do гарантирует хотя бы единократное выполнение действий, даже если условие в инструкции while не будет истинно. То есть мы можем написать:

```
int i = -1;
do
{
    Console.WriteLine(i);
    i--;
}
while (i > 0);
Хотя у нас переменная i меньше 0, цикл все равно один раз выполнится.
```

Цикл while

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

```
while (условие)
{
    действия цикла
}
Например:

int i = 6;
while (i > 0)
{
    Console.WriteLine(i);
    i--;
}
```

Цикл foreach

Цикл foreach предназначен для перебора набора или коллекции элементов. Его общее определение:

```
foreach(тип_данных переменная in коллекция)
{
    // действия цикла
}
```

После оператора foreach в скобках сначала идет определение переменной. Затем ключевое слово in и далее коллекция, элементы которой надо перебрать.

При выполнении цикл последовательно перебирает элементы коллекции и помещает их в переменную, и таким образом в блоке цикла мы можем выполнить с ними некоторые действия.

Например, возьмем строку. Строка по сути - это коллекция символов. И .NET позволяет перебрать все элементы строки - ее символы с помощью цикла foreach.

```
foreach(char c in "Tom")
{
    Console.WriteLine(c);
}
```

Здесь цикл foreach пробегается по всем символам строки "Tom" и каждый символ помещает в символьную переменную c. В блоке цикла значение переменной c выводится на консоль. Поскольку в строке "Tom" три символа, то цикл выполнится три раза. Консольный вывод программы:

```
T
o
m
```

Стоит отметить, что переменная, которая определяется в объявлении цикла, должна по типу соответствовать типу элементов перебираемой коллекции. Так, элементы строки - значения типа char - символы. Поэтому переменная c имеет тип char. Однако в реальности не всегда бывает очевидно, какой тип представляют элементы коллекции. В этом случае мы можем определить переменную с помощью оператора var:

```
foreach(var c in "Tom")
{
    Console.WriteLine(c);
}
```

Операторы continue и break

Иногда возникает ситуация, когда требуется выйти из цикла, не дожидаясь его завершения. В этом случае мы можем воспользоваться оператором break.

Например:

```
for (int i = 0; i < 9; i++)
{
    if (i == 5)
        break;
    Console.WriteLine(i);
}
```

Хотя в условии цикла сказано, что цикл будет выполняться, пока счетчик i не достигнет значения 9, в реальности цикл сработает 5 раз. Так как при достижении счетчиком i значения 5, сработает оператор break, и цикл завершится.

```
0
1
2
3
4
```

Теперь поставим себе другую задачу. А что если мы хотим, чтобы при проверке цикл не завершался, а просто пропускал текущую итерацию. Для этого мы можем воспользоваться оператором continue:

```
for (int i = 0; i < 9; i++)
{
    if (i == 5)
        continue;
    Console.WriteLine(i);
}
```

В этом случае цикл, когда дойдет до числа 5, которое не удовлетворяет условию проверки, просто пропустит это число и перейдет к следующей итерации:

```
0
1
2
3
```

4
6
7
8

Стоит отметить, что операторы break и continue можно применять в любом типе циклов.

Вложенные циклы

Одни циклы могут быть вложенными в другие. Например:

```
for (int i = 1; i < 10; i++)
{
    for (int j = 1; j < 10; j++)
    {
        Console.Write($"{i * j} \t");
    }
    Console.WriteLine();
}
```

В данном случае цикл for (int i = 1; i < 10; i++) выполняется 9 раз, то есть имеет 9 итераций. Но в рамках каждой итерации выполняется девять раз вложенный цикл for (int j = 1; j < 10; j++). В итоге данная программа выведет таблицу умножения.

Индивидуальные задания для лабораторной работы

Часть 1. Условный оператор (if else)

1. По заданному числу n закончите фразу "На лугу пасется..." одним из возможных продолжений: "n корова", "n коровы", "n коров", правильно склоняя слово "корова". Между числом и словом должен стоять ровно один пробел.
2. Товар стоит a руб. b коп. За него заплатили c руб. d коп. Сколько сдачи требуется получить? Вводятся 4 числа: a, b, c и d. Необходимо вывести 2 числа в виде: «Сдача составляет e рублей, f копеек».
3. Поле шахматной доски определяется парой чисел (a, b), каждое от 1 до 8, первое число задает номер столбца, второе – номер строки. Заданы две клетки. Определите, может ли шахматный король попасть с первой клетки на вторую за один ход. Даны 4 целых числа от 1 до 8 каждое, первые два задают начальную клетку, вторые два задают конечную клетку.

Начальная и конечная клетки не совпадают. Числа записаны в отдельных переменных.

4. Требуется определить, можно ли от шоколадки размером n × m долек отломить k долек, если разрешается сделать один разлом по прямой между дольками (то есть разломить шоколадку на два прямоугольника). Вводятся 3 числа: n, m и k; k не равно n × m. n × m <= 30000.

5. Билет на одну поездку в метро стоит 15 рублей, билет на 10 поездок стоит 125 рублей, билет на 60 поездок стоит 440 рублей. Пассажир планирует совершить n поездок. Определите, сколько билетов каждого вида он должен приобрести, чтобы суммарное количество оплаченных поездок было не меньше n, а общая стоимость приобретенных билетов – минимальна.

6. На сковородку одновременно можно положить k котлет. Каждую котлету нужно с каждой стороны обжаривать m минут непрерывно. За какое наименьшее время удастся поджарить с обеих сторон n котлет?

7. В кафе мороженое продают по три шарика и по пять шариков. Можно ли купить ровно k шариков мороженого?

8. Даны координаты двух точек на плоскости, требуется определить, лежат ли они в одной координатной четверти или нет (все координаты отличны от нуля).

9. Требуется определить, бьет ли ладья, стоящая на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке. Вводятся четыре числа: координаты ладьи (два числа) и координаты другой фигуры (два числа), каждое число вводится в отдельной переменной. Координаты - целые числа в интервале от 1 до 8.

10. Требуется определить, бьет ли слон, стоящий на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке. Вводятся четыре числа: координаты слона и координаты другой фигуры, каждое число вводится в отдельной переменной. Координаты - целые числа в интервале от 1 до 8.

Часть 2. Оператор выбора, оператор множественного выбора (switch case).

1. Составить расписание на неделю. Пользователь вводит название дня недели и у него на экране отображается, то, что запланировано на этот день. Суббота и воскресенье выходной на эти дни дел не должно быть запланировано.

2. Пользователь вводит номер дня недели. Программа должна вывести его наименование на экран.

3. Объявлено 2 переменные - одна текстовая, вторая целочисленная. Им присвоены значения. В одной переменной хранится в виде текста день недели, на который приходится первое число месяца. Во второй переменной в виде целого числа хранится значение текущего дня месяца от 1 до 31. Необходимо вывести на экран следующее сообщение: «n-е число месяца – это «название дня недели»».

4. Пользователь вводит 2 числа и математическую операцию («+», «-», «/» или «*») Программа выводит на экран результат введенной операции.

5. Пользователь вводит число от 1 до 12 (представляющее месяц), программа выводит количество дней в этом месяце (високосный год не учитывается).

6. Пользователь вводит название месяца, программа выводит время года, к которому этот месяц принадлежит.

7. Пользователь вводит жанр фильма, программа выводит список фильмов принадлежащих данному жанру (предусмотреть минимум 3 жанра, предусмотреть минимум 3 фильма для каждого жанра).

8. Пользователь вводит название факультета, программа выводит список специальностей данного факультета. Список факультетов и специальностей можно посмотреть по ссылке (https://abitur.bsuir.by/fakultety-i-spetsialnosti).

9. Пользователь вводит категорию водительского удостоверения, программа выводит соответствующее транспортное средство.

10. Пользователь вводит возраст, программа выводит категории водительского удостоверения, которые может получить пользователь.

Часть 3. Данные задачи необходимо решить при помощи цикла for, при помощи цикла while осуществить контроль введенных данных. Т.е. программа не заканчивает выполнение пока не будут введены валидные данные.

1. Вывести все квадраты натуральных чисел, не превосходящие заданного числа N. Например, если N = 50, то на экран должен быть выведен ряд 1 4 9 16 25 36 49. N должно быть в пределе от 25 до 300.

2. Написать программу, которая возводит число в целочисленную степень. Число и степень вводятся с клавиатуры. В степень возводить посредством цикла, Math.Pow не использовать. Число должно быть не более 100, степень не более 10.

3. Написать программу, проверяющую гипотезу Сиракуз для любого числа от 1 до 50. Программа должна вывести все промежуточные результаты и по достижении 1-цы завершить работу.

Гипотеза звучит следующим образом: «Возьмем любое натуральное число. Если оно четное - разделим его пополам, если нечетное - умножим на 3, прибавим 1 и разделим пополам. Повторим эти действия с вновь полученным числом. Гипотеза гласит, что независимо от выбора первого числа рано или поздно мы получим 1.»

4. Написать программу, подсчитывающую количество четных и нечетных цифр числа. Число должно быть ограничено шестью разрядами.

5. Написать программу, подсчитывающую сумму и произведение цифр числа. Число должно быть ограничено шестью разрядами.

6. Написать программу, которая запрашивает у пользователя сумму вклада, количество месяцев и процент по вкладу, а затем вычисляет конечную сумму вклада с учетом начисления процентов за каждый месяц. Процент по вкладу должен быть в пределах от 3х до 40.

7. Написать программу, подсчитывающую сумму первой и последней цифр любого целого положительного числа. Число должно быть ограничено шестью разрядами.

8. Вводится число, написать программу, которая преобразует его в другое число, цифры которого будут следовать в обратном порядке по сравнению с введенным числом. Число должно быть ограничено шестью разрядами.

9. Из натурального числа удалить заданную цифру. Число и цифру вводить с клавиатуры. Например, задано число 5683. Требуется удалить из него цифру 8. Получится число 563. Число должно быть ограничено шестью разрядами.

10. Написать программу, находящую самую большую цифру заданного числа. Число должно быть ограничено шестью разрядами.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть лабораторной работы.

2. Реализовать индивидуальное задание согласно варианту, сделать скриншоты работающих программ. Написать комментарии.

3. Написать отчет, содержащий:

1. Титульный лист, на котором указывается:

а) полное наименование министерства образование и название учебного заведения;

б) название дисциплины;

в) номер практического занятия;

г) фамилия преподавателя, ведущего занятие;

д) фамилия, имя и номер группы студента;

е) год выполнения лабораторной работы.

2. Индивидуальное задание (листинг кода программы, скриншоты консоли с результатами работы).

3. Вывод о проделанной работе.