Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем

и технологий»

**Отчёт**

по лабораторной работе №05

по дисциплине ***«Объектно-ориентированные технологии***

***программирования и стандарты проектирования»***

тема: «Базовый синтаксис языка Java. Циклические конструкции. Итерационные алгоритмы»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент группы 10701119  Прадед Кирилл Витальевич |
| Преподаватель: |  | Иванченко Виктор Викторович |

2020-2021 учебный год

ЛАБОРАТОРНАЯ РАБОТА #05

Базовый синтаксис языка Java. Циклические конструкции. Итерационные алгоритмы

**Цель работы**

Изучить синтаксис циклических конструкций языка Java для программирования итерационных алгоритмов и закрепить их на примере разработки простейших интерактивных консольных Java-приложений.

**Tребования**

1) Необходимо выполнить весь блок основных заданий, по одному соответствующему заданию – из индивидуальных заданий А и В. Дополнительное задание выполняется на Ваше усмотрение.

2) Для каждого вычислительного алгоритма необходимо спроектировать блоксхему решения, которую необходимо поместить в отчёт.

3) На базе спроектированных алгоритмов разработать простейшее консольное интерактивное приложение с использование архитектурного шаблона проектирования Model-View-Controller, MVC.

4) Создаваемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты: by.bntu.fitr.poisit.nameofstudent.javalabs.lab04.

5) При выполнения задания необходимо по максимуму пытаться разрабатывать универсальный, масштабируемый, легко поддерживаемый и читаемый код.

6) Также рекомендуется придерживаться Single Responsibility Principle, SRP (принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода.

7) Если логически не подразумевается или в задании иного не указано, то входными и выходными данными являются вещественные числа (числа с плавающей запятой).

8) Все задания необходимо решать используя только базовые операции (простые операторы), определённые над примитивными типами данных в языке программирования Java, и условные конструкции (т.е. не нужно использовать циклические конструкции, массивы, строковые данные и операции над ними и т.д.).

9) В соответствующих компонентах бизнес-логики необходимо предусмотреть «защиту от дурака».

10)Для генерирования случайных чисел воспользуйтесь методами объекта класса java.util.Random, а для реализации ввода данных с консоли (терминала) – соответствующими методами объекта класса java.util.Scanner. 11)Программа должна обязательно быть снабжена комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, название бригады (если есть), номер группы и дату разработки. Исходный текст классов и демонстрационной программы рекомендуется также снабжать поясняющими краткими комментариями.

12)Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом для взаимодействия с пользователем. Интерфейс программы и комментарии в коде должны быть на английском языке.

13) При проверки работоспособности приложения необходимо проверить все тестовые случаи. 14)При выполнении задания не рекомендуется использовать интегрированные средства разработки (Integrated Development Environment, IDE). Следует задействовать любой текстовый редактор и основные компоненты JDK (компилятор – javac, утилиту для запуска JVM – java).

15)При разработке программ придерживайтесь соглашений по написанию кода на Java (Java Code-Convention)

**Код бизнес-логики программ прикреплен в приложении A. Полный код программ можно найти на GitHub -**

**https://github.com/kirillpraded/JavaLabs**

**Задание A 1)**

Написать программу, которая подсчитывает количество цифр заданного натурального числа.

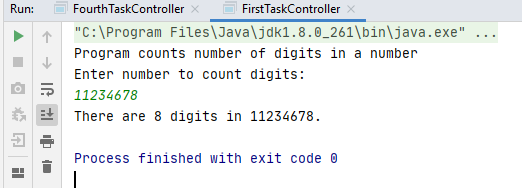
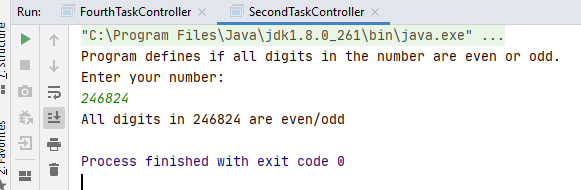


Рис. 1 – результат выполнения программы.

**Задание B 1)**

Разработайте программу, которая проверяет, что все цифры, которые входят в заданное натурального число, имеют одинаковую чётность, т.е. либо все чётные, либо все нечётные.



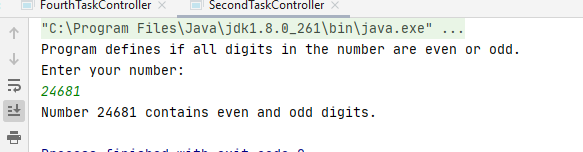
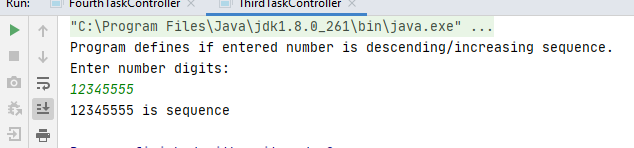


Рис. 2-3 – результат выполнения второго задания

**Задание C 1)**

Разработайте программу, которая проверяет, что цифры заданного натурального числа образуют

возрастающую (или убывающую) последовательность.



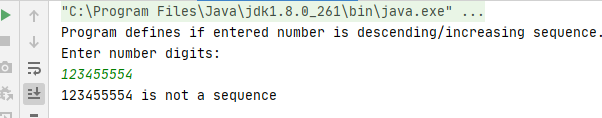


Рис. 4-5 – результат выполнения третьего задания

**Задание D 1)**

Найти количество различных цифр у заданного натурального числа.

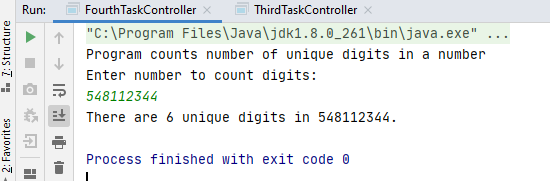


Рис. 6 – результат выполнения четвертого задания

**Задание E 1)**

Найти конкретное число Фибоначчи заданного его порядковым номером.

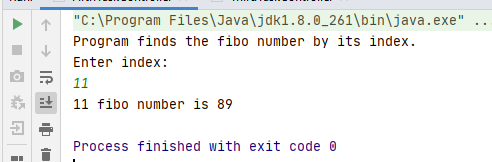


Рис. 7 – результат выполнения пятого задания

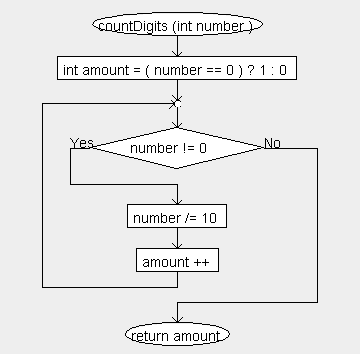
*Приложение А. Код программы.*

**Полный исходный код - https://github.com/kirillpraded/JavaLabs**

**В приложении представлена бизнес-логика приложений. Контроллеры и View можно найти по ссылке на гитхабе.**

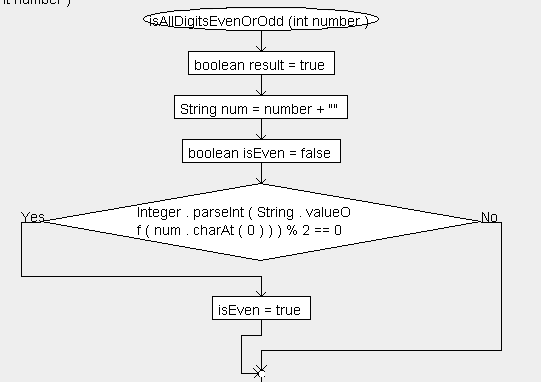
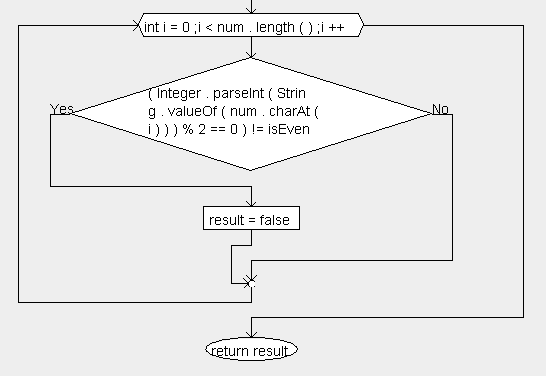
**Задание А1**

public static int countDigits(int number) {  
 int amount = (number == 0) ? 1 : 0;  
 while (number != 0) {  
 number /= 10;  
 amount++;  
 }  
 return amount;  
}



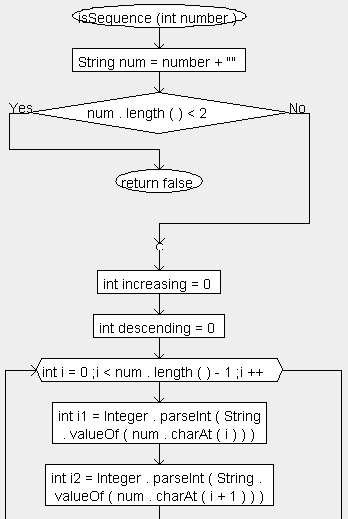
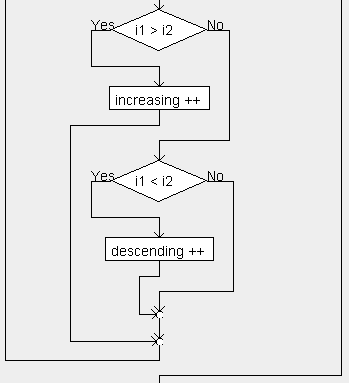
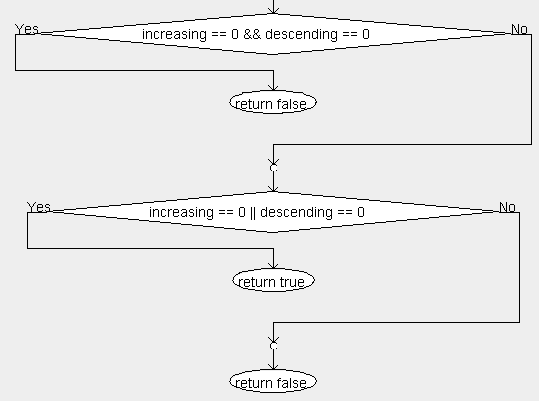
**Задание B1**

public static boolean isAllDigitsEvenOrOdd(int number) {  
  
 boolean result = true;  
 String num = number + "";  
 boolean isEven = false;  
  
 if (Integer.parseInt(String.valueOf(num.charAt(0))) % 2 == 0) {  
 isEven = true;  
 }  
  
 for (int i = 0; i < num.length(); i++) {  
 if ((Integer.parseInt(String.valueOf(num.charAt(i))) % 2 == 0) != isEven) {  
 result = false;  
 }  
 }  
 return result;  
}

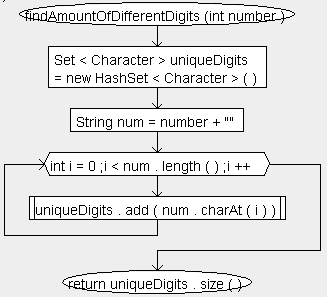
**Задание C1**

public static boolean isSequence(int number) {  
 String num = number + "";  
  
 if (num.length() < 2) {  
 return false;  
 }  
  
 int increasing = 0;  
 int descending = 0;  
  
 for (int i = 0; i < num.length() - 1; i++) {  
 int i1 = Integer.*parseInt*(String.*valueOf*(num.charAt(i)));  
 int i2 = Integer.*parseInt*(String.*valueOf*(num.charAt(i + 1)));  
 if (i1 > i2) {  
 increasing++;  
 } else if (i1 < i2) {  
 descending++;  
 }  
 }  
  
 if (increasing == 0 && descending == 0) {  
 return false;  
 }  
  
 if (increasing == 0 || descending == 0) {  
 return true;  
 }  
  
 return false;  
}

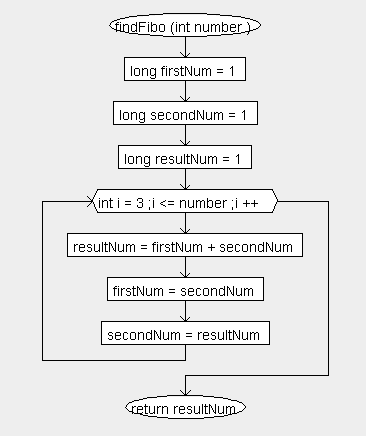
**Задание D1**

public static int findAmountOfDifferentDigits(int number) {  
 Set<Character> uniqueDigits = new HashSet<Character>();  
 String num = number + "";  
  
 for (int i = 0; i < num.length(); i++) {  
 uniqueDigits.add(num.charAt(i));  
 }  
  
 return uniqueDigits.size();  
}



**Задание E1**

public static long findFibo(int number) {  
 long firstNum = 1;  
 long secondNum = 1;  
 long resultNum = 1;  
 for (int i = 3; i <= number; i++) {  
 resultNum = firstNum + secondNum;  
 firstNum = secondNum;  
 secondNum = resultNum;  
 }  
 return resultNum;  
}



**Контрольные вопросы**

1) Какие алгоритмы называются итерационные (итеративные) и где они применяются?

Циклический процесс, который продолжается до тех пор, пока разность между соседними, уточняемыми на каждом шаге цикла (итерации) значениями, не окажется меньше или равной некоторой заданной величине.

По сути в простом представлений это алгоритм перебора чего-либо (коллекции, массива, строки посимвольно…) с помощью итераций(циклов).

2) Что такое итерация?

Итерация в программировании — в широком смысле — организация обработки данных, при которой действия повторяются многократно, не приводя при этом к вызовам самих себя(рекурсии).

3) Какие разновидности циклов существуют в мире программирования и когда они используются?

В общем циклы делятся на два – с постусловием и с предусловием.

Эти циклы еще разделяются на множество других. Циклы с постусловием используются, когда необходимо, чтобы прошла хотя бы одна итерация, во всех остальных случаях – с предусловием.

4) Какие в языке Java существуют разновидности циклических конструкций (сложных операторов циклов)?

While

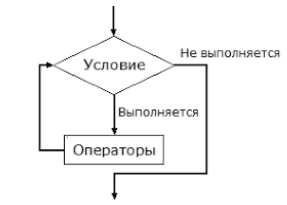
Do while

For

Foreach

Iterator – циклом не является но широко используется для работы с коллекциями (по статистике, зачастую быстрее чем просто использование for или foreach).

5) Опишите синтаксис самой универсальной циклической конструкции с предусловием while и её обозначение в виде блок-схемы? Как она работает? Когда данная конструкция применяется?



Цикл принимает в себя только Boolean значения. Соответственно и используется когда необходимо использовать когда можно подставить Boolean значение.

6) Чем является выражение внутри скобок после ключевого слова while – инициализацией, условием или обновлением?

условием

7) Опишите синтаксис циклической конструкции с постусловие do-while и её обозначение в виде блок-схемы? Как она работает? Когда данная конструкция применяется?

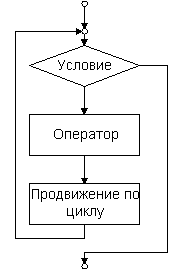
Применяется когда необходимо произвести хотя бы одну итерацию.



8) Опишите синтаксис циклической конструкции с предусловием (или со счётчиком) for и её обозначение в виде блок-схемы? Когда данная конструкция применяется?

For(int I = 0; I < 100; i++) {

}



Цикл for удобно применять когда нам нужно итерироваться по коллекции/массиву/строке как массиву символов.

9) Какие блоки в цикле for являются не обязательными? Если данные блоки пустые, то что подразумевается по умолчанию?

Все. В таком случае for(;;) подразумевается бесконечный цикл.

10) Опишите последовательность выполнения каждого блока цикла for, т.е. что за чем выполняется или не выполняется.

Первый блок выполняется до начала цикла(int I = 0)

Второй блок выполняется вначале каждой итерации.

Третий блок выполняется в конце каждой итерации.

11) Обязательно ли инициализировать и(или) использовать переменнуюсчётчик цикла for внутри самого цикла?

Нет.

12) Какая разновидность цикла for появилась в Java начиная с версии JDK 5.0? В чём предназначение данного цикла?

Цикл вида foreach (for(int a : array)). Появился как синтаксический сахар, никакого нового функционала в себе не несет.

13) Опишите синтаксис расширенного цикла for (в простонародий его обычно называют foreach). Как он работает и где его обычно используют?

Работает в коллекциях/массивах. Без индексов просто пробегает по каждому элементу массива или коллекции. Главный минус, на мой взгляд, в сравнений с for, нельзя получить индекс текущего элемента.

14) Какая из циклических конструкций в Java выполняется немного быстрее остальных?

Никакой. В любом случае итератор быстрее всех циклов в случае с коллекциями, но если речь идет не о коллекции, а о простом бесконечном цикле (for(;;) и while(true)) то в таком случае быстрее будет for. Написать 7 символов быстрее чем 11.

15) Может ли условное выражение в цикле содержать сразу значение истины? Если да, то для чего это обычно может использоваться?

Да, может, таким образом будет реализован бесконечный цикл.

16) А значение других типов данных в качестве условия цикла можно использовать в Java?

В while только Boolean

17) Что такое бесконечный цикл (условно бесконечный цикл)?

Цикл, условие выхода из которого никогда не выполнится. Может быть остановлен оператором break или завершением работы приложения.

18) Когда обычно применяется бесконечный цикл?

Для написания консольных приложений.

19) Приведите примеры реализации бесконечного цикла с использование циклических конструкций, которые есть в Java.

Do{}while(true);

While(true)

For(;;)

20) Что такое недостижимый код в Java? Является ли он правильным с точки зрения синтаксиса языка Java, т.е. будет ли данный код компилироваться?

Данный код будет компилироваться, но никогда не выполнится.

Недостижимый код может быть представлен кодом за пределами бесконечного цикла(который действительно никогда не закончится, то есть без break) или же кодом после return statement.

21) Отличительная особенность циклов Java с аналогичными циклами в других языках программирования (к примеру, в C/C++/C# и Python)?

Цикл while принимает только true.

22) Приведите примеры использования циклов для реализации следующих последовательностей чисел: a. 1 2 3 4 5 6 7 8 9 10 … ; b. 10 9 8 7 6 5 4 3 2 1 … ; c. 10 12 14 16 18 20 … ; d. 95 85 75 65 55 45 35 25 15 5; e. … а что ещё можно?

a. For(int I = 1; I <= 10; i++)

b. int I = 10;

while(I > 0) {

sout(i);

i--;

}

Итд…

Можно любую последовательность цифр имеющую определенную зависимость.

23) Какова роль использования оператора break в теле цикла?

Прерывает выполнение цикла.

24) Какова роль использования оператора continue в теле цикла? Что про данные оператор говорит соглашение по разработке на Java? Как можно избавиться в коде от оператора continue?

Пропускает итерацию цикла. По соглашению такой оператор использовать крайне не рекомендуется. Использование этого оператора говорит о некомпетентности разработчика.

Избавится от него можно изменением условия или добавлением услвоной конструкции.

25) Какова роль использования пустого оператора в теле цикла?

Пропуск итерации?

26) Что такое метка и для чего обычно на используется в языках программирования?

символьное имя, идентификатор для более удобного указания данных и кода в языках программирования

Можно вернутся к метке или используя метку в java пометить цикл(если используется несколько вложенных циклов – крайне удобно, но не рекомендуется) и прервать его, а внешний цикл продолжит итерации. Break label; - так выглядит классическая конструкция прерывающая цикл.

27) С каким оператором обычно используется метка в языка C/C++/C#?

Goto

28) В каких языках программирования злоупотребляют применения меток?

C/C++/C#

29) Для чего обычно в Java используют метки и с какими операторами?

Можно вернутся к метке или используя метку в java пометить цикл(если используется несколько вложенных циклов – крайне удобно, но не рекомендуется) и прервать его, а внешний цикл продолжит итерации. Break label; - так выглядит классическая конструкция прерывающая цикл.

30) Если вы собираетесь использовать вложенные циклы для вывода элементов матрицы в виде строк и столбцов, какой из циклов будет печатать строки: внутренний или внешний?

Внешний цикл будет в конце печатать переход на новую строку(System.out.println();)

А внутренний будет печатать элементы(System.out.print(element[i][j]);)