**Лабораторная работа №10**

**Основы программирования на Java: оператор цикла с параметром**

**Цель работы:** Освоить основные способы создания Java-программ либо с помощью обычного редактора, либо с помощью среды разработки. Приобрести навыки программирования циклических алгоритмов в Java.

**Теоретические сведения**

**Цикл —** операторы выполняются последовательно: первый оператор функции выполняется первым, затем второй и так далее. Цикл используется в ситуации, когда Вам нужно выполнить блок кода несколько раз.

Языки программирования предлагают различные структуры управления, позволяющие более сложные пути выполнения.

**Циклы**

Объявления цикла позволяет выполнить оператор или группу операторов несколько раз.

Существуют следующие циклы в Java для обработки циклических требований:

|  |  |
| --- | --- |
| **Цикл** | **Описание** |
| [while](http://proglang.su/java/loop-while) | Повторяет оператор или группу операторов, пока заданное условие является true. Цикл проверяет условие до выполнения тела цикла. |
| [for](http://proglang.su/java/loop-for) | Выполняет последовательность операторов несколько раз и сокращает код, которым управляет переменная цикла. |
| [улучшенный for](http://proglang.su/java/improved-loop-for) | Выполняет последовательность операторов несколько раз и сокращает код, которым управляет переменная цикла. |
| [do...while](http://proglang.su/java/loop-do-while) | Выполняется цикл while, за исключением того, что он проверяет условия в конце тела цикла. |

**Операторы цикла**

**Операторы цикла** - изменяют нормальное выполнение последовательности цикла в Java. Когда выполнение выходит из своей области, все объекты, которые были созданы автоматически в этой области, будут уничтожены.

Java поддерживает следующие управляющие операторы цикла.

|  |  |
| --- | --- |
| **Оператор** | **Описание** |
| [break](http://proglang.su/java/operator-break) | Завершает работу цикла или оператора switch, и передаёт выполнение следующему, который находится сразу же после цикла или оператора switch. |
| [continue](http://proglang.su/java/operator-continue) | Заставляет цикл, пропустить оставшуюся часть его тела и сразу перепроверить вновь его состояние сначала. |

**Java — Цикл for**

**Цикл for**  обладает структурой управления повторений, что позволяет эффективно написать цикл, который должен выполниться определенное количество раз. Цикл for полезен, когда Вы знаете, сколько раз задача должна быть повторена.

**Синтаксис**

Синтаксис цикла for в Java:

For (инициализация; логическое выражение; обновление)

{

//Операторы

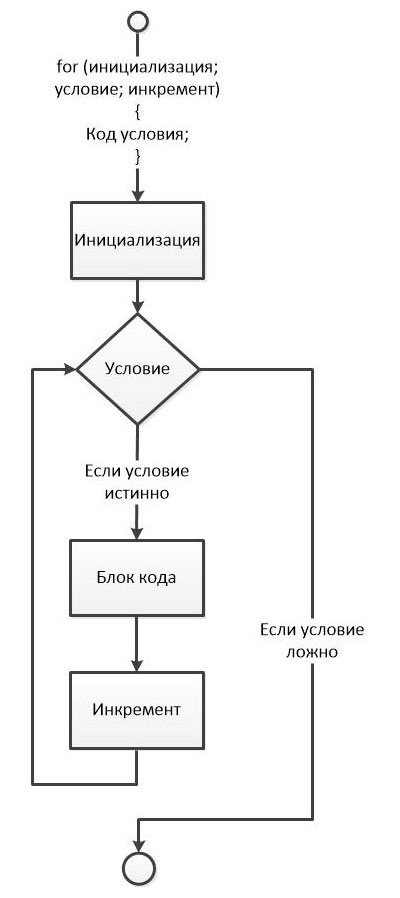
}

Процесс управления в цикле:

* Стадия инициализации выполняется первой, и только один раз. Этот шаг позволяет объявлять и инициализировать любые переменные для управления циклом, и он заканчивается точкой с запятой(;).
* Далее логическое выражение. Если истинно, то тело цикла выполняется, если оно ложно, тело цикла не будет выполнено и контроль переходит к следующему оператору мимо цикла.
* После того как тело из цикла for запускается на выполнение, контроль переходит обратно к оператору обновления. Он позволяет обновлять какие-либо переменные для управления циклом, и записывается без точки с запятой в конце.
* Логическое выражение теперь оценивается снова. Если истинно, то цикл выполняется и процесс повторяется. Если ложно, то цикл for завершается.

**Описание процесса**

Обозначение в блок-схеме



Тело цикла

Параметр цикла, границы

**Пример**

Ниже приведен пример кода цикла for в Java:

public class Test {

public static void main(String args[]) {

for(int x = 10; x < 15; x = x+1) {

System.out.print("Значение x: " + x );

System.out.print("\n");

}

}

}

Будет получен следующий результат:

Значение х: 10

Значение х: 11

Значение х: 12

Значение х: 13

Значение х: 14

**Java — Улучшенный цикл for**

**Улучшенный цикл for** в основном используется для обхода коллекцией элементов, включая массивы. Был введен начиная с Java 5.

**Синтаксис**

Синтаксис улучшенного цикла for:

For (объявление : выражение)

{

//Операторы

}

* **Объявление:** к новому объявленному блоку переменной осуществляется доступ, который имеет тип совместимый с элементами массива.
* **Выражение:** вычисляет нужное Вам в массиве цикла. Выражение может быть переменной или методом массива, вызов которого возвращает массив.

**Пример**

public class Test {

public static void main(String args[]){

int [] numbers = {10, 20, 30, 40, 50};

for(int x : numbers ){

System.out.print( x );

System.out.print(",");

}

System.out.print("\n");

String [] names ={"Олег", "Иван", "Дима", "Юля"};

for( String name : names ) {

System.out.print( name );

System.out.print(",");

}

}

}

Будет получен следующий результат:

10,20,30,40,50,

Олег,Иван,Дима,Юля,

**Java — Оператор break**

В языке программирования Java оператор break имеет следующие два использования:

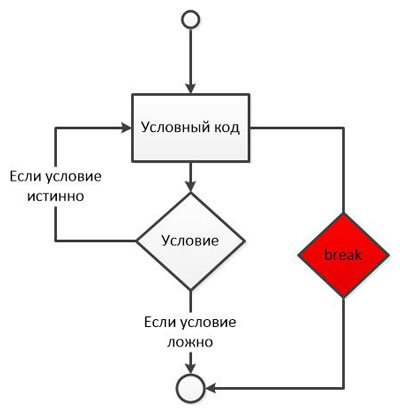
1. Когда оператор break встречается внутри цикла, то цикл прерывается, а управление программой возобновляется со следующего оператора.
2. Он может быть использован для прекращения case в операторе switch.

**Синтаксис**

Синтаксис одного оператора break в Java внутри любого цикла:

break;

**Описание процесса**



**Пример**

public class Test {

public static void main(String args[]) {

int [] numbers = {10, 20, 30, 40, 50};

for(int x : numbers ) {

if( x == 30 ) {

break;

}

System.out.print( x );

System.out.print("\n");

}

}

}

Будет получен следующий результат:

10

20

**Java — Оператор continue**

**Оператор continue -** ключевое слово, которое может быть использовано в любой структуре цикла, и оно призывает цикл сразу же перейти к следующей итерации.

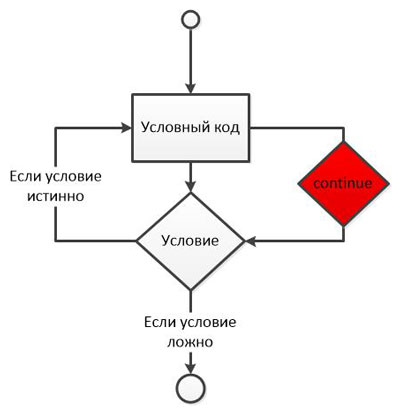
* В цикле [for](http://proglang.su/java/loop-for) ключевое слово continue распоряжается процессом так, чтобы сразу же перейти к оператору обновления.
* В циклах [while](http://proglang.su/java/loop-while) или [do...while](http://proglang.su/java/loop-do-while), контроль сразу же переходит в логическое выражение.

**Синтаксис**

Синтаксис одного оператора continue внутри любого цикла в Java:

continue;

**Описание процесса**



**Пример**

public class Test {

public static void main(String args[]) {

int [] numbers = {10, 20, 30, 40, 50};

for(int x : numbers ) {

if( x == 30 ) {

continue;

}

System.out.print( x );

System.out.print("\n");

}

}

}

Будет получен следующий результат:

10

20

40

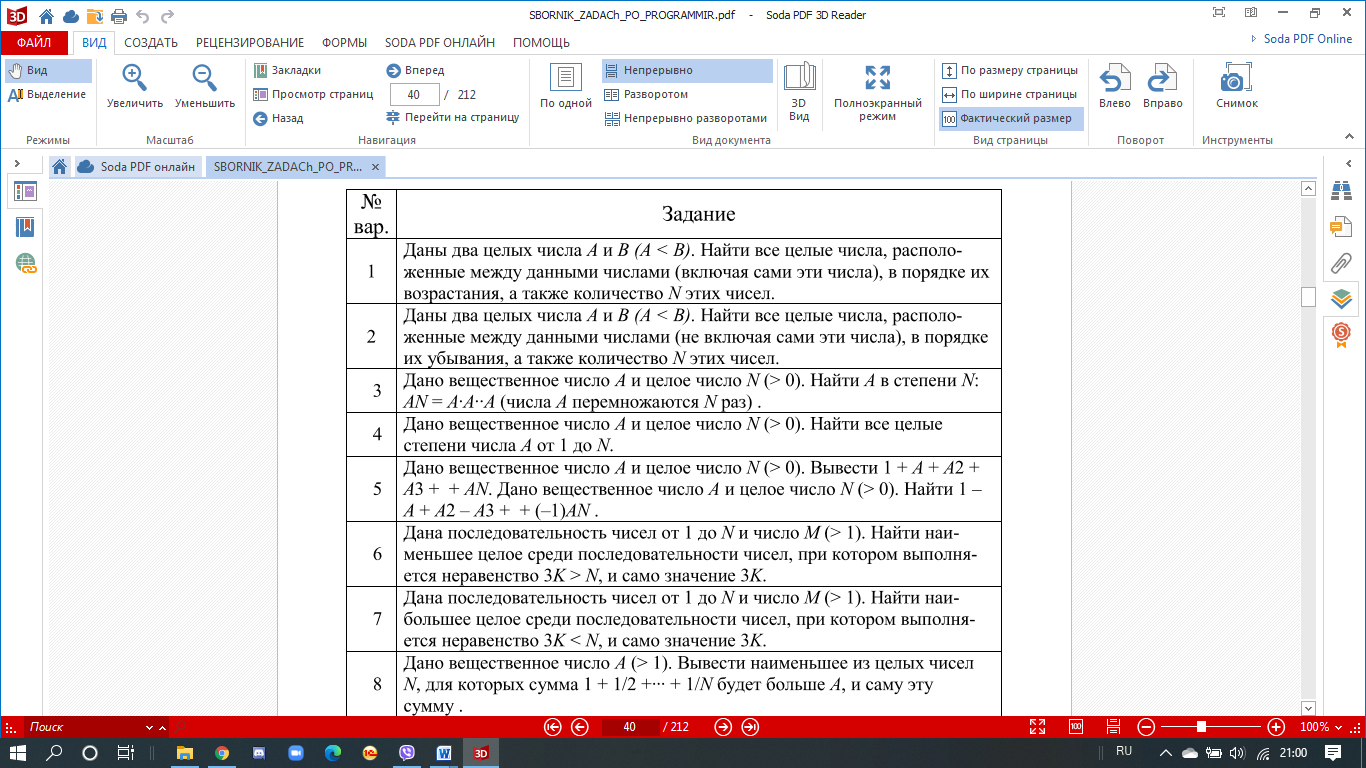
50

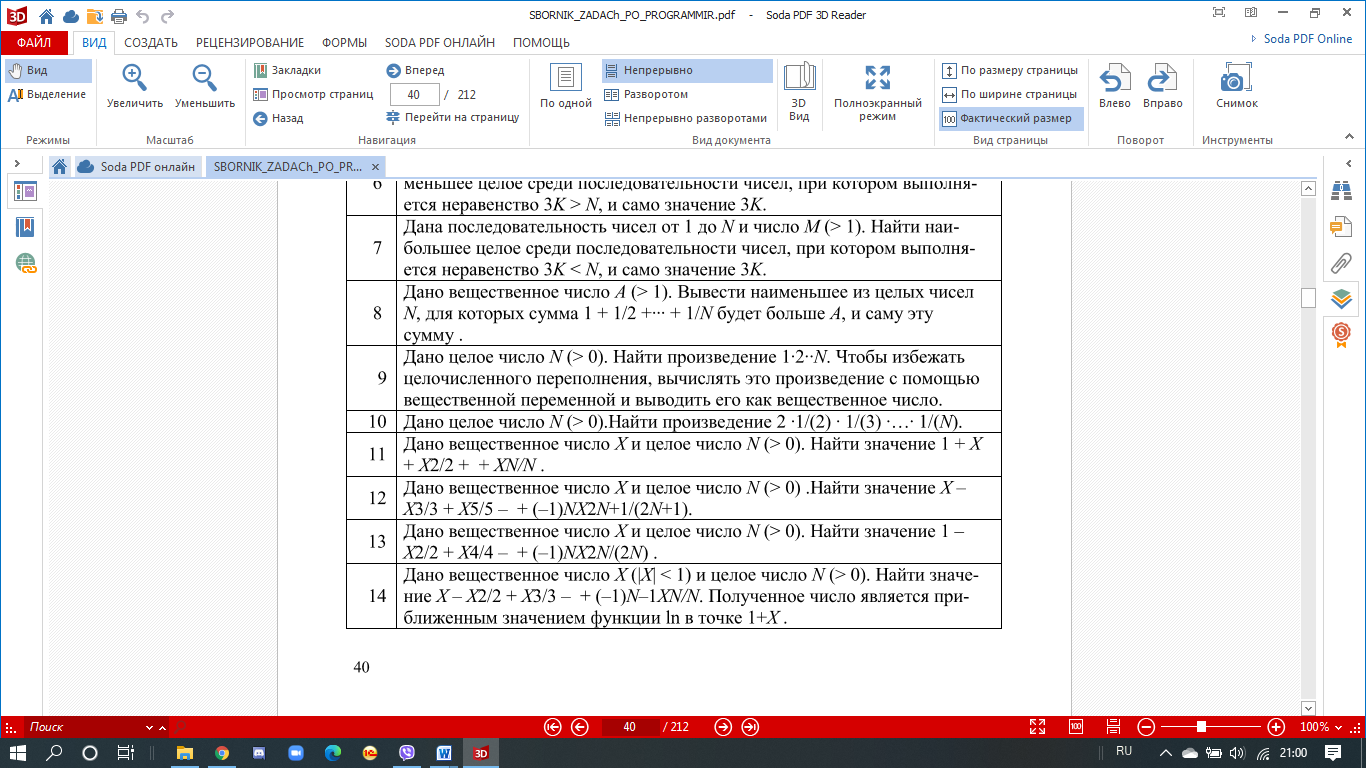
**Задание**

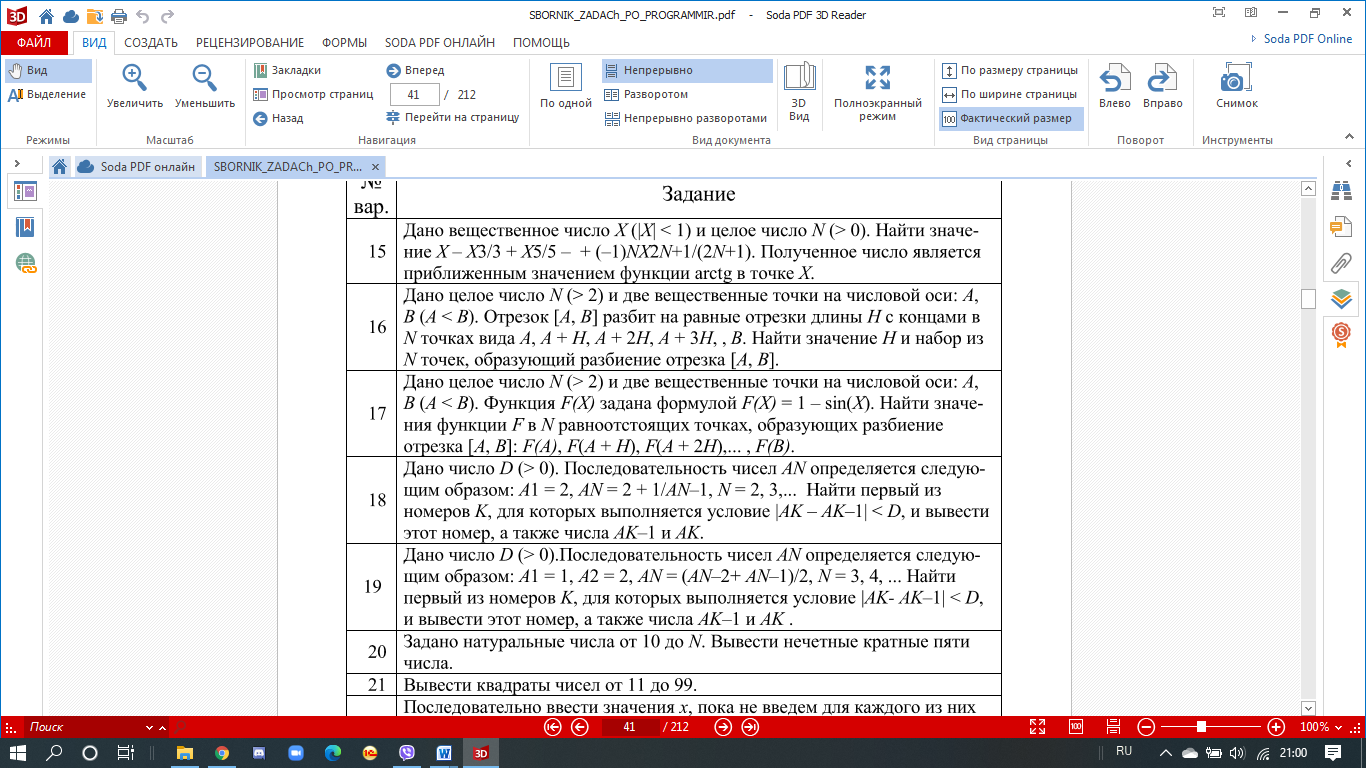
Изучить теоретический материал и выполнить задачи по вариантам.

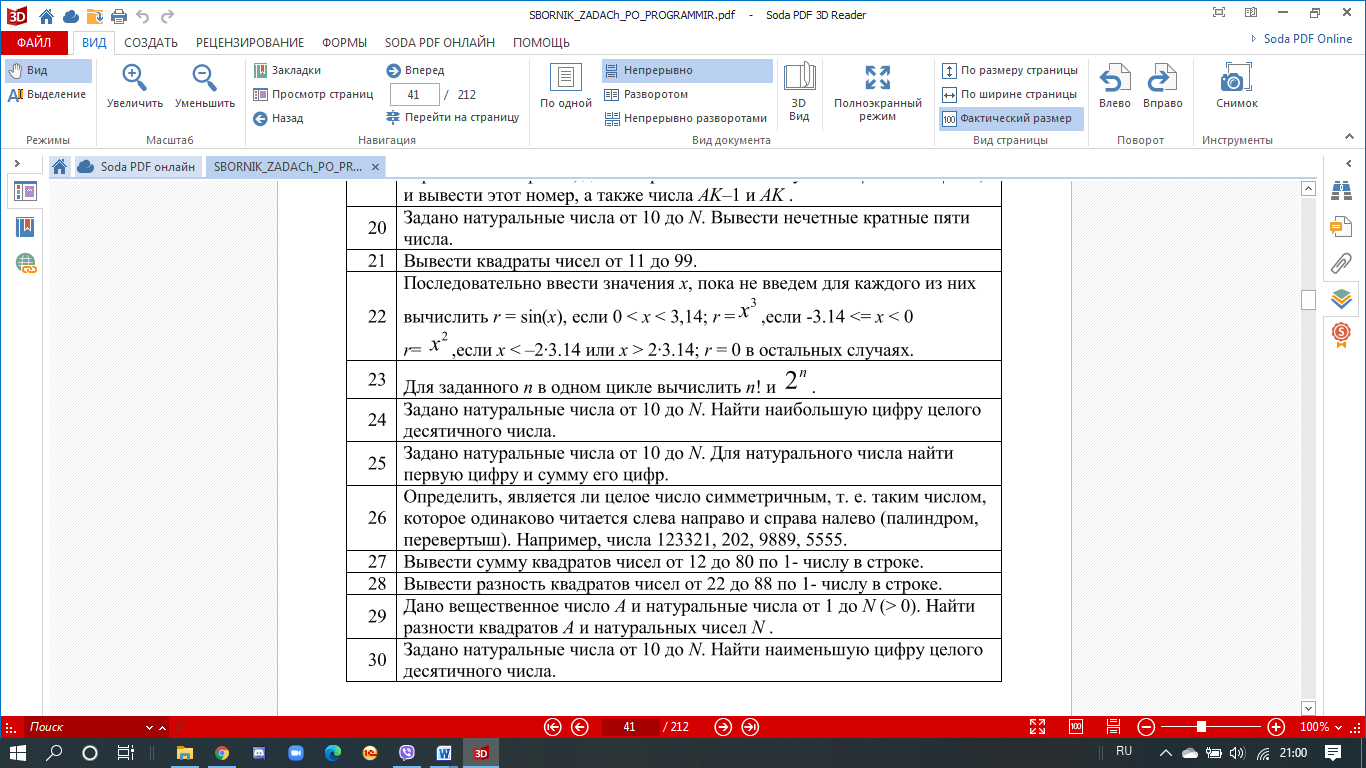
Составить блок-схему алгоритма и проект программы решения задачи, исходные данные ввести с клавиатуры.

**Задача**









**Форма отчета**

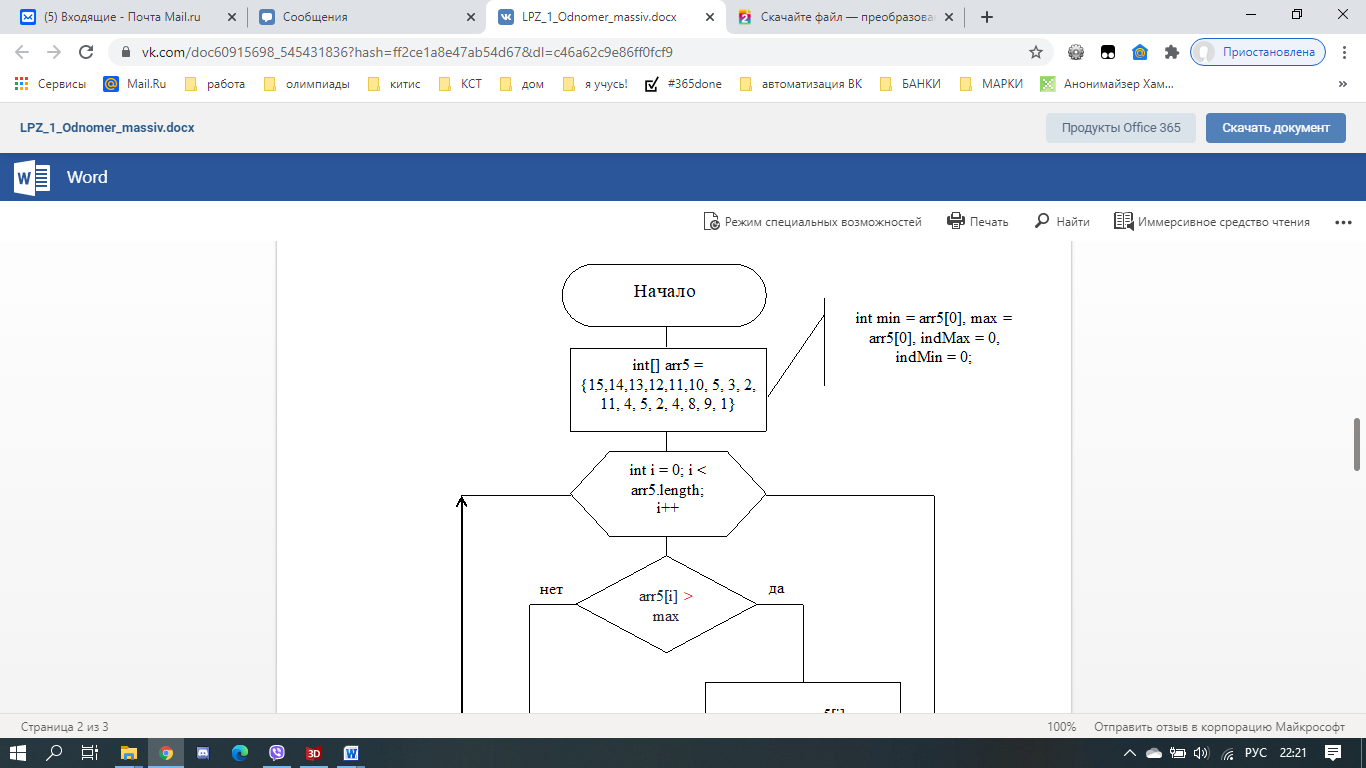
1. Тема, цель, индивидуальное задание

2. Таблица идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
|  |  |  |
|  |  |  |

3. Блок-схема алгоритма (см. Приложение)

Пример:



4. Код программы

5. Скриншот результата

**Контрольные вопросы для защиты**

1. Что такое цикл?
2. Какие операторы цикла в Java?
3. Оператор for?
4. Улучшенный цикл for. Примеры.
5. Операторы break, continue.

**Приложение**

**Условные графические обозначения в схемах алгоритмов и программ, отображающие основные операции процесса обработки данных и программирования по ГОСТ 2.708 - 81**

Размер, а следует выбирать из ряда 10, 15, 20 мм. Допускается увеличение размера а на число, кратное 5. Размер b принима­ют равным 1,5а.

|  |  |
| --- | --- |
|  | Процесс. Выполнение операций (группы операций), в результате которых изменяется значение (форма представления, расположение) данных  Решение. Выбор направления алгоритма (програм­мы) в зависимости от некоторых переменных условий |
|  | Решение. Выбор направления алгоритма (програм­мы) в зависимости от некоторых переменных условий |
|  | Модификация. Выполнение операций, меняющих команды (группы команд), изменяющих программу |
|  | Предопределенный процесс. Использование ранее созданных и описанных отдельно алгоритмов (программ) |
|  | Ввод-вывод. Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод) |
|  | Соединитель. Указание связи между прерванны­ми линиями потока (связывющие символы) |
|  | Пуск-останов. Начало, конец, прерывание про­цесса обработки данных или выполнения программы |
|  | Межстрочный соединитель |