ЛАБОРАТОРНАЯ РАБОТА №2. ТИПЫ ДАННЫХ И УПРАВЛЯЮЩИЕ СТРУКТУРЫ JAVA

Цель работы: получить практические навыки работы с управляющими структурами на языке Java.

2.1 Методические рекомендации

2.1.1 Комментарии

В языке Java поддерживается три типа комментариев: однострочный, многострочный, документационный.

1) Однострочные комментарии

Однострочный комментарий начинается символами // и завершается в конце строки. Компилятор игнорирует всё от // до конца строки.

Пример:

```
// This example demonstrates the use of single line comments
public class HelloWorld                    // Declare class as
{                                          // with name HelloWorld
        public static void main(String args []) // Define main
        {
                System.out.println( ”Hello World.” ); // Prints Hello
                                                      // World to console
        }                                             // End of main
}// End HelloWorld class
```

2) Многострочные комментарии

Многострочный комментарий начинается с /* и заканчивается */. Он может занимать одну или более строк. Компилятор игнорирует всё от /* до */.

Пример:

```
/* This is a sample class which is used to demonstrate the use of multi-line comments. This comment does not appear in the java documentation */
public class HelloWorld
{
        public static void main(String args [])
        {
                System.out.println(”Hello World. ” );
        }
}
```

3) Документационные комментарии

Документационные комментарии начинаются с / ** и заканчиваются / и могут также занимать одну или более строк. Компилятор игнорирует этот вид комментария, точно так же как игнорируются комментарии / и */.

Документационные комментарии идут дальше, позволяя создавать отдельную документацию через использование ***javadoc***-инструмента, с помощью которого пользователь может создавать документацию API в форме страниц HTML. Документаци-

онная информация собирается непосредственно от исходного текста и комментариев документа.

Также могут внедряться html-тэги в пределах комментариев документа.

Инструмент javadoc распознает только документационные комментарии, располагающиеся сразу перед классом, интерфейсом, конструктором, методом или полевыми объявлениями. Также при объявлении должен быть только один комментарий.

2.1.2 Ключевые слова

Ключевые слова – предопределенные идентификаторы, зарезервированные в Java. Программисты не могут использовать их как идентификаторы. True, false, и null не ключевые слова, но они являются зарезервированными словами, так что использовать их как имена в программах нельзя.

***Вот примеры категорий ключевых слов*:**

Примитивные типы данных:

- **byte;**
- **short;**
- **int;**
- **long;**
- **float;**
- **double;**
- **char;**
- **boolean.**

Ключевые слова выполнения цикла:

- **do;**
- **while;**
- **for;**
- **break;**
- **continue.**

Ключевые слова, выполняющие переход:

- **if;**
- **else;**
- **switch;**
- **case;**
- **default;**
- **break.**

Метод, переменная и модификаторы класса:

- **private;**
- **public;**
- **protected;**
- **final;**
- **static;**
- **abstract;**
- **synchronized;**
- **volatile;**
- **strictfp.**

Буквальные константы:

– **false;**
– **true;**
– **null.**

Ключевые слова, связанные с методом:

– **return;**
– **void.**

Ключевые слова, связанные с пакетом:

– **package;**
– **import.**

Ключевые слова, связанные с обработкой особых ситуаций:

– **try;**
– **catch;**
– **finally;**
– **throw;**
– **throws.**

Другие ключевые слова:

– **new;**
– **extends;**
– **implements;**
– **class;**
– **instanceof;**
– **this;**
– **super.**

2.1.2 Основные типы данных

Язык Java является строго *типизированным***.** Это значит, что каждая переменная и каждое выражение имеют свой определенный тип. Автоматического приведения типов попросту не существует. Иногда, конечно, возникает необходимость работать с переменными разных типов, и Java позволяет это сделать, но об этом – немного позже.

Всего в Java существует восемь простых типов данных, которые условно можно разделить на четыре группы:

– **целые (*Integers*)** – описывает целые числа с учетом знака. Включает в себя **byte, short, int, long;**

– **числа с плавающей запятой (*Floating-point number*)** – описывает «дробные» числа. Их нельзя считать полноценными дробными, т. к. они имеют определенную погрешность. Это типы: **float и double;**

– **символьный тип (*Characters*)** – это тип **char**, который предназначен для описания символов (буквы, цифры, математические знаки, и др.);

– **логический тип (*Boolean*)** – это тип **Boolean**, описывающий переменные, которые могут принимать значение «правда» (true) или «ложь» (false).

Теперь подробнее о каждом типе:

Целые – хранят значения целых чисел, без дробной части.

Название	Размер (диапазон)
byt	8 бит (от –128 до 127)

e	
short	16 бит (от –32 768 до 32 767)
int	32 бита (от –2 147 483 648 до 2 147 483 647)
long	64 бита (от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807)

Важной особенностью Java является возможность присвоения переменной не только значения в десятичной системе счисления, но также и в двоичной, восьмеричной и шестнадцатеричной.

Пример использования:

int i = 10; // переменной *i* присвоить значение 10 в десятичной системе счисления

int i2 = 0x10; // переменной *i* присвоить значение 10 в шестнадцатеричной системе счисления (16 в десятичной)

int i3 = 010; // переменной *i* присвоить значение 10 в восьмеричной системе счисления (9 в десятичной)

int i4 = b10; // переменной *i* присвоить значение 10 в двоичной системе счисления (2 в десятичной)

Типы с плавающей запятой – соответствующие стандарту IEEE754-2008.

Название	Размер (диапазон)
float	32 бита (от 3.4e–38 до 3.4e+38)
double	64 бита (от 1.7e–308 до 1.7e+308)

Здесь e – это степень 10, на которую умножается исходное число, так называемая экспонента.

То есть 1.5e+3 можно записать, как 1.5*1000, таким образом, записи 1.5e+3 и 1500 эквивалентны.

Как говорилось ранее, использование таких чисел предполагает некоторую погрешность, которая заложена изначально в стандарте IEEE754-2008. Но вы можете не беспокоиться, для простых задач начального уровня типа float более чем достаточно, не говоря уже о double.

Пример использования:

float f = 10.5; // переменной *f* присвоить значение 10 целых и 5 десятых

Следует помнить, что в Java, как и других языках программирования, дробная часть числа отделяется от целой точкой, а не запятой.

Символьный тип используется для описания символьных констант и переменных. Во многих языках, тип **char** занимает 8 бит (1 байт) и хранит ASCII-код символа, но в Java это не так. Так как Java создавался, основываясь на опыте более ранних языков, то в нем было учтено, что 255 ASCII-символов недостаточно для написания мультиязычных программ и веб-приложений. Поэтому тип char в Java занимает не 8, а 16 бит, что позволяет хранить в переменной этого типа любой символ Unicode. Также символ юникода можно ввести по его номеру, например: ‘u0061’ и ‘a’ имеют

одинаковое значение. Некоторые символы нельзя набрать с клавиатуры, либо они не имеют изображения. Например, чтобы присвоить переменной символ «одинарная кавычка», то запись **char ch = ‘’’**; будет вызывать ошибку компиляции. Для этого используются так называемые *Escape-последовательности* (комбинация символа и других). Помимо u существуют Escape-последовательности ():

- **xxx** – восьмеричный символ (xxx);
- **uxxxx** – шестнадцатеричный символ Юникод, где xxxx – номер символа;
- **\’** – одиночная кавычка;
- **\’’** – двойная кавычка;
- \ обратный слэш;
- **\r**– возврат каретки;
- **\n**– перевод строки (новая строка);
- **\f** – перевод страницы;
- \t- табуляция;
- **\b** – возврат на один символ (backspace).

Пример использования:
char ch = ‘J’; // переменной ch присвоить значение ‘J’.

Логический (булевский, boolean) тип может принимать только 2 значения: *true* и *false* (англ. «правда» и «ложь»). Этот тип может показаться совсем не нужным при написании программ. Вы спросите: «Зачем мне использовать boolean, если я могу взять int, который позволит хранить 2 000 000 000 значений?» Да, можете, но в некоторых задачах удобнее использовать именно булевский тип. Например, в условных операторах.

Пример использования:
boolean b = true; // переменной *b* присвоить значение true («правда»).

2.1.3 Идентификаторы

Идентификаторы используются в качестве имен классов, методов и переменных, чтобы однозначно определять их при создании идентификаторов, должны быть учтены следующие пункты:

- идентификатор может быть любой последовательностью букв верхнего и нижнего регистров, чисел или символов подчеркивания и знака коммерческого $;
- идентификатор может содержать только два специальных символа, символ подчёркивания (_) и знак доллара ($). Все другие специальные символы не используются;
- идентификатор не может содержать пробел.

Вот некоторые принятые и не принятые идентификаторы.
Принятые идентификаторы:

- **x1;**
- **first_Number;**
- **countNumber;**
- **This_ls_long_ldentifier.**

Непринятые идентификаторы:

- **1x;**
- **first number;**
- **x.no;**
- **44567.**

Обратите внимание*:* Java подобно javascript учитывает регистр, так что идентификатор *count* отличается от идентификатора *Count.*

2.1.4 Назначение переменных и инициализация

Переменная – значение, которое может измениться по мере необходимости в течение выполнения программы; они представлены символическими именами. Другими словами, значение переменной изменяется всякий раз, когда назначается новое значение. С точки зрения программиста, переменная имеет три характеристики:

- ***имя;***
- ***начальное значение;***
- ***область видимости.***

Имя переменной также называют идентификатором. Всякий раз, когда объявляется переменная, ей присваивается значение, и это значение является значением по умолчанию. Переменные также имеют свою область, которая определяет их видимость и время жизни в различных местах программы.

Рассмотрим синтаксис объявления переменной,

- **data type identifier [=value] [, identifier[=«value] .]**

Обратите внимание: текст, написанный между [и], является дополнительным.

Когда мы объявляем более одной переменной, мы используем запятую, чтобы отделить идентификаторы. Например,

- **int count, age; char ch.**

Переменным могут быть назначены значения, как показано ниже.

- **count = 0;**
- **age = 20;**
- **ch = "A".**

Можно объединять объявление и инициализацию вместе. Например,

- **int count = 10, age=20;**
- **char c = "A".**

Обратите внимание: объявляйте значимые идентификаторы в программе, чтобы избежать путаницы.

Переменная может быть инициализирована динамически. Следующий код показывает, как динамически инициализирована переменная.

Пример:

```
class VarInit
{
        public static void main(String args[])
```

```
    {
        int numl = 10, num2 = 20;
        int result = numl + num2;
        System.out.printing(«result =» + result);
    }
}
```

В этом коде результат суммы num1 и num2 присваивается result, то есть значение result не известно, пока выполнение программы не закончено. Следовательно, это называют динамической инициализацией.

2.1.5 Пакет Java.Lang

Пакет java.lang содержит классы, которые являются базовыми в языке Java. Этот пакет содержит классы, которые являются фундаментальными для дизайна языка программирования Java. Самые важные классы – ***Object***, который является корнем иерархии класса, и ***Class***, образцы которого представляют классы во время выполнения.

Часто необходимо представить значение примитивного типа, как будто это объект. Обертка классов ***Boolean***, ***Character***, ***Integer***, ***Long***, ***Float*** и ***Double*** реализует эту цель. **Объект типа Double**, например, содержит поле, тип которого является **double,** представляя значение таким способом, что ссылка к этому полю может быть сохранена в переменной типа ссылки. Эти классы также обеспечивают множество методов для того, чтобы преобразовать среди примитивных значений, также поддерживать такие стандартные методы подобно «равняется» и *hashCod*e. ***Void*** класс – non-instantiable класс, который содержит ссылку на объект Class, представляющий примитивный тип void.

Класс ***Math*** обеспечивает математические функции типа синуса, косинуса, и квадратного корня. Классы ***String*** и ***StringBuffer***, подобно классу Math, обеспечивают обычно используемые операции на символьных строках.

Классы ***ClassLoader***, ***Process***, ***Runtime***, ***SecurityManager*** и ***System*** обеспечивают «системные операции», которые управляют динамической загрузкой классов, созданием внешних процессов, ведущих запросов среды типа времени дня и принуждения политики безопасности.

Некоторые статические методы доступны в **java.lang**. **Класс Math** – чтобы исполнять математические операции. Статические методы – методы, к которым можно непосредственно обратиться с помощью класса, не создавая объектную ссылку класса.

abs() – этот метод возвращает абсолютное значение числа. Аргумент может иметь тип int, float, double или long. Тип byte и short преобразуется в int, если их передают как параметры.

Например:

```
int num = –1 ;
Math.abs(num); // returns 1
```

floor() – этот метод возвращает целое число, если оно меньше или равно параметру.

Примеры:

```
Math.floor(-5.6); // returns –6.0
Math.floor(201.1); // returns 201.0
Math.floor(100); // returns 100.0
```

max() – этот метод находит большее из двух значений. Аргумент может иметь тип данных int, long, double и float.

Примеры:

Math.max(100,200); // returns 200

min() – этот метод находит меньшее из двух значений. Аргумент может иметь тип данных int, long, double и float.

Примеры:

Math.min(100,200) returns 100

random() – этот метод возвращает случайное число между 0.0 и 1.0 из типа double.

round() – этот метод округляет аргумент с плавающей точкой к самому близкому числу. Например, запись Math.round (34.5) возвращается 35.

2.1.6 Управляющие структуры

Компьютер хорошо умеет исполнять повторные операции. Он может исполнять такие операции неустанно 10, 100 или даже 10 000 раз. Каждый машинный язык должен иметь особенности, которые инструктируют компьютер, как исполнять такие повторные задачи, и эти задачи решаются с помощью управляющих структур.

Управляющие структуры Java:

- **lf-else;**
- **switch;**
- **loops:**
- **while;**
- **do-while;**
- **for.**

Обсудим каждый вышеупомянутый пункт.

Конструкция if-else

Конструкция if-else также известна как *условная управляющая структура*, или *структура выбора*, потому что она проверяет данное условие и исполняет указанную задачу в зависимости от того, является ли условие истинным или ложным.

Синтаксис структуры if- else:

Для простой записи if-else

```
if(condition)
    statement;
else
    statement.
```

Для множественной записи if-else:

```
if(condition){
    statement;
    statement2;
    statementN;
} else {
    statement1;
    statement2;
    statementN;
}
```

Мы можем использовать «if» без «else» следующим образом:

```
if (condition){
      statements;
}
```

Обратите внимание: условие, записанное с «if» возвращает значение переменной типа boolean, то есть истина или ложь. Использование переменной типа boolean также позволяется.

Например,

```
boolean buttonClick = true;
if (buttonClick) {
        statements;
} else {
        statements;
}
```

Следующий код показывает пример структуры if-else.

Пример:

```
class IfElse
{
        public static void main(String args[]) {
                int num = 10;
                if(num %  2 == 0)
                        System.out.println(num + «is even number»);
                else
                        System.out.println(num + «is odd number»);
        }     //end of main
}   // end of IfElse
```

В вышеупомянутом коде переменная num типа int и имеет значение 10. Здесь в записи структуры if-else, если условие if имеет значение True, то выполняется соответствующая запись ниже if, в противном случае выполняется условие else.

Обратите внимание: если используется только одна конструкция в «if» или в «else», то нет никакой потребности в использовании пары изогнутых фигурных скобок.

Мы можем использовать вложенную структуру if-else в программе. Следующий фрагмент кода показывает, как использовать вложенную структуру if- else.

```
if (hour >= 7 && hour < 12)
    System.out.println(«Good Morning»);
else if(hour >= 12 && hour < 16)
    System.out.println(«Good Afternoon»);
else if(hour >= 16 && hour < 20)
    System.out.println(«Good Evening»);
else
    System.out.println(«Good Night»).
```

Конструкция switch

Объявление **switch** используется, когда должны быть выполнены многократные сравнения состояний. Оператор switch может также заменять длинный ряд вложенных операторов if-else-if. Оператор switch может объявляться выражением или переменной.

Синтаксис использования switch:

```
switch (condition) {
        case constant_valuel   :
                statement(s);
                break;
        case constant_value2   :
                statement(s);
                break;
        case constant_valueN   :
                statement(s);
                break;
        default:
                statement(s);
}
```

В конструкции **switch**, независимо от того, какое постоянное значение дается наряду с ключевым словом «**case**», всё согласуется с результатом условия (condition). Ключевое слово «**default**» указывает на то, что, если ни одно из значений case не соответствует с результатами условия, данного в switch, тогда выполняется условие, заданное по умолчанию.

Оператор break используется внутри switch, чтобы закончить последовательность операторов. Когда встречается оператор break, выполнение передается к первой строке кода, которая следует за оператором switch. Он создает эффект досрочного выхода из switch.

Вот некоторые пункты, которые вы должны помнить при написании условий и постоянных значений:

1) значение выражения сравнивается с каждым из указанных значений в case-операторах;
2) если соответствие найдено, то выполняется кодовая последовательность, следующая после этого оператора case;
3) если ни одна из case-констант не соответствует значению выражения, то выполняется оператор default. Оператор default необязателен;
4) если согласующихся case нет, и default не присутствует, то никаких дальнейших действий не выполняется.

Следующий фрагмент кода иллюстрирует использование оператора **switch**:

```
switch(day)   {
        case 0 : System.out.println(«Sunday»);
                        break;
        case 1 : System.out.println(«Monday»);
                        break;
```

```
        case 2 : System.out.println(«Tuesday»);
                 break;
        case 3 : System.out.println(«Wednesday»);
                 break;
        case 4 : System.out.println(«Thursday»);
                 break;
        case 5 : System.out.println(«Friday»);
                 break;
        case 6 : System.out.println(«Saturday»);
                 break;
        default : System.out.println(«Invalid day of week»);
}
```

В вышеупомянутом примере, если день между 0 и 6, то буде(у)т выполнено(ы) утверждение(я), связанно(ы)е с соответствующим case, но если день (day) больше, чем 6, то буде(у)т выполняться утверждение(я), связанно(ы)е со значением по умолчанию.

Подобно вложенным if-else, мы можем использовать вложенные структуры switch. Во вложенных структурах switch, значения внешнего выключателя и внутреннего выключателя могут быть те же самые.

2.1.7 Организация циклов

Процесс повторяемого выполнения блока утверждений известен как цикл loop. Утверждения в блоке могут быть выполнены любое количество раз, от ноля до бесконечного. Если loop продолжается всегда, его называют бесконечным циклом. Java поддерживает такие особенности циклов, которые дают возможность нам развивать короткие программы, содержащие повторяющиеся процессы.

While loop

While loop также известен как итерация. Это наиболее широко используемая структура циклов loop.

Синтаксис использования while loop:

```
while(condition)  {
        statements;
}
```

Если **while loop** содержит только одно утверждение, то изогнутые фигурные скобки писать не надо. Но если есть более одного утверждения, то они должны быть включены в пределы изогнутых фигурных скобок; иначе loop выполняет только первое утверждение, пока условие истинно, при этом другие утверждения просто игнорируются.

Подобно if-else, условие while loop возвращает значение булевской переменной или выражение, которое возвращает булевское значение.

Следующая программа демонстрирует использование while loop.

Пример:

```
class WhileDemo   {
        public static void main(String args[])  {
                int num = 10, count = 0;
                while(num <= 15)   {
                        System.out.print(«num =» + num +  «\t») ;
                        count++;
                        num++;
       }
                System.out.println(«while loop executed» + count +
«times»);
        }
}
```

Do-while

Конструкция ***do-while loop*** работает подобно while-loop за исключением того, что в данной конструкции цикл выполняется по крайней мере один раз, даже если условие не является истиной.

Синтаксис do-while loop:

```
do{
        statements;
}  while(condition);
```

Здесь do loop сначала выполняет операторы тела цикла, а затем проверяет условие после каждого выполнения. Цикл выполняется до тех пор, пока значением условия является не нуль или True (истина).

Обратите внимание: в конструкции do-while loop, точка с запятой требуется после while(condition).

For loop

For – это компактная форма **while loop** которая объединяет инициализацию переменной, проверку условия и увеличения или уменьшение значения переменной для итерации в отдельном утверждении.

Синтаксис for loop:

```
for(var init ;   condition ;   incre or decre value of variable)
   Часть 1     Часть 2        Часть 3
```

Обратите внимание: в конструкции for loop, объявление и инициализация переменной могут быть выполнены одновременно.

Последовательность шагов, которые нужно соблюдать при выполнении конструкции for loop следующие:

1) инициализация переменных (Часть 1) и проверка условия перед выполнением цикла (Часть 2). Если условие истинно, то выполняется тело цикла for loop;

2) когда последнее утверждение for loop выполнено, тогда выполняется Часть 3, где значение переменной увеличивается или уменьшается. После этого условие проверяется снова. Если значением условия (condition) является истина (True), то

тело цикла for loop выполняется ещё раз. Инициализация переменной делается только один раз, и цикл выполняется, пока значением условия является истина.

Следующая программа демонстрирует использование for loop.

Пример:

```
class ForDemo
{
        public static void main(String args []){
                for(int i = 1; i <= 5; i + + )
                        System.out.printIn(«value of i is» +i);
        }
}
```

В вышеупомянутом примере объявлена переменная *i* и присвоено значение 1. Это значение проверяется условием (*i* <= 5). Так как условие истина, то *i* будет напечатано. При увеличении значения переменной *i*, снова проверяется условие *i* <= 5 и выполняется соответствующее действие. Так продолжается до тех пор, пока *i* меньше или равно 5. Когда *i* = 6, то цикл заканчивается, т. к. условие *i* <= 5 ложь. Изогнутые фигурные скобки для цикла можно опустить, если есть одна инструкция, которая будет выполняться в цикле.

В цикле мы можем сделать следующее:

- инициализировать более одной переменной;
- задать более одного условия;
- приращение или уменьшение более чем одной переменной.

Следующий фрагмент кода показывает применение в цикле более одной переменной.

```
for(int i=l,j=5;i < 5 && j >= l;i++,j--)
System.out.println( i is   + i +   j is   +j);
```

В цикле for мы можем опустить любую часть, только помещая точку с запятой (;) Например,

```
int k =10;
for(; к <= 15; к++)  {
statementl;
statement2;
statementN;
          }
```

Вложенность циклов for:

```
for(int a = 1; a <= 2; a++){
for(int b = 1; b <= 3; b++){
statementl;
statement2;
statementN;
              }
      }
```

В вышеупомянутом примере, после инициализации переменной внешнего цикла, проверяется условие, данное во внешнем цикле, если условие истинно, то выполняется внутренний цикл for. Внутренний цикл for выполняется, пока условие истинно, и как только условие оценивается как ложь, начинает выполняться снова внешний цикл for. Внешний цикл выполняется, пока условие истинно. Здесь каждый раз управление передается от внешнего во внутренний цикл.

Обратите внимание: когда мы объявляем переменную в цикле for, область действия данной переменной только внутри цикла. В результате мы не можем использовать эту переменную вне цикла.

Оператор break прекращает выполнение оператора цикла и передает управление следующему за ним (за циклом) оператору.

Оператор continue позволяет в любой точке тела цикла прервать текущую итерацию и перейти к проверке условий продолжения цикла, определенных в предложениях for или while. В результате выполнение цикла заканчивается или начинается новая итерация.

2.1.8 Работа с массивами в Java

Массив – это список значений переменных с одинаковым именем, для обращения к значениям которых используется индекс или список индексов. Как только размер массива объявлен, то он не может быть изменен. Массивы полезны, поскольку один и тот же тип данных может быть сохранен в одном месте. Данными могут быть примитивный тип данных или объект. К индивидуальным элементам массива можно обратиться с помощью имени массива и местоположения данного элемента. Местоположение также называют *индексом*.

Массив может быть объявлен и представлен в памяти, как показано ниже

char ch[] = new char[10].

Вышеупомянутая запись создает массив 10 символов с именем ch.

Этот массив представлен следующим образом:

1) все элементы массива ch сохранены последовательно;

2) если мы хотим обратиться к символу, сохраненному в положении 6 массива ch, и назначить ему значение символьной переменной, просто пишем:

char element = ch[5];

3) если мы хотим обратиться к 5-му элементу массива ch и назначить ему значение переменной, просто пишем:

char element = ch[4];

4) здесь 5 и 4 – индексы массива ch. Индекс начинается с 0. Следовательно, чтобы обратиться к 5-му символу, индекс должен быть 4.

Рассмотрим следующие различные способы создания массивов:

char ch[]; // just a declaration.

Здесь длина массива не определена.

char ch[] = new char[10]; // declaration and creation.

Ключевое слово «new» используется для распределения памяти для массива. Первая запись (выше) объявляет массив, а вторая запись объявляет и создает массив.

Создание массива означает, что выделяется фактическая память для сохранения элементов массива. Это также называется реализацией массива. Когда массив создается, всем элементам массива назначаются значения по умолчанию в зависимости от типа массива. Если массив типа int, то всем элементам массива по умолчанию присваивается нуль. Если массив имеет переменную типа boolean, то все элементы имеют значение false. Это называется *автоинициализация*.

Следующая программа показывает пример использования массива.

Пример:

```
class ArrayDemo {
        public static void main(String args[])  {
                int integer [] = new int [5] ;
                int a = 100;
                for(int i = 0; i < 5; i++)
                        integer[i] = a++;
                for(int i = 0; i < 5; i++)
                        System.out.print(integer[i] + «\t»);
     }
 }
```

Вышеупомянутая программа создает массив, integer, типа int. Таким образом в массиве можно хранить целочисленные значения. Первый цикл for используется, чтобы сохранить значения в массив, а второй цикл for используется, чтобы напечатать все элементы массива.

Действующая инициализация

В этом типе элементы массива инициализированы во время объявления:

```
int a[] = {10,20,30,40,50};
float floatArray[] = {248.75,45.50,873.45}.
```

В действующей инициализации нет никакой потребности определять размер массива. Размер массива определен общим количеством значений, данных в пределах изогнутых фигурных скобок.

2.2 Контрольный пример выполнения лабораторной работы

Задание: написать программу, заполняющую целочисленный вектор случайными значениями в диапазоне от 100 до 200. Размер вектора 30 элементов. Вывести получившийся вектор на экран. Выполнить сортировку вектора по возрастанию. Вывести результат сортировки на экран.

Напишем программу, задающую вектор случайным образом.

```
public class ControlStructures
{
        public static int SIZE = 30;
        public static int MIN = 100;
        public static int MAX = 200;

        public static void main(String[] args) {
        int[] values = new int[ControlStructures.SIZE];
```

```
        for (int i = 0; i < ControlStructures.SIZE; i++)
        {
values[i] =ControlStructures.MIN+ (int)Math.round(Math.random()*
(ControlStructures.MAX -ControlStructures.MIN));
                System.out.print(values[i] + "\t");
        }
   }
}
```

Вынесем код, относящийся к выводу результата (печать вектора на консоль), в отдельный метод.

```
public class ControlStructures
{
public static int SIZE = 30;
public static int MIN = 100;
public static int MAX = 200;

public static void main(String[] args)
{
        int[] values = new int[ControlStructures.SIZE];

        for (int i = 0; i < ControlStructures.SIZE; i++)
       {
        values[i] =ControlStructures.MIN +(int)Math.round(Math.random()
        * (ControlStructures.MAX  ControlStructures.MIN));
        }
        ControlStructures.printVector(values);
}

private static void printVector(int[] vector)
{
        for (int i = 0; i < vector.length; i++) {
                System.out.print(vector[i] + "\t");}
}
}
```

Так как метод вывода вектора на экран будет вызываться из статического метода, то метод **printVector** также объявлен статическим.

Метод printVector будет вызываться **только внутри класса** ControlStructures, поэтому его можно объявить как **private,** тем самым понизив его область видимости.

Добавим метод, осуществляющий **сортировку**. Он также будет **static** и **private**.

```
public class ControlStructures
{
public static int SIZE = 30;
public static int MIN = 100;
public static int MAX = 200;

public static void main(String[] args)
```

```
{
        int[] values = new int[ControlStructures.SIZE];

for (int i = 0; i < ControlStructures.SIZE; i++)
{
values[i] =ControlStructures.MIN+ (int)Math.round(Math.random()*
(ControlStructures.MAX  - ControlStructures.MIN)         );
}
        System.out.println("Initial vector:");
        ControlStructures.printVector(values);

        int[] result = ControlStructures.sortVector(values);
        System.out.println("Sorted vector:");
        ControlStructures.printVector(result);
}

private static int[] sortVector(int[] vector)
{
        boolean flag;
        int temp;
        do {
                flag = false;
                for (int i = 0; i < vector.length  1; i++) {
                        if (vector[i] > vector[i+1])
{
                                temp = vector[i];
                                vector[i] = vector[i+1];
                                vector[i+1] = temp;
                                flag = true;
                        }
                }
        } while (flag);

        return vector;
}

private static void printVector(int[] vector)
{
        for (int i = 0; i < vector.length; i++)
{ System.out.print(vector[i] + "\t");}
                System.out.println();
}
}
```

Самостоятельно доработайте программу следующим образом:

1 Реализуйте метод **fillRandomIntVector()**, на вход которому подается размер вектора, максимальное и минимальное значения, которые могут принимать элементы вектора. Метод возвращает вектор, заполненный случайными целочисленными значениями.

2 Доработайте метод **printVector().** Добавьте к нему параметром **String message** сообщение, которое выводится перед распечаткой элементов одномерного целочисленного массива. Пример вызова:

ControlStructures.*printVector*(result, "Sorted vector:").

2.3 Содержание отчета

Отчет о выполнении лабораторной работы должен включать:

1 Теоретические сведения о типах данных Java и основных управляющих структурах.

2 Реализацию программы на языке Java в соответствии с индивидуальным заданием.

3 Листинг выполнения программы, отражающий все этапы ее выполнения.

4 Выводы о выполненной работе.

2.4. Контрольные вопросы

1 Приведите пример ключевых слов.

2 Приведите пример основных типов данных в порядке возрастания их размера.

3 Назовите символы Escape-последовательности отвечающие за перевод строки и за табуляцию.

4 Приведите несколько примеров правильных и не правильных идентификаторов.

5 Значение, в каком интервале возвращает метод Math.random() и какой метод отвечает за округление числа?

6 Зачем нужна управляющая конструкция switch?

7 Зачем нужны управляющие конструкции while loop, do while, for loop?

8 Как правильно определить двумерный массив?

2.5 Варианты заданий для самостоятельной работы

Задача 1. Определить одномерный массив и заполнить его случайными значениями:

1) Составить и вывести на экран новый массив с номерами элементов исходного массива, которые равны заданному значению. Заданное значение определяется константой;

2) Поменять местами максимальный и минимальный элементы массива. Вывести измененный массив на экран;

3) Все элементы массива, меньшие заданного значения, и их номера записать в новые массивы. Вывести новые массивы на экран. Заданное значение определяется константой;

4) Определить дополнительный массив разрешенных значений. Определить и вывести на экран, сколько элементов исходного массива имеют разрешенные значения;

5) Определить дополнительный массив разрешенных значений. Составить массив из элементов исходного массива, имеющих неразрешенные значения. Вывести результативный массив на экран;

6) Составить и вывести на экран массив с N максимальными значениями исходного массива. N определяется константой;

7) Переписать элементы массива в обратном порядке на том же месте. Вывести измененный массив на экран;

8) Определить дополнительный массив, состоящий из неповторяющихся элементов исходного массива и вывести его на экран;

9) Составить и вывести на экран массив номеров элементов исходного массива, встречающихся один раз;

10) Определить дополнительный массив, состоящий из повторяющихся элементов исходного массива и вывести его на экран.

11) Вычислить сколько элементов данного массива больше своего предыдущего элемента.

12) Найти максимальный элемент в одномерном массиве x. Затем каждый элемент в массиве разделить на максимальный элемент.

13) Дан массив b (n) . Переписать в массив C(n) положительные элементы массива b(n) умноженные на 5. (сжатие массива)

14) Дан одномерный массив a(n), в котором находится единственный нулевой элемент. Найти где он находится и вычислить сумму последующих за ним элементов. Выдать на экран номер элемента и сумму.

15) Определить сколько раз в этом массиве меняется знак. Например, в массиве 1,-34, 8, 14, -5, -8, -78 , 3 знак меняется 4 раза.

16) Найти минимальный элемент в одномерном массиве x. Затем каждый элемент в массиве умножить на минимальный элемент.

17) Дан массив c (n) . Переписать в массив x (n) все ненулевые элементы массива умноженные на 4. (сжатие массива)

18) Определить номера элементов = 5, количество положительных элементов для всего массива и произведение возведенных в квадрат отрицательных элементов.

19) Найти максимальный элемент и переставить его с 1-ым элементом массива.

20) Дан массив из 16 двоичных цифр (0;1). Определить сколько раз в этом массиве меняется число 0 на 1 или 1 на 0 . Например, в массиве 11110010001101 число меняется 6 раза.

21) Найти минимальный элемент в одномерном массиве x. Затем из каждого элемента массива вычесть минимальный элемент.

22) Дан массив c (n) . Переписать в массив x (n) все ненулевые элементы массива возведенные в квадрат.(сжатие массива)

23) Известно, что в массиве a(n) есть один элемент = 5. Найти где он находится и вычислить сумму элементов стоящих перед ним. (выдать на экран номер элемента и сумму).

24) Найти сумму элементов, стоящих между максимальным и минимальным элементами.

25) Дан массив a из n элементов. Вычислить сколько элементов данного массива больше в 3 раза своего предыдущего элемента.

26) Найти максимальный по модулю элемент в одномерном массиве x. Затем к каждому элементу массива прибавить этот максимальный элемент.

27) Дан массив x (n) . Переписать в массив y(n) отрицательные элементы массива x деленные на 2. (сжатие массива)

28) Известно, что в массиве a(n) есть один элемент = 5. Найти где он находится и вычислить произведение элементов стоящих перед ним. (выдать на экран номер элемента и сумму).

29) Дано два одномерных массива y(n) и x(n). Переставить между собой максимальный элемент из x(n) и минимальный из y(n) (то есть на место максимального из x(n) поставить минимальный из y(n) и наоборот.

30) Известно, что в массиве b(n) есть один отрицательный элемент. Определить где он находится и вычислить произведение элементов стоящих перед ним (выдать номер и произведени).

Задача 2.

1) Дан массив b (n) . Переписать в массив C(n) положительные элементы массива b(n) умноженные на 5 (*со сжатием, без пустых элементов внутри*) .Затем упорядочить методом «выбора и перестановки» по возрастанию новый массив.

2) Дан массив a (n) .Переписать в массив b(n) только положительные элементы массива a, умноженные на 3. (*со сжатием., без пустых элементов внутри*) . Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

3) Дан массив x (n) . Переписать в массив y(n) отрицательные элементы массива x умноженные на 2. (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

4) Дан массив b (n) . Переписать в массив C(n) отрицательные элементы массива b(n) умноженные на 4. (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

5) Дан массив b (n) . Переписать в массив C(n) положительные элементы массива b(n) деленные на 5. (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки» по возрастанию новый массив.

6) Дан массив a (n) .Переписать в массив b(n) только положительные элементы массива a, деленные на 3 (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

7) Дан массив x (n) . Переписать в массив y(n) отрицательные элементы массива x деленные на 2. (*со сжатием., без пустых элементов внутри*). Затем упорядочить по возрастанию новый массив.

8) Дан массив b (n) . Переписать в массив C(n) отрицательные элементы массива b(n). (*со сжатием., без пустых элементов внутри*) Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

9) Дан массив c (n) . Переписать в массив x (n) все ненулевые элементы массива умноженные на 4. (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

10) Дан массив c (n) . Переписать в массив x (n) все ненулевые элементы массива возведенные в квадрат. (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки» по возрастанию новый массив.

11) Дан массив c (n) . Переписать в массив x (n) все ненулевые элементы массива (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки» по возрастанию новый массив.

12) Дан массив c (n) .Переписать в массив x ненулевые элементы массива с разделенные на 5. (*со сжатием., без пустых элементов внутри*). Затем упорядочить методом «выбора и перестановки» по возрастанию новый массив.

13) Дан массив b (n) . Переписать в массив C(n) корни квадратные из положительных элементов массива b(n) деленные на 5. (*со сжатием., без пустых элементов внутри*) Затем упорядочить методом «выбора и перестановки» по возрастанию новый массив.

14) Дан массив b (n) . Переписать в массив C(n) корни квадратные из положительных элементов массива b(n) (*со сжатием., без пустых элементов внутри*) Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

15) Дан массив x (n) . Переписать в массив y(n) элементы массива x, большие 3. (*со сжатием., без пустых элементов внутри*) . Затем упорядочить методом «выбора и перестановки»по возрастанию новый массив.

16) Дан одномерный массив a(n), в котором находится единственный нулевой элемент. Найти где он находится, и упорядочить по возрастанию элементы, расположенные за ним. Выдать на экран номер элемента и упорядоченный массив.

17) Известно, что в массиве x(n) есть один элемент = 1. Определить где он находится , и упорядочить по убыванию элементы, расположенные за ним. Выдать на экран номер элемента и упорядоченный массив.

18) В массиве z(n) один отрицательный элемент . Найти где он находится , и упорядочить по возрастанию элементы, расположенные за перед ним. Выдать на экран номер элемента и упорядоченный массив.

19) Дан одномерный массив a(n), в котором находится единственный элемент равный 5. Найти где он находится , и упорядочить по убыванию элементы, расположенные перед ним. Выдать на экран номер элемента и упорядоченный массив.

20) Найти максимальный и минимальный элементы в одномерном массиве x, а также их порядковые номера. Затем упорядочить по возрастанию элементы, расположенные между максимальным и минимальным элементами.

21) Найти максимальный элемент и его порядковый номер в одномерном массиве x. Затем упорядочить по возрастанию элементы, расположенные после максимального элемента .

22) Найти минимальный элемент и его порядковый номер в одномерном массиве x. Затем упорядочить по убыванию элементы, расположенные после минимального элемента .

23) Найти максимальный элемент и его порядковый номер в одномерном массиве x. Затем упорядочить по возрастанию элементы, расположенные перед максимальным элемента .

24) Найти минимальный элемент и его порядковый номер в одномерном массиве x. Затем упорядочить по возрастанию элементы, расположенные перед минимальным элемента .

25) Дан одномерный массив a. Записать в массив z все порядковые номера элементов массива a равные 1. Затем упорядочить по возрастанию элементы массива a ,расположенные между двумя последними элементами равными 1.

26) Дан одномерный массив a. Записать в массив z все порядковые номера отрицательных элементов массива a . Затем упорядочить по убыванию элементы массива a ,расположенные между первыми двумя отрицательными элементами

27) Дан одномерный массив a. Записать в массив r все порядковые номера нулевых элементов массива a. Затем упорядочить по убыванию элементы массива a ,расположенные между двумя первыми нулевыми элементами.

28) Дан одномерный массив a. Записать в массив z все порядковые номера элементов больших 1 массива a . Затем упорядочить по возрастанию элементы массива a ,расположенные между любыми двумя элементами большими 1 .

29) Дан одномерный массив a(n), в котором находится единственный элемент ,значение которого принадлежит интервалу от 2 до 5. Найти где он находится , и упорядочить по убыванию элементы, расположенные перед ним. Выдать на экран номер элемента и упорядоченный массив.

30) Найти минимальный элемент и его порядковый номер в одномерном массиве x. Затем упорядочить по возрастанию элементы, расположенные перед минимальным элемента .

Задача 3.

Определить матрицу (двумерный массив) и ее заполнить случайными значениями.

Построить вектор B, которой возвращает –

1) число неотрицательных элементов в *i*-й строке;
2) среднее арифметическое положительных элементов в *i*-м столбце;
3) минимальное значение в *i*-й строке;
4) максимальное значение в *i*-м столбце;
5) номер максимального значения в *i*-й строке;
6) номер минимального значения в *i*-м столбце;
7) число элементов *i*-й строки, значения которых меньше заданного значения;
8) значение элемента матрицы, не равное заданному значению;
9) равно 1, если значения элементов *i*-й строки упорядочены по возрастанию, и 0, в противном случае;
10) количество четных чисел в i-й строке.
11) сумму положительных элементов в каждом столбце матрицы.
12) произведение положительных элементов в каждом столбце матрицы.
13) количество положительных элементов в каждом столбце матрицы.
14) среднее арифметическое положительных элементов в каждом столбце матрицы
15) среднее геометрическое положительных элементов в каждом столбце матрицы
16) сумму отрицательных элементов в каждом столбце матрицы.
17) произведение отрицательных элементов в каждом столбце матрицы.
18) минимальный элемент в каждой строке матрицы. Затем каждую строку матрицы разделить на минимальный элемент строки.

19) минимальный элемент в каждой строке матрицы среди положительных элементов.

20) максимальных элемент в каждой строке матрицы среди отрицательных элементов.

21) минимальный элемент в каждом столбце матрицы. Затем каждый столбец матрицы умножить на минимальный элемент.

22) максимальных элемент в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить максимальный элемент строки.

23) минимальный элемент и его номер в каждой строке матрицы. Затем из каждого элемента каждой строки вычесть номер минимального элемента строки.

24) номер минимального элемента в каждой строке матрицы. Затем к каждому элементу каждой строки прибавить номер минимального элемента строки.

25) номер максимального элемента в каждом столбце матрицы. Затем каждый элемент каждого столбца умножить на номер максимального элемента столбца.

26) номер максимального элемента в каждой строке матрицы. Затем каждый элемент каждой строки разделить на номер максимального элемента строки.

27) среднеарифметический элемент в каждой строке матрицы среди положительных элементов.

28) среднеарифметический элемент в каждой строке матрицы среди отрицательных элементов.

29) среднеарифметический элемент в каждой строке матрицы. Затем каждую строку матрицы умножить на среднеарифметический элемент строки.

30) среднегеометрический элемент в каждом столбце матрицы. Затем из каждого элемента каждого столбца вычесть среднегеометрический элемент столбца.

Задача 4.

1. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти номер минимального элемента её побочной диагонали.

2. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти сумму номеров минимального и максимального элементов её побочной диагонали.

3. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти сумму номеров минимального и максимального элементов её главной диагонали.

4. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти произведение минимального и максимального элементов её главной диагонали. Затем умножить побочную диагональ на максимальный элемент главной диагонали.

5. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти количество положительных элементов её главной диагонали. Затем умножить побочную диагональ на найденное количество.

6. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее арифметическое положительных элементов её побочной диагонали.

7. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое положительных элементов её побочной диагонали.

8. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее арифметическое положительных элементов параллели главной диагонали, расположенной выше над диагональю.

9. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти минимальный элемент среди положительных элементов параллели главной диагонали, расположенной выше над диагональю/

10. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое отрицательных элементов параллели главной диагонали, расположенной под диагональю.

11. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти произведение отрицательных элементов параллели побочной диагонали, расположенной над диагональю,

12. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти максимальный элемент среди отрицательных элементов параллели побочной диагонали, расположенной над диагональю,

13. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти сумму положительных элементов параллели побочной диагонали, расположенной под диагональю (ниже побочной диагонали).

14. Дан двумерный массив A, размером (n×n)(или квадратная матрица A). Найти сумму и количество положительных элементов параллели побочной диагонали, расположенной под диагональю (ниже побочной диагонали). Затем каждый элемент побочной диагонали умножить на количество.

15. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее арифметическое положительных элементов параллели главной диагонали, расположенной выше над диагональю.

16. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое положительных элементов верхней треугольной матрицы, расположенной выше главной диагонали, исключая саму главную диагональ.

17. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее арифметическое положительных элементов, верхней треугольной матрицы, расположенной выше главной диагонали,

18. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее арифметическое положительных элементов, нижней треугольной матрицы, расположенной ниже главной диагонали, исключая саму главную диагональ.

19. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое положительных элементов, нижней треугольной матрицы, расположенной ниже главной диагонали, включая саму главную диагональ.

20. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти количество положительных элементов, нижней треугольной матрицы, расположенной ниже главной диагонали, включая саму главную диагональ.

21. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое элементов, нижней треугольной матрицы, расположенной ниже побочной диагонали, включая саму побочную диагональ.

22. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти количество нулевых элементов, нижней треугольной матрицы, расположенной ниже побочной диагонали, включая саму побочную диагональ.

23. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое отрицательных элементов, нижней треугольной матрицы, расположенной ниже побочной диагонали, исключая саму побочную диагональ.

24. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти количество и сумму отрицательных элементов, нижней треугольной матрицы, расположенной ниже побочной диагонали, исключая саму побочную диагональ.

25. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти количество элементов, равных заданному числу x и расположенных в верхней треугольной матрице, расположенной выше побочной диагонали, исключая саму побочную диагональ.

26. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти произведение элементов, расположенных в верхней треугольной матрице, расположенной выше побочной диагонали, включая саму побочную диагональ.

27. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти количество нулевых элементов, расположенных в верхней треугольной матрице, расположенной выше побочной диагонали, включая саму побочную диагональ.

28. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее арифметическое положительных элементов, верхней треугольной матрицы, расположенной выше главной диагонали,

29. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти произведение элементов, расположенных в верхней треугольной матрице, расположенной выше побочной диагонали, включая саму побочную диагональ.

30. Дан двумерный массив A, размером (n×n) (или квадратная матрица A). Найти среднее геометрическое положительных элементов верхней треугольной матрицы, расположенной выше главной диагонали, исключая саму главную диагональ.