Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Разработка кроссплатформенных приложений»

Инструкция

по выполнению лабораторной работы
«Разработка, отладка и испытание линейных алгоритмов и программ»

Минск 2025 г.

Лабораторная работа № 1

Тема работы: «Разработка, отладка и испытание линейных алгоритмов и программ»

1 Цель работы

Сформировать умения разрабатывать линейные программы.

2 Задание

Номер варианта соответствует вашему номеру по списку, если номер по списку больше последнего варианта, то необходимо начать подсчет варианта с начала (например, учащийся с номером по списку 22 должен выполнить 2 вариант).

Варианты заданий:

1. Дана длина ребра куба. Найти объем куда и площадь его боковой поверхности.
2. Даны длины рёбер параллелепипеда. Найти его объём.
3. Составить программу нахождения значения функции:

$$A = (3\sin \propto^2 + cos^{\beta}\,\beta)/\ln(\alpha + \beta).$$

4. Определить время падения камня на поверхность земли с высоты h.
5. Составить программу нахождения значения функции:

$$S = \frac{tgx}{3x + 2cosx^2} + ln\,x^2.$$

6. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
7. Треугольник задан координатами своих вершин. Найти периметр треугольника.
8. Вычислить высоту треугольника, опущенную на сторону a, по известным значениям длин его сторон a, b, c.
9. Вычислить объём цилиндра с радиусом основания r и высотой h.
10. Написать программу нахождения значений функции:

$$K = \frac{5 * 10^{-5} + 3x}{\ln x + x^2}.$$

11. Определить расстояние, пройденное физическим телом за время t, если тело движется с постоянным ускорением a и имеет в начальный момент времени скорость V_0.

12. Составить программу нахождения значения функции:

$$X = \frac{\arccos(S*T+0.3)}{2.7*10^5*a}$$, где a = 25; S = 0.00024; T = 15.

13. Вычислить площадь треугольника по формуле Герона, если заданы его стороны.
14. Определить координаты вершины параболы $y = ax^2 + bx + c$ $(a \neq 0)$. Коэффициенты a, b, c заданы.
15. Составить программу вычисления значений функций:

$$Y = \sqrt{\frac{sin^2 5x + 2.1}{x^3 + 2z^4}} + e^x.$$

16. По данным сторонам прямоугольника вычислить его периметр, площадь и длину диагонали.
17. Составить программу вычисления функции:

$$P = \frac{2}{1+c*e^{-m*t}}$$, где c = 0.5; m = $2*10^{-3}$; t = 0.16.

18. Составить программу нахождения значения функции:

$$K = 2cos \propto + tg\beta - sin^2\gamma.$$

19. Составить программу вычисления среднего арифметического и среднего геометрического четырёх чисел.
20. Даны длины рёбер параллелепипеда. Найти площадь боковой поверхности.

3 Оснащение работы

Задание по варианту, ЭВМ, среда разработки IntelliJ IDEA.

4 Основные теоретические сведения

Java — строго типизированный язык

Прежде всего, важно уяснить, что Java — строго типизированный язык. Действительно, в определенной степени безопасность и надежность Java-программ обусловлена именно этим обстоятельством. Давайте разберемся, что это означает. Во-первых, каждая переменная обладает типом, каждое выражение имеет тип, и каждый тип строго определен. Во-вторых, все присваивания, как явные, так и посредством передачи параметров в вызовах методов, проверяются на соответствие типов. В Java отсутствуют какие-либо средства автоматического приведения или преобразования конфликтующих типов, как это имеет место в некоторых языках. Компилятор Java проверяет все выражения и параметры на предмет совместимости типов. Любые

несоответствия типов являются ошибками, которые должны быть исправлены до завершения компиляции класса.

Элементарные типы

Java определяет восемь элементарных типов данных: byte, short, int, long, char, float, double и boolean.

Часто элементарные типы называют также простыми типами, и в этой книге мы будем использовать оба эти термина. Элементарные типы можно разделить на четыре группы.

✓ Целочисленные. Эта группа включает в себя типы byte, short, int и long, которые представляют точные целые числа со знаком.

✓ Числа с плавающей точкой. Эта группа включает в себя типы float и double, которые представляют числа, определенные с точностью до определенного десятичного знака.

✓ Символы. Эта группа включает в себя тип char, которая представляет символы символьного набора, такие как буквы и цифры.

✓ Булевские значения. Эта группа включает в себя тип boolean — специальный тип, предназначенный для представления значений типа истинно/ложно.

Эти типы можно использовать в том виде, как они определены, или же для создания собственных типов классов. Таким образом, они служат основой для всех других типов данных, которые могут быть созданы.

Элементарные типы представляют одиночные значения, а не сложные объекты. Хотя во всех других отношениях Java — полностью объектно-ориентированный язык, элементарные типы данных таковыми не являются. Они аналогичны простым типам, которые можно встретить в большинстве других не объектно-ориентированных языков. Эта особенность обусловлена стремлением обеспечить максимальную эффективность. Превращение элементарных типов в объекты привело бы к слишком большому снижению производительности.

Элементарные типы определены так, чтобы они обладали явной областью допустимых значений и математически строгим поведением. Языки вроде C и C++ допускают варьирование размеров целочисленных переменных в зависимости от требований среды выполнения. Однако Java отличается в этом отношении. В связи с требованием переносимости, предъявляемым к Java-программам, все типы данных обладают строго определенной областью допустимых значений. Например, независимо от конкретной платформы, значения типа int всегда являются 32-битными. Это позволяет создавать программы, которые гарантированно будут выполняться в любой машинной архитектуре без специального переноса. Хотя в некоторых средах строгое указание размера целых чисел может приводить к незначительному снижению производительности, оно абсолютно необходимо для обеспечения переносимости программ.

Рассмотрим каждый и типов данных.

Целочисленные значения

Java определяет четыре целочисленных типа: byte, short, int и long. Все эти типы представляют значения со знаком — положительные и отрицательные.

Ширина целочисленного типа представляет не занимаемый объем памяти, а скорее поведение, определяемое им для переменных и выражений этого типа. Среда времени выполнения Java может использовать любой размер, до тех пор, пока типы ведут себя объявленным образом. Как показано в таблице 1.1., ширина и область допустимых значений этих целочисленных типов изменяются в широких пределах.

Таблица 1.1. Ширина и область допустимых значений целочисленных типов

Имя	Размер	Область допустимых значений
long	64	от -9223372036854775808 до 9223372036854775807
int	32	от -2147483648 до 2147483647
short	16	от -32768 до 32767
byte	8	от-128 до 127

Типы с плавающей точкой

Числа с плавающей точкой, называемые также действительными числами, используются при вычислении выражений, которые требуют получение результата с точностью до определенного десятичного знака. Например, такие вычисления, как вычисление квад-1 ратного корня или трансцендентных функций вроде синуса или косинуса, приводят к результату, который требует применения типа с плавающей точкой. В Java реализован стандартный (в соответствии с IEEE-754) набор типов и операций с плавающей точкой. Существуют два вида типов с плавающей точкой: float и double, которые соответственно представляют числа одинарной и двойной точности. Их ширина и области допустимых значений описаны в таблице 1.2.

Таблица 1.2. Ширина и область допустимых значений типов с плавающей точкой

Имя	Ширина в битах	Приблизительная область допустимых значений
double	64	от 4.9e-324 до 1.8e+308
float	32	от 1.4e-045 до 3.4e+038

Символы

В Java для хранения символов используется тип данных char. Однако программистам на C/C++ следует помнить, что тип char в Java не эквивалентен типу char в C или C++. В C/C++ char — это целочисленный тип, имеющий ширину 8 битов. В Java это не так. Вместо этого в нем для представления символов используется Unicode. Unicode определяет международный набор символов, который может представлять все символы, присутствующие во всех

известных языках. Он представляет собой унифицированный набор десятков наборов символов, таких как латиница, греческий алфавит, арабский алфавит, кириллица, иврит, японские и тайские иероглифы и множество других. Поэтому для хранения этих символов требуется 16 битов. Таким образом, в Java тип char является 16-битным. Диапазон допустимых значений этого типа — от 0 до 65536. Не существует отрицательных значений типа char. Стандартный набор символов, известный как ASCII, содержит значения от 0 до 127, а расширенный 8-битный набор символов, ISO-Latin-1 — значения от 0 до 255. Поскольку язык Java предназначен для обеспечения возможности создания программ, применимых во всем мире, использование кодировки Unicode для представления символов вполне обосновано. Конечно, применение Unicode несколько неэффективно для таких языков, как английский, немецкий, испанский или французский, для представления символов которых вполне достаточно 8 битов. Но это та цена, которую приходится платить за переносимость программ во всемирном масштабе.

Использование переменных типа char демонстрирует следующая программа:

```
// Демонстрация использования типа данных char.
class CharDemo {
public static void main(String args[]) {
char chl, ch2;
chl = 88; // код переменной X
ch2 = 'Y';
System.out.print ("chl и ch2: ") ;
System.out.println(chl + " " + ch2);
```

Эта программа отображает следующий вывод:

```
chl и ch2: X Y
```

Обратите внимание, что переменной chl присвоено значение 88, являющееся значением ASCII (и Unicode), которое соответствует букве X. Как уже было сказано, набор символов ASCII занимает первые 127 значений набора символов Unicode. Поэтому, все "старые трюки", применяемые при работе с символами в других языках, будут работать и в среде Java.

Хотя тип char был разработан для хранения символов Unicode, его можно считать также целочисленным типом, пригодным для выполнения арифметических операций. Например, он позволяет выполнять сложение символов или уменьшать значение символьной переменной. Рассмотрим следующую программу:

```
// Символьные переменные ведут себя подобно целочисленным значениям.
class CharDemo2 {
public static void main(String args[]) {
char chl;
chl = 'X';
System.out.println("chl содержит " + chl);
```

```
chl++; // увеличение значения chl на единицу
System.out.println("chl теперь " + chl);
)
}
```

Эта программа генерирует следующий вывод:

```
chl содержит X chl теперь Y
```

Вначале программа присваивает переменной chl значение X. Затем она увеличивает значение переменной chl на единицу. В результате хранящееся в переменной значение Становится буквой Y — следующим символом в последовательности ASCII (и Unicode).

Булевские значения

Java содержит элементарный тип, названный boolean, который предназначен для хранения логических значений. Переменные этого типа могут принимать только одно из двух возможных значений: true (истинно) или false (ложно). Этот тип возвращается всеми операциями сравнения, подобными a < b. Тип boolean обязателен для использования также в условных выражениях, которые управляют такими управляющими операторами, как if и for.

Следующая программа служит примером использования типа boolean:

```
// Демострация использования значений типа boolean.
class BoolTest {
public static void main(String args[]) {
boolean b;
b = false;
System.out.println("b равна " + b) ;
b = true;
System.out.println("b равна " + b) ;
// значение типа boolean может управлять оператором
if
if(b) System.out.println("Это выполняется.");
b = false;
if(b) System.out.println ("Это не выполняется.");
// результат выполнения операции сравнения — значение
типа boolean
System.out.println ("10 > 9 равно " + (10 > 9) ) ;
```

Эта программа генерирует следующий вывод:

```
b равна false b равна
true Это выполняется.
10 > 9 равно true
```

В приведенной программе особый интерес представляют три момента. Во-первых, как видите при выводе значения типа Boolean методом println () на экране отображается строка "true" или "false". Во-вторых, самого по себе значения переменной типа boolean достаточно для управления оператором if. Вовсе не обязательно записывать оператор if так, как показано ниже:

```
if(b == true) ...
```

В-третьих, результат выполнения операции сравнения вроде < — значение типа boolean. Именно поэтому выражение 10 > 9 приводит к отображению строки "true". Более того, выражение 10 > 9 должно быть заключено в дополнительный набор круглых скобок, поскольку операция + обладает более высоким приоритетом, чем операция >.

Переменные

Переменная — основной компонент хранения данных в Java-программе. Переменная определяется комбинацией идентификатора, типа и необязательного начального значения. Кроме того, все переменные имеют область определения, которая задает их видимость для других объектов и время существования. Мы рассмотрим эти элементы в последующих разделах.

Объявление переменной

В Java все переменные должны быть объявлены до их использования. Основная форма объявления переменных выглядит следующим образом:

```
ТИП идентификатор [=значение] [,идентификатор [-значение] ...] ;
```

тип — это один из элементарных типов Java либо имя класса или интерфейса. (идентификатор — это имя переменной. Переменной можно присвоить начальное значение (инициализировать ее), указывая знак равенства и значение. Следует помнить, что выражение инициализации должно возвращать значение того же (или совместимого) типа, который указан для переменной. Для объявления более одной переменной указанного типа можно использовать список с разделителями-запятыми.

Несколько примеров объявления переменных различных типов приведено ниже. Обратите внимание, что некоторые объявления осуществляют инициализацию переменных.

```
int a, b, c; // объявление трех переменных типа int: a, b и c
int d = 3, e, f = 5; // объявление еще трех переменных типа int c
// инициализацией d и f
byte z = 22; // инициализация переменной z
double pi = 3.14159; // объявление приблизительного значения переменной pi
char x = 'x'; // присваивание значения 'x' переменной x
```

Выбранные имена идентификаторов очень просты и указывают их тип. Java допускает применение любого правильно оформленного идентификатора с любым объявленным типом.

Динамическая инициализация

Хотя в приведённых примерах в качестве начальных значений были использованы только константы, Java допускает динамическую инициализацию переменных посредством любого выражения, допустимого в момент объявления переменной.

Например, ниже приведена короткая программа, которая вычисляет длину гипотенузы прямоугольного треугольника по длинам катетов:

```
// Этот пример демонстрирует динамическую инициализацию.
class Dynlnit {
public static void main(String args[])
{ double a - 3.0, b = 4.0;
// динамическая инициализация переменной с
double c = Math.sqrt(a * a + b * b) ;
System.out.println("Гипотенуза равна " + c);
}
}
```

Эта программа объявляет три локальные переменные — a, b и c. Две первые, a и Ь, инициализируются константами. Однако третья, c, инициализируется динамически, принимая значение длины гипотенузы (в соответствии с теоремой Пифагора). Для вычисления квадратного корня аргумента программа использует встроенный метод Java, sqrt (), который является членом класса Math. В этом примере основной момент состоит в том, что в выражении инициализации можно использовать любые элементы, которые допустимы во время инициализации, в том числе вызовы методов, другие переменные или константы.

Арифметические операции

Арифметические операции используются в математических выражениях так же, как они применяются в алгебре. Арифметические операции перечислены в таблице. 1.3.

Таблица 1.3. Арифметические операции в Java

Операция	Описание
+	Сложение
-	Вычитание (также унарный минус)
*	Умножение
/	Деление
%	Деление по модулю
++	Инкремент
+=	Сложение с присваиванием
-=	Вычитание с присваиванием
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Деление по модулю с присваиванием
--	Декремент

Операнды арифметических операций должны иметь числовой тип. Арифметические операции нельзя применять к типам boolean, но их можно применять к типам char, поскольку в Java этот тип, по сути, является поднабором типа int.

```
// Демонстрация основных арифметических операций.
class BasicMath {
public static void main(String args[]) {
//   арифметические   операции   с   целочисленными
значениями
System.out.println("Целочисленная арифметика");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c — a;
int e = -d;
System.out.println("a = " + a);
System.out.printlnC'b = " + b) ;
System.out.println("c = " + c) ;
System.out.println("d = " + d);
System.out.println("e = " + e);
// арифметические операции со значениями типа double
System.out.println("\nАрифметика     с     плавающей
точкой");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc — a;
double de = -dd;
System.out.println("da = " + da);
System.out.println("db = " + db) ;
System.out.println("dc = " + dc) ;
System.out.println("dd = " + dd) ;
System.out.println("de = " + de) ;
}
}
```

При выполнении этой программы на экране отобразится следующий вывод:

```
Целочисленная арифметика
a = 2
b = 6
c = 1
d = -1
e = 1

Арифметика с плавающей точкой
da = 2.0
db = 6.0
```

```
dc = 1.5
dd = -0.5
de = 0.5
```

Операция деления по модулю

Операция деления по модулю, %, возвращает остаток операции деления. Эту операцию можно применять как к типам с плавающей точкой, так и к целочисленным типам. Следующий пример программы демонстрирует применение операции %:

```
// Демонстрация использования операции %.
class Modulus {
public static void main(String args[]) {
int x = 42;
double y = 42.25;
System.out.println("x mod 10 = " + x % 10) ;
System.out.println("y mod 10 = " + y % 10);
}
}
```

При выполнении эта программа генерирует следующий вывод:

```
x mod 10 = 2
y mod 10 = 2.25
```

Приоритеты операций

Приоритеты операций Java, от высшего к низшему, описаны в табл. 4.7. Обратите внимание, что в первой строке таблицы указаны элементы, которые, как правило, не считают символами операций: круглые и квадратные скобки и символ точки. С технической точки зрения они являются разделителями, но в выражениях они действуют подобно операциям. Круглые скобки используют для изменения порядка выполнения операций. Квадратные скобки служат для индексации массивов. А символ точки используется для разыменования объектов, и эта операция будет рассмотрена дальше.

```
Высший приоритет
( ) [ ] .
++ -- ~ !
* / %
+ -
>> >>> <<
== !=
&
^
|
&&
||
= op=
Низший приоритет
```

Использование круглых скобок

Круглые скобки повышают приоритет заключенных в них операций. Часто это необходимо для получения требуемого результата. Например, рассмотрим следующее выражение:

```
a » b + 3
```

Вначале это выражение добавляет 3 к значению Ь, а затем сдвигает значение а вправо на полученное количество позиций. То есть используя избыточные круглые скобки, это выражение можно было бы записать следующим образом:

```
a » (b + 3)
```

Однако если вначале нужно выполнить сдвиг значения а вправо на b позиций, а затем добавить 3 к полученному результату, необходимо использовать круглые скобки следующим образом:

```
(a » b) +3
```

Кроме изменения обычного приоритета операций, иногда круглые скобки можно использовать для облегчения понимания смысла выражения. Сложные выражения могут оказаться трудными для понимания. Добавление избыточных, но облегчающих понимание смысла выражения, круглых скобок может способствовать исключению недоразумений в будущем. Например, какое из следующих выражений легче прочесть?

```
a | 4 + c » b & 7
(a | (((4 + c) » b) & 7))
```

И еще один немаловажный момент: использование круглых скобок (избыточных или не избыточных) не ведет к снижению производительности программы. Поэтому добавление круглых скобок для повышения читабельности программы не оказывает на нее отрицательного влияния.

5 Порядок выполнения работы

1. Выделить ключевые моменты задачи;
2. Построить алгоритм и теоритическую объектную модель решения задачи;
3. Запрограммировать полученный алгоритм и объектную модель;
4. Провести тестирование полученной программы.

6 Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

_____

7. Контрольные вопросы и задания

1. Перечислите типы данных языка Java.
2. Назовите арифметические операторы языка Java.
3. Какие приоритеты у операторов сравнения? Как можно изменить приоритет вычислений?

8. Рекомендуемая литература

1. **Урванов, Ф. В.** Java. Состояние языка и его перспективы. / Ф.В. Урванов. - Санкт-Петербург : БХВ-Петербург, 2023. - 368 с.
2. **Копец Дэвид.** Классические задачи Computer Science на языке Java. - Санкт-Петербург : Питер, 2022. - 288 с
3. **Васильев А. Н.** Java. Объектно-ориентированное программирование: Учебное пособие / А.Н. Васильев. - Санкт-Петербург : Питер, 2021. - 400 с.
4. **Эванс Бенджамин**. Java для опытных разработчиков. 2-е изд. — (Серия «Библиотека программиста»). - Санкт-Петербург : Питер, 2024. - 736 с.
5. **Лой Марк.** Программируем на Java. 5-е межд. изд. . - Санкт-Петербург : Питер, 2023. - 544 с.
6. **Гетц Брайан**. Java Concurrency на практике. - Санкт-Петербург : Питер, 2021. - 464 с.