МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

«Арифметические основы компьютеров. Представление информации в ЭВМ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению расчетно-графического задания
по дисциплине «Информатика»

для студентов дневной и заочной форм обучения направлений
09.03.01 – Информатика и вычислительная техника и
27.03.04 – Управление в технических системах

Севастополь
2017

Volkova T.V., Vladimirova E.S., Chengar O.V.

УДК 519.6
«Арифметические основы компьютеров. Представление информации в ЭВМ»: методические указания к выполнению расчетно-графического задания по дисциплине «Информатика» для студентов дневной и заочной форм обучения направлений 09.03.01 – Информатика и вычислительная техника и 27.03.04 – Управление в технических системах/ Сост. Т.В. Волкова, Е.С. Владимирова, О.В. Ченгарь – Севастополь: Изд-во СевГУ, 2017. – 44 с.

Целью методических указаний является оказание помощи студентам при выполнении расчетно-графического задания по дисциплине «Информатика». Методические указания предназначены для студентов первого курса дневной и заочной форм обучения направлений 09.03.01 – Информатика и вычислительная техника и 27.03.04 – Управление в технических системах.

Задания составлены так, чтобы их выполнение позволило студентам закрепить знания по темам «Системы счисления», «Представление информации в компьютере. Типы и форматы данных», «Операции над данными простых типов». Для каждого из 14 заданий разработано 30 вариантов, даны ссылки на литературные источники, с которыми студент должен ознакомиться при подготовке к выполнению РГЗ, в приложении приведен пример решения заданий РГЗ. Вся совокупность заданий разбита на 3 части, чтобы студент мог выполнять их последовательно по мере ознакомления с материалом лекций по дисциплине.

Методические указания рассмотрены и утверждены на заседании кафедры информационных технологий и компьютерных систем 19.04.2017 г., протокол № 9.

Допущено учебно-методическим центром СевГУ в качестве методических указаний.

Рецензенты:

Апраксин Ю.К., доктор технических наук, профессор кафедры «Информационные технологии и компьютерные системы»,

Доронина Ю.В., доктор технических наук, профессор кафедры «Информационные системы»

СОДЕРЖАНИЕ

1. ОБЩИЕ ТРЕБОВАНИЯ К ВЫПОЛНЕНИЮ РАСЧЕТНО-ГРАФИЧЕСКОГО ЗАДАНИЯ

1.1. Цель работы

Целью выполнения расчетно-графического задания (РГЗ) является закрепление знаний по темам «Арифметические основы компьютеров. Системы счисления», «Представление информации в компьютере. Типы и форматы данных», «Операции над данными простых типов».

1.2. Требования к отчету

Отчет о выполнении РГЗ представляется в тетради в клетку или на листах формата А4. Первый лист отчета – титульный. Пример оформления титульного листа приведен в приложении А. Отчет должен содержать следующие разделы:

1) цель работы;

2) выполнение задания (в данном разделе должны быть приведены условия заданий согласно варианту и подробные решения этих заданий с пояснениями);

3) вывод.

2. ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА

Задания с 1 по 6 составляют первую часть РГЗ, предназначенную для закрепления знаний студентов по темам: «Арифметические основы компьютеров. Системы счисления», «Представление целых чисел в компьютере. Типы и форматы данных», «Операции над целочисленными данными».

Задания с 7 по 11 составляют вторую часть РГЗ. Они предназначены для закрепления знаний студентов по темам «Представление вещественных чисел в компьютере. Типы и форматы данных», «Операции над данными с плавающей точкой», «Представление двоичных векторов в компьютере. Типы и форматы данных», «Операции над двоичными векторами».

Задания с 12 по 14 составляют третью часть РГЗ, предназначенную для закрепления знаний студентов по темам «Представление символьных и логических данных в компьютере. Типы и форматы данных», «Операции над символьными и логическими данными», «Логические выражения», «Ошибки в программах, вызванные некорректным использование типов данных».

Перед выполнением первой части РГЗ студент должен изучить соответствующие темы [1] (с. 22-25), [2] (с. 6-24), [3] (с. 36-56, с. 61-62, с. 64-65), [4] (с. 53-56, с. 82-88).

С темами, рассматриваемыми во второй части РГЗ, можно ознакомиться в [2] (с. 25-32), [3] (с. 62-64, с. 74-77, с.88-97), [4] (с. 56-58, с. 88-97), [5].

Перед выполнением третьей части РГЗ рекомендуется ознакомиться с [4], с.58-63, с. 98-105, [6].

3. ВАРИАНТЫ ЗАДАНИЙ

Ниже приведены условия и варианты заданий (таблицы 3.1 – 3.6).

Номер варианта (НВ) вычисляется по следующему правилу:

НВ = АВ % 30,

где АВ – две последние цифры номера зачетной книжки студента,

% – операция получения остатка от целочисленного деления.

Если АВ = 00, студент выполняет вариант № 30.

Задание 1. Перевести целое число X из системы счисления с основанием P=10 в систему счисления с основанием Q. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда. Варианты задания приведены в таблице 3.1.

Таблица 3.1 – Варианты для задания 1

НВ	X (P=10)	Q	НВ	X (P=10)	Q
1	8427	3	16	5929	3
2	8427	4	17	5929	4
3	8427	5	18	5929	5
4	8427	6	19	5929	6
5	8427	7	20	5929	7
6	5156	3	21	7456	3
7	5156	4	22	7456	4
8	5156	5	23	7456	5
9	5156	6	24	7456	6
10	5156	7	25	7456	7
11	4981	3	26	6798	3
12	4981	4	27	6798	4
13	4981	5	28	6798	5
14	4981	6	29	6798	6
15	4981	7	30	6798	7

Задание 2. Перевести целое положительное число X из двоичной системы счисления в восьмеричную, шестнадцатеричную и десятичную системы счисления. Варианты задания приведены в таблице 3.2.

Задание 3. Перевести правильную дробь X из системы счисления с основанием P=10 в систему счисления с основанием Q с точностью $\varepsilon = Q^{-5}$. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда. Варианты задания приведены в таблице 3.3.

Таблица 3.2 – Варианты для задания 2

НВ	Двоичное число X	НВ	Двоичное число X	НВ	Двоичное число X
1	1010111101100101	11	1110000110101000	21	1010110000010100
2	1011011110011110	12	1001111101010010	22	1100011101000110
3	1100001110101101	13	1000101100111101	23	1101001011010110
4	1100001110101110	14	1110111101010111	24	1111000010100001
5	1011110001011110	15	1100111101111000	25	1110000100001001
6	1110010111010111	16	1010100110001110	26	1000111000101010
7	1010101101011101	17	1110011010010110	27	1010101010101010
8	1001001111000100	18	1101011100110001	28	1100110111101100
9	1000001111101111	19	1100111000010010	29	1110111011101110
10	1111011110100110	20	1110001100010011	30	1100111011111001

Таблица 3.3 – Варианты для задания 3

НВ	Число X (P=10)	Q	НВ	Число X (P=10)	Q
1	0,842	3	16	0,592	3
2	0,842	4	17	0,592	4
3	0,842	5	18	0,592	5
4	0,842	6	19	0,592	6
5	0,842	7	20	0,592	7
6	0,515	3	21	0,745	3
7	0,515	4	22	0,745	4
8	0,515	5	23	0,745	5
9	0,515	6	24	0,745	6
10	0,515	7	25	0,745	7
11	0,498	3	26	0,679	3
12	0,498	4	27	0,679	4
13	4981	5	28	6798	5
14	4981	6	29	6798	6
15	4981	7	30	6798	7

Задание 4. Представить десятичное отрицательное число X в машинном виде (формат short). Перевод в дополнительный код осуществить тремя способами. Варианты задания представлены в таблице 3.4.

Задание 5. Проимитировать действия АЛУ при выполнении фрагмента java-программы.

```
…
int a = x, b = y;
int c = a − b;
…
```

Варианты задания представлены в таблице 3.5. значения переменных **x** и **y** приводятся в десятичной системе счисления. В каждом варианте нужно решить три примера и обосновать результаты.

Таблица 3.4 – Варианты для задания 4

НВ	Десятичное число X	НВ	Десятичное число X	НВ	Десятичное число X
1	− 2753	11	− 12457	21	− 892
2	− 3257	12	− 675	22	− 5945
3	− 10476	13	− 3579	23	− 9145
4	− 847	14	− 15480	24	− 25478
5	− 1089	15	− 18453	25	− 19245
6	− 27483	16	− 11900	26	− 599
7	− 9654	17	− 1356	27	− 29341
8	− 8346	18	− 30234	28	− 498
9	− 998	19	− 7982	29	− 2005
10	− 5467	20	− 8631	30	− 3985

Таблица 3.5 – Варианты для задания 5

НВ	x	y	НВ	x	y	НВ	x	y
1	295	192	11	475	273	21	821	345
	−295	−192		−475	−273		−821	−345
	−20151	20456		−27643	11673		−28015	5094
2	348	256	12	749	531	22	1673	951
	−348	−256		−749	−531		−1673	−951
	20151	−20151		27643	−11673		28015	−5094
3	545	341	13	2639	1275	23	8531	5476
	−545	−341		−2639	−1275		−8531	−5476
	−15472	19345		−16345	17521		−4271	3005
4	948	721	14	3645	1368	24	5121	3114
	−948	−721		−3645	−1368		−5121	−3114
	15472	−19345		16345	−17521		4271	−3005
5	721	456	15	1629	985	25	898	769
	−721	−456		−1629	−985		−898	−769
	−21572	18300		−23476	14327		−10254	25700
6	458	276	16	2591	1983	26	2345	1234
	−458	−276		−2591	−1983		−2345	−1234
	21572	−18300		23476	−14327		10254	−25700
7	1503	642	17	539	295	27	3456	2457
	−1503	−642		−539	−295		−3456	−2457
	−15700	19365		−19340	16750		−12005	22900
8	1703	895	18	721	482	28	7541	5265
	−1703	−895		−721	−482		−7541	−5265
	15700	−19365		19340	−16750		12005	−22900
9	991	574	19	4973	3521	29	2356	1987
	−991	−574		−4973	−3521		−2356	−1987
	−29345	5700		−22956	12354		−25897	20431
10	875	628	20	3271	1654	30	971	492
	−875	−628		−3271	−1654		−971	−492
	29345	−5700		22956	−12354		25897	−19431

Задание 6. Проимитировать операцию сложения двоичных чисел с фиксированной точкой (тип int) на двоичном сумматоре. Объяснить знак результата, описать исключительную ситуацию. Перевести слагаемые из двоичной системы счисления в десятичную при помощи программы «Калькулятор» (<Вид> → <Программист>) и доказать, что сумма действительно выходит за границы диапазона int. Для отрицательных слагаемых предварительно получить их прямые коды (для этого использовать способ 3).

Варианты задания представлены в таблице 3.6.

Таблица 3.6 – Варианты для задания 6

НВ	Слагаемые
1	01101001 11000001 11000011 11000011, 00110001 00111111 00101110 00110010
2	11111100 00101010 11000011 11000111, 10000011 00111000 10100011 10100110
3	00001111 10110011 10011000 00001111, 01111010 01110111 00111100 10100111
4	10000001 10000011 00000000 11110011, 11111110 00111101 10100101 01011100
5	00110100 11110000 10100101 11001101, 01011101 00101111 00010001 10100111
6	11111111 11111111 01011100 10100001, 10000000 00000000 10011100 11111111
7	01000000 11110000 01011010 11101110, 00111111 00011000 01000000 01111110
8	11111111 11111111 00000001 11001100, 10000000 00000000 01010101 00111101
9	01111111 11110011 01010101 11111111, 00000000 00011100 11111100 11110111
10	11111000 11001100 11100011 11001110, 10000111 00001100 11100010 11111100
11	01101001 11000001 11000011 11000011, 00110001 00111111 00101110 00110010
12	11111100 00101010 11000011 11000111, 10000011 00111000 10100011 10100110
13	00001111 10110011 10011000 00001111, 01111010 01110111 00111100 10100111
14	10000001 10000011 00000000 11110011, 11111110 00111101 10100101 01011100
15	00110100 11110000 10100101 11001101, 01011101 00101111 00010001 10100111
16	11111111 11111111 01011100 10100001, 10000000 00000000 10011100 11111111

Продолжение таблицы 3.6

НВ	Слагаемые
17	01000000 11110000 01011010 11101110, 00111111 00011000 01000000 01111110
18	11111111 11111111 00000001 11001100, 10000000 00000000 01010101 00111101
19	01111111 11110011 01010101 11111111, 00000000 00011100 11111100 11110111
20	11111000 11001100 11100011 11001110, 10000111 00001100 11100010 11111100
21	01101001 11000001 11000011 11000011, 00110001 00111111 00101110 00110010
22	11111100 00101010 11000011 11000111, 10000011 00111000 10100011 10100110
23	00001111 10110011 10011000 00001111, 01111010 01110111 00111100 10100111
24	10000001 10000011 00000000 11110011, 11111110 00111101 10100101 01011100
25	00110100 11110000 10100101 11001101, 01011101 00101111 00010001 10100111
26	11111111 11111111 01011100 10100001, 10000000 00000000 10011100 11111111
27	01000000 11110000 01011010 11101110, 00111111 00011000 01000000 01111110
28	11111111 11111111 00000001 11001100, 10000000 00000000 01010101 00111101
29	01111111 11110011 01010101 11111111, 00000000 00011100 11111100 11110111
30	11111000 11001100 11100011 11001110, 10000111 00001100 11100010 11111100

Задание 7. Представить десятичные числа
а) в обычном формате с плавающей точкой [2, 3];
б) формате IEEE-754 [5].
Использовать 32-битные форматы. Проверить результаты путем обратных преобразований. Варианты задания представлены в таблице 3.7.

Задание 8. Промоделировать алгоритм сложения (вычитания) чисел с плавающей точкой C = A + B (C = A − B). Варианты задания представлены в таблице 3.8.

Задание 9. Выполнить поразрядные логические операции И, ИЛИ, исключающее ИЛИ, НЕ над логическими векторами A и B. Варианты задания представлены в таблице 3.9.

Задание 10. Дан фрагмент java-программы. Представить значение результатов операций сдвигов C1, C2, C3, C4, C5, C6 (в двоичном виде).

```
…
int A = a;  byte B = b;  byte D = d;
int C1 = A >> D;
int C2 = A >>> D;
int C3 = A << D;
byte C4 = (byte) (B >> D);
byte C5 = (byte) (B >>> D);
byte C6 = (byte) (B << D);
```

… Варианты задания представлены в таблице 3.10.

Задание 11. Разработать и промоделировать алгоритм выполнения заданной операции над двоичным вектором A. Представить соответствующий фрагмент java-программы. Варианты задания представлены в таблице 3.11.

Задание 12. В java-программе определены следующие переменные:

```
int a, b, c;
float d, e;
char ch1, ch2;
boolean f, f1, f2;
```

Определить значение переменной f при заданных значениях остальных переменных. Представить последовательность вычисления значения логического выражения, определяющего значение переменной f. Переписать выражение без лишних, по мнению студента, скобок. Обосновать. Что изменится, если в логическом выражении использовать удвоенные знаки логических операций (&& и ||)? Как значение переменной f будет представлено в компьютере?

Варианты для задания 12 приведены в таблице 3.12 (используются буквы только английского алфавита). С кодовой таблицей Unicode, используемой Java для представления символов, можно ознакомиться в [6].

Задание 13. Разработать фрагмент java-программы, вычисляющий значение символьной (тип char) переменной *a* в зависимости от значений символьных переменных *c* и *d*. В программе для присвоения значения переменной *a* использовать троичную условную операцию java. Привести тесты для проверки правильности работы программы. Представить символьные и соответствующие им двоичные значения переменных *c*, *d* и *a* для каждого теста. Варианты для задания 13 приведены в таблице 3.13. В таблице 3.14 приведены названия библиотечных функций java, которые студент может использовать при выполнении задания.

Задание 14. Указать, какие из операторов вызовут ошибки (связанные с некорректным использованием типов данных) в java-программе, а какие нет? Обосновать. Варианты для задания 14 приведены в таблицах 3.15, 3.16.

Таблица 3.7 – Варианты для задания 7

НВ	Число 1	Число 2	НВ	Число 1	Число 2
1	463.752	− 463.752	16	545.129	− 545.129
2	567.321	− 567.321	17	461.739	− 461.739
3	834.125	− 834.125	18	577.638	− 577.638
4	542.791	− 542.791	19	635.895	− 635.895
5	123.247	− 123.247	20	792.225	− 792.225
6	647.986	− 647.986	21	199.991	− 199.991
7	524.479	− 524.479	22	267.762	− 267.762
8	385.873	− 385.873	23	352.925	− 352.925
9	850.654	− 850.654	24	925.352	− 925.352
10	864.805	− 864.805	25	831.318	− 831.318
11	300.256	− 300.256	26	657.758	− 657.758
12	341.775	− 341.775	27	775.527	− 775.527
13	256.643	− 256.643	28	185.618	− 185.618
14	351.345	− 351.345	29	593.423	− 593.423
15	723.379	− 723.379	30	690.772	− 690.772

Таблица 3.8 – Варианты для задания 8

НВ	Числа A и B в формате с плавающей точкой	Операция
1	1 00000100 00010101110…0 0 11111111 10101000000…0	+
2	1 00000100 00010101110…0 0 11111111 10101000000…0	−
3	0 11101111 11101000000…0 1 11110011 01011110010…0	+
4	0 11101111 11101000000…0 1 11110011 01011110010…0	−
5	1 11110110 00101011100…0 0 11110010 11100100000…0	+
6	1 11110110 00101011100…0 0 11110010 11100100000…0	−
7	0 00010000 10001100000…0 1 00001100 01011011010…0	+
8	0 00010000 10001100000…0 1 00001100 01011011010…0	−
9	1 00011001 01111100000…0 1 00010111 00101110010…0	+
10	1 00011001 01111100000…0 1 00010111 00101110010…0	−
11	0 11100111 10110100000…0 1 11101010 00100111010…0	+

Продолжение таблицы 3.8

НВ	Числа A и B в формате с плавающей точкой	Операция
12	0 11100111 10110100000…0 1 11101010 00100111010…0	−
13	1 11001010 01011100100…0 1 11000101 01101000000…0	+
14	1 11001010 01011100100…0 1 11000101 01101000000…0	−
15	0 00010011 10010010010…0 1 00001110 00011100000…0	+
16	0 00010011 10010010010…0 1 00001110 00011100000…0	−
17	1 00000100 00010101110…0 0 11111111 10101000000…0	+
18	1 00000100 00010101110…0 0 11111111 10101000000…0	−
19	0 11101111 11101000000…0 1 11110011 01011110010…0	+
20	0 11101111 11101000000…0 1 11110011 01011110010…0	−
21	1 11110110 00101011100…0 0 11110010 11100100000…0	+
22	1 11110110 00101011100…0 0 11110010 11100100000…0	−
23	0 00010000 10001100000…0 1 00001100 01011011010…0	+
24	0 00010000 10001100000…0 1 00001100 01011011010…0	−
25	1 00011001 01111100000…0 1 00010111 00101110010…0	+
26	1 00011001 01111100000…0 1 00010111 00101110010…0	−
27	0 11100111 10110100000…0 1 11101010 00100111010…0	+
28	0 11100111 10110100000…0 1 11101010 00100111010…0	−
29	1 11001010 01011100100…0 1 11000101 01101000000…0	+
30	1 11001010 01011100100…0 1 11000101 01101000000…0	−

Таблица 3.9 – Варианты для задания 9

НВ	A, B	НВ	A, B	НВ	A, B
1	10100101 11010011	11	11100011 10001110	21	00100101 11001011
2	11100111 10101011	12	01100111 11001001	22	01111001 10100010
3	11000011 10011001	13	00110111 01100100	23	10000111 10011001
4	10111101 01001001	14	01011110 10010010	24	11001001 00011010
5	10101010 00011001	15	11110101 10010010	25	11100110 01101001
6	01010101 11100010	16	01100110 11101000	26	10011101 01101001
7	11001100 01010110	17	10010011 00111010	27	11100001 10001001
8	11110000 01010011	18	10111110 11000100	28	10001000 10101110
9	00001111 10101100	19	10110110 01100101	29	11011100 01010110
10	11001110 01001001	20	01101010 11001100	30	10100011 00110110

Таблица 3.10 – Варианты для задания 10

НВ	a	b	d
1	10011111 00110011 11001100 11110011	11001100	00000011
2	11110000 11001100 10000001 11110111	11101100	00000100
3	10000001 00110011 01111110 10001111	10001111	00000101
4	11000111 01011100 10101010 10010011	11001101	00000010
5	11100101 11011011 01010001 10000111	11001000	00000001
6	10110100 00001111 00001111 11000001	10000011	00000110
7	10101100 11010101 00100100 00011001	10011101	00000011
8	11110011 00110011 00001111 01100011	10111001	00000100
9	10101111 00011100 00011100 10000111	11010101	00000101
10	11010011 10111011 00100100 00001111	10110011	00000010
11	10011111 00110011 11001100 11110011	11001100	00000100
12	11110000 11001100 10000001 11110111	11101100	00000101
13	10000001 00110011 01111110 10001111	10001111	00000010
14	11000111 01011100 10101010 10010011	11001101	00000001
15	11100101 11011011 01010001 10000111	11001000	00000110
16	10110100 00001111 00001111 11000001	10000011	00000011
17	10101100 11010101 00100100 00011001	10011101	00000100
18	11110011 00110011 00001111 01100011	10111001	00000101

Продолжение таблицы 3.9

НВ	a	b	d
19	10101111 00011100 00011100 10000111	11010101	00000010
20	11010011 10111011 00100100 00001111	10110011	00000001
21	10011111 00110011 11001100 11110011	11001100	00000101
22	11110000 11001100 10000001 11110111	11101100	00000010
23	10000001 00110011 01111110 10001111	10001111	00000001
24	11000111 01011100 10101010 10010011	11001101	00000110
25	11100101 11011011 01010001 10000111	11001000	00000011
26	10110100 00001111 00001111 11000001	10000011	00000100
27	10101100 11010101 00100100 00011001	10011101	00000101
28	11110011 00110011 00001111 01100011	10111001	00000010
29	10101111 00011100 00011100 10000111	11010101	00000100
30	11010011 10111011 00100100 00001111	10110011	00000101

Таблица 3.11. – Варианты для задания 11

НВ	Количество байтов в двоичном векторе	Операция
1	1	Поменять местами тетрады
2	2	Поменять местами вторую и третью тетрады
3	2	Поменять местами первую и вторую тетрады
4	2	Поменять местами третью и четвертую тетрады
5	2	Поменять местами первую и третью тетрады
6	2	Поменять местами вторую и четвертую тетрады
7	3	Поменять местами первый и второй байт
8	3	Поменять местами второй и третий байт
9	3	Поменять местами первый и третий байт
10	3	Поменять местами первую и шестую тетрады
11	3	Поменять местами вторую и пятую тетрады
12	3	Поменять местами третью и четвертую тетрады
13	3	Поменять местами первую и третью тетрады
14	3	Поменять местами вторую и четвертую тетрады
15	3	Поменять местами третью и пятую тетрады
16	3	Поменять местами первую и четвертую тетраду
17	3	Поменять местами вторую и шестую тетрады
18	3	Поменять местами третью и шестую тетрады
19	4	Выполнить циклический сдвиг влево на 4 разряда
20	4	Выполнить циклический сдвиг вправо на 4 разряда
21	4	Выполнить циклический сдвиг влево на 8 разрядов
22	4	Выполнить циклический сдвиг вправо на 8 разрядов
23	4	Выполнить циклический сдвиг вправо на 16 разрядов

Продолжение таблицы 3.11

НВ	Количество байтов в двоичном векторе	Операция
24	4	Выполнить циклический сдвиг влево на 16 разрядов
25	4	Поменять местами первый и третий байт
26	3	Выполнить циклический сдвиг влево на 4 разряда
27	2	Выполнить циклический сдвиг вправо на 4 разряда
28	4	Выполнить циклический сдвиг влево на 8 разрядов
29	1	Выполнить циклический сдвиг вправо на 3 разряда
30	3	Выполнить циклический сдвиг вправо на 5 разрядов

Таблица 3.12. – Варианты для задания 12

НВ	a	b	c	d	e	ch1	ch2	f1	f2	f
1	12	5	7	1,27	2,25	'F'	'0'	false	true	(a>(b+c)) | (d<e) & (ch1<ch2) | f1| (!(f1==f2))
2	6	8	10	5,25	3.25	'M'	'N'	true	false	(a<(b+c)) | (d>e) & (ch1<ch2) | !f1 | (f1!=f2) | ch1==88
3	10	29	17	21,5	3,2	'%'	'%'	true	true	((a>b) | (a>c)) & !(d>e) & (ch1=ch2) | f1 & f2
4	10	35	80	2,58	9,36	'a'	'b'	false	false	(a==b) | (a<c) & (d>e+3) | (ch1=ch2+1) & !(f1| f2)
5	11	15	14	98,1	15,3	'S'	'Y'	true	false	((a==b) | (a>c)) & (d<e*e) & ((ch1<ch2) | (f1==!f2))
6	4	9	1	6,89	3,54	'S'	'Y'	true	true	(c<b+a) | (d>b+a) & !(e<b) | (ch1<ch2) &(f1!= f2)
7	5	5	9	3,14	2,04	'1'	'+'	false	true	(c>21) | (a<b) & !(f1!=f2) & (ch1+2!=ch2) | (e<d+b)
8	7	98	16	56,1	46,5	'V'	'v'	false	false	(c-b>0) & (c<a) | (d-e>10,5) & !(ch1=='V' & ch2=='v') & (f1==f2)
9	10	16	30	2,75	7,98	'№'	':'	true	false	(c-5>b-10) & (d-2*e>0) | (a==0) & (ch2>ch1) | !(f1 | f2)
10	-5	-8	19	-2,5	-7,6	'Q'	'L'	true	true	((a>b+d) | (e>c-a)) & ((ch1<=ch2) | f1) & !f2
11	2	15	3	51,2	32,2	'F'	'0'	true	false	(a>(b+c)) | (d<e) & (ch1<ch2) | f1| (!(f1==f2))
12	54	-5	15	8,25	5.25	'N'	'M'	false	true	(a<(b+c)) | (d>e) & (ch1<ch2) | !f1 | (f1!=f2)
13	99	-5	19	9,51	8,14	'%'	'*'	false	false	((a>b) | (a>c)) & !(d>e) & (ch1=ch2) | f1 & f2

Продолжение таблицы 3.12

НВ	a	b	c	d	e	ch1	ch2	f1	f2	f
15	17	50	-3	-8,1	-5,3	'Y'	'S'	true	false	((a==b) | (a>c)) & (d<e*e) & ((ch1<ch2) | (f1==!f2))
16	14	99	-1	96,8	-3,5	'D'	'E'	false	false	(c<b+a) | (d>b+a) & !(e<b) | (ch1<ch2) &(f1!= f2)
17	31	15	45	-3,1	-9,3	'A'	'C'	false	true	(c>21) | (a<b) & !(f1!=f2) & (ch1+2!=ch2) | (e<d+b)
18	37	98	16	53,1	40,5	'D'	'v'	false	false	(c-b>0) & (c<a) | (d-e>10,5) & !(ch1=='V' & ch2=='v') & (f1==f2)
19	10	31	56	2,72	1,15	'@'	'#'	false	true	(c-5>b-10) & (d-2*e>0) | (a==0) & (ch2>ch1) | !(f1 | f2) | ch2<80
20	-5	-8	19	-2,5	-7,6	'Q'	'L'	false	false	((a>b+d) | (e>c-a)) & ((ch1<=ch2) | f1) & !f2
21	12	5	7	1,27	2,25	'F'	'0'	false	true	(a>(b+c)) | (d<e) & ((ch1<ch2) | f1| (!(f1==f2)))
22	6	8	10	5,25	3.25	'M'	'N'	true	false	((a<(b+c)) | (d>e)) & ((ch1<ch2) | !f1 | (f1!=f2))
23	10	29	17	21,5	3,2	'%'	'%'	true	true	(a>b) | (a>c) & !(d>e) & ((ch1=ch2) | f1) & f2
24	10	35	80	2,58	9,36	'a'	'b'	false	false	((a==b) | (a<c)) & ((d>e+3) | (ch1=ch2+1)) & !(f1| f2) | ch1>85
25	11	15	14	98,1	15,3	'S'	'Y'	true	false	(a==b) | (a>c) & (d<e*e) & (ch1<ch2) | (f1==!f2)
26	4	9	1	6,89	3,54	'S'	'Y'	true	true	((c<b+a) | (d>b+a)) & !(e<b) | (ch1<ch2) | (f1!= f2))
27	5	5	9	3,14	2,04	'1'	'+'	false	true	(c>21) & (a<b) | !(f1!=f2) & (ch1+2!=ch2) & (e<d+b)
28	7	98	16	56,1	46,5	'V'	'v'	false	false	(c-b>0) | (c<a) & ((d-e>10,5) | !(ch1=='V' | ch2=='v')) & (f1==f2)
29	10	16	30	2,75	7,98	'№'	':'	true	false	(c-5>b-10) | (d-2*e>0) & ((a==0) | (ch2>ch1) | !(f1 | f2))
30	-5	-8	19	-2,5	-7,6	'Q'	'L'	true	true	(a>b+d) & (e>c-a) | (ch1>=ch2) & f1 | !f2

Таблица 3.13. – Варианты для задания 13

№ п/п	Правило для вычисления значения переменной *a*
1	a=c+d, если c – цифра, d – буква, или c – знак '&'; a=c+2, в противном случае.
2	a=c–d, если c – буква, d – буква, c>d; a=0424_{16}, в противном случае.
3	a=c, если c='A' или c='F' или c='a' или c='f ', d – цифра; a=d, в противном случае.
4	a=c+10, если c – знак '#' или '@', d – не буква; a='G', в противном случае.
5	a= Character.toUpperCase(d), если c='Y' или '!', d – буква или цифра; a=c, в противном случае.
6	a= Character.toUpperCase(c), если c='y' или 'n', d – не знак '%'; a=d, в противном случае.
7	a=c+20, если c='R' или 'r', d – не цифра и не буква; a=d;
8	a= Character.toLowerCase(c), если c – буква, d – цифра или буква или знак '*'; a=d++, в противном случае.
9	a= Character.toLowerCase(d), если c – цифра или знак '$', d – буква; a=c—;
10	a=d—; если c ='@', или c – буква, или d – не цифра; a=c—, в противном случае.
11	a='Y', если c – буква или цифра, d – не буква и не цифра и не знак '#'; a='N', в противном случае.
12	a= Character.toUpperCase(c), если c – буква, но c не является буквами 'A' или 'a', d ='%'; a=d, в противном случае.
13	a=c+d, если c – буква, но не 'R', d – цифра, но не '9'; a='7', в противном случае.
14	a='5', если c='9' или c='0', d – не буква и не '#'; a=c++, в противном случае.
15	a=c+d+1, если c – цифра, d – буква, или c – знак '@'; a=c+2, в противном случае.
16	a=c−d, если c – цифра, d – цифра, c>d; a=0426_{16}, в противном случае.
17	a=c, если c='B' или c='b' или c='F' или c='f ', d – цифра; a=d, в противном случае.
18	a=c+15, если c – знак '&' или '?', d – буква; a='Q', в противном случае.
19	a= Character.toUpperCase(c), если c='y' или 'n', d –цифра; a=d, в противном случае.

Продолжение таблицы 3.13

№ п/п	Правило для вычисления значения переменной *a*
20	a= Character.toUpperCase(d), если d='f' или 'm', c – буква или цифра; a=c, в противном случае.
21	a=c+30, если c='L' или 'l', d – не цифра и не буква; a=d;
22	a= Character.toLowerCase(c), если c – буква, d – не цифра и не буква и не знак '*'; a=d++, в противном случае.
23	a= Character.toLowerCase(d), если c – не цифра или не знак '%', d – буква; a=c––;
24	a=d––; если c ='*', или c – буква, или d – цифра или буква; a=c––, в противном случае.
25	a='$', если c – буква или цифра, d – не буква и не цифра и не '#'; a='!', в противном случае.
26	a= Character.toUpperCase(c), если c – буква, но c не является буквами 'F' или 'f', d ='?'; a=d+5, в противном случае.
27	a=c+d+5, если c – буква, но не 'Z', d – цифра, но не '0'; a=d, в противном случае.
28	a=c, если c='1' или c='3', d – не буква, или c=d; a=d++, в противном случае.
29	a=Character.toLowerCase(d), если c – не цифра, d – буква, или c – знак '+'; a=c+d, в противном случае.
30	a=c+d, если c – не буква, d – не буква и не цифра, или d='@'; a=0416_{16}, в противном случае.

Таблица 3.14. – Функции Java для работы с символами, определенные в библиотечном классе Character

№	Тип возвра-щаемого значения	Спецификация вызова функции	Тип аргу-мента x	Описание
1	char	Character. toUpperCase (x)	Char	Возвращает символ x в верхнем регистре (большую букву)
2	char	Character. toLowerCase (x)	Char	Возвращает символ x в нижнем регистре (маленькую букву)
3	boolean	Character. isLetter (x)	Char	Возвращает true, если значение x – буква, иначе false
4	boolean	Character. isDigit (x)	Char	Возвращает true, если значение x – цифра, иначе false
5	boolean	Character. isLetterOrDigit(x)	Char	Возвращает true, если значение x – буква или цифра, иначе false

Таблица 3.15 – Варианты для задания 14

НВ	Номера операторов (последовательностей операторов) из таблицы 3.16
1	1, 2, 11, 12, 21, 22, 23, 26, 35, 43, 44
2	3, 4, 13, 14, 27, 28, 37, 40, 41, 44, 47
3	5, 6, 15, 16, 29, 30, 45, 46, 49, 51, 54
4	8, 9, 17, 18, 23, 24, 25, 29, 32, 41,48
5	10, 11, 19, 20, 31, 34, 37, 38, 39, 45, 50
6	3, 5, 9, 10, 18, 21, 30, 41, 42, 51, 53
7	4, 7, 12, 17, 20, 23, 32, 33, 35, 26, 52
8	1, 7, 8, 16, 19, 28, 31, 36, 37, 43, 40
9	10, 11, 18, 29, 30, 35, 44, 45, 47, 51, 54
10	13, 14, 19, 20, 23, 25, 31, 32, 41, 46, 49
11	2, 8, 9, 13, 14, 24, 27, 35, 36, 37, 39
12	1, 4, 9, 10, 15, 29, 34, 38, 50, 51, 53
13	8, 11, 14, 23, 26, 27, 30, 33, 35, 36, 42
14	10, 14, 15, 27, 31, 32, 40, 45, 48, 52
15	1, 6, 9, 16, 27, 32, 43, 45, 50, 51, 54,
16	3, 10, 15, 18, 21, 23, 25, 32, 41, 48, 47
17	11, 13, 14, 27, 30, 33, 36, 37, 39, 49, 50
18	5, 6, 14, 19, 24, 31, 34, 41, 44, 51, 53
19	7, 8, 11, 16, 19, 23, 26, 36, 38, 45, 46
20	3, 4, 9, 16, 17, 32, 37, 40, 41, 42, 48
21	5, 10, 13, 18, 29, 33, 34, 50, 51, 52, 54
22	2, 7, 10, 20, 21, 23, 25, 29, 30, 35, 43
23	6, 9, 15, 22, 31, 32, 33, 37, 39, 46, 47
24	5, 10, 11, 14, 21, 28, 36, 45, 49, 51, 53
25	8, 11, 17, 22, 23, 24, 26, 29, 33, 36, 50,
26	9, 16, 19, 29, 30, 37, 38, 40, 41, 44, 48
27	4, 7, 16, 17, 27, 30, 35, 42, 50, 51, 54
28	5, 6, 15, 23, 28, 29, 35, 36, 48, 25, 52
29	2, 11, 14, 21, 28, 31, 37, 39, 41, 46, 47
30	5, 10, 13, 22, 33, 44, 45, 49, 50, 51, 53

Таблица 3.16 – Данные для вариантов задания 14

№	Операторы	№	Операторы	№	Операторы
1	int n = 0937;	5	double a = 5; float b =a;	9	short n = -33768;
2	float a = 2.5f;	6	float a = 2.5f;	10	boolean f = 'A' > 'B';
3	int m = 0x9ag;	7	float a = 3.5f; long i = a;	11	short n = 32768;
4	double b = 2.5;	8	double b = 23.568E15;	12	boolean f = 23 < 15;

Продолжение таблицы 3.16

№	Операторы	№	Операторы	№	Операторы
13	int n = 0x7FFFFFFFFFFFFFFL;	27	byte b = 0x3F; b = b >> 3;	41	byte c= 300;

14	char ch1 = 'A';
	char ch2 = 'B';
	char ch3 = (char) (ch1+ch2);
		28	float a = (float) 2.5;	42	boolean f1 = true;
					boolean f2 = false;
					boolean f3 = f1==f2;

15	long n = 0x7FFFFFFFFFFFFFF;	29	short a = 3678;
			a = a / 3;
				43	boolean f1 = true;
					boolean f2 = false;
					boolean f3 = f1<f2;

16	int a = 127; byte b = a;	30	double b = 927.325E-3;	44	double b = 3.5D;

17	int n = 7FFFFFFFFF;	31	char ch1 = 'A';
			char ch2 = 'B';
			char ch3 = ch1+ch2;
				45	int a = 1, b = 2;
					double c = 2.5;
					double d = a / b + c;

18	int a = 57365; long b = a;	32	double b = 1024.75d;	46	float a = 25,349F

19	long n = 0x7fffffffffffffffffL;	33	int b = 4147483647;	47	double a = 3.5;
					double b = 2.5;
					double c = 1/(a+b);

20	int n = 0354;	34	double a = 127.351; float b = (float) a;	48	float a = 357.52; long n = (float) a;

21	float a = 2.5;	35	int s = -2500483648;	49	double a = 123.5;
					double b = -912.5;
					double c = 1/2*(a+b);

22	short m = -32768;	36	byte b = 0x3f; b = (byte) (b << 4);	50	short a = 3678; a = (short) a / 3;

23	double a = 2.0,
	b = 3.0, h = 0.1;
	double x, f = 0.0;
	for (x=a; x!=b; x=x+h)
	{y =x*x-7; …};
		37	double a = 2.0,
			b = 3.0, h = 0.1;
			double f = 0.0;
			double x = a;
			do { f = x * x ; …
			x = x + h; }
			while (x != b);
				51	double a = 2.0,
					b = 3.0, h = 0.1;
					double f = 0.0;
					double x = a;
					while (x != b)
					{ f = x * x ; …
					x = x + h; }

24	double a = 1, b = 2;
	double c = 2.5;
	double d = a / b + c;
		38	double a = 3.5;
			double b = 2.5;
			double c = 1.0/(a+b);
				52	double a = 123.5;
					double b = -912.5;
					double c = 1.0/2*(a+b);

25	int a=1, b=3, c=6; if (a+b+c)…	39	double a =1.2, b=3.5; if ((a+b) & (a-b))…	53	char d = 80, e = 'A'; if ((d+1) | (e-1))…

26	int a=1, b=3, c=6;
	if ((a>b) & (b<c))…
		40	double a =9.7, b=-3.6;
			if ((a+b) > 0 &
			(a-b) < 0)…
				54	char d = 88, e = 'V';
					boolean f = true;
					if ((d<e) | f)…

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1. Контрольные вопросы к первой части РГЗ

1) Какие системы счисления называют позиционными весомозначными? Приведите примеры.
2) Какие цифры существуют в четверичной, шестеричной, восьмеричной, шестнадцатеричной системах счисления? Сколько весит старший разряд 5-значного пятеричного числа?
2) По какому правилу осуществляется перевод целого числа из системы счисления с основанием P в систему счисления с основанием Q?
3) По какому правилу осуществляется перевод правильной дроби из системы счисления с основанием P в систему счисления с основанием Q?
4) Как перевести число из системы счисления с основанием P в десятичную систему счисления? Проиллюстрируйте применение метода составления степенного ряда на примере перевода чисел $23FD,B9_{16}$; $753,672_8$; $101001,101011_2$ в десятичную систему счисления.
5) Сколько двоичных разрядов (битов) соответствует восьмеричной цифре и почему? Назовите двоичные эквиваленты восьмеричных цифр.
6) Сколько двоичных разрядов соответствует шестнадцатеричной цифре и почему? Назовите двоичные эквиваленты шестнадцатеричных цифр.
7) Как осуществляется перевод двоичного числа в восьмеричную систему счисления и обратно?
8) Как осуществляется перевод двоичного числа в шестнадцатеричную систему счисления и обратно?
9) Какой общий формат используется для представления целых чисел в ЭВМ?
10) Чем отличаются типы byte, short, int, long, используемые для представления целых чисел в Java.
11) Как определить границы диапазона того или иного типа, используемого для представления целых чисел? Приведите примеры.
12) Каким образом представляются в ЭВМ целые отрицательные числа?
13) Чему равен дополнительный код положительного числа.
14) Дайте определение дополнительного кода отрицательного числа и опишите три способа его получения.
15) Как из дополнительного кода отрицательного числа получить его прямой код?
16) Как имея машинное представление положительного числа получить машинное представление противоположного ему отрицательного числа? Получите прямой код числа 678. На основе этого представления получите дополнительный код числа −678 (используйте третий способ получения дополнительного кода).
17) Почему можно сказать, что все целые числа (отрицательные и неотрицательные) представляются Java в дополнительном коде?
18) Как выглядит машинное представление левой границы диапазона для типов byte, int?

19) Как выглядит машинное представление правой границы диапазона для типов byte, int?
20) Представьте правую границу диапазона типа long (максимальное положительное целое число, используемое в ЭВМ) в виде восьмеричного и шестнадцатеричного числа. По какому правилу вы осуществили перевод из машинного представления?
20) Как выглядят числа 0 и −1 в формате для любого целого типа?
21) Какие арифметические операции определены для целых чисел? На какой операции базируются все остальные операции (поясните)? Какой операционный элемент используется для исполнения этой операции?
22) Почему не нужно проектировать электрическую схему двоичного вычитателя? На каком операционном элементе и как осуществляется вычитание целых чисел?
23) Сформулируйте правила двоичного сложения с учетом переноса. Что происходит со знаковыми разрядами при сложении на двоичном сумматоре?
24) Слагаемые поступают на двоичный сумматор в дополнительных кодах. В каком коде будет получен результат? Если результат отрицательный, как узнать его модуль?
25) Какая исключительная ситуация может возникнуть при сложении (вычитании) чисел с фиксированной точкой? Приведите примеры.

4.2. Контрольные вопросы ко второй части РГЗ

1) Какие форматы используются для представления вещественных чисел в ЭВМ?
2) Какое число с плавающей точкой считается нормализованным в обычном формате с плавающей точкой [2, 3]?
3) В чем состоят особенности формата IEEE-754 [5]? Какое число считается нормализованным для этого формата?
4) Как получить десятичное значение числа, представленного в формате IEEE-754?
5) Сколько разрядов у порядка и мантиссы в формате с одинарной точностью и в формате с двойной точностью?
6) Какую характеристику типа с плавающей точкой определяет число разрядов порядка?
7) Какую характеристику типа с плавающей точкой определяет число разрядов порядка?
8) Что определяет знак порядка?
9) Что определяет знак мантиссы?
10) Сформулируйте алгоритм сложения (вычитания) чисел с плавающей точкой.
11) Как вы думаете, почему порядки слагаемых выравниваются в сторону большего?

12) Назовите условие, которое должно выполняться для нормализованного числа (в формате, изображенном на рисунке 2.2), если его мантисса представлена в дополнительном коде? Назовите исключение из правила.
13) Как выполняется нормализация результата сложения чисел с плавающей точкой?
14) Как выполняются операции умножения и деления чисел с плавающей точкой.
15) Какие исключительные ситуации могут возникнуть при выполнении операций над числами с плавающей точкой?
16) Что такое двоичный вектор?
17) Какие типы данных используются для хранения двоичных векторов в Java?
18) Какие поразрядные логические операции определены для этих типов? Назовите правила их выполнения.
19) Какие операции сдвига определены в Java? Сколько операндов у операции сдвига и какие? В чем особенности выполнения логического сдвига вправо?
20) Почему для хранения двоичных векторов, длина которых меньше или равна 32 битам удобно использовать тип int?
21) Назовите элементарные задачи анализа и обработки двоичных векторов? Как они решаются?
22) Сформулируйте алгоритм выполнения перестановки групп битов.
23) Сформулируйте алгоритм выполнения циклического сдвига n-разрядного вектора на m разрядов влево (вправо), при условии m < n.

4.3. Контрольные вопросы к третьей части РГЗ

1) Для чего используется тип char. Сколько битов занимает значение типа char?
2) Что такое Unicode? Как представляется символ в Java? Сколько символов можно представить соответствующим образом?
3) Как можно задать символьный литерал в программе? Назовите 3 способа.
4) Как вывести код символа?
5) Какие операции определены для типа char? Приведите в качестве примера участок кода, требующий явного преобразования типа char в int и int в char.
6) По какому принципу выполняется сравнение символов? Чему равны значения выражений 'A'<'D' , 'C'>'F' , '9'<'A'. Какого они типа?
7) С какими методами, реализующими функции над символами и определенными в классе-оболочке Character, вы познакомились в ходе выполнения данной лабораторной работы? Значения каких типов они возвращают?
8) Для чего используется тип boolean? Какой класс оболочка ему соответствует в Java?
9) Какие значения может принимать тип boolean?
10) Чему равны значения выражений 2>5 , 3<10 ?
11) Какие операции определены для данных логического типа (boolean)?
12) Назовите правила выполнения операций булевой логики.
13) В чем состоит особенность выполнения укороченных операций && и || ?

14) Что из себя представляет условное выражение?
15) Приведите пример сложного условного выражения. Чему равно его значение?
16) Объясните синтаксис троичной условной операции. Приведите пример.
17) Какие ошибки в программах могут быть вызваны некорректным использованием типов данных? Как вы думаете, к каким типам относятся эти ошибки (ошибки, обнаруживаемые компилятором, ошибки времени выполнения, логические)?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грошев А.С. Информатика: учебник для вузов/ А.С. Грошев. – Архангельск: Арханг. гос. техн. ун-т, 2010. – 470 с.
2. Бойченко Л.П. Арифметические и логические основы компьютеров и дискретных автоматов: учеб. пособие/ Л.П. Бойченко, З.Х. Ягубов, Н.М. Выборова, Т.А. Туманова. – Ухта: УГТУ, 2011. – 100 с.
3. Апраксин Ю.К. Основы теории проектирования цифровых автоматов: Учеб. пособие для вузов/ Ю.К.Апраксин. – Севастополь: Изд-во СевГТУ, 2001. – 345 с.
4. Ноутон П. Java™ 2/ П. Ноутон, Г.Шилдт. – СПб.: БХВ-Петербург, 2008. – 1072 с.
5. Яшкардин В.Л. IEEE 754 − стандарт двоичной арифметики с плавающей точкой. – SoftElectro. Промышленная электроника и программирование [Электронный ресурс] http://www.softelectro.ru/ieee754.html
6. Таблица символов Юникода® [Электронный ресурс] − https://unicode-table.com/ru/#control-character.

ПРИЛОЖЕНИЕ А
Пример оформления титульного листа РГЗ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

Кафедра информационных технологий и компьютерных систем

ОТЧЕТ

о выполнении расчетно-графического задания

«Арифметические основы компьютеров.
Представление информации в ЭВМ»

по дисциплине «Информатика»

Вариант 15

Выполнил
студент группы ИВТ/б-11-о
Орлов И.В.
Проверил доцент Петров И.И.

Севастополь
2017

ПРИЛОЖЕНИЕ Б

Пример выполнения заданий РГЗ

Часть 1

Задание 1. Перевести целое число X из системы счисления с основанием P=10 в систему счисления с основанием Q. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда.

$X=236_{10}$, Q=4.

Перевод целого числа из системы счисления с основанием P в систему счисления с основанием Q осуществляется путем деления числа на Q, представленное в системе счисления с основанием P. В ходе деления вычисляется частное и остаток от деления, который и является очередной цифрой результата. Затем частное снова делится на Q. Процесс деления продолжается до тех пор, пока частное не станет равным нулю.

1) _ 2 3 6 : 4 = 5 9
2 0
_ 3 6
3 6
0

Определяем, сколько единиц в числе (вес цифры – 4^0)

3) _ 1 4 : 4 = 3
1 2
2

Вес цифры – 4^2

2) _ 5 9 : 4 = 1 4
4
_ 1 9
1 6
3

Определяем, сколько четверок в числе (вес цифры – 4^1)

4) _ 3 : 4 = 0
0
3

Вес цифры – 4^3

$X=236_{10}=3230_4$.

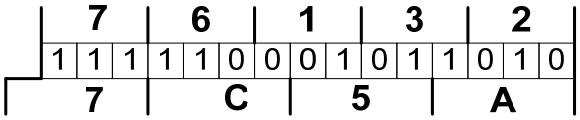
Проверка: $0\cdot 4^0+3\cdot 4^1+2\cdot 4^2+3\cdot 4^3=12+32+3\cdot 64=236$.

Задание 2. Перевести целое положительное число X из двоичной системы счисления в восьмеричную, шестнадцатеричную и десятичную системы счисления. $X=111110001011010_2$

$8=2^3$, поэтому восьмеричная цифра представляется тремя двоичными разрядами (с весами соответственно 4,2,1).

$16=2^4$, поэтому шестнадцатеричная цифра представляется четырьмя двоичными разрядами (с весами соответственно 8,4,2,1).

В десятичную систему из восьмеричной или шестнадцатеричной системы число переводим методом построения степенного ряда.



$$76132_8=2*8^0+3*8^1+1*8^2+6*8^3+7*8^4=31834$$
$$7C5A_{16}=10*16^0+5*16^1+12*16^2+7*16^3=31834$$

Задание 3. Перевести правильную дробь X из системы счисления с основанием P=10 в систему счисления с основанием Q с точностью $\varepsilon = Q^{-5}$. Сделать проверку путем обратного перевода в десятичную систему методом построения степенного ряда. $X=0{,}236_{10}$, Q=4.

Правило перевода правильных дробей из системы счисления с основанием P в систему счисления с основанием Q.

1) *Правильную дробь, записанную в системе счисления P, умножают на Q, записанное в системе счисления P.*
2) *Целую часть результата записывают как старшую цифру после запятой в системе счисления Q.*
3) *Дробную часть результата вновь умножают на Q.*
4) *Пункты 2 и 3 повторяют либо до получения нулевой дробной части, либо до достижения требуемой точности.*

0,		2	3	6
	x			4
0		9	4	4
	x			4
3		7	7	6
	x			4
3		1	0	4
	x			4
0		4	1	6
	x			4
1		6	6	4

Проверка:

$$0{,}03301_4=0\times4^{-1}+3\times4^{-2}+3\times4^{-3}+0\times4^{-4}+1\times4^{-5}=$$

$$=\frac{3\times4^3+3\times4^2+1}{4^5}=\frac{241}{1024}=0{,}235_{10}$$

Результат проверки подтверждает, что перевод приближенный.

Задание 4. Представить десятичное отрицательное число X в машинном виде (формат short языка Java). Перевод в дополнительный код осуществить тремя способами. X= -156_{10}.

Целое число в ЭВМ представляется n двоичными разрядами. Старший (самый левый) разряд – знаковый. У положительных чисел знаковый разряд равен 0, у отрицательных – 1. Число -156_{10} входит в диапазон формата short (от −32768 до 32767). Для числа в формате short n=16.

Java использует представление целых чисел в дополнительных кодах. Любой инверсный код (обратный или дополнительный) целого положительного числа равен его прямому коду.

Дополнительный код отрицательного получают как дополнение до границы диапазона (2^{n-1}). Это и есть первый способ получения дополнительного кода:

Знак=1,

Значение* = 2^{n-1} — |Значение|,

где* Значение* − *значение числа в дополнительном коде

1 способ. $2^{15} - 156 = 32768 - 156 = 32612$.

Переведем число 32612 в двоичную систему счисления. Для уменьшения числа делений, осуществим сначала перевод в восьмеричную систему, а затем, каждую восьмеричную цифру результата представим ее двоичным эквивалентом.

	3	2	6	1	2	|	8																
−	3	2	6	0	8	−	4	0	7	6	|	8											
					4		4	0	7	2	−	5	0	9	|	8							
										4		5	0	4		6	3	|	8				
														5		5	6		7				
																	7						

Результат: **1111111101100100** (знаковый разряд равен 1, т.к. число отрицательное).

2 способ. *Обратный код отрицательного числа получают из прямого путем инвертирования (изменения на противоположные) значений всех его разрядов, кроме знакового. Дополнительный код отрицательного числа получают путем прибавления 1 к обратному коду.*

Получим сначала прямой код −156. Для этого переведем число 156 в двоичную систему счисления:

	1	5	6	|	8							
−	1	5	2	−	1	9	|	8				
			4		1	6		2				
						3						

Результат перевода: 010 011 100.

Прямой код (n=16): 10000000 10011100 (знаковый разряд равен 1).
Обратный код: 11111111 01100011 (инверсия всех кроме знакового).
\+ 1
———————————
Дополнительный код: **11111111 01100100** (к обратному коду прибавили 1)

3 способ. Получаем дополнительный код из прямого: справа налево все разряды до первой единицы включительно оставляем без изменения, знаковый разряд оставляем без изменения, остальные разряды инвертируем.

10000000 10011100 → **11111111 01100100** .

Результаты перевода в дополнительный код, полученные первым, вторым и третьим способами совпали. Таким образом, число -156_{10} в формате short , будет выглядеть так:

1	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0

Задание 5. Проимитировать действия АЛУ при выполнении фрагмента java-программы.

```
…
int a = x, b = y;
int c = a − b;
…
```

В данном фрагменте java-программы определено три целочисленных переменных с именами a, b и c. Каждая из этих переменных занимает 4 байта в оперативной памяти (тип int).

*В Java рекомендуется выбирать 32-разрядный формат (**int**) для представления одиночных целых переменных, даже если их значения будут изменяться в диапазоне типа byte. Это обусловлено двумя причинами:*

во-первых, при выполнении действий над значениями коротких целых типов результат все равно всегда расширяется до int:

во-вторых, одиночные переменные и значения типа byte и short в Java могут на самом деле храниться в четырех байтах (для ускорения операций с ними), так что короткие типы характеризуют скорее «поведение», а не размер переменной (но если, например, создается массив из тысячи значений типа byte, то он будет занимать в памяти ровно 1000 байт).

АЛУ выполняет операции сложения и вычитания на двоичном сумматоре (двоичного вычитателя в АЛУ нет). Знаковые разряды слагаемых складываются на двоичном сумматоре, как и все остальные разряды. Т.к. слагаемые поступают на двоичный сумматор в дополнительных кодах, то и результат будет получен в дополнительном коде. Если результат положительный, то его дополнительный код равен прямому коду.

1) x = −7590, y = −4385;
c = a – b = x – y = −7590 − (−4385) = −7591 + 4385= − 3205;

Получим прямые коды слагаемых в 32-битном формате.

```
_ 7 5 9 0 |8          _ 9 4 8 |8          _ 1 1 8 |8          _ 1 4 |8
  7 5 8 4   _ 9 4 8     9 4 4   _ 1 1 8     1 1 2   _ 1 4       8   1
        6     9 4 4         4     1 1 2         6       8
              4               6             6
```

$(-7590)_{пр}$ = 10000000 00000000 00011101 10100110 ;
$(-7590)_{доп}$ = 11111111 11111111 11100010 01011010;

```
_ 4 3 8 5 |8          _ 5 4 8 |8          _ 6 8 |8          _ 8 |8
  4 3 8 4   _ 5 4 8     5 4 4   _ 6 8       6 4   _ 8       8   1
        1     5 4 4         4     6 4         4       8
              4               4             8
                              0
```

$(4385)_{пр}$ = $(4385)_{доп}$ = 00000000 00000000 00010001 00100001

Выполнение операции на двоичном сумматоре АЛУ:

a	+	11111111	11111111	11100010	01011010	$(-7590)_{доп}$
b		00000000	00000000	00010001	00100001	$(4385)_{доп}$
c		11111111	11111111	11110011	01111011	$(-3205)_{доп}$
		10000000	00000000	00001100	10000101	$(-3205)_{пр}$

Проверка: $5\cdot 8^0 + 0\cdot 8^1 + 2\cdot 8^2 + 6\cdot 8^3 = 5 + 128 + 3072 = 3205$

2) x = 7590, y = 4385;
c = a – b = x – y = 7590 − 4385 = 7591 + (− 4385) = 3205;
Далее пример решается аналогично предыдущему.

3) x = 26592, y = −19301;
c = a – b = x – y = 26592 − (−19301) = 26592 + 19301 = 45893;
Далее пример решается аналогично предыдущим.

Задание 6. Проимитировать операцию сложения двоичных чисел с фиксированной точкой (тип int) на двоичном сумматоре. Объяснить знак результата, описать исключительную ситуацию. Перевести слагаемые из двоичной системы счисления в десятичную при помощи программы «Калькулятор» (<Вид> → <Программист>) и доказать, что сумма действительно выходит за границы диапазона int. Для отрицательных

слагаемых предварительно получить их прямые коды (для этого использовать способ 3).

Слагаемые: 11111010 10001100 11101011 11001110,
10000101 01001100 11100010 11111100

Первое слагаемое:

11111010 10001100 11101011 11001110 – дополнительный код,

10000101 01110011 00010100 00110010 – прямой код.

В десятичной системе счисления это число –91427890.

Второе слагаемое:

10000101 01001100 11100010 11111100 – дополнительный код,

11111010 10110011 00011101 00000100 – прямой код.

В десятичной системе счисления это число –2058558724.

–91427890 + (–2058558724) = –2149986614.

Диапазон типа данных int:

от $-2^{31}=-2147483648$ *до* $2^{31}-1=2147483647$

Полученная сумма выходит за границы диапазона int, следовательно она не поместится в 32-разрядной сетке.

На двоичном сумматоре АЛУ будет получен следующий результат:

```
+  10000101 01001100 11100010 11111100
   11111010 10001100 11101011 11001110
-------------------------------------
   01111111 11011001 11001110 11001010
```

Знак суммы отличается от равных знаков слагаемых. Такая ситуация квалифицируется как переполнение с фиксированной точкой.

Переполнение – это исключительная ситуация, которая может возникнуть только при сложении чисел с одинаковыми знаками. Процессор реагирует на подобную ситуацию установкой соответствующего бита в регистре флагов прерываний.

Часть 2

Задание 7. Представить десятичные числа
а) в обычном формате с плавающей точкой;
б) формате IEEE-754.
Использовать 32-битные форматы. Проверить результаты путем обратных преобразований.
Числа: 326,527 и −326,527.

1) Обычный 32-разрядный формат с плавающей точкой: *старший бит – знак числа (мантиссы), 8 битов – порядок со знаком в дополнительном коде, 23 бита – мантисса (числовые разряды) в дополнительном коде.*

Переводим в двоичную систему отдельно целую и дробную часть числа 326,527.

–	3	2	6	|	8								5			0			6
	3	2	0	–	4	0	|	8			1	0	1	0	0	0	1	1	0
			6		4	0		5											
						0													

Чтобы представить число в формате с точностью 23 бита, нужно определить 8 восьмеричных цифр. 3 цифры получилось в целой части, значит в дробной надо определить 5.

0,		5	2	7
	x			8
4		2	1	6
	x			8
1		7	2	8
	x			8
5		8	2	4
	x			8
6		5	9	2
	x			8
4		7	3	6

101000110.100001101110100 =
=0.10100011010000110111010 0 *2^9

Получили мантиссу в нормализованном виде ($\frac{1}{2} \le |M| < 1$)

и порядок (9_{10})
Один лишний бит справа у нормализованной мантиссы нужно отбросить.

Число $326{,}527_{10}$ представляется в обычном 32-разрядном формате с плавающей точкой следующим образом:

0 00001001 10100011010000110111010

Проверка:

Значение числа равно $M \cdot 2^P$:

$$(2^{-1}+2^{-3}+2^{-7}+2^{-8}+2^{-10}+2^{-15}+2^{-16}+2^{-18}+2^{-19}+2^{-20}+2^{-22})\cdot 2^9 =$$

$$= 2^8+2^6+2^2+2^1+2^{-1}+2^{-6}+2^{-7}+2^{-9}+2^{-10}+2^{-11}+2^{-13} =$$

$$= 256+64+4+2+\frac{2^{12}+2^7+2^6+2^4+2^3+2^2+2^0}{2^{13}} =$$

$$= 326+\frac{4096+128+64+16+8+4+1}{8192} = 326+\frac{4314}{8192} = 326{,}5269 \approx 326{,}527$$

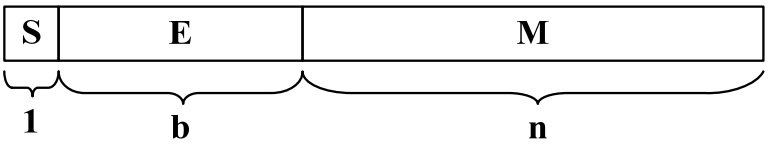
Для получения машинного представления числа $-326{,}527$ нужно получить дополнительный код мантиссы. Порядок будет тот же, что и у числа 326,527. Знаковый разряд числа равен 1.

0 00001001 10100011010000110111010 – +326,527

1 00001001 01011100101111001000110 – – 326,527

Мантисса представлена в дополнительном коде, т.к. она является отрицательным числом.

2) В формате IEEE-754 используются прямые коды порядка и мантиссы.
1 бит отводится под знак числа (*S*), b битов − под смещенный порядок (*E*) и n битов − под остаток от мантиссы (*M*):



Для 32-разрядного формата (тип float) b=8, n=23.
E (экспонента или смещенный порядок) − это целое положительное число без знака. Истинный порядок числа смещается, т.е. увеличивается на $2^{b-1}-1$, т.е. на 127. Это делается для того, чтобы число, представляющее порядок (*E*), было всегда положительным.

Двоичное число с плавающей точкой для формата IEEE-754 считается нормализованным, если 1<=Мантисса<2. Такая мантисса всегда начинается с 1 (имеет целую часть, равную 1). Нет смысла хранить эту единицу в отведенных под мантиссу n битах. Поэтому в поле *M* хранят только дробную часть − остаток от мантиссы (экономят один бит точности).

Формула вычисления значения десятичного числа, из значений, представленных в полях формата IEEE-754:

$$F = (-1)^s \cdot 2^{(E-2^{(b-1)}+1)} \cdot (1 + M/2^n)$$

При вычислении значения числа используется истинный, а не смещенный порядок $E-(2^{b-1}-1)$, к остатку от мантиссы прибавляется целая часть мантиссы, равная 1. Деление остатка от мантиссы *M* на 2^n обозначает отрицательные степени двойки, соответствующие весам разрядов *M*, например, $2^{-1}=1/2$.

Ранее получили двоичное представление числа 326,527:
$0.10100011010000110111010\underline{0}*2^9=$
$1.\underline{0100011010000110111010100}*2^8$ (последний бит уже не лишний!) – после точки записан остаток от мантиссы.

Экспонента (сдвинутый порядок):
$8+127=135_{10}=207_8=10000111_2$

_	1	3	5	|	8						2		0		7	
	1	2	8	_	1	6	|	8		0	**1**	**0**	**0**	**0**	**1**	**1**	**1**
			7		1	6		2								
						0										

Представление числа 326.527_{10} в формате float:
0 10000111 01000110100001101110100 (знак, смещенный порядок и остаток от мантиссы).

Проверка: $F=(-1)^0 2^{(135-127)}(1+(2^{21}+2^{17}+2^{16}+2^{14}+2^9+2^8+2^6+2^5+2^4+2^2)/2^{23})=$
$=2^8(1+1/2^2+1/2^6+1/2^7+1/2^9+1/2^{14}+1/2^{15}+1/2^{17}+1/2^{18}+1/2^{19}+1/2^{21})=$
$=2^8+2^6+2^2+2+1/2+1/2^6+1/2^7+1/2^9+1/2^{10}+1/2^{11}+1/2^{13}=$
$=326+(4096+128+64+16+8+4+2+1)/8192=326+4319/8192\approx$
$\approx 326.527_{10}$

Представление числа -326.527_{10} в формате float:
1 10000111 01000110100001101110100 (отличается от представления числа 326.527_{10} только значением знакового разряда).

Задание 8. Промоделировать алгоритм сложения (вычитания) чисел с плавающей точкой C = A + B (C = A − B).

A: 1 00000011 00011101100…0;
B: 0 11111111 10111000000…0;
операция – «–» (вычитание).

1) Проверка особых ситуаций: если мантисса одного из слагаемых равна нулю, то результат принимается равным другому слагаемому и выполнение алгоритма заканчивается.

Ни одна из мантисс слагаемых не равна нулю, следовательно, нудно перейти к следующему пункту алгоритма.

2) *Вычисляется разность порядков на сумматоре порядков(СП). Если разность порядков больше или равна числу разрядов мантиссы, то сумма принимается равной числу с большим порядком и выполнение алгоритма заканчивается.*

$P_A - P_B = P_A + (-P_B)$

Порядок B отрицательный (равен −1). Вычитание отрицательного числа эквивалентно сложению с положительным числом, имеющим такой же модуль. Для получения второго слагаемого сначала получим дополнительный код B: $\mathbf{(11111111)_{доп} = (\underline{1}0000001\underline{)}_{пр}}$. Затем изменим знак числа: **0**0000001.

СП

+	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**3**
	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
	0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**4**

3) *Мантисса числа с меньшим порядком сдвигается вправо на число разрядов, равное модулю разности порядков слагаемых.*

На СМП получена положительная разность порядков, следовательно, порядок A больше, чем порядок B. Для уравнивания порядков сдвигать вправо нужно мантиссу B.

Мантисса B сдвигается вправо на 4 разряда, освобождающиеся слева разряды доопределяются значением знака (знак мантиссы – это знак числа).

0 10111000000…0→ 0 00001011100…0;

4) Мантисса суммы (разности) вычисляется путем сложения (вычитания) мантисс слагаемых с помощью двоичного сумматора мантисс (СМ).

На сумматоре мантисс из мантиссы А вычитается сдвинутая мантисса В: $M_A - M_B = M_A + (-M_B)$. Знаки мантисс складываются так же, как и все остальные разряды.

M_B = 0 00001011100…0 (положительное число). Меняем знак и переводим в дополнительный код:

(1 00001011100…0)$_{пр}$ = (1 11110100100…0)$_{доп}$

	СМ												
+	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	… **0**
	1	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	… **0**
	1	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	… **0**

5) Порядок результата принимается равным большему из порядков слагаемых.

Порядок результата принимается равным большему из порядков слагаемых, т.е. порядку А: $P_C = P_A = 00000011$.

6) *Результат нормализуется (слагаемые представлены в нормализованном виде, результат также должен быть нормализован).*

В результате сложения на сумматоре мантисс переполнения не произошло (сумма имеет тот же знак, что и равные знаки слагаемых), значение знакового разряда суммы мантисс отличается от первого числового разряда, следовательно результат уже в нормализованном виде, процедуру нормализации проводить не нужно.

Окончательно получаем **C = 1 00000011 00010010000…0.**

Задание 9. Выполнить поразрядные логические операции И, ИЛИ, исключающее ИЛИ, НЕ над логическими векторами А и В.

A: 01010110; B: 10110010.

AND (&) $\wedge$

0	1	0	1	0	1	1	0
1	0	1	1	0	0	1	0
0	0	0	1	0	0	1	0

XOR (^) $\oplus$

0	1	0	1	0	1	1	0
1	0	1	1	0	0	1	0
1	1	1	0	0	1	0	0

OR (|) $\vee$

0	1	0	1	0	1	1	0
1	0	1	1	0	0	1	0
1	1	1	1	0	1	1	0

NOT (~) A

0	1	0	1	0	1	1	0
$\overline{A}$ 1	0	1	0	1	0	0	1

NOT (~) B

1	0	1	1	0	0	1	0
$\overline{B}$ 0	1	0	0	1	1	0	1

Задание 10. Дан фрагмент java-программы. Представить значение результатов операций сдвигов C1, C2, C3, C4, C5, C6 (в двоичном виде).

```
…
int A = a;  byte B = b;  byte D = d;
int C1 = A >> D;
int C2 = A >>> D;
int C3 = A << D;
byte C4 = (byte) (B >> D);
byte C5 = (byte) (B >>> D);
byte C6 = (byte) (B << D);
…
```

a: 10000111 01110011 11011100 11000011, b: 11101101, d: 00000100.

C1: 11111000 01110111 00111101 11001100
C2: 00001000 01110111 00111101 11001100
C3: 01110111 00111101 11001100 00110000

Результат любой арифметической и логической операции с byte- и short-операндами в Java имеет тип int.

При выполнении оператора **byte C4 = (byte) (B >> D);** сначала операнд B автоматически расширяется до int путем дублирования старшего разряда двоичного вектора (для чисел – это разряд знака) на 3 добавляемых байта. После этого производится сдвиг уже 32-битного вектора вправо на 4 разряда с доопределением освобождающихся слева разрядов значением старшего разряда. Затем производится «обрезка» полученного значения, в результате чего byte-переменной C4 присваивается младший байт результата операции сдвига.

Расширение B	**11111111 11111111 11111111 11101101**
B >> D	**11111111 11111111 11111111 11111110**
(byte) (B >> D)	**11111110**
C4:	**11111110**

Алгоритм выполнения оператора **byte C5 = (byte) (B >>> D);** аналогичен рассмотренному выше.

Расширение B	**11111111 11111111 11111111 11101101**
B >> D	**00001111 11111111 11111111 11111110**
(byte) (B >>> D)	**11111110**
C5:	**11111110**

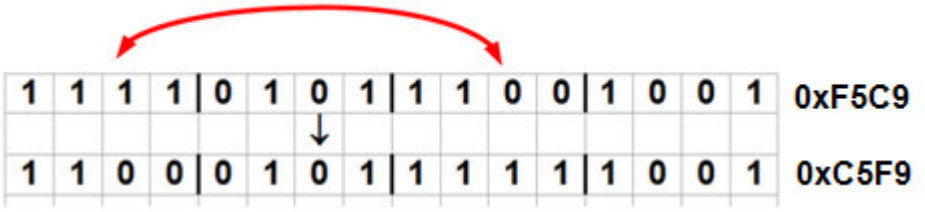
Показали, что на типах byte и short отличий в результатах операций « >> » и « >>> » не видно. Их можно увидеть только на более универсальном типе int.

Для операции **byte C6 = (byte) (B << D);** :

Расширение B	**11111111 11111111 11111111 11101101**
B << D	**11111111 11111111 11101110 11010000**
(byte) (B >>> D)	**11010000**
C6:	**11010000**

Задание 11. Разработать и промоделировать алгоритм выполнения заданной операции над двоичным вектором A. Представить соответствующий фрагмент java-программы.

Длина двоичного вектора − 16 битов. Поменять местами первую и третью тетрады вектора. Пример:



A: 1111 0101 1100 1001 (0xF5C9)

Алгоритм решения задачи (последовательность действий)

1) Выделить первую тетраду: 1111 0000 0000 0000
   B=A & 0xF000;
2) Сдвинуть вправо на 8 бит: **0000 0000 1111 0000**
   B=B >> 8;
3) Выделить третью тетраду: 0000 0000 1100 0000
   D=A & 0xF0;
4) Сдвинуть влево на 8 бит: **1100 0000 0000 0000**
   D=D << 8;
5) Объединить два полученных двоичных вектора:
   D=D | B; **1100 0000 1111 0000**
6) Обнулить первую и третью тетрады в A:
   A=A & 0xF0F; **0000 0101 0000 1001**
7) Объединить A и D:
   A=A | D **1100 0101 1111 1001**

При написании java-программы для хранения двоичных векторов выбираем тип int по следующим причинам:

1) В short не поместится константа 0xF000 (не хватит одного бита, который отдан под знак). Максимальное значение для short – 0x7FFF (32767).
2) Операнды и результат при выполнении поразрядных логических операций все равно расширяются до int.

Два лишних старших нулевых байта в int-переменной просто не будут использоваться.

Фрагмент java-программы:

```
int A = 0xF5C9;
int B = A & 0xF000;
B = B >> 8;
int D = A & 0xF0;
D = D << 8;
D = D | B;
A = A & 0xF0F;
A = A | D;
```

Часть 3

Задание 12. В java-программе определены следующие переменные:

```
int a, b, c;
float d, e;
char ch1, ch2;
boolean f, f1, f2;
```

Определить значение переменной f при заданных значениях остальных переменных. Представить последовательность вычисления значения логического выражения, определяющего значение переменной f. Переписать выражение без лишних, по мнению студента, скобок. Обосновать. Что изменится, если в логическом выражении использовать удвоенные знаки логических операций (&& и ||)? Как значение переменной f будет представлено в компьютере?

a=15; b=20; c=30; d=3,75; e=9,98; ch1='N'; ch2='M';f1=true; f2=false;
f = (a < c+b) & (c > b) | (d – e > 0) & (ch2 == ch1+1) | !f1 & f2

При вычислении значения f нужно учитывать порядок действий в соответствующем выражении [4], с. 104-105. Согласно приоритетам операции, используемые в Java можно расположить следующим образом: 1) круглые и квадратные скобки; 2) операции инкремента (++), декремента (−−), побитовой инверсии (~) и логического отрицания (!), 3) арифметические операции умножения (*), деления (/), деления по модулю (%), 4) арифметические операции сложения (+) и вычитания (−), 5) операции сдвигов, 6) операции сравнения (>, >=, <,<=); 7) операции сравнения (==, !=); 8) побитовое или логическое И (&), 9) побитовое исключающее ИЛИ (^), 10) побитовое или логическое ИЛИ (|), 11) короткое логическое И (&&), 12) короткое логическое ИЛИ (||), 13) троичная условная операция (?:), 14) операция присваивания (=).

Круглые скобки, имеющие наивысший приоритет позволяют изменить порядок действий в выражении.

$$\overset{2}{(}a<c+b\overset{1}{)}\ \&\ \overset{9}{(}c>b\overset{3}{)}\ |\ \overset{12}{(}d-e>0\overset{4}{)}\ \&\ \overset{5}{(}ch2==ch1+1\overset{10}{)}\ |\ \overset{7}{!}f1\ \&\ \overset{6}{f}2$$

1) $c+b:\quad 20+30=50;$

2) $a<c+b:\quad 15<50=true;$

3) $c>b:\quad 30>20=true;$

4) $d-e:\quad 3{,}75-9{,}98=-6{,}23;$

5) $d-e>0:\quad -6{,}23>0=false;$

6) $ch1+1:\quad 0x4D+1=0x4E;$

7) $ch2==ch1:\quad 0x4E==0x4E=true;$

8) $!f1:\quad !true=false;$

9) $\underbrace{a<c+b}_{true}\ \&\ \underbrace{c>b}_{true}:\quad true\ \&\ true=true;$

10) $\underbrace{d-e>0}_{false}\ \&\ \underbrace{ch2==ch1+1}_{true}:\quad false\ \&\ true=false;$

11) $\underbrace{!f1}_{false}\ \&\ f2:\quad false\ \&\ false=false;$

12) $\underbrace{d-e>0}_{false}\ \&\ \underbrace{ch2==ch1+1}_{true}:\quad false\ \&\ true=false;$

13) $\underbrace{a<c+b\ \&\ c>b}_{true}\ |\ \underbrace{d-e>0\ \&\ ch2==ch1+1}_{false}:\quad true\ |\ false=false;$

14) $\underbrace{a<c+b\ \&\ c>b\ |\ d-e>0\ \&\ ch2==ch1+1}_{false}\ |\ \underbrace{!f1\ \&\ f2}_{false}:\quad false\ |\ false=false;$

Анализируя приоритеты операций, можно сделать вывод, что в рассматриваемом выражении можно опустить все скобки. Результат при этом не изменится, хотя последовательность выполнения тех же действий изменится: первым станет восьмое, вторым – первое, третьим – четвертое, четвертым – шестое, пятым – второе, шестым – третье, седьмым – пятое, восьмым – седьмое, последовательность остальных действий не изменится:

$$\overset{5}{a}<\overset{2}{c}+\overset{9}{b}\ \&\ \overset{6}{c}>\overset{12}{b}\ |\ \overset{3}{d}-\overset{7}{e}>\overset{10}{0}\ \&\ \overset{8}{ch2}==\overset{4}{ch1}+\overset{13}{1}\ |\ \overset{1}{!}f1\ \&\ \overset{11}{f}2$$

Использование коротких логических операций позволяет сделать их выполнение более быстрым: если значение левого операнда полностью определяет значение, возвращаемое операцией (для && − false, для || − true), то правый операнд не вычисляется.

Если в исходном выражении использовать короткие логические операции (&& и ||), то его можно записать так:

$$\overset{2}{(}a<c+b\overset{1}{)}\ \&\ \&\ \overset{9}{(}c>b\overset{3}{)}\ ||\ \overset{12}{(}d-e>0\overset{4}{)}\ \&\ \&\ \overset{5}{(}ch2==ch1+1\overset{10}{)}\ ||\ \overset{7}{!}f1\ \&\ \&\ \overset{6}{f}2$$

Тогда правый операнд не будет вычисляться в десятом, одиннадцатом, двенадцатом и тринадцатом действиях.

В памяти компьютера переменная типа boolean занимает 1 байт. Если значение переменной равно false, то все биты этого байта равны 0. Любая другая комбинация значений битов интерпретируется как true. Переменная f после присвоения ей значения выражения будет представлена в памяти компьютера следующим образом:

f	0	0	0	0	0	0	0	0

Задание 13. Разработать фрагмент java-программы, вычисляющий значение символьной (тип char) переменной *a* в зависимости от значений символьных переменных *c* и *d*. В программе для присвоения значения переменной *a* использовать троичную условную операцию java. Привести тесты для проверки правильности работы программы. Представить символьные и соответствующие им двоичные значения переменных *c*, *d* и *a* для каждого теста.
a=c++, если c – знак ')' или '$', d – не буква;
$a=0414_{16}$, в противном случае.

Для вычисления значения переменной a используем троичную условную операцию (?:), определенную в Java. *Ее синтаксис:*

expression1 ? expression2 : expression3

expression1 – логическое выражение, (т.е. имеющее значение типа boolean).

Если expression1 − true, то вычисляется expression2, иначе (если expression1 − false), вычисляется expression3. Результатом условной операции является значение вычисляемого выражения. Требуется, чтобы expression2 и expression3 возвращали значение одного и того же типа.

Для определения, является ли значение переменной d буквой, используем функцию isLetter (x), определенную в библиотечном классе Character. Если d – буква, функция вернет true, иначе – false.

Чтобы в выражении записать символ, представленный его шестнадцатеричным кодом (в таблице unicode), можно использовать соответствующую escape (управляющую)-последовательность: '\uxxxx', где xxxx – шестнадцатеричные цифры числа. Применение обычного шестнадцатеричного литерала также возможно.

Фрагмент java-программы:

```
char a, c = ')', d = '9';
a = (c == '&' || c=='$') && ! Character. isLetter (d) ? c ++ : '\u0414';
```

Тесты – это наборы входных и соответствующих им выходных данных, посредством которых проверяют правильность работы программы. Набор тестов должен быть полным, чтобы проверить все варианты развития событий в программе.

Набор тестов для проверки предложенного фрагмента программы:

№	Вход		Выход	Примечания
	c	d	a	
1	')'	'9'	'*'	c++
2	'$'	'&'	'%'	c++
3	'$'	'L'	'Д'	'\u0414'
4	'D'	'&'	'Д'	'\u0414'

Отметим, что в четвертом тесте не имеет значения, чему равно d, т.к. используется короткая логическая операция &&, левый операнд которой в данном случае равен нулю (полностью определяет значение, возвращаемое операцией), а, значит, правый операнд вообще не будет вычисляться.

Соответствие кодов символьным обозначениям получено с помощью кодовой таблицы unicode [6]: c = ')' – код 0x29; c++ (увеличение на 1) – код 0x2A – символ '*'; escape-последовательность '\u0414' представляет шестнадцатеричный код 0x414, соответствующий символу 'Д'.

Представление значения переменной *a* в компьютере для каждого теста:

№	a		
	Символ	Unicode (16-ричный)	Машинное представление
1	'*'	0x2A	0000 0000 0010 1010
2	'%'	0x26	0000 0000 0010 0110
3	'Д'	0x0414	0000 0100 0001 0100
4	'Д'	0x0414	0000 0100 0001 0100

Задание 14. Указать, какие из операторов вызовут ошибки (связанные с некорректным использованием типов данных) в java-программе, а какие нет? Обосновать.

Номера операторов (последовательностей операторов) из таблицы 3.16:

7, 14, 15, 27, 31, 32, 40, 42, 43, 45, 48, 51, 53, 54 *(взяты для примера, не соответствуют реальным номерам).*

№	Исходная последовательность операторов	Анализ исходной последовательности операторов	Исправленная последовательность операторов
7	float b = 674.25f; long m = b;	Данная последовательность операторов повлечет за собой ошибку компиляции, т.к. не определено автоматическое преобразование из float в long. Требуется явное преобразование типа	float b = 674.25; long m = (long) b;

14	byte m = 45; int n = m;	Последовательность не вызовет ошибки. Будет выполнено автоматическое расширяющее преобразование из byte в int.	–
15	int s = −64; byte q = s;	Последовательность не вызовет ошибки, т.к. значение int-переменной *s* (−64) входит в диапазон типа byte, в противном случае компилятор выдал бы ошибку.	–
27	long k = 0x7FFFFFFFFFFFFFF;	Последовательность вызовет ошибку компиляции, т.к. тип используемого литерала – int, а значение литерала не помещается в 32 битах. Для исправления ошибки нужно явно указать, что тип литерала – long (добавить *l* или *L* в конце записи литерала).	long k = 0x7FFFFFFFFFFFFFFL;
31	float h = (float) 22.7;	Оператор не вызовет ошибки, т.к. указано явное преобразование во float литерала типа double.	–
32	byte b = 0x5F; b = b << 4;	Второй оператор последовательности вызовет ошибку компиляции, т.к. результат операции сдвига, имеющий тип int, присваивают переменной типа byte, которая на 3 байта короче int.	byte b = 0x5F; b = (byte)(b << 4);

40	boolean a = false;
	boolean b = true;
	boolean c = a > b;
		Последовательность операторов вызовет ошибку, т.к. из всех операций сравнения для boolean определены только операции равенства и неравенства.	boolean a = false;
			boolean b = false;
			boolean c = a != b;

42	int n = 01678;	Оператор вызовет ошибку компиляции, т.к. с ведущего нуля начинается восьмеричный литерал, и цифры 8 в его записи быть не может.	int n = 01677

43	double b = 927.325E-3;	Оператор не вызовет ошибки. Вещественный литерал, представленный в научном формате, имеет тип double и присваивается переменной того же типа.	–
45	int r = −2247583647;	Оператор вызовет ошибку компиляции, т.к. значение литерала, имеющего тип int выходит за границы диапазона int (от −2147483648 до 2147483647)	long r = −2247583647;
48	float a = 123.56;	Оператор вызовет ошибку компиляции «Возможна потеря точности», т.к. значение вещественного литерала, имеющего по умолчанию тип double (64 бита), присваивается переменной типа float (32 бита). Нужно явно указать тип литерала или сделать явное преобразование типа.	float a = 123.56f; или float a = 123.56F; или float a = (float)123.56;

51	double a = -1.5, b = 1.5, h = 0.2;
	double x, f = 0.0;
	for (x=a; x!=b; x=x+h)
	{y =x*x+3x-10; …};
		Данная последовательность операторов вызовет ситуацию зацикливания во время выполнения программы. Это связано с тем, что операции над вещественными числами (x=x+h) выполняются с погрешностью, вызванной ограниченной точностью представления чисел в ЭВМ. Условие выхода из цикла (x!=b) никогда не выполнится.	double a = -1.5, b = 1.5, h = 0.2;
			double x, f = 0.0;
			for (x=a; x < b; x=x+h)
			{y =x*x+3x-10; …};

53	double a = 45.38;
	double b = −0.5;
	double c = 5/24*(a*a+b*b+2.45);
		Программа выполнится успешно, но результат будет неверный (равен нулю), т.к. 5 на 24 делится по правилам целочисленного деления (частное 0 , остаток 5).	double a = 45.38;
			double b = −0.5;
			double c = 5.0/24.0 *(a*a+b*b+2.45);

54	int a=234, b=135; if (a < b−10)…	Ошибок нет, после if в скобках записано логическое выражение, которое может принимать значения true или false (при данных значениях a и b – false), что соответствует java-синтаксису оператора if.	–