

Министерство образования Украины
ОДЕССКИЙ НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Кафедра компьютеризированные системы управления

**СБОРНИК ЛАБОРАТОРНЫХ РАБОТ
С ПРИМЕРАМИ РЕШЕНИЯ ЗАДАЧ ПО АЛГОРИТМИЗАЦИИ
И ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ СИ**

Одесса – 2017

УДК 681.3 (075.8)
Д 30

Рецензенты:

Д 30 Сборник лабораторных работ с примерами решения задач по алгоритмизации и программированию на языке Си: Учебно-методическое пособие для студентов высших технических учебных заведений/ Сперанский В.А. – Одесса.: Астропринт, 2017, 115 с.

Настоящее учебно-методическое пособие представляет собой сборник лабораторных работ по информатике. Изложенный материал позволяет студентам без каких либо базовых знаний по программированию за время обучения освоить все основные конструкции языка Си и уверенно овладеть навыками алгоритмизации.

Все лабораторные работы сопровождаются примерами решения задач.

Совокупность приобретенных навыков позволяет студентам уже на первом курсе разрабатывать довольно сложные алгоритмы и подготовиться к выполнению НИРС на старших курсах на профессиональном уровне.

ВВЕДЕНИЕ

Современные тенденции все более широкого использования информационных технологий и средств компьютерного моделирования во всех отраслях экономики ставят новые требования по уровню подготовленности студентов высших технических заведений в области программирования и информатики. Создание у студентов достаточной теоретической и практической базы по этим предметам позволяет им быстро включаться в решение актуальных производственных задач, затрачивая минимальное время на переподготовку.

При этом особое внимание должно быть уделено подготовке будущих специалистов в области разработки интегральных, сенсорных систем и современных САПР. Требования к базовой подготовке студентов по данным специальностям должны быть еще выше, чем к остальным студентам.

Данное пособие позволяет студентам без каких либо базовых знаний по программированию за время обучения освоить все основные конструкции языка Си и уверенно овладеть навыками алгоритмизации. Данное пособие составлено на основе курса лабораторного практикума по дисциплинам «Информатика», «Теория алгоритмов и программирования» выполняемых в Одесском национальном политехническом университете для студентов специальностей «Экономическая кибернетика» и «Автоматизация и компьютерно-интегрированные технологии».

Изложение материала пособия позволяет его использовать не только преподавателями, но и студентами при самоподготовке. Все лабораторные работы сопровождаются примерами решения задач.

Все программы пособия выполнялись в среде Microsoft Visual Studio 2015.

Совокупность приобретенных навыков позволяет студентам уже на первом курсе разрабатывать довольно сложные алгоритмы и подготовиться к выполнению НИРС на старших курсах на профессиональном уровне.

Лабораторная работа № 1.

Тема: «Системы счисления. Арифметические операции в разных системах счисления. Перевод из одной системы счисления в другую»

Теоретические сведения. Под *системой счисления* понимается способ записи чисел с помощью символов (цифр, букв и т.д.). Системы счисления бывают *позиционные* и *непозиционные*. *Непозиционной* является, например, римская система счисления. В *позиционных* системах счисления любое число записывается в виде последовательности символов, количественное значение («вес») которых зависит от местоположения в числе, т.е. позиции в записи числа. *Основанием* позиционной системы счисления называется целое число, определяющее количество символов, используемых в ней (обозначим его через p). В позиционной системе счисления с основанием p (p -ичной системе счисления) любое число R может быть представлено в виде:

$$R_{(p)} = a_N p^N + a_{N-1} p^{N-1} + \dots + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + \dots + a_{-K} p^{-K}, \quad (1.1)$$

где коэффициенты a_i – символы в изображении числа R , которые принимают значения от 0 до $p-1$. Обычно число $R_{(p)}$ представляется записью коэффициентов a_i : $R_{(p)} = a_N a_{N-1} \dots a_1 a_0 . a_{-1} \dots a_{-K}$. Например:

$$265,23_{(10)} = 2 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0 + 2 \cdot 10^{-1} + 3 \cdot 10^{-2},$$

$$23,78_{(16)} = 2 \cdot 16^1 + 3 \cdot 16^0 + 7 \cdot 16^{-1} + 8 \cdot 16^{-2},$$

$$10001,01_{(2)} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}.$$

Величина p показывает, во сколько раз численное значение единицы данного разряда больше численного значения единицы предыдущего разряда. В вычислительной технике широко используются позиционные системы счисления (двоичная, восьмеричная, десятичная, шестнадцатеричная).

Обычно в памяти ПЭВМ, на уровне аппаратной реализации, информация представляется в двоичной системе счисления.

Двоичная система счисления. В двоичной системе счисления используются две – цифры 0 и 1. Основание двоичной системы счисления записывается в виде $2_{(10)} = 2 \cdot 10^0 = 1 \cdot 2^1 + 0 \cdot 2^0 = 10_{(2)}$.

Таблица 1.1. Двоичная таблица сложения.

+	0	1
0	0	1
1	1	10

Таблица 1.2. Двоичная таблица умножения.

×	0	1
0	0	0
1	0	1

Арифметические операции в двоичной системе счисления выполняются с помощью Таблиц 1, 2 по тем же правилам, что и в десятичной системе счисления.

Приведем некоторые примеры выполнения основных операций в двоичной системе счисления.

Пример 1.1.

а) сложение:

$$\begin{array}{r} 0,000011101 \\ + 0,110000011 \\ \hline 0,110100000 \end{array}$$

б) вычитание:

$$\begin{array}{r} 1,110010101 \\ - 0,100001011 \\ \hline 1,010001010 \end{array}$$

в) умножение:

$$\begin{array}{r} 1,1001 \\ \times 0,11 \\ \hline 11001 \\ 11001 \\ \hline 1,001011 \end{array}$$

г) деление:

$$\begin{array}{r} 111,10101 \overline{) 1,01} \\ \underline{101} \\ 101 \\ \underline{101} \\ 0000101 \\ \underline{101} \\ 0 \end{array}$$

Восьмеричная система счисления. В восьмеричной системе счисления используются цифры 0, 1, 2, 3, 4, 5, 6, 7. Основание записывается в виде $8_{(10)} = 8 \cdot 10^0 = 1 \cdot 8^1 + 0 \cdot 8^0 = 10_{(8)}$.

Таблица 1.3. Восьмеричная таблица сложения.

+	1	2	3	4	5	6	7
1	2	3	4	5	6	7	10
2	3	4	5	6	7	10	11
3	4	5	6	7	10	11	12
4	5	6	7	10	11	12	13
5	6	7	10	11	12	13	14
6	7	10	11	12	13	14	15
7	10	11	12	13	14	15	16

Таблица 1.4. Восьмеричная таблица умножения.

×	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	10	12	14	16
3	3	6	11	14	17	22	25
4	4	10	14	20	24	30	34
5	5	12	17	24	31	36	43
6	6	14	22	30	36	44	52
7	7	16	25	34	43	52	61

Приведем некоторые примеры выполнения основных операций в восьмеричной системе счисления.

Пример 1.2.

а) сложение:

$$\begin{array}{r} 54,724 \\ + 34,731 \\ \hline 111,655 \end{array}$$

б) вычитание:

$$\begin{array}{r} 1547,214 \\ - 54,331 \\ \hline 1472,663 \end{array}$$

в) умножение:

$$\begin{array}{r} 15,21 \\ \times 4,31 \\ \hline 1521 \\ 4763 \\ 6504 \\ \hline 72,1751 \end{array}$$

г) деление:

$$\begin{array}{r} 451,67 \overline{) 321} \\ \underline{321} \\ 1306 \\ \underline{1163} \\ 1237 \\ \underline{1163} \\ 00540 \\ \underline{321} \\ 217... \end{array}$$

Шестнадцатеричная система счисления. В шестнадцатеричной системе счисления используются цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 и буквы A, B, C, D, E, F. Основание шестнадцатеричной системы записывается в виде $16_{(10)} = 16 \cdot 10^0 = 1 \cdot 16^1 + 0 \cdot 16^0 = 10_{(16)}$.

Таблица 1.5. Шестнадцатеричная таблица сложения.

+	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Таблица 1.6. Шестнадцатеричная таблица умножения.

×	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	F	1E	1E	2D	3C	4B	5A	69	78	87	A5	B4	C3	D2	E1

Приведем некоторые примеры выполнения основных операций в шестнадцатеричной системе счисления.

Пример 1.3.

а) сложение:

$$\begin{array}{r} 197,AB \\ + 54,42 \\ \hline 1EB,ED \end{array}$$

б) вычитание:

$$\begin{array}{r} 1A35,F1 \\ - 1EB,ED \\ \hline 184A,04 \end{array}$$

в) умножение:

$$\begin{array}{r} 37,21 \\ \times 54 \\ \hline DC84 \\ 113A5 \\ \hline 1216,D4 \end{array}$$

г) деление:

$$\begin{array}{r} 82,B1 \overline{) 43} \\ \underline{43} \quad 1,F3... \\ 3FB \\ \underline{3ED} \\ E1 \\ \underline{C9} \\ 18... \end{array}$$

Таблица 1.7. Запись чисел в различных системах счисления.

Десятичное число	Двоичное число	Восьмеричное число	Шестнадцате- ричное число
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
...

Перевод чисел из одной системы счисления в другую.

Правило 1. Перевод смешанного числа (числа с дробной частью) из p – ичной системы счисления в q – ичную систему счисления, когда имеет место соотношение $p = q^k$ (k – целое положительное число), осуществляется поразрядно. Каждая p – ичная цифра заменяется равным ей k – разрядным числом, записанным в q – ичной системе счисления.

Для упрощения перевода по описанному выше правилу удобно использовать Таблицу 1.7.

Пример 1.4.

$$\text{а) } A61,87C_{(16)} = \underbrace{(1010)}_{A_{(16)}} \underbrace{(0110)}_{6_{(16)}} \underbrace{(0001)}_{1_{(16)}}, \underbrace{(1000)}_{8_{(16)}} \underbrace{(0111)}_{7_{(16)}} \underbrace{(1100)}_{C_{(16)}}_{(2)}.$$

$$\text{б) } 31,471_{(8)} = \underbrace{(011)}_{3_{(8)}} \underbrace{(001)}_{1_{(8)}}, \underbrace{(100)}_{4_{(8)}} \underbrace{(111)}_{7_{(8)}} \underbrace{(001)}_{1_{(8)}}_{(2)}.$$

Обратный перевод из q – ичной системы счисления в p – ичную систему счисления осуществляют, разбивая q – ичную запись числа на группы по k цифр, двигаясь от запятой вправо и влево и заменяя каждую группу цифр ее p – ичным изображением. При этом если крайние группы окажутся неполными, то их дополняют до k цифр незначащими нулями.

Пример 1.5.

$$\text{а) } 1110010,01101111001_{(2)} = \underbrace{(0111)}_{7_{(16)}} \underbrace{(0010)}_{2_{(16)}}, \underbrace{(0110)}_{6_{(16)}} \underbrace{(1111)}_{F_{(16)}} \underbrace{(0010)}_{2_{(16)}}_{(2)} = 72,6F2_{(16)};$$

$$\text{б) } 1001111,011100_{(2)} = \underbrace{(001)}_{1_{(8)}} \underbrace{(001)}_{1_{(8)}} \underbrace{(111)}_{7_{(8)}}, \underbrace{(011)}_{3_{(8)}} \underbrace{(100)}_{4_{(8)}}_{(2)} = 117,34_{(8)}.$$

Перевод смешанного числа производится отдельно для целой и дробной частей числа.

Правило 2.1. Перевод целой части. Целую часть числа, записанную в p – ичной системе счисления, делят на основание q – ичной системы счисления, записанное в p – ичной системе счисления (все операции производятся по правилам p – ичной системы счисления). Полученное в остатке число является младшей (последней) цифрой в q – ичной записи числа. Полученное частное снова делят на основание q ; остаток – предпоследняя цифра в искомой записи числа; и т.д. Операцию деления проводят до тех пор, пока в частном не получат число, меньшее q ; частное – старшая (первая) цифра в q – ичной записи числа.

Пример 1.6.

$$\text{а) } 345_{(10)} = 159_{(16)};$$

$$\begin{array}{r} 345 \overline{) 16} \\ \underline{32} \quad 21 \overline{) 16} \\ 25 \quad 16 \quad 1 \\ \underline{16} \quad 5 \\ 9 \end{array}$$

$$\text{б) } 24_{(10)} = 11000_{(2)}.$$

$$\begin{array}{r} 24 \overline{) 2} \\ \underline{24} \quad 12 \overline{) 2} \\ 0 \quad 12 \quad 6 \overline{) 2} \\ 0 \quad 6 \quad 3 \overline{) 2} \\ 0 \quad 0 \quad 2 \quad 1 \end{array}$$

Правило 2.2. Перевод дробной части. Дробную часть числа, записанную в p – ичной системе счисления, умножают на основание q – ичной системы

счисления, записанное в p -ичной системе счисления. Целая часть произведения будет старшей цифрой изображения дроби (первой после запятой) в q -ичной записи числа. Дробную часть произведения снова умножают на основание q ; целая часть – следующая цифра после запятой в q -ичной записи дроби. Процесс продолжают до тех пор, пока дробная часть не станет нулем или будет получено требуемое количество знаков после запятой в дробной записи числа. Целые части записывают в q -ичной системе счисления.

Пример 1.7.

а) $0,2_{(10)} = 0,(0011)_{(2)}$; б) $0,12_{(10)} \approx 0,1EB_{(16)}$; в) $0,A_{(16)} = 0,625_{(10)}$.

$$\begin{array}{r} 0,2 \\ \underline{2} \\ 0_{(2)} = 0,4 \\ \underline{2} \\ 0_{(2)} = 0,8 \\ \underline{2} \\ 1_{(2)} = 1,6 \\ \underline{2} \\ 1_{(2)} = 1,2 \\ \underline{2} \\ 0_{(2)} = 0,4 \\ \underline{2} \\ 0_{(2)} = 0,8 \\ \dots \end{array}$$

$$\begin{array}{r} 0,12 \\ \underline{16} \\ 1_{(16)} = 1,92 \\ 0,92 \\ \underline{16} \\ E_{(16)} = 14,72 \\ 0,72 \\ \underline{16} \\ B_{(16)} = 11,52 \\ \dots \end{array}$$

$$\begin{array}{r} 0,A \\ \underline{A} \\ 6_{(10)} = 6,4 \\ 0,4 \\ \underline{A} \\ 2_{(10)} = 2,8 \\ 0,8 \\ \underline{A} \\ 5_{(10)} = 5,0 \end{array}$$

При переводе смешанного числа результаты для целой и дробной частей, полученные по правилам 2.1, 2.2, записывают, отделяя друг от друга дробной запятой.

Пример 1.8.

$$345,02_{(10)} = 101011001,(0011)_{(2)}.$$

$$345_{(10)} = 159_{(16)} = \underbrace{(0001)}_{1_{(16)}} \underbrace{(0101)}_{5_{(16)}} \underbrace{(1001)}_{9_{(16)}}_{(2)}; \quad 0,2_{(10)} = 0,(0011)_{(2)}.$$

$$\begin{array}{r} 345 \overline{) 16} \\ 32 \quad 21 \overline{) 16} \\ \underline{25} \quad \underline{16} \quad 1 \\ \underline{16} \quad \underline{5} \\ 9 \end{array}$$

$$\begin{array}{r} 0,2 \\ \underline{2} \\ 0_{(2)} = 0,4 \\ \underline{2} \\ 0_{(2)} = 0,8 \\ \underline{2} \\ 1_{(2)} = 1,6 \\ \underline{2} \\ 1_{(2)} = 1,2 \\ \underline{2} \\ 0_{(2)} = 0,4 \\ \dots \end{array}$$

Правило 3. Перевод чисел в 10 – ичную систему счисления рекомендуется выполнять суммированием с учетом «веса» цифры в числе по формуле (1.1):

$$a_N a_{N-1} \dots a_1 a_0 . a_{-1} \dots a_{-K} = a_N p^N + a_{N-1} p^{N-1} + \dots + a_1 p^1 + a_0 p^0 + a_{-1} p^{-1} + \dots + a_{-K} p^{-K}$$

Пример 1.9.

а) $1101,111_{(2)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 13,875_{(10)}$;

б) $1A,02_{(16)} = 1 \cdot 16^1 + 10 \cdot 16^0 + 0 \cdot 16^{-1} + 2 \cdot 16^{-2} = 26,0078125_{(10)}$.

Варианты заданий.

1. Представить следующие числа в развернутом виде (1.1):

а) $54,987_{(10)}$; б) $1011,011_{(2)}$; в) $DE,7_{(16)}$;

г) $A4,187_{(16)}$; д) $14,008_{(10)}$; е) $0,00011_{(2)}$;

ж) $3214,47_{(8)}$; з) $555,654_{(8)}$; и) $123,0111_{(16)}$.

2. Выполнить указанные арифметические операции над числами в соответствующих системах счисления:

1) в двоичной системе счисления:

а) $10011,1 + 11,00111$; б) $1111,0111 - 1,0001$;

в) $111,01 \cdot 1,01$; г) $\frac{1001,11}{11,01}$.

2) в восьмеричной системе счисления:

а) $34,1 + 11,17$; б) $12,121 - 1,1755$;

в) $62,1 \cdot 67,17$; г) $\frac{174,23}{34,5}$.

3) в шестнадцатеричной системе счисления:

а) $A23,F1 + 1,7$; б) $1343,31 - D1,7F$;

в) $23,F1 \cdot A,7$; г) $\frac{231,CD}{1,67}$.

3. Перевести следующие числа из двоичной системы счисления в восьмеричную и шестнадцатеричную системы счисления:

а) $111000111,11101$; б) $-111,101$;

в) $111011,101$; г) $-10000111011,101$;

д) $-1000111,1110111101$; е) $-11011,10111$.

4. Перевести следующие числа из двоичной системы счисления в десятичную (в дробной части получить четыре знака после запятой):

а) $1100001,111$; б) $11001,1011$;

в) $-1,01110111$; г) $1001,0001$.

5. Перевести следующие числа из шестнадцатеричной системы счисления в десятичную (в дробной части получить четыре знака после запятой):
- а) 1AF01,21; б) 1931,C1;
 - в) –1,41A; г) 0,00A6.
6. Перевести следующие числа из шестнадцатеричной системы счисления в двоичную:
- а) A01C,1FF; б) - 31,F104;
 - в) – B1,1001; г) 68,01FA.
7. Перевести следующие числа из десятичной системы счисления в шестнадцатеричную (в дробной части получить четыре знака после запятой):
- а) 784,19; б) - 31,04;
 - в) –121,1001; г) - 68,976.

Лабораторная работа № 2.

Тема: «Представление информации в ПЭВМ типа IBM PC/AT»

Теоретические сведения. Информация в памяти ЭВМ хранится и обрабатывается в двоичном виде. Форма записи данных в памяти машины называется *внутренним представлением информации* в ЭВМ. Единицей хранения информации является один бит, т.е. двоичный разряд, который может принимать значения 0 или 1. Применение двоичной системы счисления позволяет использовать для хранения информации элементы, имеющие всего два устойчивых состояния. Одно состояние служит для изображения единицы соответствующего разряда числа, другое – для изображения нуля. Обычно биты памяти группируются в более широкие структуры. Группа из восьми битов называется байтом. Поля памяти ЭВМ имеют специальные названия: 16 бит – слово, 32 бита – двойное слово, 1024 байта (1 К байт) – лист. Все байты пронумерованы, начиная с нуля.

Адресом любой информации считается адрес (номер) самого первого байта поля памяти, выделенного для ее хранения.

Существует два основных способа представления чисел, называемых представлением с *фиксированной* и с *плавающей* точкой.

Двоичные числа с фиксированной точкой. Для чисел с фиксированной точкой положение точки зафиксировано после младшей цифры числа, дробная часть отсутствует, точка в изображении числа опускается. Таким образом, в виде с фиксированной точкой могут храниться только целые числа (в памяти ЭВМ они записываются в двоичном виде). Следует отметить, что в алгоритмических языках для записи числа в память используются специальные слова – *атрибуты*. В языке Си, например, для указания того, что значение переменной должно быть записано с фиксированной точкой, применяют атрибуты **int**, **short int**, **long int**, **unsigned int**.

Целое двоичное число занимает в памяти 16 или 32 двоичных разряда. Это зависит от длины числа и способа представления переменной.

Рассмотрим, как записывается число в двойном слове (в слове оно записывается аналогично). Седьмой бит первого байта является служебным при хранении чисел со знаком, и его содержимое не оказывает влияния на абсолютную величину числа. Отметим, что для положительных чисел содержимым служебного бита является нуль. Младший двоичный разряд числа записывается в 0-ой бит, т.е. число заполняют справа налево. Если число положительно, то оставшиеся биты заполняются нулями.

Например, число $127_{(10)}$ в 4 байтах будет представлено следующим образом:

$$127_{(10)} = 7F_{(16)} = 1111111_{(2)} = 1 \cdot 2^6 + 1 \cdot 2^5 + \dots + 1 \cdot 2^1 + 1 \cdot 2^0$$

7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	биты
0000 0000	0000 0000	0000 0000	0111 1111	байты
1 – ый байт	2–ой байт	3–ий байт	4–ый байт	

Из приведенного выше представления видно, что в младшем разряде записан коэффициент при 2^0 , в следующем – при 2^1 и т.д. Очевидно, что если во всех битах с 0-го по 30-ый находятся 1, то максимальное целое положительное число, которое можно записать в двойном слове, имеет вид:

$$01111111 \ 11111111 \ 11111111 \ 11111111 = 2^{31} - 1 = 2147483647_{(10)}.$$

Форму записи положительных целых двоичных чисел называют *прямым кодом*. В этом случае их запись в отведенной памяти полностью соответствует символической записи в двоичной системе счисления.

Отрицательные числа записываются в *дополнительном коде*. Использование этого кода позволяет упростить аппаратную реализацию операции вычитания, которая заменяется операцией сложения уменьшаемого, представленного в прямом коде, и вычитаемого, представленного в дополнительном коде. *Дополнительный код* получается из прямого путем инвертирования каждого бита (*обратный код*) и затем добавления 1 к младшему биту числа.

Представим число $-95_{(10)}$ как двоичное число с фиксированной точкой.

$$-95_{(10)} = -5F_{(16)} = -1011111_{(2)}$$

1. Запишем прямой код числа в четыре байта:

00000000 00000000 00000000 01011111

2. Запишем обратный код числа:

11111111 11111111 11111111 10100000

3. Прибавив к младшему биту 1, получим дополнительный код числа:

$$\begin{array}{cccc}
 11111111 & 11111111 & 11111111 & 10100000 \\
 + & & & \\
 \hline
 \underbrace{11111111}_{1\text{-ый байт}} & \underbrace{11111111}_{2\text{-ой байт}} & \underbrace{11111111}_{3\text{-ий байт}} & \underbrace{10100001}_{4\text{-ый байт}}
 \end{array}$$

При хранении целых положительных чисел существует возможность почти вдвое увеличить числовой диапазон. Это достигается за счет использования служебного седьмого бита первого байта для хранения старшего коэффициента в двоичном представлении числа. Так, например, число $65535_{(10)} = FFFF_{(16)} = 11111111 \ 11111111_{(2)}$, объявленное как целое без знака, в двух байтах запишется в виде

$$\underbrace{11111111}_{1\text{-ый байт}} \ \underbrace{11111111}_{2\text{-ой байт}}$$

Двоичные числа с плавающей точкой. Числа, в которых положение точки не зафиксировано после некоторого разряда, а указывается специальным числом, называются числами с плавающей точкой.

В общем виде любое число A в системе счисления с основанием p можно представить в виде $A = m \cdot p^k$, где m - мантисса числа A , k - порядок числа. При этом если мантисса числа удовлетворяет неравенству $1 \leq |m| < p$, то число *нормализованное*.

Размеры областей, выделяемых под числа с плавающей точкой, существенно зависят от аппаратной реализации вычислительной техники. Для IBM PC/AT, например, числа с плавающей точкой могут, в зависимости от объявленных атрибутов **float**, **double**, **long double**, располагаться соответственно в 32, 64 и 80 битах памяти в виде:

float	\pm	характеристика		нормализованная мантисса
биты	31	30	23	22 0

double	\pm	характеристика		нормализованная мантисса
биты	63	62	52	51 0

long double	\pm	характеристика		нормализованная мантисса
биты	79	78	64	63 0

Мантисса числа записывается в нормализованном виде (в двоичном представлении числа перед точкой сохраняется один значащий двоичный бит).

Чтобы увеличить диапазон представления чисел в форматах float и double, единица перед точкой в двоичной нормализованной записи числа в память не записывается; в формате long double – единица сохраняется. Знак мантиссы записывается в последнем бите первого байта.

С целью упрощения аппаратной реализации арифметических операций в представлении числа с плавающей точкой знак порядка явно не хранится. Вместо порядка в память записывается величина, называемая *характеристикой*. Характеристика получается из порядка путем прибавления поправочного коэффициента, который для чисел типа float равен $127_{(10)} = 7F_{(16)}$, для double – $1023_{(10)} = 3FF_{(16)}$, для long double – $16383_{(10)} = 3FFF_{(16)}$.

Представим число $-18.2_{(10)} = -10010.(0011)_{(2)}$, нормализованная запись которого $-1.0010(0011)_{(2)} \cdot 2^{100}$, где нормализованная мантисса $m = -1.0010(0011)_{(2)}$, а порядок $k = 4_{(10)} = 100_{(2)}$. Перейдем от порядка к его характеристике: $4 + 127 = 131_{(10)} = 83_{(16)} = 1000\,0011_{(2)}$. Получим представление числа $18.2_{(10)}$ в формате float:

1	10000011	00100011001100110011010
31	30 23	22 0

Если количество цифр в числе, записываемом в ПЭВМ или полученном в результате вычислений, больше выделенного поля памяти, то избыточные цифры отбрасываются, а число, записываемое в память, округляется с избытком. В формате double характеристика равна $4 + 1023 = 1027_{(10)} = 403_{(16)} = 100\,0000\,0011_{(2)}$, и число представимо в виде:

1	10000000011	00100011001100110011...00110011
63	62 52	51 0

Размеры выделенных для хранения данных областей определяют интервал допустимых для этого атрибута значений – диапазон (Таблица 2.1). Всякое число, меньшее по абсолютной величине положительного минимального числа, представленного в соответствующем формате, будет записано в память в виде нуля. Для данного формата это так называемый машинный нуль. Кроме того, числа данного формата не должны превышать по абсолютной величине максимальное число, представленное в этом формате. В противном случае старшие биты числа будут потеряны, а результат не верен. Описанная ситуация называется переполнением разрядной сетки, а сами числа – машинной бесконечностью.

Таблица 2.1.

Тип данных	Кол-во бит	Диапазон	
		Abs(min)	Abs(max)
float	32	$3.4 \cdot 10^{-38}$	$3.4 \cdot 10^{38}$
double	64	$1.7 \cdot 10^{-308}$	$1.7 \cdot 10^{308}$
long double	80	$3.4 \cdot 10^{-4932}$	$3.4 \cdot 10^{4932}$

Символьная информация. В ПЭВМ для внутреннего представления символьных данных используется так называемый ASCII код, согласно которому каждому символу соответствует 8-разрядный код, т.е. в один байт записывается один символ. Например, текст «1486DX2» в памяти имеет вид:

$\underbrace{00110001}_1 \underbrace{00110100}_4 \underbrace{00111000}_8 \underbrace{00110110}_6 \underbrace{01000100}_D \underbrace{01011000}_X \underbrace{00110010}_2$

Варианты заданий.

- Представить следующие числа как двоичные с фиксированной точкой. Для записи выделяется слово.
 - $-1547_{(10)}$;
 - $10001100011_{(2)}$;
 - $-D17_{(16)}$;
 - $-1A4_{(16)}$;
 - $-148_{(10)}$;
 - $-100000011_{(2)}$;
 - $347_{(8)}$;
 - $-5554_{(8)}$;
 - $123_{(16)}$.
- Представить следующие числа как двоичные с фиксированной точкой без знака. Для записи выделяется слово.
 - $34547_{(10)}$;
 - $100011111100011_{(2)}$;
 - $41A4_{(16)}$;
 - $11148_{(10)}$;
 - $63471_{(8)}$;
 - $5541_{(8)}$.
- Представить следующие числа как двоичные с плавающей точкой в формате double.
 - $-17,56 \cdot 10^4_{(10)}$;
 - $11,1000101 \cdot 10^{-11}_{(2)}$;
 - $0,234 \cdot 10^{-3}_{(10)}$;
 - $0,100011101 \cdot 10^{100111}_{(2)}$;
 - $-13,17 \cdot 10^4_{(10)}$;
 - $-A,71 \cdot 10^3_{(16)}$;
 - $-10001,100011 \cdot 10^{10111}_{(2)}$;
 - $0,0001 \cdot 10^{-12}_{(16)}$.
- Представить следующие числа как двоичные с плавающей точкой в формате float.
 - $-17,56 \cdot 10^4_{(10)}$;
 - $11,1000101 \cdot 10^{-11}_{(2)}$;
 - $0,234 \cdot 10^{-3}_{(10)}$;
 - $0,100011101 \cdot 10^{100111}_{(2)}$;

д) $-13,17 \cdot 10^4_{(10)}$; е) $-A,71 \cdot 10^3_{(16)}$;
 ж) $-0,0001 \cdot 10^{-12}_{(16)}$; з) $10001,100011 \cdot 10^{1011}_{(2)}$

5. Интерпретировать содержимое четырех байт памяти как:

- целое число со знаком;
- целое число без знака;
- вещественное число с плавающей точкой.

а)

1	01100110	11100011111100110011010
31	30 23	22 0

б)

1	11100100	00111011000000110011011
31	30 23	22 0

в)

0	00000011	10101111001100110011111
31	30 23	22 0

г)

0	10001010	00000011000000110011000
31	30 23	22 0

д)

0	11000110	10110000111100110111111
31	30 23	22 0

е)

0	00000100	00000011000000110011000
31	30 23	22 0

ж)

1	10010111	10111111111100110111111
31	30 23	22 0

з)

0	11100010	00000011011110110011000
31	30 23	22 0

1	01000010	00000011100000110111111
31	30 23	22 0

0	00000000	10000011100000110110000
31	30 23	22 0

1	10000000	1000000000000000110000000
31	30 23	22 0



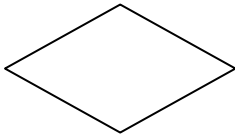

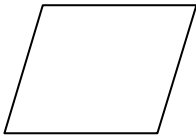


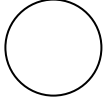
Тема: «Основы алгоритмизации. Построение блок-схем линейных и разветвляющихся вычислительных процессов»

Алгоритм – конечная последовательность точно определенных действий, приводящих к однозначному решению поставленной задачи. Главная особенность любого алгоритма – формальное исполнение, позволяющее выполнять заданные действия (команды) не только человеку, но и различным техническим устройствам (исполнителям). Процесс составления алгоритма называется *алгоритмизацией*. Алгоритмы могут быть заданы: словесно, таблично, графически (с помощью блок-схем). *Словесное* задание описывает алгоритм с помощью слов и предложений. *Табличное* задание служит для представления алгоритма в форме таблиц и расчетных формул. *Графическое* задание, или *блок-схема*, – способ представления алгоритма с помощью геометрических фигур, называемых *блоками*. Последовательность блоков и соединительных линий образуют блок-схему. Описание алгоритмов с помощью блок-схем является наиболее наглядным и распространенным способом задания алгоритмов. Блок-схемы располагаются сверху вниз. Линии соединения отдельных блоков показывают направление процесса обработки в схеме. Каждое такое направление называется ветвью. Алгоритм независимо от его структуры всегда имеет по одному блоку «Начало» и «Конец». Его ветви должны в конце сойтись, и по какой бы ветви не было бы начато движение, оно всегда должно привести к блоку «Конец».

17

отметить, что все блоки нумеруются. В этом случае номера проставляются вверху слева от блока (блоки «Начало», «Конец» и соединительные блоки не нумеруются). Стрелки на соединяющих линиях обычно не ставят при направлении сверху вниз и слева направо; если направление противоположное, то его показывают стрелкой на линии.

Таблица 3.1. Основные типы блоков.

	<p><i>Начало и конец алгоритма</i> (для функций «Вход», «Выход»)</p>
	<p><i>Блок обработки.</i> Внутри блока записываются формулы, обозначения и функции</p>
	<p><i>Блок условия.</i> Внутри блока записываются условия выбора направления действия алгоритма</p>
	<p><i>Блок предопределенного процесса</i> (функция/подпрограмма)</p>
	<p><i>Блок ввода информации</i></p>
	<p><i>Блок цикла с известным количеством повторений</i></p>
	<p><i>Блок вывода информации на печатающее устройство</i></p>
	<p><i>Соединительный блок</i></p>

Алгоритмы бывают *линейные, разветвляющиеся и циклические.*

Линейный алгоритм не содержит логических условий, имеет одну ветвь обработки и изображается линейной последовательностью связанных друг с

другом блоках. Условное изображение линейного алгоритма может быть представлено на рис. 3.1.

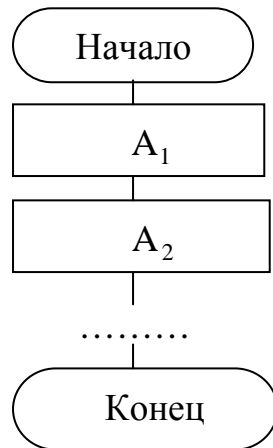


Рис. 3.1. Условное изображение линейного алгоритма.

Пример 3.1. Составить блок-схему вычисления $z = \varphi(y)$, $y = f(x)$, где φ, f – известные функции, при заданном значении переменной x (рис. 3.2).

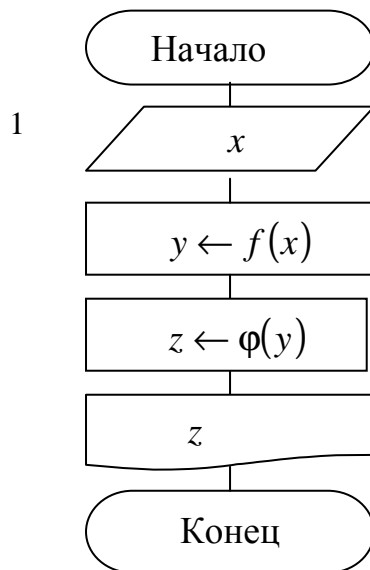


Рис. 3.2. Блок-схема примера 3.1.

Знак « \leftarrow » означает: внести значение переменной в ячейку памяти с определенным именем (операция присваивания).

Разветвляющийся алгоритм содержит одно или несколько логических условий и имеет несколько ветвей обработки (пример 3.2). Условное изображение разветвления представлено на рис. 3.3.

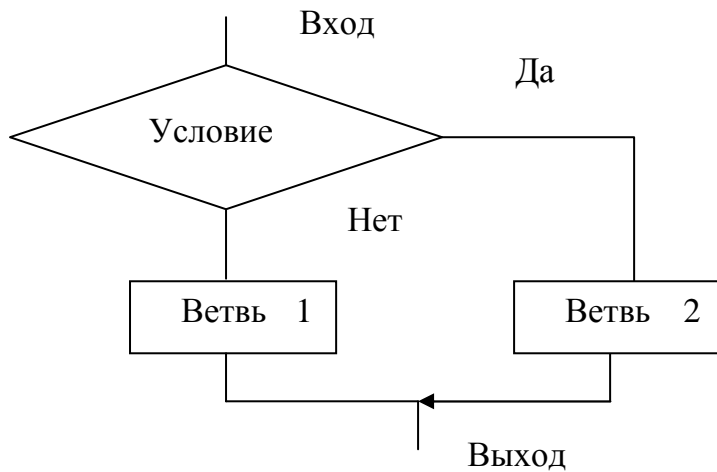


Рис. 3.3. Условное изображение разветвляющегося алгоритма.

Пример 3.2. Определить попадает ли точка внутрь круга, если радиус круга известен, а его центр и точка задаются своими координатами (рис. 3.4).

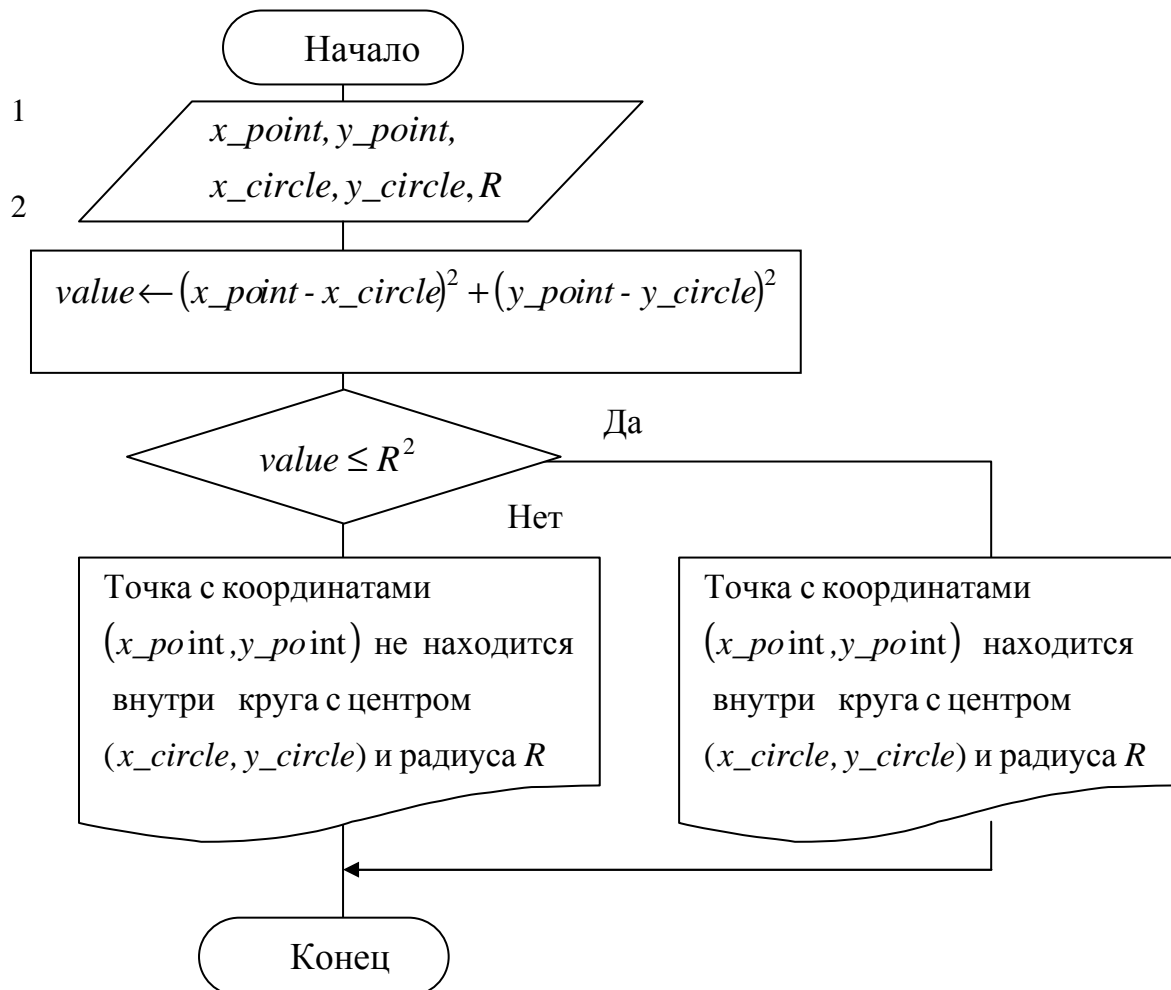


Рис. 3.4. Блок-схема примера 3.2.

Пример 3.3. Составить блок-схему нахождения корней квадратного уравнения. Коэффициенты квадратного уравнения ввести с клавиатуры (рис. 3.5).

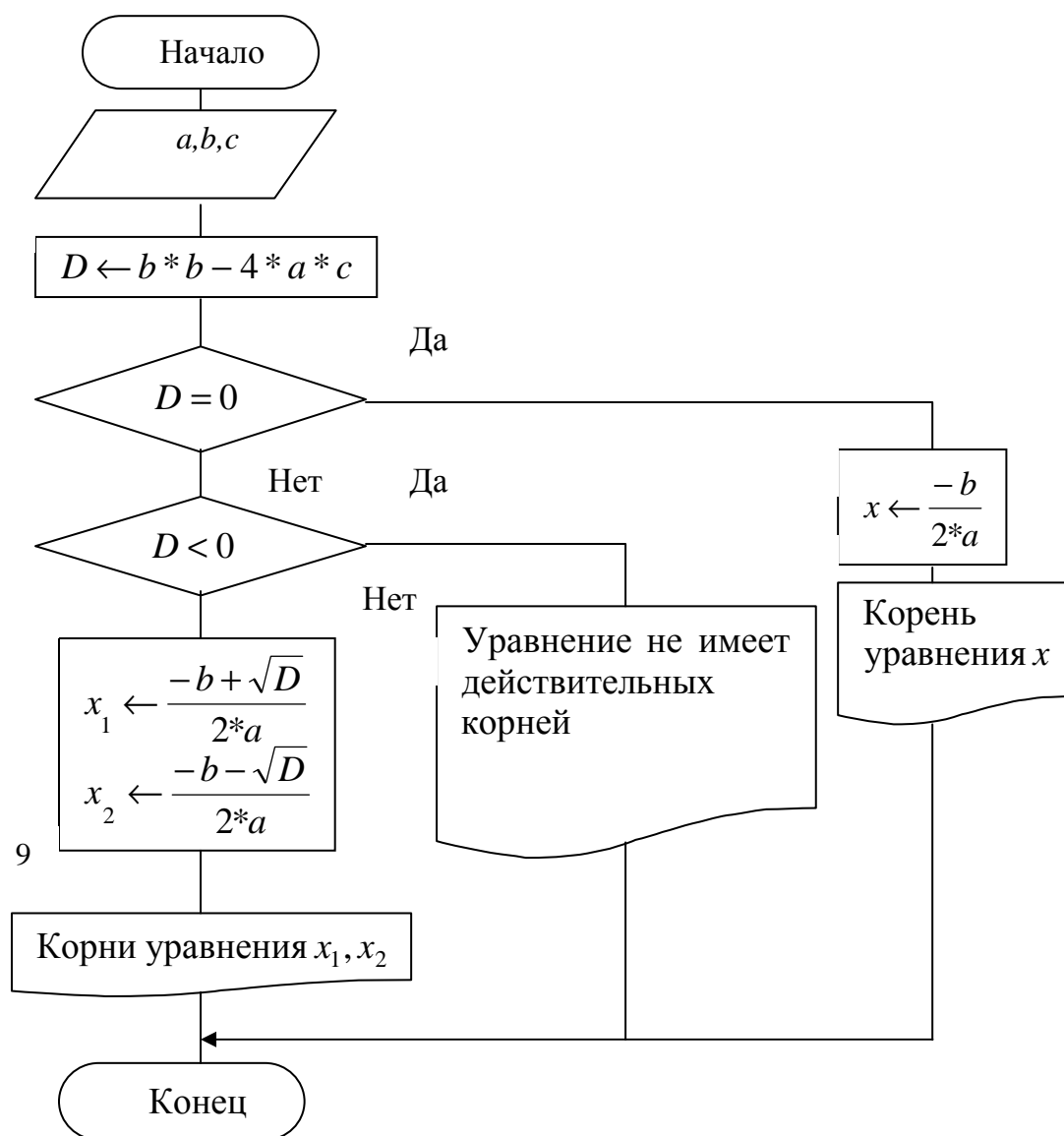


Рис. 3.5. Блок-схема примера 3.3.

Варианты заданий.

1. Даны x, y . Получить $\frac{|x| - |y|}{1 + |xy|}$.
2. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
3. Вычислить расстояние между двумя точками с данными координатами.
4. Найти площадь треугольника со сторонами a, b, c по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \text{ где } p = \frac{(a+b+c)}{2}.$$
5. Даны x, y, z . Найти w , если:

$$\text{а) } w = \sin \left| \left(y - \sqrt{|x|} \right) \left(x - \frac{y}{z^2 + \frac{x^2}{4}} \right) \right|; \quad \text{б) } w = x - \frac{x^2}{1 + \sin^2(x + y + z)};$$

$$\text{в) } w = \cos \left(z^2 + \frac{x^2}{4} + y \right); \quad \text{г) } w = y + x - \frac{\cos^2 x}{1 + \sqrt{|y + z|}}.$$

6. Даны два числа. Вывести первое число, если оно больше второго, и оба числа, если это не так.

7. Найти наименьшее из трех данных чисел.

8. Даны три числа, являющиеся длинами сторон треугольника. Определить тип треугольника (равносторонний, равнобедренный, разносторонний).

9. Даны x, y, z . Найти:

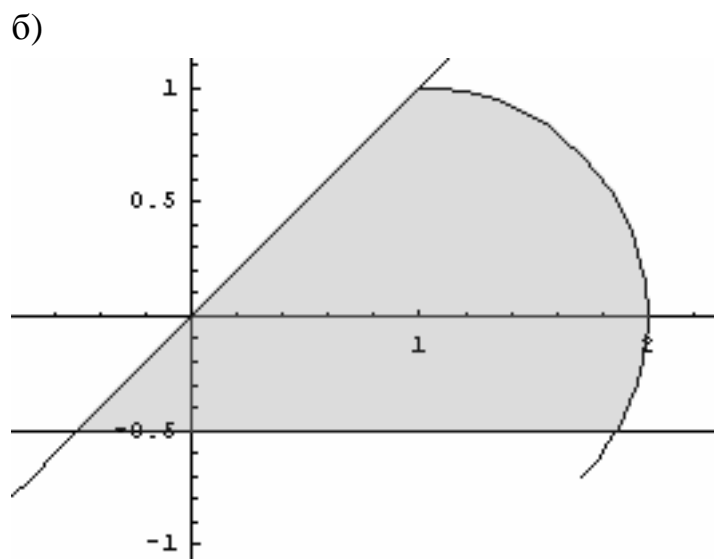
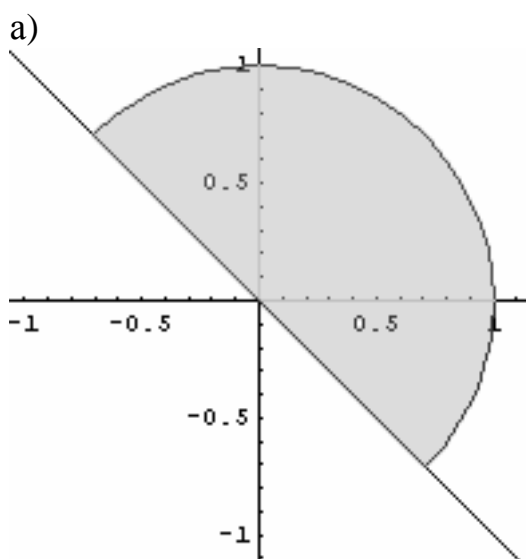
$$\text{а) } \max\{x + y + z, xyz\} + 3; \quad \text{б) } \min\{x^2 + y^2, y^2 + z^2\} - 4.$$

10. Дано x . Вычислить y , если:

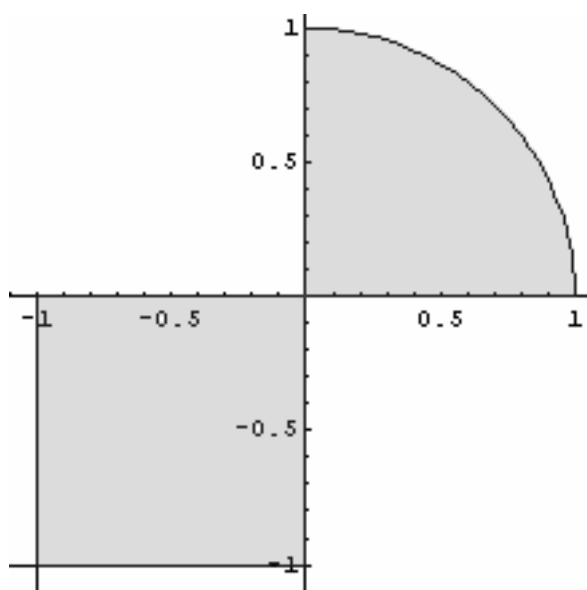
$$\text{а) } y = \begin{cases} x^2, & \text{при } -2 \leq x \leq 2, \\ 4, & \text{при } x < -2 \text{ и } x > 2; \end{cases} \quad \text{б) } y = \begin{cases} 0, & \text{при } x \leq 0 \\ x, & \text{при } 0 < x \leq 1, \\ x^4, & \text{при } x > 1 \end{cases};$$

$$\text{в) } y = \begin{cases} x^2 + 4x + 5, & \text{при } x \leq 2, \\ \frac{1}{x^2 + 4x + 5}, & \text{при } x > 2 \end{cases}; \quad \text{г) } y = \begin{cases} 0, & \text{при } x \leq 0 \\ x^2 - x, & \text{при } 0 < x \leq 1, \\ x^2 - \sin \pi x^2 - 1, & \text{при } x > 1 \end{cases}.$$

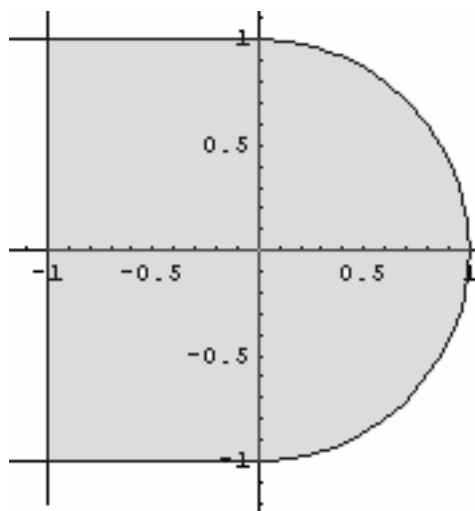
11. Определить попадает ли заданная точка внутрь заданной области.



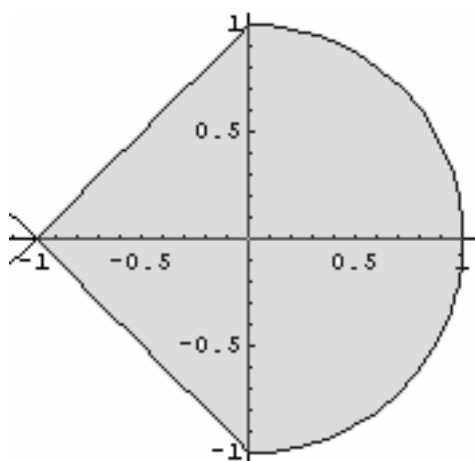
в)



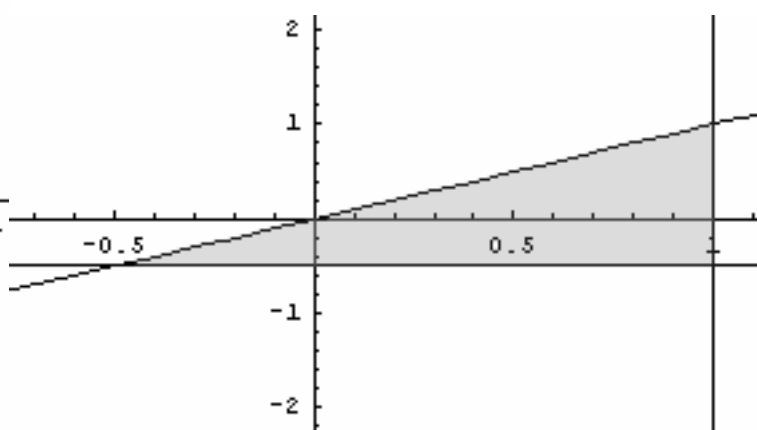
г)



д)



е)



Лабораторная работа № 4.

Тема: «Построение блок-схем циклических вычислительных процессов»

Теоретические сведения. *Циклический* алгоритм содержит один или несколько циклов. *Цикл* – это многократно повторяемая часть алгоритма. Цикл, не содержащий внутри себя других циклов, называют простым. Если он содержит внутри себя другие циклы или разветвления, то цикл называют сложным или вложенным. Любой цикл характеризуется одной или несколькими переменными, называемыми параметрами цикла, от анализа значений которых зависит выполнение цикла. *Параметр цикла* – переменная, принимающая при каждом вхождении в цикл новое значение. Условное изображение циклического алгоритма представлено на рис. 4.1.

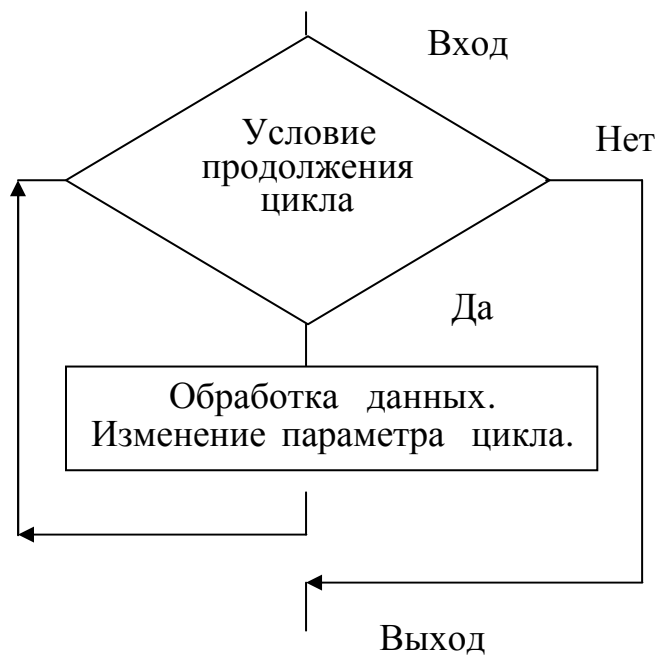


Рис. 4.1. Условное изображение циклического алгоритма.

Пример 4.1. Составить блок-схему нахождения суммы N первых целых чисел от 0 до $N - 1$ (рис. 4.2, рис. 4.3).

Вариант 1.

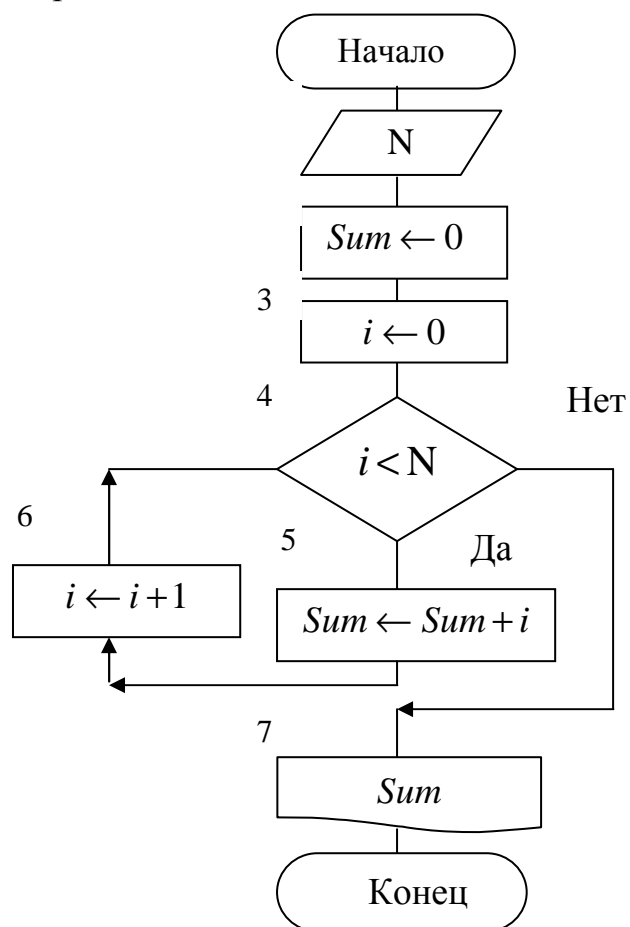


Рис. 4.2. Блок-схема примера 4.1.

Блок-схема любого циклического процесса (пример 4.1, вариант 1) независимо от многообразия сводящихся к нему задач должна содержать блок задания начального значения параметра цикла (блок 3), блок реализации необходимых вычислений (блок 5), блок изменения параметра цикла (блок 6) и блок проверки условия окончания цикла (блок 4).

Способ организации цикла зависит от условия задачи. Во многих задачах количество повторений цикла указывается (пример 4.1, вариант 2). Это так называемые *циклы с известным количеством повторений* или *циклы со счетчиком*.

Пример 4.1. Вариант 2.

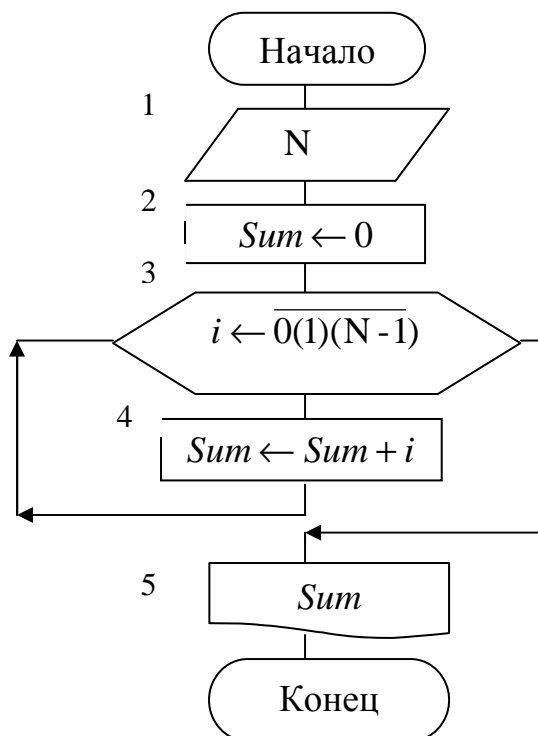


Рис. 4.3. Блок-схема примера 4.1.

Во многих задачах встречаются так называемые *циклы по простой переменной*. В этом случае задается только начальное значение переменной, закон (правило), по которому она изменяется, и конечное ее значение. В процессе решения задачи текущее значение переменной каждый раз сравнивается с ее конечным значением. Выход из цикла происходит, когда переменная достигла своего конечного значения или впервые превысила его. В таких циклах количество повторений неизвестно, но оно может быть вычислено, поскольку значение переменной изменяется по некоторому закону (пример 4.2).

Пример 4.2. Составить блок-схему нахождения значений функции $f(x) = \sin x$ для значений переменной x , изменяющейся в диапазоне от a до b шагом h (рис. 4.4).

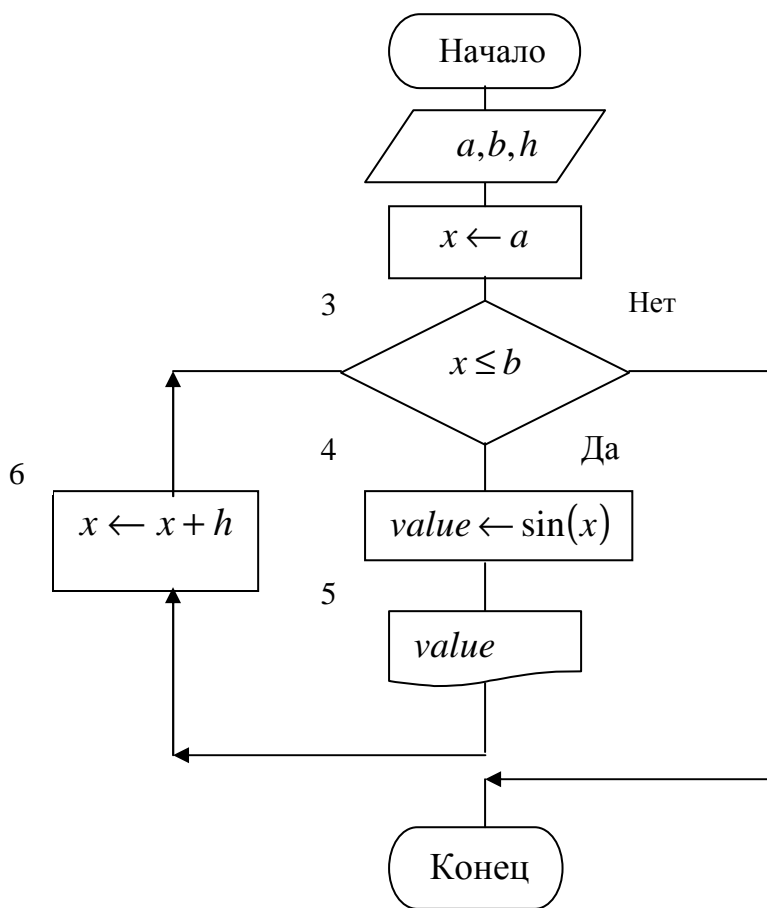
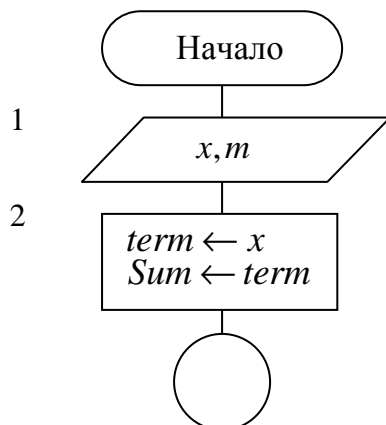


Рис. 4.4. Блок-схема примера 4.2.

Пример 4.3. Составить блок-схему вычисления суммы вида $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + (-1)^{m-1} \frac{x^{2m-1}}{(2m-1)!}$ $x \in R, m \in N$ (рис. 4.5).



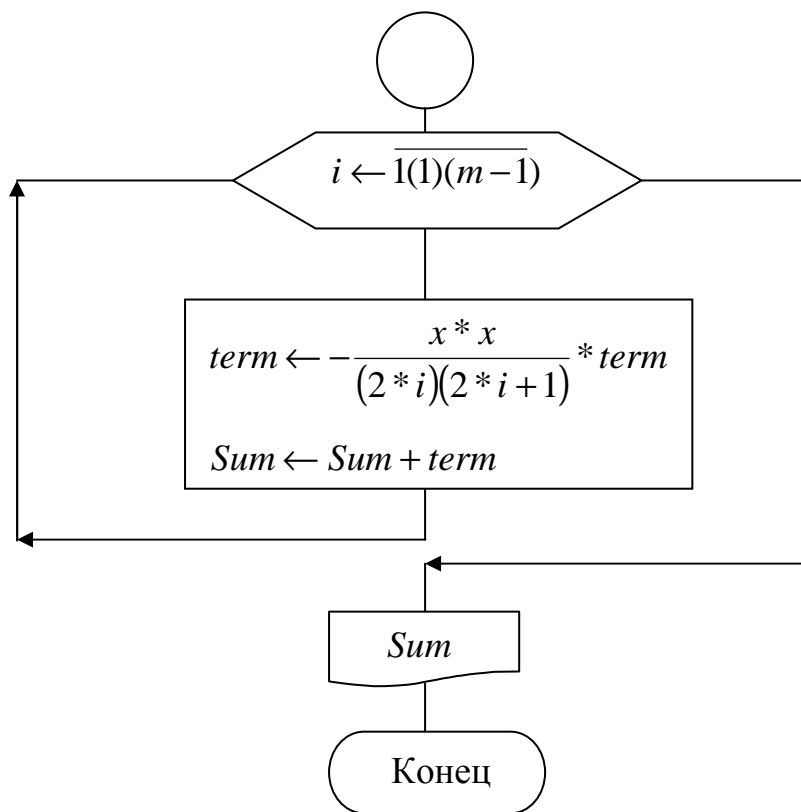


Рис. 4.5. Блок-схема примера 4.3.

Варианты заданий.

1. Дано натуральное n . Вычислить значение суммы.

а) $\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{n}$;

б) 2^n ;

в) $\left(1 + \frac{1}{1^2}\right) \left(1 + \frac{1}{2^2}\right) \dots \left(1 + \frac{1}{n^2}\right)$;

г) $\frac{1}{1 \cdot 2} - \frac{1}{2 \cdot 3} + \dots + \frac{(-1)^{n+1}}{n(n+1)}$;

д) $\frac{1}{1^5} + \frac{1}{2^5} + \dots + \frac{1}{n^5}$;

е) $\frac{1}{3^2} + \frac{1}{5^2} + \dots + \frac{1}{(2n+1)^2}$;

ж) $-\frac{1}{3} + \frac{1}{5} + \dots + \frac{(-1)^n}{2n+1}$;

з) $\frac{1!}{1} + \frac{2!}{1 + \frac{1}{2}} + \dots + \frac{n!}{1 + \frac{1}{2} + \dots + \frac{1}{n}}$;

и) $\underbrace{\sqrt{2 + \sqrt{2 + \dots \sqrt{2}}}}_{n \text{ корней}}$;

к) $\frac{1}{\sin 1} + \frac{1}{\sin 1 + \sin 2} + \dots + \frac{1}{\sin 1 + \sin 2 + \dots + \sin n}$.

2. Дано натуральное n и действительное x . Вычислить значение суммы.
Таблица 4.1.

№	Σ
1	$1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$
2	$\left(\frac{1}{1!} + \sqrt{ x }\right) + \left(\frac{1}{2!} + \sqrt{ x }\right) + \dots + \left(\frac{1}{n!} + \sqrt{ x }\right)$
3	$\left(1 + \frac{\sin x}{1!}\right)\left(1 + \frac{\sin 2x}{2!}\right) \dots \left(1 + \frac{\sin nx}{n!}\right)$
4	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + (-1)^{m-1} \frac{x^{2m-1}}{(2m-1)!}$
5	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^m \frac{x^{2m}}{(2m)!}$
6	$x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2m-1}}{(2m-1)!}$
7	$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^{2m}}{(2m)!}$
8	$x - \frac{x^2}{2} + \frac{x^3}{3} + \dots + (-1)^{n-1} \frac{x^n}{n}$
9	$x - \frac{x^3}{3} + \frac{x^5}{5} + \dots + (-1)^{m-1} \frac{x^{2m-1}}{2m-1}$
10	$-x - \frac{x^2}{2} - \frac{x^3}{3} - \dots - \frac{x^n}{n}$

3. Вычислить значения функции $f(x)$ на отрезке $[a;b]$ с шагом h .

Таблица 4.2.

№	$f(x)$	$[a,b]$	h
1	2	3	4
1	$\ln(x)$	1;1.5	0.1
2	$1 + \ln^2(x)$	0.4;1	0.1
3	$1 + e^x$	0.5;0.6	0.01
4	$e^{x^2}/2$	2;3	0.2
5	$\cos(x)e^{-x}$	1;2	0.2
6	$1/(1 + e^{-x})$	3;4	0.2
7	$\sin(x)\operatorname{sh}(x)$	1;5	1
8	$0.5 + \operatorname{sh}^2(x)$	2;3	0.2

1	2	3	4
9	$\sqrt{x}\text{ch}(x)$	3;4	0.2
10	$1/(1+\text{ch}^2(x))$	2;4	0.5
11	$\sqrt{x}\text{sh}(x)$	1;5	1
12	$e^{-x}\text{ch}(x)$	1;4	1
13	$\ln(x^2)$	1;1.4	0.1
14	$x + \ln(x)$	1;5	1
15	$1/(1 + \sin x)$	$\pi/3; \pi/6$	$\pi/10$
16	$\sin x + \sqrt{x}$	$\pi/6; \pi/4$	$\pi/10$
17	$x(1 - \cos x)$	0.4;0.8	0.1
18	$e^{x+3} \sin x$	0;2	0.5
19	$\cos(x)\text{ch}(x)$	1;5	1
20	$e^{1+x} \text{sh}(x)$	1;4	1

Лабораторная работа № 5.

Тема: «Действия над одномерными массивами в блок-схемах»

Теоретические сведения. Наиболее широко известная структура данных – *массив*. Массив состоит из компонент одного типа, называемого *базовым*. Поэтому структура массивов *однородна*. Кроме того, массивы относят к так называемым структурам с *прямым доступом*. Для того чтобы обозначить отдельную компоненту, к имени всего массива добавляется индекс. *Индекс* – это значение специального типа, определенного как тип индекса массива. Количество индексов называется *размерностью* массива. Отметим, что в памяти ПЭВМ каждой компоненте массива отводится отдельное поле равных размеров, при этом, все элементы массива расположены подряд.

Если x является переменной-массивом размерности n , то отдельная компонента обозначается с помощью имени массива, за которым следует индекс требуемой компоненты – x_i . Иногда компоненты массивов называют переменными с индексами.

Обычный прием работы с массивами, в особенности с большими массивами – выборочное изменение отдельных его компонент, а не конструирование полностью нового составного значения. При этом переменная-массив рассматривается как массив составляющих переменных, и возможно присваивание значений отдельным компонентам, например, $x_i \leftarrow 0.456$.

Тот факт, что индексы массива относятся к определенному (скалярному) типу, имеет важное следствие: индексы массива можно вычислять. На место индексирующей константы можно подставлять любое индексирующее выражение; оно будет вычислено, и результат идентифицирует требуемую компоненту. Если необходимо выполнить некоторую операцию над всеми

компонентами массива или над соседними компонентами некоторой части массива, то удобно воспользоваться циклом (пример 5.1). Поскольку в языке Си индексация элементов массива производится от нуля, то при построении блок-схем будем использовать этот факт.

Пример 5.1. Составить блок-схему нахождения в одномерном массиве из n элементов порядковых номеров максимального и минимального элементов. Значение элементов и их порядковые номера вывести на экран (рис. 5.1).

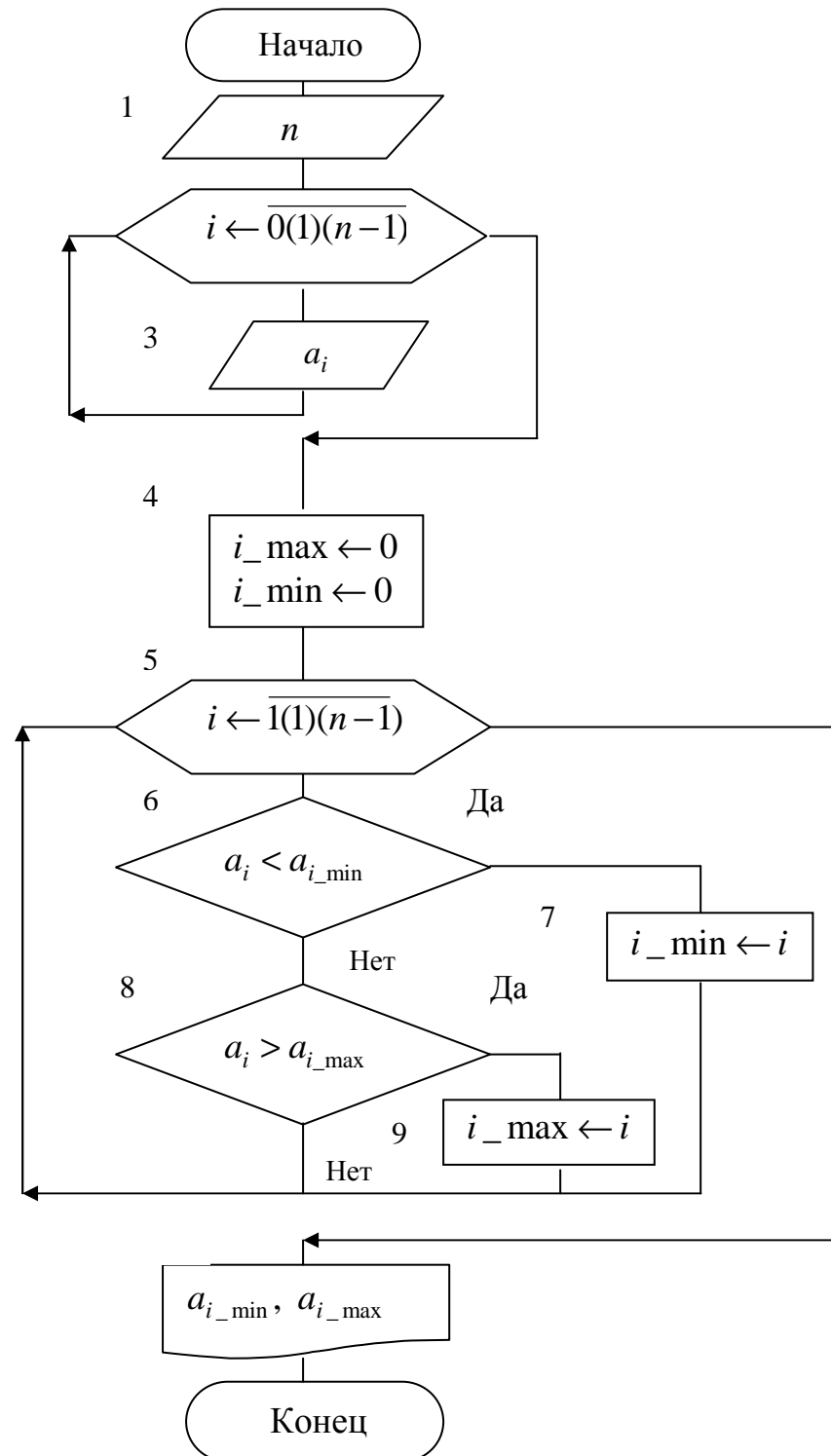


Рис. 5.1. Блок-схема примера 5.1.

Пример 5.2. Составить блок-схему вычисления значения полинома $P_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_n$ в заданной точке x . Коэффициенты, степень полинома, а также значение x ввести (рис. 5.2).

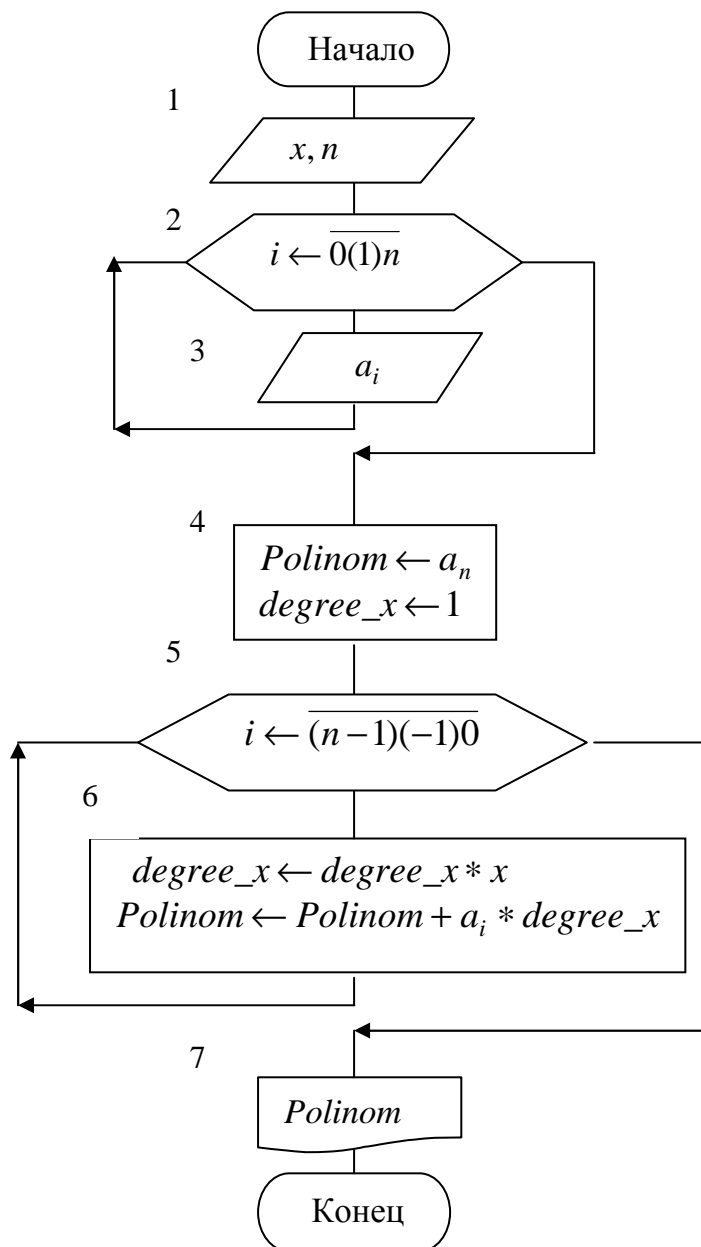


Рис. 5.2. Блок-схема примера 5.2.

Сортировка массивов. В общем случае сортировку следует понимать как перегруппировку заданного множества объектов (массива, файла и т.д.) в некотором порядке. Цель сортировки – облегчить последующий поиск элементов в таком отсортированном множестве. Выбор алгоритма сортировки зависит от структуры обрабатываемых данных. Рассмотрим основные методы сортировки применительно к одномерным массивам.

Дан одномерный массив размерности n : a_0, a_1, \dots, a_{n-1} . Сортировка данного массива есть перестановка его элементов в массив $a_{k_0}, a_{k_1}, \dots, a_{k_{n-1}}$, где при некоторой упорядочивающей функции f выполняются отношения $f(a_{k_0}) \leq f(a_{k_1}) \leq \dots \leq f(a_{k_{n-1}})$. Обычно упорядочивающая функция не вычисляется, а хранится как явная компонента каждого элемента. Ее значение называется ключом элемента. Основное условие при выборе метода сортировки массивов – выбранный метод должен экономно использовать доступную память. Это предполагает, что перестановки, приводящие элементы в порядок, должны выполняться на том же месте («in site»). Методы сортировки «in site» можно разбить в соответствии с определяющими их принципами на три основные типа – сортировка с помощью прямого включения; сортировка с помощью прямого выбора; сортировка с помощью прямого обмена.

Сортировка с помощью прямого включения. Метод основан на следующих принципах. Элементы условно делятся на «уже готовую» последовательность и исходную последовательность. При каждом шаге, начиная с $i=1$ и увеличивая i каждый раз на единицу, из исходной последовательности извлекается i -ый элемент и перекладывается в готовую последовательность, при этом, он вставляется на нужное место.

Сортировка с помощью прямого выбора. Метод основан на следующих принципах. Выбирается элемент с наименьшим значением. Он меняется с первым элементом x_0 . Затем этот процесс повторяется с оставшимися $n-1$ элементами, $n-2$ элементами и т.д. до тех пор, пока не останется один, самый больший элемент.

Сортировка с помощью прямого обмена (метод «пузырька»). Метод основан на сравнении и смене мест пары соседних элементов и продолжении этого процесса до тех пор, пока не будут упорядочены все элементы. Проходы по массиву повторяются, сдвигая каждый раз наименьший элемент оставшейся последовательности к левому концу массива. Если рассматривать массивы как вертикальные, то элементы можно интерпретировать как пузырьки в чане с водой разного веса. В этом случае при каждом проходе один пузырек поднимается до уровня, соответствующего его весу. Такая сортировка известна под названием «пузырьковая сортировка».

Каждый из изложенных выше методов сортировки допускает оптимизацию, основанную на анализе массива (отсортирован или нет), полученного на каждом шаге алгоритма.

Пример 5.3. Составить блок-схему сортировки по неубыванию одномерного целочисленного массива размерности n методом «пузырька» (рис. 5.3).

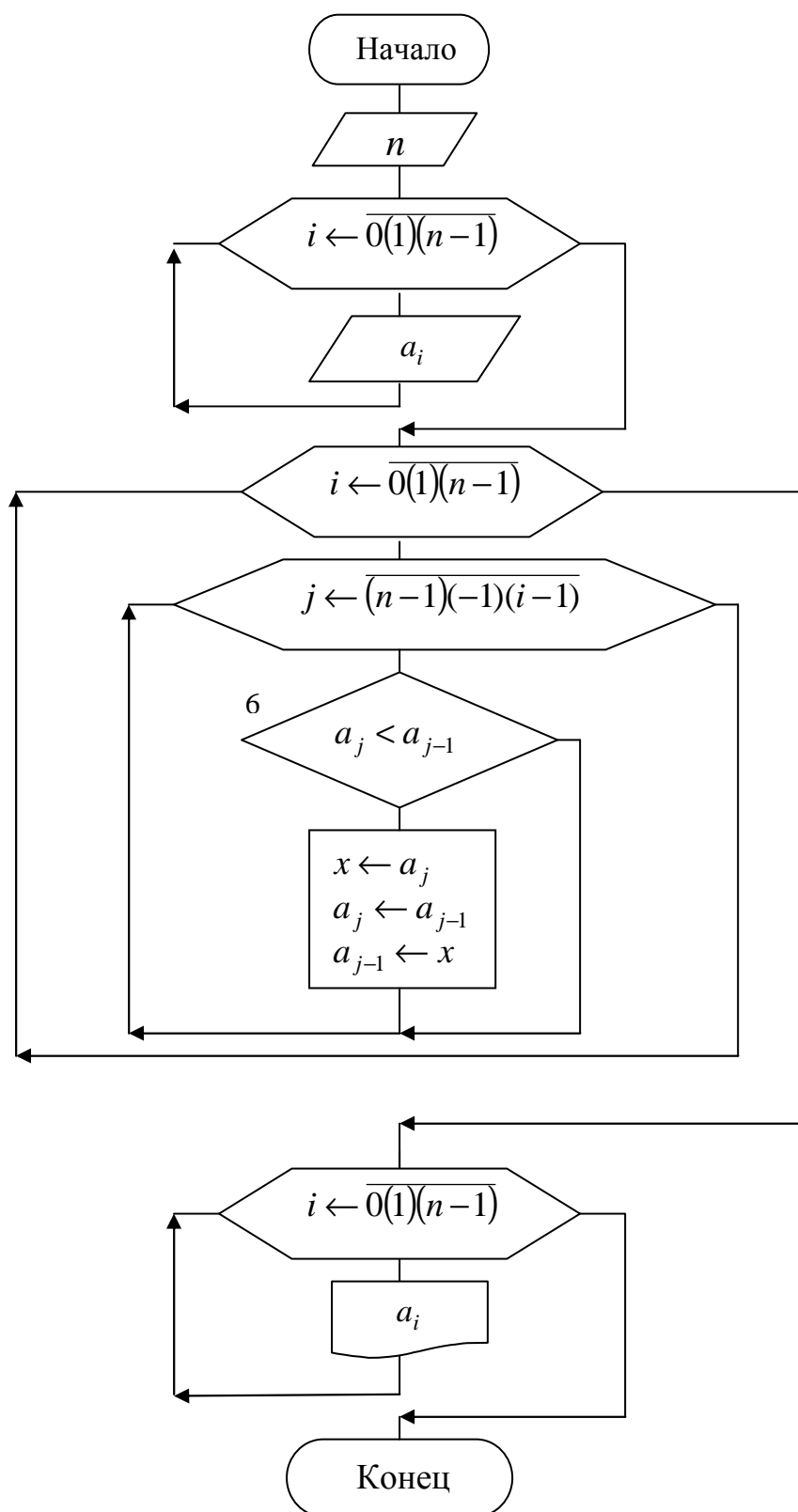


Рис. 5.3. Блок-схема примера 5.3.

Варианты заданий.

1. В одномерном массиве из n элементов найти порядковые номера первого отрицательного и последнего положительного элементов (если таковые

имеются). Значение элементов и их порядковые номера вывести на экран или выдать соответствующее сообщение.

2. Ввести одномерный массив из n элементов. Вычислить сумму всех отрицательных чисел, их количество и сумму всех положительных чисел.

3. В зависимости от того, образуют элементы заданного массива целых чисел из n элементов строго убывающую, не возрастающую, строго возрастающую, неубывающую последовательность, выдать соответствующее сообщение.

4. Ввести одномерный массив из n элементов. Сформировать на его месте новый массив, в котором первым элементом будет последний элемент старого, вторым – предпоследний и т.д.

5. Элемент называется локальным минимумом (максимумом), если у него нет соседа, меньшего (большего), чем он сам. Найти все локальные минимумы и максимумы в заданном массиве из n элементов.

6. В неубывающей последовательности из n элементов найти количество элементов, меньших заданного числа и напечатать их.

7. Элементы одномерного массива из n элементов циклически сдвинуть на k мест влево (вправо).

8. Выполнить попарное суммирование произвольного конечного ряда чисел следующим образом. На первом этапе суммировать попарно рядом стоящие числа, на втором - результаты первого этапа аналогичным образом, и т.д., пока не останется одно число.

9. Ввести одномерный массив a_0, a_1, \dots, a_{n-1} . Вычислить все суммы вида $S_i = a_i + a_{i+1} + \dots + a_j$, $0 \leq i \leq j \leq n-1$ и среди них определить максимальную сумму.

10. Даны два вектора размерности n . Вычислить их скалярное произведение.

11. Дан вектор размерности n . Пронормировать его по максимальному элементу.

12. Сложить два полинома заданных степеней (коэффициенты хранятся в массивах).

13. Умножить два полинома заданных степеней (коэффициенты хранятся в массивах).

14. В массиве из n элементов выбрать без повторений все элементы, встречающиеся более одного раза.

15. Ввести одномерный массив из n элементов. Определить число различных элементов в нем.

16. Вычислить коэффициенты от 1-ой до m -ой ($m \leq n$) производной для полинома степени n $P_n(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \dots + a_n$.

17. Ввести одномерный массив из n элементов. Отсортировать массив по неубыванию (невозрастанию) методом прямого включения.

18. Ввести одномерный массив из n элементов. Отсортировать массив по неубыванию (невозрастанию) методом прямого выбора.

19. Получить упорядоченный по возрастанию массив $C(n+m)$ путем слияния массивов $A(n)$ и $B(m)$ ($\forall n, m$); массивы $A(n)$ и $B(m)$ предварительно упорядочить по возрастанию.

Лабораторная работа № 6.

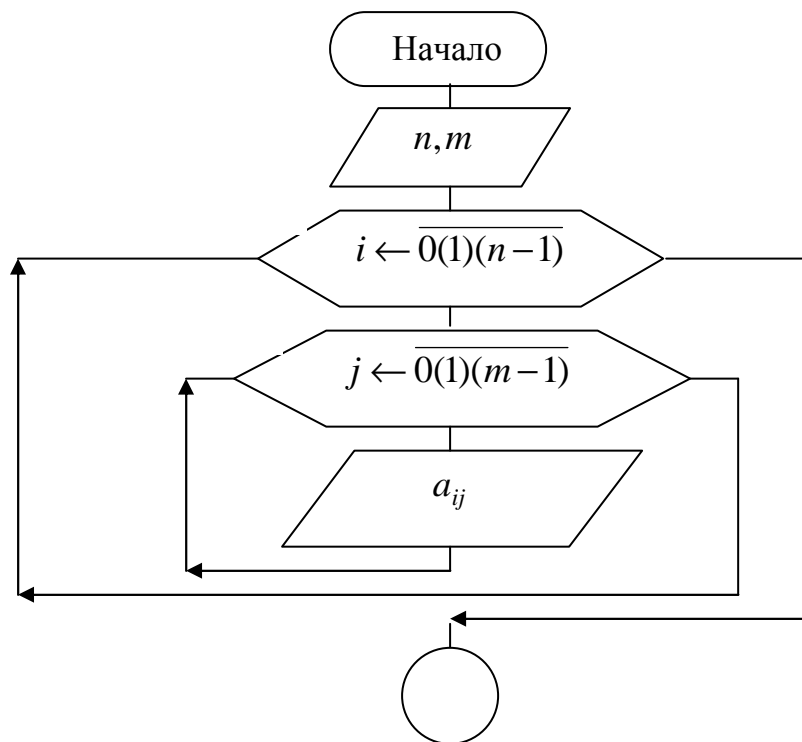
Тема: «Действия над матрицами в блок-схемах»

Теоретические сведения. Матрицы представляются двумерным массивом или одномерным массивом, элементами которого являются одномерные массивы. Матрицу удобно трактовать как таблицу из n строк и m

столбцов, записанную в виде: $A = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{0m-1} \\ a_{10} & a_{11} & \dots & a_{1m-1} \\ \dots & \dots & \dots & \dots \\ a_{n-1,0} & a_{n-1,1} & \dots & a_{n-1,m-1} \end{bmatrix}$. Элементы

матрицы A размерности $n \times m$ можно записать в виде a_{ij} , $i = \overline{0, n-1}$, $j = \overline{0, m-1}$. Здесь первый индекс i указывает положение элемента матрицы в строке, а второй индекс j - в столбце. Очевидно, что для просмотра всех элементов матрицы необходимо использование вложенных циклов.

Пример 6.1. Дана матрица $A(n \times m)$. Составить блок-схему нахождения первого по порядку максимального элемента матрицы (рис. 6.1.).



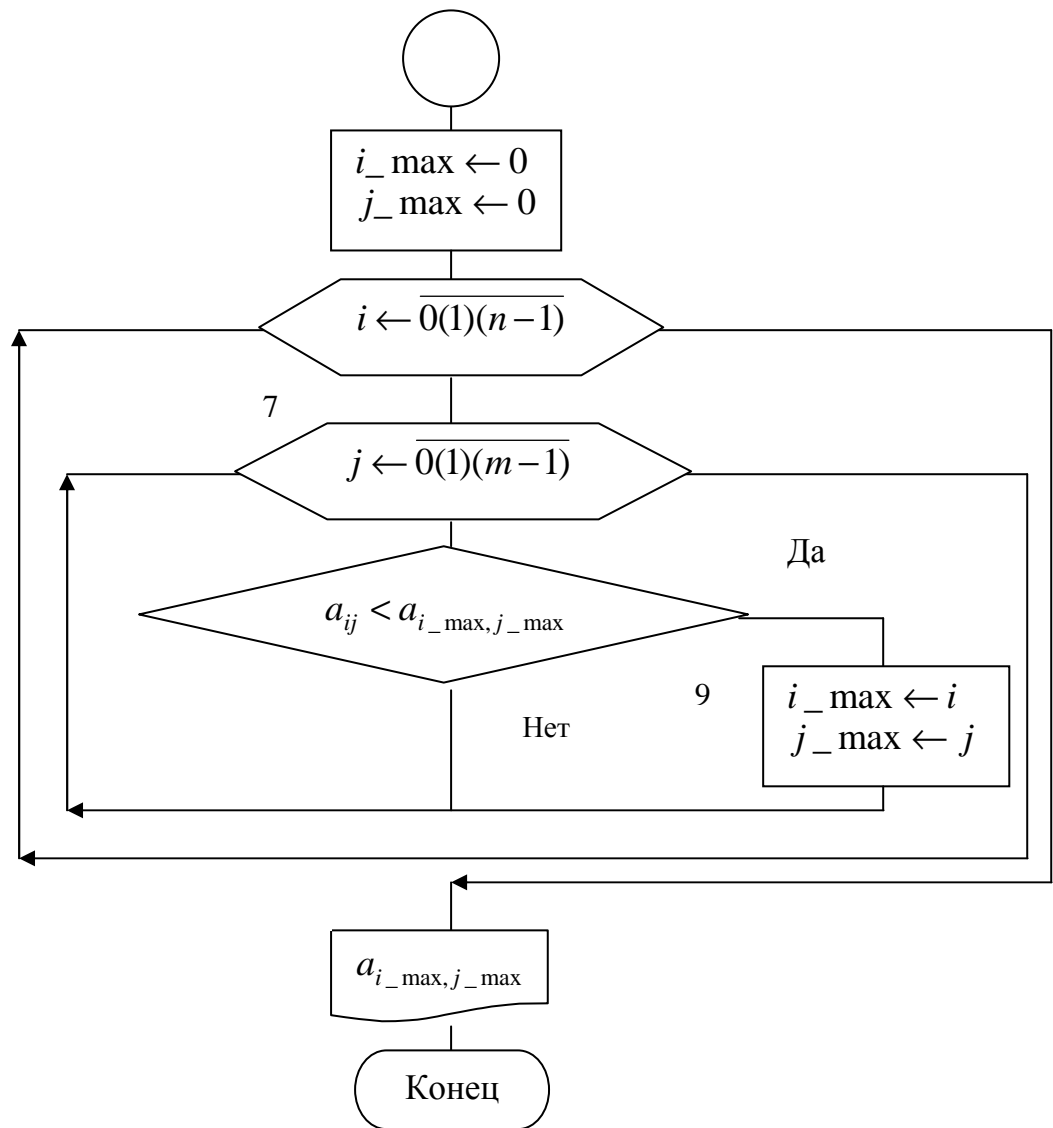
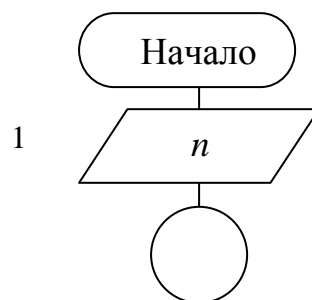


Рис. 6.1. Блок-схема примера 6.1.

Пример 6.2. Дана квадратная матрица $A(n \times n)$. Транспонировать матрицу без использования вспомогательного массива (на том же месте) (рис. 6.2).



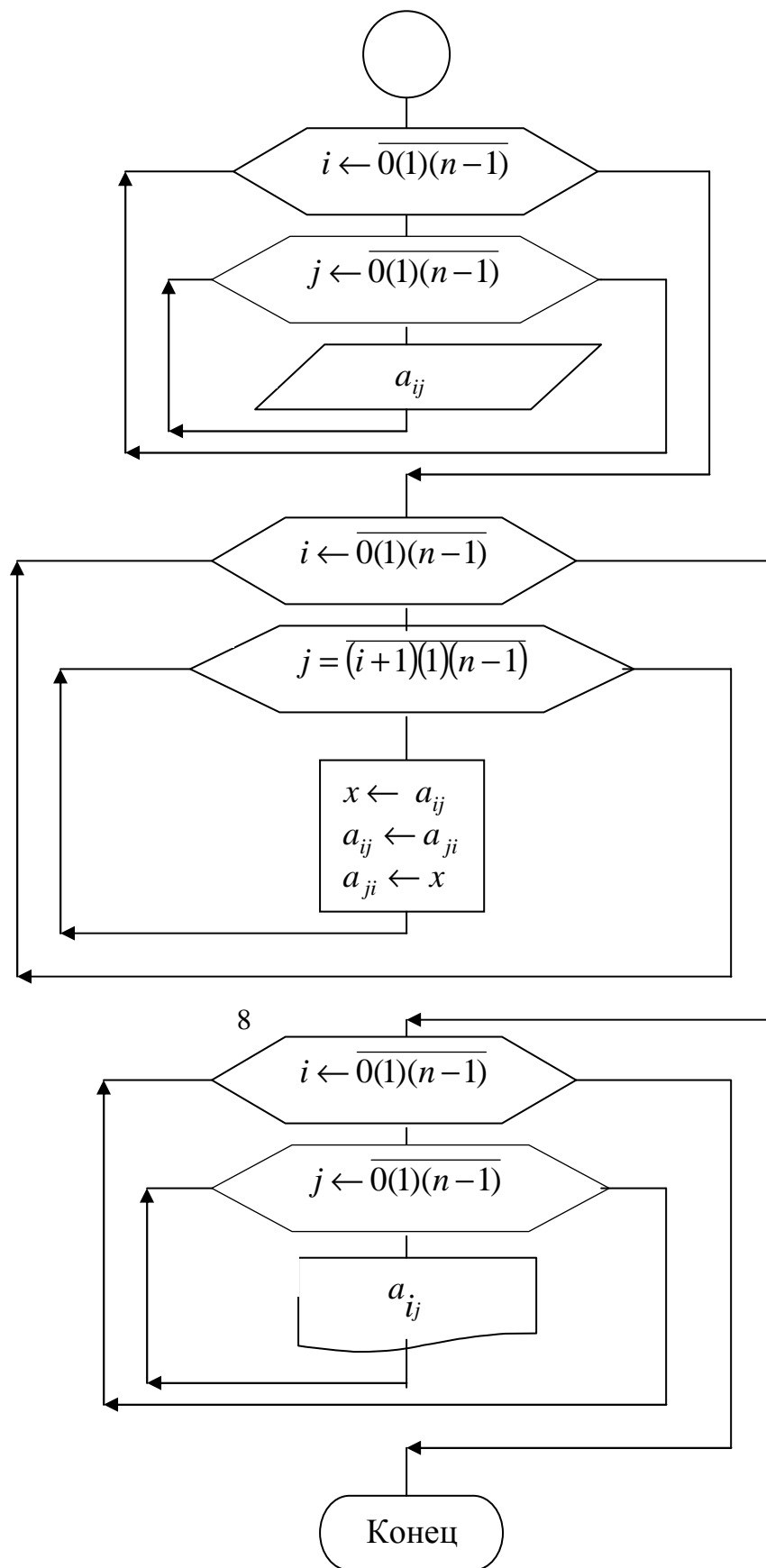


Рис. 6.2. Блок-схема примера 6.2.

Варианты заданий.

1. Дана матрица $A(n \times m)$. Найти порядковые номера первого отрицательного и последнего положительного элемента (если таковые имеются). Значение элементов и их порядковые номера вывести на экран или выдать соответствующее сообщение.
2. Дана матрица $A(n \times m)$. Найти количество отрицательных, положительных и нулевых элементов в ней.
3. Дано натуральное число n . Построить матрицу $A(n \times n)$ вида:

а)

$$\begin{bmatrix} n & 0 & \dots & 0 \\ n-1 & n & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 1 & 2 & \dots & n \end{bmatrix}$$

б)

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 0 & 1 & 1 & \dots & 1 & 1 & 0 \\ 0 & 0 & 1 & \dots & 1 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 1 & \dots & 1 & 0 & 0 \\ 0 & 1 & 1 & \dots & 1 & 1 & 0 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \end{bmatrix}$$

в)

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 1 & 0 & \dots & 0 & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 1 & 0 & \dots & 0 & 1 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}$$

г)

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 & n \\ 0 & 2 & 0 & \dots & 0 & n-1 & 0 \\ 0 & 0 & 3 & \dots & n-2 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 3 & \dots & n-2 & 0 & 0 \\ 0 & 2 & 0 & \dots & 0 & n-1 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 & n \end{bmatrix}$$

4. Дан массив $A(n)$. Построить матрицу $A(n \times n)$ вида

$$\begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \\ a_2 & a_3 & \dots & a_n & a_1 \\ \dots & \dots & \dots & \dots & \dots \\ a_n & a_1 & \dots & a_{n-2} & a_{n-1} \end{bmatrix}.$$

5. По массиву $X(n)$ построить матрицу $A(n \times n)$, где $a_{ij} = x_i \cdot x_j$, $i, j = \overline{1, n}$.

Вычислить $\sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2}$.

6. По массиву $X(n)$ построить матрицу $A(n \times n)$, где $a_{ij} = x_i \cdot x_j$, $i, j = \overline{1, n}$.
 Вычислить $\max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$.
7. Рассматривая матрицу $A(n \times n)$ как составленную из четырех квадратов, переставить в ней друг с другом центрально-симметричные квадраты.
8. Рассматривая матрицу $A(n \times n)$ как составленную из четырех квадратов, переставить в ней друг с другом квадраты по кругу.
9. Дана матрица $A(n \times m)$. Найти матрицу, полученную перестановкой столбцов (первого с последним, второго с предпоследним и т.д.) из данной.
10. Дана матрица $A(n \times m)$. Найти матрицу, полученную перестановкой строк (первой с последней, второй с предпоследней и т.д.) из данной.
11. Найти все седловые точки матрицы $A(n \times m)$. Матрица имеет седловую точку a_{ij} , если a_{ij} является минимальным элементом в i -ой строке и максимальным элементом в j -ом столбце.
12. В матрице $A(n \times m)$ найти максимальный элемент и путем перестановки строк и столбцов поместить его на место элемента a_{00} .
13. Среди строк целочисленной матрицы $A(n \times m)$, содержащих только четные элементы, найти строку с максимальной суммой модулей элементов.
14. Среди столбцов целочисленной матрицы $A(n \times m)$, содержащих только положительные элементы, найти столбец с минимальным произведением элементов.
15. Известно, что в каждой строке и каждом столбце квадратной матрицы имеется единственный отрицательный элемент. Переставить строки матрицы так, чтобы в полученной матрице отрицательные элементы находились на главной диагонали.
16. Известно, что в матрице $A(n \times m)$ нет нулей. Заменить повторяющиеся в матрице элементы нулями.
17. Дана матрица $A(n \times n)$, элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
18. Найти максимальный (минимальный) элемент среди тех строк матрицы $A(n \times m)$, элементы которых упорядочены по неубыванию (невозрастанию).
19. Найти минимальный (максимальный) элемент среди тех столбцов матрицы $A(n \times m)$, элементы которых упорядочены по невозрастанию (неубыванию).
20. Расположить строки матрицы $A(n \times m)$ в порядке убывания модулей сумм элементов строк.
21. Из матрицы $A(n \times m)$ получить на том же месте матрицу, вычеркнув из исходной матрицы строку и столбец, на пересечении которых находится первый по порядку максимальный элемент.

22. Заполнить матрицу $A(n \times n)$ по спирали массивом натуральных чисел $1, 2, \dots, n \times n$, начиная с элемента, находящегося в верхнем левом углу.
23. Начиная с центра, обойти по спирали все элементы квадратной матрицы $A(n \times n)$, распечатывая их в порядке обхода.
24. Характеристикой строки целочисленной матрицы $A(n \times m)$ назовем сумму её положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом их характеристик.
25. Характеристикой столбца целочисленной матрицы $A(n \times m)$ назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом их характеристик.
26. В матрице $A(n \times n)$ элементы каждой строки верхнего треугольника упорядочить по возрастанию, нижнего - по убыванию, диагональные - оставить без изменений.
27. Для заданной матрицы $A(n \times n)$ найти максимум среди сумм элементов диагоналей, параллельных главной диагонали, и минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали.
28. В матрице $A(n \times m)$ найти первый столбец, не содержащий отрицательных элементов, и умножить его как вектор на матрицу A .

Лабораторная работа № 7.

Тема: «Построение блок-схем итерационных вычислительных процессов»

Теоретические сведения. Существует ряд циклических вычислительных процессов, называемых *итерационными*. Итерационный процесс продолжается до тех пор, пока разность между соседними, уточняемыми на каждом шаге цикла значениями (итерациями), не окажется меньше либо равной некоторой заданной величине (точности). Характерной особенностью итерационного процесса является то, что количество циклов (итераций) в нем заранее не известно и становится определенным только после окончания вычислений. Выход из цикла происходит тогда, когда анализируемая величина или величины на очередной итерации отличаются от эталонных величин (например, заданной точности). В таком цикле, как правило, результаты вычислений предыдущего шага цикла используются как исходные данные при выполнении следующего шага цикла (пример 7.1).

Пример 7.1. Составить блок-схему нахождения приближенного значения квадратного корня $y = \sqrt{x}$, $x > 0$ как предела последовательности

$y_1, y_2, \dots, y_k, \dots$, где $y_{k+1} = \frac{1}{2} \left(y_k + \frac{x}{y_k} \right)$. Точность полученного приближения

оценить по величине отклонения двух соседних приближений y_k и y_{k+1} , $\Delta_{k+1} = |y_k - y_{k+1}| \leq \varepsilon$, где $\varepsilon > 0$ – заданная точность вычислений (рис. 7.1).

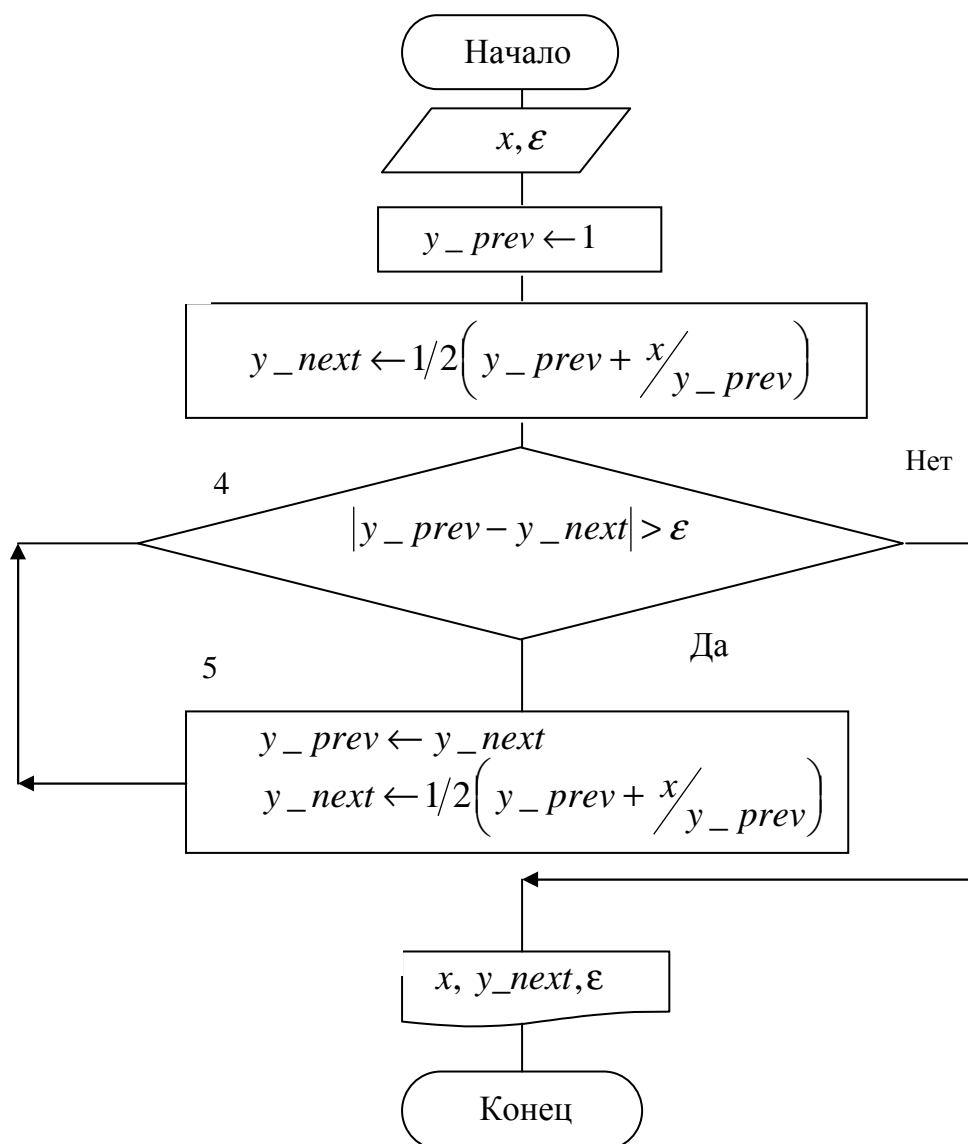


Рис. 7.1. Блок-схема примера 7.1.

Пример 7.2. Составить блок-схему вычисления с точностью $\varepsilon > 0$ отрезка степенного ряда для функции $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots + (-1)^{m-1} \frac{x^{2m-1}}{(2m-1)!} + \dots$, $x \in R$. Точность полученного значения считать достигнутой, если последний член ряда не превосходит по абсолютной величине заданного ε (рис. 7.2).

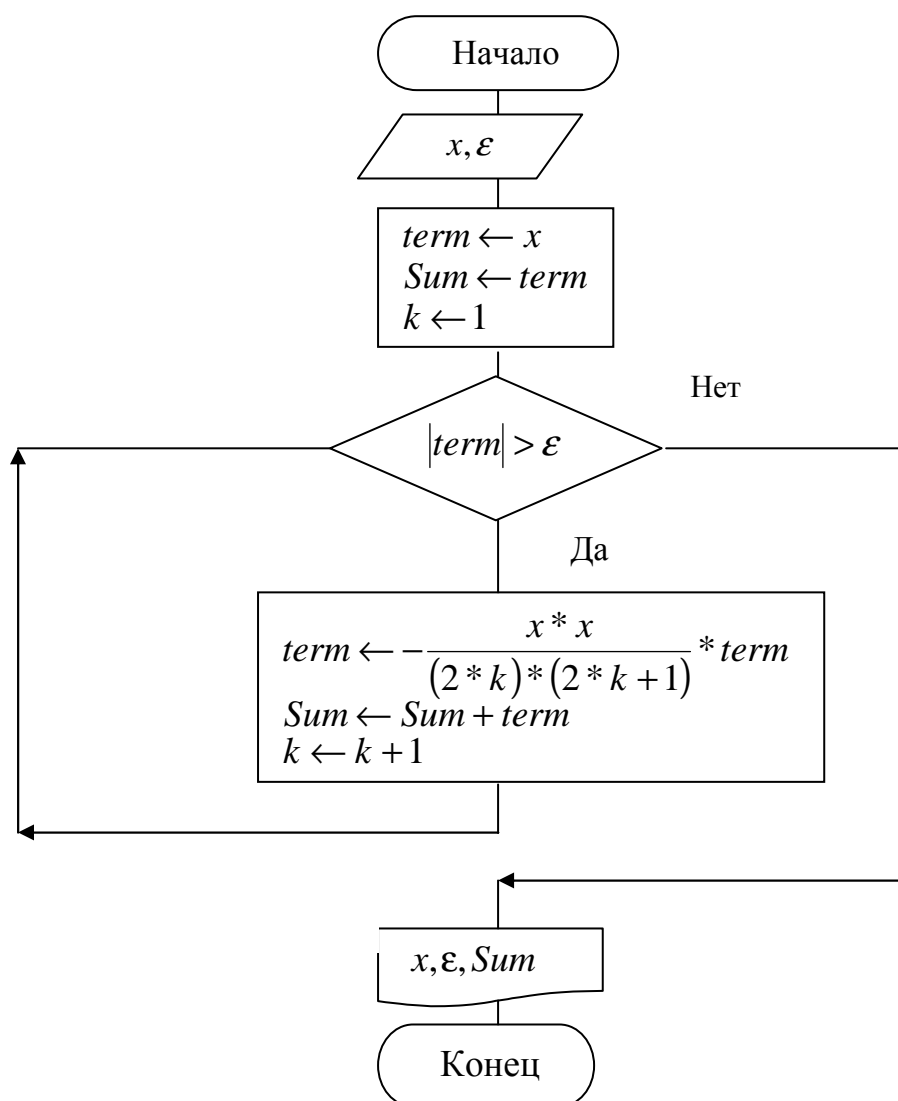


Рис. 7.2. Блок-схема примера 7.2.

Варианты заданий.

1. Даны действительные числа x, ε ($x \neq 0, \varepsilon > 0$). Вычислить приближенное значение бесконечной суммы. Вычисления выполнить с заданной точностью ε (пока текущий член ряда не превосходит по абсолютной величине заданного ε).

а) $\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{k!(2k+1)}$;

б) $\sum_{k=0}^{\infty} \frac{(-1)^k x^{4k+3}}{(2k+1)!(4k+3)}$;

в) $\sum_{k=0}^{\infty} \frac{(-1)^k x^{4k+1}}{(2k)!(4k+1)}$;

г) $\sum_{k=0}^{\infty} \frac{(-1)^k}{((k+1)!)^2} \left(\frac{x}{2}\right)^{2(k+1)}$;

д) $\sum_{k=0}^{\infty} \frac{(-1)^k}{k!(k+1)!} \left(\frac{x}{2}\right)^{2k+1}$;

е) $\sum_{k=0}^{\infty} \frac{(-1)^{k+1}}{(2k)!} \left(\frac{x}{3}\right)^{4k}$;

ж) $\sum_{k=0}^{\infty} \frac{(-1)^{k+1} x^{2k-1}}{(2k-1)(2k+1)!}$;

з) $\sum_{k=0}^{\infty} \frac{x^{2k}}{2^k k!}$;

и) $\sum_{k=1}^{\infty} \frac{1}{x^3 k^2}$;

$$\begin{array}{lll}
\text{к)} \sum_{k=1}^{\infty} \frac{x^{2k}}{x^2 + k^3}; & \text{л)} \sum_{k=0}^{\infty} \frac{(-x)^{2k}}{2k!}; & \text{м)} \sum_{k=0}^{\infty} \frac{(-1)^k x^k}{(k+1)^2}; \\
\text{н)} \sum_{k=0}^{\infty} \frac{(-1)^k (k+1)x^k}{3^k}; & \text{о)} \sum_{k=0}^{\infty} \frac{(-1)^k x^{k+2}}{(k+1)(k+2)!}; & \text{п)} \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{(2k+1)!} \left(\frac{x}{3}\right)^{4k+2}.
\end{array}$$

2. Даны действительные числа x, ε ($\varepsilon > 0$). Вычислить с заданной точностью ε приближенное значение бесконечной суммы и сравнить его с точным.

Таблица 7.1.

№	Σ	Точное значение
1	$1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots$	$e^x, x \in R$
2	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^m \frac{x^{2m}}{(2m)!} + \dots$	$\cos x, x \in R$
3	$x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2m-1}}{(2m-1)!} + \dots$	$\operatorname{sh} x, x \in R$
4	$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^{2m}}{(2m)!} + \dots$	$\operatorname{ch} x, x \in R$
5	$x - \frac{x^2}{2} + \frac{x^3}{3} - \dots + (-1)^{n-1} \frac{x^n}{n} + \dots$	$\ln(1+x),$ $x \in]-1;1]$
6	$x - \frac{x^3}{3} + \frac{x^5}{5} - \dots + (-1)^{m-1} \frac{x^{2m-1}}{2m-1} + \dots$	$\operatorname{arctg} x, x \in [-1;1]$
7	$-x - \frac{x^2}{2} - \frac{x^3}{3} - \dots - \frac{x^n}{n} - \dots$	$\ln(1-x),$ $x \in [-1;1[$
8	$1 - x + x^2 - \dots + (-1)^n x^n + \dots$	$\frac{1}{1+x}, x \in]-1;1[$
9	$1 + x + x^2 - \dots + x^n + \dots$	$\frac{1}{1-x}, x \in]-1;1[$
10	$1 + \alpha x + \frac{\alpha(\alpha-1)}{2!} x^2 + \dots + \frac{\alpha(\alpha-1)\dots(\alpha-n+1)}{n!} x^n + \dots$	$(1+x)^\alpha, x \in]-1;1[$
11	$2 \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots + \frac{x^{2n-1}}{2n-1} + \dots \right)$	$\ln \frac{1+x}{1-x}, x \in]-1;1[$
12	$1 - \frac{1}{2}x + \frac{3}{8}x^2 - \frac{5}{16}x^3 + \dots + (-1)^{n-1} \frac{(2n-3)!!}{(2n)!!} x^n + \dots$	$\frac{1}{\sqrt{1+x}}, x \in [-1;1]$

3. Вычислить предел последовательности $\{a_n\}_{n=0}^{\infty}$, где $a_n = \frac{n}{\sqrt{n^2+1} + \sqrt{n^2-1}}$.

За предел принять такое значение a_n , при котором $|a_{n+1} - a_n| \leq \varepsilon$. Указать, на каком шаге был получен искомый предел с заданной точностью ε .

4. Вычислить с заданной точностью ε корень n -ой степени из положительного числа a как предел числовой последовательности $y_1, y_2, \dots, y_k, \dots$, построенной по формуле последовательных приближений $y_{k+1} = \frac{n-1}{n} y_k + \frac{a}{ny_k^{n-1}}, k=0,1,\dots; y_0 = \frac{a+n-1}{2}$.

5. Найти наименьшее трехзначное из чисел Фибоначчи. Последовательность чисел $\{F_n\}_{n=0}^{\infty}$, где $F_0 = 1, F_1 = 1, F_n = F_{n-1} + F_{n-2}$ называется последовательностью Фибоначчи.

6. Найти на отрезке $[a, b]$ корень нелинейного уравнения вида $y = f(x)$ с точностью ε методом деления отрезка пополам.

Лабораторная работа № 8. **Тема: «Условный оператор if....»**

Обобщенная формулировка задания. Выдать соответствующее сообщение о принадлежности заданной точки $M(x, y)$ замкнутой области D .

Пример выполнения задания. Определить, попадает ли точка внутрь круга, если радиус круга известен, а его центр и точка задаются своими координатами.

Текст программы.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define TRUE 1

int main(void)
{
    float XCircle, YCircle, Radius, XPoint, YPoint;

    printf("\nEnter the coordinates of the center "
           "circle: ");
    scanf("%f%f",&XCircle,&YCircle);
    printf("\nEnter the radius of circle: ");
    scanf("%f",&Radius);
    printf("\nEnter the coordinates of the point: ");
```

```

scanf ("%f%f", &XPoint, &YPoint);

clrscr();
if ( pow(XPoint-XCircle,2)+pow(YPoint-YCircle,2) <=
    pow(Radius,2) )
{
    printf("\nThe point with coordinates "
           "(%.2f,%.2f) belong to circle\n with "
           "coordinates of the centre (%.2f,%.2f) "
           "and radius %.2f", XPoint, YPoint,
           XCircle, YCircle, Radius);
}
else
{
    printf("\nThe point with coordinates "
           "(%.2f,%.2f) don't belong to circle\n"
           "with coordinates of the centre "
           "(%.2f,%.2f) and radius %.2f ", XPoint,
           YPoint, XCircle, YCircle, Radius);
}

printf("\nPress any key to exit...");
getch();
return 0;
}

```

Варианты заданий. См. «Лабораторная работа № 3».

Лабораторная работа № 9. Тема: «Оператор цикла **for**...»

Обобщенная формулировка задания. Дано натуральное n (и действительное x). Вычислить значение указанной суммы.

Пример выполнения задания. Даны натуральное n и действительное x .
Вычислить значение суммы по формуле $\sum_{i=0}^n \frac{x^i}{i!}$.

Текст программы.

```

#include <stdio.h>
#include <conio.h>
#include <math.h>

#define TRUE 1

```

```

int main(void)
{
    double X, Sum, Term;
    unsigned Number, i;

    while (TRUE)
    {
        printf("Enter the value variable x and number of
                items:");
        scanf("%lf%u",&X,&Number);
        if (Number > 0) break;
        printf("\nParameter is incorrect!!! Try "
                "again!!!\n");
    }

    clrscr();
    Sum = 1, Term = 1;
    for ( i = 1; i < Number; i++)
    {
        Term = Term * X / i;
        Sum = Sum + Term;
    }

    printf("The value of sum is equal: %lf",Sum);

    printf("\nPress any key to exit...");
    getch();
    return 0;
}

```

Варианты заданий. См. «Лабораторная работа № 4», Таблица 4.1.

Лабораторная работа № 10. Тема: «Оператор цикла **while**...»

Обобщенная формулировка задания. Вычислить значения функции $f(x)$ на отрезке $[a;b]$ с шагом h .

Пример выполнения задания. Вычислить значения функции $f(x) = \sin x$ на отрезке $[0;\pi]$ с шагом $\frac{\pi}{10}$.

Текст программы.

```
#include <stdio.h>
```

```

#include <conio.h>
#include <math.h>

#define TRUE 1

int main(void)
{
    double BeginSegm, EndSegm, Step, Term;

    while(TRUE)
    {
        printf("\nEnter the value of regs of segment "
               "and step: ");
        scanf("%lf%lf%lf",&BeginSegm,&EndSegm,&Step);
        if ((BeginSegm < EndSegm) && (Step > 0)) break;
        printf("\nParameters are incorrect!!! Try "
               "again!!!\n");
    }

    clrscr();
    Term = BeginSegm;
    while ( Term <= EndSegm )
    {
        printf("The value of function sin in the point "
               " x=%lf is equal %lf\n",Term,sin(Term));
        Term = Term + Step;
    }

    printf("\nPress any key to exit...");
    getch();
    return 0;
}

```

Варианты заданий. См. «Лабораторная работа № 4», Таблица 4.2.

Лабораторная работа № 11

Тема: «Обработка одномерных массивов»

Обобщенная формулировка задания. Исходный массив ввести с клавиатуры, заполнить алгоритмически или инициализировать случайным образом (в зависимости от постановки задачи). При выполнении задания предусмотреть возможность введения размерности обрабатываемого массива в диапазоне объявленной максимальной размерности. Исходный массив и полученный результат вывести на экран.

Примеры выполнения заданий.

Дан одномерный целочисленный массив. Найти и вывести на экран минимальный и максимальный элементы массива.

Текст программы.

```
#include <conio.h>
#include <stdio.h>

#define TRUE 1

int main(void)
{
    const unsigned DIM = 10;
    int A[DIM];
    unsigned n, i, i_max, i_min;

    while(TRUE)
    {
        printf("Enter n <= %d - dimention of massive:",
              DIM);
        scanf("%u", &n);
        if ((n > 0) && (n <= DIM)) break;
        printf("\n Dimention is incorrect!!! Try "
              "again!!!\n");
    }

    printf("\n Enter the elements of massive:\n");
    for ( i = 0; i < n; i++)
    {
        printf("\nA[%u] = ", i);
        scanf("%d", &A[i]);
    }

    clrscr();
    printf("\t ARRAY \n");
    for ( i = 0; i < n; i++)
    {
        printf(" %d", A[i]);
    }

    i_max = 0, i_min = 0;
    for( i = 0; i < n ; i ++ )
    {
        if ( A[i] < A[i_min] ) i_min = i;
```



```

        else if ( A[i] > A[i_max] ) i_max = i;
    }

    printf("\n The maximun element of array is equal to "
           "A[%u] = %d\n", i_max + 1, A[i_max]);
    printf("\n The minimum element of array is equal to "
           "A[%u] = %d\n\n", i_min + 1, A[i_min]);

    printf("\nPress any key to exit...");
    getch();
    return 0;
}

```

Дан одномерный целочисленный массив. Отсортировать массив по неубыванию методом «пузырька».

Текст программы.

```

#include <conio.h>
#include <stdio.h>

#define TRUE 1

int main(void)
{
    const unsigned DIM = 10;
    int A[DIM], x;
    unsigned n, i, j;

    while(TRUE)
    {
        printf("Enter n <= %d - dimention of massive:",
               DIM);
        scanf("%u", &n);
        if ((n > 0) && (n <= DIM)) break;
        printf("\n Dimention is incorrect!!! Try "
               "again!!!\n");
    }

    printf("\n Enter the elements of massive:\n");
    for ( i = 0; i < n; i++)
    {
        printf("\nA[%u] = ", i);
        scanf("%d", &A[i]);
    }
}

```

```

clrscr();
printf("\tSource Array:\n");
for ( i = 0; i < n; i++)
{
    printf(" %d",A[i]);
}

for( i = 0; i < n ; i ++)
{
    for( j = n - 1; j > i; j --)
    {
        if (A[j] < A[j-1] )
        {
            x = A[j];
            A[j] = A[j-1];
            A[j-1] = x;
        }
    }
}

printf("\tSelected Array:\n");
for ( i = 0; i < n; i++)
{
    printf(" %d",A[i]);
}

printf("\nPress any key to exit...");
getch();
return 0;
}

```

Варианты заданий. См. «Лабораторная работа № 4».

Лабораторная работа № 12

Тема: «Обработка матриц»

Обобщенная формулировка задания. При выполнении задания предусмотреть возможность введения размерности обрабатываемой матрицы в диапазоне объявленной максимальной размерности. Элементы исходной матрицы ввести с клавиатуры, заполнить алгоритмически или инициализировать случайным образом (в зависимости от постановки задачи). Исходную матрицу и полученный результат вывести на экран.

Пример выполнения задания. Дана матрица $A(n \times m)$. Найти и вывести на экран порядковый номер первого по счету максимального элемента

матрицы. Элементы матрицы $A(n \times m)$ вычислить по формуле $a_{ij} = \frac{10}{i+j+1}$,
 $i = \overline{0, n-1}$, $j = \overline{0, m-1}$.

Текст программы.

```
#include <stdio.h>
#include <conio.h>

#define TRUE 1

int main(void)
{
    const unsigned DIM = 10;
    double A[DIM][ DIM];
    unsigned n, m, i, j, i_max, j_max;

    while(TRUE)
    {
        printf("Enter n , m <= %d - dimentions of "
               "matrix: ", DIM);
        scanf("%u%u", &n, &m);
        if ((n > 0) && (n <= DIM) && (m > 0) &&
            (m <= DIM)) break;
        printf("\n Dimentions is incorrect!!! Try "
               "again!!!\n");
    }

    clrscr();
    printf("\n Source Matrix\n");
    for ( i = 0; i < n; i++)
    {
        for ( j = 0; j < m; j++)
        {
            A[i][j] = 10. /( i + j + 1);
            printf("%8.4lf", A[i][j]);
        }
        printf("\n");
    }

    i_max = 0, j_max = 0;
    for ( i = 0; i < n; i++)
    {
        for ( j= 0; j < m; j++)
        {
```

```

        if (A[i][j] > A[i_max][j_max])
        {
            i_max = i;
            j_max = j;
        }
    }
}

printf(" Maximal element A[%d,%d] = "
       "%8.4lf", i_max, j_max, A[i_max][j_max]);

printf("\nPress any key to exit...");
getch();
return 0;
}

```

Варианты заданий. См. «Лабораторная работа № 5».

Лабораторная работа № 13 Тема: «Обработка строк»

Обобщенная формулировка задания. При выполнении задания предусмотреть возможность ввода текста с клавиатуры. При работе с текстом использовать строки длиной не более 80-ти символов. Исходный текст и полученный результат вывести на экран.

Пример выполнения задания. Дан массив из слов длиной не более 20-ти символов. Отсортировать массив в алфавитном порядке методом «пузырька».

Текст программы.

```

#include <conio.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    const int NUMBER_WORDS = 10;
    const int LENTH_WORD = 10;
    char MasWords[NUMBER_WORDS][LENTH_WORD],
        Str [LENTH_WORD] = "";
    int i, j;

    printf("\tEnter array from words\n");
    for( i = 0; i < NUMBER_WORDS; i++)

```

```

{
    gets(MasWords[i]);
}

clrscr();
printf("\tSource array from words\n");
for( i = 0; i < NUMBER_WORDS; i++)
{
    printf(" %d%15s\n", (i + 1), MasWords[i]);
}

for( i = 0; i < NUMBER_WORDS; i ++)
{
    for( j = NUMBER_WORDS - 1; j > i; j --)
    {
        if (strcmp(MasWords[j], MasWords[j-1]) < 0)
        {
            strcpy(Str, MasWords[j]);
            strcpy(MasWords[j], MasWords[j-1]);
            strcpy(MasWords[j-1], Str);
        }
    }
}

printf("\tSelected array from words\n");
for( i = 0; i < NUMBER_WORDS; i ++)
{
    printf("%d%15s\n", (i + 1), MasWords[i]);
}

printf("\nPress any key to exit...");
getch();
return 0;
}

```

Варианты заданий.

1. Напечатать квитанцию об оплате за телеграмму, если известна стоимость одного слова.
2. В тексте слова заданной длины заменить указанной подстрокой, длина которой может не совпадать с длиной слова.
3. В тексте каждую букву заменить ее номером в алфавите.
4. В тексте после указанного символа вставить подстроку.
5. После каждого слова текста, оканчивающегося заданной подстрокой, вставить указанный символ.

6. В тексте удалить указанный символ везде, где он встречается.
7. Из текста удалить все символы, не являющиеся буквами, кроме пробелов.
8. Из текста удалить все слова заданной длины, начинающиеся с согласных букв.
9. Найти, каких букв в тексте больше – гласных или согласных.
10. В тексте найти и вывести на экран все слова максимальной и минимальной длины.
11. В тексте найти и напечатать слова, начинающиеся и оканчивающиеся гласной буквой.
12. В тексте найти и напечатать символы, встречающиеся наиболее часто.
13. В тексте нет слов, начинающихся одинаковыми буквами. Напечатать слова текста в таком порядке, чтобы последняя буква каждого слова совпадала с первой буквой последующего слова. Если все слова нельзя напечатать в таком порядке, найти такую цепочку, состоящую из наибольшего количества слов.
14. Найти наибольшее количество предложений текста, в которых есть одинаковые слова.
15. Напечатать без повторения слова текста, у которых первая и последняя буквы совпадают.
16. Рассортировать слова английского текста в соответствии с частотой встречающейся в нём заданной буквы (по возрастанию). Слова с одинаковым количеством расположить в алфавитном порядке.
17. Рассортировать слова английского текста по возрастанию в них доли гласных букв (отношение количества гласных букв к общему количеству букв в слове).
18. Ввести текст и список слов. Для каждого слова из заданного списка найти, сколько раз оно встречается в тексте, и рассортировать список слов по убыванию количества их встречаемости.
19. В тексте найти и напечатать все пары слов, из которых одно является обращением другого.
20. Найти и напечатать, сколько раз в тексте повторяется каждое слово, которое встречается в нём.
21. Из заданного текста удалить все слова заданной длины. Оставшиеся слова напечатать в алфавитном порядке.
22. В заданном тексте выделить две группы слов: в первую определить слова, начинающиеся на согласную букву, во вторую – на гласную. Напечатать слова каждой из групп в порядке возрастания букв в них.

Лабораторная работа № 14 **Тема: «Табулирование функций»**

Обобщенная формулировка задания. Вычислить m значений заданной функции $f(x)$ на отрезке $[a, b]$. Результаты оформить в виде таблицы 14.1. Значения a, b, ε ввести с клавиатуры.

Таблица 14.1

x_i	$f(x_i)$	$\tilde{f}(x_i)$	Точность	Число итераций
$x_0 = a$				
$x_{m-1} = b$				

Столбцы таблицы: 1- значение x_i ; 2- значения функции $f(x_i)$, вычисленное с использованием библиотечных функций компилятора; 3- значения функции $\tilde{f}(x_i)$, вычисленное с помощью явного разложения в ряд с точностью $\varepsilon = 10^{-5}$; 4- точность вычислений, т.е. $|\tilde{f}(x_i) - f(x_i)|$; 5- количество требуемых для достижения заданной точности итераций.

Для построения таблицы использовать символы ‘*’, ‘-’, ‘Г’ и т.д. или символы псевдографики.

Ограничения.

- Массивов не использовать.
- Факториалы и степени, большие 2, в выражениях для членов ряда в явном виде не применять.
- Количество точек на отрезке $[a, b]$ не менее 10 (разбиение может быть равномерным либо по некоторому правилу).
- Для тригонометрических функций в программе приводить значение аргумента к величине $0 \leq x \leq \pi/2$.
- Для экспоненциальных функций выделять целую часть аргумента, а разложение применять для дробной части.

Комментарии.

1. Для уравнения $y = \sqrt[k]{x}$ ($x > 0, k > 0, k$ – целое) следует использовать рекуррентную формулу Ньютона:

$$y_{n+1} = \frac{1}{k} \left[(k-1)y_n + \frac{x}{y_n^{k-1}} \right], \quad k = 0, 1, \dots; \quad y_0 = \frac{a+n-1}{2},$$

справедливую для $y_0 > 0$. Нужная точность оценивается соотношением $|y_{n+1} - y_n| < \varepsilon$.

2. Использовать таблицу 7.1 для построения разложения элементарных функций в ряд Тейлора.

Пример выполнения задания. Построить таблицу значений функции e^x с точностью ε на отрезке $[0, 1]$ вида 14.1. Для построения таблицы использовать символы псевдографики.

Описание используемых функций. Функция LogUser с прототипом:

```
double ExpUser(double x,double Epsilon);
```

возвращает значение e^x при фиксированном значении x , посчитанное с помощью разложения в ряд Тейлора с точностью `Epsilon`.

Текст программы.

```
#include <conio.h>
#include <stdio.h>
#include <math.h>

#define TRUE 1

#define STR1 "\xCD\xCD\xCD\xCD\xCD"
#define STR2 \
"\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD\xCD"

double ExpUser ( double , double );

int main(void)
{
    double BeginSegm, EndSegm, Step, x, Epsilon;

    while(TRUE)
    {
        printf("Enter a,b,h (type real) and Epsilon "
            "(accuracy)\n");
        scanf("%lf%lf%lf%lf", &BeginSegm, &EndSegm,
            &Step, &Epsilon);
        if ((BeginSegm < EndSegm) && (Step > 0) &&
            (Epsilon < 1) && (Epsilon > 0)) break;
        printf("\nParameters are incorrect!!! Try"
            "again!!!\n");
    }

    clrscr();
    printf("\n Table of function's values \n\n"
        " \xC9"STR1"\xCB"STR2"\xCB"STR2"\xBB\n"
        " \xBA x \xBA Fun_T \xBA Fun_V \xBA\n"
        " \xCC"STR1"\xCE"STR2"\xCE"STR2"\xB9\n");

    for( x = BeginSegm;x <= EndSegm ; x = x + Step)
    {
        printf(" \xBA%4.1f \xBA%10.6f \xBA%10.6f "
            "\xBA\n",x,ExpUser(x, Epsilon),exp(x));
    }
}
```



```

printf(" \xC8"STR1"\xCA"STR2"\xCA"STR2"\xBC\n");

printf("\nPress any key to exit...");
getch();
return 0;
}

double ExpUser(double x,double Epsilon)
{
    double Sum = 1, Term = 1;
    unsigned i;

    for ( i = 0; (fabs(Term) > Epsilon); i++)
    {
        Term = Term * x / ( i + 1 );
        Sum = Sum + Term;
    }

    return Sum;
}

```

Варианты заданий.

Таблица 14.2.

№	$f(x)$	$[a,b]$	№	$f(x)$	$[a,b]$
1	$e^{-x}\sqrt{x}$	1;3	10	$\ln(x)$	1;1.5
2	$\sqrt[3]{x}$	3;4	11	$1 + \ln^2(x)$	0.4;1
3	$x\sqrt{x}$	4;5	12	$1 + e^x$	0.5;1
4	$1/\sqrt[3]{x}$	5;7	13	$e^{x^2}/2$	2;3
5	$1/\sqrt{x}$	6;8	14	$\cos(x)e^{-x}$	1;2
6	$(2x+1)/\sqrt{x}$	8;9	15	$1/(1+e^{-x})$	3;4
7	$\sqrt[4]{x}$	0.5;1	16	$\sin(x)\operatorname{sh}(x)$	1;5
8	$\sqrt{x^2+1}$	2.5;3	17	$0.5 + \operatorname{sh}^2(x)$	2;3
9	$\sqrt[5]{x}\cos x$	1;1.5	18	$\sqrt{x}\operatorname{ch}(x)$	3;4

Лабораторная работа № 15.

Тема: «Приближенное вычисление определенных интегралов»

Обобщенная формулировка задания. Найти приближенное значение определенного интеграла $\int_a^b f(x)dx$ с заданной точностью ε , используя предложенные квадратурные формулы.

Для достижения заданной точности использовать двойной пересчет. Вычислить интеграл вначале для n разбиений отрезка, затем – для $2n$; сравнить полученные результаты: если $|I_n - I_{2n}| < \varepsilon$, то $I \approx I_{2n}$, в противном случае вычислить интеграл для $4n$ и т.д.

Функцию $f(x)$ и предлагаемую квадратурную формулу оформить в виде функций. Значения a, b, ε и начальное разбиение отрезка вести с клавиатуры.

Использовать одну из следующих квадратурных формул.

1. Составная формула трапеций

$$\int_a^b f(x)dx \approx \frac{b-a}{2n} [f(a) + 2f(a+h) + \dots + 2f(a+(n-1)h) + f(b)], \quad (15.1)$$

$$h = \frac{b-a}{n}.$$

2. Составная формула Симпсона

$$\int_a^b f(x)dx \approx \frac{b-a}{3n} [f(a) + 2(f(a+2h) + \dots + f(a+(n-2)h)) + 4(f(a+h) + \dots + f(a+(n-1)h)) + f(b)] \quad (15.2)$$

$$h = \frac{b-a}{n}.$$

3. Формула левых прямоугольников

$$\int_a^b f(x)dx \approx h \sum_{i=0}^{n-1} f(x_i), \quad h = \frac{b-a}{n}, \quad x_i = a + ih, i = \overline{0, n-1}. \quad (15.3)$$

4. Формула правых прямоугольников

$$\int_a^b f(x)dx \approx h \sum_{i=1}^n f(x_i), \quad h = \frac{b-a}{n}, \quad x_i = a + ih, i = \overline{1, n}. \quad (15.4)$$

5. Формула средних прямоугольников

$$\int_a^b f(x)dx \approx h \sum_{i=0}^{n-1} f(x_i + \frac{h}{2}), \quad h = \frac{b-a}{n}, \quad x_i = a + ih, i = \overline{0, n-1}. \quad (15.5)$$

Пример выполнения задания. Используя квадратурную формулу левых прямоугольников (15.3), вычислить приближенное значение интеграла $\int_a^b \sin x \cdot e^x dx$ с точностью ε .

Описание используемых функций. Функция LeftRectangle с прототипом:

```
double LeftRectangle (double LowerLimit, double
                      UpperLimit, unsigned k);
```

возвращает значение интеграла, посчитанное по формуле левых прямоугольников для k разбиений отрезка интегрирования.

Функция SinExp с прототипом:

```
double SinExp(double x);
```

возвращает значение подынтегральной функции $\sin x \cdot e^x$ при заданном значении x.

Текст программы.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define TRUE 1

double LeftRectangle( double , double , unsigned);
double SinExp(double );

int main(void)
{
    double BeginSegm, EndSegm, PreviosIntegral=0,
           NextIntegral=1, Epsilon;
    unsigned n;

    while(TRUE)
    {
        printf("\nEnter the value of regs of segment, "
              "accuracy and partitions number: ");
        scanf("%lf%lf%lf%u",&BeginSegm,&EndSegm,
              &Epsilon,&n);
        if ((BeginSegm < EndSegm) && (n > 0) &&
            (Epsilon < 1) && (Epsilon > 0)) break;
        printf("\nParameters are incorrect!!! Try "
              "again!!!\n");
    }
}
```

```

clrscr();
while ( fabs(PreviosIntegral - NextIntegral) >
        Epsilon )
{
    PreviosIntegral = LeftRectangle (BeginSegm,
                                     EndSegm, n);
    NextIntegral = LeftRectangle (BeginSegm,
                                 EndSegm, 2*n);
    n = 2 * n ;
}

printf("\nThe value of integral of function "
        "sin(x)*exp(x) on segment ");
printf("[%f,%f] is equal %10.6f", BeginSegm, EndSegm,
        NextIntegral);

printf("\nPress any key to exit...");
getch();
return 0;
}

double LeftRectangle (double LowerLimit, double
                      UpperLimit, unsigned k)
{
    double Step = (LowerLimit + UpperLimit) / k;
    double Integral = 0;
    double t = LowerLimit;

    while ( t < UpperLimit)
    {
        Integral = Integral + SinExp (t);
        t = t + Step;
    }

    Integral = Step * Integral;
    return Integral;
}

double SinExp(double x)
{
    return sin(x)*exp(x);
}

```

Варианты заданий.

Таблица 15.1.

№	$f(x)$	$[a, b]$	Квадратурные формулы	Точность
1	\sqrt{x}	0;1	(15.1), (15.2)	$\varepsilon = 1/2 \cdot 10^{-2}$
2	e^{x^2}	0;1	(15.2), (15.3)	$\varepsilon = 1/2 \cdot 10^{-2}$
3	$1/(1+x)$	0;1	(15.3), (15.4)	$\varepsilon = 1/2 \cdot 10^{-4}$
4	$1/(1+x^3)$	0;1	(15.4), (15.5)	$\varepsilon = 1/2 \cdot 10^{-4}$
5	e^{-x^2}	0;0	(15.1), (15.2)	$\varepsilon = 1/2 \cdot 10^{-4}$
6	$e^{1/x}$	2;4	(15.2), (15.3)	$\varepsilon = 1/2 \cdot 10^{-2}$
7	$\sin x/x$	0; $\pi/2$	(15.3), (15.4)	$\varepsilon = 1/2 \cdot 10^{-4}$
8	$\sin x/x$	0;2	(15.4), (15.5)	$\varepsilon = 1/2 \cdot 10^{-3}$
9	$\sin x^2$	0; $\pi/4$	(15.1), (15.2)	$\varepsilon = 1/2 \cdot 10^{-3}$
10	$\cos x^2$	0;1	(15.2), (15.3)	$\varepsilon = 1/2 \cdot 10^{-3}$
11	$1/1-x+x^2$	0;1	(15.3), (15.4)	$\varepsilon = 1/2 \cdot 10^{-3}$
12	$1/\sqrt{x^4}$	0;1	(15.4), (15.5)	$\varepsilon = 1/2 \cdot 10^{-3}$
13	$1/x$	1;2	(15.1), (15.2)	$\varepsilon = 1/2 \cdot 10^{-4}$
15	$\ln x$	4;5.2	(15.2), (15.3)	$\varepsilon = 1/2 \cdot 10^{-3}$
15	$\sqrt{x} \cos x$	0;1	(15.3), (15.4)	$\varepsilon = 1/2 \cdot 10^{-3}$
16	$\sqrt{x} \sin x$	0;1	(15.4), (15.5)	$\varepsilon = 1/2 \cdot 10^{-3}$
17	$\sin(x)/(1+x^2)$	0;1	(15.1), (15.2)	$\varepsilon = 1/2 \cdot 10^{-3}$
18	$\cos(x)/x^2$	$\pi/4; \pi/2$	(15.2), (15.3)	$\varepsilon = 1/2 \cdot 10^{-3}$

Лабораторная работа № 16.**Тема: «Решение нелинейных уравнений»**

Обобщенная формулировка задания. Решить нелинейное уравнение вида $f(x) = 0$ с параметром с заданной точностью ε , используя предлагаемые итерационные методы. Предварительно провести графическое отделение корней, т.е. найти все отрезки, на которых существует единственный корень. Требуемую точность и начальное приближение ввести с клавиатуры.

Использовать один из следующих итерационных алгоритмов нахождения решения на отрезке $[a, b]$:

1. Метод Ньютона

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0, 1, \dots, x_0 \in [a, b] \quad (16.1)$$

2. Модифицированный метод Ньютона

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}, n = 0, 1, \dots, x_0 \in [a, b] \quad (16.2)$$

3. Модифицированный метод Ньютона

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f''(x_n)f^2(x_n)}{2[f'(x_n)]^2}, n = 0, 1, \dots, x_0 \in [a, b] \quad (16.3)$$

4. Модифицированный метод Ньютона

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f(x_n - (f'(x_n))^{-1}f(x_n))}{f'(x_n)}, n = 0, 1, \dots, \quad (16.4)$$

$$x_0 \in [a, b]$$

5. Метод секущих

$$x_{n+1} = x_n - \frac{(x_n - x_{n-1})f(x_n)}{f(x_n) - f(x_{n-1})}, n = 0, 1, \dots, x_0, x_1 \in [a, b] \quad (16.5)$$

6. Модифицированный метод секущих

$$x_{n+1} = x_n - \frac{(b - x_n)f(x_n)}{f(b) - f(x_{n-1})}, n = 0, 1, \dots, x_0 = a \quad (16.6)$$

при $f(b)f''(x) > 0, \forall x \in [a, b]$.

7. Модифицированный метод секущих

$$x_{n+1} = x_n - \frac{(x_n - a)f(x_n)}{f(x_n) - f(a)}, n = 0, 1, \dots, x_0 = b \quad (16.7)$$

при $f(a)f''(x) > 0, \forall x \in [a, b]$.

Пример выполнения задания. Решить методом Ньютона с точностью ε нелинейное уравнение $3x - \cos x - 1 = 0$, предварительно отделив корни.

Описание используемых функций. Функция `NewtonMethod` с прототипом

`double NewtonMethod (double Previous, double Epsilon);`

возвращает решение нелинейного уравнения для начального приближения `Previous`, посчитанного по формуле Ньютона с точностью `Epsilon`.

Функция `Function` с прототипом

`double Function (double x);`

возвращает значение функции $f(x) = 3x - \cos x - 1$ при заданном значении x .

Функция `Derivative` с прототипом

double Derivative (double x);

возвращает значение производной функции $3x - \cos x - 1$ при заданном значении x .

Текст программы.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define TRUE 1

double NewtonMethod ( double , double);
double Function(double );
double Derivative(double );

int main(void)
{
    double Root, Epsilon;

    while(TRUE)
    {
        printf("\nEnter the initial approximation and "
               "accuracy: ");
        scanf("%lf%lf",&Root,&Epsilon);
        if ((Epsilon < 1) && (Epsilon > 0)) break;
        printf("\nAccuracy is incorrect!!! Try "
               "again!!!\n");
    }

    clrscr();
    printf("\nThe root of the equation is equal %lf",
           NewtonMethod(Root,Epsilon));

    printf("\nPress any key to exit...");
    getch();
    return 0;
}

double NewtonMethod (double Previous, double Epsilon)
{
    double Following= Previous -
        Function(Previous)/Derivative(Previous);

    while (fabs(Previous - Following)>= Epsilon)
```

```

{
    Previous = Following;
    Following = Previous -
        Function(Previous)/Derivative(Previous);
}

return Following;
}

double Function (double x)
{
    return 3*x-cos(x)-1;
}

double Derivative (double x)
{
    return 3+sin(x);
}

```

Варианты заданий.

Таблица 16.1.

№	$f(x)$	Интервал для α	Шаг изменения α	Итерационный метод
1	2	3	4	5
1	$x^2 \cos 2x + 1 + \alpha$	1:2	0.1	(16.1), (16.7)
2	$2x - \alpha \cos x$	0:1	0.1	(16.2), (16.6)
3	$(x - 1 + \alpha)^3 + 0.5e^x$	0:0.5	0.01	(16.3), (16.5)
4	$\alpha x - \cos x - 1$	3:4	0.1	(16.4), (16.2)
5	$x^4 + \alpha x - 1$	1:5	1	(16.1), (16.7)
6	$x^4 + \cos x - \alpha$	3:7	0.5	(16.2), (16.6)
7	$3x - e^{\alpha x}$	1:5	1	(16.3), (16.5)
8	$e^x - 1 - \alpha x^3$	1:5	1	(16.4), (16.2)
9	$\alpha x - \ln x - 5$	0.5:4	0.5	(16.1), (16.7)
10	$e^{x+\alpha} - x + 1$	1:5	1	(16.2), (16.6)
11	$e^{1/x} - (x+1)^2 + \alpha$	0:5	1	(16.3), (16.5)
12	$\alpha x^3 - \sin x$	0:1	0.1	(16.4), (16.2)
13	$e^x - 6x - \alpha$	3:10	1	(16.1), (16.7)
14	$x^2 + \cos x - \alpha$	3:7	0.5	(16.2), (16.6)
15	$x - \sin x - \alpha$	2:3	0.1	(16.3), (16.5)

1	2	3	4	5
16	$\alpha x + e^x$	1:5	1	(16.4), (16.2)

Лабораторная работа № 17.

**Тема: «Указатели и массивы. Индексация с помощью указателей.
Передача массивов в функции»**

Обобщенная формулировка задания. Исходный массив ввести с клавиатуры, заполнить алгоритмически или инициализировать случайным образом (в зависимости от постановки задачи). При выполнении задания предусмотреть возможность введения с клавиатуры размерности обрабатываемого массива в диапазоне объявленной максимальной размерности. Исходный массив и полученный результат вывести на экран. Основные этапы работы с массивом (ввод с клавиатуры или инициализация, вывод на экран, работа с массивом) оформить в виде функций. Индексацию элементов массива осуществлять с помощью указателей.

Пример выполнения задания. Дан одномерный целочисленный массив. Отсортировать массив по неубыванию методом «пузырька».

Описание используемых функций. Функция `InputArray` с прототипом

```
void InputArray ( int* A, unsigned n );
```

осуществляет ввод одномерного массива A размерности n с клавиатуры.

Функция `DisplayArray` с прототипом

```
void DisplayArray ( int* A, unsigned n );
```

осуществляет вывод одномерного массива A размерности n на экран.

Функция `BubbleSort` с прототипом

```
void BubbleSort ( int* A, unsigned n);
```

выполняет сортировку методом «пузырька» одномерного массива A размерности n.

Текст программы.

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
```

```
#define TRUE 1
```

```
void InputArray ( int* , unsigned);
void DisplayArray ( int*, unsigned);
```

```

void BubbleSort( int* , unsigned);

int main(void)
{
    unsigned const DIM = 100;
    int Massive[DIM];
    unsigned n;

    while(TRUE)
    {
        printf("Enter n <= %d - dimention of massive:",
              DIM);
        scanf("%u", &n);
        if ((n > 0) && (n <= DIM)) break;
        printf("\n Dimention is incorrect!!! Try "
              "again!!!\n");
    }

    printf("\nEnter the elements of massive:\n");
    InputArray ( Massive, n );
    clrscr();
    printf("\tSource array\n");
    DisplayArray ( Massive, n ) ;
    BubbleSort( Massive, n );
    printf("\tSelected array \n");
    DisplayArray ( Massive, n ) ;
    printf("\nPress any key to exit...");
    getch();
    return 0;
}

void InputArray ( int* A, unsigned n )
{
    unsigned i;
    for ( i = 0; i < n; i++ )
    {
        printf("\nA[%u] = ",i);

        scanf("%d",&*(A + i));
    }
}

void DisplayArray ( int* A, unsigned n )
{
    unsigned i;

```

```

    for ( i = 0; i < n; i++ )
    {
        printf("%d ",*(A + i));
    }
}

void BubbleSort ( int* A, unsigned n)
{
    unsigned i, j;
    int x;

    for( i = 0; i < n ; i ++ )
    {
        for( j = n - 1; j > i; j -- )
        {
            if (*(A + j) < *(A + j - 1) )
            {
                x = *(A + j);
                *(A + j) = *(A + j - 1);
                *(A + j - 1) = x;
            }
        }
    }
}

```

Варианты заданий. См. «Лабораторная работа № 4».

Лабораторная работа № 18.

Тема: «Массивы динамической памяти»

Обобщенная формулировка задания. Ввести размеры одномерного массива. Сформировать динамический массив заданного размера. Заполнить массив значениями, введя с клавиатуры либо алгоритмически. Выполнить обработку массива согласно конкретно поставленной задаче. Вывести на экран исходный и полученный результаты. Удалить динамический массив.

Пример выполнения задания. Дан одномерный целочисленный массив. Отсортировать массив по неубыванию методом «пузырька».

Описание используемых функций. Функция InitArray с прототипом

```
int* InitArray ( unsigned n );
```

возвращает указатель на участок динамической памяти, где расположен одномерный массив размерности n, заполненный с клавиатуры.

Функция DisplayArray с прототипом

```
void DisplayArray ( int* Pointer, unsigned n );
```

осуществляет вывод одномерного динамического массива Pointer размерности n на экран.

Функция BubbleSort с прототипом

```
void BubbleSort ( int* Pointer, unsigned n);
```

выполняет сортировку методом «пузырька» одномерного динамического массива Pointer размерности n.

Текст программы.

```
#include <conio.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define TRUE 1

int* InitArray ( unsigned);
void DisplayArray ( int*, unsigned);
void BubbleSort( int* , unsigned);

int main(void)
{
    unsigned Dimension;
    int* Massive;

    while(TRUE)
    {
        printf("Enter dimention of massive: ");
        scanf("%u", &Dimension);
        if (Dimension > 0) break;
        printf("\n Dimention is incorrect!!! Try "
               "again!!!\n");
    }

    clrscr();
    Massive = InitArray (Dimension);
    clrscr();
    if ( Massive == NULL )
    {
        printf("\nDynamic array don't exist!\n");
        printf("\nPress any key to exit... ");
        getch();
        return 0;
    }
}
```

```

    }

    printf("\tSource array\n");
    DisplayArray (Massive, Dimension);

    BubbleSort (Massive, Dimension);
    printf("\n\tSelected array\n");

    DisplayArray (Massive, Dimension);
    free(Massive);

    printf("\nPress any key to exit... ");
    getch();
    return 0;
}

int* InitArray ( unsigned n )
{
    int* Pointer = (int*)malloc(n*sizeof(int));
    unsigned i;

    if ( Pointer == NULL ) return NULL;
    printf("\n Enter the elemets of array: \n");
    for ( i = 0; i < n; i++ )
    {
        printf("\nA[%u] = ",i);
        scanf("%d",&*(Pointer + i));
    }

    return Pointer;
}

void DisplayArray ( int* Pointer, unsigned n )
{
    unsigned i;

    for ( i = 0; i < n; i++ )
    {
        printf(" %d",*( Pointer + i));
    }
}

void BubbleSort ( int* Pointer, unsigned n)
{
    unsigned i, j;

```

```

int x;

for( i = 0; i < n ; i ++ )
{
    for( j = n - 1 ; j > i; j -- )
    {
        if (Pointer [j] < Pointer [j-1] )
        {
            x = Pointer[j];
            Pointer[j] = Pointer[j-1];
            Pointer[j-1] = x;
        }
    }
}

```

Варианты заданий. См. «Лабораторная работа № 4».

Лабораторная работа № 19.

Тема: «Массивы указателей и моделирование многомерных массивов»

Обобщенная формулировка задания 1. Определить максимально возможный диапазон изменения размерностей матрицы. Ввести в указанном диапазоне реальную размерность матрицы. Заполнить матрицу значениями, введя с клавиатуры либо алгоритмически. Сформировать динамический массив из указателей на строки заданной матрицы и использовать его в качестве параметра в функциях. Выполнить обработку матрицы согласно конкретно поставленной задаче. Вывести на экран исходный и полученный результаты. Удалить динамический массив.

Обобщенная формулировка задания 2. Ввести размеры матрицы. Осуществляя подмену матрицы одномерным массивом и имитируя внутри используемых функций доступ к ней, сформировать динамический одномерный массив. Заполнить массив значениями, введя с клавиатуры либо алгоритмически. Выполнить обработку массива согласно конкретно поставленной задаче. Вывести на экран исходный и полученный результаты. Удалить динамический массив.

Обобщенная формулировка задания 3. Ввести размеры двумерного массива. Сформировать динамический двумерный массив заданного размера (массив из указателей на строки матрицы). Заполнить массив значениями, введя с клавиатуры либо алгоритмически. Выполнить обработку массива согласно конкретно поставленной задаче. Вывести на экран исходный и полученный результаты. Удалить динамический массив.

Пример выполнения задания 1. Дана целочисленная квадратная

матрица порядка n вида
$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 2 & 2 & \dots & 2 \\ \dots & \dots & \dots & \dots \\ n & n & \dots & n \end{bmatrix}$$
. Транспонировать матрицу на том же месте.

Описание используемых функций. Функция `InitMatrix` с прототипом

```
void InitMatrix ( int* Pointer[], unsigned n );
```

заполняет алгоритмически массив из указателей `Pointer` на строки квадратной матрицы порядка n .

Функция `DisplayMatrix` с прототипом

```
void DisplayMatrix ( int * Pointer[], unsigned n);
```

выводит на экран содержимое участка памяти по адресу `Pointer`, по которому находится квадратная матрица порядка n .

Функция `TranspouseMatrix` с прототипом

```
void TranspouseMatrix ( int * Pointer[], unsigned n);
```

транспонирует квадратную матрицу порядка n , возвращая результат в функцию `main` через массив указателей `Pointer`.

Текст программы.

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#define TRUE 1
#define DIM 10
```

```
void InitMatrix( int* [], unsigned );
void DisplayMatrix ( int* [],unsigned );
void TranspouseMatrix ( int* [],unsigned );
```

```
int main(void)
{
    int Matrix[DIM][ DIM];
    unsigned n, i;
    int** Pointer;

    while(TRUE)
```

```

{
    printf("Enter dimention of matrix: ");
    scanf("%u", &n);
    if ((n > 0) && (n <= DIM))break;
    printf("\n Dimention is incorrect! Try
        again!!!\n");
}

clrscr();
Pointer = (int**)malloc(n*sizeof(int*));
if (Pointer == NULL )
{
    printf("\nDynamic massive don't exist!\n");
    printf("\nPress any key to exit... ");
    getch();
    return 0;
}

for ( i = 0; i < n; i++ )
{
    Pointer [i] = & Matrix [i][0];
}

InitMatrix (Pointer, n);
printf("\tSource Matrix \n");
DisplayMatrix (Pointer, n);
TransposeMatrix (Pointer, n);

printf("\n\tTransposed Matrix\n");
DisplayMatrix ( Pointer, n);
free (Pointer);

printf("\nPress any key to exit...");
getch();
return 0;
}

void InitMatrix ( int* Pointer[], unsigned n )
{
    unsigned i, j;

    for ( i = 0; i < n; i++ )
    {
        for ( j = 0; j < n; j++ )
        {

```



```

        *( Pointer [i] + j ) = i + 1;
    }
}

void DisplayMatrix ( int * Pointer[], unsigned n )
{
    unsigned i, j;

    for ( i = 0; i < n; i++ )
    {
        for ( j = 0; j < n; j++ )
        {
            printf("%4d",*( Pointer [i] + j ));
        }
        printf("\n");
    }
}

void TransposeMatrix ( int * Pointer[], unsigned n )
{
    unsigned i, j;
    int x ;

    for ( i = 0; i < n; i++)
    {
        for ( j = i + 1; j < n; j++)
        {
            x = *( Pointer [i] + j );
            *( Pointer [i] + j ) = *( Pointer [j] + i );
            *( Pointer [j] + i ) = x;
        }
    }
}

```

Пример выполнения задания 2.

Описание используемых функций. Функция `InitMatrix` с прототипом

```
int* InitMatrix (unsigned n);
```

возвращает указатель на участок динамической памяти, по которому располагается алгоритмически заполненная квадратная матрица порядка `n`.

Функция `DisplayMatrix` с прототипом

```
void DisplayMatrix ( int* Pointer , unsigned n);
```

выводит на экран содержимое участка памяти по адресу `Pointer`, по которому находится квадратная матрица порядка `n`.

Функция `TransposeMatrix` с прототипом

```
void TransposeMatrix ( int* Pointer, unsigned n);
```

транспонирует квадратную матрицу порядка `n`, находящуюся по адресу `Pointer`.

Текст программы.

```
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>

#define TRUE 1

int* InitMatrix( unsigned );
void DisplayMatrix ( int* ,unsigned );
void TransposeMatrix ( int* ,unsigned );

int main(void)
{
    unsigned Dimension;
    int* Matrix;

    while(TRUE)
    {
        printf("Enter dimention of matrix: ");
        scanf("%u", &Dimension);
        if (Dimension > 0) break;
        printf("\n Dimention is incorrect!!! Try "
               "again!!!\n");
    }

    clrscr();
    Matrix = InitMatrix (Dimension);
    if (Matrix == NULL)
    {
        printf("\nDynamic matrix don't exist!!!\n");
        printf("\nPress any key to exit... ");
        getch();
        return 0;
    }
}
```

```

printf("\n\tSource matrix \n");
DisplayMatrix(Matrix, Dimension);
TransposeMatrix(Matrix, Dimension);

printf("\n\tTransposed matrix\n");
DisplayMatrix(Matrix, Dimension);
free (Matrix);

printf("\nPress any key to exit...");
getch();
return 0;
}

int* InitMatrix (unsigned n )
{
    int* Pointer = (int*)malloc(n * n*sizeof(int));
    unsigned i, j;

    if (Pointer == NULL) return NULL;
    for ( i = 0; i < n; i++ )
    {
        for ( j = 0; j < n; j++ )
        {
            *( Pointer + n * i + j ) = i + 1;
        }
    }
    return Pointer;
}

void DisplayMatrix ( int* Pointer , unsigned n )
{
    unsigned i, j;

    for ( i = 0; i < n; i++ )
    {
        for ( j = 0; j < n; j++ )
        {
            printf("%4d",*( Pointer + n * i + j ));
        }
        printf("\n");
    }
}

```

```

void TransposeMatrix ( int* Pointer, unsigned n)
{
    unsigned i, j;
    int x ;

    for ( i = 0; i < n; i++)
    {
        for ( j = i + 1; j < n; j++)
        {
            x = *( Pointer + n * i + j );
            *( Pointer + n * i + j ) =
                *( Pointer + n * j + i );
            *( Pointer + n * j + i ) = x;
        }
    }
}

```

Пример выполнения задания 3.

Описание используемых функций. Функция InitMatrix с прототипом

```
int** InitMatrix (unsigned n);
```

возвращает указатель на участок динамической памяти, по которому располагается алгоритмически заполненная квадратная матрица порядка n.

Функция DisplayMatrix с прототипом

```
void DisplayMatrix ( int** Pointer, unsigned n);
```

выводит на экран содержимое участка памяти по адресу Pointer, по которому находится квадратная матрица порядка n.

Функция TransposeMatrix с прототипом

```
void TransposeMatrix ( int** Pointer, unsigned n);
```

транспонирует квадратную матрицу порядка n, находящуюся по адресу Pointer.

Текст программы.

```

#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>

```

```
#define TRUE 1
```

```
int** InitMatrix (unsigned);
```

```

void DisplayMatrix ( int** , unsigned);
void TransposeMatrix ( int** , unsigned);

int main(void)
{
    unsigned Dimension;
    int** Matrix;
    unsigned i;

    while(TRUE)
    {
        printf("Enter dimation of matrix: ");
        scanf("%u", &Dimension);
        if (Dimension > 0) break;
        printf("\n Dimation is incorrect!!! Try "
               "again!!!\n");
    }

    clrscr();
    Matrix = InitMatrix (Dimension);
    if (Matrix == NULL)
    {
        printf("\nDynamic matrix don't exist!\n");
        printf("\nPress any key to exit...");
        getch();
        return 0;
    }

    printf("\n\tSource matrix \n");
    DisplayMatrix(Matrix, Dimension);
    TransposeMatrix(Matrix, Dimension);

    printf("\n\tTransposed matrix\n");
    DisplayMatrix(Matrix, Dimension);
    for ( i = 0; i < Dimension; i++ )
    {
        free (Matrix[i]);
    }
    free (Matrix);

    printf("\nPress any key to exit...");
    getch();
    return 0;
}

```

```

int** InitMatrix (unsigned n )
{
    unsigned i, j;
    int** Pointer = (int**)malloc(n*sizeof(int*));

    if (Pointer == NULL) return NULL;
    for ( i = 0; i < n; i++ )
    {
        Pointer [i] = (int*)malloc(n*sizeof(int));
        if (Pointer [i] == NULL) return NULL;
    }

    for ( i = 0; i < n; i++ )
    {
        for ( j = 0; j < n; j++ )
        {
            Pointer [i][j] = i + 1;
        }
    }
    return Pointer;
}

```

```

void DisplayMatrix ( int** Pointer, unsigned n )
{
    unsigned i, j;

    for ( i = 0; i < n; i++ )
    {
        for ( j = 0; j < n; j++ )
        {
            printf("%4d",Pointer[i][j]);
        }
        printf("\n");
    }
}

```

```

void TransposeMatrix ( int** Pointer, unsigned n)
{
    unsigned i, j;
    int x ;

    for ( i = 0; i < n; i++)
    {
        for ( j = i + 1; j < n; j++)
        {

```

```

        x = Pointer [i][j];
        Pointer [i][j] = Pointer [j][i];
        Pointer [j][i] = x;
    }
}
}

```

Варианты заданий. См. «Лабораторная работа № 5».

Лабораторная работа №.20

Тема: «Указатели на функции»

Обобщенная формулировка задания. Выполнить задания лабораторных работ № 15, № 16, используя в функциях в качестве параметров указатели на функции:

- при вычислении интегралов – указатели на подынтегральную функцию и функцию квадратурной формулы передаются как параметры в функцию двойного пересчета.
- при решении нелинейных уравнений – указатель на функцию решаемого уравнения передается как параметр в функцию интерполяционного метода.

Пример выполнения задания. Используя квадратурную формулу левых прямоугольников (15.3), вычислить приближенное значение интеграла $\int_a^b \sin x \cdot e^x dx$ с точностью ε .

Описание используемых функций. Функция LeftRectangle с прототипом

```
double LeftRectangle(double LowerLimit, double UpperLimit,
                    unsigned Number, IntegralFunction F);
```

возвращает значение интеграла функции F на отрезке от LowerLimit до UpperLimit, посчитанное по формуле левых прямоугольников для Number разбиений отрезка интегрирования.

Функция DoubleConverting с прототипом

```
double DoubleConverting(double BeginSegm, double EndSegm,
                        double Epsilon, int Number,
                        IntegralFunction F, QuadratureFormula Q);
```

возвращает значение интеграла функции F на отрезке от LowerLimit до UpperLimit, посчитанное по квадратурной формуле Q, начиная с Number разбиений отрезка интегрирования, с точностью Epsilon по формуле двойного пересчета.

Функция SinExp с прототипом

```
double SinExp(double x);
```

возвращает значение подынтегральной функции $\sin x \cdot e^x$ при заданном значении x .

Текст программы.

```
#include <conio.h>
#include <stdio.h>
#include <math.h>

#define TRUE 1

typedef double(*IntegralFunction)(double);
typedef double(*QuadratureFormula)( double , double ,
                                     int , IntegralFunction );

double SinExp(double );
double LeftRectangle ( double, double , int ,
                      IntegralFunction );

double DoubleConverting(double , double, double , int ,
                       IntegralFunction,QuadratureFormula);

int main(void)
{
    double BeginSegm, EndSegm, Epsilon, Integral;
    unsigned Number;

    while(TRUE)
    {
        printf("\nEnter the value of regs of segment, "
               "accuracy and partitions number: ");
        scanf("%lf%lf%lf%u",&BeginSegm,&EndSegm,
              &Epsilon,&Number);
        if ((BeginSegm < EndSegm) && (Number > 0) &&
            (Epsilon < 1) && (Epsilon > 0)) break;
        printf("\nParameters are incorrect!!! Try "
               "again!!!\n");
    }

    Integral = DoubleConverting(BeginSegm, EndSegm,
                               Epsilon,Number, SinExp,LeftRectangle);

    clrscr();
    printf("\nThe value of integral of function"
```



```

        "sin(x)*exp(x) on the segment "
        " [%4.2f,%4.2f] is equal %10.6f", BeginSegm,
        EndSegm, Integral);

    printf("\nPress any key to exit...");
    getch();
    return 0;
}

double DoubleConverting(double BeginSegm, double EndSegm,
                        double Epsilon, int Number,
                        IntegralFunction F, QuadratureFormula Q)
{
    double PreviousIntegral, NextIntegral;

    do
    {
        PreviousIntegral = Q (BeginSegm, EndSegm,
                               Number, F);
        NextIntegral = Q (BeginSegm, EndSegm,
                           2* Number, F);
        Number = 2 * Number ;
    }
    while ( fabs(PreviousIntegral - NextIntegral) >
            Epsilon);
    return NextIntegral;
}

double LeftRectangle(double LowerLimit, double UpperLimit,
                    int Number, IntegralFunction F)
{
    double Result = 0;
    double Step = (LowerLimit + UpperLimit) / Number;
    double t = LowerLimit;

    while ( t < UpperLimit)
    {
        Result = Result + F(t);
        t = t + Step;
    }
    return ( Step * Result);
}

double SinExp(double x)
{

```

```

    return sin(x)*exp(x);
}

```

Варианты заданий. См. «Лабораторная работа № 15», «Лабораторная работа № 16».

Лабораторная работа №.21

Тема: «Структуры, массивы, указатели. Динамические массивы структур»

Обобщенная формулировка задания. Выбрать предметную область для базы данных и предложить структуру для описания отдельных записей базы данных. Выбранная структура должна иметь не менее пяти полей (элементов) двух и более типов. Для выбранной базы данных написать следующие функции:

1. Функцию формирования динамического одномерного массива структур, значения которых вводятся с клавиатуры. Предусмотреть возможность заполнения одного поля структуры, используя известные значения других полей структуры. При вводе структур можно реализовать один из следующих механизмов:
 - ввод заранее заданного количества структур;
 - ввод до появления структуры с заданным признаком;
 - диалог с пользователем о необходимости продолжать ввод.
2. Функцию просмотра содержимого динамического массива структур.
3. Функцию дополнения уже существующего массива структур новыми структурами.
4. Функцию поиска и вывода на экран структуры (структур) с заданным значением элемента.
5. Функцию упорядочения массива структур по заданному полю (элементу).

Пример выполнения задания. На основе предметной области «Человек» (фамилия, имя, отчество, пол, возраст) создать динамический массив структур. Вывести на экран содержимое массива. Сделать выборку в исходном массиве согласно полу и возрастному диапазону. Отсортировать исходный массив в алфавитном порядке.

Описание используемых функций. Функция `InitArray` с прототипом

```
Person* InitArray(int Dimension);
```

возвращает указатель на участок динамической памяти, где расположен одномерный массив структур размерности `Dimension`.

Функция `InitPerson` с прототипом

```
Person InitPerson();
```

возвращает структуру типа `Person`, заполненную с клавиатуры.

Функция `DisplayArray` с прототипом

```
void DisplayArray (Person* Massive, int Dimension);
```

осуществляет вывод на экран одномерного массива Massive структур типа Person размерности Dimension.

Функция DisplayChoise с прототипом

```
void DisplayChoise(Person* Massive, int Dimension,  
                  char* SexTag, int LowAge, int UpperAge);
```

осуществляет выборочный (по полу SexTag в возрастном диапазоне от LowAge до UpperAge) вывод на экран одномерного массива Massive структур типа Person размерности Dimension.

Функция DisplayPerson с прототипом

```
void DisplayPerson (Person Man);
```

осуществляет вывод на экран одной структуры Man типа Person.

Функция SortFirstName с прототипом

```
void SortFirstName(Person* Massive, int Dimension);
```

выполняет сортировку динамического массива Massive размерности Dimension по полю FirstName.

Текст программы.

```
#include <stdio.h>  
#include <conio.h>  
#include <string.h>  
#include <stdlib.h>  
  
typedef struct  
{  
    char FirstName[15], SecondName[15], LastName[15];  
    int Age;  
    char Sex [7];  
} Person;  
  
Person InitPerson ();  
Person* InitArray ( int);  
void DisplayArray (Person* , int );  
void DisplayChoise (Person* , int , char* , int , int );  
void DisplayPerson (Person );  
void SortFirstName(Person* , int );  
  
int main(void)  
{  
    int Dimension;
```

```

char SexTag [7];
int LowAge, UpperAge;
Person* MassiveStruct;

printf("\nEnter the number of persons:");
scanf("%d",&Dimension);

MassiveStruct = InitArray (Dimension);
if ( MassiveStruct == NULL )
{
    printf("\nDynamic array don't exist!\n");
    printf("\nPress any key to exit...");
    getch();
    return 0;
}

printf("\nThe list of persons: \n");
DisplayArray (MassiveStruct,Dimension);

printf("\nEnter the sex-tag: ");
scanf("%s",SexTag);

printf("\nEnter the boundary of age: ");
scanf("%d%d",&LowAge,&UpperAge);

printf("\n\nThe list of choise-persons: \n");
DisplayChoise(MassiveStruct,Dimension,SexTag,
              LowAge,UpperAge);

printf("\n\nThe sorting list of persons: \n");
SortFirstName(MassiveStruct,Dimension);
DisplayArray (MassiveStruct,Dimension);

free(MassiveStruct);
printf("\nPress any key to exit...\n");
getch();
return 0;
}

Person InitPerson()
{
    Person Man;

    printf("\nEnter first name:");
    scanf("%s", Man.FirstName);

```

```

    printf("Enter second name:");
    scanf("%s", Man.SecondName);
    printf("Enter last name:");
    scanf("%s", Man.LastName);
    printf("Enter age:");
    scanf("%d",&Man.Age);
    printf("Enter sex:");
    scanf("%s", Man.Sex);

    return Man;
}

Person* InitArray (int Dimension)
{
    int i;
    Person* Massive =
        (Person*)malloc(Dimension*sizeof(Person));

    if ( Massive == NULL ) return NULL;
    for( i = 0; i < Dimension; i++)
    {
        printf("\nEnter the information about %d"
               "person \n",i + 1);
        Massive[i] = InitPerson();
    }

    return Massive;
}

void DisplayArray (Person* Massive, int Dimension)
{
    int i;

    for( i = 0; i < Dimension; i++ )
    {
        DisplayPerson(Massive[i]);
    }
}

void DisplayChoise(Person* Massive, int Dimension,
                   char* SexTag, int LowAge, int UpperAge)
{
    int i;

    for( i = 0; i < Dimension; i++ )

```

```

    {
        if ( strcmp(Massive[i].Sex,SexTag)==0 &&
            Massive[i].Age <= UpperAge &&
            Massive[i].Age >= LowAge)
            DisplayPerson(Massive[i]);
    }
}

void DisplayPerson (Person Man)
{
    printf("\n%s  %s  %s, %d year, %s ",Man.FirstName,
        Man.SecondName, Man.LastName, Man.Age, Man.Sex);
}

void SortFirstName(Person* Massive, int Dimension)
{
    int i, j;
    Person Help;

    for( i = 0; i <= Dimension; i ++)
        for( j = Dimension-1; j > i; j --)
            if (strcmp(Massive[j].FirstName,
                Massive[j-1].FirstName)<0 )
            {
                Help = Massive[j];
                Massive[j] = Massive[j-1];
                Massive[j-1] = Help;
            }
}

```

Варианты заданий.

1. «Человек»:

фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира).

2. «Школьник»:

фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс.

3. «Студент»:

фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); ВУЗ; курс; группа; средний бал; специальность.

4. «Покупатель»:

фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета.

5. «Пациент»:

фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови.

6. «Владелец автомобиля»:

фамилия; имя; отчество; номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира) марка автомобиля; номер автомобиля; номер техпаспорта.

7. «Военнослужащий»:

фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); должность; звание.

8. «Рабочий»:

фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); № цеха; табельный номер; образование; год поступления на работу.

9. «Владелец стационарного телефона»:

фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона.

10. «Абитуриент»:

фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); оценки по экзаменам; проходной балл.

11. «Государство»:

название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства.

12. «Автомобиль»:

марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.

13. «Товар»:

наименование; стоимость; срок хранения; сорт; дата выпуска; срок годности.

14. «Кинофильм»:

название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль.

15. «Рейс»:

марка автомобиля; номер автомобиля; пункт назначения; грузоподъемность (в тоннах); стоимость единицы груза; общая стоимость груза.

16. «Книга»:

название; автор (фамилия; имя); год выхода; издательство; себестоимость; цена; прибыль.

17. «Здание»:

адрес; тип здания; количество этажей; количество квартир; срок эксплуатации; срок до капитального ремонта (25 лет - срок эксплуатации).

Лабораторная работа №.22

Тема: «Динамические структуры данных. Односвязный список»

Обобщенная формулировка задания. В соответствии с индивидуальным заданием необходимо создать программу для обслуживания односвязного списка. Указанные в задании действия должны быть оформлены в виде отдельных функций. Память под очередной элемент динамической структуры данных следует выделять динамически.

Выполнение действий по обслуживанию динамической структуры данных должно производиться в режиме диалога с пользователем.

Для каждого из вариантов задания необходимо разработать следующие функции:

1. Создание списка.
2. Добавление элемента в список:
 - в начало списка;
 - в конец списка;
 - после элемента с заданным номером;
 - после элемента с заданным ключом.
3. Вывод содержимого списка на экран.
4. Удаление элемента из списка:
 - из начала списка;
 - из конца списка;
 - элемента с заданным номером;
 - элемента с заданным ключом.
5. Упорядочивание элементов в списке по выбранному признаку.

Пример выполнения задания. Создать односвязный список из целых чисел. Выполнить основные операции по изменению списка.

Описание используемых функций. Функция `CreateList` с прототипом

```
List* CreateList();
```

возвращает указатель на адрес начала списка.

Функция `DisplayList` с прототипом

```
void DisplayList(List* Begin);
```

выводит на экран содержимое односвязного списка с адресом начала `Begin`.

Функция `RemoveTermBegin` с прототипом

```
void RemoveTermBegin (List** Begin);
```


удаляет из начала односвязного списка с адресом начала *Begin один элемент.

Функция RemoveTermTag с прототипом

```
void RemoveTermTag (List* Begin);
```

удаляет из односвязного списка с адресом начала Begin все элементы с заданным значением.

Функция RemoveTermEnd с прототипом

```
void RemoveTermEnd (List** Begin);
```

удаляет из конца односвязного списка с адресом начала *Begin один элемент.

Функция AddTermBegin с прототипом

```
void AddTermBegin (List** Begin);
```

добавляет в начало односвязного списка с адресом начала *Begin один элемент.

Функция AddTermEnd с прототипом

```
void AddTermEnd (List* Begin);
```

добавляет в конец односвязного списка с адресом начала Begin один элемент.

Функция AddTermTag с прототипом

```
void AddTermTag (List* Begin);
```

добавляет в односвязный список с адресом начала Begin один элемент после элемента с заданным значением.

Функция FreeList с прототипом

```
FreeList(struct List** Begin);
```

освобождает участок памяти, занимаемый односвязным списком с адресом начала *Begin.

Текст программы.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

#define TRUE 1

struct List
{
    int x;
    struct List * Next;
};

struct List * CreateList(void);
void DisplayList( struct List* );
```

```

void RemoveTermBegin( struct List** );
void RemoveTermTag( struct List* );
void RemoveTermEnd( struct List* );
void AddTermBegin( struct List** );
void AddTermEnd(struct List* );
void AddTermTag(struct List* );
void FreeList(struct List** );

int main(void)
{
    struct List* Begin;
    char Key;

    while(TRUE)
    {
        printf( "\nEnter the number- the mode of "
                "operations with Lists: "
                "\n      1 - FORMATION OF THE LIST"
                "\n      2 - VIEWING OF CONTENTS OF "
                "THE LIST"
                "\n      3 - REMOVE THE ELEMENT FROM "
                "THE BEGIN OF LIST"
                "\n      4 - REMOVE THE ELEMENT FROM "
                "THE MIDDLE OF LIST"
                "\n      5 - REMOVE THE ELEMENT FROM "
                "THE END OF LIST"
                "\n      6 - ADD THE ELEMENT IN THE "
                "BEGIN OF LIST"
                "\n      7 - ADD THE ELEMENT IN THE "
                "END OF LIST"
                "\n      8 - ADD THE ELEMENT AFTER "
                "ELEMENT-KEY OF LIST"
                "\n      9 - END OF WORK\n");

        fflush( stdin );
        scanf("%c",&Key);
        clrscr();

        switch (Key)
        {
            case '1':
                clrscr();
                Begin = CreateList();
                printf("\n\nPress any key to return in "
                    "the menu...\n");
                getch();

```

```

        clrscr();
        break;

case '2':
    clrscr();
    DisplayList(Begin);
    printf("\n\nPress any key to return in "
           "the menu...\n");
    getch();
    clrscr();
    break;

case '3':
    clrscr();
    RemoveTermBegin(&Begin);
    printf("\n\nPress any key to return in "
           "the menu...\n");
    getch();
    clrscr();
    break;

case '4':
    clrscr();
    RemoveTermTag(Begin);
    printf("\n\nPress any key to return in "
           "the menu...\n");
    getch();
    clrscr();
    break;

case '5':
    clrscr();
    RemoveTermEnd(Begin);
    printf("\n\nPress any key to return in "
           "the menu...\n");
    getch();
    clrscr();
    break;

case '6':
    clrscr();
    AddTermBegin(&Begin);
    printf("\n\nPress any key to return in "
           "the menu...\n");
    getch();

```

```

        clrscr();
        break;

    case '7':
        clrscr();
        AddTermEnd(Begin);
        printf("\n\nPress any key to return in "
               "the menu...\n");
        getch();
        clrscr();
        break;

    case '8':
        clrscr();
        AddTermTag(Begin);
        printf("\n\nPress any key to return in "
               "the menu...\n");
        getch();
        clrscr();
        break;

    case '9':
        FreeList(&Begin);
        DisplayList(Begin);
        printf("\n\nPress any key to "
               "exit...\n");
        getch();
        return 0;

    default:
        clrscr();
        printf("\nIncorrect input!!! Try "
               "again!!!\n");
        printf("\n\nPress any key to return in "
               "the menu...\n");
        getch();
        clrscr();
        break;
    }
}

}

struct List* CreateList(void)
{
    struct List* Begin = NULL, *Previos = NULL,

```

```

        *Current= NULL;
char Ok = 'y';

while (Ok == 'y')
{
    printf("\nEnter the numbers of List: ");
    Current =
        ( struct List*)malloc(sizeof(struct List));

    if (Begin==NULL)
        Begin = Current;
    else
        Previos -> Next = Current;

    Previos = Current;
    scanf("%d",&Current -> x);

    fflush( stdin );
    printf("\nContinue? (Y/N):");
    scanf("%c",&Ok);
}
Previos -> Next = NULL;
return Begin;
}

void DisplayList(struct List* Begin)
{
    struct List* Current = Begin;

    if ( Begin == NULL )
    {
        printf("List is empty\n");
        return;
    }

    printf("\nThe List of numbers:\n");
    while (Current)
    {
        printf("%d    ",Current -> x);
        Current = Current -> Next;
    }

    printf("\n");
}

```

```

void RemoveTermBegin(struct List** Begin)
{
    struct List** Current = Begin;
    *Begin = (*Current) -> Next;
    free(Current);
}

void RemoveTermTag(struct List* Begin)
{
    struct List* Current = Begin, *Previos = Current;
    int NumberTag;

    printf("\nEnter number-key: ");
    scanf("%d",&NumberTag);

    while ( Current )
    {
        if ( Current -> x == NumberTag )
        {
            Previos -> Next = Current -> Next;
            free(Current);
            Current = Previos -> Next;
        }
        else
        {
            Previos = Current;
            Current = Current -> Next;
        }
    }
}

void RemoveTermEnd(struct List* Begin)
{
    struct List* Current = Begin, *Previos;

    while (Current -> Next)
    {
        Previos = Current;
        Current = Current -> Next;
    }

    Previos -> Next = NULL;
    free(Current);
}

```

```

void AddTermBegin(struct List** Begin)
{
    struct List* Current =
        (struct List*)malloc(sizeof(struct List));

    printf("\nEnter number: ");
    scanf("%d",&Current -> x);

    Current -> Next = *Begin;
    *Begin = Current;
}

void AddTermEnd(struct List* Begin)
{
    struct List* Current = Begin, *Previos;

    while (Current)
    {
        Previos = Current;
        Current = Current -> Next;
    }

    Current = (struct List*)malloc(sizeof(struct List));
    Previos -> Next = Current;

    printf("\nEnter the number:\n");
    scanf("%d",&Current -> x);
    Current -> Next = NULL;
}

void AddTermTag(struct List* Begin)
{
    struct List* Current = Begin, *Previos = Current;
    int NumberTag;

    printf("\nEnter number-tag: ");
    scanf("%d",&NumberTag);

    while ( Current )
    {
        if ( Current -> x == NumberTag )
        {
            Previos = Current;
            Current =
                (struct List*)malloc(sizeof(struct List));

```

```

        Current -> Next = Previos -> Next;
        Previos -> Next = Current;
        printf("\nEnter the number:\n");
        scanf("%d",&Current -> x);
        break;
    }
    else
    {
        Previos = Current;
        Current = Current -> Next;
    }
}

void FreeList(struct List** Begin)
{
    struct List* Current = *Begin;

    while (Current)
    {
        *Begin = (*Begin) -> Next;
        free(Current);
        Current = *Begin;
    }
}

```

Варианты заданий. См. «Лабораторная работа № 21».

Лабораторная работа №.23

Тема: «Текстовые файлы из чисел и матриц»

Обобщенная формулировка задания. Предварительно подготовить (программно или с помощью текстового редактора) текстовый файл, содержимым которого является массив (матрица) из чисел. В соответствии с индивидуальным заданием написать программу обработки содержимого файла. Для каждого из вариантов задания необходимо выполнить следующие действия. Просмотреть содержимое исходного файла. Считать содержимое файла в одномерный (двумерный) динамический массив. Обработать динамический массив согласно варианту задания. Полученный результат записать в конец исходного файла.

Пример выполнения задания. Отсортировать одномерный целочисленный массив, находящийся в текстовом файле, методом «пузырька». Отсортированный массив дописать в конец исходного файла.

Описание используемых функций. Функция ContentsFile с прототипом

```
int ContentsFile(char* String, char* Mode);
```

возвращает из текстового файла String, открытого в режиме Mode, число элементов массива.

Функция InitMassive с прототипом

```
int* InitMassive(char* String, char* Mode,  
                unsigned Number);
```

возвращает указатель на участок динамической памяти, где расположен одномерный массива размерности Number, считанный из текстового файла String, открытого в режиме Mode.

Функция BubbleSort с прототипом

```
void BubbleSort ( int* Pointer, unsigned Number);
```

выполняет сортировку методом «пузырька» одномерного массива Pointer размерности Number.

Функция AddFile с прототипом

```
void AddFile(char* String, char* Mode, int* Pointer,  
            unsigned Number);
```

дописывает в конец текстового файла String, открытого в режиме Mode, отсортированный массив Pointer размерности Number.

Текст программы.

```
#include <conio.h>  
#include <stdio.h>  
#include <process.h>  
#include <stdlib.h>  
  
int ContentsFile(char* , char*);  
int* InitMassive(char* , char* ,unsigned );  
void BubbleSort ( int* , unsigned );  
void AddFile(char* , char* , int* ,unsigned );  
  
int main(void)  
{  
    char FileName[20];  
    int * Massive;  
    unsigned Number;  
  
    printf("\nEnter the name of file: ");  
    gets(FileName);
```

```

clrscr();
printf("\n The contents of the File <<%s>> "
      ":\n",FileName);

Number = ContentsFile(FileName, "r");
if ( Number == 0 )
{
    clrscr();
    printf("\tFile <<%s>> is empty!\n",FileName);
    printf("\n\t Press any key to exit...");
    getch();
    return 0;
}

Massive = InitMassive(FileName,"r",Number);
if ( Massive == NULL )
{
    clrscr();
    printf("\tDinamic array don't exist!\n");
    printf("\n\t Press any key to exit...");
    getch();
    return 0;
}

BubbleSort(Massive, Number);
AddFile(FileName,"a",Massive,Number);
free(Massive);

printf("\n New contents of the File <<%s>> :\n",
      FileName);
Number = ContentsFile(FileName, "r");

printf("\n Press any key to exit...");
getch();
return 0;
}

int ContentsFile(char* String, char* Mode)
{
    unsigned Number = 0, Term;
    FILE * FilePointer= fopen(String, Mode);

    if (FilePointer == NULL) return Number;
    while (!feof(FilePointer))
    {

```

```

        fscanf(FilePointer, "%d\n", &Term);
        printf( " %d", Term);
        Number ++;
    }

    fclose(FilePointer);
    return Number;
}

int* InitMassive(char* String, char* Mode,
                unsigned Number)
{
    unsigned i;
    int* Pointer = (int*)malloc( Number * sizeof(int) );
    FILE *FilePointer= fopen(String, Mode);

    if (FilePointer== NULL) return NULL;
    for(i = 0; i < Number; i++)
    {
        fscanf(FilePointer, "%d\n",&Pointer[i]);
    }

    fclose(FilePointer);
    return Pointer;
}

void BubbleSort ( int* Pointer, unsigned Number)
{
    unsigned i, j;
    int x;

    for( i = 0; i < Number; i ++)
    {
        for( j = Number - 1 ; j > i; j --)
        {
            if (Pointer [j] < Pointer [j-1] )
            {
                x = Pointer[j];
                Pointer[j] = Pointer[j-1];
                Pointer[j-1] = x;
            }
        }
    }
}

```

```

void AddFile(char* String, char* Mode, int* Pointer,
             unsigned Number)
{
    unsigned i;
    FILE *FilePointer = fopen(String, Mode);

    if (FilePointer == NULL)
    {
        printf("Can't open file to addition!");
        getch();
        exit(1);
    }

    for( i = 0; i < Number; i++)
    {
        fprintf(FilePointer, "%d\n", Pointer[i]);
    }

    fclose(FilePointer);
}

```

Варианты заданий. См. «Лабораторная работа № 4», Лабораторная работа № 5».

Лабораторная работа №.24 Тема: «Бинарные файлы. Файл из структур»

Обобщенная формулировка задания. В соответствии с индивидуальным заданием необходимо создать программу для обслуживания конкретного файла данных. Указанные в задании действия должны быть оформлены в виде отдельных функций.

Выполнение действий по обслуживанию файла данных должно производиться в режиме диалога с пользователем.

Для каждого из вариантов задания необходимо разработать следующие функции:

1. Создание файла.
2. Добавление элемента в конец файла.
3. Вывод содержимого файла на экран.
4. Удаление элемента из файла:
 - элемента с заданным номером;
 - элемента с заданным ключом.
5. Упорядочивание элементов в файле по выбранному признаку.

Пример выполнения задания. На основе структуры вида «Человек» (фамилия, имя, отчество, пол, возраст) создать бинарный файл. Вывести на

экран содержимое файла. Удалить из файла элемент с заданным номером, изменяя при этом размер файла. Добавить элемент в конец файла.

Описание используемых функций. Функция `InitFile` с прототипом

```
void InitFile(char* String, char* Mode, long n);
```

записывает в бинарный файла с именем `String`, открытый в режиме `Mode`, `n` структур.

Функция `InitPerson` с прототипом

```
Person* InitPerson();
```

возвращает указатель на участок динамической памяти со структурой типа `Person`.

Функция `DisplayFile` с прототипом

```
void DisplayFile(char* String, char* Mode);
```

выводит на экран содержимое бинарного файла с именем `String`, открытого в режиме `Mode`.

Функция `AddToEndFile` с прототипом

```
void AddToEndFile(char* String, char* Mode, long n);
```

дописывает в конец бинарного файла с именем `String`, открытого в режиме `Mode`, `n` структур.

Функция `RemoveElementFromFile` с прототипом

```
void RemoveElementFromFile(char* String, char* Mode,  
                             long Position);
```

удаляет одну структуру с порядковым номером `Position` из бинарного файла с именем `String`, открытого в режиме `Mode`.

Функция `DisplayPerson` с прототипом

```
void DisplayPerson (Person* Man);
```

выводит на экран содержимое участка динамической памяти со структурой типа `Person`.

Текст программы.

```
#include <io.h>  
#include <stdio.h>  
#include <conio.h>  
#include <stdlib.h>  
#include <process.h>
```

```
#define TRUE 1
```

```
typedef struct
```

```

{
    char FirstName[15], SecondName[15], LastName[15];
    unsigned Age;
    char Sex [7];
} Person;

void InitFile (char* , char* , long );
Person* InitPerson ();
void DisplayFile (char* , char* );
void DisplayPerson (Person* );
void AddToEndFile (char* , char* , long );
void RemoveElementFromFile (char* , char* , long );

int main(void)
{
    char FileName[20];
    char Key;
    long Number;

    puts("Enter the name of file: ");
    gets(FileName);

    clrscr();
    while(TRUE)
    {
        printf("\n Enter the number - the mode of "
               "operations with file:"
               "\n 1 - FORMATION OF THE FILE"
               "\n 2 - VIEWING OF CONTENTS OF THE FILE"
               "\n 3 - ADD AN ELEMENT TO AND OF FILE"
               "\n 4 - REMOVE THE ELEMENT IN THE FILE"
               "\n 5 - EXIT\n");

        fflush(stdin);
        scanf("%c",&Key);

        clrscr();
        switch (Key)
        {
            case '1':
                while(TRUE)
                {
                    printf("\nEnter the number of "
                           "persons:");
                    scanf("%ld",&Number);

```

```

        if (Number > 0) break;
        printf("\n Number is incorrect!"
               " Try again!!!\n");
    }
    InitFile (FileName,"wb",Number);
    printf("\n\nPress any key to return in "
           "the menu...");
    getch();
    clrscr();
    break;

case '2':
    DisplayFile(FileName,"rb");
    printf("\n\nPress any key to return in "
           "the menu...");
    getch();
    clrscr();
    break;

case '3':
    while(TRUE)
    {
        printf("\nEnter the number of "
               "persons to addition:");
        scanf("%ld",&Number);
        if (Number > 0) break;
        printf("\n Number is incorrect!"
               " Try again!!!\n");
    }
    AddToEndFile(FileName,"ab",Number);
    printf("\n\nPress any key to return in "
           "the menu...");
    getch();
    clrscr();
    break;

case '4':
    printf("\nEnter number of element from "
           "removing :");
    scanf("%ld",&Number);
    RemoveElementFromFile(FileName, "r+b",
                           Number);
    printf("\n\nPress any key to return in "
           "the menu...");
    getch();

```

```

        clrscr();
        break;

    case '5': return 0;
    default:
        printf("\nIncorrect input!"
               " Try again!!!");
        printf("\n\nPress any key to return in"
               " the menu...");
        getch();
        clrscr();
        break;
    }
}

```

```

void InitFile(char* String, char* Mode, long n)
{
    long i;
    int BufSize = sizeof(Person);
    Person* Man;
    FILE* FileStruct = fopen(String, Mode);

    if (FileStruct == NULL)
    {
        printf("Can't open file to write.");
        getch();
        abort();
    }

    for( i = 1; i <= n; i++)
    {
        printf("\nEnter information for the person"
               " number %ld \n", i);
        Man = InitPerson();
        fwrite(Man, BufSize, 1, FileStruct);
    }

    free(Man);
    fclose(FileStruct);
}

```

```

void AddToEndFile(char* String, char* Mode, long n)
{
    long i;

```



```

int BufSize = sizeof(Person);
Person* Man;
FILE* FileStruct = fopen(String, Mode);

if (FileStruct == NULL)
{
    printf("Can't open file to write.");
    getch();
    abort();
}

for( i = 1; i <= n; i++)
{
    printf("\nEnter information for the person "
           "number %ld \n", i);
    Man = InitPerson();
    fwrite(Man, BufSize, 1, FileStruct);
}

free(Man);
fclose(FileStruct);
}

void RemoveElementFromFile(char* String, char* Mode,
long Position)
{
    FILE* FileStruct = fopen(String, Mode);
    int DescriptorFile = fileno(FileStruct);
    long LengthFile = filelength(DescriptorFile);
    int BufSize = sizeof(Person);
    Person* Man = (Person*)malloc(BufSize);
    long NewLength = LengthFile - BufSize;

    if (FileStruct == NULL)
    {
        printf("Can't open file to write.");
        getch();
        abort();
    }

    fseek(FileStruct, Position * BufSize, SEEK_SET);
    while(fread(Man, BufSize, 1, FileStruct) != 0)
    {
        fseek(FileStruct, -2 * (long)BufSize , SEEK_CUR);
    }
}

```

```

        fwrite(Man,BufSize,1, FileStruct);
        fseek(FileStruct, (long) BufSize,SEEK_CUR);
    }

    DescriptorFile = chsize(DescriptorFile, NewLength);
    free(Man);
    fclose(FileStruct);
}

Person* InitPerson()
{
    Person* Man = (Person*)malloc(sizeof(Person));
    printf("\nEnter first name:");
    scanf("%s", Man ->FirstName);
    printf("\nEnter second name:");
    scanf("%s", Man ->SecondName);
    printf("\nEnter last name:");
    scanf("%s", Man ->LastName);
    printf("\nEnter age:");
    scanf("%d",& Man ->Age);
    printf("\nEnter pol:");
    scanf("%s", Man ->Sex);
    return Man;
}

void DisplayFile(char* String, char* Mode)
{
    int BufSize = sizeof(Person);
    Person* Man = (Person*)malloc(BufSize);
    FILE* FileStruct = fopen(String, Mode);

    if ( FileStruct == NULL)
    {
        printf("Can't open file to read.");
        getch();
        abort();
    }

    while(fread(Man,BufSize,1, FileStruct) != 0)
    {
        DisplayPerson(Man);
    }
    free(Man);
    fclose(FileStruct);
}

```

```
void DisplayPerson (Person* Man)
{
    printf("\n%s  %s  %s, %d year, %s ", Man->FirstName,
        Man->SecondName, Man->LastName, Man->Age,
        Man->Sex);
}
```

Варианты заданий. См. «Лабораторная работа № 21».

Лабораторная работа №.25

Тема: «Бинарные файлы. Файл из матриц»

Обобщенная формулировка задания. Сформировать бинарный файл, содержимым которого являются вещественные матрицы (структуры матриц). При этом количество матриц (компонент структуры) и их размерность вводится в процессе выполнения программы с клавиатуры. В соответствии с индивидуальным заданием обработать содержимое полученного файла. Для каждого из вариантов задания вывести содержимое исходного файла на экран до и после преобразования. При работе с матрицами, где это возможно, использовать динамические массивы.

Пример выполнения задания. В файле хранится **k** матриц размерности **n×m**. Для каждой матрицы из файла вычислить сумму элементов. Все матрицы с четными суммами заменить нулевыми матрицами (матрицами, состоящими из нулей).

Описание используемых функций. Функция FreeMemory с прототипом

```
int* FreeMemory(unsigned n, unsigned m);
```

возвращает указатель на участок динамической памяти размером $n*m*sizeof(int)$ для хранения динамической матрицы размерности $n \times m$.

Функция InitMatrix с прототипом

```
int* InitMatrix(unsigned l, unsigned n, unsigned m);
```

возвращает указатель на участок динамической памяти размером $n*m*sizeof(int)$, содержимым которого является матрица, заполненная числом **l**.

Функция SimpleMatrix с прототипом

```
int* SimpleMatrix( unsigned n, unsigned m);
```

возвращает указатель на участок динамической памяти размером $n*m*sizeof(int)$, содержимым которого является нулевая матрица.

Функция SumElemMatrix с прототипом

```
int SumElemMatrix ( int* Pointer, unsigned n,
                    unsigned m);
```

возвращает сумму элементов матрицы, находящейся по адресу Pointer.

Функция DisplayMatrix с прототипом

```
void DisplayMatrix(int* Pointer, unsigned n, unsigned m);
```

выводит на экран содержимое динамической памяти размером $n*m*\text{sizeof}(\text{int})$ по адресу Pointer.

Функция BlockWriteFile с прототипом

```
void BlockWriteFile(char* String, char* Mode, unsigned k,  
                    unsigned n, unsigned m);
```

поматрично записывает в файл с именем String, открытый в режиме Mode, k матриц размерности nxm.

Функция DisplayFile с прототипом

```
void DisplayFile(char* String, char* Mode, unsigned n,  
                unsigned m);
```

поматрично выводит на экран содержимое файла с именем String, открытый в режиме Mode.

Функция WorkFile с прототипом

```
void WorkFile(char* String, char* Mode, unsigned n,  
              unsigned m);
```

поматрично обрабатывает согласно постановке задачи содержимое файла с именем String, открытый в режиме Mode.

Текст программы.

```
#define TRUE 1
```

```
#include <stdio.h>  
#include <conio.h>  
#include <process.h>  
#include <stdlib.h>
```

```
void BlockWriteFile(char* ,char* , unsigned ,  
                    unsigned , unsigned );  
int* InitMatrix(unsigned ,unsigned , unsigned );  
int* FreeMemory(unsigned , unsigned );  
void DisplayFile( char* , char* , unsigned , unsigned );  
void DisplayMatrix (int* , unsigned , unsigned );  
int* SimpleMatrix(unsigned , unsigned );  
void WorkFile( char* , char* , unsigned , unsigned );  
int SumElemMatrix ( int* , unsigned , unsigned );
```

```
int main(void)
```

```

{
    unsigned k, n, m ;
    char FileName [20];

    while(TRUE)
    {
        printf("\nEnter k number of matrixs:\n");
        scanf("%u",&k);
        printf("\nEnter n, m dimentions of matrixs:\n");
        scanf("%u%u",&n,&m);
        if ( (k > 0) && (n > 0) && (m > 0) ) break;
        printf("\nNumber is incorrect!!! Try "
               "again!!!\n");
    }

    printf("\nEnter the name of file: \n");
    scanf("%s",FileName);

    clrscr();
    BlockWriteFile(FileName, "wb", k, n, m);

    printf("\nThe contents of file <<%s>>:\n",FileName);
    DisplayFile(FileName, "rb", n, m);
    WorkFile(FileName, "r+b", n, m);

    printf("\nThe new contents of file <<%s>>:\n",
           FileName);
    DisplayFile(FileName, "rb", n, m);

    printf("\n Press any key to exit...");
    getch();
    return 0;
}

int* InitMatrix(unsigned l, unsigned n, unsigned m)
{
    unsigned i;
    int* Pointer = (int*)malloc(n*m*sizeof(int));

    for( i = 0; i < n * m; i++)
    {
        Pointer[i] = l + 1;
    }
    return Pointer;
}

```

```

int* FreeMemory(unsigned n, unsigned m)
{
    int* Pointer = (int*)malloc(n*m*sizeof(int));
    return Pointer;
}

void BlockWriteFile(char* String, char* Mode,
                    unsigned k, unsigned n, unsigned m)
{
    int BufSize = sizeof(int) * n * m;
    int* Pointer = (int*)malloc(BufSize);
    unsigned i;
    FILE* FilePointer= fopen(String, Mode);

    if (FilePointer== NULL)
    {
        printf("Can't open file to write.");
        getch();
        abort();
    }

    for ( i = 0; i < k; i++ )
    {
        Pointer = InitMatrix(i, n, m);
        fwrite(Pointer, BufSize,1, FilePointer);
    }

    fclose(FilePointer);
    free(Pointer);
}

void DisplayFile(char* String, char* Mode, unsigned n,
                 unsigned m)
{
    int BufSize = sizeof(int)*n*m, Sector = 0;
    int* Pointer = FreeMemory(n, m);
    FILE* FilePointer= fopen(String, Mode);

    if ( FilePointer== NULL)
    {
        printf("\nCan't open file to read.");
        getch();
        abort();
    }
}

```

```

while(fread(Pointer,BufSize,1, FilePointer) != 0)
{
    printf("\n %d's  matrix \n",(Sector + 1));
    DisplayMatrix(Pointer, n, m);
    Sector++;
}

fclose(FilePointer);
free(Pointer);
}

void DisplayMatrix(int* Pointer, unsigned n, unsigned m)
{
    unsigned i, j;

    for( i = 0; i < n; i++)
    {
        for( j = 0; j < m; j++)
        {
            printf("%4d",*(Pointer + i * m + j));
        }
        printf("\n");
    }
}

int* SimpleMatrix(unsigned n, unsigned m)
{
    unsigned i;
    int* Pointer = (int*)malloc(sizeof(int)*n*m);
    for( i = 0; i < n * m; i++)
    {
        *(Pointer + i) = 0;
    }
    return Pointer;
}

int SumElemMatrix ( int* Pointer, unsigned n, unsigned m)
{
    int s = 0;
    unsigned i;
    for ( i = 0; i < n * m; i++ )
    {
        s = s + *(Pointer + i);
    }
}

```

```

        return s;
    }

void WorkFile(char* String, char* Mode, unsigned n,
              unsigned m)
{
    int* Pointer = FreeMemory(n, m);
    int BufSize = sizeof(int)*n*m;
    int* Simple, Sum = 0;
    FILE* FilePointer= fopen(String, Mode);

    if ( FilePointer== NULL)
    {
        printf("Can't open file to read.");
        getch();
        abort();
    }
    while(fread(Pointer, BufSize,1,FilePointer) != 0)
    {
        Sum = SumElemMatrix(Pointer, n, m);
        if ( Sum % 2 == 0 )
        {
            fseek(FilePointer, -1L * BufSize, SEEK_CUR);
            Simple = SimpleMatrix(n, m);
            fwrite(Simple, BufSize,1, FilePointer);
            fseek(FilePointer, 0L, SEEK_CUR);
        }
    }
    fclose(FilePointer);
    free(Pointer);
    free(Simple);
}

```

Варианты заданий.

1. В первом файле хранится k матриц, во втором - l матриц размерности $n \times m$. Те матрицы из первого файла, у которых $a_{00} = 0$, перенести в конец второго файла. Вывести на экран содержимое первого и второго файлов.
2. В первом файле хранится k матриц, во втором l матриц размерности $n \times m$. Убрать из файла, в котором больше матриц, лишние матрицы в третий файл. Вывести на экран содержимое первого файла; второго файла; третьего файла.
3. Файл состоит из k компонент структуры, где каждая компонента содержит две матрицы: первая размерности $n \times m$, вторая размерности $m \times l$.

Получить k произведений соответствующих матриц и записать их во второй файл. Вывести на экран содержимое первого и второго файлов.

4. В первом файле хранится k матриц, во втором l матриц размерности $n \times m$. Добавить во второй файл те матрицы из первого, которых нет во втором. Вывести на экран содержимое первого и второго файлов.

5. В первом файле хранится k матриц из n строк и $n + 1$ столбцов каждая (последний столбец – столбец свободных членов). Во втором файле хранится k векторов- результатов решений соответствующих систем ЛАУ с матрицами из первого файла. Вывести на экран покомпонентно исходную систему уравнений и результат, проверив его предварительно; добавить в файлы новые данные; удалить ненужную информацию.

6. В файле хранится k матриц размерности $n \times m$. Для каждой матрицы из файла вычислить сумму её положительных четных элементов. Все матрицы с четными суммами записать в другой файл, заменив их в исходном файле единичными матрицами. Вывести на экран содержимое первого и второго файлов.

7. В первом файле хранится k матриц, во втором – l матриц размерности $n \times m$. Поменять местами все нечетные (по порядковому номеру в файле) матрицы из первого и второго файлов (до конца меньшего из файлов). Вывести на экран содержимое первого и второго файлов.

8. В первом файле хранится k квадратных матриц порядка n , во втором – l квадратных матриц. Если $k \neq l$, то в файл с меньшим числом матриц добавить в конец файла недостающее количество единичных матриц. Вывести на экран содержимое первого и второго файлов.

9. В файле хранится k матриц размерности $n \times n$. Для каждой матрицы из файла вычислить сумму её диагональных элементов. Все матрицы с нечетными суммами записать в другой файл, заменив их в исходном файле транспонированными матрицами. Вывести на экран содержимое первого и второго файлов.

10. В первом файле хранится k квадратных матриц. Записать в другой файл из исходного файла все симметрические матрицы ($A = A^T$), в третий файл – остальные. Вывести на экран содержимое первого, второго и третьего файлов.

11. В первом файле хранится k матриц размерности $n \times m$, во втором – k матриц размерности $m \times l$. Получить k произведений соответствующих матриц из первого и второго файлов и записать их в третий файл в виде компонент структуры, где каждая компонента содержит три матрицы: первая размерности $n \times m$ из первого файла; вторая размерности $m \times l$ из второго файла; третья, матрица размерности $n \times l$, результат произведения. Вывести на экран содержимое первого и второго файлов.

12. В первом файле хранится k матриц порядка $n \times m$, во втором – l матриц. Поменять местами все нечетные (1, 3, 5, ... по порядковому номеру в файле) матрицы из первого файла с четными матрицами (0, 2, 4, ...) второго файла (до конца меньшего из файлов). Оставшиеся в большем файле матрицы переписать в третий файл. Вывести на экран содержимое первого и второго файлов.

СОДЕРЖАНИЕ

Лабораторная работа № 1. Тема: «Системы счисления. Арифметические операции в разных системах счисления. Перевод из одной системы счисления в другую»	4
Лабораторная работа № 2. Тема: «Представление информации в ПЭВМ типа IBM PC/AT»	11
Лабораторная работа № 3. Тема: «Основы алгоритмизации. Построение блок-схем линейных и разветвляющихся вычислительных процессов»	17
Лабораторная работа № 4. Тема: «Построение блок-схем циклических вычислительных процессов»	23
Лабораторная работа № 5. Тема: «Действия над одномерными массивами в блок-схемах»	29
Лабораторная работа № 6. Тема: «Действия над матрицами в блок-схемах»	35
Лабораторная работа № 7. Тема: «Построение блок-схем итерационных вычислительных процессов»	40
Лабораторная работа № 8. Тема: «Условный оператор if....»	44
Лабораторная работа № 9. Тема: «Оператор цикла for....»	45
Лабораторная работа № 10. Тема: «Оператор цикла while....»	46
Лабораторная работа № 11. Тема: «Обработка одномерных массивов»	47
Лабораторная работа № 12. Тема: «Обработка матриц»	50
Лабораторная работа № 13. Тема: «Обработка строк»	52
Лабораторная работа № 14. Тема: «Табулирование функций»	54
Лабораторная работа № 15. Тема: «Приближенное вычисление определенных интегралов»	57
Лабораторная работа № 16. Тема: «Решение нелинейных уравнений»	61
Лабораторная работа № 17. Тема: «Указатели и массивы. Индексация с помощью указателей. Передача массивов в функции»	65
Лабораторная работа № 18. Тема: «Массивы динамической памяти»	67
Лабораторная работа № 19. Тема: «Массивы указателей и моделирование многомерных массивов»	70
Лабораторная работа № 20. Тема: «Указатели на функции»	79
Лабораторная работа № 21. Тема: «Структуры, массивы, указатели. Динамические массивы структур»	82

Лабораторная работа № 22. Тема: «Динамические структуры данных. Односвязный список»	88
Лабораторная работа № 23. Тема: «Текстовые файлы из чисел и матриц»	96
Лабораторная работа № 24. Тема: «Бинарные файлы. Файл из структур»	100
Лабораторная работа № 25. Тема: «Бинарные файлы. Файл из матриц»	107
СОДЕРЖАНИЕ	114
ЛИТЕРАТУРА	115

ЛИТЕРАТУРА

1. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е издание. — М.: «Вильямс», 2013. — 1328 с.
2. Кнут Д. Э. Искусство программирования. Том 1. Основные алгоритмы / под ред. Ю. В. Козаченко. — 3. — Москва: Вильямс, 2002. — Т. 1. — 720 с.
3. Кнут Д. Э. Искусство программирования. Том 2. Получисленные алгоритмы / под ред. Ю. В. Козаченко. — 3. — Москва: Вильямс, 2001. — Т. 2. — 832 с.
4. Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск / под ред. Ю. В. Козаченко. — 2. — Москва: Вильямс, 2007. — Т. 3. — 832 с.
5. Кнут Д. Э. Искусство программирования, том 4, А. Комбинаторные алгоритмы, часть 1 / под ред. Ю. В. Козаченко. — 1. — Москва: Вильямс, 2013. — Т. 4. — 832 с.
6. Брайан Керниган, Деннис Ритчи. Язык программирования С. — Москва: Вильямс, 2015. — 304 с.
7. Вирт Н. Алгоритмы и структуры данных. — М., 1989, — 360 с.
8. Подбельский В.В. Программирование на языке СИ. — М., 2001, — 600 с.
9. Абрамов С.А. и др. Задачи по программированию. Москва, «Наука», 1988, 224 с.