Министерство образования и науки Украины
СЕВАСТОПОЛЬСКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Практикум по алгоритмизации и программированию

Методические указания к практическим занятиям
по дисциплине «Алгоритмизация и программирование»
для студентов направления 6.050101 — «Компьютерные науки»

Севастополь
2013

УДК 681.3 (075)

Практикум по алгоритмизации и программированию: методические указания к практическим занятиям по дисциплине «Алгоритмизация и программирование» для студентов направления 6.050101 — «Компьютерные науки» / СевНТУ ; сост. Т.И. Сметанина, В. Н. Бондарев. — Севастополь: Изд-во СевНТУ, 2013. — 82с.

Методические указания предназначены для проведения практических занятий и организации самостоятельной работы студентов всех форм обучения направления 6.050101 — «Компьютерные науки» при изучении дисциплины «Алгоритмизация и программирование»» **Цель методических указаний** — раскрыть содержание и методику проведения практических занятий, обеспечить студентов необходимым набором задач для самостоятельной работы и примерами их решений.

Методические указания составлены в соответствии с требованиями программы дисциплины «Алгоритмизация и программирование» для студентов направления 6.050101 — «Компьютерные науки» и утверждены на заседании кафедры Информационных систем (протокол № 13 от 25 мая 2013 г.)

Допущено учебно-методическим центром СевНТУ в качестве методических указаний для проведения практических занятий.

Рецензент:

Кожаев Е.А., канд. техн. наук, доцент кафедры кибернетики и вычислительной техники.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Практические занятия по дисциплине «Алгоритмизация программирование» (АиП) являются важным видом учебной работы студентов дневной формы обучения. В соответствии с учебным планом на их выполнение отводится 18 часов аудиторных занятий и 36 часов самостоятельной работы.

Цель практических занятий — систематизировать, закрепить и углубить знания теоретического материала; научить студентов приемам разработки алгоритмов различных типов, способствовать овладению навыками и умениями программирования задач обработки основных типов данных алгоритмических языков.

Основными **задачами** практических занятий являются:

- закрепление, углубление и расширение теоретических знаний, полученных на лекциях;
- обучение студентов системам счисления и форматам представления простых типов данных в памяти ЭВМ;
- обучение студентов практическим приемам и методам разработки алгоритмов линейной, разветвляющейся и циклической структуры;
- приобретение студентами умений и навыков использования синтаксических конструкций структурного программирования языка Паскаль;
- приобретение студентами умений и навыков работы с основными типами данных языка Паскаль.

Порядок проведения занятий предусматривает следующие этапы:

- формулировка темы, целей и задач занятия, пояснение связи темы с другими темами учебной дисциплины и занятиями;
- проверка готовности студентов к занятию;
- краткое обсуждение теоретических сведений, раскрывающих сущность изучаемой темы;
- коллективное решение задач основных типов в аудитории;
- выполнение самостоятельной аудиторной работы;
- выдача домашнего задания и подведение итогов занятия.

Цели и задачи отдельных этапов занятий, а также возможные формы их проведения указаны в табл. 1.

Таблица 1. — Цели, задачи и формы проведения этапов занятий

Этапы проведения практических занятий	Цели, задачи этапа.	Формы проведения
1	2	3
1. Обсуждение теоретических сведений	Актуализация материала, проверка теоретической подготовленности студентов	-Устный опрос; - Письменный опрос;

Продолжение таблицы 1.

Этапы проведения практических занятий	Цели, задачи этапа.	Формы проведения
1	2	3
2. Коллективное решение задач основных типов в аудитории	- Предупредить появление часто допускаемых ошибок; - Проанализировать возможности использования тех или иных типов данных; - Сформировать определенные умения (например, по переводу готовых алгоритмов на язык Паскаль) и др.	- Выполнение у доски с объяснением; - Выполнение в тетради, с последующим анализом результатов
3. Выполнение самостоятельной аудиторной работы и домашней работы	Формирование умений: - постановки задачи; - формулированию математической модели; - структурного программирования; - анализу правильности алгоритмов; - подготовки системы тестов для тестирования программы и др.	- Коллективно; - Индивидуально.

Студент обязан:

- перед занятиям ознакомиться с целью и задачами занятия;
- проработать теоретический материал занятия по лекционному курсу;
- активно работать в ходе занятия; выполнить самостоятельную аудиторную работу;
- оформить результаты выполнения домашнего задания в отдельной тетради (для некоторых заданий на отдельных листах белой бумаги формата А4 в виде расчетно-графического задания).

Методические указания могут использоваться для выдачи контрольных заданий студентам заочной формы обучения в первом семестре.

1. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1

Позиционные системы счисления. Двоичная, восьмеричная и шестнадцатеричная системы счисления. Расширенная форма записи числа, правила перевода чисел из различных систем счисления в десятичную систему. Перевод целых чисел из десятичной системы в иные системы.

1.1. Цели и задачи занятия

Изложить основные понятия позиционных систем счисления и научить студентов правилам перевода чисел из различных систем в счисления в десятичную систему. Научить студентов правилам перевода целых десятичных чисел в иные системы счисления.

1.2. План занятия

1.2.1. Контроль посещаемости, цель и задачи занятия (5 мин.).
1.2.2. Обсуждение теоретических вопросов (25 мин.):
- позиционные системы счисления;
- двоичная, восьмеричная и шестнадцатеричная системы счисления;
- расширенная форма записи числа;
- правила перевода чисел из различных систем счисления в десятичную систему;
- перевод целых чисел из десятичной системы в иные системы.

1.2.3. Решение аудиторных задач (20 мин.).
1.2.4. Выполнение самостоятельных аудиторной работы (20 мин.).
1.2.5. Выдача домашнего задания (5 мин.).
1.2.6. Подведение итогов занятия (5 мин.).

1.3. Содержание занятия

1.3.1. Краткие сведения о позиционных системах и правилах перевода в десятичную систему счисления

Система счисления (СС) — совокупность приёмов обозначения чисел.

Системы счисления делятся на 2 группы:
- позиционные (пример — десятичные числа);
- непозиционные (пример — римские числа).

В ***позиционных СС*** значение каждой цифры в изображении числа зависит от её положения (позиции) в последовательности цифр, изображающих число.

Номер позиции цифры считается справа налево в целой части числа (рис. 1.1.):

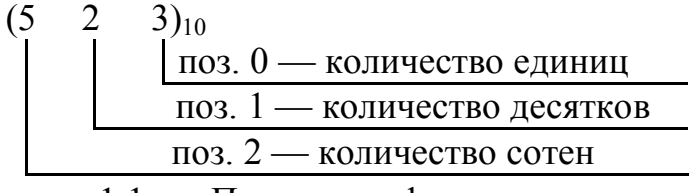


Рисунок 1.1. — Позиции цифр в записи целого числа

Основанием СС (p) называют количество цифр, использующихся для записи числа. Ниже приведены цифры и символы, используемые для записи чисел в десятичной (p = 10), восьмеричной (p = 8), шестнадцатеричной (p = 16) и двоичной (p = 2) СС.

p = 10; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
p = 8; 0, 1, 2, 3, 4, 5, 6, 7;
p = 16; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F;
p = 2; 1, 0.

В табл. 1.1 приведены примеры записи целых чисел в различных СС.

Таблица 1.1 – Числа в различных системах счисления

Десятичная СС	Восьмеричная СС	Шестнадцатеричная СС	Двоичная СС
0	0	0	00
1	1	1	01
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111

Известно, что числа можно записывать в виде разложения по степеням основания, например:

$$(523,5)_{10} = 5 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 5 \cdot 10^{-1}.$$

Такая запись называется ***расширенной записью числа***. В общем случае для записи числа в расширенной форме используется формула:

$$N = C_n \cdot p^n + C_{n-1} \cdot p^{n-1} + \ldots + C_1 \cdot p^1 + C_0 \cdot p^0 + C_{-1} \cdot p^{-1} + C_{-2} \cdot p^{-2} + \ldots, \quad (1.1)$$

где C_i — цифра в i-ой позиции числа, p — основание СС, в которой записано число.

Расширенную запись числа используют для перевода чисел из других систем счисления в десятичную.

Пример 1.1. Перевести числа в десятичную СС:

а) $(1011)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = (11)_{10}$;

б) $(341)_8 = 3 \cdot 8^2 + 4 \cdot 8^1 + 1 \cdot 8^0 = (225)_{10}$;

в) $(FA)_{16} = 15 \cdot 16^1 + 10 \cdot 16^0 = (250)_{10}$.

Формула расширенной записи числа может применяться и для обратного перевода, но при этом придется выполнять математические операции по правилам той системы, в которую выполняется перевод.

Пример 1.2. Перевести десятичные числа в двоичную СС:

а) $(9)_{10} = (1\ 001)_2$;

б) $(17)_{10} = 1\cdot10^1 + 7\cdot10^0 + \ = (1010)_2 + (111)_2 = (10\ 001)_2$.

1.3.2. Краткие сведения о переводе целых чисел из десятичной системы в иные системы

Рассмотрим перевод целого десятичного числа N в двоичную систему счисления.

Пусть $N = C_n \cdot p^n + ... + C_1 \cdot p^1 + C_0 \cdot p^0$. Поделим N на 2, тогда частное будет равно $C_n \cdot 2^n + ... + C_1$, а остаток — C_0.

Полученное частное опять разделим на 2, остаток от деления будет равен C_1 и т.д. (продолжим процесс деления частного на 2).

На n шаге получим набор остатков $C_0, C_1, C_2, ..., C_n$, которые соответствуют двоичным цифрам в записи числа N. Но мы получили их в обратном порядке. Поэтому их нужно записать в виде:

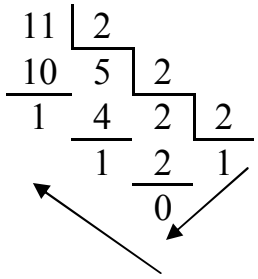
$N_{10} = (C_n\ C_{n-1} ...\ C_1\ C_0)_2$.

Общий алгоритм перевода целого числа из десятичной системы в систему с основанием p:

- последовательно выполнять деление целого десятичного числа и получаемых целых частных на ***p*** до тех пор, пока не получится частное, меньшее ***p***;
- записать полученные остатки в обратной последовательности.

Пример 1.3. Перевести число 11 из десятичной системы счисления в двоичную систему.

Выполним процесс деления.



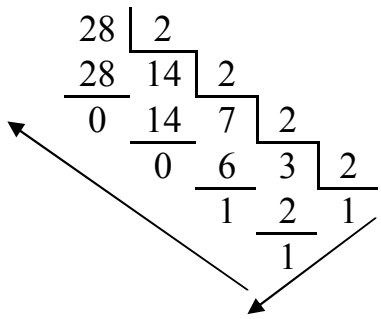
Соберем остатки от деления в направлении, указанном стрелкой, начиная с последней единицы, и получим число в двоичной системе счисления:

$$(11)_{10} = (1011)_2.$$

Сравните полученный результат с данными табл. 1.1.

Пример 1.4. Перевести число 28 из десятичной системы счисления в двоичную систему.

Решение:



Результат $(28)_{10} = (11100)_2$.

Пример 1.5. Перевести число 363 из десятичной СС в двоичную СС. Процесс деления удобно представлять в табличной форме (табл. 1.1.):

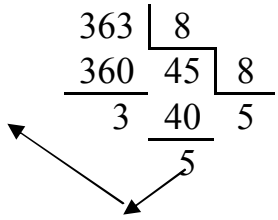
Таблица 1.1. — Перевод числа 363 из десятичной СС в двоичную СС.

Число \ частное	363	181	90	45	22	11	5	2	1
Делитель	2	2	2	2	2	2	2	2	2
Остаток	1	1	0	1	0	1	1	0	1

Собрав остатки от деления, в направлении, указанном стрелкой, получим число в двоичной системе счисления: $(363)_{10} = (101\ 101\ 011)_2$

Пример 1.6. Перевести число 363 из десятичной системы счисления в восьмеричную систему.

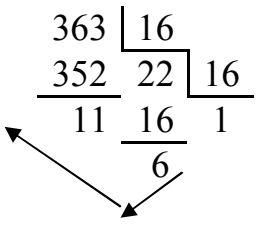
Решение:



Результат $(363)_{10} = (553)_8$.

Пример 1.7. Перевести число 363 из десятичной системы счисления в шестнадцатеричную систему.

Решение:



Результат $(363)_{10} = (16B)_{16}$.

Обратите внимание, что чем больше **p**, тем короче запись числа и проще вычисления!

1.3.3. Аудиторные задачи и примеры их решения

Задача 1.1. Перевести числа в десятичную СС:

а) $(1010)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (10)_{10}$;

б) $(10{,}11)_2 = 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{1/2} + 0 \cdot 2^{1/4} = (2{,}75)_{10}$;

в) $(34,1)_8 = 3 \cdot 8^1 + 4 \cdot 8^0 + 1 \cdot 8^{1/8} = (28,125)_{10}$;

г) $(F,A)_{16} = 15 \cdot 16^0 + 10 \cdot 16^{-1} = (15,625)_{10}$.

Задача 1.2. Перевести десятичные числа в двоичную СС с использованием формулы расширенной записи числа:

а) $(15)_{10} = 1 \cdot 10^1 + 5 \cdot 10^0 + = (1010)_2 + (101)_2 = (1111)_2$;

б) $(243)_{10} = 2 \cdot 10^2 + 4 \cdot 10^1 + 3 \cdot 10^0 = (10)_2 \cdot (1010)_2{}^2 + (100)_2 \cdot (1010)_2 + (11)_2 =$
$= (10)_2 \cdot (1\ 100\ 100)_2 + (101\ 000)_2 + (11)_2 = (11\ 001\ 000)_2 + (101\ 000)_2 + (11)_2 =$
$= (11\ 110\ 011)_2$.

1.3.4. Задания для самостоятельной аудиторной работы

Задача 1.3.: Перевести числа из двоичной СС в десятичную СС:

а) 11 101; в) 11 011;

б) 10 001; г) 101,01;

Ответы: а) $(29)_{10}$; б) $(17)_{10}$; в) $(27)_{10}$; г) $(5,25)_{10}$.

Задача 1.4.: Перевести числа из восьмеричной СС в десятичную СС:

а) 25; в) 2012;

б) 37; г) 510,1;

Ответы: а) $(28)_{10}$; б) $(31)_{10}$; в) $(1034)_{10}$; г) $(328,125)_{10}$.

Задача 1.5.: Перевести числа из шестнадцатеричной СС в десятичную С

а) 16А; в) 712;

б) 50С; г) 25,8;

Ответы: а) $(362)_{10}$; б) $(1292)_{10}$; в) $(1810)_{10}$; г) $(37,5)_{10}$.

Задача 1.6. Перевести числа из десятичной СС в двоичную СС, в восьмеричную СС и шестнадцатеричную СС:

а) 58; б) 24; в) 102.

Ответы: а) $(58)_{10} = (3A)_{16} = (111\ 010)_2 = (72)_8$;

б) $(24)_{10} = (18)_{16} = (11\ 000)_2 = (30)_8$;

в) $(102)_{10} = (66)_{16} = (1\ 100\ 110)_2 = (146)_8$.

1.3.5. Задания для домашней работы

ВАРИАНТ 1.1.

1. Перевести числа из двоичной СС в десятичную СС:

а) $(101\ 010)_2$; б) $(100\ 111)_2$.

Ответы: а) $(42)_{10}$; б) $(39)_{10}$.

2. Перевести числа из восьмеричной СС в десятичную СС:

а) $(246)_8$; б) $(643)_8$.

Ответы: а) $(166)_{10}$; б) $(419)_{10}$.

3. Перевести числа из шестнадцатеричной СС в десятичную СС:

а) $(B05)_{16}$; б) $(5F,5)_{16}$.

Ответы: а) $(2821)_{10}$; б) $(95,3125)_{10}$.

4**.** Перевести числа из десятичной СС в двоичную СС, в восьмеричную СС и шестнадцатеричную СС:

а) 55; б) 215.

Ответы: а) $(55)_{10} = (37)_{16} = (110\ 111)_2 = (67)_8$; б) $(215)_{10} = (D7)_{16} = (11\ 010\ 111)_2 = (327)_8$.

ВАРИАНТ 1.2.

1. Перевести числа из двоичной СС в десятичную СС

а) $(101\ 101)_2$; б) $(110\ 010)_2$.

Ответы: а) $(45)_{10}$; б) $(50)_{10}$.

2. Перевести числа из восьмеричной СС в десятичную СС:

а) $(352)_8$; б) $(522)_8$.

Ответы: а) $(234)_{10}$; б) $(338)_{10}$.

3. Перевести числа из шестнадцатеричной СС в десятичную СС:

а) $(C90)_{16}$; б) $(4E,2)_{16}$.

Ответы: а) $(3216)_{10}$; б) $(78,125)_{10}$.

4**.** Перевести числа из десятичной СС в двоичную СС, в восьмеричную СС и шестнадцатеричную СС:

а) 77; б) 315.

Ответы: а)$(77)_{10}=(4D)_{16}=(1001101)_2=(115)_8$; б)$(315)_{10}=(13B)_{16}=(100111011)_2=(473)_8$.

2. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2

Перевод дробных чисел из десятичной системы в иные системы. Непосредственный перевод чисел представленных в системах с основанием 2, 8 и 16. Арифметические операции в различных системах счисления.

2.1. Цели и задачи занятия

Научить студентов правилам перевода дробных десятичных чисел в иные системы счисления, быстрому переводу чисел в системах с основанием кратным двум, а также научить студентов правилам выполнения арифметических операций в различных системах счисления.

2.2. План занятия

2.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
2.2.2. Цель и задачи занятия (5 мин.).
2.2.1. Обсуждение теоретических вопросов (15 мин.):
- перевод дробных чисел из десятичной системы в иные системы;
- быстрый перевод чисел в системах с основанием 2, 8 и 16;
- выполнение арифметических операций в различных системах счисления.

2.2.2. Решение аудиторных задач (20 мин.).
2.2.3. Выполнение самостоятельных аудиторной работы (20 мин.).
2.2.4. Выдача домашнего задания (5 мин.).
2.2.5. Подведение итогов занятия (5 мин.).

2.3. Содержание занятия

2.3.1. Краткие сведения о переводе дробных чисел из десятичной системы в иные системы

Рассмотрим перевод правильной десятичной дроби в двоичную систему счисления. Пусть $A_{др}$ — правильная десятичная дробь, тогда её можно записать в виде:

$$A_{др} \approx a_{-1} \cdot 2^{-1} + a_{-2} \cdot 2^{-2} + \ldots$$

Если $A_{др}$ умножить на 2 , то в правой части получим $a_{-1}+a_{-2}\cdot 2^{-1}+a_{-3}\cdot 2^{-2}+\ldots$, где a_{-1} — целая часть, она и даст нам старший коэффициент в разложении числа $A_{др}$ по степеням 2. Оставшуюся дробную часть снова умножим на 2 и получим

$$a_{-2} + a_{-3} \cdot 2^{-1} + \ldots ,$$

где a_{-2} — второй коэффициент после запятой в двоичном представлении числа. Процесс продолжается до тех пор, пока в правой части не получим 0 или не будет достигнута требуемая точность вычислений.

Алгоритм перевода правильной десятичной дроби в систему с основанием p:

- последовательно выполнять умножение десятичной дроби и получаемых дробных частей произведения на ***p*** до тех пор, пока не получится нулевая дробная часть или не будет достигнута требуемая точность;

- записать полученные целые части произведения в порядке их вычисления.

Пример 2.1. Перевести число 0,75 из десятичной системы счисления в двоичную систему.

Решение:

$$\begin{array}{l} 0{,}75 \cdot 2 = \vert 1, \vdots 5 \\ 0{,}50 \cdot 2 = \vert 1, \vdots 0 \end{array} \downarrow$$

Результат: $(0{,}75)_{10} = (0{,}11)_2$.

Сделаем проверку: $(0{,}11)_2 = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 1/2 + 1/4 = 0{,}5 + 0{,}25 = 0{,}75$.

Пример 2.2. Перевести число 0,24 из десятичной системы счисления в двоичную систему.

Решение:

$$\begin{array}{l} 0{,}24 \cdot 2 = \vert 0, \vdots 48 \\ 0{,}48 \cdot 2 = \vert 0, \vdots 96 \\ 0{,}96 \cdot 2 = \vert 1, \vdots 92 \\ 0{,}92 \cdot 2 = \vert 1, \vdots 84 \\ 0{,}84 \cdot 2 = \vert 1, \vdots 68 \end{array} \downarrow$$

Результат: $(0{,}24)_{10} \approx (0{,}00111)_2$.

Этот процесс может продолжаться бесконечно, его обрывают на том шаге, когда считают, что получена требуемая точность.

Обратите внимание, что в общем случае дробные десятичные значения представляются в двоичной СС приблизительно, т.е. с погрешностью.

Пример 2.3. Перевести число 0,24 из десятичной системы счисления в восьмеричную СС.

Решение:

$$\begin{array}{l} 0{,}24 \cdot 8 = \vert 1, \vdots 92 \\ 0{,}92 \cdot 8 = \vert 7, \vdots 36 \\ 0{,}36 \cdot 8 = \vert 2, \vdots 88 \\ 0{,}88 \cdot 8 = \vert 7, \vdots 04 \\ 0{,}04 \cdot 8 = \vert 0, \vdots 32 \end{array} \downarrow$$

Результат: $(0{,}24)_{10} \approx (0{,}17270)_8$.

Пример 2.4. Перевести число 0,24 из десятичной системы счисления в шестнадцатеричную СС.

Решение:

$$\begin{array}{l} 0{,}24 \cdot 16 = \vert 3, \vdots 84 \\ 0{,}84 \cdot 16 = \vert 13, \vdots 44 \\ 0{,}44 \cdot 16 = \vert 7, \vdots 04 \\ 0{,}04 \cdot 16 = \vert 0, \vdots 64 \\ 0{,}64 \cdot 16 = \vert 10, \vdots 24 \end{array} \downarrow$$

Результат: $(0{,}24)_{10} \approx (0{,}3D70A)_8$.

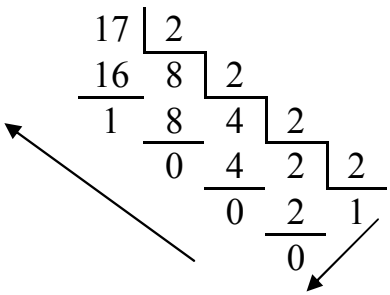
Обратите внимание, что чем больше ***p***, тем выше точность при одном и том же количестве цифр после запятой!

А если число смешанное? Тогда нужно отдельно перевести целую часть и отдельно — дробную. ***Алгоритм перевода смешанного десятичного числа в системы с основанием p:***

- перевести целую часть;
- перевести дробную часть;
- сложить полученные результаты.

Пример 2.5. Перевести число $(17{,}225)_{10}$ из десятичной системы счисления в двоичную систему.

Решение:

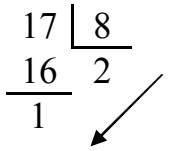


$$
\begin{array}{l|l}
0{,}225 \cdot 2 = & 0{,}45 \\
0{,}45 \cdot 2 = & 0{,}9 \\
0{,}9 \cdot 2 = & 1{,}8 \\
0{,}8 \cdot 2 = & 1{,}6 \\
0{,}6 \cdot 2 = & 1{,}2 \\
0{,}2 \cdot 2 = & 0{,}4 \\
0{,}4 \cdot 2 = & 0{,}8
\end{array}
$$

Результат: $(17{,}225)_{10} \approx (10001{,}0011100)_2$.

Пример 2.6. Перевести число $(17{,}225)_{10}$ из десятичной системы счисления в восьмеричную систему.

Решение:

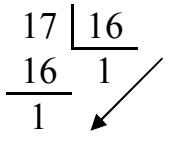


$$
\begin{array}{l|l}
0{,}225 \cdot 8 = & 1{,}8 \\
0{,}8 \cdot 8 = & 6{,}4 \\
0{,}4 \cdot 8 = & 3{,}2
\end{array}
$$

Результат: $(17{,}25)_{10} \approx (21{,}163)_8$.

Пример 2.7 Перевести число $(17{,}225)_{10}$ из десятичной системы счисления в шестнадцатеричную систему.

Решение:



$$
\begin{array}{l|l}
0{,}225 \cdot 16 = & 3{,}6 \\
0{,}6 \cdot 16 = & 9{,}6 \\
0{,}6 \cdot 16 = & 9{,}6
\end{array}
$$

Результат: $(17,25)_{10} \approx (11,399)_{16}$.

2.3.2. Непосредственный перевод чисел, представленных в системах с основанием 2, 8 и 16

Для непосредственного перевода чисел из двоичной в восьмеричную и обратно (из восьмеричной в двоичную) используется правило триад: ***каждая десятичная цифра записывается тремя двоичными и наоборот.***

Приведем примеры перевода из двоичной СС в восьмеричную и оборатно:

Пример 2.8.

(2↔8): 011 100 = (3 4)$_8$

3 4 011 100

Пример 2.9.

(2↔8): 111 101 001 = (7 5 1)$_8$

7 5 1 111 101 001

Для перевода чисел из двоичной СС в шестнадцатеричную СС применяется правило тетрад: **каждая шестнадцатеричная цифра записывается четырьмя двоичными и наоборот.** Приведем примеры перевода чисел из двоичной СС в шестнадцатеричную СС и обратно:

Пример 2.10.

(2↔16): (1 1100)$_2$= (1 C)$_{16}$

1 C 0001 1100

Пример 2.11.

(1010 0101 0011)$_2$ = (A 5 3)$_{16}$

A 5 3 1010 0101 0011

Для быстрого перевода десятичного числа в двоичную систему следует его сначала перевести в шестнадцатеричную систему, а затем, используя правило тетрад — в двоичную.

Пример 2.12. Перевести число 2367,1 из десятичной СС в двоичную СС:

Решение:

2367 | 16
2352 | 147 | 16
15 | 144 | 9
3

0,1 · 16 = | 1,6
0,6 · 16 = | 9,6
0,6 · 16 = | 9,6

Результат:$(2367,1)_{10} \approx (93F,199)_{16} = (1001\ 0011\ 1111,0001\ 1001\ 1001)_2 \approx (93F,199)_{16}$

2.3.3. Краткие сведения о выполнении арифметических операций в различных системах

Арифметические операции в позиционных системах счисления выполняются с помощью таблиц сложения и умножения.

Сложение. Рассмотрим сложение чисел в двоичной системе счисления. В его основе лежит таблица сложения одноразрядных двоичных чисел:

+	0	1
0	0	1
1	1	10

.

При сложении двух единиц происходит переполнение разряда и производится перенос в старший разряд. Переполнение разряда наступает тогда, когда значение числа в нем становится равным или больше основания.

Сложение многоразрядных двоичных чисел происходит в соответствии с вышеприведенной таблицей сложения с учетом возможных переносов из младших разрядов в старшие. В качестве примера сложим в столбик двоичные числа:

$$\begin{array}{r} \underline{1\,1}\,\underline{1} \\ 1\,0\,1\,1 \\ +\quad 1\,0\,1 \\ \hline 1\,0\,0\,0\,0 \end{array}; \qquad \begin{array}{r} \underline{1}\,\underline{1}\,\underline{1}\,\underline{1} \\ 1\,0\,1\,0\,1\,1 \\ +\quad 1\,0\,1\,1\,0 \\ \hline 1\,0\,0\,0\,0\,0\,1 \end{array}.$$

Проверка: $1011_2 + 101_2 = 11_{10} + 5_{10} = 16_{10} = 10000_2$;

$101011_2 + 10110_2 = 43_{10} + 22_{10} = 65_{10} = 1000\ 001_2$.

Вычитание. Рассмотрим вычитание двоичных чисел. В его основе лежит таблица вычитания одноразрядных двоичных чисел. При вычитании из меньшего числа (0) большего (1) производится заем из старшего разряда. В таблице заем обозначен 1 с чертой:

$$\begin{aligned} 0 - 0 &= \underline{1}0; \\ 0 - 1 &= 11; \\ 1 - 0 &= 1; \\ 1 - 1 &= 0. \end{aligned}$$

Вычитание многоразрядных двоичных чисел происходит в соответствии с вышеприведенной таблицей вычитания с учетом возможных заемов из старших разрядов. В качестве примера произведем вычитание двоичных чисел:

$$\begin{array}{r} {\scriptstyle \cdot\ \cdot} \\ 1\,1\,0 \\ -\quad 1\,1 \\ \hline 1\,1 \end{array}; \qquad \begin{array}{r} {\scriptstyle \cdot\qquad \cdot} \\ 1\,0\,1\,0\,1\,1 \\ -\quad 1\,0\,1\,1\,0 \\ \hline 1\,0\,1\,0\,1 \end{array}.$$

Проверка: $110_2 - 11_2 = 6_{10} - 3_{10} = 3_{10} = 11_2$

$101011_2 - 10110_2 = 43_{10} - 22_{10} = 21_{10} = 10101_2$

Умножение. В основе умножения лежит таблица умножения одноразрядных двоичных чисел:

x	0	1
0	0	0
1	0	1

.

Умножение многоразрядных двоичных чисел происходит в соответствии с вышеприведенной таблицей умножения по обычной схеме, применяемой в десятичной системе счисления с последовательным умножением множимого на цифры множителя. В качестве примера рассмотрим умножение двоичных чисел:

```
    1 1 0          1 0 1 1
x     1 1      x     1 0 1
-------------  -------------
    1 1 0          1 0 1 1
  1 1 0          0 0 0 0
-------------    1 0 1 1
  1 0 0 1 0    -------------
;                1 1 0 1 1 1   .
```

Проверка: $110_2 \cdot 11_2 = 6_{10} \cdot 3_{10} = 18_{10} = 10\ 010_2$

$1011_2 \cdot 101_2 = 11_{10} \cdot 5_{10} = 55_{10} = 110\ 111_2$

По сути, умножение сводится к суммированию сдвинутых копий первого сомножителя с самим собой, т.е. в двоичной системе умножение реализуется с помощью суммирования и сдвигов.

Деление. Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в десятичной системе счисления. В качестве примера произведем деление двоичного числа 110_2 на 11_2:

```
1001 | 11
 11  | 11
---
  11
  11
---
   0
```

Проверка: $1001_2 / 11_2 = 9_{10} / 3_{10} = 3_{10} = 11_2$.

Аналогично можно выполнять арифметические действия в восьмеричной и шестнадцатеричной системах счисления. Необходимо только помнить, что перенос в следующий разряд при сложении и заем из старшего разряда при вычитании производится с учетом значения основания СС.

+	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

Рисунок 2.1. — Сложение в восьмеричной СС

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

Рисунок 2.2. — Умножение в восьмеричной СС

Пример: Выполним операции сложения, вычитания и умножения в восьмеричной СС.

$$\begin{array}{r} \underline{1}\ . \\ (3\ 7)_8 \\ +\ \underline{(2\ 5)_8} \\ (6\ 4)_8 \end{array}\ ; \qquad \begin{array}{r} \underline{1}\,\underline{1}\ . \\ (7\ 5\ 6)_8 \\ +\ \underline{(5\ 7)_8} \\ (1\ 0\ 3\ 5)_8 \end{array}\ ; \qquad \begin{array}{r} (7\ 5\ 2)_8 \\ -\ \underline{(6\ 5)_8} \\ (6\ 6\ 5)_8 \end{array}\ ; \qquad \begin{array}{r} (7\ 7\ 5\ )_8 \\ \text{x}\ \underline{(5\ 6\ )_8} \\ (5\ 7\ 5\ 6)_8 \\ \underline{(4\ 7\ 6\ 1)_8} \\ (5\ 5\ 5\ 6\ 6)_8 \end{array}.$$

Для проведения арифметических операций над числами, представленными в различных системах счисления, необходимо предварительно перевести их в одну систему.

2.3.4. Аудиторные задачи и примеры их решения

Задача 2.1. Перевести числа из десятичной СС в двоичную СС, в восьмеричную СС и шестнадцатеричную СС:

а) 75, 25; б) 13,31; в) 32,62.

Ответы: а) $(75{,}25)_{10} \approx (4\text{B},4)_{16} \approx (1\ 001\ 011{,}01)_2 \approx (113{,}2)_8$;

б) $(13{,}32)_{10} \approx (\text{D},51)_{16} \approx (1\ 101,\ 010\ 100\ 011)_2 \approx (15{,}243)_8$;

в) $(32{,}62)_{10} \approx (20{,}9\text{E})_{16} \approx (100\ 000{,}100\ 111\ 10)_2 \approx (40{,}47)_8$.

Задача 2.2. Перевести числа из двоичной СС в восьмеричную СС и шестнадцатеричную СС, используя правила триад и тетрад:

а) 1101,11; б) 10110,01.

Ответы: а) $(1\ 101{,}11)_2 \approx (15{,}6)_8 \approx (\text{D},12)_{16}$;

б) $(10\ 110{,}01)_2 \approx (26{,}2)_8 \approx (16{,}4)_{16}$.

Задача 2.3. Перевести числа из шестнадцатеричной СС в двоичную СС и восьмеричную СС, используя правила триад и тетрад:

а) 5A6; б) 8F,21.

Ответы: а) $(5\text{A}6)_{16} \approx (101\ 1010\ 0110)_2 \approx (2646)_8$;

б) $(8\text{F},21)_{16} \approx (10\ 001\ 111{,}001\ 000\ 01)_2 \approx (217{,}102)_8$.

Задача 2.4. Выполнить сложение, вычитание, умножение двоичных чисел $(101\ 010)_2$ и $(110)_2$.

Ответы: а) $(101\ 010)_2 + (110)_2 = (110\ 000)_2$;

б) $(101\ 010)_2 - (110)_2 = (100\ 100)_2$;

в) $(101\ 010)_2 \text{ x } (110)_2 = (11\ 111\ 100)_2$

Задача 2.5. Выполнить сложение и умножение восьмеричных чисел $(32)_8$ и $(12)_8$.

Ответы: а) $(32)_8 + (12)_8 = (44)_8$; б) $(32)_8$ x $(12)_8 = (404)_8$.

Задача 2.6. Выполнить сложение чисел $(32)_8$ и $(32)_{16}$.

Ответ: $(32)_8 + (32)_{16} = (4C)_{16} = (114)_8$.

2.3.5. Задания для самостоятельной аудиторной работы

Задача 2.7. Перевести число из двоичной СС в восьмеричную СС и шестнадцатеричную СС, используя правила триад и тетрад $(100\ 101\ 111)_2$.

Ответ: $(100\ 101\ 111)_2 = (457)_8 = (12F)_{16}$.

Задача 2.8. Перевести число из шестнадцатеричной СС в двоичную СС и восьмеричную СС, используя правила триад и тетрад $(B36)_{16}$.

Ответ: $(B36)_{16} = (101\ 100\ 110\ 110)_2 = (5466)_8$.

Задача 2.9. Перевести числа из десятичной СС в двоичную СС, восьмеричную СС и шестнадцатеричную СС, используя алгоритм перевода

а) $(42{,}45)_{10}$; б) $(329{,}132)_{10}$.

Ответ: а) $(42{,}45)_{10} = (101\ 010{,}011\ 100\ 11)_2 = (52{,}346)_8 = (2A{,}73)_{16}$;

б) $(329{,}132)_{10} = (101\ 001\ 001{,}001\ 000\ 011)_2 = (511{,}103)_8 = (149{,}21)_{16}$.

Задача 2.10. Выполнить сложение, вычитание, умножение двоичных чисел $(10\ 100)_2$ и $(101)_2$.

Ответ: $+ (11\ 001)_2$; $- (1111)_2$; $x(1\ 100\ 100)_2$.

Задача 2.11. Выполнить сложение, вычитание и умножение восьмеричных чисел $(46)_8$ и $(15)_8$.

Ответ: $+ (63)_8$; $- (31)_8$; $x(756)_8$.

Задача 2.12. Сложить числа $17_8 + 17_{16} = (\ )_8 = (\ )_{16}$.

Ответ: $17_8 + 17_{16} = (46)_8 = (26)_{16}$.

2.3.6. Задания для домашней работы: расчетно-графическое задание 1

(таблица с вариантами заданий приведена в приложении А)

РГЗ № 1

Вариант 1

1. Перевести числа из двоичной СС в десятичную СС:

а) $(101\ 010)_2 = (\ )_{10}$; б) $(11010{,}01101)_2 = (\ )_{10}$.

2. Перевести числа из восьмеричной СС в десятичную СС:

а) $(246)_8 = (\ )_{10}$; б) $(643{,}52)_8 = (\ )_{10}$.

3. Перевести числа из шестнадцатеричной СС в десятичную СС:

а) $(B05)_{16} = (\ )_{10}$; б) $(5F{,}5)_{16} = (\ )_{10}$.

4. Перевести числа из десятичной СС в двоичную СС, используя алгоритм перевода:

а) $(25)_{10} = (\ )_2$; б) $(321)_{10} = (\ )_2$.

5. Перевести числа из десятичной СС в восьмеричную СС, используя алгоритм перевода:

а) $(25)_{10} = (\ )_8$; б) $(321)_{10} = (\ )_8$.

6. Перевести числа из десятичной СС в шестнадцатеричную СС, используя алгоритм перевода:

а) $(25)_{10} = (\ )_{16}$; б) $(321)_{10} = (\ )_{16}$.

7. Используя правила триад и тетрад, перевести числа из шестнадцатеричной СС в двоичную СС и в восьмеричную СС:

а) $(75)_{16} = (\ )_2 = (\ )_8$; б) $(1A05)_{16} = (\ )_2 = (\ )_8$.

8. Используя правила триад и тетрад, перевести числа из восьмеричной СС в двоичную СС и в шестнадцатеричную СС:

а) $(75)_8 = (\ )_2 = (\ )_{16}$; б) $(105)_8 = (\ )_2 = (\ )_{16}$.

9. Перевести числа из десятичной СС в двоичную СС, используя алгоритм перевода:

а) $(25{,}125)_{10} = (\ )_2$; б) $(123{,}123)_{10} = (\ )_2$.

10. Перевести числа из десятичной СС в восьмеричную СС, используя алгоритм перевода:

а) $(25{,}125)_{10} = (\ )_8$; б) $(123{,}123)_{10} = (\ )_8$.

11. Перевести числа из десятичной СС в шестнадцатеричную СС, используя алгоритм перевода:

а) $(25{,}125)_{10} = (\ )_{16}$; б) $(123{,}123)_{10} = (\ )_{16}$.

12. Выполнить сложение, вычитание, умножение двоичных чисел:

а) $(1001)_2$ и $(101)_2$; б) $(110\ 010)_2$ и $(1001)_2$.

13. Выполнить сложение, вычитание и умножение восьмеричных чисел:

а) $(54)_8$ и $(13)_8$; б) $(451)_8$ и $(\ 131\ )_8$.

14. Выполнить сложение $16_8 + 16_{16} = (\ )_8 = (\ )_{16}$.

15. Выполнить вычитание $16_{16} - 16_8 = (\ )_8 = (\ )_{16}$.

3. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3

Дополнительный код числа. Представление чисел в памяти ЭВМ (числа с фиксированной запятой)

3.1. Цели и задачи занятия

Выработать навыки представления и интерпретации положительных и отрицательных целых чисел в формате с фиксированной запятой.

3.2. План занятия

3.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
3.2.2. Цель и задачи занятия (5 мин.).
3.2.1. Обсуждение теоретических вопросов (20 мин.):
- дополнительный код числа;
- представление целых положительных чисел в формате с фиксированной точкой;
- представление целых отрицательных чисел в формате с фиксированной точкой.

3.2.2. Решение аудиторных задач (20 мин.).
3.2.3. Выполнение самостоятельных аудиторной работы (15 мин.).
3.2.4. Выдача домашнего задания (5 мин.).
3.2.5. Подведение итогов занятия (5 мин.).

3.3. Содержание занятия

3.3.1. Краткие сведения о формате представления целых чисел в памяти ЭВМ. Прямой, обратный и дополнительный коды

Дополнением n-разрядного числа называется число, которое необходимо добавить к исходному числу, чтобы получить нули во всех его **n** разрядах и единицу в **(n + 1)** разряде.

Например, 362 + *дополнение* = 1000. Отсюда *дополнение* = 1000 – 362 = = 638. Число 638 является *дополнением* до 1000 числа 362.

Дополнение позволяет заменить операцию вычитания сложением.

Пример: x – y = x + (1000 – y) – 1000. При этом последнее вычитание сводится к отбрасыванию цифры переноса, если результат получен в прямом коде. Пусть требуется найти разность 987 – 362. Тогда

```
  987                    987
– 362        или     +  638
-----                  -----
  625                  1| 625
                        перенос
```

Если требуется вычислить 362 – 987, то

```
  362                  362
– 987      или     + 13      дополнение до 1000 числа 987
– 625                375      -> результат получен в доп. коде
                   – 1000
                     – 625    -> прямой код
```

Результат 362 – 987 = 362 + 13 = 375 получен в дополнительном коде. Чтобы перевести его в прямой код необходимо вычесть 1000, т.е. 375 – 1000 = = 625.

В компьютере целые отрицательные числа хранятся в дополнительном коде.

Рассмотрим пример для двоичной СС, вычислим $(1111)_2 – (1010)_2$:

```
  1111
– 1010
  101   .
```

Выполним эту же операцию с использованием дополнительного кода:

```
  10000              1010   – прямой код
– 1010      или      0101   – обратный код
  0110             +    1
                     0110   – дополнительный код.
```

В двоичной системе справедлива формула:

Дополнительный код = Обратный код + 1.

Тогда, складывая число $(1111)_2$ с дополнительным кодом числа $(1010)_2$, получим

```
    1111
+   0110
  1 0101
перенос.
```

Если при сложении дополнительного кода с некоторым числом возникает перенос в старший разряд, то этот перенос отбрасывается, и результат получается в ***прямом коде***. Если переноса не возникает, то это означает, что полученный результат представлен в дополнительном коде.

Целые числа в памяти ЭВМ представляются в формате с фиксированной запятой (рис. 3.1.).

Рисунок 3.1. — Представление целого числа в памяти компьютера.

Целые числа хранятся в памяти компьютера в двоичной системе счисления, но при этом оговаривается разрядность. Обычно для хранения числа используется 2 или 4 байта. В первом бите хранится знак числа. Если в первом бите числа ноль, то число считают положительным. В противном случае число считается отрицательным и представлено в виде своего дополнения.

Пример 3.1. Рассмотрим как представляется число $(39)_{10}$ в памяти компьютера: $(39)_{10}=(27)_{16}=(10\ 0111)_2$.

Число размещают, начиная с крайней правой позиции формата.

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**1**
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*

0 0 2 7

Шестнадцатеричное представление шестнадцати разрядного формата: 0027.

Пример 3.2. Рассмотрим представление числа $(329)_{10}$ в памяти компьютера: $(\ 329\ )_{10}=(\ 149\ )_{16}=(\ 1\ 0100\ 1001\ )_2$.

329 | 16
320 | 20 | 16
9 | 16 | 1
4

0				**1**				**4**				**9**				**– 16-е представление** $(\ 329\ )_{10}$
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**– 2-е представление**$(\ 329\ )_{10}$
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*	– номера бит

Результат: 0149.

Отрицательные числа представляются в дополнительном коде, при этом дополнение выполняют до количества разрядов формата.

Пример 3.3. Рассмотрим представление числа $(-39)_{10}$ в памяти компьютера: $(-\ 39)_{10} = (-\ 0010\ 0111)_2$.

```
  0000 0000 0010 0111   – прямой код;
+ 1111 1111 1101 1000   – обратный код;
                     1
-------------------------
  1111 1111 1101 1001   – дополнительный код.
    F    F    D    9    – результат
```

Пример 3.4. Рассмотрим представление числа $(-329)_{10}$ в памяти компьютера: $(-\ 329\ )_{10} = (-\ 149\ )_{16}= (-\ 0000\ 0001\ 0100\ 1001\ )_2$;

```
  0000 0001 0100 1001   – прямой код;
+ 1111 1110 1011 0110   – обратный код;
                     1
-------------------------
  1111 1110 1011 0111   – дополнительный код.
    F    E    B    7    – результат.
```

3.3.2. Аудиторные задачи и примеры их решения

Задача 3.1. Представить десятичные числа 839 и – 839 в шестнадцатиразрядном формате с фиксированной запятой, результат записать шестнадцатеричными цифрами.

Переведем 839 в шестнадцатеричную СС:

```
839 | 16
832   52 | 16
  7   48   3
      4
```

Получим $(839)_{10} = (347)_{16}$, используя правило тетрад, переведем число в двоичную СС: $(347)_{16} = (0000\ \ 0011\ \ 0100\ \ 0111)_2$. Результат запишем шестнадцатеричными цифрами: **0347**.

Для представления числа $(– 839)_{10}$ в памяти ЭВМ найдем его дополнительный код:

```
    0000 0011 0100 0111   – прямой код;
+   1111 1100 1011 1000   – обратный код;
                      1
---------------------------
    1111 1100 1011 1001   – дополнительный код;
      F    C    B    9    – результат.
```

3.3.3. Задания для самостоятельной аудиторной работы

Задача 3.2. Представить следующие десятичные числа в шестнадцатиразрядном формате с фиксированной запятой, результат записать шестнадцатеричными цифрами: а) 1335; в) 1840;
б) –1739; г) –1941.

Ответы: а) 0537; б) F935; в) 0730; г) F86B.

3.3.4. Задания для домашней работы

Вариант 3.1. Представить следующие десятичные числа в шестнадцатиразрядном формате с фиксированной запятой, результат записать шестнадцатеричными цифрами: а) 2225; б) –2225.

Ответ: а) 08B1; б) F74F.

Вариант 3.2. Представить следующие десятичные числа в шестнадцатиразрядном формате с фиксированной запятой, результат записать шестнадцатеричными цифрами: а) 3335; б) –3335.

Ответ: а) 0D07; б) F2F9.

4. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4

Представление чисел в памяти ЭВМ. Числа с плавающей запятой. Представление символов в памяти ЭВМ.

4.1. Цели и задачи занятия

Выработать навыки представления и интерпретации положительных и отрицательных чисел в форматах с плавающей запятой в памяти ЭВМ. Выработать навыки представления и интерпретации символов в памяти ЭВМ.

4.2. План занятия

4.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
4.2.2. Цель и задачи занятия (5 мин.).
4.2.1. Обсуждение теоретических вопросов (20 мин.):
- экспоненциальная форма представления чисел, нормализация чисел;
- формат IBM-360 для представления чисел с плавающей запятой;
- формат IEEE-754 для представления чисел с плавающей запятой;
- представление символов в памяти ЭВМ.

4.2.2. Решение аудиторных задач (20 мин.).
3.2.3. Выполнение самостоятельных аудиторной работы (15 мин.).
4.2.4. Выдача домашнего задания (5 мин.).
4.2.5. Подведение итогов занятия (5 мин.).

4.3. Содержание занятия

4.3.1. Краткие сведения о представлении чисел в формате с плавающей запятой

Экспоненциальная форма представления чисел

Это представление базируется на том, что любое число X можно представить в виде произведения:

$$X = \pm q \cdot p^n, \tag{4.1.}$$

где n — порядок, p — основание системы счисления, q — правильная дробь, которая называется *мантиссой*, $0 \le q < 1$. Существует единственная мантисса, удовлетворяющая условию

$$\frac{1}{P} \le q < 1. \tag{4.2.}$$

Такая мантисса называется *нормализованной*.

Пример 4.1. Представим число 65,7 в нормализованной экспоненциальной форме: $65{,}7 = 0{,}657 \cdot 10^2$, $n=2$, $q=657$.

Мантисса всегда хранится в прямом коде независимо от того, положительное ли число, или отрицательное.

Рассмотрим 2 основных формата представления вещественных чисел в памяти ЭВМ: IBM-360 и IEEE 754.

Формат IBM-360. Содержимое 32-битного формата представлено на рис. 4.1.

Характеристика *C* предназначена для хранения порядка числа и представляет собой смещенное значение порядка.

$C = (40)_{16} + (n)_{16} = (64)_{10} + (n)_{10} = (1000000)_2 + (n)_2$.

Характеристика (экспонента) занимает 7 бит.

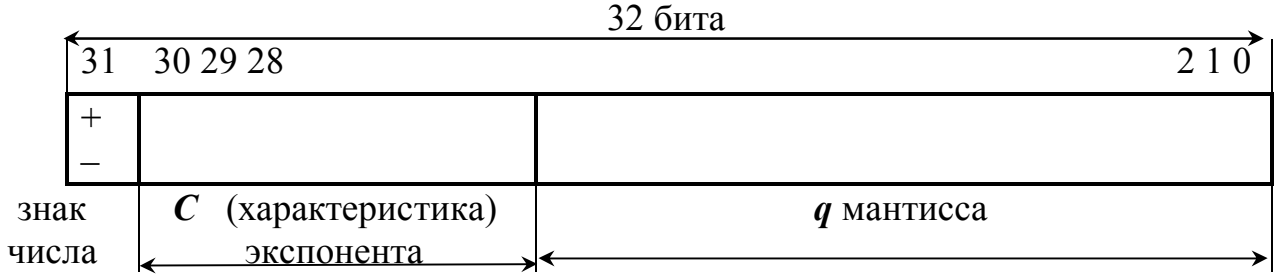


Рисунок 4.1. — Формат IBM-360

Пример 4.2. Требуется представить число 15,5 в формате IBM-360.

Переведем число из десятичной СС в шестнадцатеричную СС: $(15,5)_{10} = (F,8)_{16}$. Представим полученное число в нормализованной экспоненциальной форме: $F,8 = 0,F8 \cdot 16^1$; порядок равен 1; F8 — значение и цифры мантиссы. На следующем шаге вычислим значение характеристики: C = 40 + 1 = 41.

В двоичном коде формат запишется в виде, изображенном на рис. 4.2.

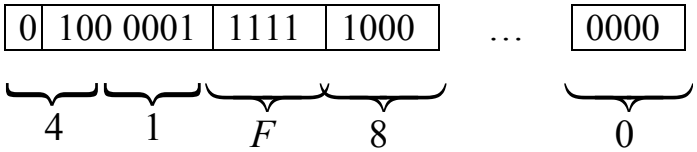


Рисунок 4.2. — Представление числа 15,5 в формате IBM-360

Для сокращения записи часто формат представляют шестнадцатеричными числами, т.е. $(15,5)_{10} \rightarrow$ 41F8 0000.

Пример 4.3. Требуется представить число -15,5 в формате IBM-360.

Выполним аналогичные действия: $(-15,5)_{10} = (-F,8)_{16}$. В двоичном коде формат запишется в виде, изображенном на рис. 4.3.

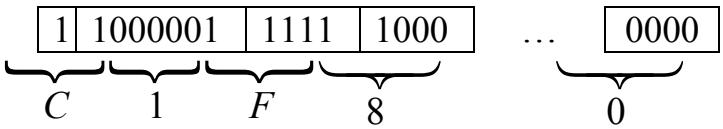


Рисунок 4.3. — Представление числа –15,5 в формате IBM-360

Запишем число шестнадцатеричными числами $(-15,5)_{10} \rightarrow C1F80000$.

В знаковом бите 0 заменяется на 1, т.к. число отрицательное. Таким образом, отрицательные числа, записанные в шестнадцатеричной форме, начинаются с C, D, E или F.

Формат IEEE 754. Этот формат также использует для представления числа формулу (4.1.), где $1 \le q < 2$. При этом старший бит мантиссы всегда равен 1 и поэтому не хранится, его называют скрытым битом. Экспонента (характеристика) занимает 8 бит и вычисляется по формуле: $C = (127)_{10} + (n)_{10} = (0111\ 1111)_2 + (n)_2 = (7F)_{16} + (n)_{16}$.

Пример 4.4. Требуется представить число 53,203 в 32-х битовом формате IEEE 754.

Сначала переведем число в двоичную СС: $53{,}203_{10} = 110101{,}001101_2$.

Полученное число представим в следующем виде:

$110101{,}001101_2 = \underline{1}, 10101001101 \cdot 2^5$, где целая часть — это скрытый бит, а дробная часть — это мантисса. Тогда n=5, а характеристика

$C = 127+5 = 0111\ 1111+101 = 1000\ 0100$.

Число в памяти компьютера будет храниться в виде, изображенном на рис. 4.4.:

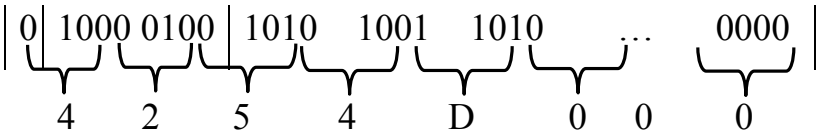


Рисунок 4.4. — Представление числа 53,203 в формате IEEE 754

Запись числа шестнадцатеричными цифрами: $(53{,}203)_{10}$ → 4254D000.

Пример 4.5. Требуется представить число –53,203 в формате IEEE 754.

Выполним аналогичные действия. Результат представим в виде, изображенном на рис. 4.5.

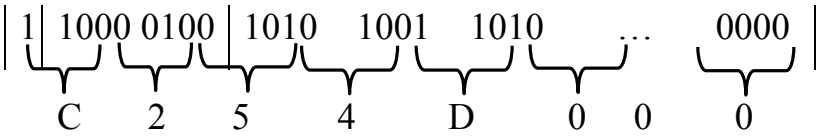


Рисунок 4.5. — Представление числа –53,203 в формате IEEE 754

Запись числа шестнадцатеричными цифрами: $(-53{,}203)_{10}$ → C254D000.

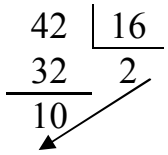
Сравнение форматов IBM-360 и IEEE 754 приведено в табл. 4.1.

Таблица 4.1. – Сравнение форматов

параметр	Формат IBM-360	Формат IEEE 754
Размер ***C***	7 бит	8 бит
Размер ***m***	24	23
Диапазон ***q***	1/p <= q < 1	1 <= q < 2
Запятая	слева от мантиссы	справа от старшего бита мантиссы
Пример:	F,8= 0,F8 · $(16)^1$ q=F8; P=16; n=1	101,101= 1,01101· $(2)^2$ q=01101; P=2; n=2
Формула ***C***	C = $(40)_{16}$ + $(n)_{16}$ = $(64)_{10}$ + $(n)_{10}$= = $(100\ 0000)_2$ + $(n)_2$.	C = $(127)_{10}$ + $(n)_{10}$ = $(01111111)_2$ + + $(n)_2$ = $(7A)_{16}$ + $(n)_{16}$

Пример 4.6. Представить числа 42,25 и –42,25 в формате IBM-360.

Переведем в шестнадцатеричную СС целую часть:



Переведём дробную часть: 0,25 ·16 = 4,0.

Запишем число в нормализованной экспоненциальной форме:

$(42,25)_{10} = (2A,4)_{16} = 0,2A4 \cdot 16^2$.

Тогда $n = 2$ и характеристика $C = 40 + 1 = (41)_{16} = (100\ 0001)_2$.

Запишем результат шестнадцатеричными цифрами $(42,25)_{10} \rightarrow$ 412A4000 (рис. 4.6.).

0	100	0001	0010	1010	0100	0000	0000	0000
4		1	2	A	4	0	0	0

Рисунок 4.6. — Представление числа 42,25 в формате IBM-360

Выполним аналогичные действия для представления числа –42,25. Получим: $(–42,25)_{10} \rightarrow$ C12A4000 (рис. 4.7.)

1	100	0001	0010	1010	0100	0000	0000	0000
C		1	2	A	4	0	0	0

Рисунок 4.7. — Представление числа –42,25 в формате IBM-360

Пример 4.7. Представить числа 42,25 и –42,25 в формате IEEE 754.

Переведем число в двоичную СС и представим в виде:

$$(42,25)_{10}=(0010\ 1010\ ,\ 0100)_2 = \underline{1},\ 010100100 \cdot 2^5.$$

Тогда $n = 5$, характеристика $C = 127 + 5 = 01111111 + 101 = 1000\ 0100$.

Запишем результат шестнадцатеричными цифрами:

$(+42,25)_{10} \rightarrow$ 12290000 (рис. 4.8.).

0	1000	0100	0101	0010	0000	0000	0000	000
4	2	2	9	0	0	0	0	

Рисунок 4.8. — Представление числа 42,25 в формате IEEE 754

Выполним аналогичные действия для представления числа –42,25. Получим: $(–42,25)_{10} \rightarrow$ C2290000 (рис. 4.9.)

1	1000	0100	0101	0010	0000	0000	0000	000
C	2	2	9	0	0	0	0	

Рисунок 4.9. — Представление числа -42,25 в формате IEEE 754

4.3.2. Краткие сведения о формате представления символов в памяти ЭВМ

Для хранения одного символа отводится один или два байта. При этом используются различные кодовые таблицы. Рассмотрим кодовую таблицу *ASCII* (*American Standard Code for Information Inter Exchange*). Эта таблица 8-битовая, 8 бит позволяют закодировать 256 символов. Коды от 0 до 127 — управляющие сигналы и символы основного стандарта. Коды от 127 до 255 представляют символы национальных алфавитов.

Пример: "И" — 136; "1" — 49; "3" — 51; "-" — 95; "к" — 170.

Запись “И-13к” в памяти будет представлена пяти байтах (рис. 4.10.)

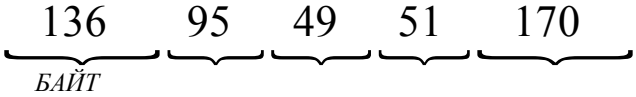


Рисунок 4.10. — Запись “И-13к” в памяти ЭВМ

В настоящее время широко применяется стандарт Юнико́д или Уникóд (англ. Unicode). Применение этого стандарта позволяет закодировать очень большое число символов из разных письменностей: в документах Unicode могут соседствовать китайские иероглифы, математические символы, буквы греческого алфавита, латиницы и кириллицы, при этом становится ненужным переключение кодовых страниц.

Стандарт состоит из двух основных разделов: универсального набор символов UCS (universal character set) и семейства кодировок UTF (Unicode transformation format).

Коды в стандарте Юникод разделены на несколько областей. Область с кодами от U+0000 до U+007F содержит символы набора ASCII с соответствующими кодами. Далее расположены области знаков различных письменностей, знаки пунктуации и технические символы. Часть кодов зарезервирована для использования в будущем. Под символы кириллицы выделены области с кодами от U+0400 до U+052F, от U+2DE0 до U+2DFF, от U+A640 до U+A69F

4.3.3. Аудиторные задачи и примеры их решения

Задача 4.1. Представить следующие десятичные числа в тридцатидвухразрядном (IBM-360) формате представления чисел с плавающей точкой в памяти ЭВМ. Записать окончательный результат шестнадцатеричными цифрами:

а) 2225,25; б) –2225,25;

Ответы: а) 2225,25 → 43 8B1400; б) –2225,25 → C3 8B1400.

Задача 4.2. Представить следующие десятичные числа в тридцатидвухразрядном (IEEE 754) формате представления чисел с плавающей точкой в памяти ЭВМ:

а) 2225,25; б) –2225,25.

Ответы: а) 2225,25 → 45 0B1400; б) –2225,25 → C5 0B1400.

4.3.4. Задания для самостоятельной аудиторной работы

Задача 4.3. Представить следующие десятичные числа в тридцатидвухразрядном (IBM-360) формате представления чисел с плавающей точкой в памяти ЭВМ. Записать окончательный результат шестнадцатеричными цифрами:

а) 3335,25; б) –3335,25.

Ответы: а) 3335,25 → 43 D07400; б) –3335,25 → C3 D07400.

Задача 4.4. Представить следующие десятичные числа в тридцатидвухразрядном (IEEE 754) формате представления чисел с плавающей точкой в памяти ЭВМ:

а) 3335,25; б) –3335,25.

Ответы: а) 3335,25 → 45507400; б) –3335,25 → C5507400.

Задача 4.5. Представить числа в двух форматах IBM 360 и IEEE754:

а) 1335,8; б) –1335,8;

в) 1841,7; г) –1841,7.

Ответы: а) 1335,8 → 43537CC0→4426F980;

б) –1335,8 → C3537CC0→C426F980;

в) 1841,7 → 43731B30→44666600;
г) –1841,7 → C3731B30→C4666600.

4.3.5. Задания для домашней работы: расчетно-графическое задание 2

(таблица с вариантами заданий приведена в приложении А.3.)

РГЗ № 2

ВАРИАНТ № 1

1. Представить следующие десятичные числа в шестнадцатиразрядном формате представления целых чисел в памяти ЭВМ. Записать окончательный результат шестнадцатеричными цифрами:

а) 1234; б) – 1234.

2. Представить следующие десятичные числа в тридцатидвухразрядном (IBM-360) формате представления чисел с плавающей точкой в памяти ЭВМ. Записать окончательный результат шестнадцатеричными цифрами: а) 1234,25; б) – 1234,25.

3. Представить следующие десятичные числа в тридцатидвухразрядном (IEEE 754) формате представления чисел с плавающей точкой в памяти ЭВМ. Записать окончательный результат шестнадцатеричными цифрами:

а) 1234,25; б) – 1234,25.

5. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5

Программирование алгоритмов линейной структуры. Простейший ввод вывод. Программирование алгоритмов разветвляющейся структуры.

5.1. Цели и задачи занятия

Выработать навыки разработки алгоритмов линейной и разветвляющейся структур, а также навыки организации простейшего ввода-вывода.

5.2. План занятия

5.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
5.2.2. Цель и задачи занятия (5 мин.).
5.2.3. Обсуждение теоретических вопросов (20 мин.):
- структура Паскаль программы;
- простые типы данных, разделы описания констант и переменных;
- операции, выражения и преобразования типов значений в выражениях;
-оператор присваивания, условный оператор, оператор выбора;
- процедуры ввода и вывода данных.

5.2.4. Решение аудиторных задач (20 мин.).
5.2.5. Выполнение самостоятельных аудиторной работы (15 мин.).
5.2.6. Выдача домашнего задания (5 мин.).
5.2.7. Подведение итогов занятия (5 мин.).

5.3. Содержание занятия

5.3.1. Краткие сведения об основных разделах Паскаль-программы, операциях и выражениях

Паскаль программа содержит следующие разделы:

```
Program ... ; {Заголовок программы }
Label ... ; { Раздел меток }
Const ... ; { Раздел констант }
Type ... ; { Раздел типов }
Var ... ; { Раздел переменных }
Procedure ... ; { Раздел процедур }
Function ... ; { Раздел функций }
Begin { Раздел операторов}
 ...;
 { Операторы }
 ...;
 End.
```

Обязательным является только раздел операторов, который начинается словом **begin**, а заканчивается словом **end** с точкой. Операторы в Паскале разделяются точкой запятой. Заголовок программы состоит из зарезервированного слова **program** и имени программы, за котором следует точка с запятой.

Раздел констант начинается зарезервированным словом **const**, за которым следует описание констант. Описание состоит из имени константы, знака = и значения константы. Описание констант отделяются друг от друга точкой с запятой.

Пример 5.1.

```
const
Name = 'Петя'; {Строковая константа}
Code = 124; {Числовая константа }.
```

Раздел переменных

Каждая встречающаяся в программе переменная должна быть описана. Описание обязательно должно предшествовать использованию переменной. Раздел описания переменных начинается зарезервированным словом *var* (*variable* — переменная), затем через запятую перечисляются имена переменных и после двоеточия следуют их тип и точка с запятой. Указанная конструкция может повторяться:

var {имя {, имя} : тип;}

Пример 5.2. var A, B, Proizved : integer;

На рисунке 5.1. изображена классификация простых типов данных языка Паскаль.

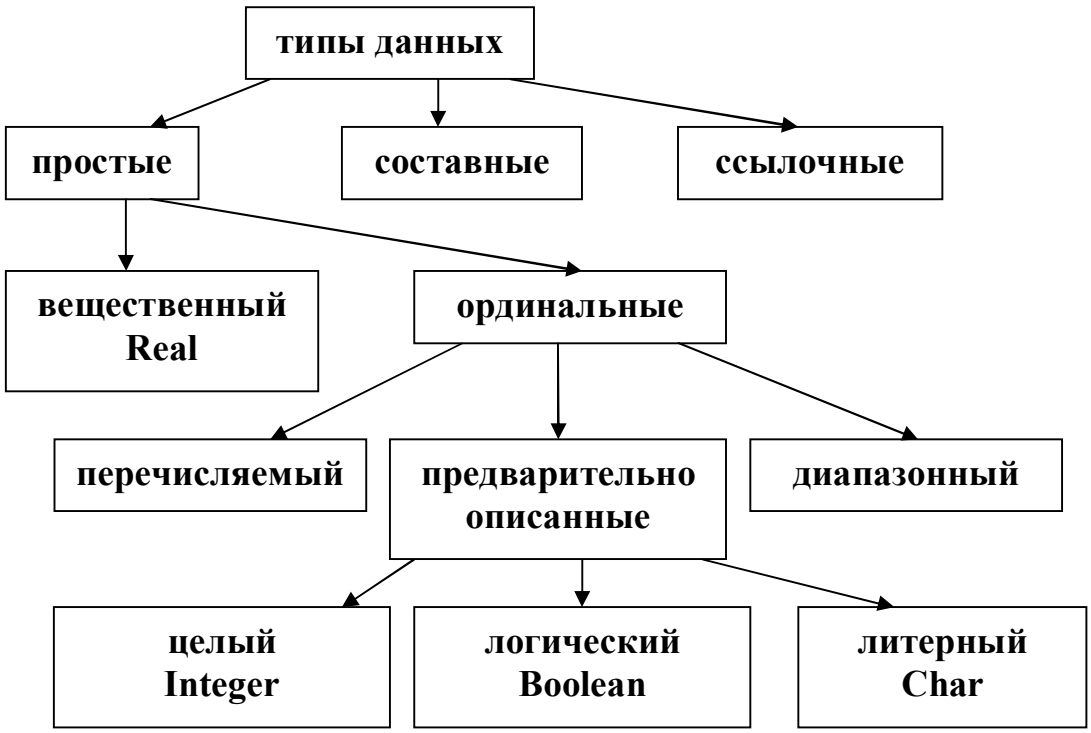


Рисунок 5.1. — Простые типы данных

Тип **real** представляет вещественные числа со знаком в диапазоне от $-1{,}7\cdot10^{38}$ до $+2{,}9\cdot10^{-39}$, занимает 6 байтов.

Тип **integer** представляет целые числа со знаком в диапазоне от –32768 до +32767, занимает 2 байта.

Тип **boolean** занимает 1 байт, может принимать только два значения: **true**, **false**.

Тип **char** занимает 1 байт, представляет все символы кода **ASCII**.

Перечислимый тип представляет собой упорядоченную последовательность констант.

Пример 5.4. Перечислимый тип данных:

```
type  Year = ( January , February , March , April , May , June , July , August ,
        September , October , November , December );
Var y:Year;
```

Диапазонный тип — это диапазон значений любого из ординальных типов, упомянутых выше. Приведем примеры определения диапазонных типов.

Пример 5.5. Диапазонный тип данных:

```
type
Days = ( monday, tuesday, wednesday, thursday, friday, saturday, sunday);
WorkDays = monday .. friday ;
WeekEnd = Saturday .. Sunday ;
```

В этом примере типы **WorkDays и WeekEnd** — диапазонные типы.

Пример 5.6. Диапазонный тип данных:

```
type  Year = 1990..2000; Day=1..31; Ch='A'..'Z';
```

Для данных целого типа определены следующие арифметические операции, результат выполнения которых также будет иметь целый тип:

- сложение (+)
- изменение знака (унарный минус –);
- вычитание (–);
- умножение (*);
- целочисленное деление (**div**);
- остаток от деления (**mod**).

Результатом выполнения операции **div** является целая часть частного, а операции **mod** — остаток целочисленного деления. Над целыми также допустима операция деления (знак /), приводящая к вещественному значению.

Для данных вещественного типа определены такие же операции как и над числами целого типа данных (сложение, изменение знака, вычитание, умножение, деление), но только результат выполнения этих операций будет вещественного типа.

В процессе выполнения программы переменные могут получать новые значения с помощью оператора присваивания:

<оператор присваивания>::=<переменная>:=<выражение>

Сначала вычисляется выражение, записанное справа от знака присваивания (:=), после чего переменная получает вычисленное значение.

При вычислении выражений типы переменных должны соответствовать применяемым операциям. Автоматические преобразования возможны только в случае преобразования целого типа в вещественный.

Порядок вычисления выражений определяется скобками (), правилом *слева-направо*, а также приоритетом операций:

1) операция отрицания **NOT**;

2) мультипликативные операции: ***, /, div, mod, and;***

3) аддитивные операции: **+, -, or;**

4) операции отношения: **<=, <, =, <>, >, >=, in.**

Пример 5.7. Для выражения $\sin^2(x)+\cos(y)\cdot 12/10 > x+y/2-12$ определим порядок вычисления выражения (см. рис. 5.2).

$$\sin^2(x)+\cos(y)\cdot 12/10 > x+y/2-12,$$

порядок вычислений→ 4 1 2 7 5 3 6

Рисунок 5.2. — Порядок вычисления выражения

5.3.2. Условный оператор и оператор выбора

Условный оператор позволяет выбрать одну из 2-х ветвей алгоритма и записывается следующим образом:

If <условие> then <оператор1> [else<оператор2>]

Часть **else** является необязательной. Обратите внимание, что после слов **then** и **else** записывается только один оператор. Если требуется разместить несколько операторов, то применяют составной оператор:

```
Begin
<оператор>;
{<оператор>}
 end;
```

Пример 5.8. Запишите код программы, представленный структурной схемой, изображенной на рис. 5.3.

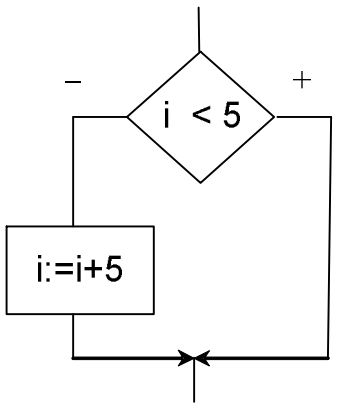


Рисунок 5.3. — Схема разветвляющегося алгоритма

Возможны несколько способов описания:

```
1) If (i<5) then     else i:=i+5;
2) If not(i<5)  then  i:=i+5;
3) If (i>=5)   then i:=i+5;
```

Рекомендуется использовать 2 вариант.

Оператор выбора реализует выбор одной из ***n*** ветвей алгоритма.

Формат записи:

```
Case S of
K1:  OP1;
K2:  OP2;
Ki:  OPi;
. . .
Kn:  OPn;
End;
```

где **Case** (выбор, вариант) и **of** (из) — служебные слова; **S** — выражение-селектор ординального типа; **Ki** — метка варианта — константа, тип которой совпадает с типом выражения-селектора; **OPi** — любой оператор в том числе пустой.

Порядок выполнения оператора следующий: вычисляется выражение-селектор; вычисленное значение последовательно сравнивается с метками вариантов

и управление передается оператору, метка варианта которого совпадает с вычисленным значением селектора; далее выполняется оператор и управление передается за пределы оператора выбора.

Пример 5.9. Написать программу, считывающую число от 0 до 4 и выводящую название этого числа.

Решение:

```
program vetvlenija;
var num :integer;
begin
  writeln('введите число от 0 до 4:');
  readln (num);
    case num of
          0:writeln ('нуль');
          1:writeln ('один');
          2:writeln ('два');
          3:writeln ('три');
          4:writeln ('четыре');
    end;
end.
```

5.3.3. Процедуры ввода-вывода

Ввод-вывод в языке Паскаль выполняется с помощью встроенных процедур **read** и **write**. Формат вызова процедуры **read**:

read (<*переменная*>{,<*переменная*>}).

Переменные должны быть следующих типов: **real, integer, char,** <*строка символов*> **(BP).** Процедура осуществляют ввод данных из стандартного файла **Input**. Процедура **readln** осуществляет ввод значений переменных и обеспечивает переход к началу новой строки.

Формат вызова процедуры **write**:

write (<*выражение*>[:<*ширина*>[:<*точность*>]]
{,<*выражение*>[:<*ширина*>[:<*точность*>]]});

где <*ширина*>— общее количество позиций, отводимых для вывода значения; <*точность*> — количество позиций после запятой. Процедура допускает вывод следующих типов выражений: **real, integer, bollean, char,** <*строка символов*>. Процедура осуществляет вывод в стандартный файл **Output.**

Пример 5.10.

```
a:= 3.8;  write('a=',a:10:7); writeln(' a=',a:10);
b:=5 write('b=',b:3)…
```

Результат:

```
a= 3.8000000 a= 3.800E+00
b=  5.
```

5.3.4. Аудиторные задачи и примеры их решения

Линейные алгоритмы

Задача 5.1. Что будет напечатано программой, если **b=**1.0, а **c=** –2.0?

```
Program корни;
Var b, c, d : real;
Begin read (b, c, d);
```

```
    D:=sqrt (sqr(b) – 4*c);
    Writeln('x1=', (–b+d)/2, '  x2=', (–b–d)/2)
End.
```

Ответ: **x1=1.0 x2=–2.0.**

Задача 5.2. Что будет напечатано программой, если введены числа 1.5 и –0.8?

```
Program less;
Var  x : real; t:boolean;
Begin read (x); t:=x<round(x);
read (x); t:=t  and ( x<trunc(x));
writeln(t)
end.
```

Ответ: **true**.

Задача 5.3. Написать программу, которая печатает **true** или **false** в зависимости от того, имеют три заданных целых числа одинаковую четность или нет.

Решение: функция **ood (x)** возвращает **true**, если **x** нечетное.

```
Var  x1, x2, x3 : integer; t:boolean;
begin  read (x1, x2, x3);
          t:=( odd( x1) and odd( x2) and odd( x3))
          or (not odd( x1))and(not odd( x2))and(not odd( x3))
       write(t)
end.
```

Задача 5.4. Что будет напечатано программой, если введены числа: 1, 2, 3?

```
Var  a, b: integer;
begin
read (a, b, a);
write(a, b, a)
end.
```

Ответ: 3 2 3.

Разветвляющиеся алгоритмы

Задача 5.5. Решить задачу с использованием условного оператора:

а)

$$Y=\begin{cases}\cos^2 x & \text{при } 0<x<2,\\ 1-\sin x^2 & \text{иначе;}\end{cases}$$

б) перераспределить значения переменных **x** и **y** так, чтобы в **x** оказалось большее из этих значений, а в **y** — меньшее;

в) найти максимум из трех значений **d = max (a, b ,c).**

Задача 5.6. Значения переменных **a, b, c** поменять местами, так, чтобы оказалось **a≥ b≥ c.**

Задача 5.7. Какое значение будет иметь переменная **z** после выполнения операторов: **z:= 0***;*

```
if x>0 then if y>0    then z:=1 else z:=2
```

при следующих значениях переменных **x** и **y**:

а) **x=y=1**; б) **x= 1 y=–1**; в) **x=– 1 y=1**.

Задача 5.8. Указать ошибки:

а) **if 1<x<2 then x:=x+1; y:=0; else x:=0; y:=y+1;**

б) **if 1<x and x<2 then begin x:=x+1; y:=0 end; else begin x:=0; y:=y+1end.**

Задача 5.9. Требуется записать условный оператор, вычисляющий значение **Z**, исходя из заданных значений **A, B, X, Y** по формуле:

$$Z = \begin{cases} 2, \text{ если } A>B; \\ Y, \text{ если } A=B; \\ X, \text{ если } A<B. \end{cases}$$

Решение: Составим схему программы (алгоритма) (рис. 5.4.) и напишем текст программы:

```
program pr1;
var Z, X, Y, A, B :integer;
  begin
    writeln('Введите числа X, Y, A, B');
    readln (X, Y, A, B);
      if A>B then Z:=2
      else if A=B then Z:=Y else Z:=X;
  writeln('Z=', Z);
  end.
```

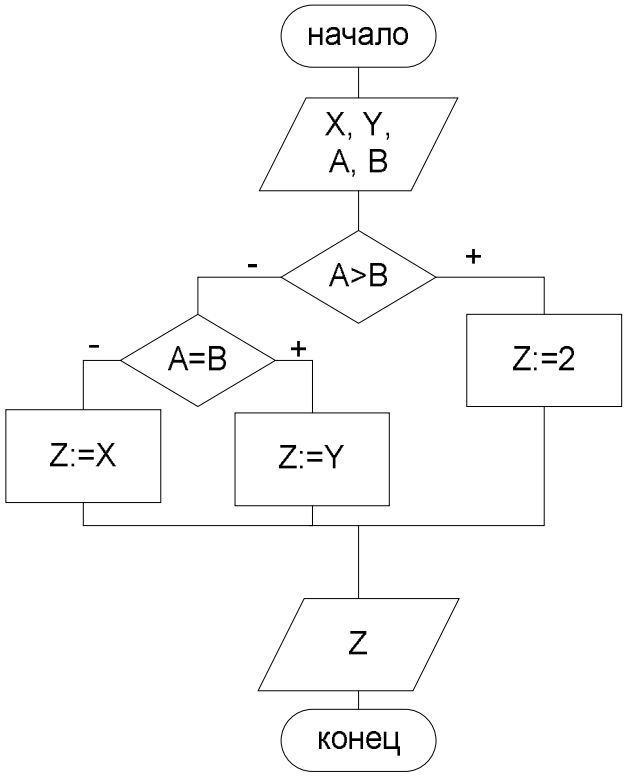


Рисунок 5.4. — Схема программы **pr1**

Задача 5.10.: Требуется написать программу, печатающую название дня недели в зависимости от введенного числа — номера дня недели.

Решение: Составим схему алгоритма, используя оператор выбора (рис. 5.5.) и напишем текст программы:

```
Program weekday.
  var i:integer;
  begin
    writeln('i->');  read (i);
        case i of
        1: writeln('понедельник');
        2: writeln('вторник');
        3: writeln('среда');
```

```
        4: writeln('четверг');
        5: writeln('пятница');
        6: writeln('суббота');
        7: writeln('воскресенье');
        end;
end.
```

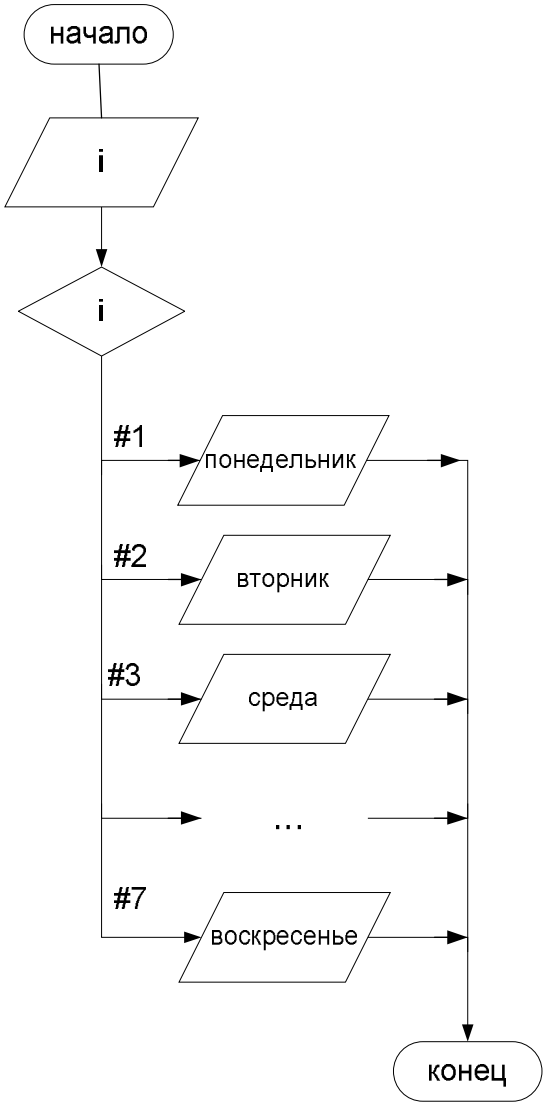


Рисунок 5.5. — Схема программы **weekday**.

Задача 5.11.: Написать программу, которая определяет, находится ли точка с заданными координатами выше прямой **y=x**. Результат программы вывести в виде сообщения: «да» или «нет» (рис. 5.6.)

Решение:

```
Var x, y, r: real;
Begin
        writeln (‘введите x, y’);read (x, y);
        if y > x     then write(‘да’)  else write(‘нет’);
end.
```

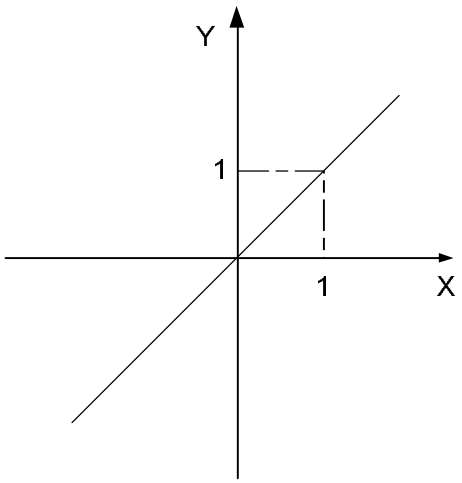


Рисунок 5.6.— График функции **y=x**

5.3.5. Задания для самостоятельной аудиторной работы

Задача 5.12. Написать программу, которая для заданного целого числа **a** печатает следующую таблицу:

```
a
a3   a6
a6   a3   a
```

Задача 5.13. Имеется программа:

```
Var x : integer;
begin
  x:=2; writeln('x+1') end.
```

Что она напечатает: **3** или **x+1**?

Задача 5.14. Найти ошибки в каждой из следующих программ:

а) **const d=5;**
begin d:=sqrt(d); writeln('d2=', d) end.**

б) **const k=true; var x:real;**
begin read(x)); writeln(ord(x)=k) end.

в) **var a, b, c:integer;**
begin read (a,b); writeln((a+b+c)/3) end.

г) **var x:real;**
begin read(x); y:=sqrt(x)+1; writeln(y) end.

д) **const B=2.5; var a, b, c:real;**
begin read(a,c); writeln(a*c>b) end.

Задача 5.15. Записать условный оператор, который эквивалентен оператору присваивания **x:=a or b and c** (все переменные — логические) и в котором не используются логические операции (например, оператору **not a** эквивалентен оператор **if a then x:=false else x:=true**).

Решение**: if a then x:=true else**
if b then x:=c else x:=false.

Задача 5.16. Записать оператор присваивания, эквивалентный условному оператору **if a then x:=b else x:=c,** где все переменные — логического типа.

Задача 5.17. Написать программу, которая определяет, попадает ли точка с заданными координатами в область, закрашенную на рисунке серым цветом. Результат программы вывести в виде сообщения: «да» или «нет» (рис. 5.7.)

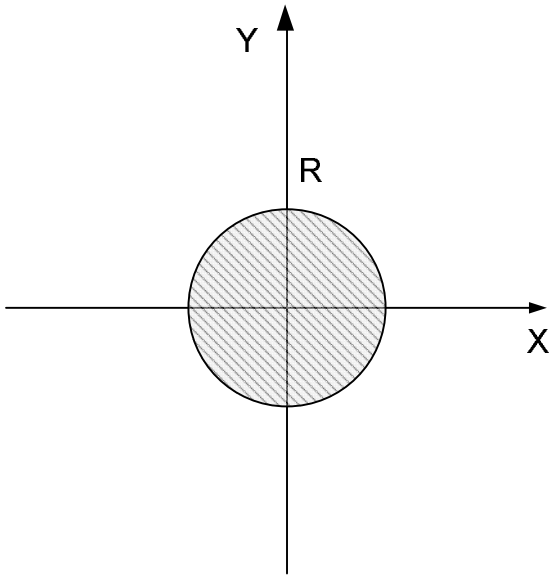


Рисунок 5.7.— График функции для задачи 5.17

Решение:

```
Var x, y, r: real;
Begin
writeln (‘введите x, y, r’);
read (x, y, r);
if sqr(x)+sqr(y)<sqr(r)
       then write(‘да’)
      else write(‘нет’);
end.
```

5.3.6. Задания для домашней работы

Вариант 5.1.

1. Определить, какие действия выполняет программа, описать входные и выходные данные, нарисовать схему алгоритма.

```
Var x,a,b,y:Real;
begin
  Writeln('Введите x, a, b');
  Readln(x,a,b);
   if x<a then  y:=x*x
   else
      if (x>a) and (x<b) then y:=sin(x)
      else y:=cos(x) +sin(x);
     Writeln('Результат:');
     Writeln('При a=',a,' b=',b, ' x=',X,' Y=',Y:5:3);
end.
```

2. Написать программу, которая определяет, попадает ли точка с заданными координатами в закрашенную область на рис. 5.8. Результат программы вывести в виде сообщения: «да» или «нет».

3. С помощью оператора **case** написать программу, которая по номеру месяца выводит название времени года, которому он принадлежит.

4. Дана схема алгоритма (рис. 5.8.). Требуется написать программу, соответствующую схеме.

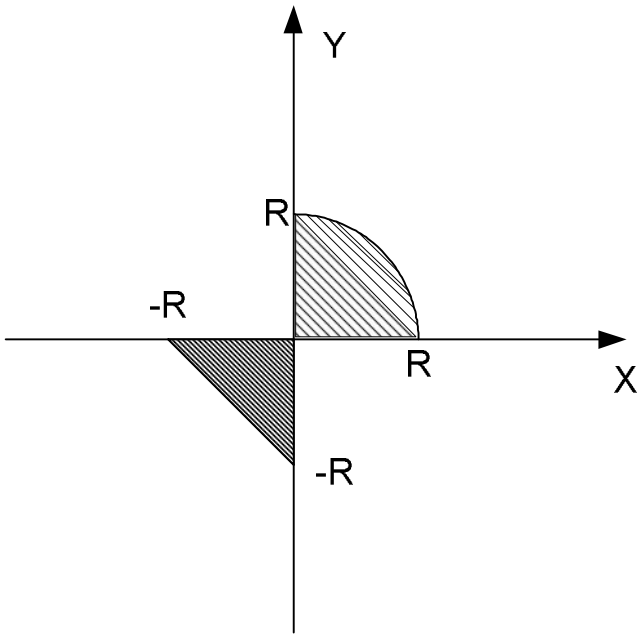


Рисунок 5.8.— Задание области для задачи 2

Вариант 5.2.

1. Определить, какие действия выполняет программа, описать входные и выходные данные, нарисовать схему алгоритма

```
var x,y,a,b:real;
begin
writeln('Введите занения x, a, b');
read(x,a,b);
 if x<a then  y:=exp(x)*sin(x);
else
   if (x>a)and(x<b)
      then y:=cos(x)*x*x;
 else   y:=x*x*x*x*x;
        writeln('y=',y:5:3);
end.
```

2. Написать программу, которая определяет, попадает ли точка с заданными координатами в закрашенную область на рис. 5.10. Результат программы вывести в виде сообщения: «да» или «нет».

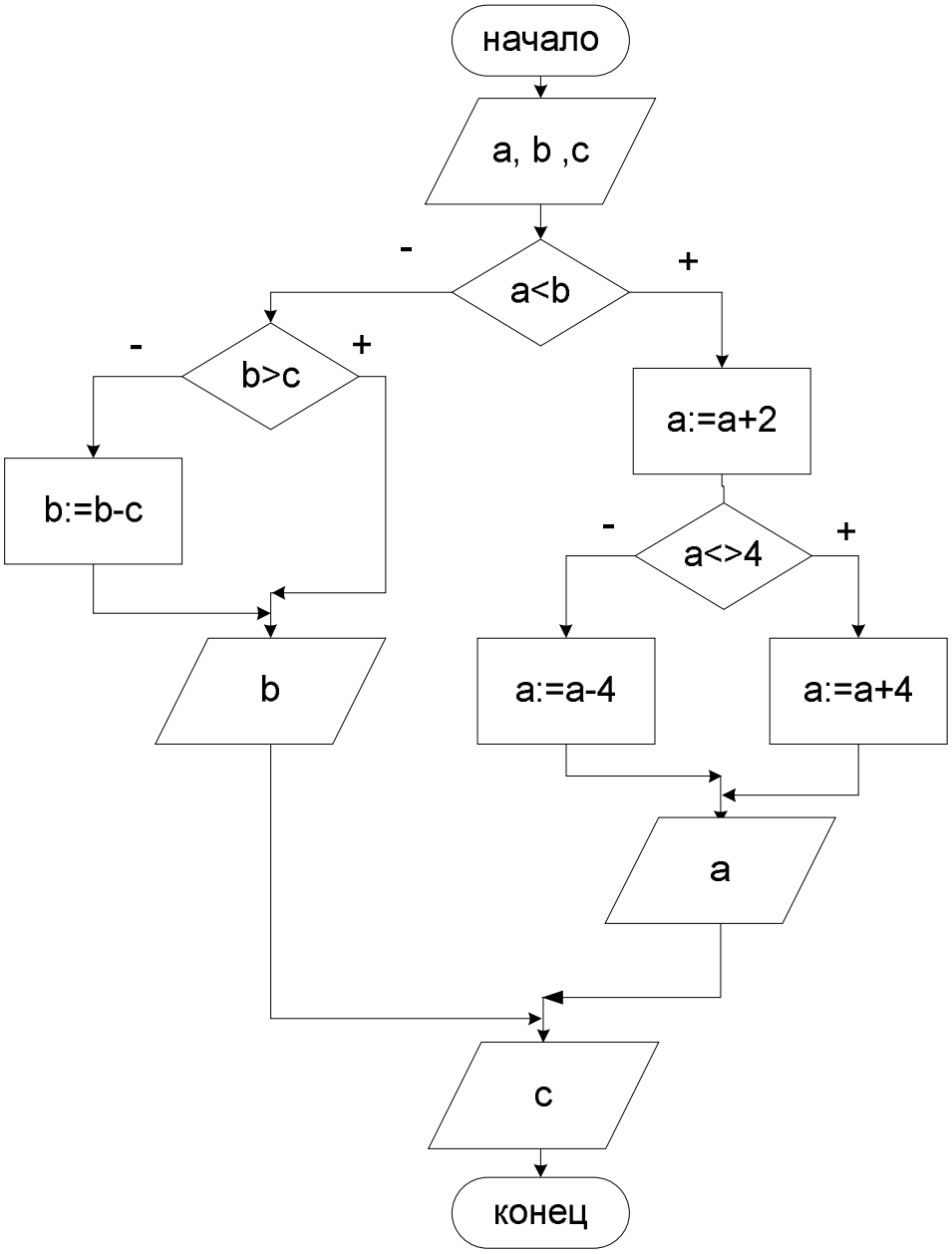


Рисунок 5.9.— Схема алгоритма для задачи 4 (вариант 5.1.)

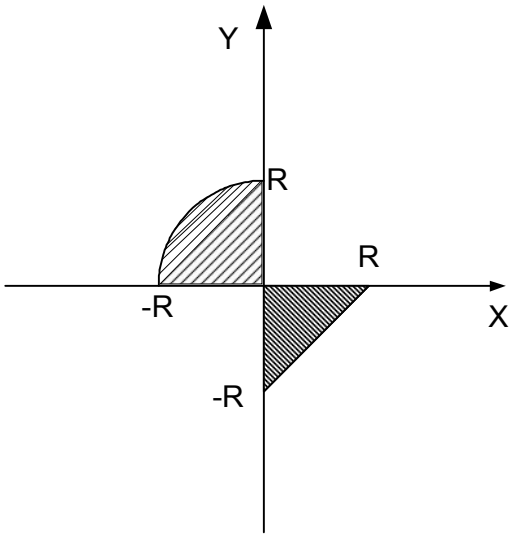


Рисунок 5.10.— Задание области для задачи 2 (вариант 5.2.)

3. С помощью оператора **case** написать программу, которая по введенной оценке (0..50) выводит название оценки (отлично: 45..50, хорошо: 36..44, удовлетворительно: 25..35, неудовлетворительно: 0..24).

4. Дана схема алгоритма (рис. 5.11.). Требуется написать программу, соответствующую схеме.

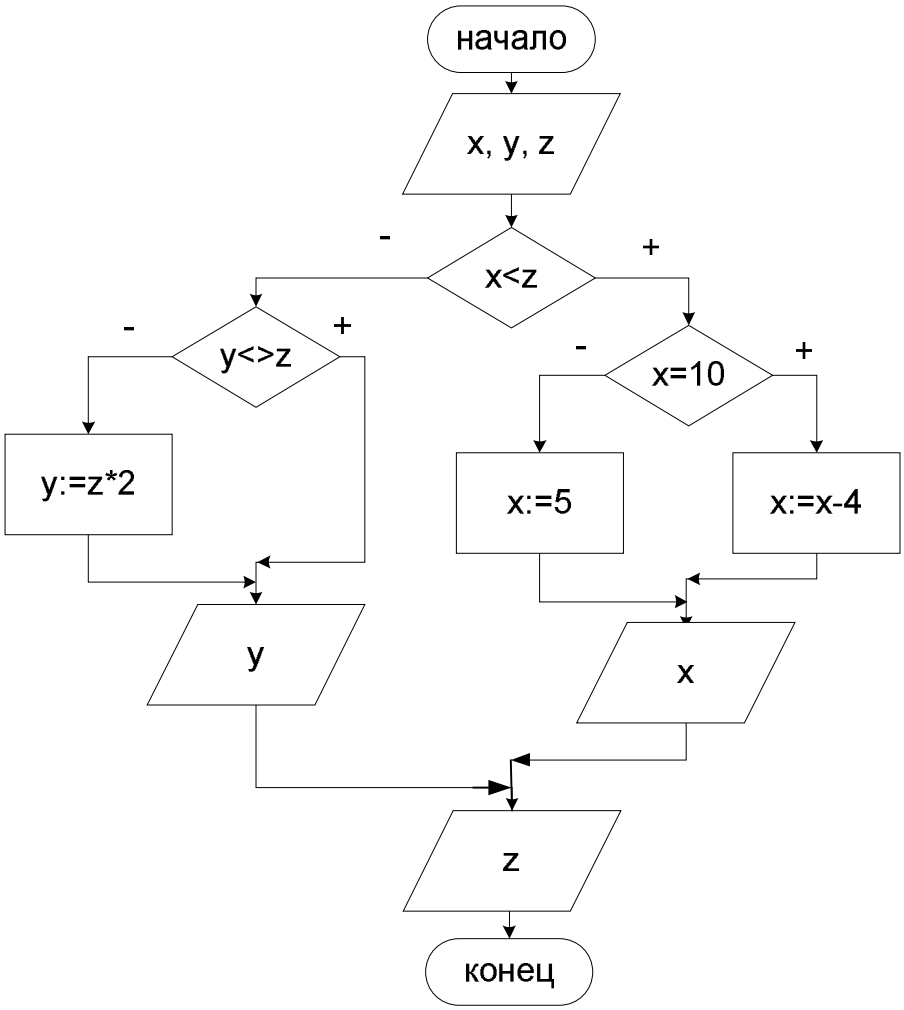


Рисунок 5.11.— Схема алгоритма для задачи 4 (вариант 5.2.)

6. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6
Программирование алгоритмов циклической структуры

6.1. Цели и задачи занятия

Выработать навыки разработки алгоритмов циклической структуры с использованием оператор циклов с предусловием, с постусловием и цикла с параметром. Выработать навыки решения задач с итерационными приближениями.

6.2. План занятия

6.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
6.2.2. Цель и задачи занятия (5 мин.).
6.2.3. Обсуждение теоретических вопросов (20 мин.):
- общее понятие цикла;
- цикл с предусловием;
- цикл с постусловием;
- цикл с параметром;
- приближенные итерационные вычисления.

6.2.4. Решение аудиторных задач (20 мин.).
6.2.5. Выполнение самостоятельных аудиторной работы (15 мин.).
6.2.6. Выдача домашнего задания (5 мин.).
6.2.7. Подведение итогов занятия (5 мин.).

6.3. Содержание занятия

6.3.1. Краткие сведения об операторах циклов

Существуют три вида циклов: цикл с параметром (**for**), цикл с предусловием (**while)** и цикл с постусловием (**repeat)**.

Цикл с параметром for является одним из основных видов циклов. Операторы, находящиеся внутри цикла, выполняются фиксированное число раз, в то время как переменная цикла пробегает определенный ряд значений.

Синтаксис операторов:

for <переменная> := <выражение 1> to <выражение 2> do <оператор>;
for <переменная> := <выражение 1> downto <выражение 2> do <оператор>;

где **<переменная>** — переменная любого ординального типа; **<выражение 1>** — выражение, определяющее начальное значение переменной; **<выражение 2>** — выражение, определяющее конечное значение переменной.

При выполнении оператора **for** вначале вычисляется выражение **<выражение 1>** и осуществляется присваивание **<переменная>:=<выражение 1>**. После этого циклически повторяется — проверка условия **<переменная> <= <выражение 2>**; если условие не выполнено, оператор **for** завершает свою работу. Для получения следующего значения переменной внутри цикла **for** автоматически применяются функции **Succ** или **Pred**. Структурная схема цикла приведена на рис. 6.1.

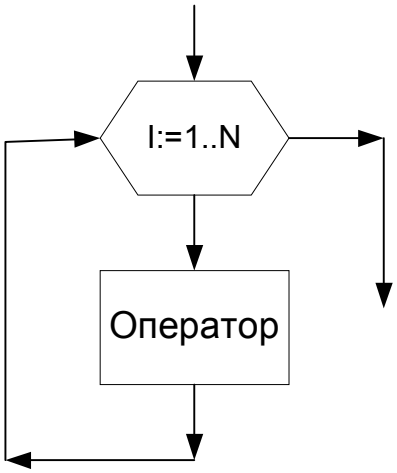


Рисунок 6.1.— Схема цикла с параметром

Пример 6.1. Печать квадратов чисел от 2-х до 10-и:

```
for x:=2 to 10 do WriteLn(x*x).
```

Пример 6.2. Печать латинского алфавита:

```
for ch:='A' to 'Z' do Writeln(ch).
```

Пример 6.3. Использование цикла с downto.

```
for i:=10 downto 1 do WriteLn(i);
```

Пример 6.4. Использование составного оператора.

```
for x:=2 to 10 do begin y:= x*x; WriteLn(y) end.
```

Синтаксис оператора цикла с предусловием:

while <логическое выражение> do <оператор>,

где **<логическое выражение>** — условие выполнения оператора; **<оператор>** — произвольный операторы Паскаля. Если **<логическое выражение>** принимает значение **True**, то выполняется **<оператор>**; после чего повторяется проверка **<логического выражения>**. Важно, чтобы **<оператор>** воздействовал на **<логическое выражение>,** чтобы через конечное число шагов, оно принимало значение **false**, иначе цикл окажется бесконечным. Схема цикла представлена на рис. 6.2.

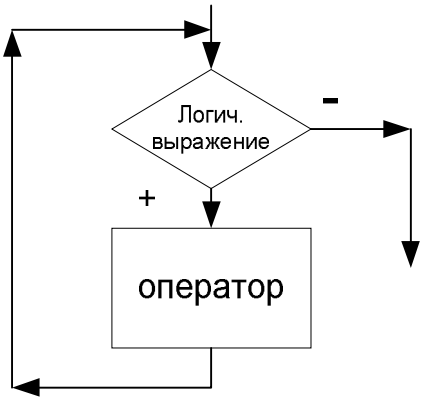


Рисунок 6.2.— Схема цикла с предусловием

Пример 6.5. Выполнить ввод символов с клавиатуры, пока не будет введен символ **‘*’**.

```
var   S: char;
begin
```

```
 While S <> '*' do Readln(S);
Write('Конец программы!');
end.
```

Пример 6.6. Программа «калькулятор»; программа должна вводить аргументы **A** и **B** знак операции, в зависимости от знака операции должны выполняться действия:

```
"+"?  -> C = A + B;
"-"? -> C = A - B;
"*"? -> C = A * B.
```

Программа должна отображать результат операции и приглашение «повторить» вычисления.

```
var     A,B,C: Integer;
              Ch, Sign: Char;
  begin Ch:='Y';
  While Ch <> 'N' do
  begin
  Write('Введите A: ');    Readln(A);
  Write('Введите B: ');     Readln(B);
  Write('Введите знак операции: ');    Readln(Sign);
  If Sign='+' then C := A + B;
  If< Sign='-' then C := A - B;
  If Sign='*' then C := A * B;
  Writeln('Результат: ', C);
  Write('Повторить? (Y/N): ');
  Readln(Ch);
  end;
  Write('Конец программы');
  end.
```

Синтаксис оператора цикла с постусловием:

repeat <оператор>{;<оператор>} until <логическое выражение>.

<Оператор> выполняется хотя бы один раз, после чего вычисляется **<логическое выражение>.** Если его значение есть **false**, то **<оператор>** повторяется, в противном случае оператор цикла **repeat until** завершает свою работу.

Схема цикла приведена на рис. 6.3.

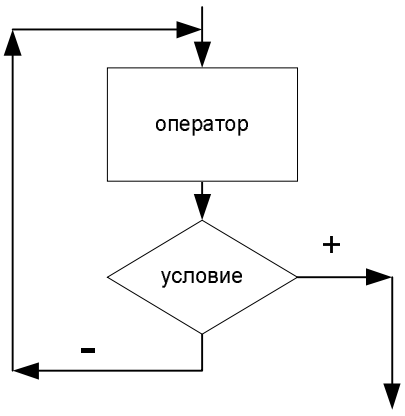


Рисунок 6.3. — Схема цикла **repeat until**

Пример 6.7. Определите какие действия выполняет программа (повторяет запрос на ввод числа, пока не будет введено положительное число):

```
Var x:real;
    begin
     repeat
        WriteLn('Введите положительное число');
        ReadLn(x);
    until x>0;
    end.
```

Операторы завершения цикла

В версии Турбо Паскаль 7.0 определены процедуры **Break** и **Continue**. Процедура **Break** выполняет безусловный выход из цикла. Процедура **Continue** обеспечивает переход к началу новой итерации цикла. Процедура **Exit** обеспечивает выход из программы.

6.3.2. Краткие сведения об итерационных приближениях

Пусть задана последовательность чисел **R1, R2, R3, …, Rn, …** Выражение **R1+R2+R3+…+Rn+…** называют бесконечным рядом, или просто рядом, а числа **R1, R2, R3, …**— членами ряда. Ряд называется сходящимся, если последовательность его частичных сумм имеет предел, и расходящимся — в противном случае. Рассмотрим задачи, сводящиеся к нахождению суммы элементов ряда; накопление суммы ряда будем выполнять до тех пор, пока очередной член ряда по абсолютной величине не станет меньше заданной величины ***E***.

Пример 6.8. Требуется вычислить e^{-x} с заданной точностью ***E***.

$$e^{-x} \approx 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots$$

Выведем реккурентную формулу для вычисления члена ряда:

$$t_n = (-1)^n \frac{x^n}{n!};$$

$$t_{n-1} = (-1)^{n-1} \frac{x^{n-1}}{(n-1)!};$$

$$\frac{t_n}{t_{n-1}} = \frac{(-1)x(n-1)!}{n!} = -\frac{x}{n},$$

т.е. $t_n = \frac{-x}{n} \cdot t_{n-1}.$

Вычисления завершаем, когда выполняется условие $|t_n - t_{n-1}| \le E.$

Пример 6.9. Требуется вычислить

$$\ln(x) \approx (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \ldots, x \ge 0.$$

Выведем реккурентную формулу для вычисления члена ряда:

$$t_n = (-1)^{n-1} \frac{(x-1)^n}{n};$$

$$t_{n-1} = (-1)^n \frac{(x-1)^{n-1}}{n-1};$$

$$\frac{t_n}{t_{n-1}} = \frac{(x-1)(n-1)}{(-1)n} = -\frac{(x-1)(n-1)}{n},$$

т.е. $t_n = t_{n-1} \cdot \frac{(x-1)\cdot(n-1)}{n}$.

Вычисления завершаем, когда выполняется условие $|t_n - t_{n-1}| \le E$.

Частный случай: $x = 1$.

Поскольку $\ln(x)$ при $x \to +\infty$ стремится к $+\infty$, то в программу необходимо будет добавить счетчик итераций!

6.3.3. Аудиторные задачи и примеры их решения

Задача 6.1. Вычислить сумму членов ряда с точностью до члена ряда меньшего 0.0001: **S=1/(2*3)+2/(3*4)+...+n/[(n+1)(n+2)]+...**

Решение: Решим задачу двумя способами, используя циклы **while** и **repeat**:

а)
```
const eps=0.0001;
  var s,el:real; n:integer;
begin
s:=0; n:=1; el:=n/((n+1)*(n+2));
        while el>eps do
        begin
                s:=s+el;
                n:=n+1;
                el:=n/((n+1)·(n+2));
        end;
write('s=', s:10:8);
end.
```

б)
```
const eps=0.0001;
   var s,el:real; n:integer;
        begin
        s:=0; n:=0; el:=0;
        repeat
                 s:=s+el;    n:=n+1;
                el:=n/((n+1)·(n+2));
        until el<=eps;
write('s=', s:10:8);
end.
```

Задача 6.2. Вычисление **f=10!** Описать каждым из трех вариантов оператора цикла

Решение:

а) **i:=2; f:=1; while i<=10 do begin f:=f*i; i:=i+1 end;**

б) **i:=2; f:=1; repeat f:=f*i; i:=i+1until i>10;**

в) **f:=1; for i:=2 to 10 do f:=f*i.**

Задача 6.3. Определить значение переменной **s** после выполнения следующих операторов:

а) **s:=0; i:=0;**
while i<5 do i:=i+1; s:=s+1/i;

б) **i:=1; s:=0;**
while i>1 do begin s:=s+1/i; i:=i-1 end;

в) **s:=0; i:=1;**
repeat s:=s+1/I; i:=i-1 until i<=1;.

г) **s:=1; n:=1;**
for i:-2 to n do s:=s+1/i.

Ответы:

а) **0.2;** б) **0;** в) **1.0;** г) **1.0.**

Задача 6.4. Вычислить **y=(2·n–1)!=1·3·5·…·(2·n–1), N>0.**

Решение: **y:=1; for i:=1 to n do y:=y·(2·i–1);**

Задача 6.5. Вычислить сумму членов ряда с точностью до члена ряда меньшего 0.001: $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

Решим задачу, используя реккурентную формулу $\frac{t_n}{t_{n-1}} = -\frac{x^2}{(2n-1)2n}$.

```
const eps=0.0001;
var s,el,x:real; n:integer;
 begin
   s:=0; n:=0; el:=1;
   writeln('x->');   read(x);
   while abs(el)>eps do
    begin
      s:=s+el;
      n:=n+1;
      el:=el*(-1)*sqrt(x)/((2*n-1)*2*n);
    end;
  write('s=',s);
 end.
```

Задача 6.6. Вычислить сумму членов ряда с точностью до члена ряда меньшего 0.001: $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

Решим задачу, используя реккурентную формулу $t_n = \frac{x}{n} \cdot t_{n-1}$.

```
const eps=0.0001;
var s,el,x:real; n:integer;
 begin
   s:=0; n:=0; el:=1;
   writeln('x->');   read(x);
   while abs(el)>eps do
   begin
     s:=s+el;
     n:=n+1;
```

```
            If n>100 then begin write('error'); break; end;
          el:=el*x/n;
        end;
      write('s=',s);
    end.
```

6.3.4. Задания для самостоятельной аудиторной работы

Задача 6.7. Вычислить:

а) $y = \cos x + \cos x_2 + \cos x_3 + \ldots + \cos x_{30}$;

б) $y = 1! + 2! + 3! + \ldots + n!$ **(n>1).**

Задача 6.8. Сколько раз будет выполняться тело следующего оператора цикла?

```
k:=0;
for i:=1 to k+3 do k:=k+1.
```

Ответ: 3 раза (выражение, стоящее после служебного слова **to**, вычисляется только один раз вначале выполнения цикла).

Задача 6.9. Если среди чисел $\sin x^n$ **(n=1, 2, ..., 30)** есть хотя бы одно отрицательное число, то логическое переменной присвоить значение **true**, а иначе — значение **false**. Для решения задачи использовать цикл **for**.

6.3.5. Задания для домашней работы

Вариант 6.1.

1. Рассмотреть какие действия выполняет программа, описать входные и выходные данные, изобразить схему программы

```
Var x,s:real;
begin
s:=0;
 repeat
   WriteLn('Введите число, для выхода введите -999');
     ReadLn(x);
       If x>0 then s:=s+x;
  until x=-999;
 writeln('s=',s:10:5)
end.
```

2. Используя цикл **while do**, написать программу вычисления членов ряда **S= 1+ 1/2+ 1/3 + …+ 1/n,** с точностью до члена ряда, меньшего **0.0005**, а также нарисовать схему программы.

3. Используя цикл **repeat until**, написать программу вычисления членов ряда **S= cos(x)+cos(2x)/4+ cos(3x)/9+... +cos(nx)/n^2**, с точностью до члена ряда, меньшего **a** по модулю. Значение **a** выбирает пользователь.

4. Написать программу, соответствующую схеме алгоритма, изображенной на рис. 6.4.

5. Написать программу, вычисляющую **y=(2n)!, n>0.**

6. Дано 100 вещественных чисел. Написать программу, определяющую, образуют ли они возрастающую последовательность.

7. Дано 200 вещественных чисел. Написать программу, определяющую, сколько из них больше своих «соседей», т.е. предыдущего и последующего чисел.

8. Дано натуральное число. Написать программу, определяющую число, получаемое выписыванием в обратном порядке цифр заданного числа.

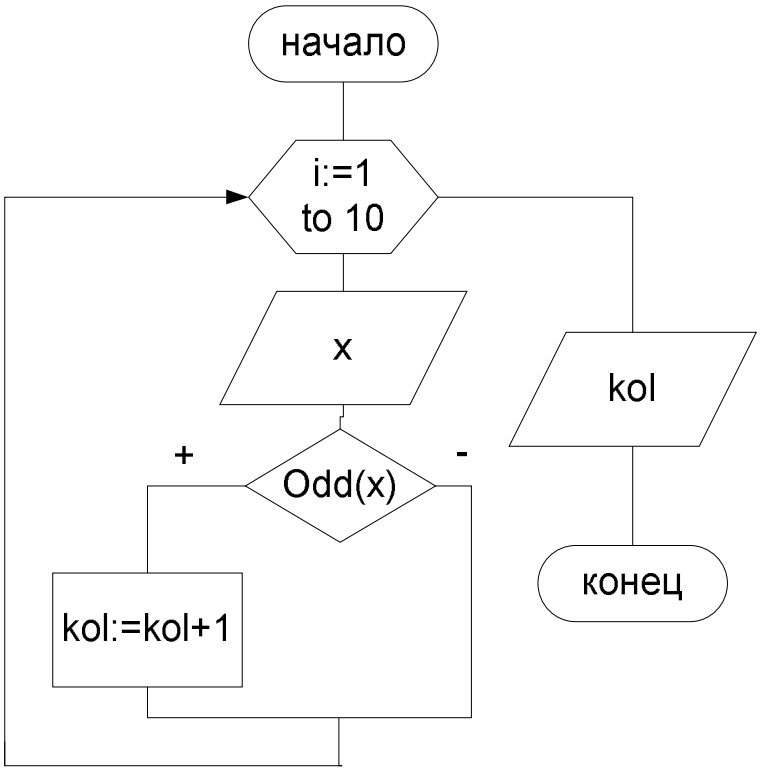


Рисунок 6.4. — Схема алгоритма для задачи 4 (вариант 6.1)

Вариант 6.2.

1. Рассмотреть какие действия выполняет программа, описать входные и выходные данные, изобразить схему программы

```
Var x:real; k:integer;
begin
k:=0;
 WriteLn('Введите число');
 ReadLn(x);
while x<>999 do
   begin
    If x<=0 then k:=k+1;
    WriteLn('Введите число');
    ReadLn(x);
   end;
writeln('k=',k:10:5)
end.
```

2. Используя цикл **while do**, написать программу вычисления членов ряда **S=1/(3*4)+2/(4*5)+...+n/[(n+2)(n+3)]**, с точностью до члена ряда, меньшего **0.0005**, а также изобразить схему программы.

3. Используя цикл **repeat until**, написать программу вычисления членов ряда **S=cos(x)+cos(2x)/2+cos(3x)/3+...+cos(nx)/n**, с точностью до члена ряда, меньшего **a** по модулю. Значение **a** выбирает пользователь.

4. Написать программу, соответствующую схеме алгоритма, изображенной на рис.6.5.

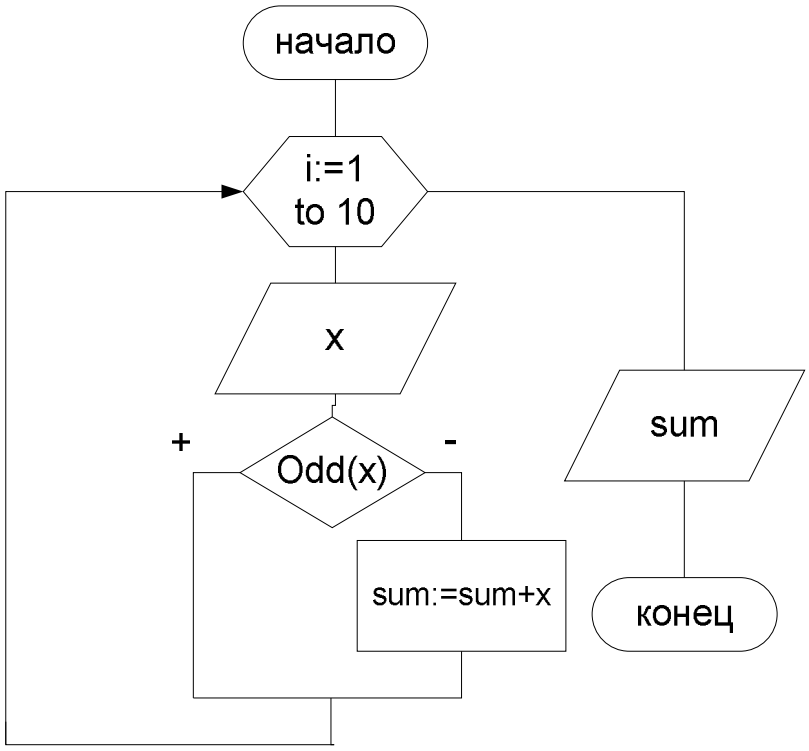


Рисунок 6.5. — Схема алгоритма для задачи 4 (вариант 6.2)

5.Написать программу, вычисляющую **y=(2n-1)!, n>0.**

6. Дано 100 вещественных чисел. Написать программу, определяющую, образуют ли они убывающую последовательность.

7. Дана непустая последовательность ненулевых целых чисел, за которой следует 0. Написать программу, определяющую, сколько раз в этой последовательности меняется знак. (Например, в последовательности 1, -34, 8, 14, -5 знак меняется 3 раза).

8. Дано натуральное число. Написать программу, вычисляющую сумму цифр этого числа.

7. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7

Программирование алгоритмов обработки массивов с использованием процедур и функций

7.1. Цели и задачи занятия

Выработать навыки разработки алгоритмов выполнения основных операций с двумерными массивами. Выработать навыки разработки программ блочной структуры с использованием процедур и функций

7.2. План занятия

7.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
7.2.2. Цель и задачи занятия (5 мин.).
7.2.3. Обсуждение теоретических вопросов (20 мин.):
- способы описания массивов;
- основные операции обработки одномерных и двумерных массивов;
- описание процедур и функций;
-обращение к процедурам и функциям.
7.2.4. Решение аудиторных задач (20 мин.).
7.2.5. Выполнение самостоятельных аудиторной работы (15 мин.).
7.2.6. Выдача домашнего задания (5 мин.).
7.2.7. Подведение итогов занятия (5 мин.).

7.3. Содержание занятия

7.3.1. Описание массивов и основные операции над массивами

Задание массива (регулярного типа) в языке Pascal выполняют с помощью следующей конструкции:

array [<тип индекса>] of [<тип элемента>]

Тип элемента массива может быть любым, в том числе и массивом. Это позволяет задавать многомерные массивы. Тип индекса массива может быть либо ограниченным, либо перечислимым, либо литерным, либо логическим. Наиболее часто используют ограниченный тип.

Пример 7.1. Объявление одномерного массива.

```
Const n=10;
Type  vector = array [1..n] of real;
Var   x: vector;
```

Многомерный массив можно рассматривать как одномерный, но элементом такого массива является массив. Приведем примеры:

Пример 7.2. Объявление двумерного массива (1 способ).

```
Const n=10; m=4;
Type stroka = array [1..n] of real;
matrix = array [1..m] of stroka;
Var   a: matrix;
```

Пример 7.3. Объявление двумерного массива (2 способ).

```
Const n=10; m=4;
```

```
Type matrix = array [1..n] of array [1..m] of real;
Var   a: matrix;
```

Пример 7.4. Объявление двумерного массива (3 способ).

```
Const n=10; m=4;
Type  matrix = array [1..n, 1..m] of real;
Var   a: matrix;
```

Чаще всего используют третий способ объявления для многомерных массивов.

Пример 7.5. Примеры обращения к элементам массива.

```
write (x[5]); x[2]:=3.12;
write (a[3][2]); a[3,2]:=3.12;
```

Пример 7.6. Чтение с клавиатуры и вывод на экран элементов массивов.

```
var x: vector; a: matrix; l,j: integer;
begin
writeln('введите элементы одномерного массива');
   for i:=1 to n do read(a[i]);
writeln('элементы массива');
  for i:=1 to n do write(‘ ‘,a[i]:5:3);
 {печать в обратном порядке}
writeln('элементы массива');
  for i:=n downto 1 do write(‘ ‘,a[i]:5:3);
writeln('введите элементы многомерного массива');
   for i:=1 to n do
     for j:=1 to m do
read(a[i][j]);
writeln('элементы массива');
   for i:=1 to n do
begin
     for j:=1 to m do   write(‘ ‘,a[i]:5:3);
     writeln;
end;
end.
```

Пример 7.7. Перестановка 2-х элементов в одномерном массиве (первого и последнего).

```
 Const n=10;
Var temp: real; x: vector; …
 begin  …
       temp:=x[1];
       x[1]:=x[n];
       x[n]:= temp; …
end.
```

Пример 7.8. Перестановка 2-х элементов в многомерном массиве с индексами [1,1] и [n,m].

```
Const n=5; m=4;
Var temp: real; a: matrix;…
 begin  …
       temp:=a[1,1];
       a[1,1]:=x[n,m];
       x[n,m]:= temp; …
end.
```

7.3.2. Краткие сведения о процедурах и функциях

Процедуры (функции) — группа логически связанных операторов, которые имеют имя, и которые можно вызвать по имени любое количество раз из различных мест программ. Процедуры и функции позволяют исключить многократную запись одних и тех же участков кода в программе.

Описание процедур располагается после раздела описаний переменных. Процедура оформляется аналогично программе. Отличие состоит в заголовке процедуры(procedure). Различают процедуры с параметрами и без. Для процедуры с параметрами в заголовке указывают список формальных параметров:

Procedure <имя> (<список формальных параметров>); <блок> ;

Список формальных параметров включает имена переменных и их типы. Формальные параметры используются для передачи процедуре исходных значений переменных и извлечения из процедуры необходимых результатов. Описание процедуры без параметров:

Procedure <имя>; <блок>;

Функция похожа на процедуру, но отличается от неё следующим:

- при описании используется слово **function**;
- функция возвращает один результат.

Функция возвращает результат через своё имя. Имя функции можно использовать в выражениях. Поэтому в тексте описания функций, должен быть оператор, присваивающий имени функции результат вычислений. Так как результат вычислений функции возвращается через её имя, то оно должно соответствовать некоторому типу значений. Описание функции:

Function <имя> (<список формальных параметров>): <тип>;
<блок>;

Обращение к процедурам и функциям выполняется с помощью оператора вызова процедур

<имя> (<список фактических параметров>),

или

<имя>.

Перепишем пример 7.6., используя процедуру.

Пример 7.9. Чтение с клавиатуры и вывод на экран элементов массивов.

```
var x: vector;    a: matrix; l, j: integer;
procedure vvod_vector;
begin
        writeln('введите элементы одномерного массива');
           for i:=1 to n do read(a[i]);
end;
{процедура без параметров}
procedure print_vector;
begin
        writeln('элементы массива');
                  for i:=1 to n do write(‘ ‘,a[i]:5:3);
                {печать в обратном порядке}
        writeln('элементы массива');
         for i:=n downto 1 do write(‘ ‘,a[i]:5:3);
 end;
```

```
procedure vvod_matrix;
begin
        writeln('введите элементы многомерного массива');
           for i:=1 to n do
             for j:=1 to m do
                read(a[i][j]);
end;
procedure print_matrix;
begin
        writeln('элементы массива');
           for i:=1 to n do
                begin
                 for j:=1 to m do write(‘ ‘,a[i]:5:3);
                 writeln;
                end;
end;
{основная программа}
begin                   {вызов процедур}
        vvod_vector;
        print_vector;
        vvod_matrix;
        print_matrix;
end.
```

Пример 7.10. Подсчет количества элементов одномерного массива, больших 10.

```
function kol:real;
begin
  kol:=0;
  for i:=1 to N do
  if a[i]>10 then kol:=kil+1;
end;
begin …
 writeln(kol); {вызов функции}
end.
```

При вызове процедур с параметрами фактические параметры подставляются на место формальных, поэтому списки фактических и формальных параметров должны полностью соответствовать друг другу. В списке фактических параметров указываются имена переменных, отделяемые друг от друга запятой. В списке формальных параметров, переменные отделяются точкой с запятой. **В списке формальных параметров запрещено выполнять задание типа, можно использовать только уже существующие типы.**

Различают ***локальные и глобальные имена***. Любое имя программы имеет свою ***область действия*** — блок, в котором оно введено, включая вложенные блоки. Переменные, описанные в основной программе — глобальные, так как доступны в любом вложенном блоке. Переменные, описанные внутри процедуры или функции — локальные. Значение локальных переменных доступны, только пока блок активен. Если локальное имя совпадает с глобальным, то оно переопределяет глобальное имя в пределах блока. Переменные, описанные в списке формальных параметров — ло-

кальные. Помимо механизма глобальной памяти различают следующие способы передачи значений процедуре и функции:

1) параметры-значения;
2) параметры-переменные;
3) параметры-процедуры;
4) параметры-функции;

Пример 7.11. Пример использования параметров-значений и параметров–переменных.

```
procedure p(x: real; var y:real);
begin
x:=x+1;
y:=y+1;
end;
var a,b: real;
begin
 a:=1; b:=1;
 p(a,b);
 write('a=', a:3:0, 'b=', b:3:0);   {a=1, b=2}
end.
```

Если в списке параметров процедуры перед переменными не стоит **var**, то они являются параметрами-значениями **(x: real).**

Параметры-значения передаются в процедуру путем копирования. При вызове процедуры на место фактических параметров-значений можно подставить выражения, например: **p (2 + x, b).**

В случае передачи параметров-переменных процедура получает адрес параметра-переменной и может менять его значение. При вызове процедуры нельзя подставлять вместо параметров-переменных выражения. Параметры-значения служат для передачи исходных данных в процедуру, а параметры-переменные обычно — для получения результата из процедуры.

Пример 7.12. Пример использования параметров-значений и параметров-переменных.

```
uses crt;
var x:real; y:integer;
procedure p(a:real; b:integer; var c:real; var d:integer);
begin
   x:=x+1; writeln('x=',x:3:3);
   y:=y+1; writeln('y=',y);
   a:=a+1; writeln('a=',a:3:3);
   c:=c+1; writeln('c=',c:3:3);
   b:=b+1; writeln('b=',b);
   d:=d+1; writeln('d=',d);
end;
 var x1,x2:real; y1,y2:integer;
begin
clrscr;
   x:=1; y:=1;   x1:=1; x2:=1;   y1:=1; y2:=1;
   p(x1,y1,x2,y2);
   write('x=',x:3:3,' y=',y,' x1=',x1:3:3,' y1=',y1,' x2=',x2:3:3,' y2=',y2);
```

```
end.
```

Результат выполнения программы:

```
x=2.000
y=2
a=2.000
c=2.000
b=2
d=2
x=2.000 y=2 x1=1.000 y1=1 x2=2.000 y2=2
```

Пример 7.13. Функция поиска элемента в массиве.

```
Type  massiv = array [1..n] of real;
function poisk(el:real, x:massiv):boolean;
var fl: boolean;
begin
    fl:=false;
    for i:=1 to n do if x[i]=el then fl:=true;
    poisk:=fl;
end;

{основная программа}
var el:real;
begin
…
  write('введите элемент'); read(el);
  if poisk(el)=true then writeln('да')
        else writeln('нет');
end.
```

7.3.3. Аудиторные задачи и примеры их решения

Задача 7.1. Вычислить сумму отрицательных элементов в одномерном массиве, двумерном массиве, с помощью функции.

Задача 7.2. Вычислить количество четных элементов в одномерном массиве, двумерном массиве, с помощью функции.

Задача 7.3. Поменять 2 строки в многомерном массиве, номера строк передать параметрами в процедуру.

Задача 7.4. Вычислить максимальный элемент в одномерном массиве и минимальный в двухмерном массиве, используя функцию.

Задача 7.5.

```
const n=3;
type matr=array [1..n,1..n] of real;
var  i,j:integer;    a:matr;
```

Вычислить сумму элементов, находящихся на главной диагонали, а также вывести их на экран.

```
function sled(a:matr):real;
var s:real;
begin s:=0; {инициализируем сумму}
for i:=1 to n do
  begin
    s:=s+a[i][i];       {накапливаем сумму}
    write(' ',a[i][i]:3:3); {выводим на экран элементы гл. диагонали}
  end;
```

```
 sled:=s;              {возвращаем значение суммы через имя функции}
end;
```

Задача 7.6. Для данных задачи 7.5. выведем на экран элементы, находящихся на вспомогательной диагонали.

```
procedure diag_vs(a:matr);
begin
  for j:=1 to n do  write(' ',a[j][n+1-j]:3:3);
end;
```

Задача 7.7. Для данных задачи 7.5. выведем на экран элементы, находящихся выше главной диагонали.

```
procedure diag_gl_(a:matr);
begin
for i:=1 to n do
        for j:=1 to n do
                if (j>=i)then write(' ',a[i][j]:3:3);
end;
```

Если поменять в условии **if (j>=i)** знак на “меньше или равно”, то на экран будут выведены элементы, находящиеся ниже главной диагонали.

7.3.4. Задания для самостоятельной аудиторной работы

Задача 7.8. Вычислить произведение нечетных положительных элементов в одномерном массиве и двумерном массиве, с помощью функции.

Задача 7.9. С помощью функции вычислить сумму элементов в одномерном массиве, расположенных между первым и последними нулями (считаем, что в массиве количество нулевых элементов больше или равно 2).

Задача 7.10. Используя процедуру, поменять 2 столбца в многомерном массиве, номера столбцов передать параметрами.

Задача 7.11. Вычислить максимальный положительный элемент в одномерном массиве и минимальный нечетный в двухмерном массиве, используя функцию.

7.3.5. Задания для домашней работы

Вариант 7.1.

1. Дано *n* натуральных чисел $a_1...,a_n$. Написать **программу**, вызывающую следующие процедуры и функции:

- процедуру ввода элементов одномерного массива с клавиатуры;
- процедуру замены нулями всех элементов, меньших двух;
- функцию вычисления суммы элементов, принадлежащих интервалу [-3, 7];
- функцию поиска количества элементов, кратных 3 и не кратных 5;
- процедуру поиска минимального по модулю элемента и его индекса (номера его позиции);
- процедуру преобразующую массив так, чтобы сначала располагались все положительные элементы, затем все остальные;
- процедуру вывода на экран элементов массива.

2. Написать программу, содержащую следующие процедуры и функции, и вызывающую их:

- процедуру ввода матрицы (представленной в виде двумерного массива) с клавиатуры;
- функцию вычисления суммы всех элементов матрицы, расположенных не на главной диагонали;
- функцию поиска в матрице строки с минимальным элементом;
- процедуру перестановки двух заданных строк матрицы;
- процедуру вывода матрицы на печать.

Вариант 7.2.

1. Дано *n* натуральных чисел $a_1...,a_n$. Написать **программу**, вызывающую следующие процедуры и функции:

- процедуру ввода элементов одномерного массива с клавиатуры;
- процедуру замены положительных элементов на их квадраты;
- функцию вычисления произведения элементов, принадлежащих интервалу [-1, 3];
- функцию поиска количества элементов, кратных 4 и не кратных 7;
- процедуру поиска максимального по модулю элемента и его индекса (номера его позиции);
- процедуру преобразующую массив так, чтобы сначала располагались все четные элементы, затем все нечетные;
- процедуру вывода на экран элементов массива.

2. Написать программу, содержащую следующие процедуры и функции, и вызывающую их:

а) процедуру ввода матрицы (представленной в виде двумерного массива) с клавиатуры;

б) функцию вычисления суммы всех элементов матрицы, расположенных на главной диагонали;

в) функцию поиска в матрице строки с максимальны элементом;

г) процедуру перестановки двух заданных столбцов матрицы;

д) процедуру вывода матрицы на печать.

8. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8

Программирование операций над строками и записями

8.1. Цели и задачи занятия

Выработать навыки разработки алгоритмов выполнения основных операций над строками. Выработать навыки разработки алгоритмов для обработки информации, представленной в форме таблиц.

8.2. План занятия

8.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
8.2.2. Цель и задачи занятия (5 мин.).
8.2.3. Обсуждение теоретических вопросов (20 мин.):
- строковый тип;
- основные операции над строками;
- комбинированный тип;
- массивы структур.

8.2.4. Решение аудиторных задач (20 мин.).
8.2.5. Выполнение самостоятельных аудиторной работы (15 мин.).
8.2.6. Выдача домашнего задания (5 мин.).
8.2.7. Подведение итогов занятия (5 мин.).

8.3. Содержание занятия

8.3.1. Строковый тип, основные операции над строками

Строка представляет собой последовательность литер и в стандартном языке **Pascal** описывается в виде упакованного одномерного массива

```
Const N=10;
Var stroka: packed array[1..N] of char;
```

В языке **Turbo Pascal** строковый тип может быть также описан с помощью зарезервированного слова **String**.

Пример 8.1. Пример описания строковой переменной из 10 литер в языке **Turbo Pascal**

```
Type stroka= string[10];
Var s:stroka;
```

Пример 8.2. Пример описания строковой переменной из 10 литер в языке **Turbo Pascal**

```
Var stroka1: string[10];
```

Целое число в квадратных скобках задает длину строки. Максимально допустимая длина строки типа **string** составляет 255 символов и назначается строковой переменной по умолчанию, если длина не указана.

Строке можно присваивать значение строковой константы:

```
stroka1: ='ABC'.
```

К строкам можно применять операцию ***конкатенации***, которая обозначается знаком «+». Конкатенация — это объединение строк:

```
stroka2: =stroka1+'DEF'.
```

Результатом такой последовательности операторов будет строка **'ABCDEF'.**

В языке Turbo Pascal имеются дополнительные процедуры и функции работы со строками.

Пример 8.3. Операцию слияния строк **str1** и **str2** выполняет функция **Concat(str1, str2):**

```
Var S: String;
 begin
    S := Concat('ABC', 'DEF');
    Write(S);          { 'ABCDE' }
 end.
```

Функция **Length(str)** определяет длину аргумента **str** строкового типа.

Пример 8.4. Процедура **Delete(str, start, nrem)** используется для того, чтобы удалить **nrem** символов в строковой переменной **str**, начиная с позиции **start**.

```
Var   s: string;
begin
  s := '123456789';
  Delete(s,3,4);
  Writeln(s); { '12789' }
end.
```

Пример 8.5. Процедура **Insert** предназначена для вставки одной строки в другую. Процедура **Insert(instr, str, start)** выполняет вставку строки **Instr** в строку **str**, начиная с позиции **start**,

```
Var  s: string;
begin
  s := '12789';
  insert('3456',s,3);
  Writeln(s);     { '123456789' }
end.
```

Пример 8.6. Функция **Copy(str1, start, n)** копирует **n** символов строки **str1**, начиная с позиции start. Это строковое значение можно присвоить другой строковой переменной.

```
var S, S1: String;
begin
  S := 'ABCDEF';
  S1 := Copy(S, 2, 3);
   writeln(s1);        { 'BCD' }
end.
```

Пример 8.7. Функция **Pos(substr, str)** определяет начальную позицию подстроки **substr** в строке **str**.

```
var S: String; N:integer;
begin
  S := 'ABCDEF';
  N := POS('CD', S);
  writeln(N);{ 3 }
end.
```

Если подстрока не встретилась в строке, значение **Pos** ***равно нулю***.

Имеются две процедуры, преобразующие числовые значения в строковые и обратно — **Str** и **Val**. Процедура **Str** должна вызываться следующим образом:

Str(num, strnum).

Процедура преобразует число **num** в строку **strnum.** Процедура **Val** выполняет обратное преобразование. Обращение к ней имеет вид:

Val(strnum, num, errcode).

Третий параметр в этой процедуре равен нулю при успешном выполнении преобразования. В том случае, когда первый параметр содержит символы, недопустимые при записи числа, значение параметра **errcode** по окончании работы процедуры будет равно номеру позиции с ошибочно заданным символом.

Две строки можно сравнивать при помощи операций >, <, <=, >=. При этом сравниваются коды символов.

Пример 8.8. Сравним две строки **s1** и **s2**.

```
var s1,s2: string; i:integer;
begin
 s1:='abcd';
 s2:='xabcd';
  if (s1=s2) then  write('=')
  else if (s1>s2) then  write('>')
    else write('<')
end. {<}
```

Пример 8.9. В качестве примера работы со строками рассмотрим сортировку слов по алфавиту. Количество слов должно быть не больше 100 (количество вводит пользователь). Количество букв в слове не более 10. Слово заканчивается символом ‘,’ или ‘.’. Слова вводим с клавиатуры, записывает их в массив **sp**, сортируем и выводим на экран.

```
const N=100; {будем считать, что вводимых слов не более 100}
shablon='__________';
type slovo=packed array[1..10]of char;
     spisok=array[1..N] of slovo;
var sp:spisok;
   sl:slovo;
   c:char;
   i,j,k:integer;
{процедура чтения слов с клавиатуры и записи их в массив sp}
procedure zapol_sp;
begin
 writeln('введите k-количество слов');
  readln(k);
  for i:=1 to k do
   begin
    sl:=shablon;
    j:=1;
    read(c);
    while (c<>',')and(c<>'.')do
     begin
     sl[j]:=c;
     j:=j+1;
     read(c);
     end;
    sp[i]:=sl;
  end;
end;
```

```
procedure sort; {сортировка методом пузырька}
   begin
   for j:=k downto 2 do
    for i:=1 to j-1 do
     if(sp[i]>sp[i+1]) then
     begin
     sl:=sp[i];
     sp[i]:=sp[i+1];
     sp[i+1]:=sl;
     end;
   end;
   procedure print_sp; {процедура печати sp}
   begin
     for i:=1 to k do
     writeln(sp[i]);
   end;
          {основная программа}
   begin
    zapol_sp;
   writeln('печать до сортировки');  print_sp;
    sort;
   writeln('печать отсортированного массива');  print_sp;
   end.
```

8.3.2. Комбинированный тип, массивы структур

Запись (комбинированный тип) — структура данных, состоящая из фиксированного числа компонент, называемых полями, каждая из которых может иметь свой тип. При объявлении записи указывают имя и тип каждого поля. Описание записи начинается служебным словом **record** и заканчивается словом **end**.

```
Type
<имя типа> = record
    <имя поля 1> : <тип>;
    <имя поля 2> : <тип>;
    <имя поля 3> : <тип>;
    …
    <имя поля n> : <тип>
    end;
```

Записи находят широкое применение при обработке табличной информации.

Пример 8.10. Пример описания записи о студенте.

```
Const  n=25;
Type
Date = Record
        Year : integer;    {Год}
        Month : 1..12;       {Число}
        Day : 1..31;       {Месяц}
        End;
Student = Record
            FIO : String[15];        {ФИО}
            Dr : Date;         {Дата рождения}
            Str : String[20];        {Страна}
            b1,b2,b3,b4,b5 : 2..5; {Поля оценок}
            End;
```

```
Vec = array[1..n] of Student;
Var I21, M43, A11 :Vec;
```

Порядок следования полей в записи не имеет значения. Для наглядности поля следует располагать на отдельных строках и снабжать комментариями. Запись может содержать вариантную часть.

При обработке записей все действия выполняются над полями записи в соответствии с их типами. Набор операций, процедур и функций определяется типом поля. Для записи в целом может быть использован только оператор присваивания. Запись можно передавать в качестве параметров в процедуру и функцию. Для обращения к полям записи используются составные имена. Составное имя — состоит из имени переменной типа запись и имени поля, разделяемых точкой. Например:

I21[1].FIO :=’Антонов’; M43[1].Dr.Day:=13.

Пример 8.11. Ввод-вывод записи.

```
Type data=record
        ch:1..31;
        m:1..12;
        year:integer;
     end;
var d:data;
begin
writeln('введите дату');
read(d.ch,d.m,d.year);
write('день=',d.ch,' месяц=',d.m,' год=',d.year);
end.
```

Пример 8.12. Напишем программу, проверяющую правильность вводимой даты.

```
function proverka (d:data): boolean; {Функция, проверяющая правильность даты:}
    var f: boolean;
    begin
     f:=true;
     if (d.ch<0)or(d.ch>31)or(d.m<0)or(d.m>12)or(d.year<0)
        then f:=false;
      if (d.m=2) and(d.year mod 2 =0) and (d.ch>29) then f:=false;
      proverka:=f;
    end;
    begin                    {основная программа}
    writeln('введите дату');
    read(d.ch,d.m,d.year);
    write('день=',d.ch,' месяц=',d.m,' год=',d.year);
      if proverka=true then writeln('правильная')
                  else  write('неправильная');
    end.
```

8.3.3. Аудиторные задачи и примеры их решения

Задача 8.1. Требуется написать следующие функции и процедуры:

- функцию сравнения 2 рациональных чисел;
- процедуру вычисления суммы двух рациональных чисел;
- функцию поиска двух одинаковых рациональных чисел в массиве.

```
const n=25;
```

```
Type rac=record
        ch, zn:integer;
        end;
mass=array[1..n] of rac;
function srav (a,b:rac):boolean;        {сравнить 2 рациональных числа}
begin
 if a.ch*b.zn=a.zn*b.ch
   then srav:=true
   else srav:=false;
end;
 procedure summa(a,b:rac; var c:rac);        {сумма 2 рациональных чисел}
begin
 c.zn:=a.zn * b.zn;
 c.ch:=a.ch * b.zn + a.zn * b.ch;
end;
 function poisk(x:mass):boolean;       {поиск 2-х одинаковых рац. чисел}
var i,j:integer;
    fl: boolean;
begin
 fl:=false;
  for i:=1 to n-1 do
   for j:=i+1 to n do
  if x[i].ch*x[j].zn=x[i].zn*x[j].ch then fl:=true;
 poisk:=fl;
end;
```

Задача 8.2. Вывести на экран фамилии всех отличников.

```
Const N=25;
Type anketa=record
        fio: string[15];
            ocenki: record m,f,p:1..5; end;
       end;
 massiv=array [1..N] of anketa;
    var gr:massiv; i:integer;
procedure otlichniki;
begin
  for i:=1 to N do
  if (gr[i].ocenki.m=5)and (gr[i].ocenki.f=5) and (gr[i].ocenki.p=5)
   then writeln(gr[i].fio);
end;
```

Задача 8.3. По номеру телефона найти фамилию человека.

```
Const N=20;
Type znakomui=record
        fio: packed array[1..n]of char;
        telefon:10000..99999;
        end;
 zapis_knigka=array ['A'..'Z'] of znakomui;
 var zp: zapis_knigka;
    i:char;
procedure poisk;
var nomer:10000..99999; b:boolean;
```

```
begin
  b:=false;
  writeln('nomer=');
  read(nomer);
  for i:='A' to 'Z' do
  if (zp[i].telefon=nomer)
   then
    begin
         b:=true;
         writeln(zp[i].fio);
         break;
   end;
 if (b=false) then writeln('нет в книге такого номера');
end.
```

Задача 8.4. Напечатать имя девочки, чей рост равен 120:

```
Type name=(anya,danya,fedya,grisha,masha,ura);
     data=record
         rost:real;
         pol:(mug,gen);
         end;
    gruppa=array[name]of data;
var gr: gruppa; i:name; c:integer;
begin
for i:=anya to ura do
  begin
  writeln('vvedite rost u pol rebenka');
  read(gr[i].rost);
  read(c); if c=1 then gr[i].pol:=gen else gr[i].pol:=mug {c=1 женский пол}
  end;
for i:=anya to ura do
   if (gr[i].rost=120)and (gr[i].pol=gen) then
   begin write('otvet=');
      case i of
      anya: writeln(' anya ');
      danya: writeln('danya');
      fedya:writeln('fedya');
      grisha:writeln(' grisha ');
      masha: writeln(' masha ');
      ura:writeln('ura');
     end;
 end;
end.
```

8.3.4. Задания для самостоятельной аудиторной работы

Задача 8.5. Написать процедуру, печатающую фамилии людей, живущих в городе (название города вводит пользователь).

```
Type anketa=record
       fio:string[15];
         address:record
                 dom:integer;
         gorod,street:string[20];
         end;
```

```
      end;
gruppa=array[1..20] of anketa;
```

Задача 8.6. Написать процедуру **Старший(Gr, Fio)**, присваивающую строке **Fio** фамилию самого старшего мужчины из группы **Gr** (считать, что он есть и он единственный).

```
Type data=record
        ch, m, year:integer;
        end;
anketa=record
      fio, pol:packed array [1..10]of char;
        d_r:data;
      end;
gruppa=array[1..20] of anketa;
```

Задача 8.7. Для структуры из задачи 8.6. написать процедуру **Печ(Gr, B)**, печатающую все фамилии людей из группы **Gr,** начинающиеся с буквы **B**, и даты рождения этих людей.

Задача 8.8. Считая, что на каждой странице записной книжки **Zp** указаны фамилии **Fio**, начинающиеся с одной и той же буквы — индекса этой страницы, описать логическую функцию **Номер (Zp, Fio, NT),** определяющую, есть ли в записной книжке **Zp** сведения о знакомом с фамилией **Fio**, и, если есть, присваивающую параметру **Nt** номер его телефона.

```
Type word_=array [1..9] of char;
        telephone_number=1000000..9999999;
        Friend=record
        Fio : word_;          {ФИО}
        number: telephone_number;    {Номер телефона}
        end;
        Page = array[1..20] of Friend;
        Notebook=array ['A'..'Z'] of Page;
```

Задача 8.9. Написать процедуру **Фамилия(Zp, NT, Fio),** определяющую, есть ли в записной книжке **Zp** сведения о знакомом с номером **Nt**, и, если есть, присваивающую параметру **Fio** фамилию этого знакомого (данные для задачи см. задачу 8.8.).

Задача 8.10. Написать процедуру **Печ(Gr, B)**, печатающую название предмета, сданного лучше всего.

```
Const  n=25;
Type
Student = Record
FIO : String[15];    {ФИО}
Gr : String[20];     {Номер группы}
b1,b2,b3,b4,b5 : 2..5;      {Оценки}
end;
        Vec = array[1..n] of Student;
Var l11:Vec;
```

8.3.5. Задания для домашней работы

Вариант 8.1.

1. Написать логическую функцию **раньше(t1,t2)**, проверяющую, предшествует ли время **t1** времени **t2** (в рамках суток)

```
Type time=record
```

```
            sec, min:1..59;
            ch:0..23;
        end;
    var t1,t2:time;
```

2. Написать функцию **интервал(d,t1,t2)**, вычисляющую время **d**, прошедшее от времени **t1** до **t2: d = t2 - t1** (считаем, что **t2>t1**). Результат в секундах.

```
    Type time=record
            sec, min:1..59;
            ch:0..23;
        end;
    var t1,t2:time; d:integer;
```

3. Написать функцию, вычисляющую средний балл по математике.

```
    Const N=25;
    Type anketa=record
                  fio: string;
                    ocenki: record m,f,p:1..5; end;
                  end;
    massiv=array [1..N] of anketa;
    var gr:massiv;
```

4. Написать процедуру **Ирония судьбы (S)**, которая печатает фамилии двух (любых) жителей из списка **S**, живущих в разных городах по одинаковому адресу.

```
    Const N=25;
    Type address=record
      street: array [1..N] of char;
      house, flat: integer
      end;
    townsman=record
                fio, town; string[N];
                end;
    spisok=array [1..N] of townsman;
    var S:spisok;
```

Вариант 8.2.

1. Написать логическую функцию **раньше(d1,d2)**, проверяющую, предшествует ли дата **d1** дате **d2**.

```
    Type data=record
            ch:1..31;
            m:1..12;
            year:integer;
        end;
    var t1,t2:data;
```

2. Написать функцию **интервал(d, d1,d2),** вычисляющую время **d**, прошедшее от времени **d1** до **d2: d = d2 - d1** (считаем, что **d2>d1**). Результат в днях.

```
    Type data=record
            ch:1..31;
            m:1..12;
            year:integer;
        end;
  var d1, d2:data; d:integer;
```

3. Написать функцию, вычисляющую средний балл по физике.

```
    Const N=25;
```

```
Type anketa=record
                fio; string;
                 ocenki: record m,f,p:1..5; end;
                end;
massiv=array [1..N] of anketa;
var gr:massiv;
```

4. Написать процедуру **Одного_роста (S)**, которая печатает фамилии двух (любых) девушек из списка **S** одного роста.

```
Const N=25;
Type anketa=record
            fio, pol: packed array [1..N] of char;
            rost:real;
        end;
spisok=array [1..N] of anketa;
var S:spisok;
```

9. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №9

Программирование операций над текстовыми и типизированными файлами

9.1. Цели и задачи занятия

Выработать навыки программирования основных операций по обработке текстовых и бинарных файлов

9.2. План занятия

9.2.1. Контроль посещаемости, проверка домашнего задания (10 мин.).
9.2.2. Цель и задачи занятия (5 мин.).
9.2.3. Обсуждение теоретических вопросов (20 мин.):
- физические и логические файлы;
- текстовые файлы и процедуры их обработки;
- типизированные файлы;
- процедуры и функции прямого доступа к типизированным файлам.

9.2.4. Решение аудиторных задач (20 мин.).
9.2.5. Выполнение самостоятельных аудиторной работы (15 мин.).
9.2.6. Выдача домашнего задания (5 мин.).
9.2.7. Подведение итогов занятия (5 мин.).

9.3. Содержание занятия

9.3.1. Краткие сведения о текстовых файлах

Текстовый файл содержит последовательность символов, организо-ванных в строки. Каждая строка файла заканчивается *меткой конец строки.*

Для описания текстовых файлов в языке **Pascal** используется встроенный файловый тип **text**. Прежде чем приступить к операциям над текстовыми файлами, необходимо описать файловую переменную типа **text**: **Var in_file: text;**

Процедура **Assign** связывает имя внешнего файла (файла, который физически расположен на жестком диске) с файловой переменной:

Assign(in_fi1e, 'C:\work\my.txt');

Здесь **my.txt** имя внешнего файла (физический файл), а **in_file** — файловая переменная (логический файл), используемая в качестве параметра процедур работы с файлами.

После установления связи внешнего файла с файловой переменной внешний файл надо открыть для записи или чтения. Текстовый файл **in_file** открывается процедурой **Reset(in_file)** только для чтения, а процедурой **Rewrite(in_file)** — только для записи. После открытия файла процедурой **Rewrite** файл считается пустым. Если файл с таким именем уже существовал, то данные, хранившиеся ранее в этом файле, становятся недоступными. Если требуется открыть существующий файл в режиме записи с возможностью дописывания данных в конец, то используют процедуру **Append (<имя файловой пере-менной>).** По завершении обработки файла программой он должен быть закрыт. После закрытия файла связанный с ним внешний файл обновляется, а файловая переменная может быть связана с другим

внешним файлом. Закрывается файл с помощью процедуры **Close(<имя файловой переменной>)**. К файловым переменным применима стандартная функция **Eof (<файло-вая переменная>),** которая принимает значение **TRUE**, если указатель файла указывает на конец файла, и значение **FALSE** в ином случае.

Для записи значений в файл или чтения из него используются соответственно процедуры:

Write(<файловая переменная>,<список вывода>);
Read(<файловая переменная>,<список ввода>).

Запись признака конца строки в текстовый файл выполняется процедурой:

Writeln(<файловая переменная>).

Для обнаружения признака конца строки при чтении текстового файла используется встроенная функция **Eoln(<файловая переменная>),** прини-мающая значение **TRUE,** если из файла считан символ, за которым стоит признак конца строки, и значение **FALSE** в противном случае. Процедура **Readln (<файловая переменная>,<список ввода>)** пропускает все символы до начала следующей строки текстового файла (т.е. переводит указатель файла на новую строку), а затем присваивает значения в соответствии со списком ввода.

Пример 9.1. Пример чтения переменных **X** и **У из файловой** переменной **in_file: Read(in_file, X, Y);**

Текстовый файл может использоваться для хранения числовых значений. При считывании таких значений из файла или их записи в файл происходит автоматическое преобразование из числового формата в символьный и наоборот.

Пример 9.2. Отобразим содержимое текстового файла 1.txt на экран.

```
var f:text;
 ch:char;
begin
assign(f, '1.txt'); reset(f); {открытие файла в режиме чтения}
  while not eof(f) do        {пока не конец файла}
   begin
      while not eoln(f) do  {пока не конец строки}
              begin
                read(f, ch);
                write(ch);
              end;
          readln(f);  writeln;
  end; end.
```

9.3.2. Краткие сведения о типизированных файлах

Типизированные файлы предназначены для хранения данных определенных типов. Для описания типизированный файлов в языке **Pascal** используются служебные слова **file of.**

Пример 9.3. Пример описания типизированного файла:

Var f: file of <тип данных>,

где **<тип данных> — тип элементов, содержащихся в файле**. Также для связывания используют процедуру **Assign, a** процедуры **Reset(f)** и **Rewrite(f)** — аналогично открывают файл **f** для чтения и записи, процедура **close(f)** — закрывает файл **f**.

Необходимо только помнить при работе с типизированными файлами должна существовать программа не только считывающая из этого файла данные, но и (обязательно) должна быть программа, записывающая эти данные в файл.

Пример 9.4. Необходимо написать функцию, подсчитывающую сумму отрицательных элементов в типизированном (бинарном) файле.

```
type seriya=file of real;
var f:seriya; { файловая переменная}
   r:real;      {число}
procedure zap_file(name:string);      {процедура записи чисел в файл}
begin                                 {параметром передаем имя файла}
 assign(f,name);
 rewrite(f);
writeln('введите числа,  999-конец ввода');
                {пока не ввели число=999 считываем числа с клавиатуры
                и записываем их в файл}
 while r<>999 do
   begin
    read(r);  write(f,r);
   end;
close(f);
end;
{функция чтения чисел из файла и подсчета суммы отрицательных}
function summa(name:string):real;
var s:real;
begin
 assign(f,name);  reset(f);
 s:=0;
 while not eof(f) do
 begin
  read(f,r);
  if r<0 then s:=s+r;
 end;
summa:=s
end;
{основная программа}
var name:string;
begin
 write('введите имя файла');   {чтение имени файла}
 readln(name);
 zap_file(name);              {запись в файл чисел}
write('s=',summa(name):10:5); {вызов функции и печать результата}
end.
```

9.3.3. Аудиторные задачи и примеры их решения

Задача 9.1. В каждой строке текстового файла содержится одно предложение. Длина строки не более 255 символов. Требуется написать программу, которая считывает из текстового файла три предложения и выводит их в обратном порядке.

```
var s:string;
i,j:integer; f:text;
begin
assign(f,'1.txt');
```

```
reset(f);
i:=0; j:=1;
while not eof(f) do
  begin
   readln(f,s);
   i:=i+1;
   if(i>3) then break;
   for j:=length(s) downto 1 do write(s[j]);
   writeln;
  end;
writeln('end');
close(f);
end.
```

Задача 9.2. Требуется написать программу, которая считает сколько раз в каждой строке файла встречается буква **x**.

```
var s:char;
i,j,k:integer; f:text;
begin
assign(f,'1.txt');
reset(f);
i:=0; {количество строк в файле}
while not eof(f) do
  begin
  k:=0; {количество букв ‘x’}
  while not eoln(f) do
    begin
    read(f,s);
    if s='x' then inc(k);
    end;
   readln(f);
  i:=i+1;
  writeln(i,'stroka k=',k);
  end;
  close(f);
end.
```

Задача 9.3. . Известно, что в каждой строке файла есть, по крайней мере, один восклицательный знак. Найти количество пробелов (для каждой строки), предшествующих первому восклицательному знаку.

```
var s:char;
i,j,k:integer; f:text;
begin
assign(f,'1.txt');
reset(f);
i:=0;
while not eof(f) do
  begin
  k:=0;
  while not eoln(f) do
    begin
    read(f,s);
    if s=' ' then inc(k);
```

```
      if s='!' then break;
      end;
    readln(f);
   i:=i+1;
   writeln(i,'stroka k=',k);
   end;
   close(f);
end.
```

Задача 9.4. Написать процедуру, которая подсчитывает, сколько раз в каждой строке файла встречается символ ‘+’ и сколько раз символ ‘*’. А также написать программу, вызывающую эту процедуру для каждой строки текстового файла.

Задача 9.5.

```
type cost=record  grn:0..maxint; kop:0..99 end;
     price=file of cost; var s:cost;  f:price;
```

Описать процедуру нахождения наименьшей цены из прайса **f**.

```
{процедура заполнения бинарного файла}
procedure vvod;
begin
assign(f,'f.dat');
rewrite(f);
while true do
 begin
   writeln('grn->(999-exit)'); read(s.grn);
   if s.grn=999 then break;
   writeln('kop');
   read(s.kop);
   write(f,s);
 end;
close(f);
end;
{процедура нахождения наименьшей цены}
 procedure min_cost;
var min:cost;
begin
assign(f,'f.dat');
reset(f);
 read(f,min);
while not eof(f) do
   begin
     read(f,s);
     if (s.grn<min.grn)or(s.grn=min.grn)and(s.kop<min.kop) then min:=s;
   end;
writeln('min_cost=', min.grn,'.', min.kop);
close(f);
end;
{основная программа}
 begin
vvod;
min_cost;
end.
```

Задача 9.6. type ряд= file of 0..999;

Описать логическую функцию, проверяющую, упорядочены ли по возрастанию элементы непустого ряда **r**.

Задача 9.7. type текст= file of char;

Описать логическую функцию, проверяющую тексты **t1** и **t2** на равенство.

Задача 9.8. type текст= file of char;

Описать процедуру, удаляющую из текста **t** все литеры **‘+’** и **‘-’**.

9.3.4. Задания для самостоятельной аудиторной работы

Задача 9.9. Преобразовать каждую строку файла, заменив все восклицательные знаки точками.

```
var s:char;
f1,f2:text;
begin
assign(f1,'1.txt');
assign(f2,'2.txt');
reset(f1);
rewrite(f2);
while not eof(f1) do
  begin
    while not eoln(f1) do
     begin
     read(f1,s);
     if s='!' then write(f2,'.')
              else write(f2,s);
    end;
   readln(f1);   writeln(f2);
  end;
close(f1);
close(f2);
end.
```

Задача 9.10. Требуется преобразовать файл, заменив любую последовательность точек одним символом ‘*’. Например исходный текст: "9.00…1……27.3.4…"; результат: "9*00*1*27*3*4*".

Задача 9.11. Написать процедуру, которая подсчитывает общее число вхождений символов ‘+’ , ‘-‘, ‘*’ в строке. А также написать программу вызывающую эту процедуру для каждой строки текстового файла.

Задача 9.12. type дата= record месяц: 1..12; число: 1..30 end;

ФД=file of дата;

Описать процедуру от трёх файлов типа ФД, которая из первого файла переписывает во второй файл все зимние даты, а в третий файл — все летние даты.

Задача 9.13. type человек= record имя: string[15]; возраст: 1..99 end;

группа=file of человек;

Описать процедуру печати имен всех людей из непустой группы **ГР**, имеющих наименьший возраст.

9.3.5. Задания для домашней работы

Вариант 9.1.

1. Известно, что в каждой строке файла есть, по крайней мере, одна запятая. Написать функцию поиска в строке порядкового номера первой слева запятой. Написать программу, вызывающую эту функцию для каждой строки текстового файла.

2. Дан текстовый файл, известно, что в каждой строке имеются пары символов ‘(‘ и ‘)’. Необходимо преобразовать данный файл, исключив то, что находится между ‘(‘ и ‘)’. Предполагается, что внутри каждой пары скобок нет других скобок.

3. Написать функцию, определяющую число вхождений группы символов “abc” в строку. Написать программу, вызывающую эту функцию для каждой строки текстового файла.

4. **type текст= file of char;**

Описать процедуру, удваивающую в тексте **t** каждую цифру.

Вариант 9.2.

1. Известно, что в каждой строке файла есть, по крайней мере, одна запятая. Написать функцию поиска в строке порядкового номера последней запятой в строке. Написать программу, вызывающую эту функцию для каждой строки текстового файла.

2. Дан текстовый файл, известно, что в каждой строке имеются пары символов ‘(‘ и ‘)’. Необходимо преобразовать данный файл, оставив только то, что находится между ‘(‘ и ‘)’. Предполагается, что внутри каждой пары скобок нет других скобок.

3. Написать функцию, определяющую число вхождений группы символов “gkl” в строку. Написать программу, вызывающую эту функцию для каждой строки текстового файла.

4. **type текст= file of char;** описать процедуру, заменяющую в тексте **t** каждую цифру на следующую по величине цифру (‘9’ заменять на ‘0’).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алексеев Ю.Е. Практикум по программированию: обработка числовых данных : учебное пособие / Под ред. Б.Г. Трусова.— М.: Изд-во МВТУ им. Н.Э. Баумана,2008. —288с.

2. Ковалюк Т.В. Основи программування / Т.В. Ковалюк — К.: Видавнича група BHV,2005.—384 с.

3. Вирт Н. Алгоритмы и структуры данных / Н. Вирт. — М.:Мир,1989. —360с

4. Павловская Т. А. Паскаль. Программирование на языке высокого уровня : практикум / Т. А. Павловская. — СПб. : Питер, 2006. — 317 с.

5. Павловская Т. А. Паскаль. Программирование на языке высокого уровня : учеб. для вузов / Т. А. Павловская. — СПб. : Питер, 2008. — 393 с.

6. Пильщиков В.Н. Сборник упражнений по языку Паскаль / В.Н. Пильщиков. — М.:Наука,1989. — 160с.

ПРИЛОЖЕНИЕ А

Таблица А.1. — Варианты РГЗ №1

Номер варианта	Задания										
	1а	1б	2а	2б	3а	3б	4а	4б	5а	5б	6а
1	10000001	1000,0001	246	246,02	88	A1,1	25	231	25	231	25
2	10000010	1000,0101	241	521,32	99	1A,1	26	233	26	233	26
3	10000011	1001,0011	242	522,64	AA	1B,2	27	234	27	234	27
4	10000100	1000,0101	243	523,07	BB	2B,3	28	235	28	235	28
5	10000101	1001,0101	244	340,32	CC	4C,C	29	341	29	341	29
6	10000110	1001,0111	245	341,64	DD	5D,C	30	342	30	342	30
7	10000111	1000,0111	246	343,04	EE	3A,3	31	441	31	441	31
8	10010001	1111,0001	247	250,64	FF	81,7	32	401	32	401	32
9	10010000	1010,1001	260	520,32	81	A1,B	33	402	33	402	33
10	10010010	1001,0011	341	521,32	82	2A,A	34	403	34	403	34
11	10010011	1000,0011	342	251,16	83	4B,1	35	481	35	481	35
12	10010110	1001,0011	343	252,64	84	1D,C	36	408	36	408	36
13	10010101	1001,0101	344	255,07	85	1C,D	37	482	37	482	37
14	10011001	1001,1001	345	541,16	86	1B,7	38	493	38	493	38
15	10011010	1001,1111	346	254,07	87	B7,1	39	182	39	182	39
16	10011011	1001,1011	347	241,64	90	D1,C	40	821	40	821	40
17	10011111	1101,1111	361	422,32	91	D2,A	41	394	41	394	41
18	10011110	1100,1111	362	201,32	A9	17,B	42	324	42	324	42
19	10101011	1010,1011	351	203,64	92	A8,C	43	351	43	351	43
20	10110011	1011,0011	221	206,32	93	8A,C	44	352	44	352	44
21	10000001	1000,0001	246	246,02	88	A1,1	25	231	25	231	25
22	10000010	1000,0101	241	521,32	99	1A,1	26	233	26	233	26
23	10000011	1001,0011	242	522,64	AA	1B,2	27	234	27	234	27
24	10000100	1000,0101	243	523,07	BB	2B,3	28	235	28	235	28
25	10000101	1001,0101	244	340,32	CC	4C,C	29	341	29	341	29
26	10000110	1001,0111	245	341,64	DD	5D,C	30	342	30	342	30
27	10000111	1000,0111	246	343,04	EE	3A,3	31	441	31	441	31
28	10010001	1111,0001	247	250,64	FF	81,7	32	401	32	401	32
29	10010000	1010,1001	260	520,32	81	A1,B	33	402	33	402	33
30	10010010	1001,0011	341	521,32	82	2A,A	34	403	34	403	34

Продолжение таблицы А.1.

Номер варианта	Задания										
	6б	7а	7б	8а	8б	8б	9а	9б	10а	10б	11а
1	231	88	A115	24	246,02	246,02	25,25	231,125	25,25	231,125	25,25
2	233	99	1A51	41	521,32	521,32	26,45	233,175	26,45	233,175	26,45
3	234	AA	1B52	42	522,64	522,64	27,75	234,135	27,75	234,135	27,75
4	235	BB	2B53	43	523,64	523,64	28,35	235,265	28,35	235,265	28,35
5	341	CC	4CC5	44	340,32	340,32	29,65	341,345	29,65	341,345	29,65
6	342	DD	5DC5	45	341,64	341,64	30,25	342,845	30,25	342,845	30,25
7	441	EE	3A35	46	343,04	343,04	31,35	441,625	31,35	441,625	31,35
8	401	FF	817A	47	250,64	250,64	32,35	401,145	32,35	401,145	32,35
9	402	81	A11B	60	520,32	520,32	33,35	402,155	33,35	402,155	33,35
10	403	82	2A2A	34	521,32	521,32	34,35	403,135	34,35	403,135	34,35
11	481	83	4B11	32	251,16	251,16	35,35	481,325	35,35	481,325	35,35
12	408	84	1D1C	33	252,64	252,64	36,15	408,175	36,15	408,175	36,15
13	482	85	1C5D	31	255,64	255,64	37,25	482,185	37,25	482,185	37,25
14	493	86	1B57	35	541,16	541,16	38,35	493,135	38,35	493,135	38,35
15	182	87	B751	36	254,16	254,16	39,45	182,125	39,45	182,125	39,45
16	821	90	D15C	37	241,64	241,64	40,25	821,145	40,25	821,145	40,25
17	394	91	D25A	61	422,32	422,32	41,55	394,235	41,55	394,235	41,55
18	324	A9	175B	62	201,32	201,32	42,65	324,215	42,65	324,215	42,65
19	351	92	A84C	51	203,64	203,64	43,45	351,365	43,45	351,365	43,45
20	352	93	8A5C	52	206,32	206,32	44,45	352,245	44,45	352,245	44,45
21	231	88	A115	24	246,02	246,02	25,25	231,125	25,25	231,125	25,25
22	233	99	1A51	41	521,32	521,32	26,45	233,175	26,45	233,175	26,45
23	234	AA	1B52	42	522,64	522,64	27,75	234,135	27,75	234,135	27,75
24	235	BB	2B53	43	523,64	523,64	28,35	235,265	28,35	235,265	28,35
25	341	CC	4CC5	44	340,32	340,32	29,65	341,345	29,65	341,345	29,65
26	342	DD	5DC5	45	341,64	341,64	30,25	342,845	30,25	342,845	30,25
27	441	EE	3A35	46	343,04	343,04	31,35	441,625	31,35	441,625	31,35
28	401	FF	817A	47	250,64	250,64	32,35	401,145	32,35	401,145	32,35
29	402	81	A11B	60	520,32	520,32	33,35	402,155	33,35	402,155	33,35
30	403	82	2A2A	34	521,32	521,32	34,35	403,135	34,35	403,135	34,35

Продолжение таблицы А.1.

Номер варианта	Задания												
	11б	12а		12б		13а		13б		14		15	
1	231,125	1101	100	110101	1001	21	10	430	25	20	20	20	20
2	233,175	1010	101	101001	1010	22	11	431	20	21	21	21	21
3	234,135	1001	110	100111	1011	23	12	432	30	23	23	23	23
4	235,265	1011	111	101110	1100	24	13	433	35	25	25	25	25
5	341,345	1101	100	110100	1110	25	14	434	40	26	26	26	26
6	342,845	1001	101	100101	1111	26	15	435	45	30	30	30	30
7	441,625	1010	110	101011	1000	30	10	541	25	31	31	31	31
8	401,145	1011	111	101101	1001	31	11	542	20	32	32	32	32
9	402,155	1100	100	110010	1010	32	12	543	30	33	33	33	33
10	403,135	1101	101	110100	1011	33	13	544	35	34	34	34	34
11	481,325	1110	110	111011	1100	34	14	545	40	35	35	35	35
12	408,175	1001	111	100101	1101	35	15	546	45	36	36	36	36
13	482,185	1010	100	101010	1110	36	10	550	25	40	40	40	40
14	493,135	1011	101	101111	1111	40	11	511	20	41	41	41	41
15	182,125	1100	110	110011	1000	41	12	512	30	42	42	42	42
16	821,145	1101	111	110101	1001	42	13	513	35	43	43	43	43
17	394,235	1110	100	111011	1010	43	14	514	40	44	44	44	44
18	324,215	1111	101	111110	1011	44	15	521	45	45	45	45	45
19	351,365	1001	110	100110	1100	45	10	515	20	46	46	46	46
20	352,245	1101	111	110111	1011	46	11	520	30	50	50	50	50
21	231,125	1101	100	110101	1001	21	10	430	25	20	20	20	20
22	233,175	1010	101	101001	1010	22	11	431	20	21	21	21	21
23	234,135	1001	110	100111	1011	23	12	432	30	23	23	23	23
24	235,265	1011	111	101110	1100	24	13	433	35	25	25	25	25
25	341,345	1101	100	110100	1110	25	14	434	40	26	26	26	26
26	342,845	1001	101	100101	1111	26	15	435	45	30	30	30	30
27	441,625	1010	110	101011	1000	30	10	541	25	31	31	31	31
28	401,145	1011	111	101101	1001	31	11	542	20	32	32	32	32
29	402,155	1100	100	110010	1010	32	12	543	30	33	33	33	33
30	403,135	1101	101	110100	1011	33	13	544	35	34	34	34	34

Таблица А.2. — Варианты РГЗ №2

номер	число 1а	число 1б	число 2а	число 2б
1	1234	1234	1234,25	1234,25
2	1335	1335	1335,26	1335,26
3	1436	1436	1436,27	1436,27
4	1537	1537	1537,28	1537,28
5	1638	1638	1638,29	1638,29
6	1739	1739	1739,31	1739,31
7	1840	1840	1840,31	1840,31
8	1941	1941	1941,32	1941,32
9	2042	2042	2042,33	2042,33
10	2143	2143	2143,34	2143,34
11	2244	2244	2244,35	2244,35
12	2345	2345	2345,36	2345,36
13	2446	2446	2446,37	2446,37
14	2547	2547	2547,38	2547,38
15	2648	2648	2648,39	2648,39
16	2749	2749	2749,41	2749,41
17	2850	2850	2850,41	2850,41
18	2951	2951	2951,41	2951,41
19	3052	3052	3052,42	3052,42
20	3153	3153	3153,43	3153,43
21	3254	3254	3254,44	3254,44
22	3355	3355	3355,45	3355,45
23	3456	3456	3456,46	3456,46
24	3557	3557	3557,47	3557,47
25	3658	3658	3658,48	3658,48
26	3759	3759	3759,49	3759,49
27	3860	3860	3860,51	3860,51
28	3961	3961	3961,51	3961,51
29	4062	4062	4062,52	4062,52
30	4163	4163	4163,53	4163,53