Вапнічний С.Д., Зубик В.В., Ребрина В.А.

Факультативний курс з інформатики «Основи програмування. Pascal. Перший рік навчання»

для учнів 7-9 класів
загальноосвітніх навчальних закладів

Схвалено для використання в загальноосвітніх навчальних закладах області
Науково-методичною Радою Хмельницького ОІППО
(протокол № 2 від 22 червня 2010 року)

Хмельницький, 2010

1) Розподіл навчального часу

№ з/п	*Тема*	*Кількість годин*
1	Основні поняття мови програмування	8
2	Базові структури мови програмування: розгалуження	8
3	Базові структури мови програмування: цикли	8
4	Підпрограми.	4
5	Функції.	4
6	Масиви.	12
7	Масиви символів, рядкові величини	8
8	Рекурсія	4
9	Робота з файлами даних	4
10	Знайомство із структурою даних: стек, черга, списки, множина.	8

2) Орієнтовне календарно-тематичне планування курсу

№ заняття	*Дата*	*Тема та зміст заняття*
1		Основні поняття мови програмування
2		Основні поняття мови програмування
3		Практична робота № 1
4		Аналіз та корекція
5		Основні поняття мови програмування
6		Основні поняття мови програмування
7		Практична робота № 2
8		Аналіз та корекція
9		Базові структури мови програмування
10		Базові структури мови програмування
11		Практична робота № 3
12		Аналіз та корекція
13		Базові структури мови програмування
14		Базові структури мови програмування

15	Практична робота № 4
16	Аналіз та корекція
17	Організація циклів
18	Організація циклів
19	Практична робота № 5
20	Аналіз та корекція
21	Організація циклів
22	Організація циклів
23	Практична робота № 6
24	Аналіз та корекція
25	Підпрограми. Формальні та фактичні параметри
26	Підпрограми. Формальні та фактичні параметри
27	Практична робота № 7
28	Аналіз та корекція
29	Функції
30	Функції
31	Практична робота № 8
32	Аналіз та корекція
33	Масиви
34	Масиви
35	Практична робота № 9
36	Аналіз та корекція
37	Масиви
38	Масиви
39	Практична робота № 10
40	Аналіз та корекція
41	Масиви
42	Масиви
43	Практична робота № 11

44	Аналіз та корекція
45	Масиви символів, рядкові величини
46	Масиви символів, рядкові величини
47	Практична робота № 12
48	Аналіз та корекція
49	Масиви символів, рядкові величини
50	Масиви символів, рядкові величини
51	Практична робота № 13
52	Аналіз та корекція
53	Рекурсивні функції.
54	Рекурсивні функції.
55	Практична робота № 15
56	Аналіз та корекція
57	Робота з файлами даних
58	Робота з файлами даних
59	Практична робота № 16
60	Аналіз та корекція
61	Структури даних
62	Структури даних
63	Практична робота № 17
64	Аналіз та корекція
65	Структури даних.
66	Структури даних.
67	Практична робота № 18
68	Аналіз та корекція.
	Підсумок роботи факультативу

Розділ 1. Основні поняття мови програмування.

Перші відомості про мову Паскаль було надруковано у 1971 році. Його автор — професор Інституту інформатики Швейцарської вищої політехничної школи Ніклаус Вірт. Мову названо на честь видатного французького фізика, математика та філософа Блеза Паскаля. Своїй величезній популярності у наші часи Паскаль зобов'язаний американській фірмі BORLAND, яка розробила діалект мови Турбо Паскаль, додавши до стандартного Паскаля додаткові можливості і розробивши дуже зручну та швидку програму-транслятор. Оскільки мікропроцесор сприймає команди, що надходять на його вхід, виключно у цифровому вигляді, а алгоритм вирішення будь-якої задачі записується мовою програмування у вигляді команд зі звичайних слів "людською" мовою, необхідно мати перекладач (транслятор) з такої мови на мову машинних цифрових кодів. Таку роботу й виконує програма-транслятор, яка здійснює, окрім цього, ще й перевірку правильності написання програми.

Абетка (алфавіт)

Абетка (алфавіт) — це набір символів, які використовуються для написання програм. Абетку мови Паскаль складають:

1) Літери A..Z, a..z та символ підкреслювання "_".
2) Цифри від 0 до 9.
3) Спеціальні символи + -*/ = <>(){}[].,:;#$'".
4) Службові слова:

and - і, та
array - масив
begin - початок
case - варіант
const - константа
div - ділення
do - виконати
downto-зменшуючи до
else - інакше
end - кінець
file - файл
for-для
function - функція
goto -перейтидо
if-якщо
in-в
label - мітка
mod - мод, остача
nil - нуль, нічого
not - не
of-з
or - або, чи
packed - упакований
procedure - процедура
program - програма
record - запис
repeat - повторювати
set - множина
then - то
to - до (збільшуючи до)
type-тип
until - доти поки
var - змінна
while - поки
with - з

5) Деякі послідовності з двох спеціальних символів:

: = надати значення
>= більше або дорівнює
<= менше або дорівнює
<> не дорівнює
.. роздільник діапазонів
(* або { початок коментарію
*) або} кінець коментарію

Програма мовою Паскаль оперує різноманітними об'єктами, що містять або певні дані (константи, змінні, типи, файли), або виконують певні дії (процедури та функції). Кожний об'єкт позначається іменем (ідентифікатором). Іменем може бути будь-яка послідовність латинських літер, цифр, знаку підкреслювання, що починається з літери чи знаку підкреслювання (пробіли в іменах ставити заборонено). Заборонено в якості імен використовувати службові слова. Довжина ідентифікатора не повинна перевищувати 127 символів.

Приклади ідентифікаторів:

Правильно	Неправильно
SQUARE	3RDROOT
Persons_Counted	Two words
_number_box	-Massa

Великі та малі літери в іменах транслятор не відрізняє. Сполучення цих літер використовується для того, аби зручніше було читати інформацію. Числа можуть бути цілими (тип integer) та дійсними (тип real). Дійсні числа записуються у 2-х формах: натуральній (з фіксованою крапкою) та експоненціальній (з плаваючою крапкою).

Приклади:
1) 0; 12; 6; -146 - цілі;
2) 1.2; -0.4; 5.13 - дійсні з фіксованою крапкою;
3) 10.43E-02; 1e20 - дійсні з плаваючою крапкою.

Записи останнього прикладу відповідають науковій формі запису чисел. Літера E (чи e) означає "помножити на 10 в степені". Тому число 1e20 читається так: "одиниця, помножена на 10 в степені 20".

Структура програм

Програма мовою Паскаль складається з заголовка, опису даних та тіла програми, що являє собою блок команд обробки даних, обмежений словами BEGIN (початок) та END (кінець). Закінчується програма крапкою.

```
PROGRAM ім'я;
Var  опис даних;
  BEGIN
     оператор;      { тіло програми  }
     ……….
     оператор;
  END.
```

Приклад програми, що обчислює площу круга та довжину кола:

```
Program circle;                          {заголовок}
const pi=3.141593;                       {константа}
var r ,len,s : real;                     {змінні}
  BEGIN
     readln(r);                          {ввести радіус}
     len:=2 * pi * r;                    {обчислити довжину кола}
     s:=p * r * r;                       {обчислити площу}
     writeln(s)                          {вивести результати}
  END.
```

У фігурних дужках записуються коментарі (пояснення до програми), які транслятором мови ігноруються. Заголовок програми і кожен опис повинні закінчуватися крапкою з комою. Команди (оператори) у тілі програми повинні відокремлюватися одна від одної крапкою з комою, навіть якщо вони записуються окремими рядками. Окремі змістові блоки програми прийнято записувати з відступом ("сходинками"), що робить програму більш зручною для читання, хоча транслятору це байдуже, навіть якщо б ви записали всю програму одним рядком. Перелік усіх даних із зазначенням їх типів на початку програми не тільки полегшує контроль за їх використанням, але й дозволяє транслятору знаходити та повідомляти вам про помилки, наявні в тілі програми та пов'язані з невірною обробкою даних.

Стандартні типи даних

Програма оперує різними елементами даних. Кожен елемент може бути визначеного типу. Цей тип задає як множину значень, що може приймати елемент, так і операції, які можуть бути з ним виконані.

Зараз ми розглянемо лише чотири стандартних типи даних:

1) **INTEGER** (цілий). Діапазон значень даних цього типу, що можуть прийматися, зазвичай від -32768 до +32767. Граничні значення діапазону записано в стандартних константах, які мають імена MAXINT та MININT, тобто MININT -32768 та MAXINT=32767. Спроба отримати значення, що виходить за межі діапазону, призводить до помилки транслятора.

Об'єкти даних — учасники операцій — називаються операндами. Над операндами цілого типу можливі такі операції: + (додавання), - (віднімання), * (множення), DIV (цілочисельне ділення з відсіченням залишку), MOD (залишок від цілочисельного ділення). Ці операції виконуються точно. Порядок виконання звичайний: спочатку множення, ділення та отримання залишку, потім додавання та віднімання. Службові слова DIV і MOD з двох сторін повинні відокремлюватися пробілами.

Приклади:
5 div 2 = 2
5 mod 2 = 1
5 div 6 = 0
5 mod 6 = 5

2) **REAL** (дійсний). Дані цього типу зберігають цілу, дробову частини та порядок (степінь числа 10). Цілі числа в операціях із дійсними даними автоматично приводяться до REAL-формату. Абсолютна величина (модуль) дійсних чисел зазвичай знаходиться в діапазоні від 1E-38 до 1E+38. Якщо число по модулю менше, ніж 1E-38, відбувається втрата значущості (перетворення на нуль), а якщо більше 1E+38, то виникає помилка (переповнення). До дійсних операндів можна застосовувати операції + (додавання), - (віднімання), * множення), / (ділення). В результаті операції також виходить об'єкт дійсного типу. Якщо один із операндів типу REAL, то й результат також типу REAL, навіть якщо інші операнди типу INTEGER.

Приклад:
24.865
-0.36
0.2485E+2
- 0.36E0

3) **CHAR** (символьний тип). Значенням символьної величини є один символ (літера, цифра і т.ін.). Всі символи занесено в спеціальну таблицю у певному порядку. Порядковий номер символу є кодом цього символу. Всього в таблиці 256 символів (порядкові номери від 0 до 255). Значення символьного типу записується у вигляді символу, взятого в одинарні лапки (апострофи). Для того, щоб представити сам апостроф, його потрібно повторити два рази.

Приклади:
‘A’
‘Ь’
‘7’
‘.’
‘;’
‘’’’ – так може бути представлений сам апостроф

Коди великих та малих літер — різні. Оскільки символи мають порядкові номери (коди), то до них можна застосовувати лише операції порівняння одного з іншим.

Наприклад, латинські літери розташовано в таблиці кодів в алфавітному порядку, тобто символ ‘A’ має мінімальний код, а ‘Z’ — максимальний (в ланцюжку великих латинських літер). Тому можна записати:

‘A’ < ‘F’
‘Y’ > ‘C’
‘E’ < > ‘G’

4) **BOOLEAN** (булевий або логічний тип). Величини цього типу приймають одне з двох значень: TRUE (істина) чи FALSE (хибність). Слова TRUE і FALSE є булеві константи. До булевих операндів можна застосовувати такі операції: AND (логічне І), OR (логічне АБО), NOT (логічне НЕ). Якщо A, B, C — булеві змінні, то мовою Паскаль булевими виразами будуть**,** наприклад:
A and B
B and (C or A) and A

Результатом операції AND буде істина (TRUE) тільки у тому випадку, коли обидва операнди мають значення TRUE, інакше результатом буде хибність (FALSE). Результатом операції OR буде Істина (TRUE), якщо хоча б один із операндів мас значення TRUE (АБО один, АБО інший, АБО разом), інакше результатом буде хибність (FALSE). Результатом операції NOT (НЕ) завжди є протилежна величина (якщо A є TRUE, то NOT A є FALSE). Якщо вираз не має дужок, порядок виконання операцій такий: спочатку NOT, потім AND і, наприкінці, OR. Крім булевих операцій, булеві значення мають операції відношення: =, <>, >, <, >=, <=

Наприклад:
3.2<5.1 є TRUE
7=8 є FALSE
‘D’ >‘A’ є TRUE

Стандартні функції

Більшість інших операцій по обробці величин виконують стандартні функції. Функція має ім'я (ідентифікатор), за яким вказано один чи більше аргументів. При зверненні до функції викликається готова вбудована в транслятор програма обробки, якій, в якості вхідних величин, передаються аргументи функції. Результат повертається в головну програму через ім'я функції.
Ось список основних функцій:
abs(x) — абсолютна величина (модуль) x. Аргумент x — цілий або дійсний. Результат — відповідно цілий або дійсний.

Приклад:

Математика	Pascal
$\frac{a+b}{\lvert c-d\rvert}$	(a+b)/ abs(c-d)

sqr(x) — число в квадраті. Аргумент x — цілий або дійсний. Результат — відповідно цілий або дійсний.

Приклад:

Математика	Pascal
ax^2+b	a*sqr(x)+b

sqrt(x) - обчислення кореня квадратного. Аргумент і результат – дійсні; x>=0

Приклад:

Математика	Pascal
$\sqrt{x^2+1}$	sqrt(sqr(x)+1)

trunc(x) — відкидає дробову частину x. Аргумент — дійсний, результат — цілий.
Якщо y:=trunc(3.5) то значення змінної y буде дорівнювати 3.
round(x) — округлення x до найближчого цілого. Аргумент — дійсний, результат — цілий.
Якщо y:=round(3.5) то значення змінної y буде дорівнювати 4.
ord(x) — дає код символьної величини x.

Приклад:
ord('B') = 66

chr(x) - дає символ, код якого дорівнює цілому x.

Приклад:
chr(66)=B

Вирази

Вирази складаються з операцій, припустимих для величин, що використовуються. Послідовність виконання операцій така:
1) операції в круглих дужках;
2) функції;
3) not;
4) *, /, div, mod, and (операції типу множення);
5) +, -, or (операції типу додавання);
6) =, >, <, <=, >= (операції типу відношення).

Операції одного старшинства виконуються послідовно зліва направо. Згідно з цим, вираз (a=b) or (c=d) не еквівалентний a=b or c=d.

Тому дуже уважно стежте за старшинством операцій і необхідним порядком їх виконання. Особливість складається в тому, що всі вирази записуються одним рядком. Таким чином, складні дроби потрібно записувати так: спочатку обчислюється чисельник, потім — знаменник і, нарешті, сам дріб.

Оператор присвоювання

В математиці часто доводиться мати справу з абстрактними змінними, тобто рахується, що в деякому виразі змінні не мають якогось конкретного значення або ж можуть мати будь-які значення. В програмуванні це зовсім інакше.

При об'яві змінної певного типу на початку програми ЕОМ відводить відповідну ділянку пам'яті, в якій може розміститися значення вказаного типу. Цій ділянці пам'яті надається ім'я, однакове з іменем змінної. Надалі в ці комірки пам'яті може бути розташовано будь-яке допустиме значення. Якщо далі транслятор зустрічає в програмі ім'я вже об'явленої змінної, він автоматично звертається до ділянки пам'яті з цим самим іменем для того, щоб або розмістити там, або взяти звідти потрібне значення.

Один з основних засобів для того щоб розмістити в комірці змінної якесь значення — це використати оператор присвоювання:

змінна := вираз

Тут символи ":=" — знаки оператора присвоювання. Послідовність виконання така: спочатку обчислюється значення виразу в правій частині оператора; це значення розміщується потім в комірку пам'яті змінної, ім'я якої вказано зліва. Як частковий випадок, виразом може бути одна константа чи змінна.

Приклади:

a:=3	- а присвоїти 3, тобто в а розмістити значення 3;
y:=cos(x)+sin(x)	- у присвоїти значення виразу;
b:=c	- в комірку b копіюється значення з c;
x:=x+1	- до значення x додається 1, отриманий результат знову розміщується в комірку x.

Тип змінної зліва повинний бути таким самим, як і тип значення виразу справа, за винятком, коли змінна може бути REAL, а вираз — INTEGER.

Оператор введення

Якщо програміст бажає записати значення в комірку змінної не під час написання програми, а в процесі її виконання в режимі діалогу (тобто та людина, що працює з програмою, сама може ввести те значення змінної, яке забажає), потрібно користуватися оператором read (читати): read (список змінних);

Наприклад:
read(x)
read(a,b,c)

При цьому виконання програми призупиняється й ЕОМ очікує, поки користувач не введе з клавіатури відповідне значення. При натисканні на клавішу "Enter" значення розташовується в комірці змінної, вказаної в дужках оператора read. Якщо в операторі read указано декілька змінних, потрібно при введенні значень або після набору кожного значення натискати "Enter", або набрати всі значення відразу, розділюючи їх пробілами, після чого натиснути клавішу "Enter". Якщо потрібно, щоб після введення останнього значення зі списку змінних оператора read курсор перейшов на початок наступного рядка (для зручності подальшого діалога), використовують різновид оператора read у вигляді:

Readln (список змінних);

Оператор виведення

Для виведення значень змінних на екран монітора користуються оператором write (писати) та його різновидом writeln у такому самому форматі:

write(список змінних);
writeln (список змінних);

Аналогічно, після закінчення роботи оператора writeln, курсор переводиться на початок наступного рядка.

Наприклад, якщо a=5, b=7,c=1.4E-19, d=106,
то оператори *write(a); writeln(b,c); write(d);*
виведуть на екран значення в такому вигляді:
571.4E-19
106
причому курсор залишається відразу за числом 106 і наступний оператор write або writeln буде продовжувати виведення в цьому самому рядку.
Звертаю увагу на те, що значення змінних a, b, c злилися і однозначно визначити що, і кому належить досить проблематично. Щоб такого не відбувалося можна виводити між змінними пропуски.

Наприклад для нашого випадку
оператори *write(a,' '); writeln(b,' ',c); write(d);*

виведуть на екран значення в такому вигляді:
5 7 1.4E-19
106

Виводити можна не тільки значення змінних, а й значення виразів:
*write(sin(2*a*b));*

Для величин, що виводяться оператором write (writeln), можна зазначати формат виведення:
1) для не REAL величин у вигляді:
Writeln(величина:n)
2) для REAL величин у вигляді:
Writeln(величина:n:m)

Тут ціле число n визначає кількість позицій для виведення величини. Якщо величина не вміщується у вказаний формат, цей формат ігнорується; якщо ж величина коротша від формату, то символи, що виводяться, зсуваються вправо, а зайві позиції зліва заповнюються пропусками. Це дозволяє роздруковувати на екрані значення у вигляді таблиць рівними стовбчиками. Для REAL-величин другий параметр (ціле число m) зазначає кількість десяткових знаків після десяткової крапки, що виводяться. Число виводиться в форматі з фіксованою крапкою з автоматичним округленням. Якщо m=0, то REAL-число виводиться як ціле (без крапки).

Приклади:

i:=25;
a:=3.48;
write (‘Результат :’i:5,a:9:3)
write (x:7:0);
Результат виведення:
Результат: 25 3.480
Перед числом 25 виведено 3 пропуски, а перед 3.480 – 8 пропусків.

Без зазначення формату REAL-величини виводяться у форматі з плаваючою крапкою (в Е-формі). Зазначення формату впливає лише на виведення і не впливає на самі значення змінних.

Як вже відмічалось, на початку програми треба описати всі змінні, що використовуються в тілі програми. Робиться це після службового слова **VAR** у такому вигляді:

VAR список змінних: тип;
……
список змінних: тип;

Якщо деякі об'єкти програми не передбачено змінювати в процесі обробки даних, їх краще об'явити на початку програми у вигляді проіменованих констант після слова CONST:

CONST ім'я = значення;
ім'я = значення;
……
ім'я = значення;

Тип константи транслятор визначає автоматично по її значенню. Причому значенням константи можуть бути не тільки символи типу char, а й рядки символів, взятих в апострофи.

Наприклад, якщо об'явлено:
const mess='значення a=';
то в тілі програми оператор
write(mess,a);
чи безпосередньо
write ('значення a=');
призведе до того самого результату — виведе на екран повідомлення (при a=5):
значення a=5

А тепер давайте спробуємо написати програму.

Задача 1. Знайти довжину кола, якщо відомий її радіус.

```
PROGRAM N1;                          {цей рядок може бути і відсутнім }
Const pi=3.1415926;
var r,l:real;
    BEGIN
        readln (r);                  {введення радіуса}
        l:=2*pi*r;                   {обчислення довжини кола}
        writetn(l:0:2);              {виведення результату}
   END.
```

Результат роботи програми.

75.40

* Службове слово *PROGRAM* на початку програми не є обов'язковим і я не завжди буду починати програму іменно так.

Задача 2. Знайти площу трикутника за формулою Герона $s = \sqrt{p(p-a)(p-b)(p-c)},$
(де p=(a+b+c)/2 - півпериметр).

```
PROGRAM 2;
Var a,b,c: real;          {сторони трикутника}
        p: real;                  {допоміжна змінна}
        s: real;          {площа}
    BEGIN
        Readln (a,b,c);
        p:=(a+b+c)/2;
        s:= sqrt(p*(p-a)*(p-b)*(p-c));
        writeln(s:7:3);
    END.
```

Результат роботи програми.
2 3 4
2.905

Якщо в цій програмі Ви введете сторони неіснуючого трикутника (сума двох сторін менша за третю), то ЕОМ видасть помилку, аварійно завершивши програму.

Завдання для самоконтролю

Якщо в умові задачі нічого не сказано про тип змінних, то ці змінні можуть бути як цілі, так і дійсні. Отже описувати їх слід REAL. Після кожної задачі буде міститися контрольний приклад вхідних та вихідних даних. Це означає те, що коли ви введете при запуску програми вхідні дані, то ваша програма має вивести результат, який співпаде із вихідними даними. Увага! Правильний вивід вихідних даних не гарантує того, що ваша програма буде правильно працювати при інших вхідних даних. Перевірку треба обов'язково продовжити самостійно.

Задача 1-1.
Дано цілі числа a, b, що ж катетами прямокутного трикутника. Скласти програму знаходження його площі. В результаті вивести два знаки після коми.
Вхідні дані
4 5
Вихідні дані
10.00

Задача 1-2.
Дано чотири дійсні числа a, b, c, d. Знайти їх суму. В результаті вивести чотири знаки після коми.
Вхідні дані
3 4 2.5 1
Вихідні дані
10.5000

Задача 1-3.
Знайти периметр прямокутника, якщо відомо дві його сторони a та b. Сторони прямокутника цілі числа.
Вхідні дані
2 6
Вихідні дані
16

Задача 1-4.
Дано цілі числа x, y. В першому рядку вивести залишок від ділення першого числа на друге, а в другому залишок від ділення другого на перше.
Вхідні дані
15 16
Вихідні дані
15
1

Задача 1-5.
Дано дійсне число p. В першому рядку вивести його заокруглене ціле значення, в другому – його цілу частину, в третьому – його дробову частину (один знак після коми).
Вхідні дані.
3.5
Вихідні дані.
4
3
0.5

Задача 1-6

Дано цілі числа x та y. Знайти значення виразу: $x^3 + \frac{x+1}{y^2+1}$. В результаті вивести один знак після коми.

Вхідні дані.
3 1
Вихідні дані.
29.0

Задача 1-7

Дано цілі числа x, y. Знайти різницю першого та другого числа.
Вхідні дані.
10 4
Вихідні дані.
6

Задача 1-8

Знайти значення виразу: $\frac{2x^2+1}{x^2+1} + 5$. В результаті вивести два знаки після коми.

Вхідні дані.
0
Вихідні дані.
6.00

Задача 1-9

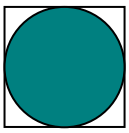
Дано дійсне число N. Вивести подвоєну його цілу частину.
Вхідні дані.
12.958
Вихідні дані.
24

Задача 1-10

Дано дійсні числа P і K. Дробову частину P помножити на K і вивести результат заокруглений до найближчого цілого.
Вхідні дані.
12.1257 1000
Вихідні дані.
126

Задача 1-11

На квадрат наклали круг такої величини, що він одночасно торкається кожної сторони квадрата. Сторона квадрата є ціле число a. Знайти найменше ціле число, що є більшим від сумарної площі частин квадрата видимих з під круга (круг не є прозорим).
Вхідні дані.
20
Вихідні дані.
86



Задача 1-12

Дано двоцифрове ціле число, що не закінчується нулем. Вивести частку від ділення першої цифри на другу.
Вхідні дані.
72
Вихідні дані.
3

Задача1-13

Дано натуральне трьохцифрове число. Вивести суму його цифр.
Вхідні дані.
123
Вихідні дані.
6

Задача 1-14

Координати точки цілі числа X і Y, причому X>0 та Y>0. Вивести у першому рядку координати точки симетричної даній відносно осі OX, у другому – відносно OY, у третьому – відносно початку координат і у четвертому площу фігури з вершинами у цих точках.
Вхідні дані.
1 2
Вихідні дані.
1 -2
-1 2
-1 -2
8

Задача 1-15

Дано два числа. Знайти суму їх дробових частин. В результаті вивести два знаки після коми.
Вхідні дані.
2.5 3.7
Вихідні дані.
1.20

Розділ 2. Логіка Паскаля

Якщо при програмуванні виникає ситуація, коли при виконанні певної умови треба виконати одні дії, а у протилежному разі — виконати інші дії, тоді використовується умовний оператор (або оператор розгалуження). Оскільки умови в Паскалі — це операції відношення, що є булевими (логічними) виразами, то умовний оператор у загальному вигляді записується таким чином:

ЯКЩО логічний вираз ТО оператор1 ІНАКШЕ оператор2;

Мовою Паскаль цей оператор записується за допомогою відповідних англійських службових слів IF, THEN, ELSE. Всю конструкцію прийнято оформлювати у вигляді вже знайомої тобі структури — "сходинками":

```
IF логічний вираз THEN
        оператор1
    ELSE
        оператор2;
```

Порядок виконання цього оператору нескладно зрозуміти. Якщо значення "логічного виразу" — істина (TRUE), то виконується "оператор1", а "оператор2" — ні. Якщо ж значення "логічного виразу" — хибність (FALSE), то навпаки, виконується "оператор2", а "оператор1" — ні. Зверніть увагу на правила запису. Якщо після умовного оператора IF в програмі є ще й інші оператори, то позаду "оператор2" обов'язково ставиться крапка з комою. Є ще й скорочена форма умовного оператора:

```
IF логічний вираз THEN
        оператор;
```

У цьому випадку, коли "логічний вираз" має значення хибність (FALSE), нічого не виконується, а відбувається перехід до наступного (після умовного) оператора. Згідно правил мови Паскаль після службових слів THEN та ELSE записується по одному оператору. Якщо ж у випадку "то" чи "інакше" потрібно виконати кілька операторів, то замість одного оператора потрібно використовувати СКЛАДЕНИЙ ОПЕРАТОР, який являє собою ланцюжок будь-яких операторів, обмежених службовими словами BEGIN та END:

begin оператор 1; оператор2;...; операторN end;

або краще записати:

```
begin
 оператор 1;
 оператор 2;
 ……;
 оператор N
end;
```

Слова BEGIN та END прийнято називати "операторними дужками", а окремі оператори всередині складеного відокремлюються один від одного крапкою з комою. Тоді у загальному вигляді умовний оператор можна записати так:

```
IF логічний вираз THEN
      BEGIN
          оператор 1;
          ……;
```

```
            оператор N
          END
    ELSE
        BEGIN
          оператор 1;
          ……;
          операторM
        END;
```

Весь оператор IF зі всіма вкладеними складовими операторами прийнято вважати одним оператором. Складені оператори можуть містити будь-яку кількість операторів — від одного й більше, враховуючи й інші умовні оператори.

Кожний END краще за все записувати суворо під відповідним йому BEGIN – це допоможе не допуститися помилки. Після першого складового оператора перед словом ELSE крапка з комою не ставиться, інакше транслятор сприйматиме це як закінчення умовного оператора (скорочена форма), а ELSE — як наступний оператор, який самостійно не використовується (виникає помилка).

Задача 1.
Дано два цілих числа a і b. Вивести більше з них.

```
var a,b : integer;
begin
        readln(a,b);
        if a>b then writeln(a)
                else writeln(b);
end.
```

Якщо ввести, наприклад, 5 3, то умова a>b буде TRUE і виконається writeln(a) і на екран буде виведене число 5. Якщо ж ввести 2 6, то умова a>b прийме значення FALSE і виконається те, що стоїть після else, тобто writeln(b). У цьому випадку програма виведе число 6.

Задача 2. З трьох цілих чисел визначити максимальне.

Визначимо 4 змінні: a, b, c, max. У перші три користувач введе свої числа, а 4-ту будемо використовувати як допоміжну. Порівняємо першу пару чисел і те з них, що більше, помістимо у змінну max. Потім порівняємо max з третім числом і знову те, яке буде більшим, збережемо в max. Наприкінці виведемо значення max на екран — воно й буде максимальним.
Програма виглядає так.

```
var   a,b,c: real;                          {три початкових числа}
      max: real;                            {допоміжна змінна}
begin
        readln(a,b,c);                      {введення чисел}
        if a>b                              {визначаємо те, що більше з двох}
                then max := a
                else max := b;
        if max < c                          {якщо третє число ще більше,}
                then max := c;          {то розміщуємо його в max; інакше}
          writeln (max)                     {в max залишається колишній результат}
end.
```

Алгоритм розв'язання цієї задачі можна побудувати і без допоміжної змінної max, але програма вийде більш громіздкою й заплутаною.

А тепер ще одна задача:

Задача 3. Визначити, чи є чотирикутник зі сторонами a, b, c та d ромбом.

Для розв'язання цієї задачі головне - правильно написати умови (логічний вираз). Відповідь можна дати стверджуючу тоді, коли всі сторони будуть дорів нювати одна одній ОДНОЧАСНО, тобто і друга сторона буде дорівню вати першій, і третя, і четверта. Тоді логічний вираз запишеться так:

(a=b) and (a=c) and (a=d)

Дужки необхідні, бо порядок старшинства операції AND вищий за операцію =, і без дужок логічний вираз буде виконуватися в іншій послідовності, що призведе до помилки. До речі, в Паскалі не можна записувати умови (логічні вирази) у вигляді a=b=c=d чи a<b<c<d, тому що результатом першої операції (a=b чи a<b) буде логічна величина True або False, яка буде потім порівнюватися з наступною величиною c. І якщо ця величина не логічного типу (boolean), то виникне помилка, бо величини несумісних типів не порівнюються. В складному логічному виразі головне — визначити, чи одночасно повинні виконуватися всі його складові частини чи достатньо, щоб виконувалася хоча б одна з них, і використати відповідні операції AND або OR.

Зараз програма буде будуватися елементарно:

```
var a,b,c,d: real;
begin
  readln(a,b,c,d);
  if (a=b) and (a=c) and (a=d)
                then writeln ('Yes')
                else writeln ('No')
end.
```

Оператор вибору

Умовний оператор використовується, коли розв'язання, в залежності від певної умови, розгалужується на дві гілки. І, хоча цей оператор можна вживати для будь-якої кількості гілок, у випадку, коли в залежності від значення деякого виразу чи змінної алгоритм роз галужується на декілька шляхів, зручніше застосувати оператор вибору:

CASE вираз OF

список констант; оператор;

список констант:оператор

ELSE оператор

END;

Оператор обчислює значення "виразу", обирає список констант, в якому міститься отримане значення, і виконує "оператор", що відповідає обраному варіанту, після чого виконання оператора CASE завершується. Якщо значення "виразу" не знайдено в жодному зі списків, тоді виконується "оператор", що йде після ELSE. До речі, перед ELSE крапка з комою також не ставиться. Якщо частина "ELSE оператор" відсутня, тоді для останнього випадку нічого виконуватися не буде. Значення "виразу" не може бути типу REAL. Самий "вираз" часто називають селектором. Як окремий випадок, роль селектора може відігравати одна змінна. Завершується оператор CASE службовим словом END.

І на заключення теоретичного блоку пропонується приклад програми.

Якщо в цілій змінній Month зберігається номер місяця, то відповідний номер кварталу року можна визначити так:

```
case Month div 3 of
        0: writeln ('1 квартал');
        1: writeln ('2 квартал');
        2: writeln ('3 квартал');
        3: writeln ('4 квартал')
    else writeln ('Такого місяця немає')
end;
```

або так:

```
case Month of
        1,2,3: writeln ('1 квартал');
        4,5,6: writeln ('2квартал');
        7,8,9: writeln ('3 квартал');
        10,11,12: writeln ('4 квартал')
  else writeln ('Такого місяця нема')
end;
```

Оператор вибору застосовується не дуже часто, але в деяких завданнях буває дуже зручним, тому що виглядає більш чітко, ніж численні вкладені умовні оператори. Його недолік полягає у тому, що не можна вибір варіантів здійснювати за умовами, як в операторі IF. В кожному варіанті оператора CASE застосовується лише один оператор, який, звісно, можна замінити складеним оператором.

Завдання для самоконтролю

Задача 2-1
Дано ціле число X. Вивести Yes, якщо воно парне, або No коли X є непарним.
Вхідні дані
56
Вихідні дані.
Yes

Задача 2-2.
Дано цілі числа a та b. Вивести подвоєне більше число.
Вхідні дані
4 6
Вихідні дані.
12

Задача 2-3
Дано ціле числа a. Якщо воно парне – вивести його частку від ділення на 2, інакше 0.
Вхідні дані
22
Вихідні дані.
11

Задача 2-4

Дано цілі числа a та b. Вивести їх в один рядок в порядку зростання, тобто, спочатку менше, а потім більше.

Вхідні дані

6 1

Вихідні дані.

1 6

Задача 2-5

Дано дійсне число p. Вивести «Yes», якщо воно ціле і «No» в іншому випадку.

Вхідні дані

2.6

Вихідні дані.

No

Задача 2-6

Дано цілі числа a, b, c, d. В одному рядку вивести спочатку найменше з них, а потім найбільше.

Вхідні дані

2 8 1 10

Вихідні дані.

1 10

Задача 2-7

Дано цілі координати точки x, y. Точка не лежить на осі координат. Вивести «Yes», якщо точка лежить у другій чверті або «No» в іншому випадку.

Вхідні дані

2 -8

Вихідні дані.

No

Задача 2-8

Дано цілі координати точки x, y. Точка не лежить на осі координат. Вивести номер чверті, якій належить дана точка.

Вхідні дані

2 -8

Вихідні дані.

4

Задача 2-9

Є додатні цілі x, y та z. Потрібно з'ясувати, чи існує трикутник зі сторонами x, y, z. Вивести «Yes» або «No».

Вхідні дані

1 4 5

Вихідні дані.

No

Задача 2-10

Дано цілі числа a, b, c, d. Вивести «Yes», якщо вони є впорядковані за зростанням, інакше вивести «No».

Вхідні дані

3 4 4 6

Вихідні дані.

No

Задача 2-11

Дано натуральне число N. Вивести «Yes», якщо воно є квадратом натурального числа, інакше вивести «No».

Вхідні дані

16

Вихідні дані.

Yes

Задача 2-12

Нехай координати клітин шахової дошки задаються за допомогою чисел, тобто буквені координати «abcd…» змінено на «1234…». Так клітини (1,1) та (1,2) є сусідніми в рядку, а клітини (1,1) та (2,1) – знаходяться по сусідству у стовпчику. Дано координати двох клітинок. Вивести «Yes», якщо вони одного кольору і «No» - якщо клітинки різних кольорів.

Вхідні дані

1 2 1 1

Вихідні дані.

No

Задача 2-13

Дано трьохзначне число. Вивести «Yes», якщо його цифри утворюють зростаючу або спадну послідовність. В іншому випадку вивести «No».

Вхідні дані

122

Вихідні дані.

No

Задача 2-14

Дано координати двох клітинок шахової дошки. Вивести «Yes», якщо слон може за один хід перейти з однієї клітинки в іншу. В іншому випадку вивести «No».

Вхідні дані

1 1 3 3

Вихідні дані.

Yes

Задача 2-15

Дано координати двох клітинок шахової дошки. Вивести «Yes», якщо кінь може за один хід перейти з однієї клітинки в іншу. В іншому випадку вивести «No».

Вхідні дані

3 1 1 2

Вихідні дані.

Yes

Розділ 3. Організація циклів.

При складанні програм часто виникає потреба повторити якусь певну команду чи групу команд кілька разів. Для цього існують спеціальні конструкції, що називаються циклами (або командами повторення) які просто зазначають, скільки разів потрібно повторити певну команду (групу команд) чи до яких пір повторювати ці команди. В мові Паскаль існує три види циклів, які починаються службовими словами WHILE (поки), REPEAT (повторювати) і FOR (для).

Перш, ніж перейти до розгляду варіантів організації циклів розглянемо задачу, яка просто розв'язується з використанням теорії, що буде описано нижче.

Цикл WHILE

Записується в такому вигляді:

```
WHILE логічний вираз DO оператор
або
WHILE логічний вираз DO
    BEGIN
        оператор;
        ……..;
        оператор
    END;
```

Читається це так: ПОКИ "логічний вираз" вірно, ВИКОНАТИ "оператор (складений оператор). Ту частину, яку потрібно виконувати, називають тілом циклу. Працює ця конструкція таким чином. Спочатку обчислюється значення "логічного виразу". Якщо воно дорівнює TRUE (істина), виконується тіло циклу після службового слова DO після чого виконується повернення на початок циклу і все повторюється знову починаючи з обчислення "логічного виразу". Цикл виконується доти, поки значенням "логічного виразу" не буде FALSE (хибність). Після цього продовжують виконуватись оператори, що записані після циклу (відразу за його тілом). Якщо при початковій перевірці значенням "логічного виразу" відразу стало FALSE, то тіло циклу не виконується жодного разу. В тілі циклу необхідно передбачити зміну даних для "логічного виразу", в протилежному випадку його значення може постійно опинятися TRUE, і цикл буде виконуватися без кінця (зациклювання).

Задача 1.
Знайти суму перших ста натуральних чисел.

```
Var s: integer;                 {тут буде накопичуватися наша сума}
    i: integer;                 { це буде змінна для організації циклу}
begin
      s:=0;                     {положили число, нейтральне до додавання}
      i:=1;                     {почнемо перегляд з 1}
      while i<=100 do begin     {виконуємо, якщо і ще менше-рівне 100}
            s:=s+i;             {додаємо чергове число у нашу суму}
            i:=i+1;             {переходимо до наступного}
      end;
      writeln(s);               {вивести суму}
end.
```

Цикл REPEAT

має вигляд:

```
REPEAT
      оператор(и)
UNTIL логічний вираз
```

Читати треба так:

ПОВТОРЮВАТИ "оператор(и)" ДОТИ, ПОКИ "логічний вираз" не стане вірним (TRUE). Зверніть увагу, що цикл завершується, коли логічний вираз стає TRUE, а не FALSE, як в циклі WHILE. Якщо в тілі циклу використано кілька операторів, їх не обов'язково обмежувати словами BEGIN та END у вигляді складеного оператора, оскільки слова REPEAT та UNTIL і без цього чітко визначають початок і кінець циклу. Тому, що умова виходу з циклу перевіряється в кінці, тіло циклу виконується ЗАВЖДИ як мінімум один раз. Якщо "логічний вираз" постійно залишатиметься FALSE, відбудеться зациклювання. Цикл REPEAT використовується в тих ситуаціях, коли виконується певна група операторів, а потім, в залежності від результатів виконаних дій, чи далі продовжується робота програми, чи ця група операторів виконується знову.

Знову розглянемо розв’язок задачі 1, але уже з використанням циклу REPEAT.

```
Var s: integer;                 {тут буде накопичуватися наша сума}
    i: integer;                 { це буде змінна для організації циклу}
begin
      s:=0;                     {положили число, нейтральне до додавання}
      i:=1;                     {почнемо перегляд з 1}
      repeat                    {початок циклу}
            s:=s+i;             {додаємо чергове число у нашу суму}
            i:=i+1;             {переходимо до наступного}
      until i=101;              {якщо наступне не «наше»- виходимо з циклу}
      writeln(s);               {вивести суму}
end.
```

Цикл FOR

У цьому циклі не перевіряється умова закінчення циклу, а просто рахується, скільки разів виконується його тіло. Тому зациклювання циклу FOR просто неможливе. Рахування йде за допомогою спеціального параметра — лічильника. У зв'язку з цим цикл FOR часто називають циклом з параметром. Існує два різновиди цього оператора:

1) FOR ім'я_змінної:=вираз1 TO вираз2 DO оператор
2) FOR ім'я_змінної:=вираз1 DOWNTO вираз2 DO оператор

"Оператор" в тілі циклу обох конструкцій може бути складеним. Змінна, яка записана після слова FOR, і є лічильником циклу. Обидва різновиди читаються так:

ДЛЯ лічильника „ім'я_змінної” що змінюється від значення "виразі" ДО значення "вираз2" ВИКОНАТИ "оператор". Лічильник і значення виразів у заголовку циклу повинні бути типу не REAL.

Виконується оператор FOR таким чином. Змінній-лічильнику надається початкове значення, яке дорівнює значенню "вираз1". Потім виконується тіло циклу. Далі в комірку

лічильника, згідно з його типом, розміщується наступне значення (для 1-го варіанта) або попереднє (для 2-го варіанта), і знову виконується тіло циклу.

Востаннє цикл буде виконано, коли значення лічильника дорівнюватиме "виразу2". Якщо у першому різновиді оператора FOR значення "виразу 1" більше, а в другому різновиді менше за значення "виразу2", то цикл не виконуватиметься жодного разу. Значення лічильника може використовуватися всередині циклу. Як частковий випадок, замість виразів у заголовку циклу можуть бути використані окремі змінні або константи.

ПОПЕРЕДЖЕННЯ!

Цикп FOR все ж можливо зациклити, якщо в тілі циклу переозначити змінну-лічильник так, щоб вона не змогла б дорівнювати значенню "виразу2". Намагайтеся НЕ ПЕРЕОЗНАЧУВАТИ ЛІЧИЛЬНИК всередині циклу FOR!

Розглянемо задачу знаходження суми ста перших натуральних чисел за допомогою циклу FOR у двох його варіантах.

Варіант 1.

```
Var s: integer;                 {тут буде накопичуватися наша сума}
    i: integer;                 { це буде змінна для організації циклу}
begin
      s:=0;                     {положили число, нейтральне до додавання}
      for i:=1 to 100 do        {початок циклу з перевіркою умови виходу (i<=100)}
            s:=s+i;             {додаємо чергове число у нашу суму}
      writeln(s);               {вивести суму}
end.
```

Варіант 2.

```
Var s: integer;                 {тут буде накопичуватися наша сума}
    i: integer;                 { це буде змінна для організації циклу}
begin
      s:=0;                     {положили число, нейтральне до додавання}
      for i:=100 downto 1 do    {початок циклу з перевіркою умови виходу (i>=1)}
            s:=s+i;             {додаємо чергове число у нашу суму}
      writeln(s);               {вивести суму}
end.
```

В чому полягає різниця виконання даних варіантів програм? Лише у порядку додавання чисел. Так, в першому варіанті, спочатку додається 1 і число 100 буде додане до суми останнім, а в другому варіанті число 100 додається першим і 1 буде додана останньою.

Задача 2.

Знайти найбільший спільний дільник двох чисел a і b.

Найбільший спільний дільник можна визначити за допомогою алгоритму Евкліда. Алгоритм ґрунтується на таких властивостях цілих чисел: якщо a>b, то NSD(a,b)=NSD((a-b),b); якщо a<b, то NSD(a,b)=NSD(a,(b-a)); якщоa=b, то NSD(a,b)=a. Застосуємо цю властивість у циклі і отримаємо таку симпатичну програму:

```
Var a,b: integer;
Begin
      Readln(a,b);
      While a<>b do
            If a>b then a:=a-b else b:=b-a;
       Writeln(a);
End.
```

Задача 3. Протабулювати функцію $y=x^2+2x-5$ на проміжку [-2;2] з кроком 0.5.

Протабулювати – значить вивести значення x, та відповідне значення y. Виведемо ці значення у вигляді таблиці. В кожному рядку будемо виводи спочатку значення x, а потім відповідне значення функції. Оскільки крок не є цілим числом, то всі змінні мають бути дійсними числами, наприклад, типу REAL.

```
Var x,y,step:real;
Begin
        step:=-2;
        while step<=2 do begin
                y:=sqr(x)+2*x-5;
                writeln(x:4:2,y:4:2);
                step:=step+0.5;
        end;
end.
```

Задача 4.

Визначити, чи є задане натуральне число простим.

Згадаємо з математики, що просте — це число, яке цілком без остачі ділиться тільки саме на себе та на одиницю, і складемо алгоритм розв'язання. Найпростіший метод — "лобова атака". Будемо ділити задане число N на всі числа підряд від 2 до N-1, тим більше, що за нас це робить ЕОМ. Якщо хоч при якомусь одному діленні вийде нульовий залишок — число N складене. Інакше — просте.

```
var    n,i: integer;                {i-число, на яке будемо ділити}
      flag: boolean;                {допоміжна змінна -"прапорець"}
 BEGIN
        readln(n);                  {вводимо з клавіатури число}
        i: = 2;                     {починаємо ділити з 2}
       flag:=true;                  {"піднімаємо" прапорець як дозвіл}
                                    {почати цикл ділень}
      while (i<n) and flag do       {працює, поки дільник не перевищує}
                                    {n і прапорець дозволяє ділити}
          begin
            if (n mod i)=0          {у випадку нульового залишку "опускаємо"}
                then flag:=false;   {прапорець, цикл ділень припиняється}
            i:=i+1;                 {не забуваємо збільшити дільник}
          end;
          if flag                   {якщо прапорець залишився "піднятим",}
                then writeln ('Yes')  {то число просте}
                else writetn('No')
 END.
```

Новим у цьому прикладі є використання прапорця. Такі прапорці (булеві змінні) часто вживаються програмістами в якості індикаторів певних подій або процесів (наприклад, TRUE — дія відбулась, FALSE — ще ні і т.ін.). У вищенаведеній програмі прапорець дуже наочно контролює процес ділення. В тілі циклу перевіряємо: якщо ділення дало ненульовий залишок, то значення прапорця не змінюється (це забезпечує повторне виконання циклу), дільник нарощується на 1 і цикл повторюється. Якщо ж залишок дорівнює 0, це означає, що число n крім себе і на 1 ділиться ще на "щось", тобто воно є складене. Прапорець скидається в FALSE, після чого дільник ще раз збільшується на 1, потім перевіряється логічний вираз у заголовку циклу (на цей раз він стане FALSE) і цикл завершується. Значення цього ж прапорця зручно використовувати і для виведення результату.

Завдання для самоконтролю

Задача 3-1

Дано N цілих чисел. Знайти кількість парних серед них.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести кількість парних чисел.

Вхідні дані

5

12 4 2 7 9

Вихідні дані

3

Задача 3-2

Дано N цілих чисел. Знайти суму трьохцифрових чисел, що є серед даного набору.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести суму трьохцифрових чисел.

Вхідні дані

5

12 4 200 7 101

Вихідні дані

301

Задача 3-3

Дано N цілих чисел. Всі непарні числа піднімати до квадрату та виводити в одному рядку через пропуск.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести квадрати непарних чисел.

Вхідні дані

5

12 4 2 7 9

Вихідні дані

49 81

Задача 3-4

Дано N цілих чисел. Знайти суму чисел кратних 5.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести суму чисел.

Вхідні дані

5

15 4 2 10 9

Вихідні дані

25

Задача 3-5

Дано N цілих чисел. Знайти кількість чисел, що закінчуються двійкою.

ТУ. У першому рядку стандартного вхідного потоку міститься N (N<1000) – кількість чисел. У наступному рядку через пропуск дано самі числа. У стандартний вихідний потік вивести кількість чисел.

Вхідні дані

5

15 4 2 10 92

Вихідні дані

2

Задача 3-6

Знайти всі числа менші N, які є квадратами натуральних чисел.

ТУ. У стандартному вхідному потоці дано ціле додатне N (N<=1000000). У стандартний вихідний потік вивести у порядку зростання відібрані числа.

Вхідні дані

30

Вихідні дані

1 4 9 16 25

Задача 3-7

Знайти дільники числа N.

ТУ. У стандартному вхідному потоці дано ціле додатне N (N<=1000000). У стандартний вихідний потік вивести через пропуск його дільники.

Вхідні дані

10

Вихідні дані.

1 2 5 10

Задача 3-8

Знайти прості дільники числа N.

ТУ. У стандартному вхідному потоці дано ціле додатне N ($N<=10^9$). У стандартний вихідний потік вивести через пропуск його прості дільники. Якщо число N ділиться на деяке просте число більше одного разу, то виводити цей дільник також більше одного разу.

Вхідні дані

20

Вихідні дані.

2 2 5

Задача 3-9

Знайти суму чисел, кратних k і менших n.

ТУ. У стандартному вхідному потоці містяться цілі числа k, n (1<=n,k<=10000). У стандартний вихідний потік вивести суму чисел.

Вхідні дані

7 20

Вихідні дані.

21

Задача 3-10

Знайти середнє арифметичне парних чисел з проміжку [n;m] (1<=n,m<=10000).

ТУ. У стандартному вхідному потоці містяться цілі числа n, m. У стандартний вихідний потік вивести результат з двома знаками після коми.

Вхідні дані

10 13

Вихідні дані.

11.00

Задача 3-11.

Дано натуральне число N (N<=30000). Перевірити чи є воно досконалим. Вивести «Yes» або «No». Число називається досконалим, якщо воно дорівнює сумі всіх своїх додатних дільників, окрім самого себе.

Вхідні дані

6

Вихідні дані.

Yes

Задача 3-12

Дано натуральне число N (N<=30000). Перевірити чи є воно простим. Вивести «Yes» або «No».

Вхідні дані

13

Вихідні дані.

Yes

Задача 3-13

Дано натуральне число N (N<=30000). Вивести всі прості числа не більші за N.

Технічні умови (ТУ). У вхідному потоці дано N, у вихідний потік через пропуск вивести прості числа.

Вхідні дані

13

Вихідні дані.

2 3 5 7 11 13

Задача 3-14

Дано натуральне число N,M (N,M<=30000, N<=M). Вивести всі прості числа з проміжку [N,M].

ТУ: У вхідному потоці дано два числа через пропуск N і M. У вихідний потік через пропуск вивести прості числа.

Вхідні дані

5 10

Вихідні дані.

5 7

Задача 3-15

Дано натуральне число N (N<=30000). Вивести кількість досконалих чисел меньших N.

Вхідні дані

100

Вихідні дані.

2

Задача 3-16

Прості числа, різниця між якими дорівнює два, називають «близнятами». Знайти кількість «близнят», що не більші N.

ТУ: У вхідному потоці дано число N (N<=30000). У вихідний потік вивести кількість «близнят».

Вхідні дані

18

Вихідні дані.

3

Задача 3-17

Дано натуральне число N (N<=30000). Знайти кількість чисел, що є повними квадратами цілих чисел та меншими n.

ТУ: У вхідному потоці дано число N (N<=30000). У вихідний потік вивести кількість чисел - повних квадратів.

Вхідні дані

50

Вихідні дані.

7

Задача 3-18

Дано натуральне число N. Скільки цифр у цьому числі? Задачу розв'язати з використанням операторів організації циклів.

ТУ: У вхідному потоці дано число N (N<=10^9). У вихідний потік вивести кількість цифр.

Вхідні дані

1234

Вихідні дані.

4

Задача 3-19

Знайти кількість чисел виду m^3-3m-3, де m=1, 2, 3….. , не більших n.

ТУ: У вхідному потоці дано число n (n<=3000). У вихідний - вивести кількість таких чисел.

Вхідні дані

10

Вихідні дані.

2

Задача 3-20

Знайти суму чисел виду m^2-2m-1, де m=0, 1, 2,…n.

ТУ: У вхідному потоці дано число n (n<=100). У вихідний - вивести суму чисел.

Вхідні дані

2

Вихідні дані.

-4

Задача 3-21

«Числа Фібоначчі»

Числами Фібоначчі називаються числа послідовності, у якій кожне наступне число дорівнює сумі двох попередніх. Вивести N-те число Фібоначчі (N<=30). Перші два числа послідовності дорівнюють одиниці.

ТУ: У вхідному потоці міститься одне число N, у вихідний потік вивести єдине число, N-те число Фібоначчі.

Вхідні дані

5

Вихідні дані.

5

Задача 3-22

«Кролики»

Є пара кроликів. Пара починає щомісяця народжувати нову пару на третій місяць життя. Скільки кроликів буде через n місяців?

ТУ: У вхідному потоці міститься число n (3<=n<=30), у вихідний потік вивести єдине число – кількість кроликів через n місяців.

Вхідні дані

6

Вихідні дані.

16

Задача 3-23

«Кролики-2»

Земляни знайшли планету, придатну для життя і відправили космічний корабель з одним кроликом щоб переконатися у придатності для життя клімату планети. Кролику сподобався клімат і уже через місяць він привів ще одного (на цій планеті для цього достатньо одного кролика). Дальше кролики почали розмножуватися з такою ж швидкістю – кожен кролик через місяць приводив ще одного. Але, розмноження кроликів почав контролювати монстр із математичними здібностями. Як тільки на початку якогось місяця кроликів ставало більше k, він поїдав k кроликів.

Треба визначити скільки кроликів буде на планеті через n місяців від початку висадки кролика на планету.

ТУ. У вхідному потоці міститься два числа n і k (0<=k,n<=30), у вихідний потік вивести єдине число – кількість кроликів через n місяців.

Вхідні дані

5 10

Вихідні дані

12

Задача 3-24

Задано послідовність із n (0<n<=30000) цілих чисел. Вивести максимальне число, яке зустрічається у цій послідовності.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести найбільше число.

Вхідні дані

7

3 5 1 1 6 6 5

Вихідні дані.

6

Задача 3-25

Задана послідовність із n (0<n<30000) цілих чисел. Вивести порядковий номер останнього максимального числа, яке там зустрічається.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести порядковий номер числа.

Вхідні дані

8

3 5 1 1 0 6 6 5

Вихідні дані.

7

Задача 3-26

Задана послідовність із n (0<n<30000) цілих чисел. Вивести порядковий номер першого максимального числа, яке там зустрічається.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести порядковий номер числа.

Вхідні дані

6

3 5 1 6 6 5

Вихідні дані.

4

Задача 3-27

Задана послідовність із n (0<n<30000) цілих чисел. Вивести кількість елементів, що ідуть підряд і утворюють найбільшу не спадну підпослідовність.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести одне число.

Вхідні дані

7

3 5 1 1 6 6 7

Вихідні дані.

5

Задача 3-28

Задана послідовність із n (0<n<30000) цілих чисел. Вивести кількість елементів, що ідуть підряд і утворюють найбільшу зубчасту підпослідовність. Наприклад, числа 1213243 утворюють зубчасту підпослідовність довжиною 7, за більшим числом іде менше, за меншим – більше.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести одне число.

Вхідні дані

7

3 5 1 1 0 6 5

Вихідні дані.

4

Задача 3-29

Задана послідовність із n (0<n<30000) цілих чисел. Знайти довжину найбільшої підпослідовності, що є монотонною.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести одне число.

Вхідні дані

11

1 1 1 2 3 8 7 5 4 3 1

Вихідні дані.

6

Пояснення. У нашому прикладі є такі монотонні підпослідовності: 111, 238, 875431.

Задача 3-30

Задана послідовність із n (0<n<30000) цілих чисел. Знайти початок та кінець найбільшої підпослідовності, що є монотонною.

ТУ. У першому рядку стандартного вхідного потоку дано число n, у другому - через пропуск n чисел. У вихідний потік вивести два числа, що є номерами початкового та кінцевого елементів підпослідовності. Якщо таких під послідовностей є декілька, то виводити найменші номери.

Вхідні дані

11

1 1 1 2 3 8 7 5 4 3 1

Вихідні дані.

6 11

Розділ 4. Процедури

В мові Паскаль використовується два види підпрограм - процедури та функції. Вони відрізняються між собою структурою та способом виклику.

При проектуванні програми визначається, які частини алгоритму треба реалізувати як процедури, а де знадобиться функція. В головній частині програми підпрограми розташовуються після розділу опису даних перед оператором Begin, а викликаються при потребі в процесі виконання головної програми або іншої підпрограми.

Структура процедури має вигляд:

```
Procedure ім’я (список формальних параметрів);

    Розділ локальних даних

  Begin

    ... {розділ виконавчих операторів}

  End;
```

Перший рядок складає заголовок процедури, ім’я процедури вибирає програміст так як і ім'я змінної. В списку формальних параметрів описуються через ; параметри та інформація про їх тип. Деякі параметри призначені для передачі даних в процедуру, інші для повернення результатів з процедури до тієї програмної одиниці, яка її викликала.

В розділі локальних даних (який взагалі може бути відсутнім) описують ті дані, які використовуються тільки для «службових» цілей в самій процедурі (параметри циклів, робочі змінні та масиви, тощо).

Всередині підпрограми записують послідовність операторів, які реалізують потрібний алгоритм. При цьому вони працюють з формальними параметрами, локальними та глобальними даними.

Зв'язок між окремими частинами програми здійснюється через списки формальних параметрів та за допомогою глобальних змінних.

Глобальні дані описуються в головній програмі вони не являються фактичними параметрами при виклику підпрограм, не описані в підпрограмах, а використовуються в одній з них і в головній програмі.

Результати роботи підпрограми можуть бути передані до головної програми через формальні параметри та глобальні дані.

Задача 1.
Дано лінійний масив з N (N<=1000) цілих чисел. Вивести на екран числа, що мають індекси від k1 до k2 (1<=k1,k2<=N, k1<=k2).
У першому рядку містяться числа N, k1, k2. У другому через пропуск вводиться N цілих чисел.

```
const max=1000;
var a           : array [1..max] of integer;
    n,k1,k2,i   : integer;
Procedure Print(a,b:integer);
var i    : integer;
begin
        for i:=a to b do
                write(a[i],' ');
        writeln;
end;
begin
        readln(n,k1,k2);
        for i:=1 to n do
                read(a[i]);
        Print(k1,k2);
end.
```

Результати роботи підпрограми можуть бути передані до головної програми через формальні параметри та глобальні дані.

Задача 2.
Дано лінійний масив з N (N<=1000) цілих чисел. Вивести на екран числа, що мають індекси від k1 до k2 (1<=k1,k2<=N, k1<=k2) та їх суму.
У першому рядку містяться числа N, k1, k2. У другому через пропуск вводиться N цілих чисел.

```
const max=1000;
var a           : array [1..max] of integer;
    n,k1,k2,I,S       : integer;
Procedure Print(a,b:integer; var c);
var i    : integer;
begin
        for i:=a to b do begin
                write(a[i],' ');
                c:=c+a[i];
        end;
        writeln;
end;
begin
        readln(n,k1,k2);
        for i:=1 to n do
                read(a[i]);
        Print(k1,k2,S);
        writeln(S);
end.
```

Можна виводити значення суми і в самій процедурі, але ми задалися ціллю продемонструвати повернення значень самою процедурою.

Завдання для самоперевірки

Задача 4-1
Задається N цілих додатних чисел не більших 1000. Знайти суму двоцифрових чисел, що є у даній послідовності.
ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести суму чисел, що відповідають умові задачі.

Вхідні дані
5
101 100 10 150 20

Вихідні дані
30

Задача 4-2
Задається N цілих додатних чисел не більших 1000. Знайти суму простих чисел, що є у даній послідовності.
ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести суму чисел, що відповідають умові задачі.

Вхідні дані
5
1 2 3 4 5

Вихідні дані
10

Задача 4-3
Задається N цілих додатних чисел не більших 1000. Серед них знайти кількість чисел, що є членами такої послідовності k^2+k+1, де k=0,1,2…..
ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести кількість чисел, що відповідають умові задачі.

Вхідні дані
5
3 5 6 7 8

Вихідні дані
2

Задача 4-4
Серед N цілих додатних чисел не більших 1000 знайти два найбільші.
ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести два числа через пропуск, що відповідають умові задачі. Спочатку вивести найбільше, а потім наступне за ним по величині.

Вхідні дані
5
1 2 3 4 5

Вихідні дані
5 4

Задача 4-5
Серед N цілих додатних чисел не більших 1000 знайти числа з найбільшою сумою цифр та найменшою. Якщо таких чисел є декілька, то слід вибирати ті, що ідуть у переліку першими.
ТУ. У першому рядку задано число N (N<=1000). У наступному рядку містяться самі числа. У вихідний потік вивести два числа через пропуск, що відповідають умові задачі. Спочатку вивести число з найбільшою сумою цифр, а потім з найменшою.

Вхідні дані
5
12 10 101 1000 102
Вихідні дані
12 10

Задача 4-6
Написати процедуру Minmax(A,B), яка записує у змінну A найменше із значень A та B, а в змінну B — найбільше із цих значень (A і B — дійсні парамети, що є одночасно вхідними та вихідними). Використовуючи чотири виклики процедури знайти найменше та найбільше із чисел A, B, C, D.

Вхідні дані
2 5 3 9
Вихідні дані
2 9

Задача 4-7
Написати процедуру SumDigit(N,S), яка знаходить суму цифр S цілого числа N (N - вхідний, S — вихідний параметр). Використовуючи цю процедуру знайдіть суму цифр для кожного із K даних чисел. Формат вхідних та вихідних даних такий, як у прикладі.

Вхідні дані
5
2 11 20 100 15
Вихідні дані
2
2
2
1
6

Задача 4-8
Дано два прямокутники з сторонами, паралельними осям координат. Знайти точки перетину сторін прямокутників. Гарантується, що лише одна вершина одного прямокутника може лежати всередині іншого і прямокутники не можуть мати більше двох спільних точок.
ТУ. У першому рядку вхідного потоку задаються координати протилежних вершин першого прямокутника, у другому – другого. Координати точок є цілими числами по модулю не більшими 10000. У вихідний потік вивести в першому рядку координату точки перетину, що лежить лівіше , а в другому – координати другої точки, якщо вона є. Якщо ж прямокутники не перетинаються, то вивести -1.

Вхідні дані
0 0 2 2
1 1 3 3
Вихідні дані
1 2
2 1

Задача 4-9

Дано два прямокутники з сторонами, паралельними осям координат. Знайти сумарну площу покриття прямокутниками площини.

ТУ. У першому рядку вхідного потоку задаються координати протилежних вершин першого прямокутника, у другому – другого. Координати точок є цілими числами по модулю не більшими 10000. У вихідний потік вивести площу покриття.

Вхідні дані

0 0 2 2

1 1 3 3

Вихідні дані

7

Задача 4-10

На площині дано N прямокутників і K точок. Які з точок не належать жодному з прямокутників?

ТУ. У першому рядку стандартного вхідного потоку дано N (N<=100). Дальше у N рядках задаються цілі X1,Y1, X2,Y2 (-10000<= X1,Y1, X2,Y2<=10000). Потім задається K (K<=1000) і у наступних K рядках ідуть координати точок X,Y (-10000<=X,Y<=10000). У вихідний потік виводити координати точок, що задовольняють умову задачі у порядку їх переліку.

Вхідні дані

1

0 0 10 10

2

1 1

11 11

Вихідні дані

11 11

Розділ 5. Функції.

Підпрограми-функції використовуються для реалізації алгоритму та повернення в головну програму одного результату в вигляді імені функції.

Ім'я функції вибирається довільно (як ім'я змінної).

Структура функції така:

```
Function ім'я(список формальних параметрів): тип імені;
  {локальні дані}
  Begin
    ...
          ім'я := ...;
          ...
  End;
```

Типом функції може бути скалярний тип, тобто: цілий, дійсний, логічний, символьний та рядковий тип String.

Відносно формальних параметрів, локальних та глобальних даних в функції діють такі ж самі обмеження та вимоги, що і в процедурах.

В виконавчій частині підпрограми-функції повинен бути хоча б один оператор в якому імені функції: призначається значення.

Звернення до функції виконується з якого-небудь арифметичного виразу так, як і до стандартних функцій тину sin(x), ln(x), тощо. Результат роботи функції передається в місце її виклику.

Задача 1. Скласти програму знаходження більшого з чотирьох чисел.

```
var x,y,z,k : integer;
Function Max(a,b:integer):integer;
begin
  if a>b then Max:=a else Max:=b;
end;
begin
  readln(x,y,z,k);
  writeln(Max(Max(x,y),Max(z,k)));
end.
```

Функція Max() повертає значення більшого з двох чисел. В основній програмі аргументом процедури writeln() є функція Max() у якої параметри – значення, які повертає ця ж функції Max(). Звичайно, можна написати цю програму і зрозуміліше, але для цього треба ще описати додаткові змінні.

```
var x,y,z,k,m1,m2 : integer;
Function Max(a,b:integer):integer;
begin
  if a>b then Max:=a else Max:=b;
end;
begin
  readln(x,y,z,k);
  m1:=Max(x,y);
  m2:=Max(z,k);
  writeln(Max(m1,m2);
end.
```

Код програми розширився, але при цьому програма стала дещо більш зрозумілою.

Після опису функції її можна використовувати у виразах поряд із стандартними функціями. Аргументами функцій можуть бути будь-які вирази. Порядок слідування і типи аргументів мають бути такими ж, як і в параметрів у заголовку функції.

Обчислення виразів, що містять звернення до функцій, відбувається за таким алгоритмом:

- ❖ обчислюється вираз для аргументів функції;
- ❖ значення аргументів присвоюються параметрам із заголовку функції;
- ❖ виконується тіло функції і обчислюється її значення;
- ❖ значення функції ставиться у вираз на місце звернення до функції;
- ❖ Обчислення виразу продовжується.

У прикладі наведеної задачі можна було звернути увагу на те, що аргументом функції може бути ця або інша функція. Для кожного виклику функції наведений алгоритм буде виконуватися окремо і самого початку.

Задача 2.
Знайти значення виразу:
(n-k)! + n!

(n -1)! + (k-1)!
5! – читається: «п’ять-факторіал».
Факторіалом числа n називається добуток всіх натуральних чисел не більших n.
Тобто, n!=1*2*3*…*(n-1)*n.
Слід зауважити, що факторіал числа надзвичайно швидко зростає і у прикладі значення n,k мають бути не більші 12.
5!=1*2*3*4*5=120.

```
var n,k : longint;
      g  : real;
function F(a:integer):longint;
var d,i : longint;
begin
        d:=1;
        for i:=1 to a do
                d:=d*i;
        F:=d;
end;
begin
        readln(n,k);
        g:=(F(n-k)+F(n))/(F(n-1)+F(k-1));
        writeln(g:0:2);
end.
```

Задача 3
Дано натуральне число N (N<=30000). Вивести всі прості числа не більші за N.

```
var i,n,s :integer;

function prost(n:integer):boolean;
var i : integer;
begin
 prost:=true;
 if n=1 then begin prost:=false;exit;end;
 for i:=2 to round(sqrt(n)) do
```

```
      if n mod i=0 then begin prost:=false; break;end;
end;

begin
 readln(n);
 for i:=2 to n do
       if prost(i) then write(i,' ');
end.
```

Завдання для самоконтролю.

Задача 5-1
Знайти найбільший спільний дільник даних чисел.
ТУ. У першому рядку задано число N (0<N<1000), у наступному рядку через пропуск задаються самі цілі числа не більші 2*10^9. У вихідний потік вивести число, що є найбільшим спільним дільником для даних чисел.

Вхідні дані
4
10 22 30 50
Вихідні дані
2

Примітка. Можна використати таке співвідношення: НСД(a,b,c)=НСД(НСД(a,b),c).

Задача 5-2
Знайти значення виразу:

$$\frac{4\cdot\sum_{k=n}^{2n}2k}{n+\sum_{k=n-1}^{3n}2k}+\sum_{k=2n}^{3n}2k$$

ТУ. У стандартному вхідному потоці міститься число N (0<N<=1000). У вихідний потік вивести значення виразу з двома знаками після коми.

Вхідні дані
1
Вихідні дані
11.85

Задача 5-3
Описати функцію Calc(A, B, Op) дійсного типу, яка буде виконувати одну операцію над ненульовими дійсними A і B. Операція визначається цілим параметром Op: 1 - віднімання, 2 - множення, 3 - ділення, будь-які інші значення – додавання. З допомогою цієї функції для кожної із N (N<1000) трійок чисел A, B, Op вивести результат операції з точністю до двох знаків після коми.

Вхідні дані
3
2 3 1
3 2 2
2 4 3
Вихідні дані
-1.00
6.00
0.50

Задача 5-4
Описати функцію IsSquare(K) логічного типу, яка повертає True, якщо цілий параметр K (K > 0) є повним квадратом цілого числа, і False в іншому випадку. З допомогою цієї функції для N (N<1000) даних натуральних чисел визначити кількість чисел, що є повними квадратами цілих чисел.

Вхідні дані
3
4 15 36

Вихідні дані
2

Задача 5-5
Описати функцію DigitCount(K) цілого типу, яка знаходить кількість цифр цілого додатного числа K (K<10^18). З допомогою цієї функції для кожного з N (N<1000) чисел K вивести його кількість цифр.
Вхідні дані
3
4 1521 36009

Вихідні дані
1
4
5

Задача 5-6
Знайти на проміжку [N,M] кількість простих чисел, які можна розбити ще на два простих числа. До таких чисел належать, наприклад числа: 23 (2 і 3), 137 (13 і 7), 739 (7 і 39).
ТУ. У стандартному вхідному потоці знаходяться числа N,M (20<N,M<50000). У вихідний потік вивести кількість чисел, що задовольняють умову задачі.

Вхідні дані
21 40
Вихідні дані
2

Задача 5-7
Серед даних чисел знайти кількість чисел Фібоначчі. Числами Фібоначчі називаються числа, перші два з яких дорівнюють одиниці, а кожне наступне рівне сумі двох попередніх.
Наприклад: 1, 1, 2, 3, 5, 8,…
ТУ. У першому рядку задано число N (0<N<10000). У наступному рядку ідуть самі числа не більші 2*10^9. У вихідний потік вивести кількість чисел Фібоначчі.

Вхідні дані
5
1 18 3 4 5
Вихідні дані
3

Задача 5-8
Серед даних чисел знайти числа з найбільшою та найменшою сумою цифр. Якщо таких чисел є декілька то виводити ті, що ідуть першими у переліку.
ТУ. У першому рядку задано число N (0<N<10000). У наступному рядку ідуть самі числа не більші 2*10^9. У вихідний потік вивести спочатку число з найменшою сумою цифр, а потім через пропуск – з найбільшою.

Вхідні дані
5
1 18 31 41 54
Вихідні дані
1 18

Задача 5-9
Дано N натуральних чисел. У кожному числі переставити цифри таким чином, щоб утворене число було мінімально можливе. Цифра «0» не може бути першою цифрою числа.
ТУ. У першому рядку стандартного вхідного потоку знаходиться число N (N<=100000). Дальше у N рядках по одному числу, що не перевищують $2*10^9$. У вихідний потік вивести числа по одному у кожному рядку, що задовольняють умову задачі.

Вхідні дані
5
1230
395
10
987
9078

Вихідні дані
1023
359
10
789
7089

Задача 5-10
Для заданих N натуральних чисел знайти їх найменше спільне кратне. Гарантується, що найменше спільне кратне не буде перевищувати 10^{18}.
ТУ. У першому рядку стандартного вхідного потоку знаходиться число N (N<=100). Дальше у N рядках по одному числу, що не перевищують $2*10^9$. У вихідний потік вивести ціле число – найменше спільне кратне..

Вхідні дані
3
7
11
13

Вихідні дані
1001

Розділ 6. Масиви

Окрім простих (скалярних) типів даних, які визначають для кожної змінної тільки одне значення у кожний конкретний момент часу, часто приходиться працювати з більшою сукупністю однотипних даних, тісно пов'язаних одне з одним. У реальному житті такі дані зазвичай представляють у вигляді таблиць.

Всю таблицю називають одним ім’ям, як одну змінну, а доступ до окремих клітинок таблиці здійснюється за їх порядковим номером — індексом. Змінюючи тільки індекс, можна отримати доступ до будь-якого значення у таблиці. Такі табличні змінні в математиці називають матрицями, в програмуванні — масивами.

Розглянемо такий приклад:

A:

1	2		4	5	6
-1,5	0	2,4	6,15	12,1	30,7

Тут представлено лінійний масив A з шести значень (всі значення розташовано одим рядком). Лінійний масив часто ще називають вектором. Доступ до будь-якого значення здійснюється по імені таблиці A та індексу, записаному після імені в квадратних дужках:

A[3]=2.4;

A[5]=12.1;

A[6]=30.7;

A1 і A[1] — це не одне й те саме. A1 — це окрема змінна, а A[1] — це перший елемент масиву A.

Крім лінійних, часто використовують двовимірні і навіть багатовимірні масиви.

У двовимірних масивах значення розташовуються по рядках та стовбчиках:

B:

	1	2	3	4
1	2	-5	0	4
2	7	1	9	12
3	-8	3	-1	6

Наприклад, щоб отримати доступ до значення, що розташоване на перехресті 2-го рядка та 3-го стовпчика, необхідно після імені масиву B в якості індексів указати через кому спочатку номер рядка, а потім номер стовпчика:

B[2,3]=9

Трьохвимірний масив можна представити як набір значень, що розташовані у просторі (по довжині, ширині та висоті). Чотирьохвимірний масив важко собі уявити, але в математиці існують і чотирьох-, п'яти-, шести-, і більш вимірні масиви. В машинному поданні багатовимірність здійснюється дуже просто: масиви вкладаються один в один. Так двовимірний масив — це лінійний масив лінійних масивів. На практиці найчастіше працюють з лінійними та двовимірними масивами.

Оскільки масив — це таблична змінна, то й описується вона у розділі змінних після службового слова VAR:

VAR ім'я_змінної: ARRAY [тип_індексів] OF тип_значень;

"Тип_індексів" може бути перерахованим, "тип_значень" — будь-яким (крім файлового).

Приклад:

```
VAR tab : array [1..100] of integer;

    list  : array[0..9] of real;

    symbol : array[0..15, 0..15] of char;
```

Тут описано tab — лінійний масив із 100 цілих значень (припустимі значення індексу від 1 до 100), list — лінійний масив із 10 дійсних чисел (припустимі значення індексу від 0 до 9) та symbol — двовимірний масив символів із 16 рядків і 16 стовбців. Масив symbol можна було описати ще й так:

```
VAR symbol : array [0..15] of array [0..15] of char;
```

Можна описувати і так: тип масиву визначається як тип користувача у розділі TYPE, а потім описуються змінні-масиви з вико ристанням уже визначених типів:

```
TYPE index =1..15;

     Vector=array [index] of real;

VAR v1,v2:vector;
```

Тут описано два дійсних вектори v1 і v2 по 15 елементів у кожному. В Паскалі один масив можна присвоїти іншому масивові ціпком тільки у тому випадку, коли обидва масиви одного й того самого типу, тобто описані за допомогою одного й того самого типу, а не однакових типів.

Наприклад, якщо описано два вектори v1 і v2

```
VAR  m1: array [100..150] of integer;

     m2: array [100..150] of integer;
```

то присвоювання

```
m1:=m2
```

транслятор сприйматиме як помилку, бо m1 і m2 рахуються різних типів (хоча й однакових). А ось для вищеописаних векторів v1 та v2 буде правильним

```
v1:=v2.
```

При цьому всі відповідні-значення вектора v2 перепишуться у вектор v1. Інші операції з масивами зазвичай здійснюються поелементно.

Для полегшення задавання початкових значень (ініціалізації) змінних у є спеціальний засіб — типізовані константи. Вони описуються у розділі CONST як змінні з визначенням значень, як у констант:

```
CONST MaxNumbe:integer = 100;

massa : real = 12.5;

letter: array [1 ..5] of char = (‘a’, ’b’, ’c’, ’d’, ’e’);

mas : array [1..2,1. .6] of integer= ((2, 4, 6, 8, 10, 12), (1, 3, 5, 7, 9, 11));
```

Типізовані константи вживаються у програмі в якості змінних, що вже мають початкове значення. В процесі виконання програми в них можуть бути записані нові значення. При ініціалізації масивів значення запи суються рядками (дивися попередній приклад).

Задача 1.

Сформувати лінійний масив цілих чисел, які отримані з допомогою генератора випадкових чисел та знайти суму парних елементів цього масиву.

```
CONST n=10;
VAR     a       : array[1..n] of integer;
        i       : integer;
        s       : longint;
BEGIN
    randomize;                              {ініціалізація генератора випадкових чисел }
    for i:=1 to n do begin
            a[i]:=random(100);              {записали випадкове число від 0 до 99 }
            write(a[i]:4);                  {вивели для контролю на екран}
    end;
    writeln;                                {перейшли на новий рядок}
    s:=0;
    for i:=1 to n do
            if a[i] mod 2=0 then s:=s+a[i]; {якщо парне, то додаємо}
    writeln(s);                             {виводимо суму}
END.
```

Виділена частина програми може бути змінена у випадку, коли описується введення чисел із стандартного потоку або з файлу.

Задача 2.

Дано N (1<=N<=10000) цілих чисел. Знайти найбільше та найменше значення серед цих чисел.

У першому рядку вхідного потоку дано N, у другому міститься через пропуск N цілих. У вихідний потік вивести два числа розділені пропуском: найбільше та найменше число.

```
VAR     a               : array[1..n] of integer;
        i, n, min, max  : integer;
BEGIN
    readln(n);
    for i:=1 to n do
            read(a[i]);
    readln;                         {перейшли на новий рядок після введення даних}
    max:=a[1];                      {нехай перший елемент буде тимчасово максимальним}
    min:=a[1];                      { те ж саме, але мінімальним}
    for i:=2 to n do begin
```

```
            if a[i]>max then max:=a[i];   {якщо є більший, то беремо його максимальним}
            if a[i]<min then min:=a[i];   {якщо є менший, то беремо його мінімальним}
    end;
    writeln(max,' ',min);
END.
```

Задача 3. Сформувати та вивести двовимірний масив цілих чисел.

```
CONST       m = 10;
            n = 5;
VAR         a       : array[1..n, 1..m] of integer;
    i, j    : integer;
BEGIN
    randomize;
    for i:=1 to n do begin
            for j:=1 to  m do begin
                    a[i,j]:=random(100);
                    write(a[i,j]:3);
            end;
            writeln;
    end;
END.
```

Задача 4.

Дана матриця розмірності NxN. Знайти суми елементів розташованих по діагоналях.

ТУ. У першому рядку дано число N (N<100). Дальше у N рядках міститься по N чисел через пропуск. У вихідний потік вивести через пропуск два числа: суми елементів матриці, що знаходяться на діагоналях.

Яким чином будемо мислити. Зрозуміло, що треба якимось чином виділяти ті елементи, що належать чи одній, чи іншій діагоналі. Запишемо індекси матриці 4x4.

11 12 13 14
21 22 23 24
31 32 33 34
41 42 43 44

Для «синіх» індексів видно, що вони мають бути рівними. Отже, при i=j будемо мати елементи головної діагоналі.

Для «зелених» індексів також прослідковується закономірність: сума індексів дорівнює 5 при даній розмірності матриці. Отже, при розмірності NxN будемо мати таку умову: i+j=N+1. Тепер нескладно написати і програму.

```
VAR         a       : array[1..100,1..100] of integer;
            i, j, s1,s2: longint;
BEGIN
    readln(n);
    for i:=1 to n do begin
            for j:=1 to n do
                    read(a[i,j]);
            readln;
    end;
    s1:=0;
    s2:=0;
    for i:=1 to n do
            for j:=1 to n do begin
```

```
                if i=j then s1:=s1+a[i,j];
                if i+j=n+1 then s2:=s2+a[i,j];
        end;
    writeln(s1,' ',s2);
END.
```

Задача 6.

Сформувати цілий двовимірний масив за шаблоном, що запи саний в іншому масиві. Символ '+' шаблону означає, що значення дорівнює сумі його індексів, '*' - добуток індексів, а '0' - значення дорівнює нулю.

```
CONST   n=3;      {кількість рядків}
                m=4;      {кількість стовбців}
        maska : array of [1..n,1..m] of char =('+00*','0**0','*+*+'); {шаблон}
VAR     a: array [1. ,n, 1. .m] of Integer;   {масив}
        i,j: integer;
BEGIN                   {початок формування масиву}
        for i:=1 to n do
                        for j:=1 to m  do
                        case maska[i, j] of
                                '+':a[i,j]:=i+j;
                                '*':a[i,j]:=i*j;
                                '0':a[i,j]:=0;
                        end;
        for i:=1 to n do  begin
                for j:=1 to m do write(a[i,j]:3);
                writeln;
        end
END.
```

Кількість рядків та стовбців у масивах задано в константах n і m, бо при такому будуванні програма дуже легко модифікується для масивів інших розмірів. Ініціалізація масиву maska здійснюється не окремими символами, а рядками символів (запам'ятай цей захід), оскільки рядок символів рахують символьним вектором. Цикл по і задає зміну індексів рядків від 1 до n. Всередині кожного рядка цикл по j змінює індекси стовбців від 1 до m. Оператор case дозволяє вибрати необхідний алгоритм формування масиву A. Далі аналогічна конструкція з двох вкладених циклів роздруковує на екрані сформований масив A. При формуванні масиву A тілом циклу for i:=1 ... є один цикл for j:=1 .... а тілом циклу for j:=1 ... є один оператор case. Тому операторні дужки Begin та End не використовуються. При виведенні значень на екран по закінченні циклу for j:=1 ..., що виводить значення одним рядком, оператор writeln переведе курсор на початок нового рядка, тобто кожен рядок масиву буде надруковано окремим рядком екрану. Шаблон-масив maska можна сформувати і всередині тіла програми за допомогою аналогічних вкладених циклів, всередині яких записати оператор read (maska[i,j]). Але тоді всі значення потрібно вводити ручним способом.

Задача 7.

Відсортувати заданий цілий лінійний масив так, щоб на початку було розміщено додатні числа та нулі (не обов'язково по порядку), а потім від'ємні числа.

Задачу розв'яжемо так.

Проходимо весь масив, продивляючись пари чисел, що стоять поруч; якщо зліва від'ємне, а справа додатне число, числа переставляємо, в протилежному разі — ні. При будь-якій перестановці скидаємо прапо рець. На початку циклу перегляду прапорець

встановлюється. По закін ченні циклу перевіряється: якщо прапорець не скинуто, цикл повторю ється, якщо ж встановлено — програма закінчується.

```
const n=10;
type a : array[1..n] of integer = (-2,0,4,-5,9,1,0,4,7,6);
Var         i, t : integer;
            flag : boolean;
Begin
        repeat
            flag:=true;
            for i:=1 to n-1 do
            if (a[i]<0) and (a[i+1]>=0) then begin
                                t:=a[i];
                    a[i]:=a[i+1];
                    a[i+1]:=t;
                                flag:=false;
                end;
        until flag;
End.
```

Завдання для самоконтролю.

Задача 6-1
Дана лінійна таблиця розмірності N. Знайти суму від'ємних елементів.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – суму від'ємних чисел.
Вхідні дані
5
2 -3 -6 1 2
Вихідні дані
-9

Задача 6-2
Дана лінійна таблиця розмірності N. Знайти кількість непарних чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – кількість непарних чисел.
Вхідні дані
5
2 -3 -6 1 2
Вихідні дані
2

Задача 6-3
Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти кількість непарних чисел, що мають парні індекси.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – кількість чисел.
Вхідні дані
5
2 3 5 1 2
Вихідні дані
2

Задача 6-4

Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти індекс останнього максимального числа.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – найбільший індекс максимального числа.
Вхідні дані
5
2 3 7 1 7
Вихідні дані
5

Задача 6-5
Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти індекс першого входження максимального числа.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – мінімальний індекс максимального числа.
Вхідні дані
5
2 3 7 1 7
Вихідні дані
3

Задача 6-6
Дана лінійна таблиця розмірності N, нумерація елементів починається з 1. Знайти суму індексів всіх непарних чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести одне число – сума індексів.
Вхідні дані
5
2 3 7 1 7
Вихідні дані
14

Задача 6-7
Дана лінійна таблиця розмірності N. Вивести спочатку всі парні числа, а потім – непарні.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск N чисел відповідно до умови задачі.
Вхідні дані
5
1 2 3 4 5
Вихідні дані
2 4 1 3 5

Задача 6-8
Дана лінійна таблиця розмірності N. Вивести ці числа у оберненому порядку.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск N чисел відповідно до умови задачі.
Вхідні дані
5
1 2 3 4 5
Вихідні дані
5 4 3 2 1

Задача 6-9
Дана лінійна таблиця розмірності N. Вивести «Yes», якщо числа впорядковані за зростанням і «No» - в іншому випадку.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести «Yes» або «No».
Вхідні дані
5
1 2 3 4 5

Вихідні дані
Yes

Задача 6-10
Дана лінійна таблиця A розмірності N. Вивести кількість таких індексів i, для яких виконується така умова: A[i]>A[j] для всіх j<i.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести кількість індексів.
Вхідні дані
5
1 1 1 4 5

Вихідні дані
3

Примітка. Перший елемент рахуємо як такий, що відповідає умові задачі.

Задача 6-11
Дана лінійна таблиця розмірності N. Впорядкувати ці числа за зростанням.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск N впорядкованих чисел.
Вхідні дані
5
3 5 3 4 1

Вихідні дані
1 3 3 4 5

Задача 6-12
Дана лінійна таблиця розмірності N. Знайти найбільшу кількість елементів, що повторюються.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести через пропуск два числа: число і кількість його повторів. Якщо таких чисел є кілька варіантів, то виводити те, що зустрілося у таблиці перше.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3

Вихідні дані
3 4

Задача 6-13
Дана лінійна таблиця розмірності N. Назвемо «близькими» такі числа, що модуль різниці їх індексів та модуль різниці самих чисел дорівнюють одиниці. Знайти кількість пар таких чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік вивести кількість пар.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
3

Примітка. Пари «близьких» чисел в порядку їх слідування в таблиці: (3,4), (1,2), (2,3).

Задача 6-14
Дана лінійна таблиця розмірності N. Знайти кількість пар «близьких» чисел та самі пари цих чисел.
ТУ. У першому рядку вхідного потоку задано N<100, у другому через пропуск N цілих чисел. У вихідний потік у першому рядку вивести K - кількість пар. У наступних K рядках виводимо по два числа через пропуск – пари «близьких» чисел.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
3
3 4
1 2
2 3

Задача 6-15
Дана лінійна таблиця розмірності N. Знайти та вивести всі прості числа, що є в даній таблиці.
ТУ. У першому рядку вхідного потоку задано N<10000, у другому через пропуск N цілих чисел не більших 100000. У вихідний потік у першому рядку вивести кількість K простих чисел, у другому – через пропуск самі прості числа.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
6
3 5 3 2 3 3

Задача 6-16
Дана лінійна таблиця розмірності N. Знайти та вивести всі прості числа, що є в даній таблиці. Ті числа, що повторюються враховувати лише один раз.
ТУ. У першому рядку вхідного потоку задано N<10000, у другому через пропуск N цілих чисел не більших 30000. У вихідний потік у першому рядку вивести кількість різних простих чисел, у другому – через пропуск ці числа.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
3
3 5 2

Задача 6-17
Дана лінійна таблиця розмірності N. Знайти та вивести кількість елементів більших за середнє арифметичне цих чисел.
ТУ. У першому рядку вхідного потоку задано N<10000, у другому через пропуск N цілих чисел. У вихідний потік у першому рядку вивести кількість К таких чисел, у другому – через пропуск самі числа.
Вхідні дані
10
3 5 3 4 1 1 2 3 1 3
Вихідні дані
6
3 5 3 4 3 3

Задача 6-18
Дана матриця розмірності NxM. Вивести рядок з максимальною сумою елементів.
ТУ. У першому рядку вхідного потоку міститься два числа N,M (0<N,M<=100). Дальше у N рядка через пропуск міститься по M чисел. У вихідний потік вивести рядок з M чисел з максимальною сумою.
Вхідні дані
3 4
4 3 1 1
1 1 1 2
5 6 3 1
Вихідні дані
5 6 3 1

Задача 6-19
Дана матриця розмірності NxN. Знайти суму елементів, що знаходяться вище головної діагоналі..
ТУ. У першому рядку вхідного потоку міститься число N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести суму.
Вхідні дані
3
4 3 1
1 1 1
5 6 3
Вихідні дані
5

Задача 6-20
Дана матриця розмірності NxM. Знайти суму «внутрішніх» елементів матриці. «Внутрішніми» елементами ми назвемо ті, що не знаходяться на крайньому рядку чи стовпцю.
ТУ. У першому рядку вхідного потоку міститься два числа N,M (0<N,M<=100). Дальше у N рядках через пропуск міститься по M чисел. У вихідний потік вивести суму або 0 якщо таких елементів немає.
Вхідні дані
3 4
4 3 1 1
1 1 1 2
5 6 3 1
Вихідні дані
2

Задача 6-21
Дана матриця розмірності NxN. Перевірити, чи є вона магічним квадратом. Магічним квадратом називається матриця, у якої сума рядків, стовпців та діагоналей рівна.
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести „Yes” або „No”.
Вхідні дані
3
1 1 1
1 1 1
1 1 1
Вихідні дані
Yes

Задача 6-22
Дана лінійна таблиця розмірності N (N<=100). Яку найменшу кількість елементів треба видалити, щоб утворилася зростаюча послідовність.
ТУ. У вхідному потоці дано N+1 ціле число: перше число N, а за ним через пропуск слідують інші числа. У вихідний потік вивести кількість чисел, які треба видалити.
Вхідні дані
5 1 1 2 2 3
Вихідні дані
2

Задача 6-23
Дано координати точок на площині. Вивести кількість різних трикутників, які можна отримати взявши дані точки за вершини.
ТУ. У першому рядку міститься число N (3<=N<100). У наступних N рядках – пари чисел, що є координатами точок Xi,Yi (-1000<=Xi,Yi<=1000). У вихідний потік вивести кількість трикутників.
Вхідні дані
4
-1 1
0 0
1 1
0 2
Вихідні дані
4

Задача 6-24
Дана матриця розмірності NxN, елементами якої є 0 та 1. Групу одиничок, що межує по горизонталі та по вертикалі з нулями назвемо «островом». Знайти кількість «островів».
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести кількість «островів».
Вхідні дані
4
1 0 0 1
0 1 1 0
1 1 1 1
1 1 1 1

Вихідні дані
3

Задача 6-25
Дана матриця розмірності NxN, елементами якої є 0 та 1. Знайти площу найбільшого «острова».
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). Дальше у N рядка через пропуск міститься по N чисел. У вихідний потік вивести ціле число – площу «острова».
Вхідні дані
4
1 0 0 1
0 1 1 0
1 1 1 1
0 0 0 0
Вихідні дані
6

Задача 6-26
Лабіринт задано матрицею з 0 та 1 розмірності NxM. Нулі показують проходи, а одиничкі перепони. Знайти довжину найкоротшого шляху з комірки (1,1) в комірку (N,M). Рухатися можна лише по горизонталі або вертикалі.
ТУ. У першому рядку вхідного потоку містяться N, M (0<N,M<=100). Дальше у N рядка через пропуск міститься по M чисел. У вихідний потік вивести довжину шляху або -1, якщо його не існує.
Вхідні дані
4 5
0 0 0 1 0
0 1 0 1 0
0 1 0 0 0
0 1 1 1 0
Вихідні дані
7

Задача 6-27
Дано N відрізків прямої. Знайти довжину їх спільної частини.
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). У наступних рядках вводиться по два цілих числа: координати лівого та правого кінця відрізку. Ліва координата завжди не більша правої. У вихідний потік вивести довжину спільної частини, або -1 якщо такої немає. Якщо спільною є лише точка, то виводити 0.
Вхідні дані
3
1 10
3 15
2 6
Вихідні дані
3

Задача 6-28
Дано N відрізків прямої. Знайти сумарну довжину частин прямої, покритої хоча би одним відрізком.
ТУ. У першому рядку вхідного потоку міститься N (0<N<=100). У наступних рядках вводиться по два додатні числа не більші 1000: координати лівого та правого кінця відрізку. Ліва координата завжди менша правої. У вихідний потік вивести довжину покриття.
Вхідні дані
3
1 10
11 12
2 6
Вихідні дані
10

Задача 6-29
Дано N цілих чисел. Треба знайти таких три числа, щоб їх добуток був максимальним.
ТУ. У першому рядку дано число N (3<=N<=100). У наступному рядку дано N чисел, кожне з яких по модулю не перевищує 1000. У вихідний потік вивести три числа у порядку їх слідування в таблиці.
Вхідні дані
9
10 3 5 1 7 9 0 9 -3
Вихідні дані
10 9 9

Задача 6-30
У початковий момент в лінійний масив записані числа від 1 до N (на і-му місці знаходиться число і). З елементами масиву проробляють таку операцію: міняють місцями числа, що мають індекси і та j.
ТУ. У першому рядку дано число N (3<=N<=100), у другому рядку ціле число K (0<K<10000) – кількість операцій, дальше у K рядках знаходяться по два числа – індекси елементів масиву. Вивести в рядок всі елементи масиву після K таких операцій.
Вхідні дані
10
2
1 3
5 3
Вихідні дані
3 2 5 4 1 6 7 8 9 10

Розділ 7. Масиви символів

Рядки символів в Паскалі обробляються як масиви символів (точніше, символьні вектори). Рядок, що вміщує один символ, є константа стан дартного типу Char. Рядок з n символів (n > 1) рахується константою типу ARRAY [1..n] of char. При цьому кожен символ є елементом масиву, до якого можна отримати доступ за допомогою індексу. Якщо рядкова змінна позначена, наприклад, як

```
x : array [1..12] of char;
```

то в програмі такій змінній можна присвоїти, наприклад, рядок символів: x := 'мова Паскаль';

Довжина рядка не може змінюватися в процесі виконання програми, тобто, якщо, наприклад, знадобиться присвоїти об'явленій змінній x рядок 'мова Бейсік', цей рядок прийдеться розширити на один пробіл, щоб його довжина дорівнювала 12:

```
x := 'мова Бейсік';
```

У протилежному випадку буде видано повідомлення про помилку.

Як і окремі символи, рядки символів можна порівнювати один з одним.
Основні співвідношення, які потрібно пам'ятати для цього, такі:
‘A’ < ‘B’ < ... <’ Y’ < ‘Z’
‘0’ < ‘1’ < ... < ‘8’ < '9'
‘A’<‘Z’< 'a' < ‘z’

Два рядки символів порівнюються таким чином: перший зліва символ першого рядка порівнюється з першим зліва символом другого рядка. Якщо він більший, то й весь перший рядок рахується більшим за другий рядок (чи навпаки). Якщо ж вони дорівнюють один одному, аналогічно порівнюються другі символи обох рядків і т.д. Це зручно для сортування рядків в алфавітному порядку. Рядки рахуються рівними між собою, якщо всі символи одного рядка та їх кількість повністю ідентичні іншому. Тому наступні співвідношення:

‘BASIC’<’PASCAL’

‘25’<’200’

‘PASCAL’<>’pascal’

‘ab’<’abc’

вірні, а

‘ABC’=’ ABC’ — хибне (у другом рядку константи є пропуск).

Майже всі операції з рядками, так само, як і зі всіма масивами, виконуються поелементно, за винятком оператора write (writeln). Для виведення всього рядка можна просто вказати ім'я рядкової змінної writeln(x);

Максимальна кількість символів у символьному масиві може сягати 65521.
Сторінки тексту можна представити двовимірним масивом:

```
Var St : array [1..60,1..80] of Char;
```

Тип STRING

На світанку свого становлення ЕОМ в основному використовувалися для математичних розрахунків. Після того, як комп'ютери отримали мож ливість розпізнавати символи (текст) і працювати з ними, сфера вико ристання ЕОМ для обробки інформації значно розширилась. Такий тип даних, як рядки символів, отримав дуже важливе значення. Тому для забезпечення роботи й розширення можливостей у цій сфері в Турбо Паскаль було введено ще один стандартний тип даних — STRING (рядок). Ця структура дуже схожа на масив символів, але дозволяє працювати з рядками змінної чи невідомої довжини. При об'явленні рядкової змінної типу STRING необхідно тільки вказати максимальну її довжину, яка не повинна перевищувати 255.

Наприклад:

```
VAR s1,s2 : string[40];
    line  : string[255];
```

Змінні s1 та s2 можуть мати довжину від 0 до 40, а line — від 0 до 255. У пам'яті змінні типу string займають одною коміркою більше, ніж їх максимальні довжини. Звернення до елементів рядка таке саме, як і в масивах символів, — по індексу. Індекс змінюється від 0 до максималь ної довжини. У нульовому елементі зберігається символ, код якого дорівнює поточній довжині рядка. А з першого по максимальний індекс можуть бути розташовані символи самого рядка.

Наприклад, якщо s1:='оператор', то ord(s1[0]) буде дорівнювати 8.
З даними типу string визначено декілька операцій, функ цій та процедур, які значно полегшують їх обробку:
1) «+» - операція конкатенації (склеювання). Якщо a=’транс’, b='форма', c=’тор’ то після s:=a+b+c змінна s дорівнюватиме ‘трансформатор’;
2) функція **length(s)** повертає довжину (ціле число) змінної s.
Length(s) еквівалентне ord(s[0]), але більш наочне;

3) функція **concat (s1,s2,...)** повертає рядок, що є конкатенацією рядків s1, s2, тобто s:=concat(s1,s2,...) еквівалентне s:=s1+s2+...;

4) функція **pos(s1,s2)** повертає позицію, з якої підрядок s1 вперше входить у рядок s2.
Наприклад, якщо s2='трансформатор', а s1='форма', то pos(s1,s2)=6;

другий приклад: pos('а',’інформатика’)=7; якщо рядка s1 нема в s2, то функція pos(s1,s2) поверне 0;

5) функція **copy(s, i, k)** повертає підрядок , якого отримано з рядка s копіюванням k символів, починаючи з i-го.
Наприклад: copy ('визначення',4,3)='нач';

6) процедура **delete(s,i,k)** видаляє з s, починаючи з i-го, k символів. Результат знову розміщується у змінну s.
Наприклад: якщо s=’значення', то після delete(s,4,2) змінна s дорівнюва тиме 'знання';

7) процедура **insert(s1, s2, i)** вставляє рядок s1 у рядок s2 перед сим волом s2[i].
Приклад: якщо s1='че', s2='знання', то після операції Insert(s1,s2,4) s2 дорівнюватиме 'значення';

8) процедура **str(x, s)** переводить числове значення x у відповідне рядкове значення і розташовує його у s.
Наприклад, якщо a=1993, то після str(a, st) st дорівнюватиме '1993';

9) процедура **val(s, v, c)** переводить рядкове значення s у ціле або дійсне значення (залежить від типу змінної v), яке розташовується у змінну v. c — це код (цілий) результату операції. Якщо операція пройш ла успішно, c=0, якщо ні — c дорівнюватиме номеру першого помил кового символа.

Приклади:
якщо s='125', після val(s, x, code) x=125, code=0;
якщо s='3R5' після val(s, x, code) x не визначено, а code=2

Процедури, на відміну від функцій, не повертають ніяких значень, а лише виконують деякі дії зі своїми аргументами. В результаті деякі аргументи можуть змінюватися.

Виведення рядкових величин, так само, як і символьних масивів, здійснюється операторами write та writeln, а введення можна виконати оператором readln (але не read).

Приклади програм.

Задача 1.

До введеного рядка (слова) побудувати обернений. Тобто, якщо ввести "інформатика", повинно бути виведено "акитамрофні".

```
var    s        : string;
       i, len   : integer;
       c        : char;
BEGIN
       readln(s);
       len:=length(s);
       for i:=1 to (len div 2) do  begin
              c:=s[len-i+1];
              s[len-i+1]:=s[i];
              s[i]:=c;
       end;
       writeln(s);
END.
```

Цикл обробки змінює індекс i від 1 до середини слова. Всередині циклу просто міняються місцями перший та останній символи, другий та другий з кінця і т.д. c — це допоміжна змінна для перестановок.

Задача 2.

Визначити, скільки разів у заданому рядку зустрічається символ ‘a’.

```
VAR   s        : string;
      i, l, k  : integer;
BEGIN
      readln(s);                    {вводимо рядок символів}
      l:=length(s);                 {знайшли його довжину }
      k:=0;                         {обнулили лічильник }
      for i:=1 to l do
            if s[i]='a' then k:=k+1;        { якщо є шуканий символ то збільшили лічильник
на 1}
      writeln(k);                   { вивели результат}
END.
```

Величини типу string зручніші за звичайні символьні масиви при оброб ці, але не дозволяють працювати з рядками, більшими за 255 символів. Однак можна робити так: більший текст зберігається у символьному ма сиві, з якого копіюються порції в окремі змінні типу string, обробляються і повертається знову в масив.

Завдання для самоконтролю.

Якщо в тексті наступних задачі не вказано довжину символьного рядка, то його довжину вважати не більшою 255 символів.

Задача 7-1
Дано рядок символів. Знайти кількість слів у даному рядку. Слова розділяються одним пропуском.
Ввід <
I love you!
Вивід >
3

Задача 7-2
Дано рядок символів. Знайти кількість слів у даному рядку. Слова розділяються довільною кількістю пропусків.
Ввід <
I love you!
Вивід >
3

Задача 7-3
Дано рядок символів. Визначити кількість літер латинського алфавіту.
Ввід <
I love you!
Вивід >
8

Задача 7-4
Дано рядок символів. Побудувати обернений рядок.
Ввід <
abcd
Вивід >
dcba

Задача 7-5
Дано рядок символів. Перевірити чи є він паліндромом. Паліндромом називаються рядки, що однаково читаються зліва направо і справа наліво. Вивести «Yes» або «No».
Ввід <
abba
Вивід >
Yes

Задача 7-6
Дано рядок символів. Знищити в ньому всі пропуски.
Ввід <
a b b a
Вивід >
abba

Задача 7-7
Дано рядок символів. Після кожного символу рядка вставити пропуск та вивести.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abba
Вивід >
a b b a

Задача 7-8
Дано рядок символів. Скопіювати цілу частину половини символів, що знаходяться на початку рядка та «приклеїти» їх в кінець.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abcde
Вивід >
abcdeab

Задача 7-9
Дано рядок символів. Додати до кінця даного рядка рядок, обернений до нього.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abc
Вивід >
abccba

Задача 7-10
Дано рядок символів. Добавити в початок даного рядка рядок, обернений до нього.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abc
Вивід >
cbaabc

Задача 7-11
Дано рядок символів. Над символами рядка виконуємо наступну операцію: переглядаємо символи починаючи з першого і кожен другий копіюємо в кінець. Виводимо результат.
ТУ. У стандартному вхідному потоці міститься рядок довжиною не більше 127 символів.
Ввід <
abc
Вивід >
abcbb

Задача 7-12
Дано рядок символів. Над символами рядка виконуємо наступну операцію: переглядаємо символи починаючи з першого і кожен другий переносимо в кінець. Виводимо результат.
Ввід <
abcd
Вивід >
acbd

Задача 7-13
Дано рядок символів. Перевірити чи є в даному рядку символи «13». Вивести «Yes» або «No».
Ввід <
abcd123abc
Вивід >
No

Задача 7-14
Дано рядок символів. Впорядкувати символи даного рядка за зростанням.
Ввід <
fedba
Вивід >
abdef

Задача 7-15
Дано рядок символів. Визначити кількість малих літер латинського алфавіту.
Ввід <
Abba
Вивід >
3

Задача 7-16
Дано рядок символів. Визначити кількість цифр у рядку.
Ввід <
Ab’ba1o o4
Вивід >
2

Задача 7-17
Дано рядок символів, що містить один символ «-». У першому рядку вихідного потоку вивести частину рядка до символу «-», у другому рядку – частину, що знаходиться після цього символу. Символ «-» не виводити в жодній частині.
Ввід <
Abba-Yes
Вивід >
Abba
Yes

Задача 7-18
Дано рядок виду: a#b=, де a та b деякі цілі додатні числа не більші 10000, а символ «#» - одна із операцій: «+», «-», «*». Знайти значення виразу s та у вихідний потік вивести рядок a#b=s.
Ввід <
2+3=
Вивід >
2+3=5

Задача 7-19
Дано рядок символів, що містить числа. У вихідний потік вивести всі числа по одному в кожному рядку.
Ввід <
Abba1980 Yes5NO1990 Ok2 5!
Вивід >
1980
5
1990
2
5

Задача 7-20
Дано рядок символів, що містить числа. У вихідний потік вивести суму цих чисел.
ТУ. Числа та сума не перевищують $2*10^9$.
Ввід <
Abba1980 Yes5NO1990 Ok2 5!
Вивід >
3982

Задача 7-21
Рядок містить не менше двох слів. Впорядкувати слова в алфавітному порядку.
ТУ. У вхідному потоці міститься рядок слів розділених пропусками. У вихідний потік вивести впорядковані слова по одному слову в рядок.
Ввід <
f e d b a
Вивід >
a
b
d
e
f

Задача 7-22
Дано два символьних рядки. Чи можна за допомогою символів першого рядка скласти другий рядок. Кожен символ першого рядка можна використовувати не більше частоти його входжень у другий рядок. У вихідний потік вивести «Yes» або «No».
Ввід <
abcdefedcba
abba
Вивід >
Yes

Задача 7-23
Дано два символьних рядки, що містять лише символи латинського алфавіту. Чи можна другий рядок отримати шляхом перестановки символів першого рядка. У вихідний потік вивести «Yes» або «No».
Ввід <
abcd
dbac
Вивід >
Yes

Задача 7-24
Дано два символьних рядки. Чи є перший рядок підрядком другого? Вивести позицію першого входження або 0 у випадку, коли такого підрядка не існує. Якщо входжень є декілька, то вивести позицію першого входження.
Ввід <
abcd
abcabcddd
Вивід >
4

Задача 7-25
Дано два символьних рядки. Знайти кількість входжень першого підрядка в другий. У стандартний вихідний потік вивести одне число – кількість входжень.
Ввід <
abca
abcabca
Вивід >
2

Задача 7-26
Дано послідовність слів, що розділяються пропусками. Треба вивести перші літери всіх слів. Якщо перша буква мала, то виводити її великою. При виконанні цього завдання слова “and”, “the”, “of” будемо ігнорувати.
ТУ. У стандартному вхідному потоці містяться слова, що складаються із літер латинського алфавіту. У стандартний вихідний потік вивести в одному рядку великі літери, що відповідають умові задачі.
Ввід <
the united states of america
Вивід >
USA

Задача 7-27
Дано речення, що закінчується крапкою в кінці. Вивести частоту входження символів латинського алфавіту у даний текст. Виводити лише символи, що зустрілися в тексті хоча би один раз. Формат виведення має бути зрозумілим із прикладу. Великі та малі літери не розрізняти.
Ввід <
Abcdaabcd.
Вивід >
a-3
b-2
c-2
d-2

Задача 7-28
Дано речення, що закінчується крапкою в кінці. Вивести символ, що повторюється у тексті найчастіше. У випадку наявності кількох таких символів, вивести той, що перший іде в алфавіті. Великі та малі літери не розрізняти.
ТУ. У стандартному вхідному потоці міститься речення, що закінчується крапкою. У стандартний вихідний потік вивести у першому рядку символ, у другому – кількість його входжень.
Ввід <
Abcdaabcd.
Вивід >
a
3

Задача 7-29
Нам відома система кодування слів, що називається «голосні латинські». Всі голосні букви видаляються із слова і дописуються в його кінець в тому ж порядку, що є вони у слові. Голосні букви є: ‘a’, ‘e’, ‘i’, ‘o’, ‘u’. Треба закодувати слово за такою системою.
ТУ. У стандартному вхідному потоці міститься слово довжиною не більше 255 символів, що складається лише з великих та малих літер латинського алфавіту. У вихідний потік вивести закодоване слово.

Вхід <
Application
Вивід >
pplctnAiaio

Задача 7-30
В заданому тексті знайти середню довжину слова. Слова складаються з букв латинського алфавіту та можуть розділятися цифрами, пропусками та символами пунктуації.
ТУ. У стандартному вхідному потоці міститься текст довжиною не більше 30000 символів, що закінчується символом «#». У стандартний вихідний потік вивести дійсне число з точністю до десятих.

Вхід <
This is div easy problem.#
Вивід >
4.0

Задача 7-31
Дано символьні рядки, що можуть містити числа. Вивести ці числа через пропуск. Крапка в тексті може бути лише у десяткових числах.
ТУ. У вхідному потоці містяться рядки довжиною не більше 255 символів. Кількість рядків не більша 1000. У вихідний потік вивести числа через пропуск у відповідних рядках.

Вхідні дані
Julja 25, Vasja 30.3
2+3=5

Вихідні дані
25 30.3
2 3 5

Задача 7-32
Дано текст довжиною не більше 30000 символів. Всі слова, що починаються з великої латинської літери вивести в окремому рядку великими літерами. Слово може складатися і з однієї літери, а самі слова розділяються будь-яким символом, що не є літерою латинського алфавіту.
ТУ. У вхідному стандартному потоці міститься текст. У вихідний потік вивести по одному у рядку великими літерами слова, що задовольняють умову задачі.
Вхідні дані
Hi-Tech

Вихідні дані
HI
TECH

Задача 7-33
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що починаються та закінчуються голосною літерою.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.

Вхідні дані
4
abbi
then
else
o

Вихідні дані
3

Задача 7-34
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що містять хоча би дві однакові літери.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.

Вхідні дані
4
abbi
then
else
o

Вихідні дані
2

Задача 7-35
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що містять лише різні літери.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.

Вхідні дані
4
abbi
then
else
o

Вихідні дані
2

Задача 7-36
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, що є паліндромами. Паліндромами називається слова, що однаково читаються зліва направо і справа наліво.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.

Вхідні дані
4
abba
then
else
o

Вихідні дані
2

Задача 7-37
Задано N слів, що складаються з малих латинських літер. Знайти кількість слів, літери яких не порушують зростаючий алфавітний порядок.
ТУ. У першому рядку стандартного вхідного потоку знаходиться N (0<N<1001), у наступних N рядках міститься по одному слову. У вихідний потік вивести кількість слів, що відповідають умові задачі.

Вхідні дані
4
abb
then
else
aabbccddzz

Вихідні дані
2

Задача 7-38
Дано набір із символьних рядків, що складаються із круглих дужок: «)», «(». Для кожного рядка визначити чи являє він правильну дужкову послідовність.
ТУ. Перший рядок стандартного вхідного потоку містить натуральне число N (N<100) – кількість символьних рядків. У наступних N рядках містяться символьні рядки. Для кожного рядка в окремому рядку вихідного потоку вивести «Yes» у випадку правильної діжкової послідовності або «No» в іншому випадку.

Вхідні дані
3
((()))
(()))(
)(

Вихідні дані
Yes
No
No

Задача 7-39

Дано набір із символьних рядків, що складаються із дужок: «)», «(», «[», «]». Для кожного рядка визначити чи являє він правильну дужкову послідовність.

ТУ. Перший рядок стандартного вхідного потоку містить натуральне число N (N<100) – кількість символьних рядків. У наступних N рядках містяться символьні рядки. Для кожного рядка в окремому рядку вихідного потоку вивести «Yes» у випадку правильної діжкової послідовності або «No» в іншому випадку.

Вхідні дані
3
(([]))
[()()]
[)(]

Вихідні дані
Yes
Yes
No

Задача 7-40

Дано набір із символьних рядків, що складаються із 26 видів дужок. Дужки домовимося представляти символами латинського алфавіту: велика літера відкриває відповідну дужку, а мала – її закриває. Для кожного рядка визначити чи являє він правильну дужкову послідовність.

ТУ. Перший рядок стандартного вхідного потоку містить натуральне число N (N<100) – кількість символьних рядків. У наступних N рядках містяться символьні рядки. Для кожного рядка в окремому рядку вихідного потоку вивести «Yes» у випадку правильної діжкової послідовності або «No» в іншому випадку.

Вхідні дані
3
ABCcba
FFFfffaA
ABbaAa

Вихідні дані
Yes
No
Yes

Розділ 8. Рекурсивні функції та процедури

Поняття рекурсії.

Об'єкт називається рекурсивним, якщо він містить сам себе або визначений за допомогою самого себе.

Наприклад:

Натуральне число:

1) 1 є натуральне число
2) ціле число, що слідує за натуральним є натуральне число.

Факторіал натурального числа:

1) 0! = 1
2) якщо n>0 , то n! = n*(n-1)!

Очевидно, що потужність рекурсії пов'язана з тим, що вона дозволяє визначити нескінчену множину об'єктів за допомогою кінцевого висловлення. Так само нескінчені обчислення можна описати за допомогою кінцевої рекурсивної програми, навіть якщо ця програма не містить явних циклів.

Необхідний та достатній засіб для рекурсивного представлення програм – це опис процедур, або підпрограм, тому що він дозволяє присвоїти якому-небудь оператору ім'я, за допомогою якого можна викликати цей оператор.

Головна програма може викликати будь-які підпрограми, які в свою чергу можуть викликати інші підпрограми. Отже, будь-яка підпрограма має принципову можливість звертатися до самої себе. Якщо процедура P містить явне звертання до самої себе, то вона називається прямо *рекурсивною*, якщо P містить звертання до процедури T, яка містить звертання до P, то P називається *непрямо рекурсивною*.

В рекурсивному визначенні не повинно бути "зачарованого кола", коли об'єкт безкінечно визначається за допомогою самого себе або за допомогою інших, але визначених через нього. Наприклад:

"Де ти гроші береш?" – "У тумбочці". – "А там вони звідки?" – "Дружина кладе". – "А в неї звідки?" – "Я даю" – "А де ти береш?" – "В тумбочці..."

В цьому старому анекдоті не називається справжнє джерело грошей. Якщо через A, B, C позначити чоловіка, його дружину і тумбочку, то переміщення грошей зображується так:

$A \leftarrow C \leftarrow B \leftarrow A...$

Щоб подібна "дурна нескінченість" не виникала в рекурсивному визначенні, повинні виконуватися наступні умови:

- множина об'єктів, що визначаються, є частково впорядкованою;
- кожна послідовність, що спадає за цією впорядкованістю елементів, закінчується мінімальним елементом;
- мінімальні елементи визначаються не рекурсивно;
- не мінімальні елементи визначаються за допомогою елементів, які менші від них за цією впорядкованістю.

Нескладно дійти до висновку, що визначення факторіалу задовольняє ці умови, а визначення із наведеного анекдоту – ні.

Розглянемо такий приклад рекурсивного означення:

1. Стадо із N корів – це 3 корови.
2. Стадо із N корів – це стадо із N-1 корови і ще одна корова.

Попробуємо використати це означення для перевірки, чи є дещо стадом корів. Нехай у нас є група із 5 корів, позначимо її K5. Об'єкт K5 не задовольняє перший пункт означення, оскільки 5 корів – це не 3 корови. Згідно другого пункту K5 – стадо, якщо там є одна корова, а інша частина K5, назвемо її K4, - також стадо корів. Розв'язок відносно об'єкта K5 відкладається, поки не буде прийняте рішення відносно об'єкту K4. Об'єкт K4 знову не

підходить під перший пункт, а згідно другого пункту К4 – стадо, якщо об'єкт К3, отриманий із К4 шляхом відділення однієї корови, також стадо. Розв'язок відносно К4 також відкладається. Нарешті, об'єкт К3 задовольняє перший пункт означення, і ми можемо твердо сказати, що К3 – стадо корів. Тепер і про К4 можна стверджувати, що це є стадо, а значить, і К5 також є стадом.

Будь-яке рекурсивне означення складається з двох частин. Перша визначає поняття через нього ж, інша частина – через інші поняття.

Як приклад розглянемо функцію обчислення факторіалу натурального числа.
Ми вже розглянули рекурсивне визначення факторіалу n!=(n-1)!*n , тобто 6! = 6*5!
Щоб знайти n! Потрібно застосувати функцію факторіала до числа n-1, потім – до n-2 і т.д. до 1.
Обов'язковим елементом в описі будь-якого рекурсивного процесу є деяке твердження, яке визначає умову завершення рекурсії – опорна умова або *"якір"* рекурсії. В опорній умові може бути задане яке-небудь фіксоване значення, яке обов'язково досягається у ході рекурсивного обчислення і яке дозволяє організувати своєчасну зупинку процесу.
Застосовано до n! це буде рівність 1! = 1.
Напишемо підпрограму (функцію):

```
Function  factorial (n : integer) : Longint;
    begin
      if n =1 then factorial := 1
              else  factorial := factorial(n-1)*n
    end;
```

Припустимо, що необхідно обчислити факторіал числа 4. Щоб знайти factorial(4), необхідно знайти factorial(3), що досягається рекурсивним звертанням функції до самої себе. Однак це означає, що обчислення 4! тимчасово призупиняється до тих пір, доки не буде отриманий результат виклику factorial(3). Як тільки це відбудеться, обчислення factorial(4) зможе завершитися: величина factorial(3) буде просто помножена на 4. Однак для обчислення factorial(3) необхідно мати значення factorial(2), яке повинно помножитися на 3, але для обчислення factorial(2) потрібен factorial(1). Обчислити factorial(1) дуже легко: за опорним визначенням 1! = 1. Саме з цим значенням ми продовжимо обчислення factorial(2) як factorial(1)*2. Далі отриманий результат повертається до обчислення factorial(3). Нарешті, можна завершити призупинене обчислення factorial(4).

Такі виклики називають вкладеними. Таким чином, виклик з аргументом 4 породжує ще три вкладених виклики. Взагалі, при виклику цієї функції з аргументом n породжується ще n – 1 виклики, і загальна кількість незакінчених викликів досягає n. Так відбувається рекурсивне занурення. Після досягнення якоря рекурсії починається рекурсивне спливання.

Таким чином, глибиною рекурсії називається максимальна кількість незакінчених рекурсивних викликів при її виконанні. При виконанні виклику з глибиною рекурсії m одночасно існують m екземплярів локальної пам'яті. Кожен екземпляр має певний розмір, і якщо глибина рекурсії буде дуже великою, то автоматичної пам'яті, що надається процесу виконання програми, може не вистачити. Загальною кількістю вкладених викликів, можна назвати кількість викликів, що породжуються викликом рекурсивної підпрограми. Це істотно впливає на час виконання виклику.
Використання рекурсивних підпрограм потребує обережності і уміння оцінити можливу глибину рекурсії та загальну кількість викликів, оскільки це може привести до використання всієї доступної пам'яті і аварійного завершення програми.

Локальні змінні в рекурсивних програмах.

В рекурсивних підпрограмах допускається оголошувати та використовувати локальні змінні. Локальні змінні – внутрішні змінні підпрограми, вони створюються в момент виклику підпрограми і автоматично знищуються при її завершенні.

```
var k : longint;
Function  factorial (n : integer) : Longint;
var p : Longint;
begin
        if n =1 then p := 1
                  else  p := factorial(n-1)*n;
        factorial:=p
end;
begin
  readln(k);
  writeln(factorial(k));
end.
```

При використанні локальних змінних необхідно розуміти:

а) при кожному виклику функції створюється новий екземпляр p;

б) всі екземпляри p накопичуються у внутрішньому стеку та

в) у будь який момент обробці доступний тільки один, хронологічно останній екземпляр змінної p, який

г) по завершенню чергового рекурсивного виклику автоматично знищується.

Задача про перевертання рядка.

Розглянемо приклад рекурсивної функції, де обробляються нечислові дані. Функція Reverse повинна переставити літери заданого рядка у зворотному порядку. Почнемо з опису рекурсивного алгоритму. Нехай st – літерний рядок. Тоді його перевернутим значенням буде reverse(st), яке визначається наступним чином:

а) якщо st – одна літера, то reverse(st) співпадає з st;

б) якщо st складається з літери ch і групи літер rst, то reverse(st) є конкатенація (додавання) reverse(rst) і ch.

```
var s : string;
Function reverse (st : string) : string;
      var ch : char;
          Rst : string;
 begin
     if length(st) = 1 then reverse := st {“якір” рекурсії}
                       else  begin
                               ch := st[1];
                               delete(st,1,1);
                               rst := reverse(st);
                               reverse := concat(rst, ch)
                             end
 end;
 begin
    readln(s);
    writeln(reverse(s));
 end.
```

Нехай рядком, який потрібно перевернути буде слово ‘кіт’. Обробка починається з перевірки довжини st, у зв’язку з тим, що довжина цього слова не дорівнює 1, виконується пункт else. Тут ми заносимо у змінну ch першу літеру ’к’, ця літера видаляється із st, де залишається ’іт’. Після чого слідує рекурсивний виклик reverse(st) - потрібно перевернути

’іт’ (значення st в цей момент). Довжина ’іт’ = 2, тому знов виконується пункт else, де ch отримує значення ’о’, а у st залишається ’т’. Знов виконується рекурсивний виклик reverse(st) з параметром ’т’. (Рекурсивне занурення) Довжина st = 1, тому виконується пункт завершення рекурсії then. Літера ’т’ присвоюється імені reverse і стає значенням функції. Тепер ми попадаємо у точку попереднього виклику, де завершується тимчасово призупинену операцію присвоювання: rst отримує значення ’т’. Далі в кінець rst приєднується літера ’і’, утворюючи рядок ’ті’, який присвоюється reverse. Після цього потрапляємо у перший виклик. До ’ті’ додається ’к’, який зберігався у ch. Отримане слово ’тік’ є кінцевим значенням функції reverse, воно і повертається у головну програму. (Рекурсивне спливання).
Рекурсивними можуть бути не тільки функції, але і процедури. Наприклад, задачу про перевертання рядка можна було розв’язати у формі рекурсивної процедури reverse. Найпростіше зробити так, щоб у неї були два параметри. Перший st – являє собою вихідний рядок, в другий rst – функція буде розміщувати результуючий рядок. Таким чином, заголовок процедури має вигляд:

Procedure reverse (st : string; var rst : string);
Виклик процедури: reverse(s, rs);
Пропонується цю процедуру написати самостійно.

Числа Фібоначчі.

Розберемо ще одну типову задачу. Числа Фібоначчі можна визначити наступним чином: перше і друге дорівнюють 1; кожне наступне, починаючи з третього є сума двох попередніх: 1, 1, 2, 3, 5, 8...
Щоб отримати, наприклад, 20 перших членів послідовності Фібоначчі, напишемо програму, де використаємо структуру 20-елементного масиву:

```
var
   Fib : array[1..20] of word;
    T   : word;
begin
  Fib[1] := 1; Fib[2] := 1;
  For t := 3 to 20 do Fib[t] := Fib[t – 1] + Fib[t – 2];
…
```

Зберігаючи основний принцип обчислення, підійдемо до задачі про числа Фібоначчі з точки зору рекурсивних визначень. Припустимо, нам потрібно отримати n-е число Фібоначчі. Як і в будь-якому рекурсивному процесі, ми повинні сформулювати умову виходу з нього, а також вказати спосіб вираження одного кроку через попередній, більш простий.

```
Function Fib (n : integer) : integer;
 begin
    if (n = 1) or (n = 2) then fib := 1
                          else fib := fib(n-1) + fib(n-2)
 end;
```

Припустимо, потрібно обчислити 6-е число Фібоначчі, тобто викликати функцію fib(6). Простежимо, наскільки довгий ланцюжок породжує такий виклик. При першому вході в функцію (n = 6) значення fib визначається як fib(5) + fib(4). Але, щоб отримати fib(5), потрібно обчислити fib(4)+fib(3); обчислення fib(4) розгортається у fib(3) + fib(2), а fib(3) - у fib(2) +fib(1). Нарешті, з’ясовується, що fib(2) і fib(1) мають фіксовані значення 1 згідно умові припинення рекурсії. Таким чином, ми розглянули лише першу частину обчислення і тільки для fib(5) – фаза рекурсивного “занурення” від fib(5) до fib(1), за якою слідує фаза “спливання”, яка супроводжується новими “зануреннями” и поступовим формуванням числових результатів: fib(3) = 1+1 = 2; fib(4) = 2+fib(2) = 2+1 =3; fib(5) = 3+fib(3) =

3+fib(2)+fib(1) = 3+1+fib(1) = 4+1 =5. А потрібно ще пройти аналогічний рекурсивний шлях для обчислення fib(4) для того, щоб його значення додати до значення fib(5), щоб підрахувти кінцеве значення fib(6). Після розміркування стає зрозуміло, що дана рекурсивна функція дуже неефективна. Але вона добре демонструє основні принципи рекурсії.

Розглянемо декілька простих прикладів, що ілюструють техніку написання рекурсивної логіки.

1. Переведення натурального числа із десяткової системи числення у двійкову.

```
Procedure Rec (n : integer);
 begin
   if n>1 then Rec(n div 2);
   write(n  mod  2)
end;
```

2. Визначити, чи є задане натуральне число простим. Тобто, число N не ділиться на жодне число, яке більше, або дорівнює 2, але менше n. Нехай m - всі дільники числа n. M знаходиться в діапазоні від 2 до n – 1.
Твердження істинно у двох випадках:
1. якщо m = n.
2. n не ділиться на m, і істинно твердження для чисел від m + 1 до n – 1.

```
Function Simple (m,n : integer) : Boolean;
   begin
       if m = n then Simple := True
                 Else Simple := (n mod m <>0) and Simple(m+1,n)
   end;
```

Перший виклик функції – Simple(2,n), де n – число, яке перевіряється.

3. Обчислити найбільший спільний дільник двох чисел.

```
Function Nod ( a, b : integer) : integer;
    begin
       if  a=b then Nod := a
               Else if a>b then Nod:=Nod(a – b, b)
                           Else Nod:=Nod(a, b – a)
   end;
```

4. Знайти максимальний елемент в глобальному масиві A.

```
Procedure Max (n : integer; var x : integer);
 begin
   if n = 1 Then x:=A[1]
            Else begin
                  Max (n-1, x);
                  if A[n]>x Then x := A[n]
                 end
 end;
```

5. Піднесення цілого числа a в цілу невід’ємну степінь n рекурсивно реалізується так:

```
   Function Pow (a,n : integer) : integer;
     begin
       if n = 0 Then Pow :=1
                Else if n mod 2 = 0 Then Pow := Pow(a*a, n div 2)
                                    Else Pow := Pow(a,n-1)*a
      end;
```

6. Процедура введення з клавіатури послідовності чисел, яка обмежена нулем та виведення її у зворотному порядку має вигляд:

```
Procedure Solve;
 var n : integer;
begin
   write('n='); readln(n);
    if n <> 0 Then Solve;
    write( n: 5)
end;
```

7. Генерування перестановок.

Перестановкою із n елементів назвемо впорядкований набір із n різних натуральних чисел, що належать інтервалу від 1 до n. Нехай n = 3. Перерахуємо всі перестановки з 3 елементів: (1 2 3), (1 3 2), (2 1 3), (2 3 1), (3 1 2), (3 2 1).

Кількість перестановок з n елементів дорівнює n! = 1*2*3* …*n.

Розглянемо, чим характеризується вказаний порядок виведення перестановок. До певної позиції сусідні перестановки співпадають (в даному випадку до другої (числа можуть не співпадати вже у першій позиції), а в цій позиції число в другій послідовності більше, ніж число в першій послідовності. Такий порядок називається лексикографічним. Розглянемо алгоритм генерування всіх перестановок в лексикографічному порядку. Запишемо перестановку в масив A.

1) Переглядаємо масив з кінця, знаходимо перший елемент, який менше свого сусіда праворуч, і запам'ятовуємо його номер i. Якщо такого елементу не знайдеться, тобто i = 0, тоді ця перестановка остання і потрібно закінчити виконання алгоритму. (Всі елементи, що знаходяться праворуч від A[i] впорядковані за спаданням).

2) Переглядаємо масив, починаючи з номеру i+1, виділяючи всі елементи, які більші A[i], і серед них відшукуємо найменший. Нехай його номер – j.

3) Міняємо місцями A[i] і A[j].

4) Перевертаємо "хвіст" перестановки, від A[i+1] до A[n], міняючи місцями A[i+1] з A[n], A[i+2] з A[n-1] і т.д.

Наприклад:

Нехай A = (2, 6, 5, 8, 7, 4, 3, 1)

1) Переглядаємо перестановку з кінця: 3>1, 4>3, 7>4, 8>7, 5<8.

Таким чином, i = 3, A[i] = 5

2) Переглядаємо перестановку, починаючи з 4-го елемента: 8>5, 7>5, 4<5. В зв'язку з тим, що "хвіст" перестановки впорядкований за спаданням, то далі перевірку робити не потрібно. Шуканий елемент має номер j = 5, A[5] = 7.

3) Міняємо місцями A[3] і A[5] : A = (2, 6, 7, 8, 5, 4, 3, 1)

4) Перевертаємо "хвіст" перестановки : A = (2, 6, 7, 1, 3, 4, 5, 8).

Запишемо ітеративний алгоритм:

```
Const n = 4;
var A : array[1..n] of integer;
     L, i, j, n1, n2, vsp : integer;
 begin
    For i:=1 to n do A[i]:=i;
    i:=n;
    While i > 0 do  begin
         For L:=1 to n do write(A[i], ' ');
         writeln;
         J := n – 1;
         While (A[j]> A[j+1]) and (j>0) do j := j – 1; {шукаємо з кінця перший елемент, який
менше його сусіда праворуч}
         i:=j; {запам'ятовуємо його номер, або 0 для останньої перестановки}
         if  i > 0 then  begin  j := n;
```

While A[i] > A[j] do j := j-1; {в зв'язку з тим, що "хвіст" перестановки впорядкований, можна переглядати з j = n і переходити до сусідніх ліворуч елементів. Шуканий – той, для якого виконується умова A[i] < A[j].}

```
          Vsp := A[i];
          A[i] := A[j];
          A[j] := vsp;
          N1:=i+1;
          N2 := n;
          While N1 < N2 do   begin          {перевертаємо "хвіст" перестановки}
                Vsp := A[N1];
              A[N1] := A[N2];
              A[N2] := vsp;
               N1 := N1+1;
               N2 := N2 – 1;
            end
        end
  end
end.
```

Напишемо тепер рекурсивну процедуру генерації перестановок від 1 до n. В чому тут суть рекурсії? Фіксуємо на першому місці наступне число і генеруємо всі перестановки цього виду. При цьому використовуємо таку ж саму схему. Фіксуємо число на другому місці и генеруємо всі перестановки вже з фіксованими елементами на першому і другому місцях.
Необхідно зрозуміти, що після генерації всіх перестановок з фіксованим елементом в позиції t перестановка повертається у вихідний стан і здійснюється новий обмін значеннями елемента із позиції з номером t+1 і i+1.

```
Const n = 4;
 Type Mas = array[1..n] of integer;
  var A : mas;
        i : integer;
  Procedure Swap (var a, b : integer);
     var c : integer;
      begin c:=a; a:=b; b:=c  end;
  Procedure Print;
    var i : integer;
   begin
     For i:=1 to n do write(a[i] : 3);
     writeln
   end;
  Procedure Solve (t : integer);
    var i : integer;
   begin
     if t >=n Then Print
                Else for i := t+1 to n do
                      begin
                         Swap(A[t+1],A[i]);
                         Solve(t+1);
                          Swap(A[t+1],A[i])
                       end;
   end;
begin
 For i:=1 to n do A[i]:=i; {перша перестановка – 1 2 3 ... n}
 Solve(0) {перший виклик}
end.
```

Переробимо алгоритм перебору всіх перестановок, використовуючи структуру множин. Всі перестановки можна отримати, вибираючи з множини один елемент всіма можливими способами і приєднуючи до нього по черзі перестановки із елементів, що залишилися. Умова завершення рекурсії – єдина перестановка із одного елементу – це сам елемент. Множину елементів, що переставляються зробимо параметром процедури. Всі перестановки будемо почергово виводити у зовнішньому масиві A із N цілих чисел.

```
Const   n = 4;
Type    intSet = set of 1..N;
var     A : array[1..N] of integer;
 Procedure Solve (S : intSet; K : integer);
    var i :integer;
     begin
       For i := 1 to N do
          if i in S then
                    begin
                      A[K] := i;
                       Solve(S – [i], K+1)
                     end;
         end;
      begin
        Solve([1..N], 1)
       end.
```

Ця програма тільки будує перестановки у масиві A і навіть не виводить їх на екран. Для виведення на екран чергової перестановки програму потрібно доповнити процедурою Print.

10. Сортування масиву.

Останній приклад показує використання рекурсії для побудови алгоритму сортування значень. Напишемо рекурсивну процедуру сортування масиву цілих чисел QuickSort, яка реалізує наступний алгоритм: на деякому відрізку масиву вибирається центральне (серединне) значення; всі елементи з лівої частини відрізку, що перевищують центральне значення, переміщуються в праву частину, і навпаки. Наступним кроком (для якого використовуються рекурсивні виклики цієї ж процедури) буде повторення алгоритму для обох частин відрізка.

```
Const    max = 100;
var      List : array[1..max] of integer;
          i : integer;
Procedure QuickSort (Left, Right : integer);
   {Процедура впорядковує за зростанням значення з масиву List в діапазоні індексів
Left..Right}
 var  i, j : integer; {робочі індекси}
       X, y : integer; { робочі елементи масиву}
begin
  i := Left; J := Right;
  X := List[ (Left + Right) div 2]; {центральне значення}
  Repeat
      {шукаємо в лівій та правій половинах відрізка значення, які належить перемістити у
протилежну половину}
     while List[i] < x do inc(i);
     while List[j] >x do dec(j);
             {міняємо місцями знайдені значення}
     if  i <= j then begin
            y := List[i];
            List[i] := List[j];
```

```
            List[j] := y;
{переходимо до наступних значень із обох половин}
            inc(i); dec(j)
          end;
  Until i > j;   {повторюємо, поки в частинах відрізка є значення, що підходять для переносу}
if Left < j then QuickSort(Left, j);
if i < Right then QuickSort( i, Right)
end;
begin
   For i := 1 to max do  readln(List[i]);
QuickSort(1, max);
For i:=1 to max do write(List[i] : 4);
writeln
end.
```

11. Лабіринт

Лабіринт задається квадратною матрицею, де 1 показує проходи, а 0 – відсутність ходу. Знайти всі можливі виходи з центру лабіринту.

```
type
lab=array[1..10,1..10] of char;
var l:lab;
i,j,x,y,n:integer;
procedure vyvid(l:lab); {вивід шляху}

 begin
  for i:=1 to n do   begin
    for  j:=1 to n do
      write(l[i,j]);
     writeln;
   end;
   writeln('__________');
 end;
 procedure step(l:lab;x,y:integer);
 begin
   if l[x,y]='1'  {якщо шлях вільний}
      then begin
            l[x,y]:='+'; {робимо крок}
            if (x=1) or(x=n) or(y=1)or(y=n) then vyvid (l) виводимо, якщо досягли краю
      else   begin     {пробуємо йти в кожному напрямку}
              step(l,x+1,y);
              step(l,x,y+1);
              step(l,x-1,y);
              step(l,x,y-1);
             end;
      l[x,y]:='1'; {повертаємося назад}
      end;
  end;
BEGIN
          readln(n);
          for i:=1 to n do
                    for j:=1 to n do
                              readln(l[i,j]);
          step(l,n div 2+1,n div 2+1);{починаємо з середини}
end.
```

Розглянувши велику кількість прикладів, можна зробити наступний висновок: потрібно уникати рекурсії, коли є очевидний ітеративний розв'язок поставленої задачі. Це не означає, що потрібно завжди позбавлятися рекурсії. В багатьох випадках маніпулювання зі стеком рекурсії так сильно завантажує програму, що зрозуміти її стає дуже важко. Таким чином, алгоритми, які за своєю природою швидше рекурсивні, ніж ітеративні, потрібно представляти у вигляді рекурсивних процедур.

Завдання для самоконтролю

Задача 8-1
Дано функцію піднесення до степеня.

$$a^n = \begin{cases} 1, & n = 0 \\ a^{n-1} * a, & n > 0 \end{cases}$$

Знайти значення функції, для даних a та n.
ТУ. У стандартному вхідному потоці дано цілі a та n (0<a<10, 0<=n<50). У вихідний потік вивести значення функції. Гарантується, що результат не буде перевищувати 10^{18}.

Вхідні дані
2 4

Вихідні дані
16

Задача 8-2
Знайти значення функції

$$Y(n) = \begin{cases} 2, & n=1 \\ 2*Y(n-1) \end{cases}$$

ТУ. У стандартному вхідному потоці дано ціле n (0<n<50). У вихідний потік вивести значення функції.

Вхідні дані
2

Вихідні дані
4

Задача 8-3
Знайти найбільший спільний дільник двох чисел A, B. Написати рекурсивну функцію або процедуру.
ТУ. У вхідному потоці задається через пропуск два цілих числа не більших 10^9. У вихідний потік вивести їх найбільший спільний дільник.

Вхідні дані
6 9

Вихідні дані
3

Задача 8-4
Послідовність чисел задається таким рекурентним співвідношенням:
G(n)=2*G(n-2), G(0)=0, де G(1)=1.
Для даного n знайти G(n).
ТУ. У вхідному потоці дано ціле n (0<=n<100). У вихідний потік вивести n-й член послідовності.

Вхідні дані
3

Вихідні дані
2

Задача 8-5
Обчислити значення функції:

$$G(n) = \begin{cases} 0, & n=0 \\ G(n-1)+2, & n>0 \end{cases}$$

ТУ. У вхідному потоці задається ціле n ($0<=n<10^4$). У вихідний потік вивести значення функції.

Вхідні дані
3

Вихідні дані
6

Задача 8-6
Знайти всі дільники числа N, які є числами Фібоначчі.
ТУ. У вхідному потоці дано N ($N<10^6$). У вихідний потік через пропуск вивести дільники у порядку зростання.

Вхідні дані
10

Вихідні дані
1 2 5

Задача 8-7
Знайти значення «91-функції Мак-Карті», як задається таким чином:

$$F(n)= \begin{cases} n-10, & n>100 \\ F(F(n+11)), & n<=100 \end{cases}$$

ТУ. У вхідному потоці задається ціле додатне N<200. У вихідний потік вивести значення функції.

Вхідні дані
199

Вихідні дані
189

Задача 8-8
Знайти значення функції, що задається таким чином:

$$S(n) = \begin{cases} n, & n<10 \\ S(n \text{ div } 10) + n \bmod 10, & n>=10 \end{cases}$$

ТУ. У вхідному потоці задається ціле додатне N ($N<=10^9$). У вихідний потік вивести значення функції.

Вхідні дані
11

Вихідні дані
2

Задача 8-9
Для заданого натурального числа, вивести його цифри у зворотному порядку.
ТУ. У вхідному потоці задається натуральне N, що містить не більше 200 цифр. У вихідний потік вивести число з оберненим порядком цифр.

Вхідні дані
1230

Вихідні дані
0321

Задача 8-10
Знайти значення функції для n , якщо

$$T(n) = \begin{cases} 1, & n=1 \\ 2*T(n \text{ div } 2), & n>1 \end{cases}$$

ТУ. В одному рядку вхідного потоку задаються n (n<=100000). Вхідні дані підібрані таким чином, що результат можна отримати. У вихідний потік вивести значення коефіцієнта.

Вхідні дані
10

Вихідні дані
8

Задача 8-11
Знайти біноміальний коефіцієнт для даних m,n (0<=n<=m), якщо

$$C(m,n) = \begin{cases} 1, & n=m,\ n=0,\ m<=1 \\ C(m-1,n-1) + C(m-1,n) \end{cases}$$

ТУ. В одному рядку вхідного потоку задаються m, n (m,n<30). У вихідний потік вивести значення коефіцієнта.

Вхідні дані
10 2

Вихідні дані
45

Задача 8-12

Розбиттям натурального числа називається його представлення у вигляді суми натуральних чисел. Суми з різним порядком доданків вважаються однаковими. Знайти кількість розбиттів числа N.

ТУ. У вхідному потоці задано число N (N<=80). У вихідний потік вивести кількість розбиттів.

Вхідні дані

4

Вихідні дані

5

Для 4-х можна отримати такі розбиття: 4, 3+1, 2+2, 2+1+1, 1+1+1+1.

Задача 8-13

Реалізувати «Індійський алгоритм» піднесення до степеня який працює за таким правилом:

$$x^n = \begin{cases} x, & n=1 \\ x * (x^{n \text{ div } 2})^2, & n>1 \text{ і } n - \text{непарне} \\ (x^{n \text{ div } 2})^2, & n>1 \text{ і } n - \text{парне} \end{cases}$$

ТУ. У стандартному вхідному потоці дано цілі x та n (0<x<10, 0<=n<50). У вихідний потік вивести значення функції. Гарантується, що результат не буде перевищувати 10^{18}.

Вхідні дані

2 4

Вихідні дані

16

Задача 8-14

Дано N впорядкованих за зростанням натуральних чисел. Визначити чи є серед даних чисел число K.

ТУ. У першому рядку вхідного потоку дано числа N,K ($N<10^6$, $K<10^9$). Наступний рядок містить N чисел. У вихідний потік вивести Yes або No в залежності від наявності числа K.

Вхідні дані

5 101

2 5 10 101 103

Вихідні дані

Yes

Задача 8-15
Знайти значення функції Аккермана, що задається таким чином:

$$A(m, n) = \begin{cases} n+1, & m=0 \\ A(m-1,1), & m>0, n=0 \\ A(m-1, A(m, n-1)), & m>0, n>0 \end{cases}$$

ТУ. У вхідному потоці в одному рядку дано натуральні m, n ($m<5, n<1000$). Вхідні дані підібрані таким чином, що результат можна отримати. У вихідний потік вивести значення функції.

Вхідні дані
3 1

Вихідні дані
13

Розділ 9. Множини

Поняття множин у мові Паскаль дещо вужче, ніж у математиці. Зазвичай це набір певної кількості зв'язаних однотипних елементів. В Паскалі кількість елементів множини може бути від 0 (порожня множина) до 256. Тип кожного елемента, що називається базовим типом, повинний бути простим, крім REAL. Тип множини описується у розділі TYPE:

TYPE ім'я_типу = SET OF тип_елементів;

"Тип_елементів" може бути або перераховним, або інтервальним ти пом. Значення елементів (або їх кодів) повинні входити у множину 0..255.

Приклад:

```
TYPE     numbers = set of 1.. 3;
          charset = set of 'A'..'Z';
        primcolor = (red, green, blue);
          VAR n1,n2 : numbers;
                chars : charset;
      color 1, color2, color3 : SET OF primcolor;
```

Тут numbers і charset — імена типів множин, primcolor — перераховний тип. Змінні описуються як з використанням вже визначених типів множин, так і безпосередньо (як color 1, color2, color3). Всі змінні, що описані вище, є множинами. Множина складається з елементів базового типу і може бути будь-якою підмножиною початкової множини (тобто повної множини об'явленого типу), включаючи й порожню множину. Значенням змінної типу SET є конкретна множина. Так значеннями змінних n1 та n2 може бути одна з наступних множин:

[], [1], [2], [3], [1,2],[1,3],[2,3],[1,2,3]

Зверніть увагу, що множина позначається квадратними дужками, всередині яких знаходиться список елементів множини. [] — порожня множина.

Значенням змінної chars може бути будь-яка з множин:

[],[A],[B],…,['Z'],['A','B'], … ['A','Z'], … ,['A','Z'].

Змінні color1, color2, color3 можуть мати значення однієї з множин: [], [red], [green], [blue], [red,green], [red,blue], [green,blue], [red, green, blue].

Для змінних, що описані вище, можливі, наприклад, такі оператори присвоювання:

```
n:= [2,3];

n2:=[];

chars := ['A'..'H','R']

color1 := [red,blue];
```

Множина, що описується за допомогою інтервалу, в якому перше граничне значення перевищує друге, трактується як порожня. Наприк лад, множини [3..1], ['C'..'A'], [blue..red] є порожніми.

При роботі з множинами можна користуватися операціями об'єд нання (+), перетину (*), різниці (-).

Наприклад:

[1,2]+[2,3] дає [1,2,3];

[1,2]*[2,3] дає [2];

[1..3] – [1] дає [2,3];

[‘A’..’H’,’P’] – [‘D’..’H’] дає [‘A’,’B’,’C’,’P’].

За допомогою таких операцій можна будувати вирази, що належать цьому певному типу. В операторах присвоювання для змінних типу множин права частина у загальному випадку є вираз того самого типу. Множини можна порівнювати за допомогою операцій відношення =, <>,<=,>=.

Логічний вираз S1=S2, де S1 та S2 — однотипні множини, має значення TRUE, якщо S1 та S2 містять одні й ті самі елементи. Якщо ж множини відрізняються хоча б одним елементом, значення виразу S1<>S2 буде TRUE.

Значення TRUE мають також вирази:
S1<=S2, якщо множина S1 є підмножиною множини S2;
S1>=S2, якщо множина S2 є підмножиною множини S1.

Наприклад:

[1..3] = [1,2,3] є TRUE;

[‘A’..’Z’] – [‘C’,’K’] <= [‘A’..’Z’] є TRUE;

[] <= [red,blue] є TRUE;

[2] <= [3] є FALSE;

[1,2]*[] <> [] є FALSE;

[3..1] >= [2] є FALSE;

Спеціальна операція IN дозволяє визначити, чи міститься якийсь елемент у цій певній множині. Результатом цієї операції є значення типу BOOLEAN. Приклади:

1)A := 5; A in [1..10] є
2)Ch := ‘Y’; Ch in [‘a’..’z’] є FALSE;
3)green in [red..blue] є TRUE;

Зверніть увагу, що лівим операндом операції IN повинно бути значення, що належить базовому типу елементів множини, що є правим операндом.
За допомогою операції IN більшість операцій відношення записується набагато простіше та наочніше. Наприклад, замість оператора

IF (x=2) or (x=4) or (x=5) or (x=6) or (x=7) THEN

можна записати

IF x in [2,4..7] THEN …

Як кажуть, коментарі тут зайві.
Операції з множинами у програмах зустрічаються доволі рідко. До того ж швидкість виконання цих операцій залишає бажати кращого. В основному використання множин зводиться до використання операції IN.

І останнє зауваження. Значення змінних типу множин (SET) ЗАБОРОНЕНО вводити чи виводити операторами read та write. Кори стуються наведеними вище операціями в основному лише для службових цілей і проміжних обчислень всередині програми. Виводити елементи множини на екран можна таким чином:

```
PROGRAM Set_Example;

 TYPE charset = set of 'A'..'Z';          {множина великих латинських літер}
 VAR   chars1,chars2,chars3: charset;
       c : char;                          {допоміжна змінна}
BEGIN
    chars := ['A'..'H','P'];
    chars2 := ['D'..'H'];
    chars3 := chars1-chars2;
    for c := 'A' to 'Z' do                {цикл по всіх елементах}
    if c in chars3                        {базової множини       }
      then write(c:2)
 END.
```

Такий приклад з різницею множин було наведено вище. Тому зали шається лише перевірити роботу цієї програми і порівняти результати.

Зараз розглянемо приклади задач з використанням теорії множин..

Задача 1.
Дана множина із N цілих додатних чисел не більших 200. Вивести ті з них, що кратні 2 і 3 одночасно. У першому рядку знаходиться N, а в наступному іде перелік самих чисел множини.

```
type numbers=set of 0..200;
var m          : numbers;
   i,n,k       : integer;
begin
 m:=[]; { m присвоїли порожню множину}
 readln(n);
 for i:=1 to n do begin {читаємо числа та добавляємо їх у множину }
      read(k);
      m:=m+[k];
 end;
 readln;
 for i:=0 to 200 do
   if (i in m) and (i mod 6=0) then write(i,' ');
 writeln;
end.
```

Задача 2
Дано текст довжиною не більше 255 символів. Вивести всі малі латинські літери, що зустрічаються в даному тексті зберігаючи порядок їх слідування.

```
var i          : integer;
   s           : string;
begin
 readln(s);
 for i:=1 to length(s) do
   if s[i] in ['a'..'z'] then write(s[i]);
 writeln;
end.
```

Задача 3.
Дано текст довжиною не більше 255 символів. Знайти кількість цифр у ньому.

```
var i,k        : integer;
   s          : string;
begin
  readln(s);
  k:=0;
  for i:=1 to length(s) do
    if s[i] in ['0'..'9'] then k:=k+1;
  writeln(k);
end.
```

Ці задачі повинні бути зрозумілими і без коментарів. У першій задачі деяка множина формувалася з чисел, що вводилися з клавіатури, а в другій та третій задачі використовувалися множини елементів певних інтервалів.

Завдання для самоконтролю

Задача 9-1
Задано множини A та B, що складаються з двоцифрових чисел. Знайти суму тих елементів, що входять як у множину A, так і в множину B.
ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. У вихідний потік вивести суму чисел.
Ввід<
3
10 12 13
4
11 12 13 14
Вивід>
25

Задача 9-2
Задано множини A та B, що складаються з двоцифрових чисел. Знайти суму тих елементів, що входять хоча би в одну з цих множин.
ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. У вихідний потік вивести суму чисел.
Ввід<
3
10 12 13
4
11 12 13 14
Вивід>
60

Задача 9-3
Задано множини A та B, що складаються з двоцифрових чисел. Знайти різницю (A-B).
ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. У єдиний рядок вихідного потоку вивести числа, що входять до (A-B).
Ввід<
4
11 12 13 14
3
10 12 13
Вивід>
11 14

Задача 9-4
Задано множини A, B та C, які складаються з цілих чисел не більших 255. Знайти числа, що входять хоча би в одну із множин A або B, але не входять у множину C.
ТУ. У першому рядку знаходиться число N – кількість чисел множини A. У наступному рядку міститься N чисел. Дальше у новому рядку задається число M – кількість чисел множини B і у четвертому рядку містяться самі числа цієї множини. В наступному рядку міститься K – кількість чисел множини C і потім у новому рядку самі елементи множини C. У єдиний рядок вихідного потоку вивести числа, що входять мають властивості описані в умові задачі. Числа виводити у порядку зростання.
Ввід<
4
1 2 6 4
3
2 3 4
3
1 4 5
Вивід>
2 3 6

Задача 9-5
Дано два слова, що складаються з малих латинських літер. Вивести ті літери, що не входять в жодне із слів.
ТУ. В першому рядку знаходиться перше слово, у другому – друге. У вихідний потік вивести малі літери латинського алфавіту.
Ввід<
qwertyuiop
asdfghjkl
Вивід>
zxcvbnm

Задача 9-6
Заданий текст, що складається з малих латинських літер. Вивести ті літери, що зустрічаються в тексті не менше двох разів.
ТУ. У вхідному потоці знаходиться текст довжиною не більше 255 символів. Вивести в алфавітному порядку через пропуск літери, що зустрічаються більше двох разів.

Ввід<
the essence of
Вивід>
e s

Задача 9-7
Заданий текст, що містить малі літери латинського алфавіту і цифри. Вивести голосні, приголосні та цифри, що зустрічаються у даному тексті.
ТУ. У вхідному потоці знаходиться текст довжиною не більше 255 символів. У вихідний потік вивести у першому рядку голосні, у другому – приголосні, у третьому – цифри. Вивід символів здійснювати у порядку зростання їх кодів.

Ввід<
the essence of 2007
Вивід>
eo
cfhnst
027

Задача 9-8
Заданий текст із латинських літер, в кінці – крапка. Вивести на друк усі літери, які входять до тексту один раз. Великі та малі літери не розрізняти, при виведенні використовувати лише малі літери.
ТУ. У вхідному потоці дано текст, що закінчується крапкою. Вивести в алфавітному порядку літери, що зустрілися в тексті лише один раз.

Ввід<
Olimpiads.
Вивід>
adlmops

Задача 9-9
Задана послідовність слів, розділених пропусками. Знайти кількість голосних в найдовшому слові. Великі та малі літери не розрізняти, при виведенні використовувати лише малі літери. Якщо таких слів є декілька, то слід брати перше у тексті.
ТУ. У вхідному потоці знаходиться текст із латинських літер довжиною не більше 255 символів. Вивести у вихідний потік число – кількість голосних у найдовшому слові.
Ввід<
the essence of
Вивід>
3

Задача 9-10
Задані n натуральних чисел. Для кожного введеного числа надрукувати в порядку зростання усі цифри, що не входять в десятковий запис цього числа.
ТУ. У першому рядку міститься число N (N<=1000). У наступних N рядках знаходяться десяткові записи чисел. Кількість цифр чисел не перевищує 100. Вивести у N рядках цифри, що відповідають умові задачі. Якщо таких цифр немає, то виводити порожній рядок.
Ввід<
1
16
Вивід>
02345789

Задача 9-11
Є група учнів, яких ми умовно будемо іменувати символами A..Z.
Також відомо хто входить в групи: Male, Female, Aerob, Karate. Знайти хто із учнів займається аеробікою та карате?
ТУ. В чотирьох різних рядках задано списки учнів, що належать даним групам у такому порядку: Male, Female, Aerob, Karate. У вихідний потік вивести по одному у рядку імена учнів, що займаються аеробікою та карате, причому спочатку вивести хлопців в алфавітному порядку, а потім дівчат.
Ввід<
A B R
C D E
A D E
B D A
Вивід>
A
D

Задача 9-12
Заданий рядок символів. Вивести символи, що не є буквами або цифрами.
ТУ. Задано текст довжиною не більше 30000 символів, що закінчується символом '#'. У тексті можуть зустрічатися лише букви латинського алфавіту та символи пунктуації, розділові знаки. У вихідний потік вивести у порядку зростання їх кодів символи, що відповідають умові задачі.
Ввід<
The day of an English pupil.#
Вивід>
.

Задача 9-13
Заданий текст із латинських літер, кінець якого позначений крапкою. Вивести перші входження букв до тексту, зберігаючи їх взаємний порядок.
ТУ. Довжина вхідного тексту не більша 1000 символів. У вихідний потік вивести літери в одному рядку. Розрізняти великі та малі літери.

Ввід<
The day of an English pupil.
Вивід>
ThedayofnEglispu

Задача 9-14
Задана послідовність слів. Вважаючи перше слово зразком, вибрати ті слова, які можуть бути отримані із зразка шляхом переставлення його літер.
ТУ. У вхідному потоці задано число N (N<=1000). Дальше у N рядках знаходиться по одному слову довжиною не більше 100 символів, слова складаються з малих латинських літер. У вихідний потік вивести шукані слова по одному у рядку.

Ввід<
4
abba
baab
bara
aabb
Вивід>
baab
aabb

Задача 9-15
Задано текст. Вивести літери латинського алфавіту, що зустрічаються в тексті менше, ніж К разів.
ТУ. У першому рядку вхідного потоку задається число К (К<100). У наступному рядку знаходиться текст довжиною не більше 30000 символів, що закінчується символом ‘#’. У вихідний потік вивести символи, що задовільняють умову задачі. Великі та мали літери не розрізняти, виводити лише малі літери в алфавітному порядку.

Ввід<
1
The day of an English pupil. #

Вивід>
bcjkmqrvwxz

Розділ 10. Робота з файлами даних

Дані, що можуть зберігатися програмою у пам’яті комп’ютера мають достатньо обмежений час життя; коли живлення виключити, то дані будуть втрачені. Для цього інформацію зберігають на різного типу носіях у файлах.

Розрізняють два види файлів: послідовного та довільного доступу. Послідовні файли складаються з елементів різної довжини, між якими знаходяться розділювачі. Щоб знайти елемент у послідовному файлі, треба переглянути всі попередні.

Файли довільного доступу складаються з однотипних елементів і цим подібні до масивів. Будь-який елемент можна знайти по його порядковому номеру.

В Паскалі послідовні файли називають текстовими, а файли довільного доступу – типізованими. Ми ж будемо мати справу у більшості випадків з текстовими файлами.

Робота з файлами складається з трьох етапів:

1. відкриття файлу;
2. читання з файлу та запис;
3. закриття файлу.

Текстові файли зберігають інформацію у вигляді послідовності символів. Символи утворюють рядки довільної. В кінці кожного рядка знаходяться два спеціальних символи: #13 #10, які розділяють рядок від наступного. Текстовий файл можна отримати за допомогою будь-якого екранного редактора.

Текстовий файл у Паскалі має представляти файлова змінна типу **text**.

var f : text;

Файлову змінну пов’язують з конкретним файлом за допомогою оператора

assign(f,’input.txt’);

Дальше у програмі ми завжди при зверненні до файлу будемо використовувати лише файлову змінну. Для роботи з файлом його треба відкрити і тут є такі три випадки:
для читання – оператором **reset(f);**
для запису – оператором **rewrite(f);**
для поповнення – оператором **append(f).**

У файл відкрити для поповнення або запису можна лише записувати, а з файлу відкритого для читання можна лише читати. Після закінчення роботи з файлом його необхідно закрити за допомогою оператора

close(f);

Після закриття файлу його знову можна відкривати з іншим статусом.

Читання з файлу відбувається за допомогою оператора read(). Якщо перед чписком вводу в цьому операторі стоїть файлова змінна, то дані будуть введені із файлу.
read(a,b,c); читаємо з клавіатури
read(f,a,b,c); читаємо з файлу

Щоб зрозуміти як відбувається читання з файлу, введемо поняття вказівника файлу. Фактично вказівник – це номер чергового символу файлу. Зразу ж після відкривання файлу, вказівник вказує на перший символ. Читання чергової порції завжди буде починатися з символу, на який вказує вказівник. Після читання першої порції, вказівник автоматично переміститься на початок наступної. Причому, якщо ми читаємо число, то порція буде поширюватися до наступного пропуску, якщо символ, то до наступного символу; якщо рядок, то до наступного рядка. Так буде продовжуватися поки вказівник не дійде до кінця файлу. Якщо продовжити читання, то програма видасть повідомлення про помилку.
Як же взнати коли ще можна читати з файлу? Для цього користуються функцією

EOF(f).

Ця функція повертає true, якщо вказівник досяг кінця файлу, і false в іншому випадку. Назва функції є скороченням слів “End Of File”.

Додатковий оператор readln() працює аналогічно, тільки після кожної порції зчитаного вказівник переходить на наступний рядок.

Задача 1. Прочитати з текстового файлу та вивести на екран. (Це не зовсім задача, швидше – приклад)

```
var     f : text;
        s : string;
begin
        assign(f,’input.txt’);
        reset(f);
        while not EOF(f) do begin
                readln(f,s);
                writeln(s);
        end;
        close(f);
end.
```

Запис інформації у файл здійснюється оператором write(), в якому перед списком виводу стоїть файлова змінна. Виведена інформація буде послідовно добавлена до тієї, що уже була виведена після відкриття файлу для запису. Вивід у файл не залежить від того як був відкритий для запису файл: за допомогою оператора rewrite(), чи оператором append(). Після закінчення запису у файл його обов’язково треба закрити. Тільки після закриття файлу відбудеться його формування.

Додатковий оператор writeln() виводить у файл аналогічно, тільки кожна порція закінчується символами #13 #10 в кінці.

Задача 2. Вивести текст введений з клавіатури у файл ‘dani.txt’.

```
var     f : text;
        s : string;
begin
        assign(f,’dani.txt’);
        rewrite(f);
        readln(s);
        while s<>’’ do begin  {виконуємо поки рядок не порожній}
                writeln(f,s);
                readln(s);
        end;
        close(f);
end.
```

Крім дисплея та запам'ятовуючих пристроїв інформацію можна виводити на принтер, в комунікаційний порт, на порожній пристрій. Операційна система виконує роль посередника при обміні інформацією між програмою та будь-яким зовнішнім пристроєм. Це виражається в тому, що програмісту пропонується працювати не з самими складними та різними пристроями, а х логічною моделлю пристою у формі текстового файлу.

Текстові файли, що моделюють пристрої, мають такі стандартні імена:

PRN - принтер
CON - консоль (дисплей+клавіатура)
COM1 – перший комунікаційний порт
NUL - порожній пристрій – ігнорує запис і генерую сигнал «кінець файлу» при читанні.

Робота з цими пристроями нічим не відрізняється від роботи з текстовими файлами.

Для визначення наявності файлу на диску є корисною функція

IOresult : integer

яка повертає ціле число, що відповідає кодові останньої помилки висновку. При нормальному завершенні операції функція поверне значення 0. Функція IOresult працює тільки з ключем компіляції {$I-}.

Задача 3. Виведення даних у файл з попередньою перевіркою наявності файлу для читання на диску.

```
var     f_in   : text;
        f_out : file;
        i, k     : integer;
        s_in, s_out : string;
begin
{$I-}
  writeln('Введіть ім'я вхідного файлу'); readln(s_in);
  assign(f_in, s_in);
  reset(f_in);
  if IOresult <>0 then begin
     writeln('Файл ', s_in,' не знайдено'); exit end;
  writeln('Введіть ім'я вихідного файлу'); readln(s_out);
  assign(f_out, s_out);
  rewrite(f_out);
{$I+}
  writeln(f_out,'Текст, записаний у файл');
  close(f_in); close(f_out);
end.
```

Задача 4
Знайти суму та кількість парних чисел.
ТУ. У першому рядку вхідного файлу numbers.dat знаходиться натуральне N (N<100000). У наступному рядку через пропуск міститься N цілих чисел не більший 30000. У вихідний файл numbers.sol у перший рядок вивести кількість парних чисел, у другий їх суму.

```
var     f : text;
        s,i,k,m : longint;
begin
        assign(f,'numbers.dat');
        reset(f);
        s:=0;
        k:=0;
        readln(f,n);
        for i:=1 to n do begin
                read(f,m);
                if m mod 2=0 then begin s:=s+m;k:=k+1;end;
        end;
        close(f);
        assign(f,'numbers.sol');
        rewrite(f);
        writeln(f,k);
        writeln(f,s);
        close(f);
end.
```

Задача не є складною і, якщо ти уже читаєш даний урок, то додаткових пояснень, думаю, не треба.

Тепер переходимо до практики. Задач, які мали би повністю файлове спрямування не дуже багато, а олімпіадних задач, що вимагають спеціальної роботи з файлами я взагалі ще не зустрічав. Отже, дальше розглядаються різноманітні задачі з використанням різних прийомів роботи з файлами.

Завдання для самоконтролю.

Примітка. У зв'язку з тим, що система автоматизованої перевірки використовує для зчитування вхідних даних файл input.txt або консоль, то у всіх задачах дані треба читати з файлу input.txt. Вихідні дані у всіх завданнях треба відправляти лише на консоль.

Задача 10-1
Знайти різницю між найбільшим та найменшим числом.
ТУ. Числа задаються у вхідному файлі по одному числу у рядку. Кількість чисел не більше 100000, числа не перевищують по модулю10000.
Приклад вхідних та вихідних даних
Вхідні дані
3
6
1
11
Вихідні дані
10

Задача 10-2
Знайти найдовше слово.
ТУ. У вхідному файлі задані слова по одному у кожному рядку. Вивести найдовше слово що першим зустрілося у переліку. Довжина слів не перевищує 255 символів, кількість слів не більше 10000.

Приклад вхідних та вихідних даних.
Вхідні дані
abba
words
files
Вихідні дані
words

Задача 10-3
Знайти у тексті кількість літер ‘a’ (латинський алфавіт).
ТУ. У вхідному файлі задається вхідний текст довжиною до 100000 символів. У вихідний потік вивести кількість повторів ‘a’.

Приклад вхідних та вихідних даних.
Вхідні дані
abba
Вихідні дані
2

Задача 10-4
Серед даних чисел вибрати лише парні.
ТУ. У вхідному файлі задано через пропуск числа не більші 30000. Кількість чисел не перевищує 50000. У вихідний потік вивести парні числа по одному у рядку.

Приклад вхідних та вихідних даних.
Вхідні дані
12 1001 12 53

Вихідні дані
12
12

Задача 10-5
Задано N натуральних чисел. Знайти число з найбільшою сумою цифр.
ТУ. У першому рядку вхідного файлу дано N (N<100000). У наступному рядку через пропуск задаються самі числа. У вихідний потік вивести перше з чисел з найбільшою сумою цифр.

Приклад вхідних та вихідних даних.
Вхідні дані
4
31 5 11 25

Вихідні дані
25

Задача 10-6
Задано N натуральних чисел. Знайти число з найбільшою сумою дільників.
ТУ. У першому рядку вхідного файлу дано N (N<1000). У наступному рядку через пропуск задаються самі числа. У вихідний потік вивести перше з чисел з найбільшою сумою дільників.

Приклад вхідних та вихідних даних.
Вхідні дані
4
31 5 11 25

Вихідні дані
31

Задача 10-7
Два прямокутники, сторони яких паралельні осям координат, задані координатами протилежних вершин. Знайти площу їх перетину.
ТУ. У вхідному файлі у двох різних рядках задані координати протилежних вершин прямокутників. Координати є цілими числами і по модулю не перевищують 10000. У вихідний потік вивести площу їх спільної частини. Якщо перетин є точка або пряма - вивести 0, якщо прямокутники не перетинаються - вивести -1.

Приклад вхідних та вихідних даних.
Вхідні дані
0 0 10 10
0 10 100 100

Вихідні дані
0

Задача 10-8
На площині задано N точок і координати двох протилежних вершин деякого прямокутника з сторонами паралельними осям координат. Вивести координати точок, що належать прямокутнику. Точки, що лежать на сторонах вважати такими, що належать прямокутнику.
ТУ. У першому рядку вхідного файлу знаходяться координати вершин прямокутника, у другому – число N (N<=100000) – кількість точок. У наступних N рядках міститься по два цілих числа в межах [-10000,10000] – координати точок. У вихідний потік вивести координати шуканих точок у порядку їх слідування у вхідному файлі.

Приклад вхідних та вихідних даних.
Вхідні дані
0 0 10 10
4
1 1
2 2
10 10
11 12

Вихідні дані
1 1
2 2
10 10

Задача 10-9

У проміжку [m,n] знайти просте число з найбільшою сумою цифр. Якщо таких чисел є декілька, то вивести те, що є найбільшим.

ТУ. У одному рядку вхідного файлу дано цілі числа m,n (0<m,n<30000). У вихідний потік вивести шукане число.

Приклад вхідних та вихідних даних.

Вхідні дані

1 10

Вихідні дані

7

Задача 10-10

Для заданого x знайти значення виразу $1 + 8 + 27 + \ldots + x^3$

ТУ. Кожен рядок вхідного файлу містить число x (1<=x<=50000). У вихідний файл для кожного значення x в окремому рядку вивести значення виразу.

Приклад вхідних та вихідних даних.

Вхідні дані

1
2
3

Вихідні дані

1
9
36

Розділ 11. Структури даних

Записи.

У житті дуже часто зустрічається інформація, яка складається із даних різного типу. Це анкети, карточки бібліотечного каталогу, відомості про зарплату працівника організації та інші. Для того, щоб їх зручніше представити у програмі використовують записи.

Запис – це складений тип даних, який об'єднує у собі різнотипні елементи (поля запису). Такий тип можна описати за допомогою такої конструкції:

```
Ім'я типу = record
        ім'я поля : тип поля;
        ім'я поля : тип поля;
        ……..
        ім'я поля : тип поля;
end
```

Розглянемо приклад запису, що містить відомості про книжку.

```
type    book = record
                tittle: string[80];     {назва}
                author: string[20];     {автор}
                year: integer;          {рік видання}
        end;
```

Над записами допустимі операції присвоювання, перевірки на рівність чи нерівність, ввід та вивід. З полем запису в програмі можна виконувати всі ті дії, що і з змінною аналогічного типу. Щоб звернутися до поля запису використовують складене ім'я, розділене «.». Наприклад, для описаного типу можна написати:

```
var     a,b     : book;
        c       : array [1..100] of  book;
begin
        a.tittle:='Три мушкетери';
        a.author:='О.Дюма';
        a.year:=1991;
        writeln(a.tittle);
        b:=a;
        writeln(b.tittle);
        c[10]:=b;
        writeln(c[10].tittle);
end.
```

Думаю, неважко передбачити результат роботи такої програми.

Тип поля може бути будь-яким, в тому числі і записом. Наприклад:

```
type FName = record
        SurName : string;
        Name : string[20];
     end;
     book = record
        tittle: string[80];
        author: FName;
        year: integer;
      end;
```

Якщо поле є записом, то звернення до його елементів(полів поля) відбувається по імені, що складається з трьох частин. Наприклад, a.author.SurName. Глибина таких вкладень нічим не обмежена.

Оператор With

Напевне, цей оператор для того, хто не любить багато писати. Він дозволяє суттєво скоротити звернення до полів запису. Записується він таким чином:

With ім'я запису do оператор

Всередині цього оператора можне опускати ім'я запису в складеному імені поля, транслятор добавить його автоматично. Наприклад:

```
type    book = record
                tittle: string[80];
                author: string[20];
                year: integer;
        end;
var     a       : book;
begin
        with a do readln(tittle, author, year);
        with a do writeln(tittle,' ',author,' ',year);
end.
```

Задача 1.
Ввести масив записів, кожен з яких містить відомості про прізвище, адресу та номер телефону. Скласти програму, що сортує ці дані за алфавітом прізвищ, знаходить в масиві записів запис, що введений з клавіатури)

```
type    anketa = record
            FIO : string;
            adress : string;
            phone : string;
        end;
var     mas : array [1..10] of anketa;
        i,j,k : integer;
        rec : anketa;
procedure vvid;
begin
 i:=0;
 repeat
        i:=i+1;
        with mas[i] do  begin
                write('FIO= ');readln(FIO);
                write('adress= ');readln(adress);
                write('tel= ');readln(phone);
        end;
 until i=10;
end;
procedure vyvid;
begin
        writeln('-FlO--adress----telephone--');
        for j:=1 to i do  begin
                with mas[j] do  begin
```

```
                write('  ',FIO );
                write('  ',adress);
                write('   ',phone);
            end;
            writeln;
        end;
end;
procedure sort;
var tmp : anketa;
begin
        for j:=1 to i-1 do
            for k:=j+1 to i do  begin
                if mas[j].fio>mas[k].fio then
                    begin
                        tmp:=mas[j];
                        mas[j]:=mas[k];
                        mas[k]:=tmp;
                    end;
            end;
end;
procedure search;
var flag:boolean;
begin
        writeln('input record for search: fio, adress, tel');
        readln(rec.fio);
        readln(rec.adress);
        readln(rec.phone);
        for j:=1 to i do
             with mas[j] do  begin
                if (fio=rec.fio)and(adress=rec.adress)and(phone=rec.phone)
                    then  begin
                        flag:=true;
                        writeln(' record found:',rec.fio,' ',rec.adress,' ',rec.phone);
                        break;
                    end else flag:=false;
            end;
        if flag=false then writeln('record not found');
end;
begin
   vvid;
   writeln('array of inputs records');
   vyvod;
   sort;
   writeln('array of records after sort');
   vyvod;
   search;
end.
```

Завдання для самоконтролю

У всіх задачах вхідні дані треба читати з файлу **input.txt**, а виводити у стандартний вихідний потік.

Задача 11-1
Дано назви геометричних фігур та їх площу. Відсортувати вхідні дані у порядку зростання спочатку назви фігури, а потім її площі.
ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість різних даних фігур. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік вивести впорядковані дані у такому форматі: в одному рядку назва фігури і через пропуск її площа.

Вхідні дані
3
kolo
2.52
figura
10.12
kolo
3.25

Вихідні дані
figura 10.12
kolo 2.52
kolo 3.25

Задача 11-2
Дано назви геометричних фігур та їх площу. Знайти фігуру із площею, що є найближчою до площі першої у списку фігури. Якщо таких є декілька, то вивести ту, що йде у переліку раніше.
ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік вивести в одному рядку назву фігури і через пропуск її площу.

Вхідні дані
3
kolo
2.52
figura
10.12
kolo
3.25

Вихідні дані
kolo 3.25

Задача 11-3
Дано назви геометричних фігур та їх площу. Знайти фігури із найбільшими та найменшими площами.
ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік виводити спочатку дані про фігури з найбільшими площами, а потім з найменшими. Вивід здійснювати в окремому рядку для кожної фігури. Дані про жодну фігуру не можуть бути виведені більше одного разу.

Вхідні дані
3
kolo
2.52
figura
2.52
kolo
3.25

Вихідні дані
kolo 3.25
kolo 2.52
figura 2.52

Задача 11-4
Дано назви геометричних фігур та їх площу. Знайти сумарну площу для кожної назви фігури.
ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік в окремих рядках виводити назву фігур та через пропуск їх загальну площу.

Вхідні дані
3
kolo
2.52
figura
2.52
kolo
3.25

Вихідні дані
kolo 5.77
figura 2.52

Задача 11-5
Дано назви геометричних фігур та їх площу. Які з фігур мають найбільшу загальну площу?
ТУ. У першому рядку вхідного файлу міститься ціле число N(N<1000) – кількість рядків даних. Дальше у наступних 2*N рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 20 символів і складається з малих латинських літер, а площа є дійсним числом з двома знаками після коми. У вихідний потік вивести назву фігур та через пропуск їх загальну площу. Якщо таких назв фігур є декілька, то виводити лише ту, що йде у переліку раніше.

Вхідні дані
3
kolo
2.52
figura
2.52
kolo
3.25

Вихідні дані
kolo 5.77

Задача 11-6
Дано список прізвищ N учнів з їх оцінками по K предметах. Знайти середні бали для кожного учня.
ТУ. У першому рядку вхідного файлу дано цілі числа N, K(0<N<=1000, 0<K<10). Дальше у 2*N рядках дані розміщені таким чином: спочатку в окремому рядку іде прізвище, а у наступному через пропуск знаходяться K оцінок. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках прізвища учнів і їх середні бали.

Вхідні дані
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5

Вихідні дані
Ivanov 4.5
Petrov 2.5
Sidorov 5.0

Задача 11-7
Дано список прізвищ N учнів з їх оцінками по K предметах. Вивести список учнів, що мають максимальний середній бал.
ТУ. У першому рядку вхідного файлу дано цілі числа N, K(0<N<=1000, 0<K<10). Дальше у 2*N рядках дані розміщені таким чином: спочатку в окремому рядку іде прізвище, а у наступному через пропуск знаходяться K оцінок. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках прізвища учнів і їх середні бали.

Вхідні дані
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5

Вихідні дані
Sidorov 5.0

Задача 11-8
Дано список M класів з прізвищами Ni учнів, що там навчаються, а також з їх оцінками по Ki предметах. Вивести список класів з найвищим середнім балом по всіх предметах для даного класу.
ТУ. У першому рядку вхідного файлу дано кількість класів M (0<M<100). У наступному рядку знаходиться назва класу з довжиною не більше 4-х символів. Після цього у новому рядку містяться цілі числа Ni, Ki (0<Ni<=40, 0<Ki<10). Дальше дані розміщені таким чином: спочатку в окремому рядку знаходиться прізвище, а у наступному через пропуск знаходяться K оцінок. Дальше все повторюється для наступного класу. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках назви класів з найвищими середніми балами у порядку їх слідування у списку.

Вхідні дані
2
10-A
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5
10-B
2 3
Kukuschkin
4 4 4
Karasik
5 3 4

Вихідні дані
10-A 4.0
10-B 4.0

Задача 11-9

Дано список M класів з прізвищами Ni учнів, що там навчаються, а також з їх оцінками по Ki предметах. Визначити учнів з найвищим середнім балом.

ТУ. У першому рядку вхідного файлу дано кількість класів M (0<M<100). У наступному рядку знаходиться назва класу з довжиною не більше 4-х символів. Після цього у новому рядку містяться цілі числа Ni, Ki (0<Ni<=40, 0<Ki<10). Дальше дані розміщені таким чином: спочатку в окремому рядку знаходиться прізвище, а у наступному через пропуск знаходяться K оцінок. Дальше все повторюється для наступного класу. Прізвища довжиною не більше 10 символів. У вихідний потік вивести в окремих рядках вивести прізвище учня, його клас та середній бал.

Вхідні дані
2
10-A
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5
10-B
2 3
Kukuschkin
4 4 4
Karasik
5 3 4

Вихідні дані
Sidorov 10-A 5.0

Задача 11-10
Дано список M класів з прізвищами Ni учнів, що там навчаються, а також з їх оцінками по Ki предметах. Визначити кількість учнів з середнім балом більшим 4.
ТУ. У першому рядку вхідного файлу дано кількість класів M (0<M<100). У наступному рядку знаходиться назва класу з довжиною не більше 4-х символів. Після цього у новому рядку містяться цілі числа Ni, Ki (0<Ni<=40, 0<Ki<10). Дальше дані розміщені таким чином: спочатку в окремому рядку знаходиться прізвище, а у наступному через пропуск знаходяться K оцінок. Дальше все повторюється для наступного класу. Прізвища довжиною не більше 10 символів. У вихідний потік вивести кількість учнів.

Вхідні дані
2
10-A
3 4
Ivanov
4 4 5 5
Petrov
2 3 3 2
Sidorov
5 5 5 5
10-B
2 3
Kukuschkin
4 4 4
Karasik
5 3 4

Вихідні дані
2

Задача 11-11
Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Визначити чи перетинаються хоча би два прямокутники. Прямокутники перетинаються тоді, коли площа їх перетину більша нуля.
ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести “Yes” або “No”, що є відповіддю на поставлене запитання.

Вхідні дані
3
0 0 10 10
10 0 10 10
20 0 10 10

Вихідні дані
No

Задача 11-12
Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти найбільшу кількість прямокутників, що перетинаються одночасно один з одним. Прямокутники перетинаються тоді, коли площа їх перетину більша нуля.
ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести кількість прямокутників.

Вхідні дані
3
0 0 10 10
10 0 20 10
20 0 30 10

Вихідні дані
2

Задача 11-13
Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти два прямокутники, площі яких відрізняються на найменшу величину.
ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести в одному рядку два порядкових номери прямокутників, що задовольняють умову задачі. Перший номер має бути мінімально можливим і меншим за другий.

Вхідні дані
3
0 0 10 10
0 0 20 10
0 0 35 10

Вихідні дані
1 2

Задача 11-14
Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти радіус найбільшого кола, яке може бути повністю поміщене в один із цих прямокутників.
ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести радіус шуканого кола з точністю до десятих.

Вхідні дані
3
0 0 10 10
0 0 20 10
0 0 35 10

Вихідні дані
5.0

Задача 11-15
Прямокутники задані координатами лівої нижньої вершини та довжиною і висотою сторін. Знайти діагональ такого прямокутника, що вмістив би у собі всі дані прямокутники одночасно.
ТУ. У першому рядку вхідного файлу дано ціле N (N<=1000) – кількість прямокутників. Дальше у наступних N рядках задаються Xi, Yi, Ai, Bi **(-10000<=Xi, Yi <=10000), (0<Ai,Bi<=10000)** відповідно координати вершини прямокутника, довжина і висота. У вихідний потік вивести діагональ шуканого прямокутника з точністю до сотих.

Вхідні дані
3
0 0 10 10
10 0 20 10
20 0 30 10

Вихідні дані
50.99