МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

МЕТОДИЧНІ ВКАЗІВКИ

до практичних занять
по курсу "Основи програмування і алгоритмічні мови"
(модуль "Алгоритмічна мова ТурбоПаскаль
як засіб вирішення задач САПР і ТММ")
для студентів спеціальності "Інформаційні технології проектування"

Затверджено
редакційно–видавничою
радою університету,
протокол № 3 від 28.12.09

Харків
НТУ "ХПІ"
2010

Методичні вказівки до практичних занять по курсу "Основи програмування і алгоритмічні мови" (модуль "Алгоритмічна мова ТурбоПаскаль" як засіб вирішення задач САПР і ТММ) для студентів спеціальності "Інформаційні технології проектування". / Уклад. Ю. В. Веретельник, В. І. Сєриков, Є. В. Пелешко. – Харків, НТУ "ХПІ», 2010. – 56 с.

Укладачі: Ю. В. Веретельник
В. І. Сєриков
Є. В. Пелешко

Рецензент С. М. Воронцов

Кафедра теорії і систем автоматизованого проектування механізмів і машин

ЗМІСТ

ВСТУП

Повсюдне використання персональних комп'ютерів в системах автоматизованого проектування механізмів та машин вимагає від сучасного інженера набуття навичок використання обчислювальної техніки. Однією з дисциплін, безпосередньо пов'язаних із застосуванням комп'ютерів, є "Основи програмування і алгоритмічні мови". Основним розділом курсу є вивчення методики розробки алгоритмів і мови програмування Паскаль для використання в створенні обчислювальних програм при проектуванні механізмів та машин, в курсовому і дипломному проектуванні.

Паскаль – одна з найбільш поширених мов програмування 80–90-х років, що підтримує найсучасніші методології проектування програм (низхідне, модульне проектування, структурне програмування). Нове життя в мову вдихнула фірма Борланд, що розробила на його базі сімейство Паскаль-систем, що називаються ТурбоПаскаль. Компілятори фірми підтримувались багатьма поширеними операційними системами персональних ЕОМ, такими як CP/M-80, MSDOS, MSX DOS, середовищами Windows і т. п. Пізніше з'явились аналогічні вільнорозповсюджувальні компілятори, наприклад FreePascal.

Інтегроване середовище, що забезпечує багатовіконну розробку програмної системи, обширний набір вбудованих в неї засобів компіляції і налагодження, доступний для роботи через легко освоюване меню – все це забезпечує високу продуктивність праці програміста.

Мова Паскаль розроблена з врахуванням принципів структурного програмування, яке на сучасному етапі визнане дієвим методом раціоналізації праці програміста. Для структурованих програм характерні легкість налагодження і коригування, низька частота помилок. Окрім цього, такі програми легко супроводжувати і модифікувати без участі розробників.

Паскаль має повний набір структурних типів даних, таких, як прості змінні, масиви, файли, множини, записи, записи з варіантами, посилальні змінні. Введення структурних типів даних сприяє створенню ефективних алгоритмів.

Особливо слід зазначити надійність Паскаль-програм, яка досягається інколи за рахунок надмірності, наприклад, обов'язкового опису змінних і відповідних типів. Надійність досягається також за рахунок простоти і природності конструкцій мови, відповідних логічному мисленню розробника програм.

Метою методичних вказівок є поліпшення підготовки студентів спеціальності "Основи програмування і алгоритмічні мови" в ході вивчення ними мови Паскаль для розробки обчислювальних програм при проектуванні, а також при створенні модулів для систем автоматичного проектування.

Методичні вказівки складаються з восьми розділів, в яких викладений необхідний теоретичний матеріал і наведені багаточисельні приклади і завдан-

ня, що дають можливість наочно продемонструвати практичне вживання даної теми. Послідовність тем підпорядкована завданням практики.

ТЕХНІКА БЕЗПЕКИ ПРИ РОБОТІ З ЕОМ

1. До роботи на ЕОМ допускаються студенти, що вивчили правила техніки безпеки, мають навики роботи з системою і текст програми для реалізації.

2. Підготовка ЕОМ до роботи, технічне обслуговування і ремонт проводяться персоналом кафедри, що має відповідну підготовку. Тому про будь-які неполадки в роботі ЕОМ необхідно повідомити викладача.

3. Студентам при роботі на ЕОМ дозволяється користуватися лише клавіатурою, маніпулятором «миша» і дисплеєм. Використання інших пристроїв без дозволу викладача забороняється.

4. ***Категорично забороняється*** користуватися дискетами, не перевіреними викладачем.

1. ОСНОВНІ КОМАНДИ СЕРЕДОВИЩА FREEPASCAL

1.1 Секція File

Секція містить підменю, зображене на рисунку 1.1.

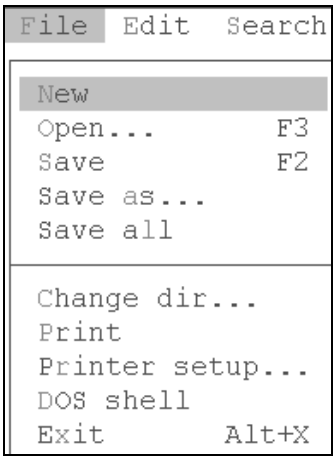

Рисунок 1.1 – Підменю секції File

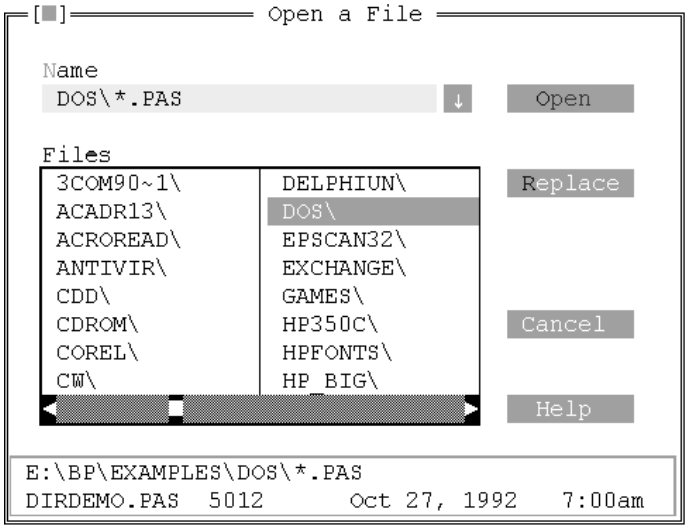

Рисунок 1.2 – Діалогове вікно Open a File

Ця секція меню надає можливості для відкриття і завантаження відповідних файлів, створення нових файлів, виходу в оболонку **DOS**, збереження файлів на диску і виходу з ТурбоПаскаля. З командами **New** і **Save as** Ви вже знайомі після вивчення теми "Текстові редактори", розглянемо решту команд.

Команда Open відкриває діалогове вікно **Open a File** (рис. 1.2), в якому можна вибрати файл для відкриття у вікні **Edit**.

Діалогове вікно містить поле введення, список файлів, інформаційну панель, стандартні «кнопки» **CANCEL** та **HELP**, дві інші кнопки (**OPEN**, **REPLACE**), список передісторії, приписаний до вхідного поля Name.

Вхідне поле **Name** – це місце для розміщення (набору за допомогою клавіатури) імені файлу, який Ви хочете завантажити у вікно редагування. Тут же можна помістити маску для використання в якості фільтру змісту поля **Files**.

Поле **Files** містить імена файлів в поточному каталозі, відповідно до маски, встановленої в полі **Name**. Наприклад, якщо в полі **Name** записане *.PAS, то в полі **Files** з'являться імена всіх файлів каталогу, що містять розширення .PAS.

Список передісторії додає до вхідного поля всі імена, які з'являлись у ньому під час останніх викликів діалогового вікна **Open a File.** У список передісторії можна увійти в тому випадку, якщо праворуч від вхідного поля **Name** Ви бачите стрілку «вниз». Для входу в список слід натискувати курсор «вниз». Цей список використовується для повторного входження в текст, в який Ви вже входили.

Вибір потрібного елементу здійснюється курсором, при цьому підсвічується вибрана позиція. Потім слід натиснути клавішу **ENTER**. Вибране ім'я файлу потрапляє у вхідне поле **Name**. Виведення тексту файлу в редакційне вікно здійснюється натисненням клавіші **ENTER**. Якщо вибір не зроблений, для виходу із списку передісторії натискуйте клавішу **ESC**.

Інформаційна панель відображує шляхове ім'я вибраного файлу, його ім'я, дату, час створення і розмір.

Кнопка **OPEN** відкриває нове вікно редагування і поміщає вибраний текст в це вікно.

Кнопка **REPLACE** замінює файл в активному вікні редагування вибраним файлом, не відкриваючи нове вікно.

Кнопка **CANCEL** відміняє всі дії і виводить з діалогового вікна.

Кнопка **HELP** виводить вікно з підказкою.

Перемикання усередині діалогового вікна здійснюються клавішею TAB. Усередині спискових полів перемикання здійснюється курсором.

Команда New Вам вже знайома.

Команда Save записує файл з активного вікна на диск. Якщо файл має ім'я за умовчанням (NONAME00.PAS), ТурбоПаскаль відкриває діалогове вікно **Save File As** для перейменування файлу і збереження його в іншому каталозі або на іншому диску.

Діалогове вікно містить вхідне поле, список файлів, інформаційну панель, стандартні перемикачі **OK**, **CANCEL**, **HELP** та список передісторії.

У вхідному полі **Save file as** записується ім'я, під яким Ви збираєтеся зберегти файл (або файлова маска для поля **Files**).

Поле **Files** змістовно не відрізняється від однойменного поля, описаного нижче в цій главі. Те ж саме відноситься і до решти елементів діалогового вікна: інформаційної панелі, перемикачів **CANCEL** і **HELP.**

Перемикач **OK** служить для підтвердження виконаних дій.

Команда Save as Вам вже знайома.

Команда Save All запам'ятовує всі файли у відкритих вікнах. Якщо серед файлів були пойменовані за умовчанням як NONAME00.PAS, FreePascal для них проведе діалог з використанням діалогового вікна **Save File as**.

Команда Change dir відкриває діалогове вікно **Change Directory**, в якому можна задати каталог, який Ви хочете зробити поточним. Поточний каталог – той, який використовується FreePascal для запам'ятовування файлів та їх пошуку.

Вхідне поле **Directory Name** призначене для введення шляхового імені нового каталогу.

Поле **Directory Tree** містить дерево каталогів. Тут Ви вказуєте ім'я каталогу, який хочете зробити поточним. Зробивши вибір курсором, натисніть клавішу **ENTER**, потім або виберіть **OK**, або вийдіть з меню клавішею **ESC**.

Перемикач **Chdir** ініціює зміну каталогу як тільки Ви вибрали ім'я нового каталогу в дереві або набрали його у вхідному полі. Якщо Ви передумали, виберіть **CANCEL**.

Перемикач **Revert** повертає до попереднього каталогу, якщо Ви ще не вийшли з діалогового вікна.

Команда Print друкує зміст активного вікна редагування. FreePascal розширює знаки TAB (замінює їх відповідним числом пробілів) і потім посилає зміст вікна у DOS Print Handler. Команда стає неправомочною, якщо вікно не може бути надруковане. Для друкування вибраного тексту наберіть **CTRL–K–P**.

Команда DOS Shell виводить із середовища ТурбоПаскаля в середовище **DOS**, для виконання команд **DOS** або входу в іншу програму. Для повернення в середовище FreePascal наберіть у відповідь на підказку **DOS** слово **EXIT**, потім натисніть клавішу **ENTER**.

1.2 Секція Edit

Секція містить підменю, зображене на рисунку 1.3.

```
Undo     Alt+BkSp
Redo
-------------------
Cut      Shift+Del
Copy      Ctrl+Ins
Paste    Shift+Ins
Clear     Ctrl+Del
-------------------
Show clipboard
```

Рисунок 1.3 – Секція Edit

Секція меню **EDIT** забезпечує команди для виділення, копіювання і вставки фрагментів тексту в редакційні вікна. Можна також відкрити «вікно вставок» **Clipboard** (вікно, в яке FreePascal поміщає для зберігання всі виділені фрагменти) для огляду або редагування його змісту.

Команда Restore Line повертає останній відредагований рядок. Вона також забирає **CTRL-Y** або текст, набраний на порожньому

рядку. Команда працює лише на останньому модифікованому або забраному рядку.

Команда Cut забирає виділений текст з документа і поміщає його в **Clipboard.** Для виділення тексту використовуйте клавішу **Shift** і клавішу управління курсором (одночасно). Для вклеювання виділеного фрагмента з **Clipboard** в інше місце використовується команда **Paste** (див. далі). Текст залишається в **Clipboard**, і його можна вклеювати необмежене число разів (поки він залишається виділеним в **Clipboard**).

Команда Copy не змінює вибраний текст, поміщаючи копію виділеного фрагмента в **Clipboard**. Виділений фрагмент може бути вклеєний в будь-яке місце з використанням команди **Paste**.

Команда Paste вставляє вибраний текст з **Clipboard** в поточне вікно за позицією курсору. Вклеюється текст, виділений в **Clipboard**.

Команда Clear видаляє вибраний текст, не поміщаючи його в **Clipboard**. Текст, забраний цією командою, втрачається. Команда **Paste** не повертає його.

Команда Show Clipboard відкриває вікно **Clipboard**, в якому запам'ятовується текст, вибраний з редакційних вікон за допомогою команд меню **EDIT** (**Cut, Copy, Copy Example**). Останній вибраний текст додається до **Clipboard**, не забираючи попереднього. Тільки що доданий текст виділений (підсвічується) в **Clipboard**, саме цей текст буде використаний FreePascal при виконанні команди **Paste**. Текст у вікні **Clipboard** можна відредагувати. Після використання команди **Paste** текст залишається у вікні **Clipboard** і може використовуватися для вклеювання скільки завгодно разів (поки він залишається підсвіченим).

1.3 Секція Search

Секція містить підменю, зображене на рисунку 1.4.

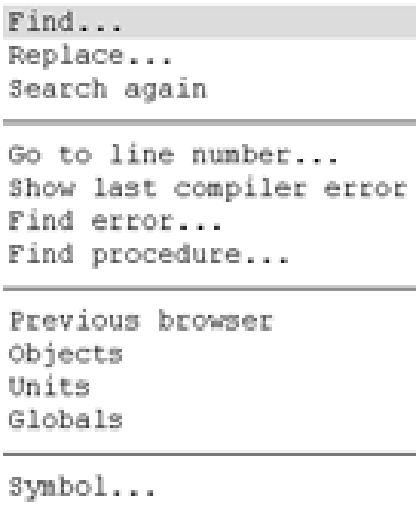

Рисунок 1.4 – Підменю секції Search

Меню **Search** забезпечує набір команд для пошуку і заміни тексту, пошуку процедур і місце розташування помилок у Вашій програмі.

Команда Find (CTRL-Q-F) відкриває діалогове вікно **Find**, що містить вхідне поле із списком передісторії, три набори перемикачів, групу керуючих установок і стандартні кнопки **OK, CANCEL** та **HELP**.

Для швидкого входу в діалогове вікно досить натиснути клавіші **CTRL-Q-F.**

Для виходу в список передісторії при висвіченому **Text to find** досить натиснути

клавішу управління курсором «вниз». Відкриється вікно, що містить список використаних Вами раніше текстів. Вибір потрібного здійснюється курсором, після чого необхідно натиснути клавішу **ENTER**. Вихід із списку при невибраній позиції здійснюється натисненням клавіші **ESC**.

Перемикання між полями, наборами і клавішами діалогового вікна проводиться клавішею **TAB**, усередині наборів – курсором.

Слід запам'ятати, що установки, відмічені прямими дужками, незалежні, тобто вони можуть бути вибрані для одночасної дії. Перемикачі, відмічені круглими дужками, альтернативні, тобто активний лише один з них.

Для зміни активності установки усередині поля досить «наїхати» на неї курсором і натискувати клавішу **ПРОБІЛ**, так саме працюють і з перемикачами. Включена установка містить знак **X** усередині прямих дужок ([X]), вимкнена не містить ([ ]). Для перемикачів таким розпізнавальним знаком є крапка усередині круглих дужок.

Вхідне поле **Text to Find** призначене для введення шуканого рядка, після чого слід вибрати **ОК** для початку пошуку.

Якщо Ви хочете ввести рядок, який Ви вже шукали, використовуйте клавішу управління курсором «вниз». Відкриється додаткове вікно, що містить список передісторії, в якому можна зробити вибір.

Поле **Directions** перемикає напрямок пошуку від поточної позиції курсору: вперед (Forward) або назад, на початок тексту (Backward). За умовчанням вибраний напрямок вперед (Forward).

Поле **Scope** визначає об'єм файлу, що піддається перегляду при пошуку. За умовчанням перемикач встановлюється в позицію **Global** – є видимим весь файл. Якщо встановити Selected Text, пошук відбуватиметься лише в межах поміченого блоку тексту в напрямку, вказаному в **Direction**.

Відмітка тексту здійснюється за допомогою блокових команд, зведених в таблицю 1.1.

Поле **Origin** задає початок пошуку. При виборі From cursor пошук починається з поточної позиції курсору в напрямку, заданому в **Direction**.

При виборі **Entire Scope** є видимим або цілий блок вибраного тексту, або весь файл (незалежно від того, де встановлений курсор).

Поле **Options** представляє групу установок, що задають типи рядків, які шукає FreePascal.

При включеному **Case Sensitive** розрізняються верхній і нижній регістри.

Таблиця 1.1

Призначення команди	Використовувані клавіші
Маркування початку блоку	Clrl–K–B
Маркування кінця блоку	Clrl–K–K
Маркування єдиного слова	Clrl–K–T
Копіювання блоку	Ctrl–K–C
Пересування блоку	Clrl–K–V
Виключення блоку	Ctrl–K–Y
Читання блоку з диска	Ctrl–K–R
Запис блоку на диск	Clrl–K–W
Показати/Скрити блок	Ctrl–K–H
Друк блоку	Clrl–K–P
Індентація блоку (відступ)	Ctrl–K–l
Деіндентація блоку (забрати відступ)	Clrl–K–U

При виборі **Whole words only** FreePascal шукає лише слова, тобто шуканий рядок повинен мати пунктуацію або пропуски з обох боків. При включеному **Regular Expression** FreePascal розпізнає в шуканому рядку спеціальні символи, показані в табл. 1.2.

Таблиця 1.2

Символ	**Тлумачення символу**
^	Циркумфлекс на початку виразу відповідає початку рядка
$	Знак долара в кінці виразу відповідає кінцю рядка
.	Крапка використовується замість будь-якого символу
*	Зірочка після символу відповідає будь-якому числу символів (включаючи нуль). Наприклад, b* відповідає bot, b, boo, а також be
+	Знак плюс після символу відповідає одній або більше (але не нульовій) появі символу. Наприклад, bo + відповідає bot і boo, але не b або be
[...]	Символи в дужках відповідають будь-якому символу, який з'явився в дужках, але не іншим
[^]	Циркумфлекс у дужках на початку рядка означає NOT. Тобто [^bot] відповідає будь-якому символу, окрім b, o або t
[–]	Дефіс в дужках означає діапазон символів. Наприклад, [b–o] відповідає будь-якому символу від b до o
\	Бекслеш перед спецсимволом повідомляє FreePascal, що цей символ потрібно сприймати буквально, а не як спецсимвол. Наприклад \^ відповідає символу ^, а не символу початку рядка

Команда Replace відкриває діалогове вікно **Replace**, в якому набирається

як текст, що підлягає заміні, так і новий текст. Як і в випадку інших діалогових вікон, до полів запису можуть бути викликані списки передісторії.

Майже всі поля ідентичні полям діалогового вікна **Find** за винятком установки **Promt on replace** в полі **Options** і кнопки **Change all**. [ ] **Promt on replace**. Установка забезпечує підказку ТурбоПаскаля кожного разу перед заміною знайденого текстового рядка на новий. Кнопка **Change all** використовується для початку пошуку. При знаходженні заданого Вами тексту FreePascal запитує, чи хочете Ви зробити заміну. При натиснутій кнопці **Change all** підтвердження потрібне для кожної заміни.

Команда Search again повторює останні команди **Find Replace**. Всі установки, зроблені у вікні **Find**, залишаються діючими.

1.4 Секція Run

Секція містить підменю, зображене на рисунку 1.5.

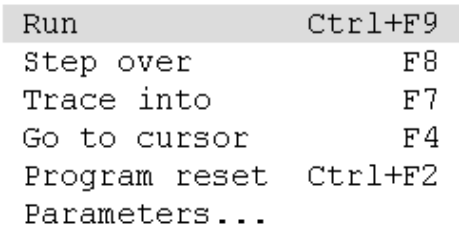

Рисунок 1.5 – Підменю секції Run

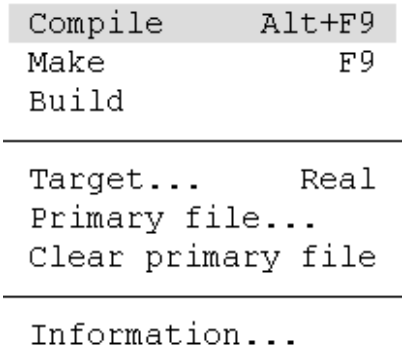

Рисунок 1.6 – Підменю секції Compile

Команди меню **Run** дозволяють запускати програму на виконання, починати і закінчувати сеанс налагодження.

Команда **Step over** дозволяє проводити покрокове виконання програми без заходу в підпрограми (у тому числі процедури і функції).

Команда **Trace into** дозволяє проводити покрокове виконання програми із заходом в підпрограми.

1.5 Секція Compile

Секція містить підменю, зображене на рисунку 1.6.

Меню **Compile** містить набір функцій, що забезпечують функції компіляції і виконання Вашої програми.

Команда Compile компілює файл в активному редакційному вікні. При компіляції або виконанні команди Make на екрані висвічується вікно стану з результатами.

Після завершення компіляції або команди Make для ліквідації вікна стану компіляції досить натиснути будь-яку клавішу. При виникненні помилки у

верхній частині редакційного вікна з'являється повідомлення.

Команда Make включає вбудований **Project Manager** для створення файлу .EXE.

Файлы рекомпілюються відповідно до наступних правил:

• Якщо Compile/Primary File містить в списку первинний файл, він компілюється, інакше компілюється останній файл, завантажений в редактор. FreePascal перевіряє всі файли, від яких залежить компільований файл.

• Якщо початковий файл для даного модуля модифікувався після того, як об'єктний код (.TPU) файлу був створений, модуль перекомпілюється.

• Якщо інтерфейс для даного модуля змінений, всі інші модулі, від нього залежні, перекомпілюються.

• Якщо модуль включає файл типа Include і він новіший, ніж файл з розширенням .TPU даного модуля, модуль перекомпілюється.

Команда Build перебудовує всі файли незалежно від їх новизни. Команда ідентична команді **Make**, але не є умовною (**Make** перебудовує лише файли, які не є поточними).

1.6 Секція Debug

Секція містить дворівневе підменю, зображене на рисунку 1.7.

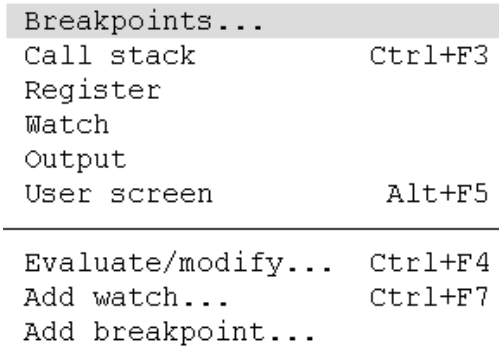

Рисунок 1.7 – Підменю секції Debug

Команди меню **Debug** управляють всіма засобами інтегрованого налагоджувача. Можна змінювати установки, зроблені за умовчанням, за допомогою меню **Options/Debugger.**

Команда Evaluate/modify. За допомогою цієї команди можна:

• обчислити значення змінної або вираз;

• спостерігати за значеннями, які приймає змінна або інший елемент даних;

• змінити значення простих елементів даних. Команда відкриває діалоговий бокс **Evaluate and Modify**.

Вхідне поле **Expression** показує слово, помічене курсором в редакційному вікні. Це слово є спостережуваним виразом за умовчанням. Для обчислення виразу за умовчанням необхідно натиснути клавішу **ENTER**. Можна відредагувати цей вираз або замінити його, в т.ч. вибравши потрібне із списку передісторії, що розкривається натисненням клавіші управління курсором «вниз».

Якщо налагоджувач може оцінити вираз, він показує значення в полі **Result.**

Якщо вираз посилається на змінну або простий елемент даних, Ви може-

те пересунути курсор в позицію **New Value** і ввести вираз як нове значення.

Для видалення вікна натисніть клавішу **ESC**. Якщо Ви змінили зміст поля **New Value** і не натискували **ENTER**, налагоджувач проігнорує зміни.

Поле **Result** висвічує значення виразу, введеного у вхідне поле Expression.

Поле **New Value** призначене для введення нового значення. Кнопка **Evaluate** використовується для обчислення виразу, введеного в бокс **Expression**.

Кнопка **Modify** використовується для зміни значення змінної або простого елементу даних, введеного в **Expression**, на значення, введене в **New Value.**

Команда Watches

*Команда **Add watch*** поміщає спостережуваний вираз у вікно **Watch**. При виборі **Add Watch** налагоджувач відкриває діалогове вікно **Add Watch**. У вхідному полі Watch expression висвічується вираз за умовчанням (то, на яке вказує курсор в редакційному вікні). Для пошуку і вибору іншого виразу (з числа, що вже використовувалися) можна відкрити список передісторії.

Якщо Ви вводите допустимий вираз натисканням клавіші **ENTER** або задіявши **ОК,** налагоджувач додає вираз і його поточне значення у вікно **Watch**.

Якщо вікно **Watch** є активним, для введення нового виразу для спостереження потрібно натискувати клавішу **INS**.

*Команди **Delete Watch** і **Edit watch.*** Ці команди стають доступними в рядку стану, коли йде робота з вікном **Watch.** Команда **Delete Watch** забирає поточний вираз **Watch** з вікна **Watch** (поточний вираз помічений у вікні Watch). Того ж ефекту можна досягти, натискуючи DEL або CTRL-Y.

Команда **Edit watch** дає можливість редагувати вираз у вікні **Watch**. Команда відкриває діалогове вікно **Edit Watch**, що містить копію поточного спостережуваного виразу. У полі **Watch expression** редагується вираз, заданий за умовчанням (виразом за умовчанням вибирається вираз, на який встановлений курсор в редакційному вікні). Можна використовувати список передісторії. Відредагований вираз замінює початкову версію при натисненні клавіші **ENTER** або виборі кнопки **ОК**.

Команда Toggle breakpoint. Цією командою можна встановити або зняти безумовну точку переривання (точку останову при налагоджуванні) на рядку, що позиціюється курсором. Встановлена точка відмічається підсвічуванням. Після повторного натиснення Ctrl-F8 точка переривання знімається.

2 ОСНОВИ МОВИ

2.1 Синтаксичні діаграми

В даний час одним з найбільш поширених формальних методів, які використовуються для опису синтаксичних правил мов програмування, є діаграми, звані синтаксичними. Н. Вірт – творець мови Паскаль, популяризував синтаксичні діаграми, і тому вони носять назву синтаксичних діаграм Вірта. На синтаксичних діаграмах використано два види чотирикутників - з прямими і закругленими кутами. У прямокутники поміщають елементи мови, значення яких має бути визначене (так звані «нетермінальні» символи). У чотирикутниках із закругленими кутами (або інколи в колах) розміщують так звані термінальні символи, або ієрогліфи мови. Значення їх визначення не потребує. Стрілки на діаграмі показують напрямок руху по діаграмі при розкритті структури поняття, записаного при вході в діаграму.

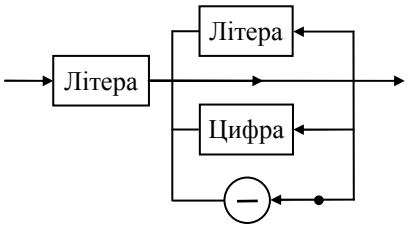

Рисунок 2.1. Приклад синтаксичної діаграми

Приклад синтаксичної діаграми для визначення синтаксису поняття «ідентифікатор» показаний на рис. 2.1.

Щоб отримати правильні граматичні конструкції мови, використовуючи синтаксичні діаграми, потрібно йти по шляхах, вказаних стрілками, від одного чотирикутника до іншого до тих пір, поки не зустрінеться вихід.

Там, де передбачено більш за один напрямок руху, можна вибрати будь-яке. Якщо по дорозі зустрілося посилання на іншу синтаксичну діаграму, то треба увійти до цієї нової діаграми, пройти по ній, вийти з неї і повернутися на старе місце в первинній діаграмі. Якщо на шляху руху зустрілася точка, то це означає, що даний шлях є характерним лише для ТурбоПаскаля і є розширенням стандарту.

У ТурбоПаскале в ідентифікаторах можуть використовуватися не лише великі, але і малі латинські літери, оскільки компілятори не роблять між ними відмінності. Діаграма для малих букв аналогічна діаграмі для великих.

2.2 Структура Паскаль-програми

За стандартом мови кожна Паскаль-програма починається із слова **Program,** за яким слідує ім’я програми. Це ім’я ідентифікує програму, але не має жодного значення усередині неї. Оператори, що складають тіло програми, мають бути обмежені так званими операторними дужками **begin** і **end**. Після **end** в кінці програми обов’язкова присутність «.» (крапки), яка служить для компілятора ознакою логічного кінця програми.

Найпростіша Паскаль-програма має вигляд:

Program First (Output);

```
begin
  Writeln ('Наша перша програма')
end.
```

Як перший рядок програми записується її заголовок. Окрім імені програми в заголовку можуть записуватися імена файлових змінних, використовуваних в даній програмі. В кінці заголовка програми обов'язково ставиться символ « ; ». З синтаксичної діаграми, зображеної на рис. 2.2, видно, що всі секції є необов'язковими, за винятком секції операторів.

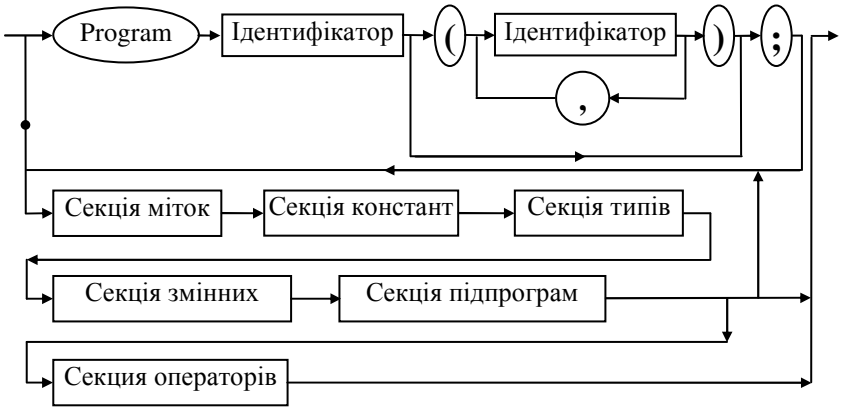

Рисунок 2.2 – Діаграма Паскаль-програми

Оператором є складений оператор, що містить один або послідовність операторів, поміщених в операторні дужки **begin** і **end**. Паскаль-програми можуть містити одну або більше операторних дужок **begin-end**. Кожній дужці **begin** повинна відповідати **end**. У програмі допустима довільна вкладеність операторних дужок. Єдиним обмеженням при використанні вкладених дужок є наступне: кожна більш зовнішня пара операторних дужок повинна повністю включати більш внутрішню пару операторних дужок, тобто вони не повинні перекриватися.

Наприклад,

```
begin
   begin
      (оператори 1–х)
   end
   begin
      (оператори 2–х)
   end
end.
```

Рекомендується першу пару операторних дужок Паскаль-програми розміщувати на тому ж рівні, що і слово **Program**. Секції опису можна зміщати праворуч від слова **Program**. З метою наочності рекомендується кожну насту-

пну пару операторних дужок зміщати праворуч.

У ТурбоПаскале заголовок програми і параметри оператора **Program** не обов'язкові. Якщо заголовок присутній, то він перевіряється на правильність синтаксису, але жодного впливу на програму не робить. Крім того, на глобальному рівні програми (тобто на обмеженому рамками процедури) FreePascal допускає довільний порядок дотримання секцій **label**, **const, type**, **var** і довільну їх кількість.

Одна з можливих структур Паскаль-програми має вигляд:

```
var    I: integer;
const  Pi=3.1415;
var    Pi2: real;
type   f2=text;
label  1,2;
type   f3=file of real;
begin
   { оператори}
end.
```

Крапка з комою використовується в Паскалі для розділення двох операторів, що йдуть один за одним.

За зарезервованим словом **begin** ніколи не ставиться крапка з комою. Можна опустити крапку з комою і перед зарезервованим словом **end**, оскільки **begin** і **end** аналогічні дужкам.

В тому випадку, якщо перед **end** все ж стоїть крапка з комою, то мається на увазі наявність пустого оператора.

2.3 Константи

Константою в Паскалі може бути ідентифікатор константи, ціле або дійсне число, рядок. Число без знаку вважається дійсним, якщо в його склад входить десяткова точка або символ «**E».** Решта всіх чисел вважається за цілі. Символ «**E»** при представленні чисел у формі з плаваючою точкою замінює основу степеня 10, а безпосередньо за ним слідує порядок.

У стандарті Паскаля передбачені іменовані константи. Іменована константа - це фіксоване значення, якому при оголошенні константи в секції констант дається ім'я. Наприклад, хай зроблено оголошення:

```
const Speed=120;
      Pi=3.14159;
      Beta=Speed;
```

Присвоювання імен константам робить програму зручнішою для розуміння і внесення виправлень в програму. При зміні констант досить змінити їх значення в секції констант. Синтаксична діаграма іменованої константи при-

ведена на рис. 2.3.

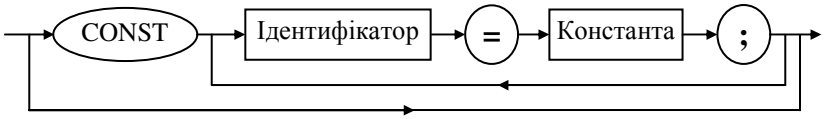

Рисунок 2.3 – Синтаксична діаграма іменованої константи

2.4 Змінні. Типи змінних

Змінна – це величина, звернення до якої проводиться за іменем. Будь-яка змінна, яка зустрічається у Паскаль-програмі, має бути оголошена в секції змінних. Виняток становлять лише зумовлені імена, тобто імена із заздалегідь визначеним компілятором смислом. Краще використовувати осмислені імена, оскільки це робить програму доступнішою для розуміння. Діаграма секції змінних приведена на рисунку 2.4.

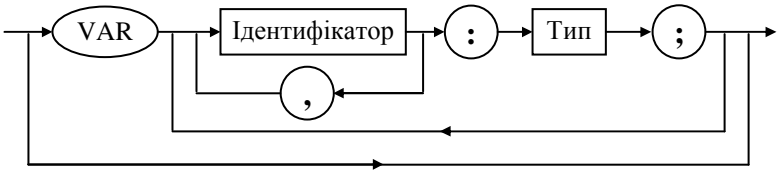

Рисунок 2.4 – Діаграма секції змінних

За допомогою оголошення імен в секції змінних встановлюється не лише факт існування змінної, але і задається її тип. У стандарті мови обумовлено чотири стандартні типи змінних:

- **integer** (цілий); значення – всі цілі числа;
- **real** (дійсний); значення – всі дійсні числа;
- **boolean** (логічний); значення – **true** або **false**;
- **char** (символьний); значення – елементи скінченної і впорядкованої множини символів.

Елементами множини символів є символи на пристроях введення-виведення, тому однієї, стандартної множини немає. Проте незалежно від реалізації для множини символів повинні виконуватися наступні умови:

– впорядкованість в алфавітному порядку множини великих латинських літер A, B, C, ... Z (безперервність для цієї множини не потрібна);

– множина десяткових цифр має бути впорядкованою і безперервною;

– в множину має бути включений символ «пробіл». Для коду ASCII всі ці вимоги виконуються. Кожен символ цього коду має невід'ємний порядковий номер. Тобто множина символів є порядковою. Множина чисел типа **real** порядковою не є.

Розмір змінних залежно від їх типа наведений в таблиці 2.1.

Таблиця 2.1

Тип	Діапазон значень	Кількість значущих цифр	Розмір у байтах
Real	2.9E–39..1.7E38	11–12	6
Single	1.5E–45..3.4E38	7–8	4
Double	5.0E–324..1.7E308	15–16	8
Extended	3.4E–4932..1.1E4932	19–20	10
Comp	–9.2E18..9.2E18	19–20	8
Shortint	–128..127	Signed	8–bit
Integer	–32768..32767	Signed	16–bit
Longint	–2147483648..2147483647	Signed	32–bit
Byte	0..255	Unsigned	8–bit
Word	0..65535	Unsigned	16–bit
Boolean			8–bit

Приклади оголошення змінних:

Var A: rea1;
BI,CI: integer;
L: boolean;
S: char:

3 ВИРАЗИ І ФУНКЦІЇ В ПАСКАЛЕ

3.1 Вирази

Н. Вірт пише: «Виразами є конструкції, що задають правила набуття значень змінними і створення шляхом застосування операцій нових значень». Вирази складаються з операндів і операцій.

Існують наступні типи операцій:

– арифметичні;

– відношення;

– логічні (булеві);

– операції над множинами.

У таблиці 3.1 наведені арифметичні операції, типи операндів і результатів.

Операції «+» або «-» (на відміну від решті арифметичних операцій) можуть бути використані з одним операндом. Ряд арифметичних операцій може використовуватися з операндами типа **integer** або **real**, результат при цьому буде типа **real**, якщо хоча б один з операндів має тип **real**.

Таблиця 3.1

Операція	Операція	Тип операндів	Тип результату
+	додавання	integer, real	integer, real
–	віднімання	integer, real	integer, real
*	множення	integer, real	integer, real
/	ділення	integer, real	real
div	ділення без остачі	integer	integer
mod	обчислення остачі від ділення	integer	integer

Приклади запису арифметичних виразів:

10**+**4 { дорівнює 14};

9*****(–3) { дорівнює –27};

10 **div** 3 { дорівнює 3};

15 **div** 3 { дорівнює 5}.

Операція **div** обчислює цілу частину частки, а його остачу можна знайти за допомогою операції **mod**. Вимога стандарту Паскаля полягає в тому, щоб результат виконання операції **mod** був позитивним, незалежно від знаків операндів. Для двох позитивних цілих операндів A і B для цієї операції виконується рівність виду:

A mod B = A – (A div B) * B

Операції «*****», «**/**», «**+**», «**–**» необов'язково оточувати роздільниками, наприклад, пробілами, але пробіли обов'язкові при вживанні зарезервованих слів **mod**, **div**, оскільки **AMODB** є ідентифікатором, а **A mod B** – виразом, використовуваним для обчислення остачі від ділення **A** на **B**.

При обчисленні арифметичних виразів прийняті наступні правила, що визначають пріоритет операторів. Операції «*», «**/**», **div**, **mod** мають вищий пріоритет, ніж операції «+» і «-». Операції з вищим пріоритетом виконуються раніше. Якщо операції мають однаковий пріоритет, то операція, що стоїть лівіше, виконується першою.

У Паскале передбачені 3 булеві (логічні) операції:

– **not** – заперечення (логічна інверсія);

– **and** – кон'юнкція (логічне множення):

– **or** – диз'юнкцій (логічне додавання);

Ці операції є фундаментом булевої логіки, розробленої в XIX столітті математиком Джорджем Булем. Результат операції **and** є істинним, якщо обидва її операнда істинні. Результат операції **or** є істинним, якщо який-небудь з її

операндів істинний. Операція **not** має один операнд і утворює його логічне заперечення. Результати логічних операцій зведені в таблицю 3.2.

Булеві вирази можуть бути складними. Операнди цих виразів мають бути булевого типу. Найпріоритетнішою булевою операцією є операція **not,** за нею слідує операція **and**, а потім операція **or.**

Приклад булевих виразів: **not** C **and** K, де C і K – булеві змінні.

Таблиця 3.2

Операція	Змінна	Значення			
	X	True	True	False	False
	Y	False	True	False	True
Заперечення	**NOT** X	False	False	True	True
Кон'юнкція (логічне множення)	X **AND** Y	False	True	False	False
Диз'юнкція (логічне додавання)	X **OR** Y	True	True	False	True

Булевий тип в мові Паскаль визначений таким чином, що false менше true (при цьому порядковий номер false дорівнює нулю, true - одиниці). Тому до булевих операндів можуть бути застосовані операції відношення. Приклади виразів:

(X=0) and (Y=0) {при X=5, Y= –2 вираз має значення false}.

У таблиці 3.3 наведений порядок виконання операцій в порядку убування пріоритетності

Таблиця 3.3

Пріоритетність	Тип операції	Ім'я
0	інверсія	not
1	мультиплікативні операції	+, /. div, mod. and
2	адитивні операції	+, –, or
3	операції відношення	=, <>, >, >=, <, <=

3.2 Стандартні функції

Для виконання обчислювальних операцій, що часто зустрічаються, і перетворення даних, що відносяться до різних типів, існують заздалегідь визначені стандартні функції. Нижче приведена таблиця типів аргументів і результатів стандартних функцій мови Паскаль. У всіх тригонометричних функціях (Sin**,** Cos**,** Arctan) аргумент задається в радіанах. Дії функцій наступні: Abs(X) – обчислює абсолютне значення X; Exp(X) – ***e*** підноситься до степені X; Ln(X) – обчислюється натуральний логарифм X; Sqr(X) – X підноситься до квадрату; Sqrt(X) – обчислюється квадратний корінь з X.

Для стандартних функцій **Arctan**, **Cos**, **Sin**, **Exp**, **Ln**, **Sqr**, **Sqrt** аргумент може бути цілим або дійсним, але результат – завжди дійсний.

Функції **Trunc**, **Round**, **Chr**, **Ord** прийнято називати функціями перетворення типів.

Trunc(X) – X має тип real. Результатом є ціла частина числа X. Наприклад, **Trunc**(6.7) = 6, **Trunc**(–6.7) = – 6.

Round(X) – X має тип real. Результатом є найближче до X ціле число. Наприклад, **Round**(4.8) = 5, **Round**(4.5) = 5.

Chr(I) – I має тип integer. Результатом є символ, порядковий номер якого на множині символів дорівнює I. Наприклад, **Chr**(66) = B.

Ord(C) – має порядковый тип. Результатом є номер (код) C у впорядкованій послідовності елементів. Наприклад, **Ord**('B')=66, **Ord**(False) = 0.

Для функцій **Ord** и **Chr** на множині символів справедливе співвідношення **Chr**(**Ord**(C))=C.

Функції **Pred, Succ** прийнято називати функціями порядкових типів.

Pred(K) – **K** має порядковий тип. Результатом є попередній **K** елемент на порядковій множині. **Pred(K)** вважається невизначеним, якщо **K** є першим за порядком елементом множини. Наприклад: **Pred**('B')='A', **Pred**(10)=9.

Succ(K) **– K** має порядковий тип. Результатом є наступний по порядку за **K** елемент на порядковій множині. **Succ**(K) вважається невизначеним, якщо **K** є останнім елементом на порядковій множині. Наприклад, **Succ**('A')='B', **Succ**(false)=true

Функції **Odd**, **Eoln**, **Eof** є булевими, оскільки результат цих функцій має тип **boolean**.

Odd(X) – X має цілий типа. Якщо X – непарний, то результат приймає значення **true**, якщо парний – **false**.

Eo1n(X) – X – файлова змінна. Результат набуває значення **true,** якщо при читанні текстового файлу досягнутий кінець поточного рядка. У решті випадків результат дорівнює **false**.

Eof(X) – X – файлова змінна. Результат набуває значення **true**, якщо при читанні текстового файлу досягнутий кінець файлу, в протилежних випадках результат дорівнює **false**.

Більшість функцій в якості аргументів можуть використовувати числа, змінні, вирази, типи яких повинні відповідати вказаним в таблиці 3.4.

Таблиця 3.4

	Integer	Real	Boolean	Char	File
Integer	Pred, Succ, Abs, Sqr	Trunc, Round	Ord	Ord	
Real	Sin, Cos, Arctan, Ln, Exp, Sqr	Abs, Sqr, Sin, Cos, Arctan, Ln, Exp, Sqrt			
Boolean	Odd		Pred, Succ		Eof, Eoln
Char	Chr			Pred, Succ	

У ТурбоПаскале до арифметичних функцій додано три функції: **Pi** – по-

вертає значення числа **Pi**; **Int** – повертає цілу частину аргументу; **Frac** – повертає дробову частину аргументу.

Для змінних порядкового типа в ТурбоПаскалі введені додаткові функції: **Dec** – зменшує значення змінної; **Inc** – збільшує значення змінної. Приклад:

```
var D, R, A: char;
    S, B: real;
    K, l: integer;
begin
    B:=1.75;
    S:=lnt(B) ; { Значення S дорівнює 1 }
    A:='P';
    R:==lnc(A) ; { Значення R дорівнює 'R' }
    D:=Dec(R) : { Значення D дорівнює 'P' }
    I:= 25;
    К := lnc(l) ; { Значення К дорівнює 26 }
end.
```

Для того, щоб піднести до будь-якого степеня число, необхідно скористатися відомою формулою $a^x = e^{x \cdot \ln a}$, тобто **exp(x*ln(a)).**

Завдання для контролю. Напишіть вирази в тому вигляді, в якому його необхідно записати в Паскаль-программі:

1) $c = d : 6 \cdot V$; 2) $\sqrt{\frac{a \cdot b^{3,1}}{a+b}}$; 3) $y = \ln(x)\frac{e^c}{a+b}$.

4 ОПЕРАТОРИ ПАСКАЛЯ

4.1 Прості і структурні оператори

Алгоритм розв'язання задачі, записаний на мові Паскаль, є послідовністю операторів. **Оператор** – це основний елемент вхідної мови. Оператори діляться на прості і складні (структурні).

Прості оператори не містять усередині себе інших операторів. **Структурні** є конструкціями, до складу яких входять прості оператори. До простих операторів відносяться оператор присвоєння, пустий оператор, оператор переходу і оператор виклику процедури.

До структурних операторів віднесені складений оператор, умовний оператор, всі оператори циклу, оператор варіанту, оператор приєднання.

Всі оператори мови Паскаль представлені на діаграмі (рис. 4.1) в наступному порядку:

– оператор присвоєння;

– оператор процедури (або виклику процедури);

– складений оператор;
– умовний оператор (**if**);
– оператор варіанту (**case**);
– оператор циклу з передумовою (**while**);
– оператор циклу з пост-умовою (**repeat**);
– оператор циклу з параметром (**for**);
– оператор приєднання (**with**);
– оператор переходу (**goto**);
– пустий оператор (оператор не виконує ніякої дії, тому на діаграмі він представлений прямою лінією).

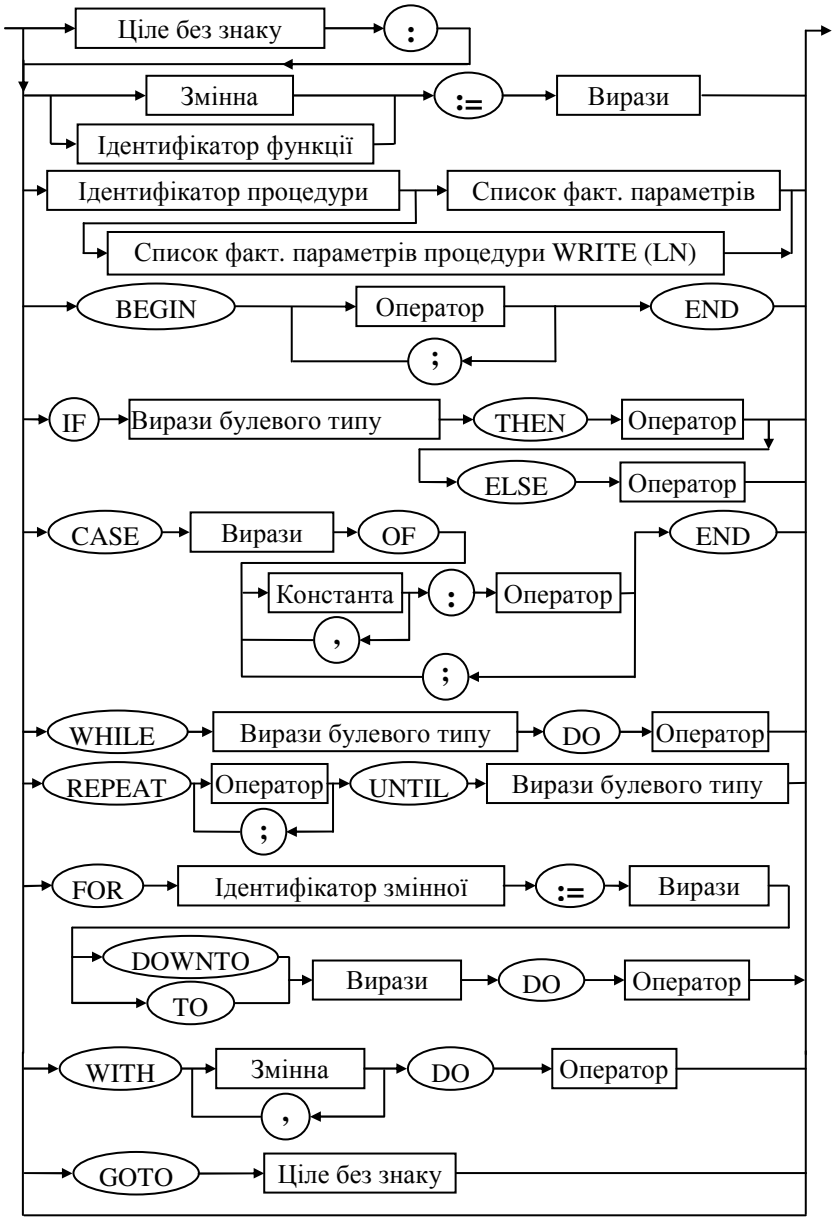

Рисунок 4.1 – Діаграма оператора

4.2 Оператор присвоєння

Форма запису оператора має вигляд: змінна **:=** вираз. Знак «**:=**» назива-

ється знаком операції присвоєння.

Дія оператора полягає в тому, що спочатку обчислюється значення виразу і після цього знов обчислене значення присвоюється змінній. Змінна і вираз повинні мати той же самий тип. Приклади:

A:=0;

К := 2 * К + М + Sqr (X + 4);

Фрагмент програми:

```
Var   A, B: Integer;
      Rez: Real;
begin
  Rez := A+B; (*При виконанні цього оператора обчислюється значення
              виразу справа (він має цілий тип). Обчислене значення пере-
              творюється у дійсний тип і після цього присвоюється змін-
              ній REZ *)
  A := Rez; (*Змінна Rez - дійсного типа, а A – цілого, тому операція при-
              своєння помилкова, оскільки в Паскалі заборонено присвоювати
              значення виразу дійсного типа змінній цілого типа *)
```

Приклад 4.1. Приведемо програму обчислення відстані між двома точками за наступною формулою [7]:

$$L_{AB}=\sqrt{(X_B^2-X_A^2)+(Y_B^2-Y_A^2)}, \qquad (4.1)$$

де прийняті наступні позначення: X_A, X_B, Y_A, Y_B – відповідні координати точок A та B.

```
Program TEST;
Var Xb,Xa,Yb,Ya,Lab: Real;
Begin
   Xb:=45; Xa:=38; Yb:=123.9;Ya:=93;
   Lab:=sqrt(sqr(xb-xa)+sqr(yb-ya));
   Writeln('Lab:=',Lab);
end.
```

4.3 Оператор переходу

Форма запису оператора: **Goto** <метка>. **Оператор переходу** – простий оператор, вказуючий, що подальша робота програми повинна продовжуватися з оператора, на якому стоїть мітка. **Мітка** - ціле число без знаку. Не більше чотирьох цифр. Мітки програми мають бути обов'язково описані в секції мітки (рис. 4.2). Оператор переходу слід використовувати у виняткових ситуаціях, коли доводиться порушувати природну структуру алгоритму.

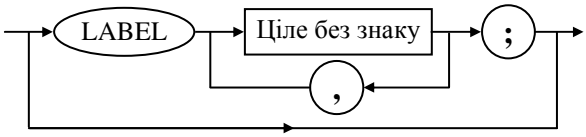

Рисунок 4.2 – Діаграма секції міток

4.4 Оператор виклику процедур

Форма запису оператора процедур має вигляд: ім'я_процедури [(список фактичних параметрів)]. Квадратні дужки означають, що список фактичних параметрів може бути відсутнім. Поняття процедури, її структура, спосіб виклику будуть розглянуті нижче. Насамперед розглянемо стандартні процедури введення-виведення, оскільки важко реалізувати просту Паскаль-програму без організації введення-виводу даних.

4.5 Введення-виведення в Паскаль-стандарті

Завдання забезпечення взаємодії людини і ЕОМ прийнято називати введенням-виведенням. У Паскалі ці функції реалізовані за допомогою чотирьох стандартних процедур: **Read, Readln, Write, Writeln.**

Для введення інформації використовуються дві процедури виду:

Read (C1, C2, ... Cn), **Readln** (C1, C2, ... Cn)

Read – процедура читає дані з ім'ям Ci. Дія процедури **Read** закінчується, як тільки вичерпається список змінних C1,..., Cn.

Дія процедури **Readln** аналогічна. Проте після вичерпання змінних (фактичних параметрів) дія процедури закінчується переходом курсору на початок наступного рядка.

Оператори виведення мають вид:

Write (P1, P2, ... Pn), **Writeln** (P1, P2, ... Pn)

P1, ..., Pn – імена змінних, що складають список виведення.

Процедура **Write** реалізує виведення значень змінних P1,..., Pn.

Процедура **Writeln** реалізує виведення значень змінних P1..., Pn в один рядок і переходить на початок наступного рядка. Процедура **Writeln** без параметрів реалізує пропуск рядка (перехід рядка).

Таблиця 4.1

Приклади	Значення	Пристрій виведення
Write (I:4)	I = 5	_5
Write (I:5, J:4)	I = – l, J = 297	_ _ _ – 1_297
Write (I, J)	I = – l, J = – 435	_– 1_.._– 435
Write (I:1, J:l)	I = – 1, J = – 435	– 1 – 435
Write(I:l, ‘,’, J:l)	I = – l, J = – 435	– 1, – 435
Write (‘Метод Гаусса’)		Метод Гаусса
Write(‘ I = ‘, l:2)	I = – 1	I = – 1

При цьому слід пам’ятати, що параметри P1...Pn можуть бути виду:

E

E : m

E : m : n ,

де **E** – вираз, значення якого необхідно вивести, m, n- вирази типа **integer**, необов’язкові параметри, які вказують відповідно ширину поля, що виводиться, і кількість дробових цифр.

Вираз **E** може бути типа real, integer, char, boolean, а також може бути рядком символів. Конструкція виду **E : m : n** може використовуватися лише для даних типа **real**. Для даних решти типів вживається конструкція виду **E : m**.

Якщо дані, що виводяться, мають менше знаків, чим **m**, то вони доповнюються зліва пробілами. Якщо більше, то виводиться стільки знаків, скільки необхідно для коректного представлення результату.

Якщо параметри **m** і **n** опущені, то маються на увазі їх деякі, залежні від реалізації, значення.

Приклад 4.2. Приведемо програму обчислення відстані між двома точками за формулою (4.1), яка читатиме значення початкових даних, а виведе результат з відповідним коментарем.

```
Program Test;
Var Xb,Xa,Yb,Ya,Lab: Real;
Begin
   Write(‘Напишіть X та Yкоординати точки A’);
   Readln(Xa, Ya);
   Write(‘Напишіть X та Yкоординати точки B’);
   Readln(Xb, Yb);
   Lab:=sqrt(sqr(xb-xa)+sqr(yb-ya));
   Writeln(‘Відстань між двома точками = ‘, fk, ‘одиниць’);
   Writeln(‘Lab:=‘,Lab);
   Readln;
end.
```

Завдання для контролю. Написати програму, яка запропонує користувачеві ввести коефіцієнти ширини зубчастих коліс швидкохідної та тихохідної ступеней редуктора, передатного числа всього редуктора і виведе на екран відповідне значення передатного числа швидкохідної ступені, що обчислюється за наступною формулою:

$$i_Б = \frac{i - \sqrt[3]{i\frac{\Psi_T}{\Psi_Б}}}{\sqrt[3]{i\frac{\Psi_T}{\Psi_Б}} - 1}, \qquad (4.2)$$

де ψ_T – коефіцієнт ширини зубчастого коліса швидкохідної ступені редуктора; $\psi_Б$ – коефіцієнт ширини зубчастого коліса тихохідної ступені редуктора; i –передатне число всього редуктора.

4.6 Умовний оператор IF

В Паскалі існують засоби, за допомогою яких можна організувати галуження в програмі. Одним з таких операторів є оператор **If**, представлений на синтаксичній діаграмі (рис. 4.1).

Дія оператора полягає в наступному: обчислюється значення виразу булевого типа. Якщо воно істинне, то виконується оператор, наступний за словом **then**. Якщо значення помилкове, виконується оператор, наступний за словом **else**. Якщо після зарезервованих слів **then** або **else** потрібно виконати декілька операторів, то їх потрібно об'єднати в складений оператор за допомогою операторних дужок **begin**, **end**.

Умовні оператори можуть бути вкладеними, ступінь їх вкладеності Паскалем не обмежений. Перед зарезервованим словом **else** не ставиться крапка з комою. Окрім цього **else** – частина в конструкції умовного оператора, яка може опускатися. В цьому випадку, якщо булевий вираз має значення **false**, жодні дії не виконуються, управління в програмі передається операторові, наступному за умовним оператором.

Необхідно відзначити, що оператор, що фігурує в синтаксичній діаграмі умовного оператора, може бути складеним.

Приклад 4.3. Напишемо програму, яка визначає і друкує найбільше з двох чисел:

```
Program Max;
Var   A, B: Integer;
begin Read (A,B);
  if A > B
     then Writeln (A)
     else Writeln (B)
end.
```

Для наочності бажано дотримуватися виду запису умовних операторів. Альтернативні частини цього оператора повинні зміщатися по відношенню до умови:

```
if Умова
        then оператор
        else оператор
```

Якщо оператори складені - то їх розташування може бути наступне:

```
if умова
  then begin
```

операторі
end
else begin
операторі
end

4.7 Оператор варіанту CASE та його розширення в ТурбоПаскалі

Більш загальним випадком умовного оператора є оператор варіанту **case**. Умовний оператор залежно від значення булевого виразу забезпечує виконання одного з двох можливих операторів. Оператор **case** дає можливість виконувати один з декількох операторів залежно від значення виразу, що фігурує на синтаксичній діаграмі (рис. 4.1) і часто називається селектором. Іншими словами **case** виконує роль перемикача залежно від значення селектора. Селектор може бути цілого, символьного або булевого типа.

Константи на синтаксичній діаграмі (рис. 4.1) називають мітками варіантів. Мітки повинні мати той самий тип, що і селектор. Після кожної мітки (послідовності міток) після знаку двокрапки записується оператор, який може бути складеним.

Якщо для деяких значень варіанту жодних дій проводити не треба, то записується пустий оператор. Кожне відмінне від інших значення селектора може з'явитися лише в одному елементі **case-**списку**.**

Дія оператора полягає в наступному: обчислюється значення виразу (селектора). Після цього виконується лише той оператор, який має мітку варіанту, значення якої збігається із значенням селектора.

Бажано, щоб будь-яке можливе значення селектора було вказане серед констант **case-**оператора**.** У ситуації, коли якась константа відсутня, управління передається операторам, наступним за службовим словом **else**.

Приклад 4.4. Розглянемо Паскаль-програму, в якій обчислюється значення умовного навантаженииия підшипника [Киркач-Баласанян], яке визначається типом підшипника (значення типів наведені в таблиці). Хай значення усіх інших параметрів, що використовуються у формулі наперед відомі та задаються константами (таблиця 4.2).

Таблиця 4.2

Номер за порядком	Тип підшипника	Формула
1	Радиальні шарикопідшипники	$P = (XVF_r + YF_a)K_б K_T$ $P=(XVF_r+YF_a)K_б K_T$
2	Підшипник з цилиндрическими роліками	$P=VF_r K_б K_T$
3	Упорно-радиальний підшипник	$P=(XF_r+YF_a)K_б K_T$
4	Упорний підшипник	$P=F_a K_б K_T$

Програма на Паскале має вид:

```
Program Test;
Const X=0.56; V=1;Fr=950;Y=1.45;Fa=588;Kb=1.2;Kt=1.05;
Var P: real;
     Tip: integer;

Begin
   Writeln ('Оберіть тип підшиника та введіть цифру');
   Writeln ('1-Радиальні шарикопідшипники');
   Writeln ('2-Підшипник з цилиндрическими роліками');
   Writeln ('3-Упорно-радиальний підшипник');
   Writeln ('4-Упорний підшипник');
   Readln (Tip);
   Case Tip of
      1: P:=(X*V*Fr+Y*Fa)*Kb*Kt;
      2: P:=V*Fr*Kb*Kt;
      3: P:=(X*Fr+Y*Fa)*Kb*Kt;
      4: P:=Y*Fa*Kb*Kt;
   else Writeln ('Значення типу невірно');
   end;
   Writeln ('Умовне навантажениия підшипника P=', P:4:2);
   Readln;
End.
```

У наведеній програмі передбачена обробка ситуації, коли замість цифр 1 – 4 зчитана інша цифра. В цьому випадку, можна видавати повідомлення «значення типу невірно». Для видачі такого повідомлення в програмі в case-оператор після рядка з міткою 4 добавлений оператор **Writeln** ('значення ....'). Оператор, наступний за словом **else**, називається оператором умовчання і виконується, якщо жодне значення мітки не відповідає значенню селектора.

4.8 Оператори циклу

Для реалізації циклів в Паскалі передбачені три оператори. Оператори **while** та **repeat** використовуються в тому випадку, якщо число повторень оператора (може бути складеним) невідоме. Оператор **for** використовується, якщо число повторень оператора відоме заздалегідь.

Дія оператора циклу з постумовою **repeat** полягає в наступному. Виконується оператор або послідовність операторів, наступних за словом **repeat**. Обчислюється значення логічного виразу, розташованого за ключовим словом **until**. Якщо значення помилкове, то повторюється виконання оператора (послідовності операторів), наступних за ключовим словом **repeat**. В разі послідо-

вності операторів операторні дужки **begin**, **end** не потрібні. Якщо значення виразу - істинно, то виконання вказаних операторів припиняється.

Приклад 4.5. Хай необхідно розрахувати питому ефективну витрату палива ДВЗ в діапазоні зміни кутової швидкості колінчастого валу двигуна від мінімального значення ω_{min} = 50 рад/с до максимального значення $\omega_{max} = \omega_N$ = 300 рад/с з кроком $\Delta\omega$ = 5 рад/с за наступною формулою

$$g_e = g_{eN}\left[1{,}55 - 1{,}55\frac{\omega}{\omega_N} + \left(\frac{\omega}{\omega_N}\right)^2\right], \qquad (4.3)$$

де g_{eN} – питома ефективна витрата палива в режимі максимальної потужності ДВЗ – приймемо рівним 80 мг/кДж.

Паскаль–програма, що реалізовує даний приклад, матиме наступний вид:

```
Program Test;
Const geN=80; Wmin=50; WN=300;dW{ΔW}=5;
Var    W, ge: integer;
begin W:=Wmin;
   Repeat
      ge:=geN*(1.55–1.55*W/WN+sqr(W/WN));
      Writeln(‘  W=‘,W,’  ge=‘,ge);
      W:=W+dW
   Until W>WN;
end.
```

Дія оператора циклу з передумовою **while** полягає в наступному. Обчислюється значення логічного виразу. Якщо воно істинне, то виконується оператор (або складений оператор), наступний за ключовим словом **do**. Якщо воно помилкове, то на цьому виконання циклу припиняється і виконується оператор, наступний за **while-конструкцией.**

Реалізація прикладу 4.5 з використанням **while-конструкции** матиме наступний вигляд:

```
Program Test;
Const geN=80; Wmin=50; WN=300;dW{ΔW}=5;
Var    W, ge: integer;
begin W:=Wmin;
   while W<=WN do
      begin
      ge:=geN*(1.55–1.55*W/WN+sqr(W/WN));
      Writeln(‘  W=‘,W,’  ge=‘,ge);
      W:=W+dW
```

```
        end;
    end.
```

Ідентифікатор оператора **for** визначає керуючу змінну циклу. Керуюча змінна набуває значень, починаючи з першого (їм є вираз після символу « := ») і закінчуючи останнім (їм є вираз після ключових слів **to** або **downto**). При кожному новому значенні керуючої змінної виконується оператор (або складений оператор), наступний за ключовим словом **do**. Типи керуючої змінної і виразу повинні співпадати. Керуюча змінна має бути порядкового типа, тобто для якого визначені функції **Succ** (в разі ключового слова **to**) або **Pred** (в разі ключового слова **downto**).

Якщо використовується ключове слово **to,** значення керуючої змінної зростає в ході виконання циклу, якщо використовується ключове слово **downto,** значення керуючої змінної убуває.

Якщо перше значення керуючої змінної перевищує останнє (для **to**) або менше останнього (для **downto**), то оператор, що стоїть після ключового слова **do**, не виконується жодного разу.

Приклад 4.6. Написати Паскаль-програму обчислення факторіалу числа.

```
Program Fact;
Var   F, I, N: integer;
begin F:=l;
   Read (N);
   for l:=l to N do F:=F*I;
   Writeln(F)
end.
```

Пример 4.7. Роздрукувати послідовність латинських літер від ‘A’ до ‘Z’

```
Program Letter;
Var   L:char; I: integer;
begin for I:=‘A’ to ‘Z’ do Write(I) ;
        Writeln
end.
```

Завдання для контролю. Написати програму, що реалізує приклад 4.5, з використанням оператора **for** .

5 ПРОСТІ І СТРУКТУРОВАНІ ТИПИ ДАНИХ

5.1 Перераховний і обмежений тип

Раніше розглядалися прості стандартні типи даних – **integer**, **boolean, char**, **real**. Перші три типи даних є порядковими, тобто до змінних цих типів застосовуються стандартні функції **Succ** и **Pred.**

Окрім цих типів в Паскалі дозволено введення нових типів. Ці нові прості типи мають бути описані в секції типів, яка представлена на синтаксичній

діаграмі (рис. 5.1). Відповідно до стандарту мови ця секція розташовується між секцією констант і секцією змінних. Введення нових типів розширює можливості мови Паскаль, підвищує читабельність програм.

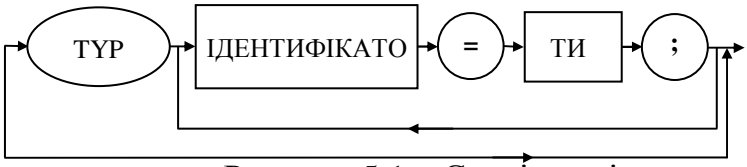

Рисунок 5.1 – Секція типів

Припустимо, що необхідно представити послідовність місяців року. Користуючись передумовними типами даних, можна використовувати прийом, який ставить у відповідність число 1 – січню, 2 – лютому і т. д. Проте таке представлення незручне, оскільки треба трактувати числові значення. Зручніше було б написати **Month := MAY.**

Щоб можна було в програмі маніпулювати з назвами місяців, а не числами, в Паскалі дозволено ввести новий тип. Змінні, що мають цей тип, можуть набувати значень місяців року.

type

Month = (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC) ;

var M: Month;

В цьому випадку створюється новий тип даних, що складається з послідовності значень, замкнених в круглі дужки. Такий тип даних прийнято називати перераховним (див. рис. 4.1, середня гілка). Змінній перераховного типа може бути призначене будь-яке значення в межах типа.

В Паскалі відсутні засоби, які б дозволяли здійснювати безпосереднє введення-виведення змінних перераховного типа. Якщо записати

M:=APR;

Write (M) ;

то слово APR виведено не буде. Тому виводити значення змінних перераховного типа потрібно програмним шляхом. Проте можна вивести вираз виду **Write(Ord(M))**. При цьому буде виведено число 3, відповідне порядковому номеру ідентифікатора **APR** в списку значень. Змінні перераховного типа можуть бути використані в булевих виразах. Окрім цього до даних цього типа застосовні операції порівняння, наприклад:

if M>MAY and M<SEP then Writeln (‘Літній місяць ‘)

Приклад 5.1. Хай перелік спеціальностей навчальної кафедри заданий наступними позначеннями: "Інформаційні технології проектування" – ITP, "Гусеничні та колісні машини" – GKM; "Електричні системи та комплекси транспортних засобів" – ESKTS. Написати програму, в якій змінній перераховного типа, присвоюється одне із значень GKM, ESKTS, ITP залежно від

введеного одного символу. Якщо введені символи помилкові, то видати відповідне повідомлення.

```
program TM;
type spes = (KGM,ETTS,AT) ;
var S: spes;
   Well: boolean;
   C: char;
begin
      Read (C);
      Well := false;
      case C of
      'G': begin Well:= true; S:= KGM; end;
      'E': begin Well:=true; S:= ETTS; end;
      'I': begin Well:=true; S:= AT; end;
      end; { case}
      Writeln(Ord(S));
      Readln;
   if not Well then Writeln ('Помилковий символ');
end.
```

Впорядкованість елементів перераховного типа визначається порядком їх наступності. Найлівіший елемент має мінімальне значення, а правіший - максимальне.

Окрім перераховного типа в Паскалі дозволено введення так званого обмеженого або інтервального типа, форма запису якого представлена на нижній гілці синтаксичної діаграми (рис. 4.1). Наприклад: **Type year = 1900..2000; letter = 'A'..'Z'; spes= 1..3;.**

Ліва і права константи задають діапазон значень, і їх називають відповідно нижньою і верхньою границею обмеженого типа. Значення цих констант повинні задовольняти умові: **ліва константа <= права константа**.

Введення обмеженого типа покращує читабельність програм, оскільки представляє діапазон значень, які може приймати змінна цього типа. Константи у визначенні обмеженого типа повинні відноситися до того ж самого базового типа (цілого, символьного, порядкового). Базовий тип констант визначає допустимість відповідних операцій над даними обмеженого типа. І перераховний, і обмежений типи змінних відносять до простого типа.

Простий – це такий тип, який може представляти лише одне значення. Наприклад, змінна типа **integer** може зберігати лише одне ціле число.

5.2 Структуровані типи даних

Поряд з простими типами Паскаль містить ряд структурованих типів да-

них, які можуть представляти сукупності значень. В Паскалі є наступні структуровані типи: масиви, файли, множини, записи.

Змінні цих типів мають структури, які визначаються Н.Віртом, як «сукупність зв'язаних даних і множину правил, що визначають їх організацію і спосіб доступу до елементів даних». Вибором тієї або іншої структури даних ми визначаємо і алгоритм обробки даних, від якого залежить ефективність їх обробки.

5.2.1 Масиви

Масив – це впорядкований набір змінних одного типа. Масиви містять фіксоване число компонент, яке задається при визначенні змінних типа масив. Тип компонент масиву – базовий. Тип індексу (індексів) в описі замикається в квадратні дужки і відноситься до простого.

Обмежень на кількість індексів немає. Індекс задає місце елементу в масиві. Тип компоненти масиву може бути будь-яким.

Приклади опису масивів:

Mas: array [1..15] of real ; (*описаний масив з 15 дійсних чисел*)

Spes: array [(GKM, ESKTS, ITP)] of integer; (*описаний масив цілих чисел, індекси елементів масиву мають перераховний тип і набувають значень назв спеціальностей GKM, ESKTS, ITP*)

B : array ['A'..'Z'] of boolean; (*описаний масив елементів булевого типа, тип індексів - обмежений символьний*)

C : array [1..3, 1..5] of real; (*описаний двовимірний масив **C** дійсних чисел, що містить три рядки і п'ять стовпців *)

D : array [(BLACK, WHITE)] of 11..20; (*описаний масив **D** цілих чисел з індексами BLACK, WHITE. Кожен елемент масиву може набувати значень від 11 до 20*)

До першого елементу масиву Mas можна звернутися Mas[l], до другого – Mas[2] і т.д. Приклад циклу, який обнуляє елементи масиву Mas, має вид:

for I:=1 to 15 do Mas[I]:=0;

Змінна Mas **[I]** називається змінною з індексом. В якості індексу може бути будь-який вираз, що має той самий тип, що і індекс.

Приклад 5.2. Написати програму для обчислення суми елементів масиву з 100 дійсних чисел.

```
Program Test;
Const N=100;
Type  Mas = array [1..N] of real;
var     Sum: real ; M: Mas;
       I: integer;
begin
   Sum:=0;
```

```
   for I:=l to N do
   Sum:=Sum+M[I];
   Writeln('Sum=',Sum);
end.
```

В Паскалі багатовимірний масив можна описати як одновимірний, елементами якого є масиви. Згадану вище матрицю **C** можна описати як одновимірний масив з трьох елементів типа рядок, кожний з яких є масивом з п'яти елементів:

```
Type Mas = array [1..3] of array [1..5] of real;
Var  A, B : Mas;
```

При послідовному записі кожного рядка матриці в пам'ять дотримуються правила розміщення двовимірного масиву по рядках. Посилання на елемент матриці, що лежить на перетині i-го рядка і j-го стовпця, виглядає таким чином: **A[I,J]**

Приклад 5.3. Написати програму обчислення добутку двовимірних дійсних матриць A розмірністю (N,M) та B розмірністю (M,L). Результуюча матриця C матиме розмірність (N,L), причому кожен її елемент обчислюватиметься за формулою

$$c_{ij} = \sum_{k=1}^{M} a_{ik} \cdot b_{kj}, \qquad (i = 1,..., N;\ j = 1,..., L).$$

```
program Test;
const  N=5; M=2; L=3;
var    A: array [1..N,1..M] of real ;
       B: array [1..M,1..L] of real ;
       C: array [1..N,1..L] of real ;
       i,j,k: integer;
begin {Ввод массивов A, B}
   for i:=1 to N do
       for j:=1 to L do
       begin C[i,j]:=0;
             for k:=1 to M do C[i,j]:=C[i,j]+A[i,k]*B[k,j];
             Writeln('C',i,j,'=',C[i,j]);
       end;
end.
```

5.5.2 Символьні рядки. Тип String в ТурбоПаскалі

Рядок – це масив, компоненти якого мають тип **char**, а тип індексу має нижню границю, рівну 1.

Рядки і рядкові константи можна використовувати в операторах присвоєння, а також у процедурах Write, Writeln як фактичні параметри.

До рядків застосовні всі 6 операцій відношень, але при цьому рядки повинні мати однакову довжину.

В ТурбоПаскалі введений тип даних **String**. Тип даних **String** інколи називають стрінговим. Приклади опису стрінгових змінних:

var Name: string[20]; Title: string[40]; Rez: string [70] ;

Пам'ять, відведена для зберігання значень змінної Name, складає 21 байт. У кожному байті зберігається один символ (символ – це цифра, літера, крапка або будь-який інший знак). Кожна літера є значенням типа **char**. Один байт змінної Name містить її поточну довжину. Це значення не повинне перевищувати 255. Наприклад:

Name := 'Автор';

Title := ' Програмування на мові Паскаль ';

При присвоєнні значення стрінгової змінної береться в лапки (апострофи, але не парні лапки). Змінна **Title** займає всього 34 байти, один байт містить її поточну довжину. У стрінгових виразах використовується лише одна операція – конкатенація (злиття), яка позначається знаком "**+**".

Приклад. **Rez := Name+Title;**

Значення виразу **Rez: 'Автор Програмування на мові Паскаль'.** До стрінгових змінних можуть застосовуватися наступні операції порівняння: **=, < >, >, >=, <, <=.** При цьому стрінгові змінні повинні мати однакову довжину, інакше фіксується помилка. Приклад:

var Age: string [3] ;...

Age := 'тридцать';...

Змінній Age буде присвоєно значення «три», оскільки максимальна довжина змінної Age не повинна перевищувати три. Зайві праві літери усікаються.

Для змінних типа String в ТурбоПаскалі допускається застосування процедур Read, Readln, Write, Writeln.

5.5.3 Записи

Запис – найбільш загальний і гнучкий структурований тип у Паскалі. Запис складається з фіксованого числа компонент, що називаються полями, які можуть бути різних типів. Цим запис істотно відрізняється від масиву, всі компоненти якого мають бути того ж самого типа.

Приклад 5.4.

```
type Date = record
      Year: integer;
      Month : 1 ..12 ;
      Day: 1..31
    end;
{Тип Date включає три поля: Year, Month, Day.}
```

```
type Book = record
        Title: string [40] ;
        Author: string [50] ;
        Entry: Date
      end;
var Dl: Date;
    b: Book;
```

Оголошення типа запис здійснюється за допомогою ключового слова **record**, за яким слідує список полів запису. Завершується опис цієї структури ключовим словом **end**. Для кожного поля мають бути вказані ім'я і тип. Ім'ям поля може бути будь-який ідентифікатор.

Тип **Book** містить три поля, одне з яких має раніше визначений тип **Date.**

В Паскалі дозволено використовувати масиви записів.

Щоб звернутися до окремої компоненті запису, необхідно задати ім'я запису, за ним крапку і відразу за крапкою написати назву потрібного поля, наприклад,

```
Dl.day := 25;
B.title := 'computer games';
B.author := 'Levy';
```

6 ПРОЦЕДУРИ І ФУНКЦІЇ

6.1 Підпрограма в Паскалі

В Паскалі існує два види підпрограм – процедури і функції. Для того щоб підпрограми ввести в Паскаль-програму, їх треба описати в секції підпрограм. Ця секція поміщається в програмі за секцією опису змінних. Кожна процедура або функція описується лише один раз, але може використовуватися багато разів.

Наявність всіх секцій (окрім секції оператора) в блоці підпрограм необов'язково. В FreePascal дозволена будь-яка кількість секцій типів в блоці підпрограм.

Кожна підпрограма (процедура або функція) має ім'я, яке визначається за правилами утворення ідентифікаторів. Список параметрів, які прийнято називати формальними, містить перелік початкових даних, з якими працює підпрограма, а також ідентифікаторів, що містять значення результатів.

У блоці функції або процедури можна використовувати будь-які змінні, описані в зовнішньому блоці або блоці головної програми. Ці змінні прийнято називати глобальними.

Окрім цього, в блоці можна описати додаткові змінні, які використовуватимуться лише в даній функції або процедурі. Ці змінні тимчасові і називаються локальними. Викликальній програмі недоступні значення локальних

змінних.

Якщо, наприклад, змінна з ім'ям **T** є глобальною, тобто була описана в головній програмі, то всі підпрограми, що входять до складу головної програми можуть звертатися до змінної **T.** Але, якщо змінна **T** була описана в конкретній підпрограмі, то в цій підпрограмі значення глобальної змінної з ім'ям **T** стає недоступним, а вибирається значення локальної змінної з ім'ям **T.**

З міркувань надійності не рекомендується використовувати однакові ідентифікатори в викликальній програмі і підпрограмі. Якщо програма (функція або процедура) є вкладеною, то до неї можна звертатися лише усередині програми (наприклад, з ім'ям **К**), в якій вона описана. Причому звернення до неї може відбуватися як із самої програми **К**, так і з описаних в **К** інших підпрограм.

6.2 Функції

Функції звичайно використовуються для опису підпрограми, в результаті виконання якої значення результату набуває одна змінна. Ця змінна – ім'я функції. Тому в заголовку функції через двокрапку описується тип функції. У тілі функції має бути оператор, який присвоює імені функції значення результату. Звернення до функції відбувається аналогічно зверненню до стандартних функцій типа SQR, SIN та ін. Звернення до функції відбувається у виразі. Для цього у виразі записується ім'я функції, услід за яким в круглих дужках перераховуються фактичні параметри, тобто параметри, які необхідні для обчислення значення функції. Для кожного формального параметра, що входить в список формальних параметрів (у заголовку функції) при зверненні до функції має бути вказаний фактичний параметр. Порядок слідування, кількість, тип формальних параметрів повинні строго відповідати порядку слідування, кількості, типу фактичних параметрів.

Приклад 6.1. Написати Паскаль-програму обчислення ефективного крутного моменту двигуна внутрішнього згоряння за формулою

$$M_e(\omega) = \frac{N_e(\omega)}{\omega} \tag{6.1}$$

для значень кутової швидкості колінчастого валу двигуна ω, що змінюється в межах від 50 рад/с до 300 рад/с з кроком $\Delta\omega$= 10 рад/с. Реалізувати за допомогою підпрограми-функції обчислення значень функції $N_e(\omega)$ за формулою:

$$N_e(\omega) = N_{eN}\left[a_\omega \frac{\omega}{\omega_N} + b_\omega\left(\frac{\omega}{\omega_N}\right)^2 - \left(\frac{\omega}{\omega_N}\right)^3\right], \tag{4.2}$$

де ω_N – максимальна кутова швидкість колінчастого валу ДВЗ - дорівнює 300 рад/с; N_{eN} – максимальна ефективна потужність ДВЗ; коефіцієнт a_Δ= 0,15. У

формулі (6.2) коефіцієнти a_ω, b_ω для двотактного дизеля відповідно дорівнюють: a_ω= 0,8; b_ω= 1,2. Програма повинна друкувати таблицю значень функції $M_e(\omega)$.

```
Program MOMENT;
Const wN=300.; aw=0.8; bw=1.2; wmin=50.; dw=10.;
Var   w, Me, NeN: real ;
Function Ne (w: real ): real ;
Var   v: real;
begin
      v:=w/wN;
      Ne:=NeN*(aw*v+sqr(v)*(bw–v));
end;
begin   w:=wmin;
      Write ('NeN=');Readln (NeN);
        Repeat
          Me:=Ne(w)/w;
          Writeln('  w=',w,'  Me=',Me);
          w:=w+dw
        Until w>wN;
end.
```

6.3 Процедури

Іноді не можна підпрограму реалізувати як функцію (наприклад, робота з масивами, сортування в пам'яті та ін.). Якщо за алгоритмом потрібна видача більш ніж одного результату, то, як правило, використовуються **підпрограми-процедури**. Відміни процедур від функцій:

– спосіб виклику процедури відрізняється від виклику функції, оскільки виклик проводиться за допомогою спеціального оператора виклику процедури;

– у заголовку процедури відсутній опис типа її імені, оскільки ім'я процедури ніяк не пов'язане з результатом (одним або декількома), що викликається нею;

– для передачі результатів процедури вживаються формальні параметри в заголовку процедури, описані за допомогою ключового слова **var**.

У описах програм-процедур і функцій розрізняються два типи параметрів – параметри-змінні і параметри-значення. Ключове слово **var** перед ідентифікатором в заголовку підпрограми позначає параметр-змінну; оскільки в процедурах ім'я процедури не використовується для передачі результату, то для цієї мети використовуються формальні параметри-змінні. У списку фактичних параметрів формальним параметрам-змінним повинні відповідати посилання, а параметрам-значенням – вирази. Формальні параметри-змінні пере-

дають свої значення відповідним фактичним параметрам при поверненні з процедури в викликальну програму. Відрізняються ці параметри-змінні від решти формальних параметрів, не помічених ключовим словом **var**, тим, що вони можуть передати значення і повернути результат.

Приклад 6.2. Написати процедуру обчислення ефективної потужності і ефективного крутного моменту двигуна внутрішнього згоряння відповідно за формулами (6.1) і (6.2) для значень кутової швидкості колінчастого валу двигуна ω, що змінюється в межах від 50 рад/с до 300 рад/с з кроком $\Delta\omega$= 10 рад/с.

```
program DVS;
Const wN=300.; aw=0.8; bw=1.2; wmin=50.; dw=10.;
Var   w, Me, Ne, NeN: real ;
procedure EFF(w: real; Var N, M: real);
Var   v: real;
begin
      v:=w/wN;
      N:=NeN*(aw*v+sqr(v)*(bw–v));
      M:=N/w;
end;
begin   w:=wmin;
      Write (‘NeN=‘);Readln (NeN);
        Repeat
          EFF(w, Ne, Me);
          Writeln(‘  w=‘,w,’  Ne=‘,Ne,’  Me=‘,Me);
          w:=w+dw
        Until w>wN;
end.
```

6.4 Процедурний тип даних

Процедурний тип даних необхідний в ситуаціях, коли при виклику старшого блоку йому в підпорядкування передається один з декількох однотипних блоків, тобто може бути вибраний будь-якій з них. Наприклад, старший блок (процедура або функція) обчислює певний інтеграл і знаходиться в модулі М, а підпорядковані блоки описують різні підінтегральні функції і задані в основній програмі.

Формат опису даних процедурного типа має наступний вид для підпорядкованих блоків-функцій

TYPE ім’я_типа = Function (список формальних параметрів): тип функції;

і для підпорядкованих блоків-процедур

TYPE ім’я_типа = Procedure (список формальних параметрів);

Змінній процедурного типа можна присвоювати імена лише тих процедур або функцій, для яких встановлений дальній виклик **{$F+}** [5,6].

Приклад 6.3. Використовуючи єдиний блок підсумовування, обчислимо з точністю E=0,0001 наступні суми:

$$S_1 = \frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \frac{1}{3\cdot 4} + ... + \frac{1}{j\cdot(j+1)} + ...;$$

$$S_2 = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + ... + \frac{1}{(2i-1)^2} + ...;$$

$$S_3 = \frac{1}{1\cdot 2\cdot 3} + \frac{1}{2\cdot 3\cdot 4} + \frac{1}{3\cdot 4\cdot 5} + ... + \frac{1}{k\cdot(k+1)\cdot(k+2)} + ...$$

```
Program Summing;
Const   E=0.001;
Type    Fun = Function (m: word): real; { оголошення процедурного типа}
Var     S1, S2, S3: real;
   Function Sum(f: Fun; E: real): real; { початок блоку підсумовування }
   Var  S: real; m: word;
   Begin        S:=0; m:=1;
        Repeat S:=S+f(m);
        m:=m+1
        until f(m)<E;
        Sum:=S;
   End;{ кінець старшого блоку }
{$F+}                          (*включення дальнього виклику *)
   Function F1(j: word): real;          { поширюється на три}
   Begin        F1:=1/(j*(j+1));        {нижче лежачих блока–функції }
   End;
   Function F2(i: word): real;
   Begin        F2:=1/sqr(2*i–1);
   End;
   Function F3(k: word): real;
   Begin        F3:=1/(k*(k+1)*(k+2));
   End;
{$F–}   (*відключення дальнього виклику *)
Begin   {початок основної програми }
  S1:=Sum(F1, E);
  S2:=Sum(F2, E);
  S3:=Sum(F3, E);
  Writeln ( 'S1=', S1, 'S2=', S2, 'S3=', S3);
```

End.

В прикладі 6.3 **F1, F2, F3** є блоками-параметрами.

Підпорядковані блоки, використовувані як параметри, не можуть бути вкладеними. Заголовки блоків – параметрів загального старшого блоку (у прикладі – **Sum**) мають бути подібні; лише імена блоків та їх параметрів (у прикладі **i,j,k**) можуть розрізнятися, останнє співпадає. Ця подібність дозволяє описати загальний вид заголовка як процедурного типа, що має на меті контроль правильності звернень до старшого блоку. Ім'я блоку в описі типа опущене як неіснуючий елемент. Наприклад, третій рядок програми потрібно читати так: "Тип **Fun** – це якась дійсна функція одного аргументу типа **word**".

Таким чином, процедурний тип вказує:

– клас підпорядкованого блоку;
– скільки у нього параметрів;
– тип і порядок запису параметрів;
– для блоку-функції – тип результату.

Приклад 6.4. Хай задані допустима похибка E і границі *a*, *b* (*a* < *b*) зони пошуку локального максимуму функції F. Складемо програму, яка за допомогою функції Xmax (описана нижче) знаходить максимальне значення ефективного крутного моменту двигуна внутрішнього згоряння Memax за формулою (6.1) на інтервалі зміни кутовій швидкості колінчастого валу двигуна ω від 50 рад/с до 300 рад/с.

```
Program Maximum;
Const   wN=300.; aw=0.8; bw=1.2;
Var     E, NeN, Memax: real;
Type    fx = Function (x: real): real;
Function Xmax (F:fx; E:real; Var a,b: real): real;
  {Начало блока поиска локального максимума}
Var     h: real;
Begin
  Repeat        h:=(b–a)/3;
        If F(a+h) < F(b–h) Then a:=a+h
                           Else b:=b–h
  Until h < E;
  Xmax:= a+h;
End;    {Конец блока поиска локального максимума}
{$F+}
Function Me (w: real ): real ;          {Функция вычисления}
Var     Ne,v: real;             {эффективного крутящего}
Begin                                   {момента ДВС}
  v:=w/wN;
```

```
  Ne:=NeN*(aw*v+sqr(v)*(bw–v));
  Me:=Ne/w;
end;
{$F–}
Begin          {начало основной программы}
  Write ('Введите точность вычисления ='); Readln (E);
  Write ('NeN='); Readln (NeN);
  Memax:= Xmax( Me, E, 50, 300);
  Writeln ('Максимальное значение эффективного крутящего момента ДВС = ', Memax);
End.
```

У даному прикладі в ролі абстрактної функції *F* виступає конкретна функція Me. Функцію Xmax можна використовувати для відшукання локального максимуму будь-якої дійсної функції одного дійсного аргументу.

7. БІБЛІОТЕКА GRAPH

7.1 Ініціалізація графічного режиму

Відзначимо перш за все, що система координат в графічному режимі не відповідає системі координат текстового режиму. Текстовий режим забезпечує, як правило, видачу символів в 25 рядків і 80 стовпців. Графічні ж режими забезпечують видачу точок, різну для різних засобів:

640 x 200 для EGA;
640 x 350 для EGA;
640 x 480 для VGA.

Для виведення графічних зображень на екран FreePascal надає користувачеві бібліотеку Graph. Це означає, що програма повинна містити оголошення цієї бібліотеки після заголовка.

```
Program Ім'я_програми;
uses Graph; { обов'язково для роботи в графічному режимі }
```

Робота програми починається з ініціалізації графічного режиму процедурою **InitGraph** і завершується процедурою **CloseGraph**.

Будь-яка програма має вид:

```
Program Ім'я_програми;
uses Graph;
var
grDriver, grMode, errCode: Integer;
{ ці змінні використовуються процедурою InitGraph}
begin { тіло програми }
grDriver := Detect; { визначення номера драйвера }
InitGraph(grDriver,grMode, 'Путь к драйверу ');
```

```
  { шлях до драйвера, наприклад, 'C:\TP\BCI'}
  errCode := GraphResult;
  if errCode = grOK then
  begin
    { режим відкритий і Ви можете працювати }
    .............
    CloseGraph; { закриває режим графіки }
  end
  else
  begin { режим не удалося відкрити. Чому?}
    writelnC...'); { у цьому місці Ви повідомляєте причину
    невдачі, яку узнаєте в результаті аналізу
    змінної errorCode}
  end;
end.
```

Змінна ErrorCode зберігає значення, отримане функцією GraphResult, яке може бути числом в діапазоні від 0 до 14.

Про безпомилкову роботу свідчить значення, рівне 0 (мнемоніка цієї константи – grOk); решта значень вказує на причину неможливості ініціалізації графічного режиму:

- 1 (grNolnitGraph) – графіка не ініціалізована (використовуйте InitGraph),
- 2 (grNotDetected) – не виявлений графічний пристрій,
- 3 (grFileNotFound) – не знайдений драйвер пристрою,
- 4 (grInvalidDriver) – невірний драйвер.

7.2 Побудова графіків функцій

При виведенні графіків на екран виникають проблеми вибору приросту аргументу і масштабування; необхідно оцінити границі зміни аргументу і функції, співвіднести дійсним їх значенням цілочислові значення номерів позицій на екрані.

Приведена нижче програма виводить на відрізку $[x_n, x_k]$ графік будь-якої заданої функції $F(x)$, для якої задані значення абсолютного мінімуму $F_{\min}$ і абсолютного максимуму $F_{\max}$ (на відрізку). Якщо фактичні значення функції опиняться більше заданого $F_{\max}$ (менше $F_{\min}$), графік обрізується зверху (знизу).

Якщо значення $F_{\min}$ та $F_{\max}$ завышены по абсолютной величине, то экран недоиспользуется по вертикали. Алгоритм побудований таким чином, що вісь ординат (у) не викреслюється, якщо $x_n > 0$ або $x_k < 0$, щоб використовувати весь екран по горизонталі.

Приклад 7.1. Накреслити графік функції $y = \frac{\sin(x)}{x}$ на відрізку $[x_n, x_k]$.

```
Program Grafic;
Uses Graph;
Const n=200; {количество точек}
Var       grDriver, grMode: Integer;
{эти переменные используются процедурой InitGraph}
     x, y, x0, y0, xm, ym, j: Integer;
     cvet, fon: Word;          {цвет и фон графики соответственно}
     xn, xk, dx, xx, Fmax, Fmin, Razmax: Real;
Function F(x: Real): Real;      {выводимая на график функция}
Begin
   if  x=0  then  F := 1  else  F := sin(x)/x;
End;
Begin { тело программы }
   Writeln ('Fmax, Fmin, xn, xk = ');
   Readln (Fmax, Fmin, xn, xk);
   Writeln ('Цвет графика, фон = ');
   Readln (cvet, fon);
   dx := (xk–xn)/n;
   Razmax := Fmax–Fmin;
   if  Fmax<0  then  Razmax := –Fmin;
   if  Fmin>0  then  Razmax := Fmax;
   grDriver := Detect;
   InitGraph(grDriver,grMode, ' ');
   SetColor (cvet);
   SetBkColor (fon);
   xm := GetmaxX;
   ym := GetmaxY;
   x0 := Round (–xn / (xk–xn)*xm);   {x0 задает положение оси y, y0 за-
   дает положение оси x }
   y0 := Round (Fmax / Razmax*ym);
   if  Fmax<0  then  y0 := 0;
   Line (0, y0, xm, y0);   {вычерчивание оси x}
   Line (x0, 0, x0, ym);   {вычерчивание оси y }
   For  j := 0  to  n  do
      Begin
        xx := xn + j*dx;
        x := Round (j / n * xm);
        y := Round (y0 – F(xx) / Razmax * ym);
```

```
      { PutPixel (x, y, cvet); } {вывод графика в виде точек}
        if  j=0  then MoveTo (x, y)  else  LineTo (x, y);
        {вывод графика в виде ломаной линии}
      End;
    Readln;
    CloseGraph;
End.
```

8. КОРОТКА ДОВІДКА

8.1 Процедури FreePascal

Append (var ім'я файлу: text) – відкриває існуючий текстовий файл для приєднання.

Arc (координата X центра, координата Y центра : Integer; початковий кут, кінцевий кут: Word; радіус: Word) – малює геометричний образ дуги.

Assign (var ім'я файлу; зовнішнє ім'я: string) – призначає ім'я зовнішнього файлу файловій змінній.

Bar (координата X лівої верхньої вершини, координата Y лівої верхньої вершини, координата X правої нижньої вершини, координата Y правої нижньої вершини: Integer) – малює геометричний образ суцільного бруса.

Bar3D (координата X лівої верхньої вершини, координата Y лівої верхньої вершини, координата X правої нижньої вершини, координата Y правої нижньої вершини : Integer, глибина: Word; вершина : Boolean) – малює геометричний образ тривимірного бруса. Параметр вершина може приймати два значення: TRUE (мнемоніка константи – TopOn) – є верхня площинка, FALSE (мнемоніка константи – TopOff) – немає верхньої площинки.

Circle (координата X центра, координата Y центра : Integer; радіус : Word) – малює геометричний образ кола.

Close (var ім'я файлу) – закриває відкритий файл.

CloseGraph – завершує роботу в графіці.

ClrScr – очищає активне вікно, заповнюючи його кольором, заданим в TextBackGround.

Dec (var змінна 1; змінна 2: Longint]) – зменшує змінну 1 на значення змінної 2, якщо друга змінна не задана, то зменшує на 1.

Delay (тривалість в мілісекундах: Word) – затримує виконання наступної операції.

Delete (var рядок: string; номер символу, з якого починається видалення: Integer; число символів, що видаляються: Integer) – видаляє підрядок з рядка.

DetectGraph (var драйвер, режим : Integer) – перевіряє техніку і визначає драйвер.

DrawPoly (число вершин: Word; var масив точок) – малює геометричний образ контуру з ліній.

Ellipse (координата X центра, координата Y центра : Integer; початковий кут,

кінцевий кут: Word; Храдіус, Yрадіус: Word) – малює геометричний образ еліптичної дуги.

Exec (повне ім'я, командний рядок: string) – виконує задану програму.

Exit – здійснює негайний вихід з поточного блоку, якщо поточним блоком є програма, вона завершується.

Ellipse (координата X центра, координата Y центра : Integer; Храдіус, Yрадіус: Word) – малює геометричний образ заповненого еліпса.

FillPoly (число вершин : Word; var масив точок) – малює геометричний образ заповненого контура.

F1oodFill (координата X, координата Y, колір границі : Word) – заповнює область.

GetDate (var рік, місяць, день, день тижня: word) – повертає набір поточної дати в операційній системі.

GetTime (var години, хвилини, секунди, соті: word) – повертає час, встановлений в операційній системі.

GoToXY (X,Y:Byte) – позиціонує курсор в текстовому режимі (усередині вікна) в точку з координатами X,Y.

GraphDefaults – відновлює графічну систему по параметрах за умовчанням, позиціонує курсор.

Halt[(код виходу: word)] – зупиняє програму.

Inc (var змінна 1 [;змінна 2: Longint]) – збільшує змінну 1 на значення змінної 2, якщо друга змінна не задана, то збільшує на 1.

InitGraph (var номер драйвера: Integer; var режим графіки Integer; шлях до драйвера: string) – ініціалізує графічну систему.

Insert (підрядок: string; var строка: string; позиція: Integer) вставляє підрядок в рядок, починаючи із заданої позиції.

Line(координата X початку лінії, координата Y початку лінії, координата X кінця лінії, координата Y кінця лінії: Integer) – малює геометричний образ лінії (з точки до точки).

LineRel (приріст по координаті X, приріст по координаті Y : Integer) – малює геометричний образ лінії (від показника до точки в приростах).

LineTo (координата X кінцевої точки, координата Y кінцевої точки: Integer) – малює геометричний образ лінії (від показника до точки в координатах).

MoveRel (приріст по X, приріст по Y: Integer) – переміщає курсор на заданий приріст.

MoveTo (координата X, координата Y : Integer) – переміщає курсор в задану точку.

OutTextXУ (координата X точки видачі тексту, координата Y точки видачі тексту: Integer; текстовий рядок: siring) – видає рядок, починаючи із заданої точки.

PieSlice (координата X центра, координата Y центра : Integer; початковий кут,

кінцевий кут: Word; радиус: Word) – малює геометричний образ заповненого сектора.
PutPixel(координата X, координата Y: Integer; код кольору: word) – встановлює піксель в задану точку.
Randomize – ініціалізує вбудований генератор випадкових чисел.
Read([var файл: text;] змінна 1 [,змінна 2,..., змінна n]) – читає одне або більше значень в одну або більше змінних.
Readln – виконує процедуру Read, потім переходить до наступного рядка файла.
Rectangle(координната X лівої верхньої вершини, координата Y лівої верхньої вершини, координата X правої нижньої вершини, координата Y правої нижньої вершини: Integer) – малює геометричний образ прямокутника.
Reset(var ім'я файлу [:file; розмір запису:word]) – відкриває існуючий файл для читання.
RestoreCrtMode – відновлює текстовий режим.
Rewrite(var ім'я файлу:file[;розмір запису:word]) – створює і відкриває новий файл для запису.
Sector(координата X центра, координата Y центра : Integer; початковий кут, кінцевий кут, Xрадіус, Yрадіус: Word) – малює геометричний образ заповненого еліптичного сектора.
SetBkColor(Колір: word) – встановлює поточний фоновий колір.
SetCo1or(колір: Word) – встановлює поточний колір.
SetGraphMode(режим : Integer) – встановлює систему в графічний режим і очищає екран, допустимий режим залежить від використовуваного устаткування.
SetLineStyle(тип лінії: word; образ: word; товщина: word) – встановлює поточну ширину і тип лінії.
Str(число [:widlh[:decimals]]; var рядок: string) – перетворює числове значення в рядок.
TextBackGround (колір: byte) – вибирає фоновий колір.
TextColor(колір: byte) – вибирає колір видаваних символів тексту в текстовому режимі.
TextMode(режим : Integer) – задає режим видачі тексту.
Val(Строка: string; var змінна; var код: Integer) – перетворює значення рядка на його цифрове значення.
Window (координата X лівого верхнього кута, координата У лівого верхнього кута, координата X правого верхнього кута, координата У правого верхнього кута: byte) – визначає текстове вікно на екрані.
Write([var файл: text;] змінна1, змінна2, ..., змінна n]) – записує одне або більше значень у файл, без необов'язкових параметрів видає на екран вказані значення .

Writeln – виконує процедуру Write, потім видасть маркер end–of–file (кінець _файла) в файл, без необов'язкових параметрів видає на екран вказані значення з нового рядка.

8.2 Функції ТурбоПаскаля

Abs (аргумент) : той самий тип, що у параметра – повертає абсолютне значення аргументу.
ArcTan (аргумент : real): real – повертає арктангенс аргумента.
Chr(номер : Byte) : Char – повертає символ із заданим порядковим номером.
Concat(рядок1 [рядок 2, рядок 3, ..., рядок n]:string): string – здійснює конкатенацію рядків.
Copy (рядок: string; номер_символа_початку_копіювання: Integer; число_копійованих_символів: Integer): string – повертає підрядок заданого рядка.
Cos (аргумент: real): real – обчислює косинус аргументу.
Eof (var ім'я_файлу): Boolean – повертає статус кінця файлу для оголошеного в type файлу.
Eoln [(var ім'я_файлу: text)]: Boolean – повертає статус кінця рядка текстового файлу.
Exp(аргумент :real): real – повертає експоненту аргументу.
Frac(число : real): real – повертає дробову частину числа.
GetGraphMode: Integer – видає поточний режим.
GetMaxColor: word – видає максимальний колір в SetColor.
GetMaxMode: Integer – видає максимальний номер режима.
GetMaxX : Integer – видає роздільну здатність по X.
GetMaxY : Integer – видає роздільну здатність по Y.
GetPixel(координата_X, координата_Y :Integer): word – видає значення кольору пікселя в заданій точці.
GetX : Integer – выдает координату X поточного показника.
GetY : Integer – выдает координату Y поточного показника.
GraphResult : Integer – видасть помилковий код для останньої графічної операції.
Int (число: real): real – повертає цілу частину числа.
KeyPressed : Boolean – повертає значення TRUE, якщо на ключовій панелі натиснута клавіша.
Ln (число: real): real – повертає натуральний логарифм аргументу.
Odd (аргумент : Longint) : Boolean – перевіряє, чи є аргумент непарним числом.
Ord(элемент): Longint – повертає порядковий номер елементу порядкового типа.
Pi: real – повертає значення числа пі.
Pos (подстрока :string; строка : string): Byte – шукає підрядок в рядку і видає номер позиції.

Random (верхня_границя : word) : тот самий тип, що у параметра – видає випадкове число в діапазоні від нуля до заданого параметра.
ReadKey : Char – прочитує символ з ключової панелі.
Round (число: real):Longint – округлює дійсне число до цілого.
Sin (аргумент: real): real – повертає синус аргументу.
Sqr (аргумент): тип той самий, що у аргумента – повертає квадрат аргументу.
Sqrt (аргумент: rea1) :rea1 – повертає квадратний корінь аргументу.
Trunc(число: real):Longint – обрізує значення типа real до значення типа Integer.
UpCase(Символ: Char) :char – перетворює символ в символ верхнього регістра.

9. ПИТАННЯ І ЗАВДАННЯ ДЛЯ КОНТРОЛЮ

Глава 1

1. З яких компонент складається інтегроване інструментальне середовище FreePascal?
2. Як запустити середовище FreePascal?
3. Назвіть пункти рядка меню середовища FreePascal.
4. Як вибрати команду з пункту меню за допомогою клавіш на клавіатурі?
5. Як створити новий документ?
6. Як вказати шлях до потрібної папки? Як зберегти файл у свою папку?
7. Як відправити програму на виконання?
8. Як відключити вікно середовища FreePascal і проглянути результат виконання програми? Як повернутися у вікно редактора?
9. Як можна затримати виконання програми до натиснення клавіші введення?
10. Як очистити екран? Змінити колір фону? Колір символів?
11. Як зберегти зміни в програмі? Як зробити резервну копію програми?
12. Як вийти з середовища FreePascal?
13. Як відкрити текст програми з Вашої папки?
14. Як в програмі переміститися на одне слово вліво? Вправо?
14. Як перейти до початку рядка? У кінець рядка?
15. Як потрапити на перший рядок? На останній?
16. Як переміститися на один екран вниз? Вгору?
17. Знайдіть і замініть один ідентифікатор іншим.
18. Як здійснити пошук потрібного фрагмента по всьому документу?
19. Що таке блок?
20. Покажіть два способи виділення блоку. Як зняти-повернути виділення?
21. Приведіть два способи переміщення, копіювання і видалення блоку.
22. Як перемикатися між відкритими вікнами?
23. Як закрити вікно документа?

Глава 2, 3, 4

1. Обчислити висоту трикутника, опущену на сторону *a,* за відомими

значеннями довжин його сторін a, b, c.

2. Визначити координату середини відрізку (a, b), якщо $a = 0.5$, $b = 2$.

3. Обчислити об'єм циліндра з радіусом основи r та висотою h.

4. Визначити відстань, пройдену фізичним тілом за час t, якщо тіло рухається з постійним прискоренням a і має в початковий момент часу швидкість V_0.

5. Визначити час вільного падіння фізичного тіла з висоти H.

6. Обчислити площу прямокутного трикутника, а також по вибору користувача:

а) довжину гіпотенузи за двома його катетами;

б) довжину одного з його катетів за гіпотенузою і другим катетом.

7. Дано натуральне n. По вибору користувача визначити:

а) Скільки цифр в числі n?

б) Чому дорівнює сума його цифр?

в) Знайти першу цифру числа n.

8. Визначити, яка з двох точок - $M_1(x_1, y_1)$ або $M_2(x_2, y_2)$ – розташована ближче до початку координат. Вивести на екран дисплея координати цієї точки.

9. Визначити, яка з двох фігур (круг або квадрат) має більшу площу. Відомо, що сторона квадрата дорівнює a, радіус круга r. Вивести на екран назву і значення площі більшої фігури.

10. Визначити, чи потрапляє точка M (X, Y) до круга радіусом r з центром в точці (X_0, Y_0).

11. Чи зможе куля радіусу R пройти в отвір у формі ромба із стороною P та гострим кутом Q?

12. Перевірити, чи можна з чотирьох даних відрізків скласти паралелограм.

13. Обчислити множину значень функції $y = x^2 + b$ для x, що змінюється від – 10 до 10 з кроком 2, при $b = 5$.

14. Обчислити k перших членів арифметичної прогресії, заданих рекурентною формулою $a_{n+1} = a_n + 2$, де a_n – n–й член арифметичної прогресії.

15. Обчислити добуток m членів арифметичної прогресії, якщо відомі значення першого члена a_1 і різниця арифметичної прогресії h.

16. Сформувати послідовність, елементи якої обчислюються за формулою $a_n = \frac{n}{n+1}$, n=1,2,...,20.

17. Обчислити значення n! для n=7.

18. Не використовуючи стандартні функції (за винятком Abs), обчислити з точністю Eps > 0: $y = e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^n}{n!} + ...$ Вважати, що необхідна точність досягнута, якщо черговий доданок по модулю менший Eps, – все пода-

льші доданки можна вже не враховувати.

19. Знайти перший степінь числа 3, що перевищує дане ціле число *a*.

20. Знайти найбільший степінь числа 2, що ділить дане ціле число *a*.

21. Перевірити, чи містить квадрат даного натурального числа *n* цифру 3 в своєму запису.

22. Знайти найменше позитивне число x, що задовольняє умові $1+x > 1$.

Глава 5

Обробити на ЕОМ масив відповідно до варіанту завдання.

Варіант	Масив	Дії	Умови та обмеження
1	2	3	4
1	X(100)	Обчислити суму і кількість елементів масиву X	$0 \le x[i] \le 1$
2	A(80)	Обчислити середнє арифметичне значення елементу масиву A	$a\ [i] > 0$
3	X(70)	Переписати елементи масиву X в масив Y і підрахувати їх кількість	$-1\ x[i] \le 1$
4	B(50)	Визначити максимальний елемент масиву B та його порядковий номер	$x\ [i] > 0$
5	C(40)	Обчислити мінімальний елемент масиву та його номер	$x\ [i] < 0$
6	D(80)	Знайти максимальний та мінімальний елементи масиву D і поміняти їх місцями	
7	Y(20)	Обчислити середнє геометричне елементу масиву Y	$y\ [i] > 0$
8	Z(30)	Розташувати в масиві R спочатку позитивні, а потім негативні елементи масиву Z.	
9	N(50)	Визначити суму елементів масиву N, кратних трьом	$n[i]/3*3 = n[i]$
10	X(N)	Обчислити суму і кількість елементів масиву X	$N \le 40$

Задание Б.

Варіант	Матриця	Дії	Умови та обмеження
1	2	3	4
1	A(10,15)	Обчислити і запам'ятати суму і число позитивних елементів кожного стовпця матриці. Результати відображувати у вигляді двох рядків	$a\ [i, j]>0$
2	A(N,M)	Обчислити і запам'ятати суми і числа негативних елементів кожного рядка матриці. Результати відображувати у вигляді двох стовпців	N <= 20 M <= 15

3	B(N,N)	Обчислити суму і число елементів матриці, що знаходяться під головною діагоналлю і над нею	N ≤12
4	C(N,N)	Обчислити суму і число позитивних елементів матриці, що знаходяться над головною діагоналлю	c[i,j]>0 N ≤12
5	D(K,K)	Записати на місце негативних елементів матриці нулі і відображувати її в загальноприйнятому виді	K ≤ 10
6	D(10,10)	Записати на місце негативних елементів матриці нулі, а на місце позитивних - одиниці. Відображувати нижню трикутну матрицю в загальноприйнятому вигляді	
7	F(N,M)	Знайти в кожному рядку матриці максимальний і мінімальний елементи і помістити їх на місце першого і останнього елементу рядка відповідно. Матрицю вивести в загальноприйнятому вигляді	N ≤20 M ≤10
8	F(10,8)	Транспонувати матрицю і вивести на друк елементи головної діагоналі і діагоналі, розташованої під головною	
9	N(10,10)	Для цілочислової матриці знайти для кожного рядка число елементів, кратних п'яти, і найбільший з отриманих результатів	n_{ij} /5*5 = n_{ij}
10	P(N,N)	Знайти в кожному рядку матриці найбільший елемент і поміняти його місцями з елементом головної діагоналі. Роздрукувати отриману матрицю в загальноприйнятому вигляді	N ≤15

Глава 6

1. Описати функцію Stepen (*x*, *n*) від дійсного *x* та натурального *n*, що обчислює (за допомогою множення) величину x^n, і використати її для обчислення $b = 2.7^k + (a+1)^{-5}$.

2. Дані відрізки *a*, *b*, *c* і *d*. Для кожної трійки цих відрізків, з яких можна побудувати трикутник, надрукувати площу даного трикутника. Визначити процедуру Plo (*x*, *y*, *z*), що друкує площу трикутника із сторонами *x, y* і *z,* якщо такий трикутник існує.

3. Описати процедуру Socr (*a*, *b*, *p*, *q*) від цілих параметрів ($b \neq 0$), яка

приводить дріб $\frac{a}{b}$ до нескоротного виду $\frac{P}{q}$.

4. Хай процедура Socr(*a, b, p, q*) від цілих параметрів (b≠0) приводить дріб $\frac{a}{b}$ до нескоротного виду $\frac{P}{q}$. Описати дану процедуру і використати її для приведення дробу $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{20}$ до нескоротного виду $\frac{c}{d}$.

5. Хай процедура maxmin (*x*, *y*) присвоює параметру *x* найбільше з дійсних чисел *x* і *y*, а параметру *y* – найменше. Описати дану процедуру і використати її для перерозподілу значень дійсних змінних *a*, *b* і с так, щоб стало $a \geq b \geq c$.

6. Описати функцію $F(m,n)=\frac{n!m!}{(n+m)!}$, де *n* і *m* – невід'ємні цілі числа.

Глава 7

1. Зобразити коло діаметром *d*, що переміщається по вертикалі через центр екрану.

2. Побудувати графік функції *y=x*cos(x)+sin(x)*, для $x \in [-4,4]$ з кроком $h = 0{,}1$.

3. Зобразити квадрат із стороною *a*, що переміщається по горизонталі на відстані 100 точок від початку координат.

4. Побудувати графік функції $y = -6x^2 + 3x$.

5. Зобразити прямокутник з довжиною основи *L* і висотою *H*, що переміщається по діагоналі екрану.

СПИСОК ЛІТЕРАТУРИ

1. Епанешников А. М. Программирование в среде Turbo Pascal 7.0. / А. М. Епанешников, В. А. Епанешников. – 3–е изд. стереотип. – М.: ДИАЛОГ МИФИ, 1998. – 282с.

2. Зуев Е. А. Программирование на языке Turbo Pascal 6.0,7.0. / Е. А. Зуев. – М.: Веста: Радио и связь, 1993. – 304с.

3. Зуев Е. А. Turbo Pascal. Практическое программирование. / Е. А. Зуев. – М.: Стрикс, 1997. – 334с.

4. Турбо Паскаль 7.0 – К.: Издательская группа BHV, 1996. – 448с. : ил.

5. Зубов В. С. Программирование на языке TURBO PASCAL (верстi 6.0 и 7.0). / В. С. Зубов. – Издание 2–е, перераб. и доп. – М.: Информационно-издательский дом «Филинъ», 1997. – 320с.

6. Довгаль С. И. Персональные ЭВМ : ТурбоПаскаль V 6.0, Объектное проектирование, Локальные сети. / С. И. Довгаль, Б. Ю. Литвинов, А. И. Сбитнев. – К.: «Информсистема сервис», 1993. – 440 с.

7. Теория механизмов и машин : учебник / Заблонский К.И., Белоконев И.М., Щекин Б.М. – К.: Выща шк., 1989. – 376 с.

Навчальне видання

Методичні вказівки до практичних занять з курсу „Основи програмування і алгоритмічні мови” (модуль „Алгоритмічна мова як засіб вирішення задач САПР і ТММ”) для студентів спеціальності "Інформаційні технології проектування"

Укладачі: **Веретельник** Юрій Вікторович
Сєриков Володимир Іванович
Пелешко Євген Віталійович

Відповідальний за випуск М. А. Ткачук
Роботу рекомендував до видання В. К. Бєлов

В авторській редакції

План 2009 р., поз. 54/

Підписано до друку __.__.10. Формат 60×84 1/16. Папір офсетний.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 3,5.
Обл. вид. арк. 4,0. Наклад 100 прим. Зам № . Ціна договірна.

Видавничий центр НТУ „ХПІ”, 61002 Харків, вул. Фрунзе, 21
Свідоцтво про державну реєстрацію ДК №3657 від 24.12.2009 р.

Друкарня НТУ „ХПІ”, 61002 Харків, вул. Фрунзе, 21