МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ПЕДАГОГІЧНИЙ
УНІВЕРСИТЕТ імені М.П.ДРАГОМАНОВА

Рамський Ю.С., Цибко Г.Ю.

ОСНОВИ ПРОГРАМУВАННЯ
(мова Паскаль)

Київ 2004

УДК
ББК

Рецензенти:

Схвалено Вченою радою Національного педагогічного університету (протокол № від)

Ю.С.Рамський, Г.Ю.Цибко. Основи програмування (мова Паскаль): Навчальний посібник. – К., 2004. –- 140 с.

Подано теоретичні відомості та інструкції до лабораторних робіт з основ програмування мовою Паскаль (в системі програмування Turbo Pascal v.5.5 і вище). Посібник складено з урахуванням досвіду викладання програмування на фізико-математичних і індустріально-педагогічних факультетах вищих педагогічних навчальних закладів. Може бути використаний у курсі “Програмування та математичне моделювання”.

Для вчителів, студентів і учнів.

Моделювання як метод пізнання. Основні поняття математичного моделювання. Етапи розв'язування задач з допомогою комп'ютера.

Курс "Програмування і математичне моделювання" входить до складу загального курсу "Інформатика".

Інформатика – наука, яка вивчає структуру і загальні властивості інформації, а також методи і засоби збирання, опрацювання, зберігання, пошуку, передавання і використання інформації в різноманітних галузях людської діяльності.

Поняття інформації інтуїтивно зрозуміле, проте не має точного означення і вважається основним, неозначуваним поняттям інформатики.

Слово "інформація" означає відомості, повідомлення, пояснення, знання, навчання, інструктаж, виклад тощо (від латинського informatio – роз'яснення, ознайомлення, обізнаність).

За визначенням академіка В.М.Глушкова, "під інформацією в сучасній науці прийнято розуміти міру неоднорідності розподілу матерії і енергії в просторі і часі".

Як самостійна наука, інформатика сформувалася в кінці 60–х років XX століття. Поштовхом до цього стало активне впровадження ЕОМ в усі галузі діяльності людей, що забезпечило кількісне і якісне зростання можливостей автоматизованого опрацювання інформації.

Як відомо, **наука** – це сфера людської діяльності, функцією якої є вироблення і використання теоретично систематизованих об'єктивних знань про дійсність.

Як і будь–яка наука, інформатика має свої об'єкт, предмет і методи.

Об'єкт науки – це фрагмент дійсності, який вивчає дана наука.

Предмет науки – це конкретизований об'єкт, тобто суттєве в об'єкті.

Метод науки – це спосіб, у який наука досліджує свій предмет.

Об'єктом вивчення інформатики є інформація (наукова, економічна, соціально–політична, технічна і ін.)

Предметом інформатики є інформаційні технології – системи методів і засобів збирання, накопичення, зберігання, пошуку, опрацювання і видачі інформації.

Методом інформатики є інформаційне моделювання.

Моделювання є одним з основних методів пізнання людиною дійсності. В сучасній науці термін "модель" використовується в найрізноманітніших значеннях. Наведемо лише два означення моделі.

1. В самому загальному смислі **моделлю** називається спеціально створена форма об'єкта для подання певних характеристик справжнього об'єкта, які підлягають дослідженню.

2**. Модель** – матеріальний об'єкт, система математичних залежностей або програма, що імітує структуру або функціонування досліджуваного об'єкта.

Моделювання – це науковий метод дослідження різноманітних об'єктів, процесів, явищ, систем шляхом побудови їх моделей, які зберігають основні, виділені особливості об'єкта дослідження.

Вивчаючи функціонування побудованих моделей, отримані дані переносять на об'єкт дослідження.

Інформаційною моделлю називається певним чином структурована інформація про досліджуваний об'єкт.

Процес побудови інформаційної моделі включає ряд етапів:

1) аналіз проблеми, вибір джерел інформації про досліджуваний об'єкт;
2) відбір потрібної інформації, аналіз відібраної інформації з допомогою відповідних критеріїв;
3) переробка відібраної інформації і її подання у вигляді, зручному для використання і прийняття рішень. На цьому етапі, в залежності від поставленої задачі, інформація може бути подана різними мовами, у тому числі і мовою математичною.

Отже, інформаційні моделі є основою для створення математичних моделей.

Математична модель – це система математичних залежностей, що описують структуру або функціонування об'єкта.

Метод математичного моделювання був відомий задовго до інформатики. З появою ЕОМ він отримав нові можливості. Математичне моделювання і інформатиці реалізується в основному як машинний (обчислювальний) експеримент.

Обчислювальний експеримент – це створення і дослідження математичних моделей за допомогою ЕОМ. В основі таких експериментів лежить задання математичних моделей досліджуваних процесів і аналіз їх поведінки в різноманітних змінюваних умовах. Таким чином експерименти з моделлю дозволяють пізнати сутність досліджуваного об'єкта.

Створення математичної моделі є важливою складовою процесу розв'язування будь–якої задачі з допомогою комп'ютера. Розглянемо цей процес докладніше.

Етапи розв'язування задач за допомогою комп'ютера

Розв'язання будь–якої практичної задачі містить ряд послідовних етапів:

1) розробка математичної моделі (математична постановка задачі);
2) вибір методу розв'язування задачі;
3) побудова алгоритму розв'язування;
4) складання програми (запис алгоритму мовою програмування);
5) введення програми в комп'ютер і перевірка її правильності (налагодження і тестування);
6) виконання програми за допомогою комп'ютера;
7) аналіз отриманих результатів.

Таким чином, на першому етапі розв'язування задачі вона набуває математичної постановки – дослідження реального об'єкта чи явища замінюється дослідженням його математичної моделі. Математична задача, відображуючи найбільш суттєві властивості реального досліджуваного об'єкта чи явища, не тотожна цьому об'єкту, а є лише наближеним його описом. У зв'язку зі складністю реальних об'єктів при побудові їх математичних моделей

доводиться використовувати ряд припущень, що дозволяють чітко поставити математичну задачу. Завдяки різним припущенням можна побудувати різні математичні моделі, що описують об'єкт чи явище з різними ступенями точності.

Метод математичного моделювання реальних явищ виник і отримав свій розвиток у фізиці. Так, ще в XVII ст. Г.Галілеєм була запропонована добре відома зараз математична модель, що описувала рух тіла, кинутого під кутом до горизонту з заданою початковою швидкістю. Перша велика математична модель у фізиці – механіка Ньютона. На початку XX ст. з'явилась спеціальна теорія відносності А.Ейнштейна, що пояснювала цілий клас нових фізичних явищ, для яких модель Ньютона вже не підходила. При цьому за умови малого значення $\lambda = v^2/c^2$, де v – власна швидкість руху, а c – швидкість світла, результати, отримані в рамках обох моделей, практично співпадали. Всі ці моделі чудово витримали випробовування практикою, а їх роль у формуванні світогляду і в практичній діяльності людей важко переоцінити. Взагалі в даний час фізика має цілу систему математичних моделей.

Впровадження математичних методів дослідження в інші науки також тісно пов'язане зі створенням математичних моделей.

Приклад. Розглянемо рух тіла маси m вздовж похилої площини з кутом α. Треба визначити час, за який тіло пройде задану довжину шляху S, якщо в початковий момент тіло рухалось зі швидкістю v_0.

Найпростішу математичну модель явища можна отримати, знехтувавши силами тертя і опору повітря. Цю модель дає другий закон Ньютона: $mg\sin\alpha = ma.$ Звідси $a = g\sin\alpha.$ Шуканий час знайдемо з рівняння:

$$S = v_0 t + \frac{gt^2 \sin\alpha}{2}.$$

Якщо врахувати дію сили тертя $F_T = kmg\cos\alpha$, де k – коефіцієнт тертя, то можна уточнити початкову математичну модель: $mg\sin\alpha - kmg\cos\alpha = ma;$

$$a = g(\sin\alpha - k\cos\alpha);$$

$$S = v_0 t + \frac{gt^2(\sin\alpha - \cos\alpha)}{2}.$$

В цьому випадку розв'язок задачі буде більш точним. Процес уточнення математичної моделі можна продовжити, врахувавши ще силу опору повітря.

На другому етапі відбувається пошук методу розв'язування сформульованої математичної задачі. Мета його – перейти від рівнянь математичної моделі до розрахункових формул, що пов'язують вхідні дані з шуканими результатами. Якщо знайти розв'язок у вигляді формули не вдається, використовують чисельні методи розв'язування задачі.

На третьому етапі розв'язування задачі подається у вигляді послідовності дій, формальне виконання яких приводить до необхідних результатів, тобто розробляється алгоритм розв'язання, який можна подати різними способами.

На четвертому етапі розроблений алгоритм записується у вигляді програми відповідною мовою програмування.

На п'ятому етапі здійснюється введення програми в ЕОМ і її налагодження та тестування. Налагодження – це пошук і виправлення

синтаксичних помилок, які виникають внаслідок порушення правил мови програмування. Тестування – виявлення змістовних (семантичних) помилок, які приводять до неправильних результатів. Тестування здійснюється шляхом розв'язування контрольної (тестової) задачі – виконання програми з такими вхідними даними, для яких відомо правильний результат.

На шостому етапі відбувається безпосереднє розв'язування задачі за допомогою ЕОМ – програма виконується з відповідними умові вхідними даними.

На сьомому етапі здійснюється аналіз здобутих результатів. Він показує, чи задоволений користувач розв'язком, чи необхідне уточнення математичної моделі і повторне розв'язування. Розробляючи програму розв'язування задачі, слід передбачити те, що результати повинні бути подані у формі, зручній для наступного аналізу.

Алгоритмічні мови. Мови програмування. Інтерпретація та компіляція. Мова програмування Паскаль: основні поняття

Введемо ряд означень, на які будемо спиратися при розгляді подальшого матеріалу.

Мова – система позначень і правил для фіксації повідомлень.

Алгоритм – точна система вказівок для здійснення послідовності дій, спрямованих на досягнення поставленої мети або розв'язування задачі.

Алгоритмічна мова – система позначень і правил, призначених для запису алгоритмів і їх виконання.

Виконавцем алгоритму може бути як людина, так і ЕОМ. Алгоритмічна мова, призначена для запису алгоритмів, що орієнтовані на виконання комп'ютером, називається мовою програмування.

Кожна алгоритмічна мова має свій алфавіт і синтаксис.

Алфавіт – сукупність символів, які можна використовувати у даній мові при описі алгоритмів.

Синтаксис – сукупність правил для запису алгоритмів даною алгоритмічною мовою.

Алгоритм, записаний мовою програмування, називається **програмою.**

Для виконання програми на комп'ютері необхідно команди мови програмування перетворити у команди, які здатен виконувати процесор комп'ютера. Існує два типи таких перетворень – інтерпретація і компіляція, які здійснюються спеціальними програмами–трансляторами.

При **інтерпретації** транслятор аналізує кожну команду програми і, якщо команда не містить синтаксичних помилок, перетворює її в машинний код і відразу виконує.

При **компіляції** вся програма аналізується і при відсутності помилок записується у вигляді машинного коду в оперативну пам'ять комп'ютера або на диск. Утворений на диску файл є виконуваним файлом і може бути виконаний засобами операційної системи без використання трансляторів.

Слід відмітити такі особливості розглянутих типів трансляції програм:

- відкомпільовані програми працюють значно швидше, ніж інтерпретовані;
- програми, розроблені для певного транслятора, не можуть виконуватись без нього;
- при інтерпретації програми легше налагоджуються;
- інтерпретовані програми простіше розробляти для виконання на різних типах комп'ютерів.

Питання для самоконтролю

1. *Що називається алгоритмічною мовою, мовою програмування?*
2. *Що таке алфавіт і синтаксис мови?*
3. *Що називається програмою?*
4. *Для чого призначені інтерпретація і компіляція?*

5. *Як відбувається інтерпретація?*
6. *Як відбувається компіляція?*

Мова програмування Паскаль: основні поняття

Мова програмування Паскаль була розроблена швейцарським вченим Ніколасом Віртом для навчання програмуванню студентів університету. Пізніше завдяки своїм якостям ця мова набула промислового характеру. Перше повідомлення про мову Паскаль з'явилось у 1971 році. Зараз фірма Borland розробляє системи програмування, засновані на Паскалі: Turbo Pascal для DOS і Delphi для Windows. Оскільки далі буде вивчатися система Turbo Pascal v 6.0, для позначення мови Паскаль будемо також використовувати абревіатуру TP.

У TP, як і у будь–якій іншій мові програмування, можна виділити чотири групи елементів: символи, слова, вирази, оператори. Розглянемо ці групи послідовно.

Символи

Алфавіт TP включає в себе великі і малі латинські літери, пропуск ' ', цифри 0–9, спеціальні символи + – * / = < > [ ] . , ( ) : ; ^ @ { } $ #, складені символи, які сприймаються як один символ (пропуски між їх складовими недопустимі): <= >= := (* *) (. .) .. (див. також таблицю 1)

Таблиця 1. Алфавіт мови програмування Паскаль

Літери латинського алфавіту	A –Z, a – z
Арабські цифри	0, 1, 2, …, 9
Спеціальні символи	< > [ ] . , ( ) : ; ^ @ { } $ # '
Знаки операцій	– (віднімання), + (додавання), * (множення), / (ділення), := (присвоювання)
Знаки відношень	< (менше), <= (не більше), > (більше), >= (не менше), = (дорівнює), <> (не дорівнює), in (є елементом множини)

Таблиця 1. Алфавіт мови програмування Паскаль

Розділові знаки	. (для відокремлення цілої частини числа від дробової), , (для відокремлення елементів списків), ; (для відокремлення операторів), (* *) або { } (для запису коментарів), ' ' (лапки для запису символьних констант), ( ) (відкриваюча та закриваюча дужки), .. (задання діапазону), [ ] (для звернення до елементів масиву) : (для відокремлення міток) $ (шістнадцятковий код), ^ (для задання посилального типу) @ (для вказання адреси певної величини) #(для отримання символу за його кодом)

Слова

Слово – це послідовність літер алфавіту, яка у даній мові має певне значення. ТР має велику кількість зарезервованих (службових) слів, які несуть певне смислове навантаження при написанні програми.

Службове слово – це послідовність літер, яка трактується транслятором як одне ціле, і завжди в одному розумінні.

Приклади. Begin, End, if, then, else, Array, for, to, do, of, Array, Set, Record, File.

Об’єкти програми на ТР (типи, змінні, константи, процедури, функції, модулі, мітки) мають імена, або ідентифікатори.

Правила утворення ідентифікаторів

1. Ідентифікатор – послідовність латинських літер та цифр (можна використовувати знак підкреслювання).
2. Обов'язково починається з літери.
3. Не може бути службовим словом.
4. Значимими є перші 64 символи ідентифікатора.
5. Бажано, щоб ідентифікатор відображав зміст величини.

Приклади. Max3, a, ALPHA, M34, Zavd1_3.

Зауваження. В ідентифікаторах мови ТР великі і малі літери не розрізняються.

Програма на ТР оперує з даними (величинами), над якими виконуються певні дії для отримання кінцевого результату. В програмі кожний елемент даних (величина) є або константою, або змінною. Основними характеристиками величини є її ім'я (ідентифікатор), значення і тип.

Поняття типу даних є одним з фундаментальних понять будь–якої мови програмування. Об'єкти, якими оперує програма (константи, змінні, функції, вирази) відносяться до певного типу. **Тип** – множина значень, яких можуть набувати об'єкти програми, і сукупність операцій над цими значеннями.

ТР є мовою з сильною системою типізації. Це означає, що всі дані, які опрацьовує програма, повинні належати до якогось заздалегідь відомого типу.

У мові ТР визначено багато стандартних типів даних, і передбачена можливість оголошення нових типів, які відповідають конкретній задачі.

Приклади. Integer (цілі числа), Real (дійсні числа), Char (одиничні символи), Boolean (логічні значення – TRUE (істина), FALSE (хибність)).

Константа – елемент даних, значення якого відоме до виконання програми і в процесі її виконання не змінюється. Для визначення констант у ТР використовується службове слово Const (від англ. Constant – константа).

Змінна – елемент даних, який може змінювати своє значення в процесі виконання програми. Для визначення змінних у ТР використовується службове слово Var (від англ. Variable – змінна).

Ідентифікатори і типи змінних, які будуть використовуватись у програмі, вказуються у розділі визначення змінних:

```
Var
  ідентифікатор: тип;
або
Var
  список ідентифікаторів: тип;
```

Приклад.

```
Var a: Integer; Var a,b,c:Integer;
Var a,b: Integer; c,d: Real;
```

Значення змінним надаються в основному блоці програми.

Ідентифікатори констант і їх значення вказуються у розділі опису констант:

```
Const
  ідентифікатор=вираз;
приклад.
Const
  Max=1000;
  Min=0;
  Interval=Max-Min+1;
  Ok=TRUE;
   SpecChar='\';
```

Типи констант автоматично визначаються компілятором і тому явно не вказуються.

Вирази

Вираз – це записане у вигляді тексту правило, що задає обчислення одного значення певного типу.

Якщо в результаті реалізації виразу дістають число, вираз називається арифметичним, якщо текст – текстовим (літерним), якщо логічне значення "істина" або "хибність" – логічним (булевим).

Оператори

Оператор – це вказівка для виконання певної дії чи скінченної послідовності дій.

Розрізняють прості і складені оператори. Прості оператори програми на ТР розділяються крапкою з комою. Складений оператор – це послідовність

простих операторів, взятих в операторні дужки – службові слова Begin (початок) і End (кінець). Перед словом End крапку з комою ставити не обов'язково.

Паскаль-програма

Програма на ТР складається з двох основних розділів:

– розділ (блок) описів;

– розділ операторів (виконуваний блок), якій обмежується операторними дужками – Begin і End. Після слова End, яке завершує програму, ставиться крапка.

Найкоротша програма на ТР має вигляд:

```
Begin
End.
```

Повний варіант програми на ТР:

```
program ім'я_програми;
uses
  список використовуваних бібліотек (модулів);
label
  список міток;
Const
  визначення констант програми;
Type
  визначення типів;
Var
  визначення глобальних змінних програми;
Begin
  основний блок програми;
End.
```

Крім конструкцій мови, програма може містити коментарі.

Коментар – це довільний текст у будь-якому місці програми (де дозволений пропуск), взятий у дужки { } або (* *).

Коментарі містять пояснювальні тексти і полегшують читання і розуміння програм.

Турбо Паскаль дозволяє програмі (тексту) управляти режимом компіляції: включати або відключати контроль помилок, використовувати або емулювати математичний сопроцесор, змінювати розподіл пам'яті і ін. Для зміни режиму використовуються ключі (директиви) компіляції: спеціальні коментари, які містять символ "$" і літеру–ключ, за якою ставиться "+" (включити режим) або "–" (відключити). Можна об'єднувати кдючі в один коментар.

Приклад. {$R–} – відключити перевірку діапазонів індексів масивів;

{$N+} – використовувати сопроцесор 80X87 і т.д.

{$N+,R–}

Відкриваючі дужки, символ "$" і ключ повинні бути написані без пропусків між ними.

Такі конструкції можна вставляти в програму, і при компіляції вона сама буде задавати режим роботи компілятора. Ключі поділяються на глобальні (раз встановлений режим вже не може бути змінений) та локальні (можна змінювати режими для різних частин програми). Глобальні ключі повинні стояти на початку програми, а локальні – де потрібно програмісту.

Деякі ключі задають не режим, а компонування програми із зовнішніх складових частин.

Наприклад, ключ {$I Ім'яФайлуВключення} називається командою включення в програму зовнішнього текстового файла. Тут замість знаків "+" або "–" стоїть пропуск і ім'я файла. Ця команда вказує компілятору вважати заданий зовнішній файл частиною програми. Це дещо уповільнює процес компіляції, але може зекономити багато місця в програмі, так як вона перетворюється в "ланцюжок" різних файлів на диску.

Приклад. {$I incproc.pas}

Тексти, що включаються, самі можуть містити команди включення. Максимальний рівень вкладеності при цьому дорівнює восьми.

Питання для самоконтролю.

1. *Які символи входять до алфавіту мови ТР?*
2. *Що називається службовим словом? Наведіть приклади.*
3. *Для чого потрібні ідентифікатори? Назвіть правила утворення ідентифікаторів.*
4. *Які основні характеристики мають величини, якими оперує програма?*
5. *Що таке тип даних?*
6. *Що називається константою? Як визначаються константи у програмі на ТР?*
7. *Що називається змінною? Як визначаються змінні у програмі на ТР?*
8. *Що таке вираз? Наведіть приклади виразів.*
9. *Що таке оператор?*
10. *Що таке простий і складений оператори? Що таке операторні дужки?*
11. *Яку структуру має Паскаль-програма?*
12. *Що таке коментар? Для чого призначені коментарі? Як вони записуються?*

Основні оператори мови Паскаль. Оператор присвоювання

Оператор присвоювання призначений для надання значень змінним.

Формат оператора:

ім'я_змінної := вираз;

Тип змінної повинен співпадати з типом виразу. Винятком є випадок, коли змінній дійсного типу присвоюється значення цілочисельного виразу.

Приклад.

```
Var A,B: Real; C, D: Integer; E: Char;
Begin
  A:=3.5; B:=-10; C:=4; D:=0; E:='*'; {правильні присвоювання}
  A:=(B-3*C)/(D+54); D:=(C+15)*3; {правильні присвоювання}
  C:=A;  D:=C/10; E:=B; {неправильні присвоювання}
```

Оператори введення і виведення

Крім оператора присвоювання, змінним надаються значення за допомогою операторів введення.

Формат операторів:

Read(список введення);

Readln(список введення);

Список введення = ім'я_змінної1[, ім'я_змінної2, …, ім'я_змінноїN]

В квадратних дужках вказуються необов'язкові параметри.

Виконання оператора:

Елементи списку введення вводяться з клавіатури через пропуск або Enter. При використанні оператора Read після введення останнього елемента списку позиція введення (курсор) залишається в тому ж рядку. При використанні Readln позиція введення поміщується на початок наступного рядка.

Список введення може бути відсутнім. В цьому випадку програма зупиняє свою роботу до введення з клавіатури будь–якої інформації. Ця властивість використовується для затримки на екрані результатів роботи програми.

Приклад. Read(A,B,C); Readln(m,n); Readln;

Для виведення результатів роботи програми на екран монітора використовуються оператори виведення.

Формат операторів:

Write(список виведення);

Writeln(список виведення);

Список виведення = вираз1[, вираз2, виразN]

Елементом списку виведення може бути вираз цілого, дійсного, символьного, рядкового, логічного типу, або типів, утворених з них. Якщо список виведення відсутній (Writeln;), на екран виводиться порожній рядок.

Різниця між операторами Write і Writeln аналогічна різниці між Read і Readln.

Розглянемо особливості дії операторів Write і Writeln при різних форматах елементів списку виведення (таблиця 2)

Таблиця 2. Виведення даних операторами Write і Writeln

Тип виразу	**Формат**	**Дія оператора виведення**
INTEGER(I) (цілочисельний)	**I**	**виводиться десяткове подання значення I**
INTEGER(I) (цілочисельний)	**I:N**	**виводиться десяткове подання значення I і вирівнюється за правою межею поля шириною N символів**
REAL (R) (дійсний)	**R**	**виводиться десяткове подання значення R, вирівнюється за правою межею поля шириною 18 символів у форматі з плаваючою крапкою**
	R:N	**виводиться десяткове подання значення R, вирівнюється за правою межею поля шириною N символів у форматі з плаваючою крапкою**
	R:N:M	**виводиться десяткове подання значення R, вирівнюється за правою межею поля шириною N символів у форматі з фіксованою крапкою з M цифрами після десяткової крапки**
CHAR (CH) (символьний)	**CH**	**виводиться символ CH**
	CH:N	**символ CH виводиться з правого боку поля шириною N символів (символу CH передує N–1 пропусків)**
STRING (S) (рядковий)	**S**	**виводиться рядок S**
	S:N	**рядок S виводиться вирівняним за правою межею (йому передує N – LENGTH(S) пропусків)**
BOOLEAN(L) (логічний)	**L**	**в залежності від значення виводиться TRUE або FALSE**
	L:N	**в залежності від значення виводиться TRUE або FALSE, вирівняне за правою межею поля N символів**

Питання для самоконтролю.

1. *Для чого призначений оператор присвоювання? Вкажіть його формат і наведіть приклади використання.*
2. *Для чого призначений оператор введення? Які існують його модифікації? Як вони виконуються?*
3. *Для чого призначений оператор виведення? Які існують його модифікації? Як вони виконуються?*

Стандартні типи даних та операції над ними

Стандартні типи даних описані в стандартній бібліотеці TP, яка автоматично підключається до кожної програми. Тому такі типи не потребують опису в розділі визначення типів Паскаль-програми. Всі допустимі в мові TP типи поділяються на прості (скалярні, неструктуровані) і складені (структуровані).

Таблиця 3. Стандартні скалярні типи

Тип	**Позначення**	**Приклади**
Цілі числа	**Integer**	**5 –3 0 +123**
Цілі довжиною 1 байт	**Byte**	**2 67 125**
Короткі цілі	**ShortInt**	**–127 25 100**
Довгі цілі	**LongInt**	**200000 340005**
Цілі довжиною 1 слово	**Word**	**65534 345**
Дійсні числа	**Real**	**23.56 12E2 –3E–13**
Дійсні одинарної точності	**Single**	**1.45 –2E+33**
Дійсні подвійної точності	**Double**	**0.00005 34565.565**
Зовнішні або розширені	**Extended**	
Логічний (булевський) тип	**Boolean**	**TRUE FALSE**
Символьний (літерний) тип	**Char**	**'d' 'v' '$' '5' 'ф'**
Посилальний тип	**Pointer**	

Таблиця 4. Структуровані типи

Рядковий тип	String	'Computer' '12345'
Масиви	Array	
Записи	Record	
Множини	Set	[2,3,5,7,11] ['s', 'r', 'd']
Файли	File, File of	
Списки		
Черги		
Стеки		

Числа у TP подаються:
1. В десятковій системі числення (діапазон для цілих –32768..32767, для дійсних
1E–38..1E+38).
2. В шістнадцятковій системі числення (діапазон $0000..$FFFF).

Таблиця 5. Подання цілих чисел

Тип цілих	Діапазон	Формат
Byte (коротке довжиною 1 байт)	0..255	1 байт без знаку
ShortInt (коротке ціле)	-128..128	1 байт зі знаком
Word (ціле довжиною 1 слово)	0..65535	2 байти без знаку
LongInt (довге ціле)	-2147483648.. 2147483647	4 байти зі знаком
Integer (ціле)	-32768..32767	2 байти зі знаком

Таблиця 6. Подання дійсних чисел

Тип дійсних	Діапазон	Кількість значущих цифр, формат
Real (дійсне)	2.9E-39..1.7E+38	11-12, 6 байтів
Single (одинарна точність)	1.5E-45..3.4E+38	7-8, 4 байти
Double (подвійна точність)	5E-324..1.7E308	15-16, 8 байтів
Extended (зовнішні або розширені)	3.4E-4932..1.1E4932	19-20, 10 байтів

Змінні і константи всіх типів використовуються у виразах. Вираз задає порядок виконання дій над величинами і складається з операндів (констант, змінних, звернень до функцій), круглих дужок і знаків операцій. Операції визначають дії, які треба виконати над операндами. Наприклад, у виразі (X+Y–10) X, Y і 10 – операнди, "+" і "–" – знаки операцій додавання і віднімання. У найпростішому випадку вираз може складатися з одної змінної або константи. Круглі дужки ставляться так, як у звичайних арифметичних виразах для управління порядком виконання операцій. Використання круглих дужок навіть там, де в них немає необхідності з точки зору синтаксису, є прийнятним, якщо вони роблять порядок обчислень візуально більш чітким і зрозумілим.

Таблиця 7. Операції для величин цілого типу

Назва	**Призначення**	**Приклади**
NOT	інверсія числа	NOT 0 = +1; NOT –15 = 14
SHL	циклічний зсув вліво	3 SHL 2 = 12; 2 SHL 7 = 256
SHR	циклічний зсув вправо	3 SHR 2 = 0; 257 SHR 7 = 2

*	множення	4 * 2 = 8; 134 * 12 = 1608
DIV	ділення націло	5 DIV 2 = 2; 2 DIV 3 = 0
/	ділення (результат – не ціле!)	10 / 2 = 5; 997 / 4 = 249.25
MOD	знаходження остачі від ділення	13 MOD 5 = 3; –13 MOD 5=–3

Таблиця 7. Операції для величин цілого типу

Назва	Призначення	Приклади
AND	логічне множення	5 AND 2 = 0; 15 AND 7 = 7
XOR	виключаюче "АБО"	5 XOR 2 = 7; 15 XOR 7 = 8
+	додавання	5 +2 = 7; –3+2 = –1
–	віднімання	11 – 7 = 4; –4 – 9 = –13
OR	логічне додавання	5 OR 2 = 7; 15 OR 7 = 15
=; <>; <; >; <=; >=	операції відношення	3 > 4 → FALSE; 5 <= 8 → TRUE

Таблиця 8. Функції для величин цілого типу

Назва	Призначення	Приклади
ABS(i)	Знаходження абсолютного значення величини	ABS(–48) = 48; ABS(48) = 48
SQR(r)	Піднесення до квадрату	SQR(4) = 16; SQR(11) = 121
TRUNC(r)	Знаходження цілої частини числа	TRUNC(8.53) = 8; TRUNC(–9.2) = –9
ROUND(r)	Знаходження найближчого цілого (з округленням)	ROUND(8.53) = 9; ROUND(8.3) = 8; ROUND((–8.53) = –9
PRED(i)	Знаходження попереднього значення	PRED(6) = 5; PRED(–6) = –7
SUCC(i)	Знаходження наступного значення	SUCC(6) = 7; SUCC(–6) = –7
ODD(i)	Перевірка на непарність	ODD(11) = TRUE; ODD(20) = FALSE

Таблиця 9. Операції для величин дійсного типу

Назва	Призначення	Приклади
*	множення	1.3 * 2.4 = 3.12
/	ділення (результат – не ціле!)	1.3 / 2.4 = 0.5416
+	додавання	1.3 + 2.4 = 3.7
–	віднімання	1.3 – 2.4 = –1.1
=; <>; <; >; <=; >=	операції відношення	1.3 > 2.4 → FALSE

Таблиця 10. Функції для величин дійсного типу

Назва	Призначення	Приклади
ABS(r)	Знаходження абсолютного значення величини	ABS(–48) = 48; ABS(48) = 48
SQR(r)	Піднесення до квадрату	SQR(4) = 16; SQR(11) = 121
SQRT(r)	Знаходження квадратного кореня (r0)	SQRT(2.48)=6.1504; SQRT(4)=2.0
TRUNC(r)	Знаходження цілої частини числа	TRUNC(8.53) = 8; TRUNC(–9.2) = –9
ROUND(r)	Знаходження найближчого цілого (з округленням)	ROUND(8.53) = 9; ROUND(8.3) = 8; ROUND((–8.53) = –9
FRAC(r)	Знаходження дробової частини	FRAC(2.4)=0.4; FRAC(–2.4)=–0.4
SIN(r)	Знаходження значення синуса	SIN(2.48)=0.61437
COS(r)	Знаходження значення косинуса	COS(2.48)=–0.789
ARCTAN(r)	Знаходження значення арктангенса	ARCTAN(2.48)=1.1857
LN(r)	Знаходження значення значення натурального логарифма (r>0)	LN(2.48)=0.090825
EXP(r)	Знаходження значення експоненціальної функції	EXP(2.48)=11.9412
INT(r)	Знаходження найближчого цілого числа (без округлення)	INT(2.98)=2.0; INT(–1.3)=–1.0

Таблиця 11. Операції для величин логічного типу

Назва	Призначення	Приклади
NOT	заперечення	NOT TRUE = FALSE; NOT FALSE = TRUE
AND	кон’юнкція (логічне множення)	TRUE AND FALSE = FALSE; TRUE AND TRUE = TRUE
OR	диз’юнкція (логічне	FALSE OR FALSE = FALSE;

	додавання)	TRUE OR FALSE = TRUE

Таблиця 11. Операції для величин логічного типу

Назва	Призначення	Приклади
XOR	виключаюче "або"	TRUE XOR TRUE = FALSE; TRUE XOR FALSE = TRUE
=; <>; <; >; <=; >=	операції відношення	FALSE<TRUE -> TRUE

Таблиця 12. Функції для величин логічного типу

Назва	Призначення	Приклади
PRED(L)	знаходження попереднього значення для L	PRED(TRUE) = FALSE; PRED(FALSE) = TRUE
SUCC(L)	знаходження значення, наступного за L	SUCC(TRUE) = FALSE; SUCC(FALSE) = TRUE
ORD(L)	знаходження порядкового номера значення L	ORD(FALSE)=0; ORD(TRUE)=1

Таблиця 13. Операції для величин символьного типу

Назва	Призначення	Приклади
=; <>; <; >; <=; >=	операції відношення	'A' < 'B' -> TRUE

Таблиця 14. Функції для величин символьного типу

Назва	Призначення	Приклади
ORD(c)	порядковий номер символа c в заданому символьному наборі	ORD('A') = 65; ORD(#4) = 4
CHR(i)	символ, що відповідає числу i згідно з кодами ASCII	CHR(65)='A'
PRED(c)	символ, що передує символу c	PRED('B') = 'A'
SUCC(c)	символ, що є наступним за символом c	SUCC('A')='B'
UPCASE(c)	символ верхнього регістру, відповідний ‘a’..’z’	UPCASE(‘a’)=’A’

Питання для самоконтролю.

1. *Які типи даних називаються стандартними?*
2. *Які типи даних мови ТР призначені для подання цілих чисел?*
3. *Які операції і функції ТР застосовні до цілих чисел?*
4. *Які типи даних мови ТР призначені для подання дійсних чисел?*
5. *Які операції і функції ТР застосовні до дійсних чисел?*
6. *Які операції і функції ТР застосовні до даних символьного типу?*
7. *Які операції і функції ТР застосовні до даних логічного типу?*

Класифікація типів даних. Типи даних, що визначаються програмістом

Всі типи даних, що використовуються у ТР, поділяються на прості і складені.

Простий тип даних визначає тип одного окремого значення.

Складений тип даних визначає структуру з простих типів.

До простих типів даних відносять цілочисельні, дійсні типи, логічний, символьний, перелічувальний тип та діапазон.

До складених типів відносяться масив, запис, множина, файл, посилання, об'єкт.

Прості типи даних, в свою чергу, поділяються на дискретні та неперервні.

Дискретний тип даних – такий тип даних, кількість можливих значень якого обмежена (скінченна). Наприклад, тип Byte має 256 значень.

Неперервний тип даних – тип даних, кількість можливих значень якого нескінченна.

Неперервними у Турбо Паскалі є лише дійсні типи. Решта простих типів є дискретними.

Як уже зазначалося, у ТР передбачена можливість створювати нові типи даних на основі існуючих. Нові типи описуються в розділі визначення типів програми:

Type ім'я_типу = опис типу;

Ім'я_типу – ідентифікатор, опис типу надається у відповідності з синтаксисом мови.

До простих типів, що визначаються програмістом, відносяться перелічувальний тип та діапазон.

Перелічувальний тип даних – простий дискретний тип, який визначає скінченний набір констант. Змінні цього типу можуть набувати значення лише з визначеного набору.

Формат опису перелічувального типу:

Type ім'я_типу = (константа1[, константа2, …константаN]);

Імена констант є ідентифікаторами.

Приклад.

```
Type Rainbow = (Red, Orange, Yellow, Green, LightBlue, Blue, Magenta);
Var A: Rainbow;
Begin
  A:= Orange;
End.
```

Будь–який перелічувальний тип має внутрішню нумерацію, перший елемент має номер 0, другий – 1 і т.д. Отже, вважається, що кожна константа більша за попередню і менша наступної.

До змінних перелічувального типу можна застосовувати операції порівняння =; <>; <; >; <=; >=, функції PRED, SUCC, ORD.

Приклади. PRED(Green) = Yellow; SUCC(Blue) = Magenta; ORD(Orange)=1; ORD(Red)=0;

Існує можливість перетворити ціле число у константу перелічувального типу. Для цього використовується функція, ім'я якої співпадає з назвою перелічувального типу.

Приклад. A := Rainbow(3); {A=Green}

До даних перелічувального типу не застосовні стандартні операції введення–виведення.

Діапазон – простий дискретний тип, який визначає підмножину значень певного базового типу. Базовим може бути будь–який простий дискретний тип, в тому числі і перелічувальний, введений програмістом.

Формат опису типу діапазон:

Type ім'я_типу = мінімальне_значення .. максимальне_значення;

Приклад.

```
Type Century20=1901..2000; {для цього типу базовим є тип Integer}
     HotColors=Red..Yellow; {для цього типу базовим є тип Rainbow}
Var  C: HotColors;
…
C:=Red; {правильне присвоювання}
C:= Green; {помилка}
```

До даних типу діапазон застосовні всі дії, які застосовні до даних базового типу.

Питання для самоконтролю.

1. *Наведіть загальну класифікацію типів даних мови ТР.*
2. *Які характеристики мають прості типи даних?*
3. *Чим розрізняються дискретні і неперервні типи даних?*
4. *Які характеристики мають складені типи даних?*
5. *Що таке перелічувальний тип даних? Як він описується в програмі на ТР?*
6. *Які операції і функції застосовні до даних перелічувального типу? Наведіть приклади.*
7. *Що таке тип даних діапазон? Як він описується в програмі на ТР?*
8. *Які операції і функції застосовні до даних типу діапазон? Наведіть приклади.*

Основні оператори мови Паскаль. Умовний оператор та оператор варіанту

При необхідності змінити порядок виконання операторів програми в залежності від виконання певних умов, тобто здійснити розгалуження процесу обчислень, використовують умовний оператор і оператор варіанту. При програмуванні розгалуження на дві гілки доцільно використовувати умовний оператор (його також називають оператором розгалуження), на більшу кількість гілок – оператор варіанту.

Умовний оператор застосовується в двох формах – повній і скороченій.

Формат скороченої форми оператора:

if логічний_вираз then оператор1;

if (якщо), then(то) – службові слова, оператор1 – довільний оператор мови ТР.

Виконання оператора: обчислюється значення логічного виразу. Якщо воно дорівнює TRUE, виконується оператор1, якщо FALSE – відбувається перехід до виконання наступного оператора програми. Оператор1 може бути як простим, так і складеним. В останньому випадку він береться в операторні дужки.

Формат повної форми оператора:

if логічний_вираз then оператор1 else оператор2;

Виконання оператора: обчислюється значення логічного виразу. Якщо воно дорівнює TRUE, виконується оператор1, якщо FALSE – виконується оператор2, після чого відбувається перехід до виконання наступного оператора програми. Оператор1 і оператор2 можуть бути як простими, так і складеними. Перед службовим словом else крапка з комою не ставиться.

Приклад. Розв’язування квадратного рівняння.

```
Var a,b,c,D,x1,x2:Real;
Begin
  Writeln('Введіть коеф–ти рівняння a,b,c:');
   Readln(a,b,c);
   D:=b*b–4*a*c;
   if D>=0 then
     Begin
        x1:=(–b–sqrt(D))/(2*a);
        x2:=(–b+sqrt(D))/(2*a);
         Writeln('x1=',x1:4:2,'x2=',x2:4:2)
     End
  else
    Writeln('Дійсних коренів немає');
  Readln;
End.
```

Оператор варіанту дозволяє виконувати ті чи інші дії в залежності від значення виразу дискретного типу.

Формат оператора:

```
case вираз_дискретного_типу of
   список1: оператор1;
   список2: оператор2;
   …
   списокN: операторN
[else
    операторN+1]
End;
```

Case(варіант), of(з) – службові слова, список1 – перелік значень виразу_дискретного_типу, розділених комами, оператор1 – довільний оператор мови ТР. Вираз дискретного типу називається також селектором.

Виконання оператора: обчислюється значення виразу, якщо воно входить до списку1, виконується оператор1 і відбувається перехід до виконання наступного оператора програми, якщо до списку2 – оператор2 і т.д. Якщо значення виразу не входить в жоден список, то при наявності частини else виконується операторN+1, а при її відсутності – виконання оператора варіанту завершується.

Приклад. Визначення природи введеного символу: цифра, голосна або приголосна літера.

```
Var c: Char;
Begin
  Writeln('Введіть символ:');
   Readln(c);
    case c of
      '0'..'9' : Writeln('Цифра');
      'a','e','i','o','u','y' : Writeln('Голосна')'
      'b'..'d','f'..'h','j'..'n','p'..'t','v'..'x','z' : Writeln('Приголосна')
     else Writeln('Інший символ')
     End;
End.
```

Питання для самоконтролю.

1. *Які оператори мови ТР призначені для організації розгалужень?*
2. *Вкажіть формат повної та скороченої форм умовного оператора. Як виконується умовний оператор? Наведіть приклади.*
3. *Вкажіть формат оператора варіанту. Як виконується цей оператор? Наведіть приклади.*

Основні оператори мови Паскаль. Циклічні оператори

Цикл у програмуванні – це повторюване виконання одних і тих самих простих або складених операторів. У Паскалі реалізовані три способи організації циклічних обчислень.

Оператор циклу з передумовою

Формат оператора:

while логічний_вираз do оператор;

while (поки), do (виконувати) – службові слова, оператор – довільний оператор мови ТР, який називається тілом циклу. Складений оператор береться в операторні дужки.

Виконання оператора: обчислюється значення логічного виразу, якщо воно дорівнює TRUE, виконується оператор і відбувається повернення до умови. Процес продовжується доти, поки значення логічного виразу не стане рівним FALSE. Після цього виконання циклу закінчується.

Якщо при першому обчисленні значення логічного виразу його значення дорівнює FALSE, тіло циклу не виконується жодного разу.

Зауваження. 1. Щоб цикл коли-небудь закінчив роботу, в його тілі повинен бути оператор, що впливає на значення логічного виразу. Краще, якщо цей оператор останній у тілі циклу.

2. Логічний вираз має бути коректним, тобто його значення повинно бути визначеним ще до першого виконання тіла циклу.

Приклад. Обчислити значення $\sum_{n=1}^{\infty}\frac{1}{n^2}$ з точністю до 10^{-6}.

```
Const eps=1E-6;
Var n: Integer;              {сумування проводиться до тих пір,}
    x,S: Real;         {поки черговий доданок не стане менше}
Begin                  {заданої точності обчислень}
  n:=1;  S:=0;  x:=1;
  while abs(x) >= eps do
     Begin
        S:=S+x;
        n:=n+1;
        x:=1/sqr(n)
     End;
  Writeln('S=', S)
End.
```

Оператор циклу з післяумовою

Формат оператора:

repeat оператор until логічний_вираз;

repeat (повторювати), until (до) - службові слова; оператор - довільний оператор мови ТР, в тому числі порожній. Складений оператор в операторні дужки не береться.
Виконання оператора: виконується тіло циклу, потім обчислюється значення логічного виразу. Якщо воно дорівнює FALSE, відбувається повернення до виконання тіла циклу, якщо TRUE – виконання циклу закінчується.

На відміну від циклу з передумовою, тіло циклу з післяумовою завжди виконується принаймні один раз. Це треба враховувати при виборі оператора для виконання циклічних обчислень.

Для циклу з післяумовою також справедливе зауваження 1.

Приклад. Фрагмент програми, в якому від користувача вимагається ввести ціле додатне число.

```
repeat
  Write('Введіть додатне число: ');
   Readln(n)
until n>0;
```

Приклад. Перепишемо попередню програму підрахунку суми, використовуючи оператор циклу з післяумовою:

```
Const eps = 1E-6;
Var n: Integer;
     x, S : Real;
Begin
   n:=1; S:=0; x:=1;
repeat
    S:=S+x;
   n:=n+1;
   x:=1/sqr(n)
until abs(x) < eps;
Writeln('S=',S)
End.
```

Цей приклад показує, що оператор циклу з передумовою вигляду while B do S; може бути переписаний як оператор циклу з післяумовою repeat S until not B;

Оператор циклу з параметром

За умови, коли відома кількість повторень тіла циклу, зручно використовувати оператор циклу з параметром.

Формат оператора:

for параметр_циклу:= вираз1 to вираз2 do оператор;

for (для), to (до), do (виконувати) – службові слова, параметр_циклу (або лічильник циклу) - ім'я змінної дискретного типу, вираз1 і вираз2 – вирази, тип яких співпадає з типом параметра циклу. Оператор – довільний оператор мови ТР. Складений оператор береться в операторні дужки.

Виконання оператора: обчислюється значення виразу1, це значення присвоюється параметру циклу. Потім обчислюється значення виразу2. Якщо значення параметра не перевищує значення виразу2, виконується тіло циклу, до параметра застосовується функція SUCC, отримане значення знову порівнюється зі значенням виразу2 і т.д. Після того як значення параметра перевищить значення виразу2, виконання оператора циклу закінчується.

Кількість повторень тіла циклу =
значення_виразу2 - значення_виразу1 + 1

Якщо значення виразу1 більше значення виразу2, тіло циклу не виконується жодного разу.

Різниця між двома сусідніми значеннями параметра циклу називається **<u>кроком циклу.</u>**

Крок циклу з параметром завжди постійний і залежить від типу параметра.

Розглянутий формат оператора циклу використовується, коли параметр циклу зростає. Якщо параметр циклу спадає, використовується модифікований формат:

for параметр_циклу := вираз2 downto вираз1 do оператор;

На відміну від попереднього формату циклу, при виконанні оператора до параметра циклу застосовується функція PRED. Значення виразу2 повинно бути більше значення виразу1.

Приклад. Вивести у порядку спадання коди латинських літер.

```
Var ch :Char;
Begin
  for ch:=’z’ downto ‘a’ do
    Writeln(ch, ‘ - ‘, ord(ch))
End.
```

Узагальнимо отримані відомості про оператори циклу в TP (таблиця 15)

Таблиця 15. Циклічні оператори TP

Оператор	**Умови застосування**	**Особливості**
WHILE …DO	**невідома кількість повторень (0)**	**тіло циклу може не виконатись жодного разу**
REPEAT…UNTIL	**невідома кількість повторень (0)**	**тіло циклу виконується хоча б один раз**
FOR…TO…DO	**відома кількість повторень**	**тіло циклу може не виконатись жодного разу; крок циклу фіксований**

Приклад. Обчислення факторіалу заданого натурального числа з використанням різних операторів циклу.

```
Var N, Fact: Integer; i: Byte;
Begin
  Write( ‘N>0:’); Readln(n);
  { for}
  Fact:=1;
  for i:=1 to N  do
    Fact:=Fact*i;
  Writeln(N,’! = ‘, Fact);
  {while}
  i:=1; Fact:=1;
  while i<=N do
    Begin
      Fact:=Fact*i;
      i:=i+1
    End;
 Writeln(N,’! = ‘, Fact);
  {repeat}
  i:=1; Fact:=1;
  repeat
    Fact:=Fact*i;
    i:=i+1
  until i>N;
Writeln(N,’! = ‘, Fact);
End.
```

Приклад. Обчислити суму ряду $\sum_{i-1}^{n} \frac{i}{2i+1}$ для заданого n;

```
Var i,n: Integer;
     S:Real;
Begin
  Readnl(n);
   S:=0;
   for i:=1 to n do
     S:=S+i/(2*i+1);
   Writeln(‘S=’, S:6:2)
End.
```

Приклад. Обчислити суму ряду $\sum_{n=1}^{20} \frac{(-1)^n}{n(n+1)}$;

```
Var  S:Real;
      n, w: Integer;
Begin
```

```
  S:=0;
 w:=-1; {чисельник  першого доданку}
 for n:=1 to   20 do
   Begin
     S:=S+w/(n*(n+1));
      w:=-w {зміна знаку доданка}
   End;
Writeln('S=',S)
End.
```

Табулювання функцій

При проведенні обчислювальних експериментів часто виникає необхідність протабулювати певну функцію.

Табулювання функції f(x) на відрізку [a,b] з кроком h полягає у побудові таблиці значень функції f(x) в точках x=a, x=a+h, x+2h,…, x=b:

x	f(x)
a	f(a)
a+h	f(a+h)
…	…
b	f(b)

Функція f(x) має бути визначена на відрізку [a,b].

Відрізок [a,b] називається відрізком табулювання, h – кроком табулювання, точки a+hi (i=0,1,2,…) – вузлами табулювання, відрізки [a+hi, a+h(i+1)] – відрізками розбиття.

Може статися, що a-b не ділиться націло на h, тобто точка b не є вузлом табулювання, наприклад при a=1, b=5, h=1.5.

Якщо така ситуація небажана, задачу формулюють інакше:

Дано відрізок [a,b] (a<b), функція f(x), натуральне n (кількість відрізків розбиття).

Побудувати таблицю значень функції f(x): $y_i=f(x_i)$, де $x_i=a+(b-a)i/n$, i=0,1,…,n.

Приклад. Протабулювати функцію $y(x)=x^2+\sin(x)$ на відрізку [a,b], якщо задана кількість відрізків розбиття.

```
Var x,y,a,b,h : Real;
    n:Byte;
Begin
   Write('Введіть a,b(a<b)');
   Readln(a,b);
   Write('Введіть n>0');
   Readln(n);
   h:=(b-a)/n;
   x:=a;
   Writeln('  x     y(x)  ');
```

```
   while x<=b do
     Begin
       y:=sqr(x)+sin(x);
       Writeln(x:6:2, y:6:2);
       x:=x+h
     End
End.
```

Приклад. Протабулювати функцію y(x)=cos(x)-x на відрізку [a,b] з кроком h. Якщо на [a,b] функція набуде заданого значення K, вивести його на екран і припинити табулювання.

```
Var x,y,a,b,h,k: Real;
Begin
  Write(a,b (a<b)'); Readln(a,b);
  Write('h>0'); Readln(h);
  Write('k'); Readln(k);
  x:=a;
  repeat
    y:=cos(x)-x;
    Writeln(x:6:2, y:6:2);
    x:=x+h
 until (x>b) or (y=k);
End.
```

Приклад. Дано a,b (a<b), натуральне n, функція $y(x)=x^5$. Для $x_i=a+ih$ (i=0,1,…,n), h=(b-a)/n обчислити значення функції $y(x_i)$ (i=0,1,…,n).

```
Var i,n:Integer;
    a,b,x,y,h: Real;
Begin
   Readln(a,b,n);
   h:=(b-a)/n;
   for i:=0 to n do
     Begin
       x:=a+i*h;
       y:=exp(5*ln(x));
       Writeln(x:7:4, y:7:4)
    End;
End.
```

Вкладені цикли

В усіх операторах циклу мови ТР оператор, який є тілом циклу, може сам бути оператором циклу або містити у собі оператор циклу. Утворена конструкція називається вкладеним циклом.

Приклад. Обчислити суму ряду $\sum_{i=1}^{10}\sum_{j=1}^{5}\frac{1}{i+j^2}$.

```
Var  S:  Real;
      i,j: Integer;
Begin
  S:=0;
  for i:=1 to 10 do
    for j:=1 to 5 do
      S:=S1/(i+j*j);
  Writeln('S=',S);
End.
```

Питання для самоконтролю.

1. *Що називається циклом у програмуванні? Які оператори мови ТР призначені для організації циклів?*
2. *Вкажіть формат оператора циклу з передумовою. Як виконується цей оператор? Які особливості його виконання?*
3. *Вкажіть формат оператора циклу з післяумовою. Як виконується цей оператор? Які особливості його виконання?*
4. *Вкажіть формат оператора циклу з параметром. Які існують модифікації цього оператора? Як вони виконуються?*
5. *Що називається кроком циклу?*
6. *Порівняйте особливості виконання трьох операторів циклу мови ТР.*
7. *Що таке табулювання функції? Наведіть приклади.*
8. *Що називається вкладеним циклом? Наведіть приклади.*

Структури даних. Масиви

Особливістю мови Паскаль є вимога чіткого опису всіх використовуваних в програмі об'єктів. Так, блок описів програми відокремлений від виконуваного блоку, у блоці описів відокремлені розділи опису констант, типів, змінних тощо. Кожний елемент даних (константа чи змінна) повинен бути описаний перед використанням. Кожна підпрограма (процедура чи функція) має бути визначена перед її викликом.

Такий чіткий опис об'єктів полегшує програмісту написання програми, а транслятору – її перевірку і виконання.

Для зрозумілості програми не лише автору, а й іншим особам, Паскаль вимагає чіткої структуризації програми - щоб та чи інша інформація знаходилася у визначеному для неї місці.

Особливу роль відіграє структуризація даних. Програма задає правила обробки даних. Отже, поки не визначені самі дані, неможливо вдало розробити правила їх обробки.

Для спрощення написання і виконання програми окремі дані часто буває зручно об'єднувати в певні структури. Від того, наскільки вдало будуть вибрані ці структури, суттєво залежить ефективність програми.

В Паскалі та чи інша структура даних задається за допомогою певного типу даних.

Прості типи даних задають тривіальні структури даних – окреме значення.

Більш складні структури даних задаються за допомогою складених (структурованих) типів.

Значення складеного типу в загальному випадку є нетривіальною структурою, тобто містить більш ніж одну компоненту. При цьому кожна компонента структури може бути значенням як простого, так і складеного типу.

Найбільш уживаним складеним типом даних є регулярний тип, або масив.

Масив – це скінченний упорядкований набір однотипних значень (компонентів або елементів масиву).

Тип елементів масиву називається **базовим типом масиву**.

Кожен елемент масиву має принаймні один індекс – адресу, за якою можна звернутися до цього елемента.

Формат опису масиву у розділі опису типів:

Type ім'я_масиву = Array[список типів індексів] of базовий_тип;

Array (масив), of(з) – службові слова, ім'я_масиву – ідентифікатор, тип індексів – діапазон (підмножина значень простого дискретного типу).

Приклад.

```
Type  Dim3 = Array[1..3] of Real;
Var  W,V: Dim3;
```

Змінна V є структурою з трьох дійсних чисел: V[1], V[2], V[3]. Числа 1,2,3 – індекси.

Якщо базовим типом масиву є інший масив, утворюється структура, яка називається багатовимірним масивом.

Приклад.

```
Type Vector = Array [1..4] of Integer;
     Matrix = Array [1..4] of Vector;
Var Matr: Matrix;
```

Таку ж структуру можна отримати, використовуючи іншу форму запису:

```
Type Matrix = Array [1..4,1..4] of Integer;
Var Matr: Matrix;
```

або

```
Var Matr: Array [1..4,1..4] of Integer;
```

Останній запис показує, що тип можна задавати безпосередньо при описі змінних.

Елементи масиву Matr: Matr[1,1], Matr[2,3], Matr[4,4] (всього 16 елементів)

Для звернення до окремого елемента масиву треба вказати ім'я масиву і в квадратних дужках індекси елемента. Елемент одновимірного масиву має один індекс, багатовимірного – стільки індексів, яка вимірність масиву.

Двовимірні масиви широко використовуються для подання матриць.

В загальному випадку матриця записується так:

$$A_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & & a_{mn} \end{bmatrix}$$

У ТР така матриця є масивом типу Array[1..m,1..n] of Real;

Якщо кількість рядків матриці дорівнює кількості стовпців, матриця називається квадратною. Головна діагональ квадратної матриці проходить з верхнього лівого кута до правого нижнього. Побічна діагональ – з нижнього лівого кута до правого верхнього.

Над змінними типу масив в цілому можна виконувати операції присвоювання (якщо змінні однотипні) і перевірки на рівність: W:= V; V=W (?)

Решта операцій над масивами виконується поелементно.

Приклад. Надання значень елементам масиву:

а) V[1]:=10; V[2]:=20; V[3]:=45.7;

б) for i:=1 to 3 do V[i]:=0;

в) for i:=1 to 3 do Readln(V[i]);

Приклад. Описи масивів.

```
Type
Month = (January, February, March, April, May);
DaysInMonth = Array[Month] of Byte; {кількість днів у місяці}
SpringType = Array[March..May] of Byte;
Var
        Days: DaysInMonth;
      Spring: SpringType;
      Alpha: Array[‘A’..’Z’] of Char;
      Vector: Array[1..10] of Real;
      Matrix: Array[1..5,1..5] of Real;
      Vector1: Array[-10..10] of Byte;
Begin
  Days[January]:=31;
  Spring[April]:=30;
  Alpha[‘A’]:=’1’;
  Vector[10]:=0.5;
  Matrix[1,1]:=4;
  Vector1[-5]:=0
End.
```

Загальна схема розв’язування задач на опрацювання масивів:

1) введення масиву (надання елементам масиву початкових значень);
2) опрацювання елементів масиву;
3) виведення результатів.

Оскільки масиви містять фіксовану кількість елементів з індексами дискретних типів, то для їх опрацювання зручно використовувати цикл з параметром.

Приклад. Дано масив з 10 дійсних чисел. Підрахувати суму, добуток і середнє арифметичне елементів масиву.

```
Const maxV=10;
Type mas = Array[1..maxV] of Real;
Var  v: mas;          {масив}
     D,S,A: Real;   {добуток, сума, середнє  арифметичне}
     i:Byte;          {лічильник}
Begin
  {введення елементів масиву}
 Writeln(‘Введіть’, maxV, ‘чисел:’);
 for i:=1 to maxV do
   Begin
     Write(‘V[‘,i,’]=’);
      Readln(v[i])
   End;
  {опрацювання елементів масиву}
 D:=1; S:=0; A:=0;
```

```
  for i:=1 to maxV do
    Begin
      D:=D*v[i];
      S:=S+v[i]
    End;
  A:=S/maxV;
Writeln('Сума = ',S:4:1, 'добуток = ',D:4:1, 'сер. арифм. = ', A:4:1);
End.
```

Приклад. Дано масиви A, B (кожний з 10 цілих чисел). Побудувати масив C за правилом:

$c_i = a_i^2 + b_i^2$, i=1,..,10.

```
Type mas = Array[1..10] of Integer;
Var A,B,C: mas;
      i: Byte;
{введення масивів A і B}
Begin
   for i:=1 to 10  do
      Begin
         Write('A[',i,']=');
          Readln(A[i])
      End;
for i:=1 to 10  do
      Begin
         Write('B[',i,']=');
          Readln(B[i])
      End;
{формування масиву C}
for i:=1 to 10  do
   C[i]:=sqr(A[i])+sqr(B[i]);
 {виведення масиву C}
Writeln('Масив C:');
for i:=1 to 10  do
   Write(C[i]:3);
end.
```

Питання для самоконтролю.

1. *В чому полягає структуризація програм і даних у ТР?*
2. *Що таке тривіальні і нетривіальні структури даних?*
3. *Що називається масивом?*
4. *Що таке базовий тип масиву, індекс масиву?*
5. *Як описуються масиви у програмі на ТР?*
6. *Що таке вимірність масиву? Як описуються багатовимірні масиви?*
7. *Які операції можна виконувати над масивами в цілому? Над елементами масивів? Наведіть приклади.*

Масиви. Приклади розв'язування задач

Приклад. Дано масив з 15 дійсних чисел;
а) знайти максимальне значення елемента масиву;
б) підрахувати кількість елементів масиву з таким значенням.

```
Type mas=Array[1..15] of Real;
Var A: mas;  N,i: Byte; Max: Real;
Begin
{ведення масиву A}
Begin
  for i:=1 to 15  do
    Begin
      Write('A[',i,']=');
       Readln(A[i])
    End;
{знаходження максимуму}
Max:=A[1];
for i:=2 to  15  do
  if A[i]>Max then Max:=A[i];
{знаходження кількості  максимальних елементів}
N:=0;
for  i:=1 to 15 do
  if  A[i]=Max  then N:=N+1;
Writeln('макс. ел-т ', Max:4:1, 'зустрічається у масиві ', N ' разів');
End.
```

Приклад. Упорядкувати за зростанням масив з n дійсних чисел методом "бульбашки".

```
 Const n=10;
Type mas=Array[1..n] of Real;
Var a: mas; c:Real; i,j:Byte;
Begin
{введення масиву}
for i:=1 to n do
  Begin
    Write('a[',i,']=');
    Readln(a[i,j])
End;
{упорядкування масиву}
for i:=1 to n–1 do   {фіксація елементів з 1–ї по n–1 позицію}
  for j:=i+1 to n do {перегляд елементів, що стоять праворуч}
    if a[i]>a[j] then {від зафіксованого}
       Begin
         c:=a[i];      {обмін значеннями елементів }
         a[i]:=a[j];   {на i–й і  j–й позиціях}
```

```
          a[j]:=c
        End;
      {виведення упорядкованого масиву}
       Writeln('упорядкований масив:');
       for i:=1 to n do

    Write(a[i]:3)
End.
```

Приклад. Дано квадратну матрицю розмірності 4. Обчислити слід матриці – суму елементів її головної діагоналі.

```
Const n=4;
Type mas=Array[1..n,1..n] of Real;
Var a:mas; i,j:Byte; trace:Real;
Begin
{введення матриці}
  Writeln('Введіть елементи матриці 4x4 порядково,');
  Writeln('в кінці кожного рядка – Enter');
  for i:=1 to n do     {цикл по рядках}
    Begin
      for j:=1 to n do          {цикл по стовпцях}
        Read(a[i,j]);  {введення елемента рядка}
      Writeln;          {перехід на новий рядок}
    End;
{знаходження сліда матриці}
trace:=0;
  for i:=1 to n do
    trace:=trace + a[i,j];
Writeln('Trace(A)=', trace:4:1)
End.
```

Приклад. Дано цілочисельну матрицю розміром m×n. Знайти мінімальний елемент матриці та його індекси.

```
Const m=4; n=3;
Type mas=Array[1..m,1..n] of Integer;
Var a:mas; min:Integer; i,j,imin,jmin:Byte;
Begin
  for i:=1 to m do
     Begin
        for j:=1 to n do
          Read(a[i,j]);
       Writeln;
     End;
min:=a[1,1]; imin:=1; jmin:=1;
for i:=1 to m do
```

```
    for j:=1 to n do
      if a[i,j]<=min then
         Begin
            min:=a[i,j];
            imin:=i;
           jmin:=j
         End;
  Writeln(min елемент матриці = ',min,'[',imin,',',jmin,']');
End.
```

Приклад. Сформувати і надрукувати одиничну матрицю порядку n (n<6).

```
Type mas= Array[1..n,1..n] of Byte;
Var e:mas; n,i,j:Byte;
Begin
  Write('Введіть порядок матриці n (n<6): ');
  Readln(n);
  {формування матриці E}
   for i:=1 to n do
     for j:=1 to n do
       if i=j then e[i,j]:=1
        else e[i,j]:=0;
   {виведення матриці E}
  Writeln('Матриця E:);
  for i:=1 to n do
    Begin
      for j:=1 to n do
        Write(e[i,j]:2);
      Writeln;
    End;
End.
```

Приклад. Дано дійсну матрицю розміром m×n. Сформувати масиви з:
а) максимальних елементів рядків матриці;
б) сум елементів стовпців матриці;
в) середніх арифметичних рядків матриці.

```
Const m=5; n=7;
Type mas=Array[1..m,1..n] of Real;
vect1=Array[1..m] of Real;       {для а) і в)}
vect2=Array[1..n] of Real;       {для б)}
Var a:mas; b1,b3:vect1; b2:vect2; i,j:Byte;
Begin
{введення матриці A}
for i:=1 to m do
    Begin
       for j:=1 to n do
```

```
          Read(a[i,j]);
        Writeln;
     End;
{а) побудова масиву з макс. ел–тів рядків матриці}
for i:=1 to m do {цикл по рядках}
  Begin
    b1[i]:=a[i,1];
    for j:=2 to n do
      if a[i,j]>b1[i] then
         b1[i]:=a[i,j];
    End;
{б) побудова масиву з сум ел–тів стовпців матриці}
for j:=1 to n do {цикл по стовпцях}
  Begin
    b2[j]:=0;
    for i:=1 to m do
      b2[j]:=b2[j]+a[i,j];
    End;
{в) побудова масиву з середніх арифметичних рядків м–ці}
for i:=1 to m do
  Begin
    b3[i]:=0;
    for j:=1 to n do
      b3[i]:=b3[i]+a[i,j];
    b3[i]:=b3[i]/n;
    End;
{виведення результатів}
Writeln('Макс. ел–ти рядків м–ці:');
for i:=1 to m do Write(b1[i]:3); Writeln;
Writeln('Суми ел–тів стовпців м–ці:');
for i:=1 to n do Write(b2[i]:3); Writeln;
Writeln('Середні арифм. рядків м–ці:');
for i:=1 to m do Write(b3[i]:3); Writeln;
End.
```

Процедури і функції у мові Паскаль

У практиці програмування часто виникає ситуація, коли одну й ту саму групу операторів, які реалізують певну частину загальної задачі, треба повторити без змін в різних місцях програми. Для вирішення цієї проблеми була запропонована концепція підпрограми, вперше описана М.Уілксом у 1957 р. Вона отримала широке розповсюдження практично у всіх мовах програмування.

Підпрограмою називається поіменована логічно завершена група операторів мови, яку можна викликати для виконання за іменем довільну кількість разів у різних місцях програми.

Програма, яка використовує підпрограми, по відношенню до них називається **головною** або **основною**.

У мові Паскаль для організації підпрограм використовуються процедури і функції.

Всі процедури і функції мови Паскаль поділяються на дві групи: вбудовані і визначені користувачем.

Вбудовані (стандартні) процедури і функції є частиною мови і можуть викликатися за іменем без попереднього опису (наприклад, Read, Write, Sin, Cos, Pred, Succ тощо).

Процедури і функції користувача обов'язково описуються в програмі у розділі визначення процедур і функцій.

Використання процедур і функцій забезпечує структурованість програми, полегшує її написання і читання.

Процедури користувача

Процедура – це іменована група операторів, яка реалізує певну частину загальної задачі і викликається за іменем з будь–якої позиції у головній програмі.

Формат опису процедури:

```
Procedure ім'я_процедури [(список формальних параметрів)]; – заголовок процедури
розділи описів;   ⎤
Begin             ⎥
  оператори       ⎥  – тіло процедури
End;              ⎦
```

Procedure (процедура) – службове слово, ім'я_процедури – ідентифікатор,
список формальних параметрів = форм_пар1: Тип1, форм_пар2: Тип2, …
розділ описів – опис локальних об'єктів процедури (за синтаксисом співпадає з розділом описів програми на ТР).

Приклад. Заголовки процедур:

```
Procedure  Sort(A:Integer; B:Real);  Procedure Gauss;
Procedure Kvadr(Alpha, Beta: Integer; Gamma: Boolean);
```

Тіло процедури за структурою аналогічне програмі на ТР, ім'я процедури – ідентифікатор, унікальний у межах програми.

Для звернення до процедури в головній програмі використовується оператор виклику процедури.

Формат оператора виклику процедури:

Ім'я_процедури[(список фактичних параметрів)];

список фактичних параметрів = факт_пар1, факт_пар2, …

Приклад. Оператори виклику процедур:

Sort(a1,b1); Gauss; Kvadr(14, 25, TRUE);

Параметри використовуються для обміну інформацією між процедурою і основною програмою.

Параметри, які використовуються при описі процедури, називаються **формальними**.

Параметри, які використовуються при виклику процедури, називаються **фактичними**.

Кількість, тип і порядок слідування фактичних параметрів повинні бути такими, як у формальних параметрів. Параметри можуть мати будь-який тип, але визначений заздалегідь. Процедура може взагалі не мати параметрів.

При введенні у програму підпрограм виникає поділ даних і їх типів на глобальні і локальні.

Глобальні змінні (константи, типи) – ті, які описані в головній програмі. Ними можна користуватися і в головній програмі, і в підпрограмі.

Локальні змінні (константи, типи) – ті, які описані всередині підпрограми: або у списку параметрів (тільки змінні), або у розділах Const, Type, Var тіла підпрограми. За межами підпрограми локальні об'єкти недоступні.

У мові Паскаль є два способи передавання параметрів: за значенням і за посиланням.

Параметри, які передаються за значенням, називаються **параметрами–значеннями**, параметри, які передаються за посиланням, називаються **параметрами–змінними**.

Параметри-значення передаються з головної програми в процедуру, але не повертаються з неї. Фактичний параметр, відповідний формальному параметру-значенню, може бути будь-яким виразом того ж типу. Характерна риса параметрів-значень – зміна формальних параметрів не викликає зміни фактичних параметрів. Тому параметри-значення використовуються в процедурах для подання їх аргументів (вхідних даних).

Приклад. Procedure P1(A,B,C: Integer; D: Real);

Параметри-змінні і передаються в процедуру, і повертаються з неї в основну програму. Фактичний параметр, відповідний параметру-змінній, може бути тільки іменем змінної. Характерна риса параметрів-змінних – будь-яка зміна формального параметра означає зміну фактичного параметра. Параметри-змінні використовуються в процедурах для подання їх результатів, для того щоб результати могли бути передані і використані в основній програмі. У

списку формальних параметрів перед іменами параметрів-змінних ставиться службове слово Var.

Правильно написана підпрограма спілкується з основною програмою тільки через параметри.

Приклад. Procedure MN(Var R,T:Real);
Procedure NSD(a,b:Integer; Var k: Integer);
Procedure A(a,x:Byte; Var M:Integer; Var p,k: Real);

Приклад. Знайти площу трикутника зі сторонами a,b,c за формулою Герона, використавши процедуру.

```
Var a,b,c,S:Real; {глобальні змінні}
Procedure Geron(x,y,z: Real; Var pl:Real);
{x,y,z – параметри-значення, pl – параметр-змінна}
Var p:Real; {локальна змінна }
Begin
  p:=(x+y+z)/2;
  pl:=sqrt(p*(p–x)*(p–y)*(p–z));
End;
{основна програма}
Begin
 Writeln('Введіть довжини сторін трикутника: ');
 Readln(a,b,c);
 Geron(a,b,c,S);
 Writeln('Площа трикутника = ', S:4:1);
End.
```

Приклад. Скласти процедуру обміну значеннями двох дійсних змінних

```
Procedure swap(Var a,b:Real);
Var  tmp: Real;
Begin
  tmp:=a;
  a:=b;
  b:=tmp
End;
```

Процедури і функції у мові Паскаль. Функції користувача

У ТР є досить широкий вибір вбудованих функцій, проте користувач має можливість створювати необхідні йому функції і використовувати їх у своїх програмах.

Функція – це іменована група операторів, яка реалізує певну частину загальної задачі і має своїм результатом одне значення простого типу. Ім’я функції використовується у виразах як операнд.

Формат опису функції:

```
Function Ім'я_функції [(список форм. пар–рів)]: тип_результату;  – заголовок функції
розділи описів;    ⎤
Begin              ⎥ – тіло функції
  оператори        ⎥
End;               ⎦
```

Function (функція) – службове слово, тип_результату – ім'я простого типу.

Тіло функції за структурою аналогічне тілу процедури і програми на ТР. Ім'я_функції – ідентифікатор, унікальний у межах програми.

В розділі операторів повинен знаходитись принаймні один оператор присвоювання, в лівій частині якого стоїть ім'я функції, а в правій – певний вираз. Значення цього виразу функція повертає в основну програму. Якщо таких присвоювань декілька, результатом функції є значення останнього оператора присвоювання.

Для звернення до функції у головній програмі використовується ім'я функції з указаним в дужках списком фактичних параметрів:

Ім'я_функції [(список факт. пар–рів)];

Співвідношення між формальними і фактичними параметрами у процедур і функцій аналогічне. У заголовку функції можуть використовуватись як параметри-значення, так і параметри-змінні.

Приклад. Обчислити значення виразу

$f(x)=x^2+x+3+e^{x^{2+x+3}}+\sin^2(x^2+x+3)$ x=1,2,…,10.

```
Var x: Byte; f:Real;
Function y(x:Real):Real;
Begin
  y:=x*x+x+3;
End;
Begin
 for x:=1 to 10 do
   Begin
     f:=y(x)+exp(y(x))+sqr(sin(y(x));
     Writeln(f)
   End;
End.
```

Приклад. Знайти довжину вектора, заданого своїми координатами у n–вимірному просторі.

```
Const n=10;
Type vector=Array[1..n] of Real;
Function VLength(v:vector):Real;
Var i:Integer; s:Real;
Begin
 s:=0;
 for i:=1 to n do
```

```
    s:=s+sqr(v[i]);
    VLength:=sqrt(s);
End;
Var a:vector;
Begin
   for i:=1 to n do Readln(a[i]);
   Writeln('Довжина вектора a=',VLength(a):5:2);
End.
```

Відмінності процедур і функцій

1. Опис. В заголовку функції крім списку параметрів вказується тип результату. В тілі функції повинен бути оператор, який присвоює імені функції певне значення.
2. Результати роботи. Результатом роботи процедури може бути довільна кількість значень довільних типів. Результатом роботи функції є одне значення простого типу, вказаного в заголовку функції.
3. Використання. Процедура викликається у головній програмі за допомогою окремого оператора. Функція викликається не відокремлено, а як операнд виразу.

Правила локалізації

Програма на ТР має модульну структуру і може складатися з ряду вкладених один в одного блоків. Головна програма – це найбільший блок. Об'єкти, описані в цьому блоці, є глобальними і можуть використовуватись у всіх вкладених блоках. Описані в них об'єкти є локальними і недоступні у зовнішніх блоках.

Для правильного використання об'єктів слід дотримуватись таких правил щодо їх ідентифікаторів:

1. Кожний ідентифікатор має бути описаний перед його використанням.
2. Областю дії ідентифікатора є блок, в якому він описаний.
3. Всі ідентифікатори в блоці мають бути унікальними, тобто не повторюватися.
4. Один і той самий ідентифікатор може бути по–різному визначений в кожному окремому блоці.

Приклад. Дано a,b,c – довжини сторін деякого трикутника. Знайти медіани трикутника, сторонами якого є медіани даного трикутника.

Довжина медіани, проведеної до сторони a, дорівнює

$$m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$$

```
Var a,b,c:Real; {сторони даного трикутника}
m1,m2,m3:Real; {медіани даного трикутника}
m11,m22,m33:Real; {медіани трикутника, утвореного з медіан даного}
Function med(x,y,z:Real):Real;
Begin
```

```
  med:=0.5*sqrt(2*x*x+2*y*y–z*z)
End;
Begin
  Readln(a,b,c);
  m1:=med(b,c,a); {ma}
  m2:=med(a,c,b); {mb}
  m3:=med(a,b,c); {mc}
  m11:=med(m2,m3,m1);
  m22:=med(m1,m3,m2);
  m33:=med(m1,m2,m3);
  Writeln(m11,m22,m33)
End.
```

Рекурсія

Використання рекурсії є традиційною перевагою мови Паскаль. Під **рекурсією** розуміють виклик функції (процедури) з тіла цієї ж самої функції (процедури). Рекурсивність часто використовується в математиці. Так багато означень математичних формул є рекурсивними.

Приклад. Формула обчислення факторіала:

$n!=1$, якщо $n=0$;

$n!=n*(n–1)!$, якщо $n>0$.

Формула обчислення цілого степеня числа:

$x^n=1$, якщо $n=0$;

$x^n=x*x^{n–1}$, якщо $n>0$.

З прикладів видно, що для обчислення кожного наступного значення треба знати попереднє. В Паскалі рекурсія записується так само, як і у формулах. Розглянемо функції обчислення факторіала і цілого степеня числа.

```
Function Fact(n: Word):LongInt;
Begin
 if n=0 then Fact:=1
   else Fact:=n*Fact(n–1);
End;
Function IntPower(x: Real; n: Word):Real;
Begin
  if n=0 then IntPower:=1
    else IntPower:=x*IntPower(x,n–1);
End;
```

Якщо у функцію передаються $n>0$, відбувається наступне: запам’ятовуються відомі значення членів виразу гілки else (для факторіалу це n, для степеня – x), а для обчислення невідомих викликаються ті ж функції, але з "попередніми" аргументами. При цьому знову запам’ятовуються (але в іншому місці пам’яті) відомі значення членів і відбуваються виклики. Так відбувається доти, поки вираз не стане повністю визначеним (в розглянутих прикладах – це присвоювання гілки then), після чого алгоритм починає "розкручуватись" в інший бік, беручи з пам’яті "відкладені" значення. Оскільки при цьому на

кожному черговому кроку всі члени виразів вже будуть відомі, через n таких "зворотних" кроків буде отримано кінцевий результат.
Необхідністю для працездатності рекурсивних процедур є наявність умови закінчення рекурсивних викликів (наприклад, перевірка значення параметра, що змінюється). Дії, пов'язані з такою перевіркою, вже не можуть містити рекурсивних викликів. Якщо ця умова не буде виконуватись, то глибина рекурсії стане нескінченною, що призведе до аварійної зупинки програми.
Часто внесення рекурсивності в програми надає їм привабливості, але при цьому програми витрачають більше пам'яті. Це відбувається тому, що кожний "відкладений" виклик функції чи процедури – це черговий набір значень всіх локальних змінних цієї функції, розміщених у стеку. (У TP розмір стеку не може перевищувати 64К).
Незважаючи на наочність рекурсивних описів, в багатьох випадках ті ж задачі більш ефективно розв'язуються ітераційними методами, які не вимагають "зайвої" пам'яті при аналогічній швидкості обчислень. Наприклад, функція обчислення цілого степеня числа X може бути переписана так:

```
Function IntPower(x: Real; n: Word): Real;
Var i: Word; m: Real;
Begin
  m:=1;
  for i:=1 to n do m:=m*x;
  IntPower:=m
End;
```

Побічний ефект

Побічним ефектом називається виконання функцією дій, відмінних від обчислення її основного значення. Зокрема, це зміна значень глобальних змінних, виведення певної інформації, не пов'язаної з призначенням функції тощо.

Побічного ефекту слід уникати. Всередині функції необхідно використовувати тільки параметри та локальні змінні.

Приклад.

```
Var a,b: Real;
Function f(x:Real):Real;
Begin
   f:=2*x*x+0.5;
   a:=0; {побічний ефект – зміна значення глобальної змінної}
End;
Begin
   a:=3.14; b:=f(3);
   Writeln(a,b);
End.
```

В результаті виконання такої програми на екран буде виведено 0 і 18.5.

Основні оператори мови Паскаль. Оператори завершення роботи програми

ТР містить засоби безумовного виходу з програмних блоків (процедур, функцій, основного блоку програми). До таких операторів завершення відносяться виклики системних процедур Exit і Halt.

Виклик Exit завершує роботу свого програмного блоку. Якщо виконується Exit у процедурі, то її виконання завершиться і відбудеться перехід до виконання оператора, наступного за оператором процедури. При цьому процедура поверне значення, які встигли обчислитися на момент виконання Exit (якщо вона повинна була їх повернути). Сама програма не перерветься. Якщо ж Exit виконується в основному блоці програми, вихід з нього буде еквівалентним її нормальному завершенню.

Процедура Halt, або більш повно Halt(n), діє більш радикально. Незалежно від того, де вона знаходиться, її виконання завершує роботу програми з кодом завершення n, який потім може бути проаналізований. Значення n=0 відповідає нормальному коду завершення. Виклик процедури Halt без параметра еквівалентний виклику Halt(0).

Питання для самоконтролю.

1. *Що називається підпрограмою, основною програмою?*
2. *Чим відрізняються вбудовані процедури і функції від процедур і функцій користувача?*
3. *Як описуються і використовуються в програмі процедури користувача?*
4. *Що таке формальні і фактичні параметри?*
5. *Що таке глобальні і локальні змінні?*
6. *Які способи передавання параметрів існують в програмах на ТР? В чому їх особливості?*
7. *Як описуються і використовуються в програмі функції користувача? Які особливості має опис заголовку і тіла функції?*
8. *Вкажіть відмінності процедур і функцій.*
9. *Вкажіть правила локалізації ТР.*
10. *Що таке рекурсія? Наведіть приклади використання рекурсії.*
11. *Що таке побічний ефект? Наведіть приклади.*
12. *Для чого призначені стандартні процедури Exit і Halt?*

Рядкові величини у мові Паскаль

Написання більшості програм потребує використання не лише числових, а й текстових даних. Для подання текстової інформації ТР підтримує два стандартні типи: Char і String.

Значення типу Char – це непорожній символ з алфавіту ПЕОМ, взятий в одинарні лапки, наприклад, ’ ’, ’A’, ’7’ і т.п. Можна подавати символи і за допомогою їх кодів ASCII з допомогою спеціального префікса #:

#97 = Chr(97) = ’a’ (символ ’a’),
#0 = Chr(0) = (нульовий символ),
#32 = Chr(32) = ’ ’ (пропуск).

Операції над символами

Символи можна лише присвоювати і порівнювати один з одним. При порівнянні символів вони вважаються рівними, якщо рівні між собою їх ASCII-коди; і один символ більше за інший, якщо має більший ASCII-код. Операції і функції, що застосовні до величин символьного типу, наведені в таблицях 13 і 14.

Таблиця 13. Операції для величин символьного типу

Назва	Призначення	Приклади
=; <>; <; >; <=; >=	операції відношення	'A' < 'B' -> TRUE

Таблиця 14. Функції для величин символьного типу

Назва	Призначення	Приклади
ORD(c)	порядковий номер символу *c* в заданому символьному наборі	ORD('A') = 65; ORD(#4) = 4
CHR(i)	символ, що відповідає числу *i* згідно з кодами ASCII	CHR(65)='A'
PRED(c)	символ, що передує символу *c*	PRED('B') = 'A'
SUCC(c)	символ, що є наступним за символом c	SUCC('A')='B'
UPCASE(c)	символ верхнього регістру, відповідний ‘a’..’z’	UPCASE(‘a’)=’A’

Рядок у ТР є послідовністю символів. При використанні у виразах рядок береться в одинарні лапки. Кількість символів у рядку (довжина рядка) може набувати значень від 0 до 255. Для опису даних рядкового типу використовується тип String.

Формат опису рядкових даних:

Type

```
  Ім'я_типу1 = String[максимальна довжина рядка];
  Ім'я_типу2 = String;
Var
  змінна1: Ім'я_типу1;
  змінна2: String[макс. довжина рядка];
  змінна3: String;
```

String (рядок) – службове слово, максимальна довжина рядка – ціле число n, $0 \le n \le 255$.

Опис String еквівалентний String[255].

Приклад. Опис рядкових змінних.

```
Const
    Address = 'вул.Гетьмана Полуботка, 53';
 Type
    Str20 = String[20];
    Strmax = String[255];
    St  = String;
Var
    A: Strmax;  B: St20; C: St;  {довжини рядків A і C можуть бути від 0 до 255 символів}
    D: String[30];                {довжина B – від 0 до 20, D –від  0  до  30 символів}
```

Рядки, що мають різні довжини, можна присвоювати один одному і порівнювати. При використанні рядків у процедурах і функціях треба стежити, щоб формальні і фактичні параметри були одного типу. Краще використовувати тип String.

Рядки можна розглядати як масиви символів. Будь–який символ рядка можна отримати за його номером, починаючи з 1.

Приклад.

```
Var
   i : Byte;
   S : String[15];
Begin
   S := 'Масив символів.';
    for i:=1 to 15 do Writeln(S[i]); {друкування рядка S посимвольно}
    Readln  {пауза до натиснення Enter}
End.
```

Символ рядка з номером 0 містить код, який дорівнює кількості символів у рядку, тобто довжина рядка S завжди дорівнює Ord(S[0]).

Окремий символ сумісний за типом зі значенням типу Char.

На початку роботи програми рядкові змінні містять "сміття", тому завжди рекомендується перед використанням рядкових змінних надавати їм початкові значення, наприклад, порожній рядок ''. Для введення значень рядкових змінних використовується оператор Readln, а не Read. Одним оператором можна ввести тільки один рядок.

Приклад. `Readln(S); - правильне введення рядка,`
`Readln(S,S1); Read(S); - неправильне.`

Операції над рядками

Рядки можна порівнювати, присвоювати і зливати (отримувати суму).

Приклад.

```
Var
   S1, S2, S3 :String;
Begin
   S1 :=  ’Вам ’;
   S2 :=  ’привіт’;
   S3 :=  S! + S2;   {S3 = ’Вам  привіт’}
   S3 :=  S3 + ’!’    {S3 = ’Вам  привіт!’}
End.
```

Порівняння рядків відбувається посимвольно, починаючи з першого символу в рядку. Рядки вважаються рівними, якщо вони мають однакову довжину і посимвольно еквівалентні. Якщо при посимвольному порівнянні виявиться, що один символ більше за інший (його код більший), то рядок, що його містить, також вважається більше. Решти рядків і їх довжини не грають ролі. Будь–який рядок більше "порожнього місця".

Приклад.

```
’abcd’ =  ’abcd’      (TRUE),
’abcd’ <> ’abcde’     (TRUE),
’abcd’ > ’abcD’       (т.я. ’d’ > ’D’),
’abcd’ > ’abc’        (т.я.  ’d’ >  ’’),
’aBcd ’ < ’ab’        ( т.я. ’B’ < ’b’),
     ’ ’  >  ’’        (т.я.  #32 > ’’).
```

Для роботи з рядками у TP реалізована велика кількість процедур і функцій (таблиця 16).

Таблиця 16. Процедури і функції для опрацювання рядків

Процедури і функції	**Призначення, пояснення, приклади використання**
Length(S : String) : Byte	**Видає довжину рядка S. Аналогічно діє Ord(S[0])**
Concat(S1, S2, …, Sn): String	**Повертає конкатенацію або злиття рядків S1, S2,…,Sn. Аналогічно діє операція "+":** S3 :=Concat(S1,S2); {те ж саме, що S3:=S1 + S2}

Таблиця 16. Процедури і функції для опрацювання рядків

Процедури і функції	**Призначення, пояснення, приклади використання**
Copy(S : String; Start, Len : Integer) : String	**Повертає підрядок довжиною Len, який починається з позиції Start рядка S. Якщо Start більше всієї довжини рядка S, то функція повертає порожній рядок, а якщо Len більше, ніж кількість символів від Start до кінця рядка S, то повертається залишок рядка S від Start до кінця.** а) SCopy:=Copy(’ABC***123’,4,3); {SCopy = ’***’} б) SCopy:=Copy(’ABC’,4,3); {SCopy =’’} в)SCopy:=Copy(’ABC***123’,411); {SCopy = ’***123’}
Delete(Var S : String; Start, Len : Integer)	**Вилучає з S підрядок довжиною Len, який починається з позиції Start рядка S. Після вилучення підрядка його залишки склеюються. Якщо Start=0 або перевищує довжину рядка S, або Len=0, то рядок не змінюється. Якщо Len більше, ніж залишок рядка, вилучається підрядок від Start і до кінця S.** S:=’РЯДОК’; Delete(S,2,2); {S = ’РОК’}
Insert(SubS : String; Var S : String; Start : Integer)	**Вставляє в S підрядок SubS, починаючи з позиції Start. Якщо змінений рядок S виявляється надто довгим, то він автоматично скорочується з правого боку до оголошеної довжини S.** S:=’Ранок–вечір’; Insert(’день–’,S,7); {тепер S = ’Ранок–день–вечір’}
Pos(SubS , S : String) : Byte	**Шукає входження підрядка SubS в рядок S і повертає номер першого символа SubS в S або 0, якщо S не містить SubS. Недоліком даної функції є те, що вона повертає найближчу стартову позицію SubS в**

	S від початку рядка, ігноруючи наступні, якщо вони є. Так, виклик P:=Pos(’noo’, ’Boonoonoonoos’); завершує роботу, повернувши значення 4, хоча є ще 7 і 10.

Таблиця 16. Процедури і функції для опрацювання рядків

Процедури і функції	**Призначення, пояснення, приклади використання**
Str(X [:Width [: dec]]; Var S : String)	**Перетворює числове значення X в рядкове S. X може бути змінною або значенням цілого або дійсного типу. Можна задавати поля формату, вказавши ширину поля для числа і кількість десяткових знаків. Якщо число має менше знаків, ніж дано в полі, воно вирівнюється по правому краю, порожнє місце заповнюється пропусками. Якщо задати поле Width від'ємним, вирівнювання здійснюватиметься по лівому краю, а залишки стираються. Якщо формат з крапкою обмежує кількість знаків у дробовій частині, то вона буде округлена при перетворенні. Значення самого числа при цьому не зміниться.** Str(6.66:8:2, S); {S=' 6.66'} Str(6.66:–8:2, S); {S='6.66'} Str(6.66:8:0, S); {S =' 7' }
Val(S : String; Var X; Var ErrCode : Integer)	**Перетворює рядкове значення S (рядок цифр) в значення числової змінної X. Якщо перетворення можливе, то змінна ErrCode=0, інакше вона містить номер символу в S, на якому відбулася помилка. Тип X повинен відповідати вмісту рядка S.**

Приклад. Визначити, чи є задане слово паліндромом (паліндром – слово, що читається однаково зліва направо і навпаки, наприклад, "шалаш", "казак").

```
Type Str = String[20];
Var  S:Str; P:Boolean; i:Byte;
Begin
    ClrScr;
    Writeln ('Введіть слово');  Readln(S);
     P:=TRUE;
     for i:=1 to Length(S) div 2 do
        if S[i] <> S[Length(S)–i+1]  then P:=FALSE;
```

```
        if P then Writeln('паліндром')  else Writeln('не паліндром')
End.
```

Приклад. Дана послідовність слів (до 10 слів), в кожному до 8 літер. Надрукувати послідовність в зворотному порядку.

```
Type MasStr=Array[1..10] of  String[8];
Var   Sl:MasStr;  i:Integer;
Begin
  for i:=1  to 10  do  Readln(Sl[i]);
  for i:=100   downto  1  do  Writeln(Sl[i])
End.
```

Приклад. Дано речення, яке містить 1 – 20 слів, в кожному слові 1 – 5 літер, між словами – пропуски. Надрукувати слова речення у зворотному порядку.

```
Uses Crt;
Type  Slovo = String[5];  {окреме слово}
        Rech = String;        {дане речення}
        MasSliv = Array[1..20] of Slovo;  { створюваний масив слів}
Var  S:Slovo;  R:Rech; M: MasSliv;

{Функція розбиття речення на слова}
Function GetWord(S:Rech; Var N:Byte):Slovo;
Var Tmp: String;
Begin
    Tmp:='';
     while (N <= Length(S)) and (S[N] = ' ')  do N:=N+1;  {ігнорування пропусків}
     while (N <=  Length(S)) and (S[N] <> ' ') do
          Begin
             Tmp:=Tmp+S[N];  {створення окремого слова  – елемента масиву слів}
              N:=N+1
           End;
      GetWord:=Tmp
End;

Var i:Byte; {лічильник символів речення}
      N:Byte;{лічильник слів}
      k:Byte;
Begin
    {створення масиву слів}
    i:=1; N:=0;
    while i <= Length(R) do
        Begin
           N:=N+1;
           M[N]:=GetWord(R,i)
         End;
{друкування слів з масиву у зворотному порядку}
```

```
    for k:=N downto 1 do Writeln(M[k])
End.
```

Питання для самоконтролю.

1. *Для чого призначені типи даних Char і String?*
2. *Як описуються змінні символьного і рядкового типів засобами TP?*
3. *Чи сумісні типи Char і String?*
4. *Для чого використовуються Pred, Succ, Chr, Ord? Опишіть їх дію на прикладах.*
5. *Як забезпечується доступ до символу за його кодом без використання функції Chr?*
6. *Як здійснюється доступ до окремого символу рядка?*
7. *Вкажіть два способи визначення довжини рядка.*
8. *Як перетворити число у рядок та навпаки?*
9. *Чи можна порівнювати рядки? Як?*
10. *Якими способами можна з'єднати кілька рядків?*
11. *Яку максимальну довжину може мати рядкова змінна?*
12. *За допомогою яких операторів рядкові змінні вводяться з клавіатури?*
13. *Як перетворити маленькі латинські літери у великі?*
14. *Яка процедура призначена для вставки підрядка у рядок? Наведіть приклади її використання.*
15. *Яка процедура призначена для вилучення підрядка з даного рядка? Наведіть приклади її використання.*
16. *Яка функція призначена для отримання підрядка даного рядка? Наведіть приклади її використання.*
17. *Як дізнатися, чи входить один рядок до складу іншого? Наведіть приклади.*

Множини у мові Паскаль

Множина – це структурований тип даних, який подає набір взаємопов'язаних за певною ознакою об'єктів. Кожний об'єкт у множині називається елементом множини.

Всі елементи множини повинні належати одному типу. Цей тип називається **базовим типом** множини. Базовим може бути будь-який простий дискретний тип. Він задається діапазоном або переліком значень. Область значень типу множина – набір всіх підмножин, складених з елементів базового типу. Якщо базовий тип має N значень, то тип множина для нього буде мати 2^N значень.

Формат опису типу множина:

```
Type
   Ім'я_типу> = Set of  базовий_тип;
Var
   Ім'я_змінної : Ім'я_типу;
```

Можна задати множинний тип і без попереднього опису:

```
Var
   Ім'я_змінної : Set of  базовий_тип;
```

Set (множина), of (з) - службові слова.

Приклад.

```
Type
SetChar = Set of Char; {множина з символів}
   SetByte = Set of Byte; {множина з чисел}
   SetDigit =  Set of 0..9; {множина з чисел від 0 до 9}
   SetDChar = Set of  '0'..'9'; {множина з символів '0', '1',…,'9'}
SetSpring = Set of  (March, April, May); {множина з весняних місяців, базовий
   SetGolosn = Set of  ('А', 'О', 'У', 'І', 'Є', 'И', 'Ї' ); {множина з великих голосних
літер}
```

В мові ТР дозволяється визначати множини, які складаються не більше, ніж з 256 елементів. Стільки ж елементів містять типи Byte (0..255) і Char, і це ж число є обмеженням кількості елементів у будь–якому іншому перелічувальному базовому типі множини, який задає користувач. Кожному елементу перелічувального типу приписується певний номер. Для типу Byte номер дорівнює значенню числа, у типі Char номером символу є його ASCII–код. Нумерація здійснюється від 0 до 255. З цієї причини не можуть бути базовими для множин типи ShortInt(–128..127), Word(0..65535), Integer(–32768..32767),
LongInt(–2147483648..2147483647).

Змінна типу множина підлягає певному синтаксису. Елементи множини повинні братися у квадратні дужки. Порожня множина записується як [].

Приклад.

```
SByte := [1,2,3,4,10,20,30,40];
SChar := ['a', 'b', 'c'];
SChar :=  [ 'd'];
SSpring := [April];
SDiap := [1..4]; {те ж саме, що [1,2,3,4]}
SComp := [1..4, 5,7,10..20];
Empty := []; {порожня множина – підмножина будь-якої множини}
```

Порядок слідування елементів всередині дужок не має значення, не має значення і кількість повторення елементів. Повторні входження елементів ігноруються.

Так, записи ['a','b','b','a'] і ['a','b'] еквівалентні.

Приклад. Всі множини, які можна побудувати на базовому типі 1..3.

```
Var S:Set of 1..3;
Begin
  {допустимі присвоювання}
  S:=[]; S:=[1]; S:=[2]; S:=[3]; S:=[1,2]; S:=[1,3]; S:=[2,3]; S:=[1,2,3];
End.
```

Операції над множинами

Операції над множинами поділяються на такі, результатом яких є логічне значення, і такі, результатом яких є множина (таблиця 17).

Таблиця 17. Операції над множинами

	Назва	**Формат**	**Пояснення, приклади використання**
=	Перевірка на рівність	S1 = S2	Результатом є логічне значення, рівне TRUE, якщо S1 і S2 складаються з однакових елементів незалежно від порядку слідування, і FALSE у протилежному випадку. [1,2,3] = [1,3,2]; [] = []; ['a'..'c'] = ['a','b','c']
<>	Перевірка на нерівність	S1<>S2	Результатом є логічне значення, рівне TRUE, якщо S1 і S2 відрізняються хоча б одним елементом, і FALSE у протилежному випадку. [1,2] <> [1]; [] <> [3]; ['a'..'c'] <> ['a','c']
<=	Перевірка на підмножину	S1<=S2	Результатом є логічне значення, рівне TRUE, якщо всі елементи S1 містяться і в S2 незалежно від їх порядку слідування, і FALSE у протилежному випадку. ['a','b'] <= ['a'..'z']; [] <= [4]; [1,2] <= [1,2,3]

Таблиця 17. Операції над множинами

	Назва	**Формат**	**Пояснення, приклади використання**
>=	Перевірка на надмножину	S1>=S2	Результатом є логічне значення, рівне TRUE, якщо всі елементи S2 містяться в S1, і FALSE у протилежному випадку. ['x'..'z'] >=['y']; [1..10] >=[1..5]; [5,7] >=[]
in	Перевірка входження елемента у множину	E in […] E in S1	Результатом є логічне значення, рівне TRUE, якщо значення E належить базовому типу множини і входить у множину […] (S1). Якщо множина не містить у собі значення K, то результатом є FALSE. 5 in [0..5]; 's' in ['a'..'z']; not (7 in [9..20])
+	Об'єднання множин	S1 + S2	Результатом об'єднання є множина, отримана злиттям елементів цих множин і виключенням елементів, що повторюються. [1,2]+[2,3,4] = [1,2,3,4]; {[1..4]} ['s'] + [] = ['s']
–	Різниця множин	S1 – S2	Результатом операції отримання різниці S1 – S2 є множина, складена з елементів, які входять в S1, але не входять в S2. [5,7,9] – [7] = [5,9]; ['1','2'] – ['8','9'] = ['1','2']
*	Перетин множин	S1 * S2	Результатом перетину є множина, що складається лише з тих елементів S1 і S2, які містяться одночасно і у S1, і у S2. [3,4,5,6,7] * [4,5,8] = [4,5]; ['x'] * [] = []

Переваги типу множина:
- компактність подання – один елемент множини займає один біт пам'яті;
- відсутність необхідності заздалегідь вказувати кількість елементів множини – по ходу програми множина може розширюватись або скорочуватись;
- поліпшення наочності програм і гнучкості мови програмування.

Основним недоліком є те, що Турбо Паскаль не дозволяє виводити множини на екран і отримувати окремі елементи з множин. Введення множин можливе лише по елементам.

Робота з довільною множиною переважно відбувається за такою схемою:
- перевірити всі елементи базового типу на входження в дану множину;
- в разі входження вивести елемент на екран або потрібним чином опрацювати.

Приклад. Створити дві множини з елементів символьного типу, знайти перетин, об'єднання, різницю цих множин, надрукувати елементи кожної з множин у алфавітному порядку.

```
Uses Crt;
Type S = Set of 'a'..'z';
Var A, B, C: S;
```

```
      sym: Char;
{Введення множини}
Procedure VvedSet(Var A:S; g: Char);
Var ch: Char;
Begin
  Writeln('Множина ', g, 'створюється з маленьких латинських літер.');
  Writeln('Після введення кожної літери – ENTER');
  Writeln('Закінчення введення – крапка');
    A: = [];              {Ініціалізація множини}
    Read(ch); Write(' ');
    While ch <> '.' do
         Begin
           A :=  A + [ch];  {Поповнення множини новим елементом}
           Read(ch); Write(' ')
         End;
End;
{Перетин множин}
Procedure Peretyn(A, B: S; Var C: S);
Begin
   C := A *  B;
   Writeln('Перетин: ')
End;
{Об’єднання множин}
Procedure Obiedn(A,B: S; Var C: S);
Begin
   C := A + B;
   Writeln('Об’єднання: ')
End;
{Різниця множин}
Procedure Rizn(A, B: S; Var C: S);
Begin
    C := A – B;
  Writeln('Різниця:')
End;
{Виведення множини}
Procedure VyvedSet(A: S; g: Char);
Var ch: Char;
Begin
   Writeln('Множина ', g, ':');
   for ch := 'a' to 'z' do
     if ch in A then Write(ch:3)
End;
Begin
    repeat
      ClrScr;
```

```
      VvedSet(A, 'A'); VvedSet(B, 'B');
       Peretyn(A, B, C); VyvedSet(C, 'C');
       Obiedn(A, B, C);  VyvedSet(C, 'D');
       Rizn(A, B, C);  VyvedSet(C, 'E');
       Writeln('Повторити?');
       sym := ReadKey
    until not (sym in ['Т','т','Д','д','Y','y']
End.
```

Приклад. Дана послідовність латинських літер, яка закінчується крапкою. Вивести перші входження літер у послідовність, зберігаючи їх початковий взаємний порядок. (Наприклад, вхідний текст – qqwerrtyy., вихідний текст – qwerty).

```
Uses Crt;
Var Let : Set of 'a'..'z';
     ch : Char;
Begin
  ClrScr;
   Let := []; {Множина літер у розглянутій частині тексту}
   Read(ch);
   while ch <> '.' do
      Begin
         if not (ch in Let) then   {Перше входження літери}
             Begin
                Write(ch);
                 Let := Let + [ch]
              End;
         Read(ch)
       End
End.
```

Питання для самоконтролю.

1. *Для чого призначений тип даних Set? Які його переваги і недоліки?*
2. *Як описується множинний тип засобами ТР?*
3. *Які типи даних можуть використовуватись як базовий тип множини?*
4. *Скільки елементів може містити множина в ТР? Скільки різних множин можна утворити на основі заданого базового типу?*
5. *Вкажіть операції над множинами у ТР, результатом яких є логічне значення. Наведіть приклади їх використання.*
6. *Вкажіть операції над множинами у ТР, результатом яких є множина. Наведіть приклади їх використання.*
7. *Як здійснюється доступ до окремого елемента множини?*
8. *Чи має значення порядок слідування елементів у множині?*

Записи у мові Паскаль

Розглянуті раніше структуровані типи даних мови ТР (масив, множина), складаються з компонентів лише одного типу. Проте для розв'язання широкого класу задач потрібно групувати дані різних типів, які відносяться до одного об'єкта. Наприклад, доцільно об'єднати дані про власника автомобіля (номер, марка машини, пробіг, прізвище власника, адреса) в одній структурі. Для цього у мові ТР передбачений тип даних запис, або комбінований тип.

Запис – це структурований тип даних, якій складається з фіксованої кількості компонентів різних типів. Опис типу починається службовим словом Record (запис) і закінчується службовим словом End. Між ними розташовується список компонентів, які називаються полями, з указанням імен полів і типу кожного поля.

Формат опису типу запис:

```
Type
   Ім'я_типу = Record
                    ім'я_поля1: тип_поля1;
                    …
                    ім'я_ поляN : тип_поляN
                   End;
Var
   Ім'я_змінної : Ім'я_типу;
```

Приклад.

```
Type
Avto = Record
          Nomer : String[8]; {номер}
          Marka :  String[20]; {марка автомобіля}
          Probig : Integer; {пробіг у км}
          PIB : String[40]; {прізвище – ім'я – по-батькові власника}
       End;
Var M, V : Avto; {змінні типу запис}
```

Якщо декілька полів мають однаковий тип, то їх імена в описі типу просто перераховуються:

Приклад.

```
Type
PointType = Record
               x, y : Integer
            End;
Var Point : PointType;
```

Після опису в програмі змінної типу запис до кожного її поля можна звернутися, вказавши спочатку ім'я змінної – запису, а потім через крапку – ім'я поля. Така конструкція називається складеним іменем.

M.Nomer, M.Marka, M.Probig, M.PIB, Point.x, Point.y – це складені імена, що подають значення полів записів, але при цьому M – комбінація чотирьох значень, Point – двох.

Значення полів запису можуть використовуватися у виразах. Імена окремих полів не використовуються по аналогії з іменами змінних. Складене ім'я можна використовувати всюди, де дозволяється використання типу поля. Для надання полям значень використовується оператор присвоювання.

Приклад. M.Nomer := '00035MK'; M.Marka := 'ВАЗ-2108'; M.Probig := 16000;M.PIB := 'Ткачук П.П.';
Point.x :=12; Point.y := 24;

Складені імена можна використовувати, зокрема, в операторах введення-виведення. При цьому введення і виведення записів відбувається не в цілому, а по полях.

Приклад. Read(M.Nomer, M.Marka, M.Probig, M.PIB); Write(M.Nomer:7, M.Probig:6);

Допускається застосування оператора присвоювання і до записів в цілому, якщо вони мають один і той самий тип, наприклад V :=M;. Після виконання цього оператора значення полів запису V дорівнюватимуть значенням відповідних полів запису M.

В ряді задач зручно користуватися масивами із записів. Їх можна описувати, зокрема, так:

Приклад.

```
Type
Person = Record
              PIB: String[30];
               Vik: 0 .. 90;
               Prof : String[40]
          End;
ListPerson = Array [1..50] of Person;
Var  L : ListPerson;
```

Тоді доступ до полів запису, який у масиві розташований за номером i, здійснюватиметься за такими складеними іменами: L[i].PIB, L[i].Vik, L[i].Prof.

Складені імена часто бувають громіздкими і незручними у використанні. Для компактності програм використовується **оператор приєднання** With, який має такий формат:

With змінна_типу_запис do оператор;

With (з), do (виконувати) - службові слова, оператор – довільний оператор, простий або складений. В останньому випадку необхідні операторні дужки.

Один раз вказавши змінну типу запис в операторі With, можна працювати з іменами полів, як із звичайними змінними, тобто без вказування перед іменем поля імені змінної-запису.

Приклад. Надати значення полям запису Point за допомогою оператора With.

```
With Point do
   Begin
      x :=12;
      y := 24
```

```
    End;
```

Турбо Паскаль допускає вкладення записів один в один (тобто поле запису може в свою чергу бути записом), відповідно оператор With також може бути вкладеним:

```
With R1 do
   With R2 do …
      With Rn do
```

що еквівалентно написанню

```
With   R1,R2,…,Rn do…
```

Рівень вкладеності не повинен перевищувати 9.

Приклад.

```
Type man = Record
                surname, town: String[20];
                address: Record
                         street: String[30];
                         house, flat: 1..999
                           End;
              End;
Var citizen:man;
Begin
{citizen}
With citizen  do
  Begin
   surname:=’Пеньков’;
   town:=’Чернігів’;
     With address do
       Begin
         street:=’вул. Преображенська’;
         house:=1;
         flat:=6
       End
    End;
```

Приклад. Описати комбінований тип, що містить відомості про людей: прізвище, місто проживання, адресу (назва вулиці, номери будинку та квартири). Вивести прізвища будь-яких двох людей зі списку, що живуть в різних містах, але за однаковими адресами.

```
Uses Crt;
Type Man = Record
                  Surname, Town: String[20];
                   Address: Record
                                Street: String[20];
                                 House, Flat : 1..999
                               End
                End;
```

```
        List = Array [1..15] of Man;
 Var Hum : List;
       i, j, n : Byte;
       b1, b2, b3, b4 : Boolean;
Begin
   ClrScr;
Write('Введіть кількість людей у списку: ');
Readln(n);
Writeln('Введіть дані про кожну людину:');
for i :=1 to n do
  With Hum[i] do
      Begin
         Writeln(i:2);
         Write('Прізвище: '); Readln(Surname);
         Write('Місто: '); Readln(Town);
           With Address do
              Begin
                Write('Вулиця: '); Readln(Street);
                Write('Будинок N '); Readln(House);
                Write('Квартира N '); Readln(Flat);
              End
        End;
    for i :=1 to n-1 do
      for j := i+1 to n do
        Begin
          b1 := Hum[i].Town <> Hum[j].Town;
          b2 := Hum[i].Address.Street  = Hum[j].Address.Street;
          b3 := Hum[i].Address.House  = Hum[j].Address.House;
          b4 := Hum[i].Address.Flat  = Hum[j].Address.Flat
           if b1 and b2 and b3 and b4 then
              Begin
                 Writeln('За іронією долі');
                 Writeln(Hum[i].Surname,' і  ', Hum[j].Surname);
                 Writeln('живуть в різних містах за однаковими адресами')
               End
        End
 End.
```

Приклад. Дано відомості про групу людей: ім'я, стать, зріст. Описати: 1) процедуру, що визначає середній зріст жінок у групі; 2) функцію, що визначає ім'я самого високого чоловіка; 3) логічну функцію, яка визначає, чи є у групі хоча б дві людини однакового зросту.

```
Const n=20;
Type standard=140..215;
      one = Record
```

```
                name:String;
                sex:Char;
                height:standard
          End;
   group = Array[1..n] of one;
Procedure Inputgroup(Var hum:group);
Var i:Integer;
Begin
   Writeln('Введіть дані про ',n,' людей:');
   for i:=1 to n do
     With hum[i] do
       Begin
          Writeln('Введіть ім'я:'); Readln(name);
          Writeln('Введіть стать  (ч,ж): '); Readln(sex);
          Writeln('Введіть зріст (140..215):'); Readln(height);
      End;
End;
Procedure AverageH(hum:group);
Var i, nn: Integer; s:Real;
Begin
  s:=0; nn:=0; {сумарний зріст і кількість жінок}
   for i:=1  to  n do
     With hum[i] do
       if sex='ж' then begin s:=s+height; nn:=nn+1 end;
Writeln('Середній зріст жінок у групі: ',s/nn:6:2)
End;
Function MaxH(hum:group):String;
Var i:Integer; max: standard; nn:String;
Begin
   max:=140;
   for  i:=1 to n do
     if (hum[i].sex='ч') and (hum[i].height>=max) then
       Begin
          max:=hum[i].height;
          nn:=hum[i].name
       End;
    MaxH:=nn
End;
Function SameH(hum:group):Boolean;
Var i,j:Integer; p:Boolean;
Begin
   p:=FALSE;
   for i:=1 to n-1  do
      for j:=i+1  to n do
        if hum[i].height=hum[j].height then
```

```
        p:=TRUE;
  SameH:=p;
End;
Var  L:Group;
Begin
  InputGroup(L);
  AverageH(L);
  Writeln('Серед чоловіків найвищий ',MaxH(L));
  if SameH(L)  then
     Writeln('У групі є  люди однакового зросту')
    else Writeln('У групі немає людей однакового зросту')
End.
```

Приклад. Описати процедуру, яка знаходить час між двома подіями, що відбулися протягом одної доби.

```
Type
  time=Record
          hour, minute, second: Integer
        End;
Procedure Period(a,b: time; var c:time);
Var t, t1, t2: Integer;
Begin
 {переведення даних величин часу в секунди}
  t1:=3600*a.hour+60*a.minute+a.second;
  t2:=3600*b.hour+60*b.munute+b.second;
 {знаходження шуканого часу в секундах}
  t:=abs(t2-t1);
  {переведення результату у години, хвилини і секунди}
  With c  do
    Begin
      hour:=t div 3600;
      minute:=(t-hour*3600) div 60;
      second:= t-hour*3600-minute*60
    End;
End;
```

Приклад. Описати комбінований тип, що містить відомості про студентів деякого вузу: прізвище, ім'я, стать, рік народження, курс. Визначити: 1) номер курсу, на якому навчається найбільша кількість чоловіків, 2) найпоширеніше жіноче ім'я.

```
Uses Crt;
Type Stud = Record   {Відомості про окремого студента}
                    Surname, Name: String[20];
                    Sex : Char;
                    BirthYear : 1900..2000;
```

```
                Course : 1..5
              End;
        Inform = Array [1..200] of  Stud;  {Відомості про всіх студентів}
        Courses = Array [1..5] of Byte;     {Кількість студентів-чоловіків на
кожному курсі}
 Var St : Inform; C: Courses;
      i, ii, j, max, n, p : Byte;
Begin
   ClrScr;
Write('Введіть кількість студентів: ');
Readln(n);
Writeln('Введіть дані про кожного студента:');
for i :=1 to n do
  With St[i] do
      Begin
         Writeln(i:2);
         Write('Прізвище: '); Readln(Surname);
         Write('Ім'я: '); Readln(Name);
         Write('Стать (літери m  або w: '); Readln(Sex);
         Write('Рік народження: '); Readln(BirthYear);
         Write('Курс: '); Readln(Course);
       End;
    for i :=1 to 5  do  C[i] := 0; {Ініціалізація масиву C}
      for i := 1 to n do
         With St[i] do
            Begin
              if  Sex = 'm' then
                 Case Course of
                    1: C[1] := C[1] +1; {Кількість чоловіків на першому курсі}
                    2: C[2] := C[2] +1;
                    3: C[3] := C[3] +1;              { … }
                    4: C[4] := C[4] +1;
                    5: C[5] := C[5] +1;  {Кількість чоловіків на п'ятому курсі}
                 End
            End;
   max := C[1];     {Найбільша кількість чоловіків на курсі }
   ii :=1;              {Номер відповідного курсу}
      for i :=2 to 5 do
        if C[i] > max then
           Begin
              max := C[i];
              ii := i
           End;
     Writeln('Найбільша кількість чоловіків навчається на ', ii:2, '-му курсі '); )
```

```
 ii :=1;       {Номер дівчини з найпоширенішим ім'ям}
 max := 1;  {Кількість дівчат з найпоширенішим ім'ям}
    for i := 1 to n - 1 do
      Begin
        if St[i].Sex = 'w' then
          Begin
            p :=1;
            for j := i + 1 to n  do
              if St[i].Name = St[j].Name  {Пошук тезок}
                then p := p + 1;
            if p > max then
              Begin
                max := p;
                 ii := i
               End
          End
      End;
   Writeln('Найпоширеніше жіноче імя: ', St[ii].Name)
End.
```

Записи з варіантами

Розглянуті вище записи мають строго визначену структуру. В деяких випадках це обмежує можливості їх застосування. Тому в мові ТР є можливість задати тип запису, який містить довільну кількість варіантів структури. Такі записи називаються записами з варіантами. Записи з варіантами забезпечують засоби об'єднання записів, які схожі, але не ідентичні за формою. Вони складаються з необов'язкової фіксованої і варіантної частин. Використання фіксованої частини не відрізняється від описаного вище. Варіантна частина формується за допомогою оператора Case. Він задає особливе поле запису – поле ознаки, яке визначає, який з варіантів в даний момент буде активізований. Значенням ознаки в кожний поточний момент виконання програми має бути одна з розташованих далі констант. Константа, яка є ознакою, задає варіант запису і називається константою вибору.

Формат:

```
Type
Zap = Record
        Case поле_ознаки : ім'я_типу of
           константа_вибору_1 :  (поле, …: тип);
            …
           константа_вибору_n : (поле, …: тип)
         End;
```

Компоненти кожного варіанту (імена полів і їх типи) беруться у круглі дужки. У частини Case немає окремого End, як цього можна було б чекати по аналогії з оператором Case. Одне слово End завершує всю конструкцію запису з варіантами.

Приклад.

```
Type
Zap = Record
          Nomer : Byte;
          Artycul: Integer;
              Case  Flag : Boolean of
                   TRUE : ( Cina1 : Integer);
                    FALSE : (Cina2 : Real)
         End;
Var PZap: Zap;
```

Поля Nomer і Artycul розташовані у фіксованій частині запису, вони доступні у програмі в будь-який поточний момент незалежно від поля ознаки. Поле Cina1 може використовуватися лише в тому випадку, коли значення поля ознаки Flag дорівнює TRUE. Поле Cina2 доступне у протилежному випадку, тобто якщо значення Flag дорівнює FALSE.

При використанні записів з варіантами слід дотримуватися таких правил:
всі імена полів повинні відрізнятися одне від одного принаймні одним символом, навіть якщо вони зустрічаються у різних варіантах;
запис може мати лише одну варіантну частину, причому варіантна частина повинна бути розташована в кінці запису;
якщо поле, що відповідає певній мітці, є порожнім, то воно записується так: мітка: ( );.

Питання для самоконтролю.

1. *Для чого призначений тип даних Record? Які його особливості?*
2. *Як описується комбінований тип засобами ТР?*
3. *Чи повинні всі поля запису мати однаковий тип?*
4. *Як відбувається звертання до полів запису? Які дії можна виконувати з полями записів у ТР?*
5. *Які операції можна виконувати в ТР над змінними-записами вцілому?*
6. *Для чого призначений оператор With? Наведіть приклади його використання.*
7. *Чи можуть записи бути вкладеними?*
8. *Для чого призначені записи з варіантами?*

Робота з файлами у системі Турбо-Паскаль

Для будь-якого обміну інформацією необхідні джерело інформації, приймач інформації і канал передавання. При роботі Паскаль-програми відбувається обмін даними між оперативною пам'яттю комп'ютера і файлом.

Поняття файла досить широке. Це може бути звичайний файл даних на диску або комунікаційний порт, принтер або інше. Файл може бути джерелом інформації – тоді відбувається читання з файла, або приймачем – тоді відбувається запис інформації у файл.

Файлова система, що реалізується у Турбо Паскалі, складається з двох рівнів: логічних і фізичних файлів.

Фізичний файл у TP розуміється так, як і в операційній системі MS-DOS: це область пам'яті на диску, що має ім'я. Фізичний файл визначається рядком з його іменем. В TP імена можуть бути рядковими константами або рядковими змінними, значення яких відповідає правилам MS-DOS.

Приклад. Можливі імена файлів:
'C:\PAS\TESTFILE.PAS',
'A:TEST.TXT',
'..\PROBA.BAS'.

До фізичних файлів відносяться пристрої MS-DOS. Пристрої мають фіксовані імена, які можуть використовуватись у файлових операціях TP з текстовими файлами. Імена пристроїв і зауваження по них наведені у таблиці 18:

Таблиця 18. Стандартні пристрої MS-DOS

Ім'я	Пристрій	Пояснення
CON	Консоль (клавіатура і екран)	Введення з CON – це читання з клавіатури, а виведення в CON – це запис на екран
LPT1 LPT2 LPT3	Паралельні порти (типу Centronix) номер 1..3 (якщо встановлені)	Через ці імена файлів відбувається виведення даних на принтери або інші пристрої з інтерфейсом типу Centronix
PRN	Принтер. Синонім імені LPT1	Ім'я звертання до принтера, включеного у порт LPT1
COM1 COM2	Послідовні порти (типу RS-232) номер 1..2 (якщо встановлені)	Імена файлів-пристроїв для введення-виведення даних через серійні порти комунікації
AUX	Синонім імені COM1	Файл-пристрій COM1

NUL	**Фіктивний пристрій**	**Це бездонний файл, який приймає що завгодно, але завжди порожній**

Логічний файл описується як змінна одного з файлових типів, визначених у ТР. Після опису файлової змінної її можна пов'язати з будь-яким фізичним файлом, незалежно від його природи. Введення логічного файла дозволяє програмісту не замислюватися про технічні проблеми організації обміну даними, а займатися програмуванням самого потоку даних. Різні фізичні файли мають різні механізми введення і виведення інформації. Логічні ж файли стандартизують роботу з фізичними файлами, що дозволяє працювати не безпосередньо з пристроями ПЕОМ, а з їх логічними позначеннями.

Файлові типи Турбо-Паскаля

Файл як структура даних – це скінченний упорядкований набір елементів, характер яких визначається типом файла.

Турбо Паскаль підтримує три файлові типи:

1. Текстові файли (типу Text);

Формат опису:

Var Ім'я_змінної : Text;

2. Типізовані файли (типу File Of …);

Формат опису:

Type Ім'я_ типу = File of тип_елементів;

Var Ім'я_змінної : Ім'я_типу;

3. Нетипізовані файли (типу File).

Формат опису:

Var Ім'я_змінної : File;

Для всіх типів файлів мінімальною одиницею інформації, що в них зберігається, є байт.

Для всіх типів файлів перед роботою з ними необхідне пов'язування їх логічних позначень (файлових змінних) з фізичними файлами.

Файлові змінні, описані в програмі, не можуть брати участь в операторах присвоювання.

При використанні файлових змінних як параметрів процедур і функцій вони повинні бути описані як параметри-змінні.

Доступ до елементів файла здійснюється через файловий покажчик (буферну змінну). При читанні або запису цей покажчик переміщується до наступного елемента і робить його доступним для опрацювання. Елементи файлу нумеруються, починаючи з нуля. Буферна змінна на відміну від усіх інших змінних не може брати участь у виразах і операторах присвоювання.

Існує два способи доступу до елементів файла: послідовний і довільний (прямий). При **послідовному** способі доступу пошук починається з початку файла і перевіряється по черзі кожний елемент, доки не буде знайдено потрібний. При **прямому** доступі до потрібного елемента можна звернутися за

його порядковим номером у файлі. Текстові файли є файлами послідовного доступу, типізовані і нетипізовані – прямого.

Стандартні процедури і функції для роботи з файлами

Для організації ефективної і зручної для користувача роботи з файлами в TP існує ряд процедур (таблиця 19) і функцій (таблиця 20). При їх описі використаємо такі позначення:

F – файлова змінна;

Str – рядковий вираз;

P – змінні p1, p2, …, pn того ж типу, що і елементи змінної F;

n – цілочисельний вираз.

Таблиця 19. Стандартні процедури для роботи з файлами

Назва	Призначення	Пояснення
Assign(F, Str)	Надати ім'я файлу	Файлова змінна F пов'язується з іменем фізичного файла, заданим у рядку Str
Rewrite(F)	Відкрити файл для запису	Ця процедура служить для створення нового файла на диску. Ім'я файла було попередньо визначене в процедурі Assign. Якщо на диску вже був файл з таким іменем, він знищується. Покажчик файла встановлюється в першу позицію (з номером 0). Фактично файл не містить жодного елемента, він лише підготовлений до завантаження
Reset(F)	Відкрити файл для читання	Виконання процедури забезпечує встановлення покажчика файла на перший елемент. Якщо процедура застосована до файла, що не існує, виникає помилка введення-виведення
Read(F, P)	Читати з файла	Відбувається читання з дискового файла, визначеного файловою змінною F, значень p1, p2, …, pn. Після завершення виконання процедури покажчик переміщується до наступного елемента
Write(F, P)	Записати у файл	Змінні p1, p2, …, pn записуються у дисковий файл,

		визначений змінною F. Після виконання процедури покажчик переміщується до наступного елемента

Таблиця 19. Стандартні процедури для роботи з файлами

Назва	Призначення	Пояснення
Seek(F,n)	Встановити покажчик на елемент з порядковим номером n у файлі	Покажчик переміщується до елемента з номером n, починаючи відлік з 0, тобто перший елемент має номер 0, другий – 1 і т.д.
Flush(F)	Очистити буфер сектора	Виконання процедури викликає виштовхування внутрішнього буфера у файл, якщо раніше виконувались операції запису. Фактично відбувається очищення буфера лише текстового файла. До закритого файла процедура Flush не застосовується
Close(F)	Закрити файл	Виконання процедури забезпечує закриття файлу, призначеного змінній F. Якщо файл був відкритий, ніколи не слід виходити з програми, попередньо не закривши його
Erase(F)	Знищити файл	Виконання процедури викликає знищення файла, призначеного змінній F. Якщо здійснюється знищення відкритого файла, його треба попередньо закрити за допомогою процедури Close
Rename(F, Str)	Перейменувати файл.	Виконання процедури викликає занесення у каталог диску нового імені файла, визначеного змінною F. Нове ім’я визначається значенням Str
Truncate(F)	Відсікти частину файла	Знищуються всі елементи файла, починаючи з місця поточного положення покажчика, і файл підготовлюється для запису

Таблиця 20. Стандартні функції для роботи з файлами

Назва	Призначення	Пояснення
Eof(F)	Перевірити маркер "кінець файла"	Значення функції дорівнює TRUE, якщо покажчик файла знаходиться відразу за останнім елементом файла, і FALSE в іншому випадку

FilePos (F)	Визначити поточний номер елемента файла. Відлік номера елемента починається з нуля	Функція повертає цілочисельне значення, рівне номеру елемента, на якому встановлений в даний момент покажчик файла, відповідного змінній F. Відлік номера елемента починається з нуля

Таблиця 20. Стандартні функції для роботи з файлами

Назва	Призначення	Пояснення
FileSize(F)	Визначити довжину файла	Функція повертає цілочисельне значення, рівне кількості елементів файла, відповідного змінній F. Ця функція переважно використовується для перевірки, чи містить файл певні дані, чи є порожнім. Якщо FileSize(F) = 0, то файл порожній, інакше файл містить дані
IOResult(F)	Перевірити результат вико-нання останньої операції введення-виведення на наявність помилок	Якщо помилку знайдено, функція повертає номер помилки, якщо помилок немає, повертає значення 0. Ця функція використовується при пасивному стані директиви I ({$I-}) для організації обробки помилок самим користувачем. Якщо програма для обробки помилок відсутня, наявність помилки введення-виведення не викликає переривання програми, і виконується наступний оператор

Текстові файли

Текстовими є файли, в яких:

- інформація подається у текстовому вигляді за допомогою символів у коді ASCII;
- вміст може поділятися на рядки. Ознакою кінця рядка є символ #13 (код 13 – CR). Він може бути об'єднаний з символом переведення рядка #10 (код 10 – LF);
- кінець файла позначається явно символом ^Z (код 26);
- при запису чисел, рядків і логічних значень вони перетворюються у символьний (текстовий) вигляд;
- при читанні чисел і рядків вони автоматично перетворюються з текстового подання у машинне.

У системній бібліотеці Турбо Паскаля визначені дві текст-файлові змінні: Input і Output. Вони пов'язані з пристроєм 'CON' (або з фіктивним пристроєм CRT, якщо підключений модуль CRT) автоматично. Якщо у процедурах введення пропущене ім'я файла, вважається, що введення здійснюється з системного файла Input (це клавіатура), а якщо ім'я файла пропущене в операторі виведення, то виведення здійснюється у файл Output (на екран).

Текстові файли у Турбо Паскалі – не аналоги файлів типу File of Char.

Стандартні процедури і функції для роботи з текстовими файлами

Для роботи з текстовими файлами призначені такі процедури і функції (таблиця 21):

Таблиця 21. Процедури і функції для роботи з текстовими файлами

Назва	Призначення
Assign(F, Str)	Надання імені текстовому файлу.
Rewrite(F)	Відкриття нового текстового файла для запису.
Reset(F)	Відкриття існуючого текстового файла для читання.
Append(F)	Відкриття існуючого текстового файла для дописування інформації.
Close(F)	Закриття текстового файла.
Read(F, Ch)	Зчитування символу Ch з текстового файла F.
Write(F, Ch)	Запис символьної змінної Ch у файл F.
Readln(F, Str)	Читання з файла F рядка Str.
Writeln(F, Str)	Запис рядка Str у файл F.

Eoln(F)	**Перевірка кінця рядка або кінця файла**

Таблиця 21. Процедури і функції для роботи з текстовими файлами

Назва	Призначення
SeekEoln (F)	Функція повертає TRUE, якщо досягнуто кінець рядка або кінець файла, або перед ними стоять лише пропуски і (або) символи табуляції (#9), інакше – FALSE
SeekEof(F)	Функція повертає TRUE, якщо досягнуто кінець файла, або перед ними стоять лише пропуски, ознаки кінців рядків і (або) символи табуляції.

Для роботи з текстовим файлом необхідно:

1) Визначити файлову змінну (змінну логічного файла):

Var F : Text;

З текстового файла можна зчитувати інформацію і записувати інформацію у файл. Нові дані завжди записуються в кінець файла.

2) Після визначення файлової змінної необхідно пов'язати її з іменем фізичного файла на диску. Для цього призначена процедура

Assign(Var Файлова_змінна : Файловий_тип; Ім'я_файла : String);

Приклад. Assign(f, 'ABC.txt'); Assign(f, 'c:\DOC\klm.doc');

Readln(Filname); Assign(f, Filname);

3) Після пов'язування файлової змінної з фізичним файлом необхідно відкрити файл, тобто підготувати його для опрацювання. Текстовий файл можна відкривати трьома способами:

а) Reset(Var Файлова_змінна : Text); – відкриває текстовий файл для зчитування з нього інформації;

б) Rewrite(Var Файлова_змінна : Text); – відкриває файл для запису. Ця процедура служить для створення нового файлу. Якщо файл з таким ім'ям вже існує, дані з нього будуть знищені;

в) Append(Var Файлова_змінна : Text); – відкриває текстовий файл для дописування в нього інформації.

Приклад. Rewrite(f); Reset(f); Append(f);

4) Для зчитування даних з текстового файла використовуються процедури:

Read(Var Файлова_змінна: Text; Змінна1 [, Змінна2,…, ЗміннаN]: Char); – читання з файла даних символьного типу;

Readln(Var Файлова_змінна: Text; Змінна1: String); – читання з файла рядкових даних;

Readln(Var Файлова_змінна: Text); – перехід на новий рядок у файлі.

5) Для запису даних у текстовий файл використовуються процедури:

Write(Var Файлова_змінна: Text; Змінна1 [, Змінна2,…, ЗміннаN]: Char); – запис до файла даних символьного типу;

Writeln(Var Файлова_змінна: Text; Змінна1: String); – запис до файла рядкових даних;

Writeln(Var Файлова_змінна: Text); – запис у файл порожнього рядка.
6) Для визначення кінця рядка у файлі і кінця файла призначені функції:
EOLn(Var Файлова_змінна: Text): Boolean; – повертає TRUE, якщо досягнуто кінець рядка або кінець файла, інакше – FALSE
EOF(Var Файлова_змінна: Text): Boolean; – повертає TRUE, якщо досягнуто кінець файла, FALSE в інших випадках.
7) Після опрацювання файла його необхідно закрити. Для цього призначена процедура
Close(Var Файлова_змінна: Text);

Загальна схема створення текстового файла:
1) оголосити файлову змінну як текстову;
2) надати файлу ім'я (Assign);
3) відкрити файл для запису (Rewrite);
4) підготувати рядок для запису (наприклад, ввести з клавіатури);
5) записати рядок у файл (Writeln);
6) закрити файл (Close).
Пункти 4) і 5) повторюються необхідну кількість разів.

Приклад. Створити текстовий файл з довільним іменем і типом на диску C:. Файл містить рядки довжиною не більш 60 символів. При введенні рядка 'zzz', припинити запис рядків у файл.

```
Type Str =  String[60];
Var  TexFile : Text;
       FileName : String[12];
        S: Str;
Procedure StvorTFile(Var TexFile: Text);
Begin
   Write('Введіть ім'я файла, що створюється: ');
   Readln(FileName);
   Assign(TexFile, FileName);
   Rewrite(TexFile);
   Write('Вводіть рядки файла. zzz  – закінчення введення ');
      Readln(S);
        While S <> 'zzz' do
           Begin
             Writeln(TexFile, S);
              Readln(S);
            End;
       Close(TexFile)
End;
```

Схема опрацювання текстового файла:
1) надати файлу ім'я (Assign);

2) відкрити файл для читання (Reset);
3) прочитати елемент файла (Readln);
4) опрацювати елемент (наприклад, надрукувати);
5) закрити файл (Close).
Пункти 3) і 4) повторюються необхідну кількість разів.

Приклад. Вивести на екран всі елементи файла, створеного у попередньому прикладі.

```
Procedure OprTFile(Var TexFile: Text);
Begin
  Write(‘Введіть ім’я файла для опрацювання:’);
   Readln(FileName);
   Assign(TexFile, FileName);
   Reset(TexFile);
   While not EOF(TexFile) do
      Begin
        Readln(TexFile, S);
       Writeln(S)
      End;
    Close(TexFile)
End;
```

Коректування текстового файла полягає у розширенні його новими елементами. Схема коректування текстового файла:
1) надати файлу ім’я (Assign);
2) відкрити файл для внесення нових елементів (Append);
3) записати новий елемент (Writeln);
4) закрити файл (Close).
Пункт 3) повторюється необхідну кількість разів.

Приклад. Дописати до створеного файла нові рядки. Якщо значення рядка, що вводиться, дорівнює ‘zzz’, припинити введення.

```
Procedure RozshTFile(Var TexFile: Text);
Begin
   Write(‘Введіть ім’я файла для опрацювання:’);
   Readln(FileName);
   Assign(TexFile, FileName);
   Append(TexFile);
   Write('Вводіть рядки.  zzz – закінчення введення ');
    Readln(S);
      While S <> ‘zzz’ do
         Begin
           Writeln(TexFile, S);
            Readln(S)
          End;
```

```
   Close(TexFile)
End;
```

Приклад. Дано текстовий файл. Рядки файла, що мають максимальну довжину, вивести на екран і записати до іншого файла.

```
Uses Crt;
Var  f1, f2 : Text;
       FN1, FN2 : String[12];
       S: String[60];
       l, max : Byte;
Begin
  StvorTFile(f1);
  Write('Введіть ім'я файла для опрацювання:');
   Readln(FN1);
   Assign(f1, FN1);
   Reset(f1);
   Write('Введіть ім'я нового файла:');
   Readln(FN2);
   Assign(f2, FN2);
    Rewrite(f2);
    {пошук максимальної довжини рядка}
max := 0;
   While not (EOF(f1)) do
        Begin
          Readln(f1, S);
          l := Length(S);
          if l > max then max := l
         End;
    Close(f1);
    Writeln('Максимальна довжина рядка = ', max);
    Writeln('Рядки, що мають максимальну довжину:');
      Reset(f1);
     While not (EOF(f1)) do
       Begin
          Readln(f1, S);
          l:=Length(S);
          if l = max then
             Begin
                Writeln(S);
                Writeln(f2, S);
             End;
       End;
     Close(f1); Close(f2);
End.
```

Приклад. Дано текстовий файл data1.txt. Деякі зі слів, записаних у нього, містять знаки перенесення. Переформатувати файл, вилучивши з нього знаки перенесення і об'єднавши відповідні частини слів. Результат записати у новий файл data2.txt.

(Наприклад, *Сьогодні по-* перетворити у *Сьогодні понеділок*)
неділок

{У кожному рядку, що зчитується з файла, перевіряється останній символ. Якщо це знак перенесення, то читається наступний рядок до першого пропуску, цей "хвіст" слова дописується до попереднього рядка, а з поточного вилучається}

```
Var  TFil1, Tfil2 : Text;
       St, St1 : String;
       i : Byte;
Begin
  Assign(Tfil1, 'data1.txt');
  Assign(Tfil2, 'data2.txt');
  Reset(Tfil1); Rewrite(Tfil2);
  Readln(Tfil1, St); {читання першого рядка}
  While not (EOF(Tfil1)) do
     Begin
       Readln(Tfil1, St1); {читання наступного рядка}
       if St[Length(St)] = '– ' {останній символ – знак перенесення}
          then
             Begin
               Delete(St, Length(St),1);{вилучення знака перенесення}
               i :=1;
               While St1[i] <> ' ' do i := i + 1; {знаходження "хвоста" слова у
наступному рядку}
               St := St + Copy(St1, 1, i); {дописування "хвоста" до попереднього
рядка}
               Delete(St1, 1, i)  {вилучення "хвоста" з наступного рядка}
             End;
         Writeln(Tfil2, St); {запис виправленого рядка у новий файл}
        St := St1            {переприсвоювання рядків}
     End;
   Writeln(Tfil2, St);  {запис останнього рядка у новий файл}
  Close(Tfil1);  Close(Tfil2)
End.
```

Опрацювання файлових помилок

При виникненні у роботі програми помилки введення–виведення можливі дві ситуації:

1) програма переривається свою роботу;
2) опрацювання помилки перекладається на програміста.

Управління опрацюванням помилок здійснюється за допомогою директив компіляції:

{$I+}– режим перевірки правильності введення–виведення включений. При включеній перевірці будь-яка файлова помилка вважатиметься фатальною і перерве роботу програми.

{$I–} – режим перевірки правильності введення-виведення виключений. В цьому режимі при виникненні помилки програма не перерветься, а продовжить роботу з наступного оператора. Про виникнення помилки можна дізнатися за допомогою функції

IOResult: Integer;

Якщо файлових помилок не виникло, ця функція повертає 0, якщо виникла помилка – повертає код помилки. Після виникнення помилки і до моменту виклику функції IOResult всі файлові операції ігноруються. Використовувати функцію можна лише один раз після кожної операції введення або виведення, оскільки вона обнулює своє значення при кожному виклику.

За замовчуванням діє режим $I+ . Цей ключ компіляції має локальну сферу впливу. Можна багаторазово включати і виключати режим, вставляючи у текст програми конструкції {$I+} і {$I–}, створюючи області з контролем введення–виведення і без нього.

Приклад.

```
…
{$I–}                    {відключення режиму перевірки введення–виведення}
Assign(f, filname);
Reset(f);
if IOResult <>0 then          {якщо файл не може бути відкритий, вивести
повідомлення}
  Writeln('Такого файла не існує')
else                     {інакше (код=0) можна працювати з файлом}
 Begin

  Read(f, …);
   …
  Close(f)
 End;
{SI+}                    {відновлення перевірки}
…
```

Робота з командним рядком

При написанні програм опрацювання файлів часто буває зручно компілювати їх як виконувані файли і запускати з командного рядка, вказуючи імена файлів, що опрацьовуються, як параметри.

Функції ParamCount і ParamStr необхідні для роботи самостійних програм (EXE–файлів) з параметрами командного рядка. Під параметрами командного рядка розуміється набір параметрів, розділених пропусками (або знаками табуляції), який вказується після імені виконуваного файла в рядку MS–DOS, наприклад:

C:\> FORMAT A: /S

Тут FORMAT – ім'я виконуваного файла, A: – перший параметр, /S – другий параметр.

Параметр може містити що завгодно, крім пропусків і табуляцій. Якщо після імені файла в рядку MS–DOS немає параметрів, то ParamCount повертає значення 0, інакше – кількість параметрів у рядку, розділених пропусками або табуляцією. Це можна використовувати для зупинки програм, яким при запуску не були передані параметри–аргументи.

Приклад.

```
…
if  ParamCount = 0 then
  Begin
    Writeln('Задайте параметри!');
     Halt
  End;
…
```

Після того як визначена кількість параметрів, можна вибрати будь–який з них, використовуючи функцію ParanStr(N), яка повертає у вигляді рядка параметр з номером N.

Приклад. Нехай запуск програми має вигляд

C:\TURBO\PAS> Myprog abc /z /c/d 123

Тут C:\TURBO\PAS> – запрошення MS–DOS, Myprog – ім'я програми, abc, /z, /c/d, 123 – параметри. В цьому випадку функція ParamCount поверне значення 4, а ParamStr поверне такі значення типу String:

ParamStr(0) = 'C:\TURBO\PAS\MYPROG.EXE'
ParamStr(1) = 'abc'
ParamStr(2) = '/z'
ParamStr(3) = '/c/d'
ParamStr(4) = '123'
ParamStr(5) = ''
ParamStr(n)= '' , (n>5)

Слід відмітити, що 1) завжди має смисл виклик ParanStr(0), який повертає повне ім'я запущеної програми; 2) цифри у рядку параметрів трактуються як рядок, тому їх треба перетворювати процедурою Val. При запуску з середовища програмування ParamStr(0) поверне повне ім'я файла TURBO.EXE (з указанням шляху до нього).

Щоб забезпечити зручність користування і "дружність" програми, слід робити так, щоб при відсутності параметрів при запуску програма запитувала їх з екрану. Це легко зробити за допомогою розглянутих функцій

Приклад. Нехай програма запитує ім'я файла даних, і якщо користувач не вводить його, приймає деяке значення імені за замовчуванням.

```
Var S : String;
Begin
if ParamCount > 0                {чи є параметри?}
   then s:=ParamStr(1)           {так}
   else Begin                          {ні}
         Write('[Default.dat]');
         Readln(s);
         if s='' then s:='Default.dat'
        End;
{ …виконання програми – опрацювання файла s}
End.
```

Типізовані файли

Типізований файл являє собою скінченний упорядкований набір елементів певного типу. Елементи файла пронумеровані, нумерація починається з нуля.

Формат опису:

Type Ім'я_ типу = File of тип_елементів;

Var Ім'я_змінної : Ім'я_типу;

Тип_елементів (базовий тип) може бути довільний, крім файлового.

Типізовані файли є файлами довільного доступу. Файли довільного доступу створюються для розв'язування задач, які вимагають оперативного доступу до інформації. Послідовний доступ до елементів типізованих файлів можливий, але неефективний.

Для роботи з типізованим файлом необхідно:

1) Визначити файлову змінну (змінну логічного файла):

Var F : File of …;

2) Після визначення файлової змінної необхідно пов'язати її з іменем фізичного файла на диску. Для цього, як і у випадку текстових файлів, призначена процедура

Assign(Var Файлова_змінна : Файловий_тип; Ім'я_файла : String);

Приклад. Assign(f, 'ABC.dat'); Assign(f, 'c:\DOC\klm');

Readln(Filname); Assign(f, Filname);

3) Після пов'язування файлової змінної з фізичним файлом необхідно відкрити файл, тобто підготувати його для опрацювання. Типізований файл можна відкривати двома способами:

а) Reset(Var Файлова_змінна : Файловий_тип); - відкриває існуючий файл для зчитування, зміни інформації і додавання нових даних;

б) Rewrite(Var Файлова_змінна : Файловий тип); - відкриває файл для запису (аналогічно текстовим файлам). Ця процедура служить для створення нового файлу. Якщо файл з таким ім'ям вже існує, дані з нього будуть знищені.

Як вказувалося вище, доступ до елементів файла здійснюється через так званий файловий покажчик – буферну змінну, яка на відміну від усіх інших змінних не може брати участь у виразах і операторах присвоювання. Відразу після відкриття файла покажчик вказує на початок файла, тобто елемент з номером нуль. В дійсності нумеруються не самі елементи файла, а межі між ними: перед першим елементом – номер 0, між першим і другим – 1 і т.д.

Отже, реальний номер елемента завжди на 1 більше номера межі перед ним.

4) Для зчитування даних з типізованого файла використовується процедура:

Read(Var Файл_змінна: Файл_тип, Змінна1 [, Зм2,…, ЗмN]: Тип_елемент); – читання з файла даних, тип яких є базовим типом файла;

Після операції зчитування файловий покажчик встановлюється за останнім зчитаним елементом і вказує на наступний елемент, готовий до опрацювання.

5) Для запису даних у типізований файл використовується процедура:

Write(Var Файл_змінна: Файл_тип, Зм1 [, Зм2,…, ЗмN]: Тип_елемент); – запис до файла даних базового типу;
Після операції запису файловий покажчик встановлюється за останнім записаним елементом і вказує на наступний елемент, готовий до опрацювання.

Зчитування і запис даних здійснюються у позицію, вказану файловим покажчиком.

6) Для з'ясування кількості елементів у файлі використовується функція:
FileSize(Var Файл_змінна: Файл_тип):LongInt;– повертає реальну кількість елементів у відкритому файлі. Для порожнього файла значення функції =0;
7) Для з'ясування поточної позиції файлового покажчика використовується функція
FilePos(Var Файл_змінна: Файл_тип): LongInt; – повертає поточну позицію файлового покажчика у відкритому файлі. Враховуючи специфічну нумерацію елементів файла, слід зробити такі зауваження щодо роботи функції FilePos:
1. На початку файла FilePos повертає значення 0.
2. В самому кінці файла FilePos повертає число, рівне реальній кількості елементів у файлі, тобто FileSize
3. В решті випадків функція FilePos повертає значення, на 1 менше реального номера елемента, який готовий для читання або створення.
8) Для реалізації довільного доступу до відкритого файла передбачена процедура:
Seek(Var Файл_змінна : Файл_тип, N: LongInt); – встановлює файловий покажчик у необхідну позицію. В параметрі N має бути заданий номер умовної межі між елементами.

Приклад. Для роботи з елементом файла f, який має реальний номер 3, треба задати позицію на межі перед ним, тобто виконати оператор Seek(f, 2).
Для читання або запису першого елемента треба задати Seek(f,0).
Для встановлення покажчика за останнім елементом – Seek(f, FileSize(f)).
Для доступу до останнього елемента файла – Seek(f, FileSize(f)–1).
9) Для визначення кінця файла призначена функція:
EOF(Var Файл_змінна: Файл_тип): Boolean; – повертає TRUE, якщо досягнуто кінець файла, FALSE в інших випадках (аналогічно текстовим файлам).
10) Після опрацювання файла його необхідно закрити. Для цього, як і у випадку текстових файлів, призначена процедура
Close(Var Файл_змінна: Файл_тип);

Схема створення типізованого файла:
1) оголосити файлову змінну потрібного типу;
2) надати файлу ім'я (Assign);
3) відкрити файл для запису (Rewrite);
4) підготувати дані для запису (наприклад, ввести з клавіатури);
5) записати дані у файл (Write);
6) закрити файл (Close).

Пункти 4) і 5) повторюються необхідну кількість разів.

Схема опрацювання типізованого файла:
1) надати файлу ім’я (Assign);
2) відкрити файл (Reset);
3) встановити покажчик на потрібний елемент (Seek);
4) зчитати потрібний елемент (Read);
5) виконати опрацювання зчитаної інформації;
6) закрити файл (процедура Close).
Пункти 3) – 5) повторюються необхідну кількість разів.

Коректування типізованого файла полягає у зміні значень елементів або поповненні файла новими елементами.

Схема коректування типізованого файла:
1) надати ім’я файлу (Assign);
2) відкрити файл, що коректується (Reset);
3) встановити покажчик файла на елемент, що коректується (Seek);
4) зчитат елемент, що коректується (Read);
5) відкоректувати елемент файла;
6) встановити покажчик файла на відкоректований елемент (Seek);
7) записати відкоректований елемент (Write);
8) закрити файл (Close).
Пункти 3) – 5) повторюються необхідну кількість разів.

Повторне встановлення покажчика необхідне, оскільки після зчитування елемента покажчик переміститься до наступного елемента. Тому, щоб відкоректований елемент зберегти на старому місці, покажчик слід перемістити на одну позицію назад.

Приклад. Створити файл із заданої користувачем кількості дійсних чисел.

```
Uses Crt;
Var  Fil : File of Real;
       FilName: String[12];
       comp :Real;
        i, k : Integer;
Begin
   Write('Введіть ім’я файла, що створюється: ');
   Readln(FilName);
   Assign(Fil, FilName);
   Rewrite(Fil);
   Write('Введіть кількість елементів  файла: ');
   Readln(k);
      for i :=1 to k  do
         Begin
           Write('Введіть дійсне число: '); Readln(comp);
           Write(Fil, comp);
         End;
```

```
    Close(Fil)
End.
```

Приклад. Дано файл дійсних чисел. Замінити елементи цього файла з парними номерами елементом з максимальним значенням.

```
Uses Crt;
Var  Fil : File of Real;
       FilName: String[12];
       comp :Real;
        i, k, max  : Integer;
Begin
   Write('Введіть ім’я файла для опрацювання: ');
   Readln(FilName);
   Assign(Fil, FilName);
   Reset(Fil);
   Read(Fil, max); {читання першого елемента, надання  його значення змінній max}
   {перше проходження файла – знаходження максимального значення елемента}
   for i :=1 to FileSize(Fil) - 1  do
         Begin
           Read(Fil, comp);
           if comp >= max then max :=comp
         End;
   {друге проходження файла по елементам з парними номерами – }
   {заміна їх значень на значення максимального елемента}
   for i :=0 to FileSize(Fil) div 2  + 1  do
         Begin
           Seek(Fil, i*2);
           Write(Fil, max)
         End;
    Close(Fil)
End.
```

Приклад. Дано файл F, елементи якого – цілі числа. Отримати файл G з елементів файла F, які мають парні номери, кратні 3, але не кратні 5.

```
Uses Crt;
Var  FilOld, FilNew : File of Integer;
       NamOld, NamNew: String[12];
       comp :Real;
        i  : Integer;
Begin
   Write('Введіть ім’я старого файла: ');
   Readln(NamOld);
   Assign(FilOld, NamOld);
   Reset(FilOld); {відкриття старого файла для читання}
```

```
   Write('Введіть ім’я нового файла: ');
   Readln(NamNew);
   Assign(FilNew, NamNew);
   Rewrite(FilNew); {відкриття нового файла для запису}
   i :=0;  {найменший парний номер елемента}
   While not EOF(FilOld) do
      Begin
        Seek(FilOld, i); Read(FilOld, comp);
          if (comp mod 3 = 0) and (comp mod 5 <> 0)
            then Write(FilNew, comp); {поповнення нового файла}
          i := i +2  {проходження по парним номерам елементів файла}
       End;
   Close(FilOld); Close(FilNew);
{виведення вмісту нового файла}
Reset(FilNew);
While not EOF(FilNew) do
   Begin
      Read(FilNew, comp);
      Write(comp:3)
    End;
Close(FilNew)
End.
```

Нетипізовані файли

У TP вводиться особливий файловий тип, який фактично є узагальненим файловим типом. Нетипізований файл, як і типізований, складається з машинних представлень даних. Проте типізовані файли працюють з даними заздалегідь оголошеного типу, а нетипізовані – з довільними наборами байтів незалежно від їх структури і природи.

Нетипізований файл – потужний засіб роботи з файлами, оскільки дозволяє маніпулювати з даними, не замислюючись про їх тип.

Введення–виведення у нетипізовані файли здійснюється спеціальними процедурами BlockRead і BlockWrite. Крім того, розширюється синтаксис процедур Reset і Rewrite. В іншому принципи роботи лишаються такими самими, як і з типізованими файлами.

Перед використанням файлова змінна повинна бути описана.

Формат опису:

Var Ім'я_змінної : File;

Процедури Assign і Close зберігають свій синтаксис і зміст.

Процедури Reset і Rewrite можуть мати розширений синтаксис:

Reset([Var Файл_змінна : File [, Розмір_буфера : Word]);

Rewrite([Var Файл_змінна : File [, Розмір_буфера: Word]);

Розмір_буфера – розмір елемента (буфера файла) у байтах (за замовчуванням – 128 байт).

Параметр Розмір_буфера задає кількість байтів, які зчитуються з файла за одне звертання до нього або записуються у нього.Чим більше це значення, тим швидше відбувається обмін даними між носієм файла (як правило, диском) і оперативною пам'яттю ПЕОМ. Але тим більше і витрати пам'яті.

Мінімальний блок, який може бути записаний або прочитаний з файла, – 1 байт. Максимальний розмір блока не може перевищувати 64K.

Для зчитування і запису даних у нетипізований файл використовуються процедури:

BlockRead(Var Файл_змінна : File; Var BufIn; N : Word [; Var NIn : Word]);

BlockWrite(Var Файл_змінна : File; Var BufOut; N : Word [; Var NOut : Word]);

Ці процедури здійснюють читання в змінну BufIn або запис зі змінної BufOut N блоків, які складаються з кількості байтів, визначеної у процедурі відкриття файла. Повинна виконуватись умова N * Розмір_буфера < 64K. Необов'язковий параметр NIn повертає кількість блоків (буферів), зчитану поточною операцією BlockRead. Аналогічний параметр Nout після кожної операції запису показує кількість блоків (буферів), записану в даний файл операцією BlockWrite.

Якщо операції запису або читання пройшли успішно, то значення NIn і NOut будуть дорівнювати відповідним значенням параметра N. Але якщо відбувся збій при введенні–виведенні, то параметри NIn і NOut будуть містити цілу кількість успішно перенесених блоків.

Приклад. Копіювання файла. Програма має бути попередньо відкомпільована на диск. Імена файлів передаються через командний рядок.

```
Const BufSize=4096;
Var FIn, FOut: File;
      buf: Array[1..BufSize] of Byte;
      N, Nr, Nw : LongInt;
 Begin
   if ParamCount<>2 then
     Begin
       Writeln('Потрібні 2 імені файлів!');
       Halt;
{$I–}
Assign(FIn, ParanStr(1));
Reset(FIn,1);
if IOResult <>0 then
   Begin
     Writeln('Не вдалося створити файл ', ParamStr(2));
      Halt
   End;
N:=BufSize;
repeat
   BlockRead(FIn, Buf, N, Nr);
   BlockWrite(FOut, Buf, N, Nw)
until (Nr=0) or (Nw<>Nr);
Close(FIn);
Close(FOut)
End.
```

Питання для самоконтролю.

1. *Як розуміється файл у системі Турбо Паскаль?*
2. *Як оголошуються файли у Паскаль-програмах? Наведіть приклади.*
3. *Які файлові типи підтримуються системою ТР? Які особливості цих типів?*
4. *Які особливості використання файлових змінних у Паскаль – програмах?*
5. *Як здійснюється доступ до окремих елементів файла? Як розрізняються файли за способом доступу до їх елементів?*
6. *Чи повинні всі елементи файла бути одного типу ?*
7. *В яке місце файла можна додавати нові елементи: на початок, в кінець, в будь-яке місце, нікуди?*
8. *Як оголошуються текстові файли у Паскаль-програмах? Наведіть приклади.*
9. *Які особливості мають текстові файли?*
10. *Як здійснюється доступ до елементів текстового файла?*
11. *В яке місце текстового файла можна додавати нові елементи: на початок, в кінець, куди завгодно, нікуди ?*

12. *Значення яких елементів текстового файла можна змінювати: тільки першого, тільки останнього, яких завгодно, ніяких ?*
13. *Значення яких елементів текстового файла можна вилучати?*
14. *Чи можна порівнювати текстові файли ?*
15. *Чи можна присвоювати один текстовий файл іншому ?*
16. *Як оголошуються типізовані файли у Паскаль-програмах? Наведіть приклади.*
17. *Чи можна, зчитавши з файла п'ятий елемент, потім зчитати другий?*
18. *Чи можна одночасно читати інформацію з файла і записувати в нього нову інформацію?*
19. *Як здійснюється створення нового файлу? Наведіть приклади.*
20. *Як здійснюється опрацювання файла з послідовним доступом? Наведіть приклади.*
21. *Як здійснюється опрацювання файла з довільним доступом? Наведіть приклади.*

Поняття про структурне програмування. Бібліотеки підпрограм. Модулі у системі Турбо - Паскаль

Структурне програмування – система принципів проектування, кодування, тестування і документування програм. Мета застосування цих принципів – створення великих програм і можливість розбивати їх на незалежні частини.

Основні методи структурного програмування:

а) використання трьох базових алгоритмічних конструкцій: слідування, розгалуження, повторення;

б) метод покрокової розробки програм (покрокової деталізації, програмування методом "зверху–вниз"). Метод полягає в тому, що задача розбивається на окремі частини, кожна з яких поступово уточнюється. При цьому відбувається перехід від абстрактного виконавця, що здатен розв'язати задачу за один крок, до коду, записаного мовою програмування.

Існує метод програмування "знизу–вгору", в якому відбувається протилежний перехід від виконавця до виконавця.

На практиці використовуються обидва підходи, причому на етапі створення програм переважно використовується метод "зверху–вниз", а при оновленні готових програм – "знизу–вгору".

Приклад. Обчислити біноміальні коефіцієнти двочлена $(a+b)^n$ при заданому n.

$$(a+b)^n = C_n^0 a^n b^0 + C_n^1 a^{n-1} b^1 + C_n^2 a^{n-2} b^2 + \ldots + C_n^n a^0 b^n.$$

$$C_n^k = \frac{n!}{k!(n-k)!}.$$

Застосуємо метод покрокової деталізації програми, переходячи від найбільш загальної схеми розв'язування задачі до програми мовою ТР.

```
Begin
      розв'язати задачу
    End.
Begin
      ввести дані;
      опрацювати дані;
      вивести результат
    End.
Var n: Integer;
Begin
  Input(n);          {Введення степеня двочлена}
  Binom(n)           {Обчислення біноміальних коефіцієнтів}
End.
Procedure Input(Var n: Integer);
Begin
      Write('Введіть степінь двочлена: ');
       Readln(n)
     End;
```

```
      Procedure Binom(k:Integer);
      Var i: Integer;
      Begin
        for i:=0 to k do
         Begin
             Write(C(k,i));
             Write('a^', k–i);
             Write('b^',i);
              if i<k then Write('+')
          End;
       End;
       Function C(m,n: Integer):Integer;
       Begin
         C:=Round(Fact(n)/(Fact(m)*Fact(n–m));
       End;
       Function Fact(n: Integer): Integer;
       Var i, F:Integer;
       Begin
         F:=1;
         for i:=1 to n do
           F:=F*i;
         Fact:= F
       End;
```

Остаточний вигляд програми:

```
Var n: Integer;
Function Fact(n: Integer): Integer;
Var i, F: Integer;
Begin
   F:=1;
   for i:=1 to n do
     F:=F*i;
     Fact:= F
 End;
Function C(m,n: Integer):Integer;
Begin
   C:=Round(Fact(n)/(Fact(m)*Fact(n–m));
End;
Procedure Binom(k: Integer);
Var i: Integer;
Begin
  for i:=0 to k do
    Begin
      Write(C(k,i));
      Write('a^', k–i);
      Write('b^',i);
```

```
      if i<k then Write('+')
    End;
 End;
Procedure Input(Var n: Integer);
Begin
   Write('Введіть степінь двочлена: ');
    Readln(n)
End;
{основна програма}
Begin
   Input(n);
   Binom(n)
End.
```

При програмуванні "знизу–вгору" доцільно залучати до програми процедури і функції, заздалегідь визначені у модулях ТР.

Модуль у ТР (UNIT) – це спеціально оформлена бібліотека визначень типів, констант, змінних, процедур і функцій. Модулі зберігаються у файлах з розширенням .TPU (Turbo Pascal Unit). Для підключення модуля до програми (чи іншого модуля) слід вказати директиву:

Uses Ім’я_модуля1 [, Ім’я_модуля2, …];

(тут Uses (використовує) – службове слово)

і далі використовувати вміст підключених бібліотек.

Переваги використання модулів:

1) можливість створення власних бібліотек процедур і функцій, які можуть підключатись до різних програм, не вимагаючи переробки останніх;
2) можливість створення програм великих розмірів (ні програма, ні модуль не можуть створити виконуваний код об’ємом більше 64К, якщо вони не використовують інші модулі).

Структура модуля:

1) заголовок модуля (UNIT Ім’я);
2) блок оголошень або інтерфейс (INTERFACE);
3) блок реалізації (IMPLEMENTATION)
4) блок ініціалізації (Begin…End).

Приклад. Порожній модуль має вигляд

```
Unit Empty;
Interface
Implementation
End.
```

1) Заголовок модуля – ім’я, за яким модуль буде підключатися до інших програм. Воно повинно бути унікальним і відповідати імені файла, в якому зберігається вихідний текст модуля.
2) Інтерфейсна частина містить описи типів, констант і змінних, які привноситимуться в програми при підключенні модуля. Тут також

описуються заголовки процедур і функцій, які і складають бібліотеку підпрограм.

3) В розділі реалізації описуються всі процедури і функції, заголовки яких вказані в інтерфейсній частині. При цьому заголовок може не містити параметрів. В розділі реалізації можуть бути описані свої типи, константи і змінні, які недоступні за межами модуля.
4) Розділ ініціалізації має вигляд

```
Begin
  оператори
End.
```

Якщо він відсутній, пишеться лише слово End. В цьому розділі описуються дії, які мають виконуватись перед виконанням програми, до якої підключено модуль. Найчастіше це надання змінним початкових значень. Більшість модулів не має блоку ініціалізації.

Приклад. Опишемо модуль, який містить функцію обчислення факторіалу.

```
Unit Math;
Interface
Function Fact(a: Integer): Integer;
Implementation
Function Fact[(a: Integer): Integer];
Var i, F : Integer;
Begin
  F:=1;
  for i:-=1 to a do F:=F*i;
  Fact:=F;
End;          {implementation}
End.          {initialization}
```

Щоб розробленим модулем можна було користуватися в інших програмах, слід відкомпілювати його на диск, встановивши у середовищі програмування опцію

Compile – Destination – Disk.

В системі TP визначено ряд стандартних модулів. Всі вони, крім System, підключаються до програми директивою Uses:

System – автоматично підключається до будь–якої програми. Включає стандартні процедури і функції Паскаля (Ln, Exp, Sin, Cos, …);

Dos – призначений для використання у програмі функцій операційної системи;

Crt – містить процедури і функції для роботи з текстовим екраном і клавіатурою;

Graph – призначений для роботи з графічним екраном;

Printer – призначений для організації виведення інформації на принтер.

Питання для самоконтролю.

1. *Що таке структурне програмування? Які основні методи структурного програмування?*
2. *У чому полягає метод покрокової деталізації? Наведіть приклади.*

3. *Що таке модуль у ТР? Як підключаються модулі до Паскаль-програм?*
4. *Яку структуру має модуль у ТР?*
5. *Як створити модуль у ТР-системі?*
6. *Опишіть стандартні модулі ТР.*

Вказівники у системі Турбо-Паскаль

При розв'язуванні практичних задач за допомогою комп'ютера виникають дві основні проблеми: проблема швидкодії і проблема пам'яті. Перша проблема розв'язується за допомогою використання ефективних алгоритмів, друга – зокрема за допомогою введення динамічних змінних, або змінних–вказівників.

Розглядувані раніше змінні є **статичними**. При їх описі виділяється певний об'єм пам'яті, який не змінюється протягом виконання програми. Так, всі глобальні змінні і типізовані константи, оголошені в TP–програмі, розміщуються в неперервній області оперативної пам'яті, яка називається сегментом даних і займає 64К. Локальні змінні і параметри процедур і функцій розміщуються у стеку (64К). Ця пам'ять виділяється при компіляції програми. Решта вільної пам'яті називається вільно розподіленою пам'яттю або "кучею"

Динамічні змінні характерні тим, що їх розмір заздалегідь невідомий, пам'ять під них виділяється в "кучі" під час виконання програми, а після їх використання ця пам'ять звільняється для інших потреб. Робота з "кучею" відбувається через вказівники.

Вказівник – це змінна або константа, в якій міститься адреса певної ділянки пам'яті.

Розмір вказівника – 4 байти. Вказівники бувають типізовані і нетипізовані.

Якщо кажуть, що змінна X вказує на змінну Y, це означає, що змінна X містить адресу змінної Y.

Типізований вказівник вказує на ділянку пам'яті певного типу.

Формат опису типізованого вказівника:

Var Ідентифікатор: ^Ім'я_базового_типу;

Базовим може бути довільний тип мови TP.

Приклад. Var pr :^Real; – після такого опису pr може вказувати на ділянку пам'яті, в якій зберігається дійсне число.

Перед тим, як працювати з потрібною ділянкою пам'яті, треба виділити цю пам'ять у "кучі". Для виділення пам'яті, на яку вказуватиме типізований вказівник, використовується процедура

New(Var p: Типізований_вказівник);

Приклад. new(pr);

Після виконання цієї процедури створюється нова змінна дійсного типу, адреса якої зберігається в pr. Ця змінна має ім'я pr^. Спочатку вона не має значення. Надати значення змінній можна будь–яким зі способів, визначених у TP, наприклад, за допомогою оператора присвоювання:

pr^:=3.2;

Зі змінною pr^ можна працювати, як і з будь–якою дійсною змінною.

Щоб показати, що вказівник не вказує на жодну реальну ділянку пам'яті, йому можна присвоїти значення Nil. Це зарезервована константа TP.

pr:=Nil;

Для звільнення пам'яті, на яку вказує типізований вказівник, використовується процедура

Dispose(Var p: Типізований_вказівник);

Слід розрізняти вказівник і вміст пам'яті, на яку він вказує.

Приклад. Розглянемо дві змінні–вказівники:

Var i, j : ^Integer;

На початку роботи програми **i** і **j** ні на що не вказують:

i ⊗

j ⊗

Далі в **i** і **j** записуються адреси, на які вони будуть вказувати:

New(i); **i** ○→○ **i^**

New(j); **j** ○→○ **j^**

Після цього цілим змінним **i^** і **j^** можна надати значення:

i^ := 8; **i** ○→∩ **i^**

j^ := 6; **j** ○→⊕ **j^**

Якщо змінній **i^** присвоїти значення **j^**, результат виглядатиме так:

i^ := j^; **i** ○→⊕ **i^**

j ○→⊕ **j^**

Якщо змінній **i** присвоїти значення **j**, результат буде таким:

i := j; **i** ○↘ ⊕ – це значення недоступне до кінця роботи програми, бо

j ○→⊕ **j^** його адреса загублена

Приклад. Опис вказівників і прості дії з ними.

```
Type
Student = Record
            ↘Name : String;
              Group : Byte
            End;
Mas = Array[1..10] of Integer;
Var
pSt : ^Student;
pMas : ^Mas;i : Byte;
Begin
  new(pSt); new(pMas);
  pSt^.Name := 'Петренко';
  pSt^.Group := 55;
  for i := 1 to 10 do pMas^[i] := i;
  …
  Dispose(pSt); Dispose(pMas);
End.
```

Нетипізовані вказівники використовуються тоді, коли тип пам'яті, на яку буде здійснюватись посилання, не важливий. Для опису таких вказівників

призначений тип Pointer. Його значення, як і значення типізованих вказівників, займають 4 байти.

Приклад. Var p, p1 : Pointer;

Для виділення пам'яті, на яку вказує нетипізований вказівник, використовується процедура

GetMem(Var p : Pointer; Size : Word);

Тут p – нетипізований вказівник, Size – кількість байтів, яку треба виділити у "кучі".

За цією процедурою у "кучі" виділяється Size байтів пам'яті, і адреса першого байта поміщується у змінну p.

Для звільнення пам'яті, пов'язаної з нетипізованим вказівником, використовується процедура

FreeMem(Var p : Pointer; Size : Word);

Тут параметри p і Size мають таке ж значення, як у процедурі GetMem. При використанні цієї процедури треба слідкувати, щоб звільнялося стільки пам'яті, скільки було виділено раніше.

Операції над вказівниками

Вказівники можна присвоювати. Якщо вказівники типізовані, вони повинні належати до одного типу. Нетипізованому вказівнику можна присвоїти значення типізованого, але не навпаки.

Вказівники можна порівнювати. При цьому вказівники вважаються рівними, якщо вказують на одну і ту саму ділянку пам'яті.

Якщо вказівник не вказує на жодну ділянку пам'яті, йому доцільно присвоїти значення Nil.

Функція Addr(x) і оператор @ повертають значення типу Pointer – адресу об'єкта x.

Приклад.

```
Var X : String;
    p, q : Pointer;
…
 p:=Addr(X);
 q:=@X;
 {Тепер p=q, і адреси рівні; вони вказують на один об'єкт}
```

Питання для самоконтролю.

1. *В чому полягає різниця між статичними і динамічними змінними?*
2. *Що називається вказівником? Які існують види вказівників?*
3. *Як відбувається робота з типізованими вказівниками?*
4. *Як відбувається робота з нетипізованими вказівниками?*
5. *Які операції можна виконувати над вказівниками у ТР?*

Динамічні структури даних

Статичні структури даних – такі структури, в яких завжди міститься постійна кількість елементів (наприклад, масив, запис постійної довжини).

Динамічними структурами даних називаються такі структури, які під час виконання програми можуть змінювати свої розміри (наприклад, множина, запис з варіантами).

На основі стандартних типів даних Паскаля можна будувати нові динамічні структури даних. Найпростішими серед них є стек і черга. До більш складних структур відносяться список і дерево.

Стек – це впорядкований набір елементів, з одною точкою доступу, яка називається вершиною стека.

Доступ здійснюється лише до елемента, розташованого у вершині стеку (рис.1):

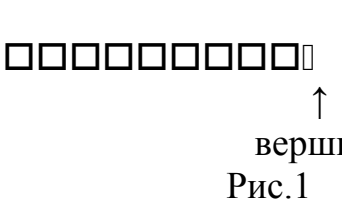

Рис.1

Новий елемент додається у вершину стеку. Вилучається елемент також з вершини стеку. Отже, елемент, який останнім потрапив до стеку, буде вилучений першим. Тому стек називається структурою LIFO (last in – first out) (останній прийшов – перший вийшов).

Черга – це впорядкований набір елементів з двома точками доступу, які називаються початок і кінець (або голова і хвіст) черги. Новий елемент додається в кінець черги, вилучається елемент з початку черги (рис.2)

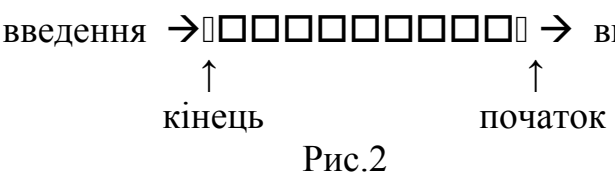

Рис.2

Елемент, який першим потрапив у чергу, першим буде вилучений з неї. Тому чкргу називають структурою FIFO (first in – first out) (першим прийшов – першим вийшов).

Існують різні способи реалізації стеку і черги. Універсальною структурою даних Паскаля, за допомогою якої можна змоделювати будь–яку динамічну структуру, є масив. Розглянемо реалізацію стека і черги за допомогою масивів.

Приклад. Опишемо стек з дійсних чисел з максимальним розміром StackSize.

```
Const StackSize = 15;
Type  Stack = Record
                 Elem: Array[1..StackSize] of Real;
               Top: 0..StackSize
              End;
Var   OneStack : Stack;
```

Кожний елемент масивуOneStack.Elem є елементом власне стека. Оскільки стек може збільшуватися і зменшуватися, потрібно вказувати поточний розмір стека. Для цього призначене поле OneStack.Top. Якщо стек порожній, OneStacl.Top = 0. Якщо стек містить від 1 до StackSize елементів, то OneStack.Top вказує на верхній елемент стеку в межах масиву.

Розглянемо процедуру, яка додає елемент у вершину стека. Параметр NewElem має тип Real, так як описаний стек містить елементи дійсного типу. Оскільки неможливо занести елемент у стек, який вже заповнений, процедура перевіряє, чи не повний стек (обмежений розмір стека є основною незручністю при описі стека за допомогою масиву).

```
Procedure Push(Var NewStack: Stack; NewElem : Real);
Begin
  if  NewStack.Top = StackSize then
    Writeln('Стек заповнений')
  else
    Begin
      NewStack.Top := NewStack.Top + 1;
      NewStack.Elem[NewStack.Top] := NewElem
    End;
End;
```

Розглянемо процедуру, яка вилучає елемент стека з його вершини. Кожного разу, коли викликається ця процедура, вказівник вершини стека зменшується на одиницю, щоб вказувати на нову вершину. Отже, коли зі стека вилучені всі елементи, вказівник набуває значення 0.

```
Procedure Pop(Var OldStack: Stack; Var OldElem: Real);
Begin
  if OldStack.Top = 0 then
    Writeln('Стек порожній')
  else
    Begin
      OldElem := OldStack.Elem[OldStack.Top];
      OldStack.Top := OldStack.top – 1
  End
End.
```

Розглянемо реалізацію черги за допомогою масиву.

```
Const QueueSize = 15;
Type Queue = Record
                  Elem : Array[0..QueueSize] of Real;
                     Front, Rear : 0..QueueSize
                  End;
Var OneQueue : Queue;
```

Відмінності опису черги від опису стека:

1) необхідні вказівники на початок і кінець черги;
2) масив для черги починається з індексу 0, тобто масив містить QueueSize+1 елемент.

Оскільки початок і кінець черги постійно переміщуються в межах масиву, зручно представити масив у вигляді кільця. Щоб обходити масив "по кільцю", використовується операція mod. Якщо Rear<QueueSize–1, він збільшується звичайним способом. Після того як Rear досягає значення QueueSize–1, він перетворюється у 0. Це можна перевірити: (QueueSize–1)+1=QueueSize, а QueueSize mod QueueSize = 0. У даній схемі Elem[QueueSize] не використовується.

Приклад. Розглянемо чергу:(5 4 3 2 1 0) Тут F=0, R=5, QueueSize=6. R+1=6, 6 mod 6=0. Отже, F=R, тобто черга заповнена.

Розглянемо процедуру занесення елемента в чергу.

```
Procedure AddQ(Var NewQueue:Queue; NewElem:Real);
Begin
  With NewQueue do
    Begin
      Rear := (Rear+1) mod QueueSize;
      if Rear = Front then
        Writeln('Черга заповнена')
      else
        Elem[Rear] := NewElem
    End;
End;
```

Розглянемо процедуру вилучення елемента з черги. Як визначити, чи порожня черга? Це визначається аналогічно тому, як перевіряється, чи заповнена черга. Процедура AddQ перед перевіркою збільшує Rear, тим самим вона перевіряє, чи є більше одної вільної позиції між Front і Rear. Якщо є лише одна вільна позиція, черга вважається заповненою.

```
Procedure DeleteQ(Var OldQueue:Queue; Var OldElem:Real);
Begin
  With OldQueue do
    if Rear = Front then
      Writeln('Черга порожня')
    else Begin
      Front := (Front+1) mod QueueSize;
      OldElem := Elem[Front]
        End
End;
```

Найзручніше подавати динамічні структури даних за допомогою вказівників. Розглянемо реалізацію стека. Елементи стека назвемо вузлами; нехай вони будуть даними рядкового типу.

Перед описом стека слід зауважити, що при описі типів вказівників можна посилатися на ті типи, які будуть описані пізніше.

```
Type pNode = ^Node;
       Node = Record
                Txt : String;          {елемент стека}
                Next : pNode  {посилання на наступний елемент}
```

```
                End;
```
Опишемо вершину стека, через яку буде здійснюватись доступ до стека:
```
Var Top : pNode;
```
Якщо треба вказати, що стек порожній, треба присвоїти Top:= Nil;
Розглянемо процедуру занесення даних у стек:
```
Procedure Push(S: String);
Var pN : pNode;
Begin
  New(pN);
  pN^.Txt := S;
  pN^.Next := Top;
  Top := pN
End;
```
Процедура, яка вилучає елемент з вершини стека.
```
Procedure Pop;
Var pN : pNode;
Begin
  if Top <> Nil then
    Begin
      pN := Top^.Next;
      Dispose(Top);
      Top := pN
    End;
End;
```
Процедура, яка друкує дані, що знаходяться у стеку.
```
Procedure Print;
Var pN : pNode;
Begin
  pN := Top;
  while pN <> Nil do
    Begin
      Writeln(pN^.Txt);
       pN := pN^.Next
    End;
End;
```

Списки і дерева

Список – це упорядкований набір елементів, кожний з яких є записом з двох полів. Одне поле містить дані певного типу, інше є вказівником, який вказує на наступний елемент.

Над списками визначені такі основні операції: пошук елемента у списку, вставка елемента у вказане місце списку, вилучення елемента із списку.

Розглянемо опис списку (він аналогічний опису стека):

```
Type pNode = ^Node;
        Node = Record
                    Data : String;          {елемент списку}
                    Next : pNode  {посилання на наступний елемент}
                End;
```

Поточний вказівник списку вказує на вузол, який формується останнім.

Приклад. Розглянемо програму формування списку з указаної кількості елементів.

```
Var list, list0 : pNode;
Procedure FormList(Var list :pNode);
Var i,n :Integer;
Begin
  list:=Nil;
  Writeln('Введіть кількість елементів списку:');
   Readln(n);
   for i:=1 to n do
     Begin
       New(list0);
       Readln(list0^.data);
       list0^.next:=list;
       list:=list0
     End;
End;
```

Список закінчується вказівником зі значенням Nil.

Процедура виведення даних, які знаходяться у списку:

```
Procedure OutList(list : pNode);
Begin
  while list <> Nil do
    Begin
      Writeln(list^.data);
       list:= list^.next
    End;
End;
```

Дерева.

Якщо кожен вузол списку має два вказівники, така структура називається **бінарним деревом**.

A Початкова точка бінарного дерева називається кореневим вузлом. Кожен вузол
\ дерева має дві гілки – праву і ліву. Виключенням є лише вершини найнижчого
C яруса. Дерева зручно використовувати, коли необхідно додавати і вилучати ін–
\ формацію з певним чином упорядкованої структури. Наприклад, дерево на рис.3
K зберігає послідовність A C K D Y.
/ \
D Y Рис.3

Опис дерева:

```
Type pTree = ^Tree;
        Tree = Record
                 Data : String;
                 Left, Right : pTree;
               End;
```

Приклад. Використання дерева для збереження множини рядків у алфавітному порядку. Кожний вузол дерева містить один рядок. Якщо ліве піддерево визначене, його корінь містить лексикографічно менший рядок. Якщо визначене праве піддерево, його корінь містить лексикографічно більший рядок (див. рис.3). При обході дерева спочатку друкується ліве піддерево, потім вузол, потім – праве піддерево.

```
 Var Base : pTree;
…
Base := Nil;
```

Функція вставки елемента у дерево.

```
Function AddTree(Base: pTree; S: String): pTree;
Begin
  if Base = Nil then
    Begin
      New(Base);
      Base^.Data:=S;
      Base^.Left:=Nil;
      Base^.Right:=Nil;
    End
else
  if Base^.Data>S then Base^.Left:=AddTree(Base^.Left,S)
  else Base^.Right:=AddTree(Base^.Right,S);
AddTree:=Base;
End;
```

Процедура друку дерева.

```
Procedure PrintTree(Base: pTree);
Begin
  if Base<>Nil then
```

```
    Begin
      PrintTree(Base^.Left);
      Writeln(Base^.Data);
      PrintTree(Base^.Right)
    End;
End;
```

Функція пошуку у дереві: повертає TRUE, якщо елемент є у дереві, і FALSE, якщо ні.

```
Function SearchTree(Base: pTree; S: String): Boolean;
Begin
  SearchTree:=FALSE;
  while Base<>Nil do
    if Base^.Data=S then
      SearchTree:=TRUE
    else if Base^.Data>S then
            Base:=Base^.Left
          else Base:=Base^.Right;
End;
```

Питання для самоконтролю.

1. *Чим розрізняються статичні і динамічні структури даних?*
2. *Що таке стек, який принцип роботи стеку?*
3. *Як реалізується стек у ТР? Наведіть приклади.*
4. *Що таке черга, який принцип роботи черги?*
5. *як реалізується черга у ТР? Нведіть приклади.*
6. *Що таке список? Як здійснюються основні операції зі списками у ТР?*
7. *Що таке дерево? Як здійснюється опис і опрацювання дерев у ТР?*

Основи комп'ютерної графіки. Робота з графікою в системі Турбо-Паскаль

Монітор комп'ютера призначений для виведення на екран текстової і графічної інформації. Монітор працює під управлінням спеціального апаратного пристрою – відеоадаптера, який може працювати в ддвох режимах - текстовому і графічному.

В текстовому режимі екран розбивається на 25 рядків по 80 позицій у кожному (всього 2000 позицій). В кожну позицію (знакомісце) може ббути виведений певний символ з кодової табллиці.

В графічному режимі екран складається з окремих точок, які називаються **пікселями** (від англ. picture element – pixel). Кожен піксель характеризується своїми координатами і кольором.

В IBM-сумісних комп'ютерах може бути встановлено декілька режимів роботи графічного екрану. Набір режимів залежить від віддеоадаптера, обсягу відеопам'яті, монітора.

Режим екрану характеризується кількістю пікселів по горизонталі і вертикалі (роздільною здатністю) і кількістю можливих кольорів для кожного пікселя. Наведемо параметри графічних екранів для різних адаптерів:
адаптер CGA (Color Graphic Adapter) – 640x200x2, 320x200x4;
EGA (Enhanced Graphic Adapter) – 640x350x16;
VGA (Video Graphic Array) – 640x480x16;
SVGA (SuperVGA) – 800x6000x256 (2^{24}), 1024x768x2^{24}.

Система TP дозволяє працювати у графічному режимі. Для цього призначений модуль Graph, який забезпечує управління графічними режимами різних адаптерів, і містить більше 50 графічних процедур і функцій. Крім модуля Graph, для роботи з графікою необхідне певне програмне забезпечення, яке формує команди для графічного адаптера - графічні драйвери. вони знаходяться на диску у вигляді файлів *.BGI (CGA.BGI, EGAVGA.BGI). Файл BGI – це графічний інтерфейс фірми Borland (Borland Graphic Interface). Він забезпечує взаємодію програми з графічними пристроями.

Робота з графікою в TP відбувається у три етапи:
1) ініціалізація графічного режиму;
2) виконання графічних побудов;
3) закриття графічного режиму.

Для ініціалізації графічного режиму призначена процедура:
Procedure InitGraph(Var GraphDriver: Integer; Var GraphMode: Integer; DriverPath: String);
GraphDriver – номер графічного драйвера,
GraphMode – номер графічного режиму для даного драйвера,
DriverPath – шлях до графічного драйвера.

Процедура переводить екран у графічний режим, якщо це можливо. Якщо ні – повідомляє про це у графічну бібліотеку. У модулі Graph визначені

константи для задання параметра GraphDriver, наприклад, Detect=0; CGA=1; EGA==3; VGA=9 тощо.

Якщо перший параметр процедури має значення Detect, система переходить в режим автовизначення. При цьому встановлюється найкращий графічний режим з можливих для даного адаптера. Про успішність ініціалізації може повідомляти функція:

Function GraphResult: Integer;

Якщо результатом функції є константа GrOk, ініціалізація пройшла успішно, якщо інше значення, то ні.

Для закінчення роботи з графікою призначена процедура

Procedure CloseGraph;

Вона звільняє пам'ять, зайняту під графічний драйвер, і переводить екран у текстовий режим.

Приклад. Відкриття і закриття графічного режиму.

```
Uses Graph;
Var GD, GM : Integer;
Begin
  GD:= detect;
   InitGraph(GD, GM, 'C:\TP\BGI');
  {побудови}
  CloseGraph
End.
```

Система координат екрана має вигляд:

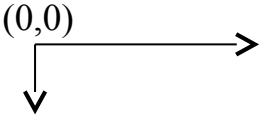

Рис.4.

Для визначення максимальних координат екрана використовуються функції:

Function GetMaxX: Integer;

Function GetMaxY: Integer;

Існує поняття "графічний курсор" (ГК), який виконує ті ж функції, що і у текстовому режимі, але є невидимим. Для визначення поточного положення ГК на екрані використовуються функції:

Function GetX: Integer; Function GetY: Integer;

Procedure MoveTo(x,y: Integer); переміщує ГК в точку з координатами (x,y),

Procedure MoveRel(dx,dy: Integer); переміщує ГК на dx по горизонталі і dy по вертикалі від поточної позиції.

Для очищення екрану в графічному режимі призначені процедури

Procedure ClearDevice; - очищує екран і переводить графічний курсор в позицію (0,0);

Procedure GraphDefaults; - очищує екран і перевстановлює всі параметри як за замовчуванням.

На графічному екрані може бути визначене графічне вікно – прямокутна частина екрану, у верхній лівий кут якої переноситься початок системи координат. Для цього призначена процедура:

Procedure SetViewPort(x1, y1, x2, y2: Integer; ClipMode:; Boolean);

(x1, y1)-(x1, y2) – діагональ вікна;

параметр ClipMode – може мати два значення: ClipOn(TRUE) – зображення не виводиться за межі графічного вікна (обрізається) і ClipOff(FALSE) – зображення "не зважає" на межі вікна.

Для очищення робочого простору графічного вікна (факттично, відмови від нього), використовується процедура:

Procedure ClearViewPort;

Побудова основних графічних примітивів

Procedure PutPixel(x.y: Integer; Color: Word); - зафарбовує піксель (x,y) у колір Color;

Function GetPixel(x,y: Integer):Word; - повертає значення кольора точки (x,y);

pProcedure Line(x1, y1, x2, y2: Integer); - будує відрізок, кінці якого мають координати (x1,y1), (x2,y2);

Procedure LineTo(x,y: Integer); - будує відрізок з поточної точки до точки (x,y);

proocedure LineRel(dx, dy: Integer); - будує відрізок відносно поточної позиції ГК.

Лінії будуються за попередньо встановленими стилем і кольором. Для встановлення стилю ліній використовується процедура:

Procedure SetLineStyle(LineStyle, Pattern, Thickness: Word);

параметр LineStyle – тип ліній: 0 – звичайна лінія, 1 – точечна, 2 – штрихпунктирна, 3 – пунктирна, 4 – тип ліній явно задається шаблоном;

параметр Pattern, якщо LineStyle ≠ 4, дорівнює 0, інакше необхідно задати шаблон з 16 пікселів (1 - світиться, 0 – ні: 1100110011001100), перевівши у десяткову або шістнадцяткову систему числення;

параметр Thickness – товщина ліній. 1 – нормальна товщина (1 піксель), 3 – потовщена лінія.

Для встановлення кольору фону і кольору пера (побудов) використовуються процедури:

Procedure SetColor(Color: Word); - колір пера;

Procedure SetBkColor(Color: Word); - колір фону.

Отримати поточні значення відповідних кольорів можна за допомогою функцій:

Function GetColor: Word; - повертає поточний колір пера;

Function GetBkColor: Word; - повертає поточний колір фону.

Константи, відповідні кольорам:

Black=0; чорний

Blue=1; синій
Green=2; зелений
Cyan=3; блакитний
Red=4; червоний
Magenta=5; бузковий
Brown=6; коричневий
LightGray=7; яскраво-сірий
DarkGray=8; темно-сірий
LightBlue=9; яскраво-синій
LightGreen=10; яскраво-зелений
LightCyan=11; яскраво-блакитний
LightRed=12; яскраво-червоний
LightMagenta=13; яскраво-бузковий
Yellow=14; жовтий
White=15; білий

Для зафарбовування замкнених областей попередньо встановлюється шаблон і колір заливки:

Procedure SetFillStyle(Pattern: Word; Color: Word);

Pattern визначає вигляд шаблону заливки;
Color визначає колір шаблону.
Для параметра передбачені константи:
0 – заливка (суцільна) кольором фону;
1 – заливка (суцільна) поточним кольором;
2 – заливка лініями типу = = = = = =;
3 – заливка лініями типу ////////////////;
…
11 – заливка точками ::::::::::::::.

Procedure Rectangle(x1,y1,x2,y2: Integer); - малює прямокутник з діагоналлю (x1,y1) – (x2,y2);

Procedure DrawPoly(NumPoints: Word; Var PolyPoints); - зображує ламану, задану набором координат певної множини точок. Це може бути як геометрична фігура, так і таблично задана функція.

NumPoints – кількість точок (якщо необхідно зобразити замкнений N-кутник, треба задати N+1 точку, і координати N+1-ї точки повинні дорівнювати координатам першої);
PolyPoints – масив з двокомпонентних записів:
Type PointType = Record X,Y: Inteer End;
Var PT: Array[1..n] of PointType;

Для зображення кола використовується процедура

Procedure Circle(x,y: Integer; Radius: Word);

(x,y) – центр кола, Radius – радіус кола.

В модулі Graph є процедури малювання еліпсів, дуг, секторів. Вони запитують параметри StartAngle і EndAngle, які позначають початковий і

кінцевий кут дуги. Кути відміряються від додатного напрямку осі X проти годинникової стрілки у градусах .

Procedure Arc(x, y: Integer; StartAngle, EndAngle, Radius: Word); - дуга радіуса Radius з центра (x, y0 від кута StartAngle до EndAngle;

Procedure Ellipse(x, y: Integer; StartAngle, EndAngle, XRadius, YRadius: Word); - еліптична дуга, XRadius і YRadius – розміри горизонтальної і вертикальної полуосей. Для зображення еліпса слід задати StartAngle=0, EndAngle=360.

Заливка областей зображення

Procedure Bar(x1, y1, x2, y2: Integer); - прямокутник з діагоналлю (x1, y1)-(x2, y2), внутрішня область якого залита за поточним шаблоном;

Procedure Sector(x, y: Integer; StartAngle, EndAngle, XRadius, YRadius: Word); - сектор еліпса, залитий за заданим шаблоном;

Procedure PieSlice(x, y: Integer; StartAngle, EndAngle, Radius: Word); - сектор кола еліпс поточного кольору, залитий за поточним шаблоном;

Procedure FillEllipse(x, y: Integer; XRadius, YRadius: Word); - еліпс поточного кольору, залитий за поточним шаблоном;

Procedure FillPoly(NumPoints: Word; Var PolyPoints); - заповнення багатокутника за поточним шаблоном;

Procedure FloodFill(x,y: Integer; Border: Word); - універсальна процедура, яка заливає всю область навколо точки (x,y), обмежену лініями кольору Border.

Виведення тексту в графічному режимі

Для виведення тексту використовуються процедури:

Procedure OutText(TextString: String); - виводить на екран рядок TextString від позиції графічного курсора;

Procedure OutTextXY(x, y: Integer; TextString: String); - виводить на екран рядок TextString від позиції (x,y);

Можна задати власні параметри тексту, що буде виводитись, за допомогою процедури:

Procedure SetTextStyle(Font, Direction: Word; CharSize: Word);

параметр Font визначає номер шрифта: 0 – матричний шрифт 8x8 (за замовчуванням), 1 – напівжирний шрифт, 2 – потончений, 3 – рублений, 4 – готичний;

Direction – розташування тексту: 0 - горизонтальне (за замовчуванням), 1 – вертикальне знизу вгору;

CharSize – розмір символів: 0 – стандартний розмір, 1-10 - стандартний розмір з відповідним коефіцієнтом.

Робота з фрагментами зображень

У модулі Graph є можливість запам’ятати в буфері прямокутний фрагмент графічного зображення і потім вивести цей фрагмент в потрібному місці екрана в потрібному режимі. Так можна створювати рухомі зображення.

При роботі з фрагментом зображення треба його створити (намалювати). Далі необхідно визначити об'єм зображення в байтах (для подальшого збереження). Для цього призначена функція:

Function ImageSize(x1, y1, x2, y2: Integer): Word;

(x1,y1)-(x2,y2) – діагональ прямокутника, що включає фрагмент.

Отже, об'єм зображення: Size = ImageSize(x1, y1, x2, y2);

Визначений обсяг пам'яті необхідно виділити в "кучі" під спеціальну змінну – нетипізований вказівник (Var P: Pointer):

GetMem(P, Size);

Запис зображення в буфер здійснюється процедурою:

Procedure GetImage(x1,y1, x2, y2: Integer: Var Q);

x1, y1, x2, y2 мають ті ж значення, що у ImageSize; Q - змінна, яка займає область пам'яті розміром ImageSize. Це змінна P^:

GetImage(x1,y1,x2,y2, P^);

Для виведення зображення з буфера на екран використовується процедура:

Procedure PutImage(x1, y1: Integer; Var Q; Mode: Word);

x1, y1 – координати лівого верхнього кута нового положення фрагмента, Var Q вказується, як у GetImage, Mode – режим виведення зображення (суміщення зображень фрагменту і фона). У модулі Graph визначені такі константи для параметра Mode:

CopyPut=0; заміщення (очищення перед малюванням)

XORPut=1; виключаюче "або"

ORPut=2; "або"

ANDPut=3; "і"

NOTPut=4; "не"

Для створення рухів використовується режим XORPut. Якщо вивести зображення в цьому режимі 2 рази підряд, воно з'явиться, а потім зникне, не псуючи фону. Щоб побачити зображення, треба між викликами процедури зробити затримку (наприклад, за допомогою процедури з модуля CRT: Delay(ms); ms – кількість мілісекунд):

PutImage(x11, y11, P^, XORPut); Delay(20);

PutImage(x11, y11, P^, XORPut);

Приклад. Прямокутник, що рухається по екрану, відштовхуючись від його меж.

(x1, y1), (x2, y2) – координати області екрана, яка буде поміщена в буфер;

(sx, sy) – координати початкової точки;

sxmove, symove – кроки переміщення прямокутника.

```
Uses Crt, Graph;
Const r = 20;
Var x1, x2, y1, y2, sx, sy: Integer;
    sxmove, symove, maxx, maxy: Integer;
    GD, GM: Integer;
    size: Word;
    p: Pointer;
Begin
```

```
GD:=Detect;
InitGraph(GD, GM, 'C:\TP\BGI');
x1:=1; y1:=1; x2:=r; y2:=r;
maxx:=GetMaxX; maxy:=GetMaxY;
sx:=x1; sy:=y1;
sxmove:=3; symove:=3;
SetColor(4); SetBkColor(0);
SetFillStyle(1,14);
Bar(x1,y1,x2,y2);
size:=ImageSize(x1,y1,x2,y2);
GetMem(p, size);
GetImage(x1,y1,x2,y2,p^);
repeat
  PutImage(sx,sy,p^,XORPut);
  Delay(20);
  PutImage(sx,sy,p^,XORPut);
  if (sx<r) or (sx>maxx–r) then sxmove:=–sxmove;
  if (sy<r) or (sy>maxy–r) then symove:=–symove;
  sx:=sx+sxmove;
  sy:=sy+symove
until KeyPressed;
FreeMem(p, size);
CloseGraph
End.
```

Питання для самоконтролю.

1. *Чим розрізняється робота в текстовому і графічному режимах роботи комп'ютера?*
2. *Що таке роздільна здатність?*
3. *Як реалізується графічний режим у програмах на ТР?*
4. *Як відбуваються ініціалізація і закриття графічного режиму?*
5. *Що таке графічний курсор? Як його переміщувати по екрану?*
6. *Як відбувається побудова основних графічних примітивів? Наведіть приклади.*
7. *Як відбувається виведення тексту у графічному режимі? Наведіть приклади.*
8. *Як відбувається робота з фрагментами зображень у ТР?*

Лабораторний практикум з програмування

Лабораторна робота №1

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ЛІНІЙНІ ПРОГРАМИ. ПРОГРАМИ З РОЗГАЛУЖЕННЯМИ

МЕТА: Ознайомитись з алфавітом, зарезервованими словами, структурою програми і системою типів мови Турбо Паскаль (ТР). Ознайомитись з основними операторами ТР (поняття простого і складеного операторів, оператори присвоювання, введення–виведення, умовний оператор). Вивчити основні стандартні математичні процедури і функції ТР. Закріпити вивчений матеріал при створенні власних нескладних лінійних програм і програм з розгалуженнями.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання

1. Які символи входять до алфавіту мови програмування Паскаль?
2. Що таке ідентифікатор? За якими правилами утворюються ідентифікатори? Наведіть приклади.
3. Що таке константа, змінна? Як вони описуються у програмі? Як визначаються їх типи, надаються значення? Наведіть приклади.
4. Що таке вираз? Які бувають вирази? Наведіть приклади.
5. Що таке оператор?
1. Опишіть структуру Паскаль–програми.
2. Що таке коментар? Для чого призначені коментари?
3. Для чого призначений оператор присвоювання? Наведіть приклади його використання.
4. Опишіть дію операторів введення і виведення. Наведіть приклади.
5. Назвіть стандартні скалярні типи мови Паскаль. Які характеристики вони мають?
6. Що таке структурований тип даних? Наведіть приклади.
7. Назвіть основні операції і функції для величин цілого типу. Наведіть приклади.
8. Назвіть основні операції і функції для величин дійсного типу. Наведіть приклади.
9. Назвіть основні операції і функції для величин логічного типу. Наведіть приклади.

10. Назвіть основні операції і функції для величин символьного типу. Наведіть приклади.
11. Вкажіть формат і опишіть дію умовного оператора. Наведіть приклади.
12. Вкажіть формат і опишіть дію оператора варіанту. Наведіть приклади.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1

1. Дано числа a, b і c (де a <> 0). Знайти дійсні корені рівняння $ax^4 + bx^2 + c=0$
2. Відомо, що із чотирьох заданих чисел два рівні між собою і не рівні іншим числам. Обчислити значення величини y, яке визначається співвідношенням y=a/(b-c) , де a- значення двох рівних чисел, b - значення більшого, а c - значення меншого із двох нерівних чисел.

ВАРІАНТ 2

1. Дано дійсні числа x і y. З'ясувати, в якій координатній чверті знаходиться точка з координатами (x,y).
2. Дано координати двох центрів кіл та їх радіуси O(x1,y1),r1, O(x2,y2),r2. Вивести, як розташовані ці кола (перетинаються чи ні, одне в іншому).

ВАРІАНТ 3

1. Дано три числа a,b,c. Якщо ці числа можуть розглядатись як сторони трикутника, то необхідно обчислити площу цього трикутника за формулою Герона. В протилежному випадку вважати S=0.
2. Дано чотири числа. Якщо сума найменшого і найбільшого більше суми двох інших чисел, то знайти їх середнє геометричне, інакше знайти середнє арифметичне.

ВАРІАНТ 4

1. З'ясувати, в якій координатній чверті знаходиться трикутник, утворений прямою, заданою рівнянням y=ax+b і осями координат.
2. Трикутник задано координатами вершин A(x1,y1), B(x2,y2) і C(x3,y3). Якщо трикутник рівнобедрений, то вивести значення кута між рівними сторонами, інакше вивести значення його периметру.

ВАРІАНТ 5

1. Нехай D - четверта координатна чверть. Знайти f(x,y), якщо

$$f(x,y)=\begin{cases} \cos x + \sin 3x, & \text{якщо } (x,y) \text{ належить } D \\ x \cdot y/(x-y) & \text{у протилежному випадку} \end{cases}$$

2. Дано чотири числа b, c, d, e. Переозначити ці числа так, щоб b=max(b,c,d,e), c=min(b,c,d,e), а d було б менше за e.

ВАРІАНТ 6

1. Нехай D - коло з центром в точці (3,2) і радіусом 6.
Знайти f(x,y), де

$x^3 - 1 + y$, якщо (x,y) належить D

f(x,y)=

$\sqrt[3]{|x - y + 1|}$ в протилежному випадку

2. Три точки задані своїми координатами A(x1,y1), B(x2,y2) і C(x3,y3). Знайти ту з трьох точок, яка лежить найближче до початку координат. Координати інших точок замінити координатами середини відрізку, вершинами якого є ці точки.

ВАРІАНТ 7

1.Нехай D - квадрат, вершини якого мають координати (3,0), (0,3), (-3,0), (0,-3). Знайти f(x,y), де

$\sqrt[4]{(x^2 + y^2 - 1)}$, якщо (x,y) належить D

f(x,y)=

ln |x+y| + 1/(x·y) в протилежному випадку

2. Знайти полярні координати R і φ точки на площині за її прямокутними координатами.

ВАРІАНТ 8

1.Нехай D - перша координатна чверть. Знайти f(x,y), якщо

$x^{-5} + e^{-y}$, якщо (x,y) належить D

f(x,y)=

$1/(x^2 + 4 \cdot y - 5)$ в протилежному випадку

2. Дано чотири числа, визначити найбільшу та найменшу суму двох із них.

ВАРІАНТ 9

1.Нехай D - друга координатна чверть. Знайти f(x,y), якщо

$e^{-x+y}/(x^3 - y)$, якщо (x,y) належить D

f(x,y)=

$|x-y|^2$ в протилежному випадку

2. Дано точки A(x1,y1), B(x2,y2), C(x3,y3). Визначити, чи є вектори AB і BC колінеарними або перпендикулярними.

ВАРІАНТ 10

1.Нехай D - третя координатна чверть. Знайти f(x,y), якщо

$y - \sqrt{|x - 1|}$, якщо (x,y) належить D

f(x,y)=

$(x^2+y^2)/(x - y)$ у протилежному випадку

2. З'ясувати, якими будуть два заданих координатами своїх вершин трикутника: рівними чи подібними.

Лабораторна робота №2

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ЦИКЛІЧНІ ПРОГРАМИ (ЦИКЛИ WHILE, REPEAT)

МЕТА: Ознайомитись з циклічними операторами ТР (оператори циклу з передумовою та післяумовою). Вивчити особливості використання кожного з операторів. Навчитися застосовувати циклічні оператори при програмуванні ітераційних процесів та при табулюванні функцій.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання

1. Що називається циклом у програмуванні? Які оператори мови ТР призначені для організації циклів?
2. Вкажіть формат оператора циклу з передумовою. Як виконується цей оператор? Які особливості його виконання?
3. Вкажіть формат оператора циклу з післяумовою. Як виконується цей оператор? Які особливості його виконання?
4. Порівняйте особливості виконання двох операторів циклу мови ТР.
5. Що таке табулювання функції? Наведіть приклади.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Зауваження. До першого завдання всіх варіантів: Скласти програму для табулювання функції y=f(x) на відрізку [a,b] з кроком h.

Результати подати у вигляді таблиці

x	F(x)
.	. .

Одне з завдань виконати за допомогою циклу WHILE, інше – REPEAT.

ВАРІАНТ 1

1. $y=x^{4/3}+6$. Якщо обчислене значення y перевищує деяке наперед задане Z, вивести його на екран та припинити подальші обчислення.
2. Вивести на екран перші k додатних членів послідовності: $A_n=\sin(n)$.

ВАРІАНТ 2

1. $y=x^{7/3}$ - x. Якщо обчислене значення y менше деякого наперед заданого Z, вивести його на екран та припинити подальші обчислення.
2. Вивести на екран перші k від'ємних членів послідовності: $A_n=\cos(n)$.

ВАРІАНТ 3

1. $y=x^4 - 2x^2 + 6$. Якщо обчислене значення y дорівнює деякому наперед заданому Z, вивести його на екран та припинити подальші обчислення.
2. Вивести на екран перші k членів послідовності, менших за m, та їх номери: $A_n=(\sin(n))$.

ВАРІАНТ 4

1. $y=x^2 +7x+10$. Якщо обчислене значення y дорівнює 0, вивести його на екран та припинити подальші обчислення.
2. Вивести на екран суму перших k членів послідовності, більших 0.01: $A_n=\sin(n)$.

ВАРІАНТ 5

1. $y=x^2 +5x+6$. Виведення на екран припиняється, якщо два обчислених значення y дорівнюють 0.
2. Вивести на екран перші k членів послідовності, більших за m, та їх номери: $A_n= \sin(n)+2$ (всього перебирати не більше 100 членів послідовності)

ВАРІАНТ 6

1. $y=x^2 +5x+6$. Виведення на екран припиняється, якщо два обчислених значення y менше деякого наперед заданого Z.
2. Вивести на екран суму перших k членів послідовності, менших за 0 : $An= 2 - n\sin(n)$.

(всього перебирати не більш 100 членів послідовності)

ВАРІАНТ 7

1. $y=x^2 +5x+6$. Виведення на екран припиняється, якщо два обчислених значення y більше деякого наперед заданого Z.
2. Обчислити нескінченну суму ряду з заданою точністю eps, вважаючи, що потрібна точність досягнута, якщо наступний доданок менший по модулю за eps:

$$\sum_{i=1}^{\infty} \frac{(-1)^i}{i(i+1)(i+2)}$$

ВАРІАНТ 8

1. $y=x^2 +\sqrt{-x+6}$. Виведення на екран припиняється, якщо значення x не належить ОДЗ. Подається відповідне повідомлення.
2. Обчислити нескінченну суму ряду из заданою точністю eps, вважаючи, що потрібна точність досягнута, якщо наступний доданок менший по модулю за eps:

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{4^i + 5^{i+2}}$$

ВАРІАНТ 9

1. y=x^2 + $\sqrt{-x + \sin(x)} + 6$. Виведення на екран припиняється, якщо значення x не належить ОДЗ. Подається відповідне повідомлення.
2. Обчислити нескінченну суму ряду из заданою точністю eps, вважаючи, що потрібна точність досягнута, якщо наступний доданок менший по модулю за eps:

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{i!}$$

ВАРІАНТ 10

1.y=1/x+$\sqrt{x+3}$ +6. Виведення на екран припиняється, якщо значення x не належить ОДЗ. Подається відповідне повідомлення.
2. Обчислити нескінченну суму ряду из заданою точністю eps, вважаючи, що потрібна точність досягнута, якщо наступ ний доданок менший по модулю за eps:

$$\sum_{i=0}^{\infty} \frac{(-1)^i}{2^i}$$

Лабораторна робота №3

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ЦИКЛИЧНІ ПРОГРАМИ (ЦИКЛ FOR)

МЕТА: Ознайомитись з циклічним оператором ТР - оператором циклу з параметром. Вивчити особливості використання оператора. Засвоїти порівняльні характеристики всіх операторів циклу ТР. Навчитися застосовувати оператор циклу з параметром при розв'язуванні задач на опрацювання матриць.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання

9. Що називається циклом у програмуванні? Які оператори мови ТР призначені для організації циклів?
10. Вкажіть формат оператора циклу з параметром. Які існують модифікації цього оператора? Як вони виконуються?
11. Що називається кроком циклу?
12. Порівняйте особливості виконання трьох операторів циклу мови ТР.
13. Що називається вкладеним циклом? Наведіть приклади.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1

1.Обчислити суму перших n членів послідовності:

y=1+1/8+1/27+1/64+.....

2. Дано цілі числа a_1, a_2, a_3. Отримати цілочисельну матрицю B_{3x3} , для якої $b_{ij}=a_i-3a_j$, i, j = 1,2,3. Обчислити середнє арифметичне елементів побічної діагоналі цієї матриці.

ВАРІАНТ 2

1. Обчислити добуток перших n членів послідовності:

y=(1/2+1/3)*(1/4-1/9)*(1/8+1/27)*.....

2. Дано дійсні числа $a_1,\ldots,a_3$, $b_1,\ldots b_6$. Отримати дійсну матрицю C_{6x3}, для якої $c_{ij}=a_j/(1+|b_i|)$. Обчислити суму додатних елементів цієї матриці.

ВАРІАНТ 3

1. Обчислити суму перших n членів послідовності:

y=1+3/4+5/8+7/16+9/32+11/64+.....

2. Отримати A_{10x12} - цілочисельну матрицю, для якої a_{ij}=i+2j. Обчислити суму елементів, розташованих на головній діагоналі і вище.

ВАРІАНТ 4

1.Обчислити добуток перших n членів послідовності:

y=2/3*3/4*4/5*5/6*6/7*7/8*8/9*9/10*.....

2. Дано дійсну квадратну матрицю A_{nxn}. Отримати дві квадратні матриці B_{nxn}, C_{nxn}, для яких

$$b_{ij} = \begin{cases} a_{ij}, j \quad i \\ a_{ji}, j < i \end{cases}, \qquad c_{ij} = \begin{cases} a_{ij}, j < i \\ -aij, j \quad i \end{cases}.$$

ВАРІАНТ 5

1. Обчислити суму перших n членів послідовності:

y=1/9+1/25+1/49+1/81+.....

2. Дано натуральне число n. З'ясувати, скільки додатних елементів містить матриця A_{nxn}, якщо $a_{ij}=\sin((i^2-j^2)/n)$. Матрицю вивести на екран.

ВАРІАНТ 6

1. Обчислити суму перших n членів послідовності:

y=1+2/4+3/16+4/36+5/64+.....

2. Дано цілочисельну матрицю A_{nx}n.Отримати b1,…bn, де

$$b_i = \sum_{j=1}^{n} a_{ij}^2 .$$

ВАРІАНТ 7

1.Обчислити добуток перших n членів послідовності:

y=1/2*3/4*5/6*.....

2. Дано цілочисельну матрицю A_{nxn}. Отримати $b_1,\ldots,b_n$, де

$$b_i = \prod_{j=1}^{n} a_{ij}.$$

ВАРІАНТ 8

1. Обчислити добуток перших n членів послідовності:

y=1*3/2*5/3*.....

2. Дано цілочисельну матрицю A_{nxn}. Отримати $b_1,\ldots,b_n$, де

$$b_i = \sum_{j=1}^{n} (-1)^{i+j} a_{ij}.$$

ВАРІАНТ 9

1.Обчислити добуток перших n членів послідовності:

y=1*1/2*1/(2*3)*1/(2*3*4)*1/(2*3*4*5)*.....

2.Дано цілочисельну матрицю A_{nxn}.Отримати $b_1,\ldots,b_n$, де

$$b_i = \sum_{j=1}^{n} \left| a_{ij} \right| .$$

ВАРІАНТ 10

1. Обчислити суму перших n членів послідовності:
y=1/2+1/4+1/16+1/32+1/64+.....
2. Дано натуральне число n. З'ясувати, скільки від'ємних елементів містить матриця A_{nxn}, якщо $a_{ij}=\cos(i^2+n)$. Матрицю вивести на екран.

Лабораторна робота №4

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ПРОЦЕДУРИ І ФУНКЦІЇ

МЕТА: Ознайомитись з поняттям підпрограми, засобами реалізації підпрограм у мові TP: процедурами і функціями. Вивчити особливості опису і виклику процедур і функцій. Вивчити класифікацію параметрів підпрограм, способи передавання параметрів. Знати правила локалізації, поняття рекурсії та побічного ефекту. Навчитися розв'язувати задачі з максимальним використанням підпрограм.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання

13. Що називається підпрограмою, основною програмою?
14. Чим відрізняються вбудовані процедури і функції від процедур і функцій користувача?
15. Як описуються і використовуються в програмі процедури користувача?
16. Що таке формальні і фактичні параметри?
17. Що таке глобальні і локальні змінні?
18. Які способи передавання параметрів існують в програмах на TP? В чому їх особливості?
19. Як описуються і використовуються в програмі функції користувача? Які особливості має опис заголовку і тіла функції?
20. Вкажіть відмінності процедур і функцій.
21. Вкажіть правила локалізації TP.
22. Що таке рекурсія? Наведіть приклади використання рекурсії.
23. Що таке побічний ефект? Наведіть приклади.
24. Для чого призначені стандартні процедури Exit і Halt?

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1.

1. Дано дійсні числа a,b. Отримати u=min(a,b), v=min(ab,a+b), $\min(u+v^2, 3.14)$
2. Дано дійсні числа s,t. Отримати
$h(s,t) + \max(h(s-t,st), h(s-t,s+t)) + h(1,1)$, де $h(a,b)= a/(1+b^2) + b/(1+a^2)$

ВАРІАНТ 2.

1. Три трикутники задаються координатами своїх вершин на площині. Визначити трикутник, що має найбільшу площу, та обчислити його периметр (для знаходження найбільшої площі також використати процедуру або функцію).
2. Дано дійсні числа s,t. Отримати f(t,-2s,1.17) + f(2.2,t,s-t),

$$\text{де } f(a,b,c)= \frac{2a-b-\sin(c)}{5+c}$$

ВАРІАНТ 3.

1. Чотирикутник (довільний) задається координатами вершин на площині. Знайти мінімальну і максимальну відстань між двома вершинами чотирикутника (розглянути і несуміжні вершини).
2. Дано дійсні числа a,b,c. Отримати

$$\frac{\max(a,a+b) + \max(a,b+c)}{1 + \max(a+bc,1,15)}$$

ВАРІАНТ 4.

1. Три трикутники задаються координатами своїх вершин на площині. Знайти трикутник, що має найбільший периметр, та обчислити площу цього трикутника.
2. Дано натуральні числа n,m, цілі числа a[1],...,a[n], b[1],...,b[m], c[1],...,c[5]. Отримати

$$L = \begin{cases} \min(b[1],...,b[m]) + \min(c[1],...,c[5]), & \text{при } \min(a[1],...,a[n]) > 10, \\ 1 + (\max(c[1],...,c[5]))^2 & \text{в протилежному випадку} \end{cases}$$

ВАРІАНТ 5.

1. Три квадратні рівняння задаються своїми коефіцієнтами a,b,c. Визначити та надрукувати те з них, що має найбільший корінь.
2. Дано дійсні числа s,t, a[0],...,a[5]. Отримати $p(1)-p(t)+p^2(s-t)-p(3)$, де

$$p(x)= a[5]x^5 + a[4]x^4 + ... + a[0]$$

ВАРІАНТ 6.

1. Три зведені квадратні рівняння задаються своїми коренями x_1, x_2. Визначити та надрукувати те з них, що має найбільший додатний коефіцієнт при x (вивести рівняння в звичайному математичному вигляді).
2. Дано дійсні числа u_1,u_2, v_1,v_2, w_1,w_2. Отримати $2u + 3uw/(2+w-v) - 7$, де u,v,w - комплексні числа u_1+iu_2, v_1+iv_2, w_1+iw_2. (Визначити процедури виконання арифметичних операцій над комплексними числами).

ВАРІАНТ 7.

1. Три трикутники задаються координатами своїх вершин на площині. Визначити трикутник, що має найменшу площу, та обчислити його периметр.
2. Дано 4-елементні дійсні вектори x, y і z. Обчислити величину (a,a) - (b,c), де a означає той з цих векторів, в якому найбільший мінімальний елемент (вважаючи, що такий вектор єдиний), b і c означають два інших вектора, а (p,q) - скалярний добуток p і q.

ВАРІАНТ 8.

1. Три вектори у 5-вимірному просторі задаються своїми координатами. Скласти програму, яка виводить на екран вектор з найбільшим середнім арифметичним координат.
2. Дано дійсне число y. Отримати $\dfrac{1.7f(0.25) + 2f(1+y)}{6 - f(y^2-1)}$,

де $f(x)= x+x^3 +x^5 +x^7$

ВАРІАНТ 9.

1. Чотири вектори у 6-вимірному просторі задаються своїми координатами. Скласти програму, яка виводить на екран скалярний добуток двох векторів, що мають найбільшу та найменшу норми (норму вектора визначити як найбільшу абсолютну величину координат вектора).
2. Дано дійсні числа s,t. Отримати g(1.2,s) + g(t,s) - g(2s-1,st),

де $g(a,b)= \dfrac{a^2+b^2}{a^2+2ab+3b^2+4}$

ВАРІАНТ 10.

1. Чотири трикутники задаються координатами своїх вершин на площині. Визначити трикутник, що має найменший периметр, та обчислити його площу.
2. Дано дійсне число y. Отримати $\dfrac{1.7t(0.25) + 2t(1+t)}{6 - t(y^2-1)}$,

де $t(x)= \sum_{k=1}^{10} \dfrac{x^{2k}}{2k}$

Лабораторна робота №5

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ОПРАЦЮВАННЯ СИМВОЛІВ І РЯДКІВ

МЕТА: Ознайомитись з можливостями мови Турбо Паскаль (ТР) в опрацюванні символьних і рядкових величин. Вивчити стандартні процедури і функції опрацювання символів і рядків у ТР. Закріпити вивчений матеріал при створенні власних нескладних програм опрацювання текстової інформації.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання.

1. Для чого призначені типи даних Char і String?
2. Як описуються змінні символьного і рядкового типів засобами ТР?
3. Чи сумісні типи Char і String?
4. Для чого використовуються Pred, Succ, Chr, Ord? Опишіть їх дію на прикладах.
5. Як забезпечується доступ до символа за його кодом без використання функції Chr?
6. Як здійснюється доступ до окремого символа рядка?
7. Вкажіть два способи визначення довжини рядка.
8. Як перетворити число у рядок та навпаки?
9. Чи можна порівнювати рядки? Як?
10. Якими способами можна з'єднати кілька рядків?
11. Яку максимальну довжину може мати рядкова змінна?
12. За допомогою яких операторів рядкові змінні вводяться з клавіатури?
13. Як перетворити маленькі латинські літери у великі?
14. Вкажіть формат процедури Insert. Наведіть приклади її використання.
15. Вкажіть формат процедури Delete. Наведіть приклади її використання.
16. Вкажіть формат функції Copy. Наведіть приклади її використання.
17. Вкажіть формат функції Pos. Наведіть приклади її використання.
18. Вкажіть формат процедури Val. Наведіть приклади її використання.
19. Вкажіть формат процедури Str. Наведіть приклади її використання.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Зауваження.

1. Слід використовувати підпрограму розбиття речення на слова.
2. Завдання виконуються з максимальним використанням підпрограм.

Дано текст, в якому від 1 до 30 слів, в кожному слові від 1 до 8 літер, слова розділені пропусками.

ВАРІАНТ 1

1. Надрукувати всі слова тексту,що містять рівно три літери 'o'.
2. Надрукувати всі симетричні слова тексту.

ВАРІАНТ 2

1. Надрукувати всі слова, що мають мінімальну довжину.
2. Порівняти кількість слів парної довжини і слів, що містять літеру 'k'.

ВАРІАНТ 3

1. Другу літеру кожного слова замінити на 'z'.
2. Надрукувати слова, перша літера яких входить до них ще раз.

ВАРІАНТ 4

1. Надрукувати слова, в яких немає літер, що повторюються.
2. З кожного слова вилучити першу літеру.

ВАРІАНТ 5

1. Вилучити з тексту всі слова, що складаються з однієї літери, підрахувавши їх кількість.
2. Надрукувати слова, які входять в текст не менше двох разів.

ВАРІАНТ 6

1. Підрахувати кількість входжень в текст першого слова.
2. В кожному слові тексту першу літеру перенести в кінець слова.

ВАРІАНТ 7

1. Визначити номер слова, що містить максимальну кількість букв 'o'.
2. В кожному слові останню літеру перенести на початок слова.

ВАРІАНТ 8

1. Порівняти кількість слів, що містять і не містять літеру 'm'.
2. Надрукувати слова, довжина яких максимальна.

ВАРІАНТ 9

1. Порівняти кількість слів, що містять літеру 'k' і слів, що складаються з чотирьох літер.
2. Кожне слово тексту надрукувати в зворотньому порядку.

ВАРІАНТ 10

1. Надрукувати всі слова, що мають довжину, рівну довжині останнього слова.
2. Перед кожним словом надрукувати кількість входжень в нього літери 'o'.

Лабораторна робота №6

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ОПРАЦЮВАННЯ ЗАПИСІВ

МЕТА: Ознайомитись з можливостями мови Турбо Паскаль (ТР) в опрацюванні даних комбінованого типу. Вивчити особливості опрацювання записів у ТР. Закріпити вивчений матеріал при створенні власних нескладних програм опрацювання записів.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання.

1. Для чого призначений тип даних Record? Які його особливості?
2. Як описується комбінований тип засобами ТР?
3. Чи повинні всі поля запису мати однаковий тип?
4. Як відбувається звертання до полів запису? Які дії можна виконувати з полями записів у ТР?
5. Які операції можна виконувати в ТР над змінними-записами вцілому?
6. Для чого призначений оператор With? Наведіть приклади його використання.
7. Чи можуть записи бути вкладеними?
8. Для чого призначені записи з варіантами?

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1

1. Описати комбінований тип, який містить основні відомості про літаки: назва, швидкість, вантажність, назву країни-виробника. Визначити:
   а) назви літаків, що виробляє задана користувачем країна;
   б) всі відомості про літак з найбільшою швидкістю.
2. Описати комбінований тип, який містить відомості про розклад занять: назва дисципліни, викладач, вид заняття (лекція, практичне, лабораторне), номер групи, аудиторія, номер пари, день тижня. Визначити номери аудиторій, в яких будуть заняття у середу.

ВАРІАНТ 2

1. Описати комбінований тип, який містить основні відомості про літаки: назва, швидкість, вантажність, назву країни-виробника. Визначити:
 а) назви країн, що виробляють лтаки із швидкістю, більшою за вказану;
 б) всі відомості про літак з найменшою вантажністю.
2. Описати комбінований тип, який містить відомості про розклад занять: назва дисципліни, викладач, вид заняття (лекція, практичне, лабораторне), номер групи, аудиторія, номер пари, день тижня. Визначити, чи працює певний викладач в певний день та його навантаження в цей день.

ВАРІАНТ 3

1. Описати комбінований тип, який містить основні відомості про літаки: назва, швидкість, вантажність, назву країни-виробника. Визначити:
 а) назву країни, що виробляє літак з найменшою вантажністю;
 б) всіі відомості про літак з найбільшою швидкістю, що виготовляє ця країна.
2. Описати комбінований тип, який містить відомості про розклад занять: назва дисципліни, викладач, вид заняття (лекція, практичне, лабораторне), номер групи, аудиторія, номер пари, день тижня. Визначити номера груп, що не мають занять в понеділок на 2-й парі.

ВАРІАНТ 4

1. Описати комбінований тип, який містить основні відомості про магазин: номер магазину, відомості про наявність у продажу масла та м'яса, ціни на ці продукти та розмір товарообігу. Визначити:
 а) номери магазинів, де є у продажу лише один з продуктів;
 б) загальний товарообіг усіх магазинів.
2. Описати комбінований тип, який містить основні відомості про розклад руху літаків: місце призначення, рейс, вартість квитка, назва літака, чи є квітки. Визначити назви рейсів і місця призначення літаків, час польоту яких менш двох годин, і на них є квитки.

ВАРІАНТ 5

1. Описати комбінований тип, який містить основні відомості про магазин: номер магазину, відомості про наявність у продажу масла та м'яса, ціни на ці продукти та розмір товарообігу. Визначити:
 а) ціну масла в магазині з указаним номером;
 б) всі відомості про магазин з найменшим товарообігом.
2. Описати комбінований тип, який містить основні відомості про розклад руху літаків: місце призначення, рейс, вартість квитка, назва літака, чи є квитки. Визначити всі рейси на Москву, на які є квитки.

ВАРІАНТ 6

1. Описати комбінований тип, який містить основні відомості про магазин: номер магазину, відомості про наявність у продажу масла та м'яса, ціни на ці продукти та розмір товарообігу. Визначити:

а) магазини, в яких є масло;

б) той з них, товарообіг в якому найбільший.

2. Описати комбінований тип, який містить основні відомості про розклад руху літаків: місце призначення, рейс, вартість квитка, назва літака, чи є квитки. Визначити найдорожчий рейс на Київ та вивести всі відомості про нього.

ВАРІАНТ 7

1. Описати комбінований тип, який містить основні відомості про магазин: номер магазину, відомості про наявність у продажу масла та м'яса, ціни на ці продукти та розмір товарообігу. Визначити:

а) магазини, в яких є м'ясо;

б) той з них, товарообіг в якому найменший.

2. Описати комбінований тип, який містить основні відомості про розклад руху літаків: місце призначення, рейс, вартість квитка, назва літака, чи є квитки. Визначити ті рейси, вартість квитка на які менша двохсот гривень і на них немає квитків.

ВАРІАНТ 8

1. Описати комбінований тип, який містить основні відомості про магазин: номер магазину, відомості про наявність у продажу масла та м'яса, ціни на ці продукти та розмір товарообігу. Визначити:

а) магазини, в яких немає м'яса та масла;

б) загальний товарообіг цих магазинів.

2. Описати комбінований тип, який містить основні відомості про розклад руху літаків: місце призначення, рейс, вартість квитка, назва літака, чи є квитки. Визначити три найдорожчи рейси, на які є квитки.

ВАРІАНТ 9

1. Описати комбінований тип, який містить основні відомості про магазин: номер магазину, відомості про наявність у продажу масла та м'яса, ціни на ці продукти та розмір товарообігу. Визначити:

а) магазин, в якому найдорожче м'ясо;

б) загальний товарообіг решти магазинів.

2. Описати комбінований тип для подання анкети школяра, яка містить в собі прізвище, ім’я, рік народження, стать, номер класу, букву класу, та оцінки з алгебри, геометрії, інформатики. Визначити відмінників, що навчаються в 11-"А".

ВАРІАНТ 10

1. Описати комбінований тип, який містить основні відомості про літаки: назва, швидкість, вантажність, назву країни-виробника. Визначити:
   а) швидкості літаків, що виробляє певна країна;
   б) всі відомості про літак з найбільшою вантажністю.
2. Описати комбінований тип, який містить відомості про розклад занять: назва дисципліни, викладач, вид заняття (лекція, практичне, лабораторне), номер групи, аудиторія, номер пари, день тижня. Визначити номера груп, що не мають занять у вівторок на 3-й парі.

Лабораторна робота №8

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ОПРАЦЮВАННЯ ТЕКСТОВИХ ФАЙЛІВ

МЕТА: Ознайомитись з можливостями мови Турбо Паскаль (ТР) в опрацюванні файлів. Засвоїти особливості опрацювання текстових файлів у ТР. Вивчити стандартні процедури і функції опрацювання файлів. Закріпити вивчений матеріал при створенні власних нескладних програм опрацювання текстових файлів.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання.

1. Яка структура даних ТР називається файлом ?
2. Як оголошуються текстові файли у Паскаль-програмах? Наведіть приклади.
3. Які особливості мають текстові файли?
4. Як здійснюється доступ до елементів текстового файла?
5. В яке місце текстового файла можна додавати нові елементи: на початок, в кінець, куди завгодно, нікуди ?
6. Значення яких елементів текстового файла можна змінювати: тільки першого, тільки останнього, яких завгодно, ніяких ?
7. Значення яких елементів текстового файла можна вилучати?
8. Чи можна порівнювати текстові файли ?
9. Чи можна присвоювати один текстовий файл іншому ?

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Зауваження. Виконання першого завдання всіх варіантів передбачає попереднє створення файла на диску (файл створюється з указаної користувачем кількості рядків або до введення указаної ознаки закінчення)

ВАРІАНТ 1

1. Дано текстовий файл, розбитий на рядки. Надрукувати всі рядки, що мають мінімальну довжину.
2. Дано текстовий файл f. Переписати у файл g всі рядки файла f, що містять більше 30 символів.

ВАРІАНТ 2

1. Дано текстовий файл, розбитий на рядки. Надрукувати всі рядки, що містять дві літери "a".
2. Дано текстовий файл f. Переписати в файл g всі елементи файла f с заміною в них символа 0 на символ 1 і навпаки.

ВАРІАНТ 3

1. Дано текстовий файл, розбитий на рядки. Передостанню літеру кожного рядка замінити на 'm'.
2. Дано текстовий файл f. Записати в перевернутому вигляді рядки файла f в файл g. Порядок рядків у файлі g повинен співпадати з порядком рядків у файлі f.

ВАРІАНТ 4

1. Дано текстовий файл, розбитий на рядки. Кожний рядок тексту надрукувати в зворотньому порядку.
2. Дано текстовий файл f. Отримати найдовший рядок файла. Якщо в файлі є кілька рядків з найбільшою довжиною, отримати один з них.

ВАРІАНТ 5

1. Дано текстовий файл, розбитий на рядки. Надрукувати рядки, що мають непарну довжину, підрахувавши їх кількість.
2. Дано текстовий файл f. Переписати елементи файла f в файл g, вставляючи в початок кожного рядка літеру “o”. Порядок рядків повинен бути збережений.

ВАРІАНТ 6

1. Дано текстовий файл, розбитий на рядки. Підрахувати кількість входжень в текст першого рядка.
2. Дано текстовий файл f. Переписати в файл g всі рядки з f, в яких друга літера співпадає з передостанньою.

ВАРІАНТ 7

1. Дано текстовий файл, розбитий на рядки. Визначити номер рядка, що містить три літери "o".
2. Дано текстовий файл f, рядок s. Отримати всі рядки файла f, фрагментом яких є рядок s.

ВАРІАНТ 8

1. Дано текстовий файл, розбитий на рядки. Порівняти кількість рядків, що містять літеру 'k' і рядків, що складаються з чотирьох літер.
2. Дано текстовий файл f, розбитий на рядки. Переписати в файл g всі рядки з f, в яких перша літера співпадає з останньою.

ВАРІАНТ 9

1.Дано текстовий файл, розбитий на рядки. Останній рядок тексту надрукувати в зворотньому порядку.
2. Дано текстовий файл f. Вилучити пропуски, що містяться в його рядках. Результат помістити в файл g.

ВАРІАНТ 10

1. Дано текстовий файл, розбитий на рядки. Надрукувати всі рядки, що мають довжину, рівну довжині останнього рядка.
2. Дано текстовий файл f. У початок кожного рядка вставити його довжину. Результат помістити в файл f1.

Лабораторна робота № 8

ТЕМА: МОВА ПРОГРАМУВАННЯ ТУРБО ПАСКАЛЬ. ОПРАЦЮВАННЯ ТИПІЗОВАНИХ ФАЙЛІВ

МЕТА: Засвоїти особливості опрацювання типізованих файлів у ТР. Вивчити стандартні процедури і функції опрацювання типізованих файлів. Закріпити вивчений матеріал при створенні власних нескладних програм опрацювання типізорваних файлів.

ОБЛАДНАННЯ: технічне забезпечення: ПЕОМ, програмне забезпечення: система програмування Turbo Pascal 6.0.

ЗАВДАННЯ ДО РОБОТИ:

Вивчити необхідний теоретичний матеріал.

Відповісти на контрольні запитання.

Виконати відповідні практичні завдання з варіантів для самостійного виконання.

Оформити звіт (завдання до роботи, тексти програм, контрольні приклади та результати їх виконання).

Контрольні запитання.

1. Як розуміється файл у системі Турбо Паскаль?
2. Як оголошуються файли у Паскаль-програмах? Наведіть приклади.
3. Які файлові типи підтримуються системою ТР? Які особливості цих типів?
4. Які особливості використання файлових змінних у Паскаль – програмах?
5. Як здійснюється доступ до окремих елементів файла? Як розрізняються файли за способом доступу до їх елементів?
6. Чи повинні всі елементи файла бути одного типу ?
7. В яке місце файла можна додавати нові елементи: на початок, в кінець, в будь-яке місце, нікуди?
8. Чи можна, зчитавши з файла п’ятий елемент, потім зчитати другий?
9. Чи можна одночасно читати інформацію з файла і записувати в нього нову інформацію?
10. Як здійснюється створення нового файлу? Наведіть приклади.
11. Як здійснюється опрацювання файла з послідовним доступом? Наведіть приклади.
12. Як здійснюється опрацювання файла з довільним доступом? Наведіть приклади.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Зауваження

1. Виконання першого завдання всіх варіантів передбачає попереднє створення файла (з указаної користувачем кількості елементів або з використанням ознаки закінчення введення).

2. Виконання другого завдання передбачає використання умови другого завдання з лабораторної роботи “Опрацювання записів”. Необхідно створити на

диску файл, елементами якого є записи, відповідні умові задачі (4 – 8 записів), і проводити опрацювання інформації, зчитуючи її з файла.

ВАРІАНТ 1

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) добуток елементів файла F;
б) суму квадратів елементів файла F.

ВАРІАНТ 2

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) модуль суми та квадрат добутку елементів файла F;
б) останній елемент файла F;

ВАРІАНТ 3

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) найбільше із значень елементів файла F;
б) різницю останнього та першого елементів файла F.

ВАРІАНТ 4

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) найбільше із значень модулів елементів файла F з непарними номерами;
б) середнє армифметичне значень елементів файла F.

ВАРІАНТ 5

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) суму елементів файла F;
б) найменше із значень елементів файла F.

ВАРІАНТ 6

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) добуток першого та останнього елементів файла F.
б) суму найбільшого та найменшого значень елементів файла F.

ВАРІАНТ 7

1.Дано файл F, елементи якого – цілі числа. Створити файл G з квадратів елементів файла F, файл H – з непарних елементів файла F.

ВАРІАНТ 8

1.Дано файл F, елементи якого – цілі числа. Знайти:
а) найменше із значень непарних елементів файла F;
б) суму кубів елементів файла F з парними номерами.

ВАРІАНТ 9

1.Дано файл F, елементи якого – цілі числа. Знайти:
а) середнє арифметичне другого і передостаннього елементів файла F;

б) кількість елементів файла F, кратних 9.

ВАРІАНТ 10

1.Дано файл F, елементи якого – дійсні числа. Знайти:
а) різницю добутків усіх парних і непарних елементів файла F;
б) кількість елементів файла F, які кратні 3, але не кратні 6.

Зміст

Література

1. Абрамов С.А. и др. Задачи по программированию. – М.: Наука, 1988. – 224 с.
2. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык паскаль. – М.: Наука, 1988. – 320 с.
3. Верлань А.Ф., Апатова Н.В. Інформатика: Підруч. для учнів 10-11 кл. загальноосв. шк. – К.: Квазар-Мікро, 1998. – 200 с.
4. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир. – 1985.
5. Вьюкова Н.И., Галатенко В.А., Ходулев А.Б. Систематический подход к программированию. – М.: Наука, 1988. – 208 с.
6. Грогоно П. Программирование на языке Паскаль. – М.: Наука, 1982.
7. Жалдак М.І., Рамський Ю.С. Інформатика: Навч. Посібник / За ред. М.І.Шкіля. – К.: Вища школа, 1991. – 319 с.
8. Йенсен К., Вирт Н. Паскаль: руководство для пользователя и описание языка. – М.: Финансы и статистика, 1982.
9. Основы информатики и вычислительной техники: Пробное учеб. Пособие для средних учеб. заведений / А.П.Ершов, А.Г.Кушниренко, Г.В.Лебедев и др. – М.: Просвещение, 1988. – 207 с.
10. Пильщиков В.Н. Сборник упражнений по языку Паскаль. – М.: Наука, 1989. – 160 с.
11. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль (версия 5.5). – М.: изд-во МАИ, 1992. –576 с.
12. Следзінський І.Ф., Ломакович А.М., Рамський Ю.С., Зароський Р.І. Техніка обчислень і алгоритмізація. – К.: Вища школа, 1991. – 199 с
13. Фаронов В.В. Программирование на персональных ЭВМ в среде Турбо-Паскаль. – 2-е изд. – М.: Изд-во МГТУ, 1992. – 448 с.
14. Шкиль Н.И., Жалдак М.И., Морзе Н.В., Рамский Ю.С. Изучение языков программирования в школе. – к.: Рад. шк., 1988. – 368 с.

Зміст