М.В. Дудик, Ю.С. Рамський, Г.Ю. Цибко

ОСНОВИ ПРОГРАМУВАННЯ

Навчальний посібник

для студентів вищих навчальних закладів
фізико-математичних та
індустріально-педагогічних спеціальностей

Київ 2005

ББК 22.18я73

Рекомендовано Міністерством освіти і науки України
(лист № 14/18.2- 1440 від 25.06.2005)

Рецензенти:
М. І. Жалдак, академік АПН України, доктор педагогічних наук, професор;
В. І. Клочко, доктор педагогічних наук, професор
Г.В.Гуя», кандидат фізико-математичних наук, доцент

Дудик М. В., Рамський Ю. С, Цибко Г. Ю.
Основи програмування: Навчальний посібник для студентів вищих навчальних закладів фізико-математичних та індустріально-педагогічних спеціальностей. - К.: Міленіум, 2005. - 168 с.

ISBN - 966-8063-94-X

У посібнику подаються основні поняття математичного моделювання, огляд основних парадигмів проірамування, теоретичні відомості з основ програмування мовою Паскаль. Теоретичний матеріал супроводжується низкою задач, які можна розглядати як на лекціях, так і на практичних заняттях. Для проведення лабораторних занять пропонується практикум з восьми робіт, які охоплюють основний зміст курсу, і містять контрольні запитання і варіанти завдань для самостійного виконання.

Посібник складено з врахуванням досвіду викладання основ програмування студентам фізико-математичних та індустріально-педагогічних спеціальностей вищих педагогічних навчальних закладів. Може бути використаний вчителями, студентами і старшокласниками на заняттях з інформатики, а також у процесі самостійного вивчення основ програмування.

ББК22.18я73

Передмова

Програмування традиційно є складовою частиною курсів інформатики, що читаються студентам фізико-математичних і індустріально-педагогічних спеціальностей вищих педагогічних навчальних закладів. При цьому мовою програмування, на прикладі якої розглядаються основи процедурного програмування, переважно є І Іаскаль. Це мова програмування високого рівня, що була створена більше тридцяти років тому саме з навчальною метою, проте вийшлася вдалою з точки зору ефективності і самої мови, і машинних програм, які утворюються в результаті трансляції. Завдяки цьому мова Паскаль здобула широке застосування в практичній сфері, разом з тим вона залишилася класичною навчальною мовою програмування завдяки вдалій наочній реалізації основних алгоритмічних структур, можливостям структуризації програм і даних, відокремленню блоків описів і операторів. У зв'язку з цим Паскаль є найбільш методично доцільною мовою для оволодіння основами алгоритмізації і програмування як у школі, так і у вищому навчальному закладі.

У посібнику викладено зміст курсу лекцій з основ програмування мовою Паскаль, які читають студентам фізико-математичних і індустріально-педагогічних спеціальностей вищого педагогічного навчального закладу в рамках навчальних дисциплін "Інформатика і програмування" та "Програмування і математичне моделювання". Лекційний курс пропонується розпочати з розгляду таких питань: основні поняття математичного моделювання; етапи розв'язування практичних задач з допомогою комп'ютера; алгоритмічні мови, мови програмування, поняття про трансляцію і програми-транслятори, історія розвитку мов програмування, огляд основних парадигм програмування. При розгляді основ мови програмування І Іаскаль матеріал викладається у такій послідовності: алфавіт і синтаксис мови; поняття про типізацію даних, класифікація типів даних мови, стандартні типи даних та операції над ними, типи даних, що визначаються програмістом; основні оператори мови: присвоювання, введення-виведення, розгалуження; організація циклів. Значна увага приділяється питанням структуризації даних у мові Паскаль і розгляду стандартних структурованих типів даних (масиви, рядки, множини, записи, файли послідовного і прямого доступу), а також динамічних структур даних (стеки, черги, списки, дерева). З метою ефективної розробки програм розглядаються основні поняття структурного програмування, докладно вивчається механізм застосування процедур і функцій, використання стандартних і створення власних бібліотек підпрограм. На завершення курсу коротко опи-

суються основи комп'ютерної графіки і деякі аспекти побудови графічних зображень засобами мови Паскаль.

Теоретичний матеріал супроводжується низкою задач, які можна розглядати як на лекціях, так і на практичних заняттях. Для проведення лабораторних занять пропонується практикум з восьми робіт, які охоплюють основний зміст курсу, і містять контрольні запитання і варіанти завдань для самостійного виконання. У посібник не включені питання роботи в інтегрованому середовищі програмування, як такі, що значною мірою залежать від конкретних умов навчання.

У залежності від спеціалізації студентів лекційно-лабораторний курс може бути запропонований як завершений курс основ програмування, або як пропедевтичний до курсу візуального програмування на основі об'єктно-орієнтованого підходу.

Матеріал посібника може бути використаний студентами як конспект лекцій, вчителями в процесі підготовки і проведення уроків, школярами для самостійного вивчення основ програмування.

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 повідомлень в різноманітних галузях людської діяльності.

Поняття інформації інтуїтивно зрозуміле, проте не має точного означення і вважається основним, неозначуваним поняттям інформатики. її часто тлумачать як одну із сторін відображення навколишньої дійсності організмом чи іншою системою.

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

За визначенням академіка В.М.Глушкова "Інформація в найзагальнішому її розумінні є мірою неоднорідності розподілу матерії і енергії в просторі і часі, мірою змін, якими супроводжуються всі процеси, що протікають у світі".[1]

Спроби розглянути категорію інформації з позиції основного питання філософії призвели до виникнення двох підходів, які протистоять один одному: атрибутивного і функціонального. Дослідники атрибутивного підходу (О.Д.Урсул, І.О.Ачкурін, В.М.Глушков, Ю.М. Тесля та ін.) розглядають інформацію як властивість всіх матеріальних об'єктів (яка міститься в їх упорядкованості, різноманітності, організованості), тобто як атрибут матерії. Сутність їх доказів полягає в тому, що різноманітність характеризує будь-який матеріальний об'єкт як і відображення. А оскільки інформація характеризує різноманітність, вона є властивістю всієї матерії.

Представники функціонального підходу (В.Г.Афанасьев, І.І.Гришкін, М.М.Моісєєв та ін.) пов'язують інформацію лише з функціонуванням складних самокерованих, самоорганізованих систем, тлумачать її як одну із сторін активного, цілеспрямованого відображення.

"Я не можу погодитися з думкою тих вчених, - пише М.М.Моісєєв, - які вважають інформацію всезагальною властивістю матеріального світу... Всі явища в неживій природі можуть бути

[1] Глушкон В.М. Кибернетика. Вопросы теории и практики. - М.: Наука, 1986. - С. 14.

пояснені і зрозумілі без залучення поняття "інформація", і описані тільки мовою фізики і хімії. Необхідність введення поняття "інформація" виникає тільки в процесі опису етапів розвитку реального світу, коли в ньому зароджується життя".[2]

Деякі дослідники заявляють, що обидва підходи скоріше всього неповні, оскільки природа свідомості, духу за своєю суттю є інформаційною. Таким чином, інформація і інформаційні процеси, якщо мати на увазі розв'язання основного питання філософії, опосередковують матеріальне і духовне, тобто замість класичної постановки основного питання філософії виникає два: співвідношення матерії і інформації і співвідношення інформації і свідомості (духу)[3].

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 даних.

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 повідомлень.

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 найзагальнішому смислі **моделлю** називається спеціально створена форма об'єкта для подання певних його характеристик, які підлягають дослідженню.

" Моисеев Н.11. Алгоритмы развития. - М.: I Iаука, 1987. - С. 147.
[3] Могилев А.В., ПакН.И., Хённер Е.К. Информатика. - М.: Паука, 1999. - С.28.

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) опрацювання 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 математичною мовою.

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 виникнення інформатики як науки. З появою ЕОМ він отримав нові можливості. Математичне моделювання в інформатиці реалізується в основному як машинний (обчислювальний) експеримент.

Обчислювальний експеримент - це створення і дослідження математичних моделей за допомогою ЕОМ. В основі таких експериментів лежить задания математичних моделей досліджуваних процесів і аналіз їх поведінки в різноманітних змінюваних умовах. Таким чином, експерименти з моделлю дають змогу 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'язування 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 пам'ять комп'ютера 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 дають змогу 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 із 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 $A, = v^2/c^2$, де v - власна швидкість руху, а с - швидкість світла, результати, отримані в рамках обох моделей, практично збігалися. Всі ці моделі чудово витримали випробовування практикою, а їх роль у формуванні світогляду і в практичній діяльності людей важко переоцінити.

Використання математичних методів дослідження в інших науках також тісно пов'язане зі створенням математичних моделей.

Приклад. Розглянемо рух тіла маси ***m*** вздовж похилої площини з кутом *a*. 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: $tng \sin a = ma$. Звідси $a = gsina$. Шуканий час знайдемо з рівняння:

$$0 \quad \frac{er^2 \ \sin or}{2}$$

И* і но праху вати дію сили тертя $F^T = kmg$ cos аг, де $к$ - коефіцієнт **tt**|»in. го можна уточнити початкову математичну модель:

$$mg \sin a - king \cos a = ma;\ a = g(\sin a - к \cos ar);$$

$$S = v^{Q}t + \frac{Я'^{2}(\sin a - \cos a)}{2} - .$$

У **ньому** разі розв'язок задачі буде більш точним. Процес уточнен- № мптематичної моделі можна продовжити, врахувавши ще силу **tMiopv** повітря.

На другому етапі відбувається пошук методу розв'язування сформульованої математичної задачі. Мета його - перейти від рів. **мміп.** математичної моделі до розрахункових формул, іцо **ний'**«чують вхідні дані з шуканими результатами. Якщо знайти **(Ніш** нзок з' вигляді формули не вдається, використовують чисельні мешди розв'язування задачі.

На третьому етапі розв'язування задачі подається у вигляді іик іідовності дій, формальне виконання яких приводить до необхідних результатів, тобто розробляється алгоритм розв'язування, **имііі** можна подати різними способами.

На четвертому етапі розроблений алгоритм записується у ви- І піді програми відповідною мовою програмування.

На п'ятому етапі здійснюється введення програми в пам'ять И)М і її налагодження та тестування. Налагодження - це пошук і виправлення синтаксичних помилок, які виникають внаслідок пор\ іисння правил мови програмування. Тестування - виявлення зміі'-юких (семантичних) помилок, які призводять до неправильних речудьтатів. Тестування здійснюється шляхом розв'язування контрольних (тестових) задач - виконання програми з такими вхідними дшніми, для яких відомо правильний результат.

На шостому етапі відбувається безпосереднє розв'язування іпдачі за допомогою ЕОМ - програма виконується з відповідними \ мові вхідними даними.

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

Зазначимо, що процес розв'язування прикладної задачі може містити й меншу кількість етапів, зокрема, якщо для реалізації моим і є готове програмне забезпечення. Тоді замість побудови алгоритму та програми буде етап, що передбачає вибір відповідного інструментального засобу у складі прогрманого забезпечення м шп'ютера.

3) побудова алгоритму розв'язування;
4) складання програми (запис алгоритму мовою програмування);
5) введення програми в пам'ять комп'ютера 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 дають змогу 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 із 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 $X = v^2/c^2$, де v - власна швидкість руху, ас -швидкість світла, результати, отримані в рамках обох моделей, практично збігалися. Всі ці моделі чудово витримали випробовування практикою, а їх роль у формуванні світогляду і в практичній діяльності людей важко переоцінити.

Використання математичних методів дослідження в інших науках також тісно пов'язане зі створенням математичних моделей.

Приклад. Розглянемо рух тіла маси ***m*** вздовж похилої площини з кутом *a*. 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: $mg \sin a = ma$. Звідси $a\text{-}gsina$. Шуканий час знайдемо з рівняння:

$$gt^2 since$$

Якщо врахувати дію сили тертя $F^T = kmg\cos a$, де κ - коефіцієнт іертя, то можна уточнити початкову математичну модель:

$$mg\sin a - hng\cos a = ma; \qquad a - g(\sin a - \kappa\cos a);$$

$$S = v^0 t + \frac{et^2(\sin a - \cos a)}{2}.$$

У цьому разі 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'язування, 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 пам'ять КОМ і її налагодження та тестування. Налагодження - це пошук і виправлення синтаксичних помилок, які виникають внаслідок порушення правил мови програмування. Тестування - виявлення змістових (семантичних) помилок, які призводять до неправильних результатів. Тестування здійснюється шляхом розв'язування контрольних (тестових) задач - 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.

Зазначимо, що процес розв'язування прикладної задачі може містити й меншу кількість етапів, зокрема, якщо для реалізації моделі є готове програмне забезпечення. Тоді замість побудови алгоритму та програми буде етап, що передбачає вибір відповідного інструментального засобу у складі прогрманого забезпечення комп'ютера.

Алгоритмічні мови. Мови програмування. Інтерпретація та компіляція. Розвиток мов програмування. Парадигми програмування

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, в якій визначено порядок і 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 налагоджувати;
- інтерпретовані програми простіше розробляти для виконання на різних типах комп'ютерів.

Розвиток мов програмування

Комп'ютер може опрацьовувати лише двійкові дані, тому для перших комп'ютерів програмістам доводилося писати програми у двійкових кодах - машинною мовою. Написання програми машинною мовою займає багато часу, у ній легко зробити помилку, причому знайти і усунути її дуже складно. Для полегшення створення програм була розроблена мова асемблера. В ній машинні команди подані мнемонічно, тобто символічними інструкціями, з якими людині працювати простіше, ніж з двійковими кодами. Наприклад, щоб завантажити у 16-бітовий регістр AX (внутрішній регістр мікропроцесора) десяткове число 157, програміст повинен написати команду MOV AX, 157. Порівняйте її з командою на машинній мові, яка виглядає так: 1011 1000 0000 0000 1001 1101. Коли програма на асемблері написана, її потрібно перетворити в програму на машинній мові. Це виконується автоматично спеціальною програмою, яка називається транслятором асемблера. Кожній команді асемблера в основному відповідає одна команда машинної мови, тому асемблер називають мовою низького рівня.

У подальшому тенденція полегшення для людини процесу створення програм лишалася домінуючою. В результаті були розроблені мови високого рівня, в яких програмні конструкції схожі на фрази англійської мови. Прикладами мов високого рівня є Basic, FORTRAN, Pascal. Більшість мов високого рівня універсальні, тобто призначені для розв'язування найширшого кола задач.

Створення мов високого рівня мало на меті також зробити програми незалежними від архітектури конкретного комп'ютера. Тоді їх можна було б без змін використовувати на різних комп'ютерах. На жаль, майже всі реалізації мов високого рівня не досягають цієї мети. Наприклад, програма мовою Pascal, написана для IBM PC, не буде працювати на комп'ютерах Apple Macintosh без ґрунтовної переробки вихідного коду, тобто початкового тексту програми мовою високого рівня. Мова Java, розроблена компанією Sun Microsystems, була першою комп'ютерною мовою, яка не залежала від архітектури комп'ютера. Програми на Java працюють однаково на будь-яких комп'ютерах, що підтримують мову Java, причому для цього у програми не треба вносити жодних змін.

Більшість мов високого рівня, які розроблено в останні роки, є об'єктно-орієнтованими. Це означає, що в них підтримується створення і застосування об'єктів. Прикладами об'єктно-орієнтованих мов є C++, Java, Object Pascal і Delphi. Концепція об'єктно-орієнтованого програмування наближає комп'ютерні програми до реального життя. В об'єктно-орієнтованих програмах, як і у реальному житті, створюються й існують об'єкти, над якими ви-

конуються певні операції. На основі цієї концепції створено велику кількість додатків у багатьох галузях людської діяльності.

У залежності від порядку виконання програм, мови високого рівня діляться на компільовані і інтерпретовані. Мови C++, FORTRAN і Pascal є компільованими.

Приклади інтерпретованих мов - Basic і JavaScript. Поділ мов на інтерпретовані і компільовані не є строгим. Краще було б їх назвати "схильними" до інтерпретації або компіляції. Наприклад, для мови Basic зараз існують як інтерпретатори, так і компілятори.

Парадигми програмування

Розвиток мов програмування відбувався за різними напрямками, пов'язаними з альтернативними підходами до процесу програмування (парадигмами програмування). Виділяють чотири парадигми програмування, які розвиваються незалежно одна від одної:

Імперативна (до • неї належать мови програмування FORTRAN, Basic, C, Ada, COBOL, ALGOL, APL, Pascal).

Декларативна (мови програмування GPSS, Prolog).

Функціональна (мови програмування LISP, ML, Scheme).

Об'єктно-орієнтована (SIMULA, C++, Ada 95, Smalltalk, Delphi, Java).

Імперативна або процедурна парадигма представляє традиційний підхід до процесу програмування. Саме у відповідності з цією парадигмою побудовано цикл опрацювання команди центрального процесора комп'ютера: "добути - декодувати - виконати". Як випливає з назви, імперативна парадигма визначає процес програмування як запис послідовності команд, для опрацювання даних, необхідної для отримання бажаного результату. Таким чином, для розв'язування задачі програміст повинен спочатку знайти алгоритм її розв'язання.

У протилежність цьому, **декларативна** парадигма ставить питання "Що являє собою задача?", а не "Який алгоритм потрібен для розв'язання задачі?" Основна проблема тут полягає у тому, щоб створити і реалізувати загальний алгоритм розв'язування задач. Після цього задачі можна формулювати у вигляді, сумісному з цим алгоритмом, а потім застосовувати його. В цьому випадку роль програміста полягає у точному формулюванні задачі, а не пошуках і реалізації алгоритму її розв'язування.

Основною проблемою в розробці декларативних мов програмування є вибір базового алгоритму розв'язування задач. З цієї причини ранні декларативні мови були вузькоспеціалізованими за своєю природою і орієнтованими на специфічні додатки. Наприклад, декларативний підхід уже багато років застосовується для

моделювання експертних систем (економічних, фізичних, політичних і т.п.) з метою перевірки висунутих гіпотез. У цьому разі базовий алгоритм, по суті, є процесом моделювання проходження часу шляхом багаторазово повторюваного обчислення значень параметрів (зростання внутрішнього продукту, торгового дефіциту тощо) виходячи з обчислених раніше значень. Використання декларативної мови для виконання такого моделювання зводиться, у першу чергу, до реалізації алгоритму, ***що*** виконує вказану повторювану процедуру. В результаті програмісту лишається описати взаємовідношення параметрів, що моделюються. Далі базовий алгоритм моделювання просто імітує хід часу, використовуючи вказані співвідношення для виконання потрібних обчислень.

Декларативна парадигма отримала нове застосування - завдяки усвідомленню того факту, що застосування методів математичної формальної логіки дає змогу створювати прості алгоритми розв'язування задач, які можна використати у системах декларативного програмування загального призначення. Так, в рамках декларативного підходу виникло так зване логічне програмування. Логічне програмування - це напрям у програмуванні, при якому логіка предикатів першого порядку використовується як мова програмування високого рівня.

Функціональна парадигма розглядає процес розробки програми як конструювання її з деяких "чорних шухляд", кожна з яких отримує певні вхідні дані і виробляє відповідний результат. Математики називають такі "шухляди" функціями, тому такий підхід називається функціональною парадигмою. Мовні конструкції функціональних мов програмування складаються з елементарних функцій, на основі яких для розв'язування поставленої задачі програміст повинен утворювати складніші функції. Отже, функціональний підхід до програмування базується на тій простій ідеї, що вся обробка даних і одержання шуканого результату можуть бути подані у вигляді вкладених і/або рекурсивних викликів функцій, які виконують певні дії, так що значення однієї функції використовуються як аргумент іншої. Значення цієї функції стають аргументами наступної і т.д. поки не буде одержано остаточний результат - розв'язок задачі.

Програми будують з логічно розчленованих визначень функції. Визначення складаються з управляючих структур, які організовують обчислення, і з вкладених викликів функцій. Основними методами функціонального програмування є композиція і рекурсія. Все це представляє собою реалізацію ідеї теорії рекурсивних функцій.

Перевага функціональної парадигми програмування над імперативною виявляється у тому, що вона стимулює модульний підхід до конструювання програм. Той факт, що програми розгляда-

ються як функції, які, в свою чергу, мають складатись з інших функцій, спонукає програміста мислити в термінах модулів. З цієї причини прихильники функціонального програмування стверджують, що цей підхід в порівнянні з імперативним призводить до створення досконаліших програм.

Об'єктно-орієнтована парадигма, яка передбачає застосування методів об'єктно-орієнтованого програмування (ООП) - це ще один підхід до процесу розробки програмного забезпечення. В рамках цього підходу елемент даних розглядається як активний "об'єкт", а не як пасивний елемент, як це прийнято в традиційній імперативній парадигмі. Пояснимо це на прикладі списку імен. В традиційній імперативній парадигмі цей список розглядається просто як сукупність деяких даних. Будь-яка програма, що отримує на вході цей список, повинна містити алгоритм виконання над ним потрібних дій. Таким чином, список є пасивним об'єктом, оскільки він опрацьовується управляючою програмою, а не опрацьовує себе сам. При об'єктно-орієнтованому підході список розглядається як об'єкт, що містить деяку сукупність даних разом з набором процедур для їх опрацювання. Цей набір може містити процедури для вставки у список нового елемента, вилучення елемента зі списку або сортування списку. Тому в програмі, яка отримує доступ до списку для його опрацювання, не потрібно записувати алгоритм для виконання вказаних дій. При необхідності вона просто виконує процедури, які надаються самим об'єктом. У цьому смислі об'єктно-орієнтована програма замість сортування списку (як при імперативній парадигмі) скоріше "просить" список відсортувати самого себе.

Як інший приклад використання об'єктно-орієнтованого підходу розглянемо задачу розробки графічного інтерфейсу користувача. В цьому разі всі відображувані на екрані графічні елементи реалізуються як об'єкти. Кожний з них містить власний набір процедур, що визначають реакцію об'єкта на виникнення певних подій, - вибір цього об'єкта, натиснення на ньому кнопки миші або перетягування його по екрану. Таким чином, система в цілому виглядає як сукупність об'єктів, кожен з яких знає, як реагувати на певну подію.

Багато переваг об'єктно-орієнтованого проектування є наслідком модульної структури, яка виникає як природний побічний ефект від застосування об'єктно-орієнтованого підходу. В рамках цього підходу кожний об'єкт реалізується у вигляді окремого, точно визначеного елемента. Після того як властивості деякої сутності будуть визначені подібним чином, отримане визначення можна повторно використовувати щоразу, коли виникає потреба в цій сутності. З цієї причини прихильники об'єктно-орієнтованого програму-

шіння стверджують, що дана парадигма надає природне середовище для конструювання програмного забезпечення з "будівельних блоків".

Ще однією перевагою модульної структури, яка отримана в результаті застосування об'єктно-орієнтованої парадигми, є те, що псі взаємодії модулів здійснюються шляхом пересилання повідомлень - основного способу взаємодії машин у комп'ютерних мережах. Таким чином, концепція об'єктів, які посилають один одному повідомлення, - це природний підхід до розробки програмного забезпечення, розподіленого в мережі. Реалізація такої мережевої системи об'єктів є метою технології CORBA (Common Object Request Broker Architecture - технологія побудови розподілених об'єктних додатків, запропонована компанією IBM) - відкритої системи специфікацій для реалізації механізмів передавання повідомлень між об'єктами в мережі.

Хоча розглянуті концепції названі парадигмами програмунання, вони мають відгалуження і за рамками власне процесу про- і рамування. Ці парадигми подають цілком різні підходи до методів розв'язування задач взагалі і тому впливають на всю галузь розробки програмного забезпечення. В цьому смислі більш правильно і оворити не про парадигми програмування, а про парадигми розробки програмного забезпечення.

Запитання для самоконтролю

I. Що називається алгоритмічною мовою, мовою програмування?
.?. Що таке алфавіт і синтаксис мови?
л *Що називається програмою?*
I. Для чого призначені інтерпретація і компіляція?
V *Як відбувається інтерпретація'?*
(>. Як відбувається компіляція?
/. Охарактеризуйте сучасні парадигми програмування.

Мова програмування Паскаль: основні поняття

Мову програмування Паскаль (Pascal) було розроблено швейцарським вченим Ніколасом Віртом для навчання програмування студентів університету. Але завдяки своїм властивостям ця мова з успіхом використовується також як практична мова для написання ефективних і надійних програм. 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 містить 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, ..
Спеціальні символи	> [ ] ( ) { } $ #
Знаки операцій	- (віднімання), + (додавання), * (множення), / (ділення), := (присвоювання)
Знаки відношень	< (менше), <= (не менше), (більше), >= (не більше), (дорівнює), о (не дорівнює), m (с елементом множини)
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), .. (задания діапазону), [J (для звернення до елементів масиву) : (для відокремлення міток) $ (шіетнадцятковий код), ^ (для задания 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 іранслятором як одне ціле, і завжди в одному розумінні.

Приклади. Begin, End, if, then, else, Array, for, to, do, of, 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. Значущими є перші 64 символи ідентифікатора.
5. Бажано, щоб ідентифікатор відображав зміст величини.

Приклади. Max3, a, ALPHA, M34, Zavdl_3.

U/уважения. В ідентифікаторах мови ТР великі і малі літери не /хирі знаються.

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

Поняття типу даних є одним з фундаментальних понять (>\дь-якої мови програмування. Об'єкти, якими оперує програма (константи, змінні, функції, вирази) належать до певного типу. **Тип** - множина значень, яких можуть набувати об'єкти програми, і су купність операцій над цими значеннями.

ТР є мовою з сильною системою типізації. Це означає, що всі іані, 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 індповідають конкретній задачі.

Приклади. Integer (цілі числа), Real (дійсні числа), Char (одиничні символи), Boolean (логічні значення - TRUE (істинне), FALSE (\ибне)).

Константа - елемент даних, значення**.^когл-ыдода'** виконання програми і в процесі її виконай^^^з^фрв^**еб**^Для^ризна-

17.г0г:ч:іяй універсал* .

ч ті* зла \

чення констант у ТР використовується службове слово Const (ві англ. Constant - константа).

Змінна - елемент даних, який може змінювати своє значе в процесі виконання програми. Дші визначення змінних у ТР вико ристовується службове слово Var (від англ. Variable - змінна).

Ідентифікатори і тили 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 вказують у розділі опису констант: * •

```
Const
    ідентифікатор=вираз;
```

Приклад.

```
Const
    Max=1000;
    Min=0;
    Interval=Max-Min+l;
    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 "істинне" або "хибне" - логічним (булевим).

Оператори

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

Розрізняють прості і складені оператори. Прості оператори програми на ТР відокремлюють крапкою з комою. Складений оператор - це послідовність простих операторів, взята в операторні

дужки - службові слова Begin (початок) і End (кінець). Перед слоном 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), який 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 бібліотек (модулів), які використовуються в програмі;
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 читанним і розуміння програм.

Турбо Паскаль дає змогу програмі (тексту) управляти режимом компіляції: включати або відключати контроль помилок, використовувати або емулювати математичний співпроцесор, змінювані розподіл пам'яті і ін. Для зміни режиму використовують ключі (директиви) компіляції: спеціальні коментарі, які містять символ "%" і літеру-ключ, за якою ставиться "+" (включити режим) або "-" (підключити). Можна об'єднувати ключі в один коментар.

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

{$N+} - використовувати співпроцесор 80X87 і т.д.
{$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 {$1 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. {$1 incproc.pas}

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

Запитання 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 операторы дужки?*
11. *Яку структуру має Паскаль-програма?*
12. *Що таке коментар? Для чого призначені коментарі? Як їх записують?*

Основні оператори мови Паскаль.

Оператор присвоювання

Оператор присвоювання призначено для надання значень імпшим.

Формат оператора:

ім'язмінної := вираз;

Тип змінної повинен збігатися з типом виразу. Винятком є иишідок, коли змінній дійсного типу присвоюється значення цілошильного виразу.

Приклад.

```
V.H A, B: Real; C, D: Integer; E: Char;
П'" i i n
  /\: '3.5; B:=-10; C:=4; D:=0; E:='*'; {правильні присвоювання}
  A:-(B-3*C)/(D+54); D:=(C+15)*3;  {правильні присвоювання}
  i':=A; D:=C/10; E:=B;                {неправильні присвоювання}
```

Оператори введення і виведення

Крім оператора присвоювання, змінним можна надати знамення за допомогою операторів введення.

Формат операторів:

Kca(i(список введення);

Rcadln(cnHCOK введення);

С'пиюквгдення=ім'я_змінна'1[,ім'я^імінної2,.. ,,ім'я_змінноїгі|

В квадратних дужках вказують необов'язкові параметри.

Виконання оператора:

Елементи списку введення вводять з клавіатури через проIIVCK або Enter. При використанні оператора Read після введення і >с і аннього елемента списку позиція введення (курсор) залишається и іому самому рядку. При використанні Readln позиція введення помішується на початок наступного рядка.

Список введення може бути відсутнім. В цьому випадку проірпма зупиняє свою роботу до введення з клавіатури будь-яких іаиих. 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:

\¥гпс(список виведення);

Writeln(cnHCOK виведення);

Список виведення = вираз 1 [, вираз2, виразів]

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.

Відмінність між операторами Write і Writeln аналогічна відмінності 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© (цілочисельний)	1	виводиться десяткове подання значення 1
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

Приклад. Дано довжину ребра куба. Знайти площу грані, площу повної поверхні і об'єм куба.

```
Uses Crt;
Var a:Real;
Begin
  ClrScr;
```

```
   Wi 11eln('Введіть довжину ребра куба:');
   l'>Mdln(a) ;
   Wiiteln('Площа грані s=',sqr(a):4:2);
   Wi iteln('Площа повної поверхні S=',sqr(a)*6:4:2);
   Wi i Г.eln('Об"єм V=',sqr(a)*a:4:2);
   м-peat until keypressed
Ii.• i.
```

Приклад. Дано дійсні числа а, ***Ь.*** Поміняти місцями значення \\\\\ чисел.

```
Пи,-.-;     Crt;
Veti л,b,c:Real;
И», i) n
   i' Ir Scr;
   Wi iteln('Введіть два дійсних числа :');
   Wt Lte('a=');
   l'iMdln(a) ;
   Wiite('b=');
   l'.Mdln(b) ;
   i•: -a; a:=b; b:=c;
   Wtiteln('a=',a:4:2,'   b=',b:4:2);
   icpeat until keypressed
```

Запитання для самоконтролю

I 11 я чого призначений оператор присвоювання? Вкажіть його формат і наведіть приклади використання.

' ,/(яя чого призначений оператор введення? Які існують його модифікації? Як вони виконуються?

** i\ля чого призначений оператор виведення? Які ісіеують його модифікації? Як вони виконуються?*

Стандартні типи даних та *операції* над ними

Стандартні типи даних описані в стандартній бібліотеці ТР, к к а автоматично підключається до кожної програми. Тому такі типи не потребують опису в розділі визначення типів Паскальнрограми. Всі допустимі в мові ТР типи поділяються на прості (скалярні, нсструктуровані) і складені (структуровані).

Таблиця 3. Стандартні скалярні типи

Тип	Позначення	Приклади
1 *[\:гі* числа	Integer	5 -3 0 +123
1 (мі довжиною 1 байт	Byte	2 67 125
їм >роткі цілі	Shortlnt	-127 25 100
Довгі цілі	Longlnt	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	1.75E+4537
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	(1.2,0.5, 1.7,2.2)
Записи	Record	(D:25; M:Jun; Y:2005)
Множини	Set	[2,3,5,7,11] [V, V, 'd']
Файли	File, File of	
Списки		
Черги		
Стеки		

Числа у TP подаються:

1. У десятковій системі числення (діапазон для цілих -2768..32767, для дійсних 1E-38.. 1E+38).
2. У 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
Shortlnt (коротке ціле)	-128..127	1 байт зі знаком
Word (ціле довжиною 1 слово)	0..65535	2 байти без знаку
Longlnt (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-^5..3.4E+38	7-8, 4 байти

limbic (подвійнаточність)	5E-324..1.7E308	15-16, 8 байтів
чі ended (зовнішні або «і шширені)	3.4E-4932..1.1E4932	19-20, 10 байтів

Змінні і константи всіх типів використовуються у виразах. Инраз задає порядок виконання дій над величинами і складається з шісрандів (констант, змінних, звернень до функцій), круглих дужок І іиаків операцій. Операції визначають дії, які треба виконати над операндами. Наприклад, у виразі (X+Y-10) X, Y і 10 - операнди, " і" **і**"-" - знаки операцій додавання і віднімання. У найпростішому випадку вираз може складатися з одної змінної або константи. Кру-**І м** дужки ставляться так, як у звичайних арифметичних виразах і **пі** управління порядком виконання операцій. Використання круг-***лих*** дужок навіть там, де в них немає необхідності з точки зору синтаксису, є прийнятним, якщо вони роблять порядок обчислень hi iv ально більш чітким і зрозумілим.

Таблиця 7. Операції для величин цілого типу

Назва	Призначення	Приклади
NOT	інверсія числа	NOT0 = -1; NOT-15= 14
Sill.	циклічний зсув вліво	3 SHL 2 = 12; 2 SHL 7 = 256
SI IK	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
і	додавання	5 +2 = 7; -3+2 = -1
	віднімання	И _7 = 4; -4-9 = -13
< Ж	логічне додавання	5 OR 2 = 7; 15 OR 7= 15
; <>; <; >; <=; >=	операції відношення	3 > 4 -> FALSE; 5 <= 8 -» TRUE

'Гпблиця 8. Функції для величин цілого типу

Назва	Призначення	Приклади
ЛИЭД	Знаходження абсолютного зна-	ABS(-48) = 48;
	чення величини	ABS(48) = 48
SQR(r)	1 [іднееееиня до квадрату	SQR(4)= 16;
		SQR(lT)= 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
=; o; <; >; <=; >=	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 (й0)	SQRT(2.89)=1.7; 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; RQUND(8.3) = 8; ROUND((-8.53) = -9
FRAC(r)	Знаходження дробової частини	FRAC(2.4)=0.4; KRAC(-2.4)=-0.4
SIN(r)	Знаходження значення синуса	SIN(2.48)=0.61437
COS(r)	Знаходження значення косинуса	C()S(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

1 Xl'(r)	Знаходження значення експоненціальної функції	EXP(2.48)=11.9412
IN Id)	Знаходження найближчого цілого числа (без округлення)	ШТ(2.98)=2.0; ПЧТ(-1.3)=-1.0

Таблиця 11. Операції для величин логічного типу

Назва	Призначення	Приклади
N( )T	заперечення	NOT TRUE = FALSE; NOT FALSE = TRUE
AND	кон'юнкція (логічне множення)	TRUE AND FALSE = FALSE; TRUE AND TRUE = TRUE
Ok	диз'юнкція (логічне додавання)	FALSE OR FALSE = FALSE; TRUE OR FALSE = TRUE

Таблиця 11. Операції для величин логічного типу

Назва	Призначення	Приклади
NOR	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
l'RED(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
()RD(L)	знаходження порядкового номера значення **L**	ORD(FALSE)=0; ORD(TRUE)=l

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(e)	порядковий номер символа **с** в заданому символьному наборі	ORD('A') = 65; 0RD(#4) = 4
CHR(i)	символ, що відповідає числу і згідно з кодами ASCII	CHR(65)='A'

PRED(c)	символ, що передує символу с	PRED('B') = 'A'
SUCC(c)	символ, що є наступним за символом с	SUCC('A')-B'
UPCASE(c)	символ верхнього регістру, відповідний 'a'..'z'	UPCASE(V)='A'

Запитання для самоконтролю

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 належать масив, запис, множина, файл, посилання, об'єкт.

Прості типи даних, в свою чергу, поділяються на упорядковані та неупорядковані.

Упорядкований тип даних характеризується тим, що довільне його значення має порядковий номер. Для будь-якого упорядкованого типу існує його найменше і найбільше значення.

Неупорядкованими у Турбо Паскалі є лише дійсні типи (множина значень дійсного типу не є упорядкованою). Решта простих типів є упорядкованими.

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

Type ім'ятипу = опис типу;

Ім'ятипу - ідентифікатор, опис типу надається у відповідності з синтаксисом мови.

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

Перелічувальний тип даних - простий упорядкований тип, кміїі визначає скінченний набір констант. Змінні цього типу моїм її. набувати значення лише з визначеного набору.

Формат опису перелічувального типу:

Type ім'ятипу = (константаЦ, константа2,... константагТ|);

Імена констант є ідентифікаторами.

Приклад.

```
Cy) > • Rainbow = (Red, Orange, Yellow, Green, IightBlue, Blue, Magenta);
v.) i A: Rainbow;
hi4\j n
  A: = Orange
tunl.
```

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 =; о; <; >; <=; >=, функції PRED, SUCC, ORD.

Приклад. PRED(Green) = Yellow; SUCC(Blue) = Magenta; ()KIXOrange)=l; ORD(Red)=0;

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

Приклад. A := Rainbow(3); {A=Green}

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

Діапазон - простий упорядкований тип, який визначає підмножину значень певного базового типу. Базовим може бути будь-чкий простий упорядкований тип, в тому числі і перелічувальний, иисдений програмістом.

Формат опису типу діапазон:

Type ім'я_типу = мінімальнезначення.. максимальнсзначсння;

Приклад.

```
i'ype Century20=1901..2000; {для цього типу базовим є
                             тип Integer}
      HotColors=Red..Yellow; {для цього типу базовим є
                             тип Rainbow}
v.ir C: HotColors;

1
1 :=Red; {правильне присвоювання}
 ':= Green; {помилка}
```

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

Запитання для самоконтролю

1. *Наведіть загальну класифікацію типів даних мови ТР,*
2. *Які характеристики мають прості типи даних?*
3. *Чим розрізняються упорядковані і неупорядковані типи даних?*
4. *Які характеристики мають складені типи даних?*
5. *Що таке перелічувальний тип даних? Як він описується в прогр 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? Наведіт приклади.*

Основні оператори мови Паскаль. Умовний оператор та оператор варіанту

При необхідності змінити порядок виконання операторі програми в залежності від виконання певних умов, тобто здійсниг розгалуження процесу обчислень, використовують умовний опера тор і оператор варіанту. При програмуванні розгалуження на дв гілки доцільно використовувати умовний оператор (його також на зивають оператором розгалуження), на більшу кількість гілок оператор варіанту.

Умовний оператор застосовується в двох формах - повній [1] скороченій.

Формат скороченої форми оператора:

if логічнийвираз then операторі

if (якщо), thcn (То) - службові слова, операторі -довільний оператор мо ТР.

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

Формат повної форми оператора:

if логічнийвираз then операторі else опсратор2

Виконання оператора, обчислюється значення логічного ви-разу. Якщо воно дорівнює TRUE, виконується операторі, якщо FALSE - виконується опсратор2, після чого відбувається перехід до виконання наступного оператора програми. Операторі і опера-

W»p.' можуть бути як простими, так і складеними. Перед службовим інпшм else крапка з комою не ставиться.

Приклад. Розв'язування квадратного рівняння.

```
Vet  л, b, c, D, xl, x2 : Real;
    iu
    Writeln('Введіть коеф-ти рівняння a,Ь,c:');
    I'.cadln(a,b, c) ;
    И:=Ь*Ь-4*a*c;
     if D>=0 then
       Begin
            xl:=(-b-sqrt(D))/(2*a);
            x2:=(-b+sqrt(D))/(2*a);
            Writeln('xl=',xl:4:2,' x2=',x2:4:2)
       End
     else
       Writeln('Дійсних коренів немає');
    Headln
Кім i.
```

Приклад. Дано дійсне число ***a.*** Обчислити ***/(a),*** якщо ***/(a)*** =0 при *й^1 li./(a)=a* при *0<a<1* і ***f(a)^a^4*** у протилежному випадку.

```
KBIV;    Crt;
V«i  a, f:Real;
Йпі.I i n
  i' IrScr;
  Wi iteln('Введіть дійсне число:');
  Write('a=*);
  Ki'ddln(a);
  i I  a<=0
     then f:=0
     else  if  a<=l
               then  f;=a
               else  f:=a*a*a*a;
  Writeln('f(a)=',f:4:2);
  M-peat  until  keypressed
I E i.
```

Оператор варіанту дає змогу виконувати ті чи інші дії в зале-
* пості від значення виразу упорядкованого типу.

Формат оператора:

```
case вираз_упорядкованого_типу of
  список1: оператор1;
  список2: оператор2;

  cnncoKN: операторів
(else
   onepaTopN+1]
i'.nd;
```

Case**(ВаріаНТ),** ог(з) - службові слова, список 1 - перелік зна-
чінь виразу_упорядкованого_типу, розділених комами, операторі

- довільний оператор мови TP. Вираз упорядкованого типу нази вається селектором.

Виконання оператора: обчислюється значення виразу, якщ воно входить до списку 1, виконується операторі і відбувається пе рехід до виконання наступного оператора програми, якщо до спис ку2 - оператор2 і т.д. Якщо значення виразу не входить в жоде список, то при наявності частини else виконується операТОрN+1, при її відсутності - виконання оператора варіанту завершується.

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

```
Var c: Char;
Begin
  Writeln('Введіть символ:');
  Readln(c);
  case c of
     'O'..'9' : Writeln('Цифра');
     'aVeViVo'/uVy' : Writeln('Голосна');
     'bV.'oVf..•h','j'..'rJ,'p'..'t', V..'x','z' : Writeln ('Гршэгосна');
       else Writeln('Інший символ')
     End;
End.
```

Приклад. Вивести назву дня тижня за його номером.

```
Uses Crt;
Var drbyte;
Begin
  ClrScr;
  Write('Введіть номер дня тижня: ');
  Readln(d);
  case d of
     1: Writeln('Понеділок');
     2: Writeln('Вівторок');
     3: Writeln('Середа');
     4: Writeln('Четвер');
     5: Writeln('ГГ'ятниця');
     6: Writeln('Субота');
     7: Writeln('Неділя');
     else Writeln('Неправильний номер дня!')
  End;
  repeat until keypressed
End
```

Запитання для самоконтролю

1. *Які оператори мови TP 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. Оператори циклу

Цикл у програмуванні - це повторюване виконання одних і »н\ самих простих або складених операторів. У Паскалі реалізо-**нтіі** іри способи організації циклічних обчислень.

Оператор циклу з передумовою

Формат оператора:

while логічнийвираз do оператор;

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

Виконання оператора: обчислюється значення логічного вири **іу,** якщо воно дорівнює TRUE, виконується оператор і відбува-**і п.оі** повернення до умови. Процес продовжується доти, поки значения логічного виразу не стане дорівнювати FALSE. Після цього виконання циклу закінчується.

Якщо при першому обчисленні значення логічного виразу Пою значення дорівнює FALSE, тіло циклу не виконується жодно-**ііі** разу.

Зауваження.

I 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.

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

Приклад. Обчислити значення £ —~ з точністю до

```
'••'ii.yt eps=lE-6;
v.ir  n: Integer;     {сумування проводиться до тих пір,}
    x,S: Real;       {поки черговий доданок не стане меншим}
H'-<(in              {заданої точності обчислень}
   n:=l;  S:=0;  x:=1;
   while abs(x) >= eps do
        Begin
            S:=S+x;
            n:=n+l;
            x:=l/sqr(n)
        End;
   Writeln('S=', S)
Mid.
```

Оператор циклу з післяумовою

Формат оператора:

repeat оператор until логічнийвираз;

repeat (повторювати), until (до) - службові слова; оператор - до вільний оператор мови ТР, в тому числі порожній. Складений ода ратор брати в операторні дужки немає потреби.

Виконання оператора: виконується тіло циклу, потім обчио люється 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 післяумо вою завжди виконується принаймні один раз. Це треба враховуваті при виборі оператора для виконання циклічних обчислень.

Для циклу з післяумовою також справедливе зауваження 1.

Приклад. Фрагмент програми, в якому від користувача вимага ється 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:=l; S:=0; x: =1;
   repeat
     S:=S+x;
     n:=n+l;
     x:=l/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;

Приклад. Дано ціле число ***к,*** дійсне число от. Вивести на екран перші ***к*** членів послідовності ***a„-sm(n),*** більших за ***m,*** та їх номери.

```
Uses Crt;
Var k,kO,n:Integer;
    a,m:Real;
Begin
```

```
  '•IrScr;
  Wi iteln(1Введіть кількість членів послідовності:');
  Hr-adln (k) ;
  Wi iteln('Введіть число m:');
  lo'ddln (m) ;
  n: 0;       {нумерація членів послідовності}
  ki):=0;   {лічильник членів послідовності, більших за т}
  while k0<=k do
    Begin
      a:=sin(n);
      if a>m then {член послідовності задовольняє умо-
ву I
        Begin
          k0:=k0+l;
          Writeln(kO:4,1    ','a(',n,1)=',a:4:2)
        End;
      n:=n+l {наступний член послідовності}
    End;
   repeat until keypressed
Миі.
```

Приклад. Дано натуральне число ***n.*** Знайти:

а) кількість цифр у числі и;

б) суму цифр числа ***n;***

в) першу цифру числа ***n.***

```
Il:ii;s Crt;
v.ir  n, c, nl: Longlnt;
    dig,sum:Integer;
    first:Integer;
It' 'gin
  ClrScr;
  Write(1n=1); Readln(n); nl:=n; {збереження початкового
                                  значення числа n}
 I знаходження кількості цифр у числі }
  c:=0;
  repeat
    c:=c+1;
    n:=n div 10 {число без останньої цифри}
  until n=0;
  Writeln('Число складається з ',c,' цифр');
 { знаходження суми цифр числа }
  n:=nl;
  sum:=0;
  while n>0 do
    Begin
      dig:=n mod 10; {остання цифра числа}
      sum:=sum+dig;
      n:=n div 10
    End;
  Writeln(1Сума цифр числа = ',sum);
 { Знаходження першої цифри числа }
```

```
    n:=n1;
    first:=0;
    while n>0 do
      Begin
        dig:=n mod 10;
        n:=n div 10
      End;
    first:=dig;
    Writeln('Перша цифра числа = ',first);
    repeat until keypressed
End.
```

Приклад. Обчислити суму $1-\frac{1}{2}+\frac{1}{3}-\ldots+\frac{1}{99}-\frac{1}{100}$

```
Uses Crt;
Var i,w:Integer;
      S:Real;
Begin
  ClrScr;
  S:=0;      {початкове значення суми)
  i:=1;      {знаменник першого доданку]
  w:=1;      {чисельник першого доданку}
  while i<=100 do
    Begin
        S:=S+w/i;
        w:=-w;
    End;
  Writeln('S=',S:6:4);
  repeat until keypressed
End.
```

Оператор циклу з параметром

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

Формат оператора:

for параметр_циклу:= виразі to вираз2 do оператор;

for (для), to (до), do (виконувати) - службові слова, пара мстрциклу (або лічильник циклу) - ім'я змінної упорядкованого типу, вираз1 і вираз2 - вирази, тип яких збігається з типом пара метра 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 - значений виразу 1+1

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

Різниця між двома сусідніми значеннями параметра циклу ни іивається **кроком циклу.**

Крок циклу з параметром завжди постійний і залежить від і ні іу параметра.

Розглянутий формат оператора циклу використовують, коли шпчення параметра циклу зростає. Якщо значення параметра циклу **t** ішдає, використовують модифікований формат:

for параметр_циклу := вираз2 downto виразі do оператор;

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

Приклад. Вивести у порядку спадання коди латинських літер.

```
v.t r  ch  :Char;
Иг-qin
    for ch:=  1 z' downto 'a' do
       Writeln(ch, ' - ', ord(ch))
кiid.
```

Приклад. Дано натуральне число ***n.*** Обчислити добуток перших *n* множників виду:

а) $1\frac{3}{2}\frac{5}{3}\ldots$; б) $\frac{1}{2}\frac{3}{4}\frac{5}{6}\ldots$.

```
ily.es     Crt;
v.ir  n: Integer;  i:Byte;
     a,b:Integer;  P:Real;
c.' -gin
   ClrScr;
   Writeln('Введіть кількість множників:  ');
    Readln(n);
    (а) ]
    P:='l;              (початкове значення добутку)
    a:=l; b:=l;         (початкові значення чисельника і
                         знаменника множника}
    for i:=1 to n do
        Begin
             P:=P*(a/b);
             a:=a+2;  b:=b+l
        End;
    Writeln('Добуток =  ',P:6:2);
      (б) }
    P:=l; a:=l;
    for i:=l to n do
        Begin
```

```
            P:=P*(a/(a+1));
            a:=a+2;
        End;
    Writeln('Добуток =      ',9:6:2);
  repeat until keypressed
End.
```

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

Таблиця 15. Циклічні оператори ТР

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 хоч б один раз**
FOR...TO...DO	**відома кількість повторене**	**тіло циклу може не виконатись жодного разу; крок циклу фіксований**

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

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

Приклад. Обчислити суму $\sum_{i=1}^{n} \frac{i}{(2i+1)!}$ для заданого ***n.***

```
V.n  i,n:  Integer;
     S:Real;
Br-. Ii.n
     Readln (n);
     S:=0;
     for i:=l to n do
         S:=S+i/(2*1+1);
     Writeln('S=',  S:6:2)
find.
```

Приклад. Обчислити суму $\sum_{n=1}^{n^2+n}$.

```
v.it    S:Real;
        n, w. Integer;
M<-qin
   S:=0;
   w:=-l;  {чисельник першого доданку}
   for n:=l to 2 0 do
       Begin
           S:=S+w/(n*(n+1));
            w:=-w      {зміна знаку доданка)
       End;
 Writeln('S=',S:6:3)
Kiid.
```

Приклад. Дано ціле число а, натуральне число ***n.*** Обчислити:

а) ***a"***

6J ***a(a*** і ***I)...(a*** і и-іУ

```
n.ses  Crt;
v.ir  a:Integer;  n:Byte;
      P:LongInt;  i:Byte;
Hi'gin
   ClrScr;
   Writeln('Введіть ціле а, натуральне n: ');
     Readln(a,n);
  (a) }
     P:=l;
     for i:=1 to n do
         P:=P*a;Л
     Writeln('a n=', P);
  16) }
     P:=l;
     for i:=0 to n-1 do
         P:=P*(a+i);
     Writeln('a(a+1)...(a+n-1)=  ',  P);
   repeat  until  keypressed
к nd.
```

Приклад. Дано дійсне число ***x,*** натуральне число ***n.*** Обчислити:

а) sinx+sinr^2+...+sinx"

б) sinx+sin smx+... +sin ... sinx
'——————и——————'

```
Uses Crt;
Var x:Real; n:Byte; S:Real;
    a:Real; i:Byte;
Begin
  ClrScr;
  Writeln(1 Введіть дійсне x, натуральне n: ');
  Readln(x,n);
 {a)}
  S:=0; a:=x;
  for i:=l to n do
      Begin
           S:=S+sin(a);
           a:=a*x
      End;
   Writeln('Sl=', S:6:2);
 (6))
  S:=0; a:=sin(x);
  for i:=l to n do
      Begin
           S:=S+a;
           a:=sin(a)
      End;
   Writeln('S2=', S:6:2);
  repeat until keypressed
End.
```

Табулювання функцій

Проводячи обчислювальні експерименти, часто виникає не обхідність протабулювати певну функцію.

Табулювання функції ***f(x)*** на відрізку ***\a,Ь*** \ з кроком ***h*** поляга у побудові таблиці значень функції ***f(x)*** в точках ***x-a, x-a+h x+2h,..., x~b:***

X	*f(x)*
a	*/Га)*
a+h	*.ffa+h)*
b	***f(b)***

Функція ***/fx)*** має бути визначена на відрізку ***\a,Ь\.***

Відрізок ***[a, Ь]*** називається відрізком табулювання, ***h*** - кроко табулювання, точки ***a+hi (i~0J,2,...)*** - вузлами табулювання, від різки ***[a+hi, a+h(i+l)*** \ - відрізками розбиття.

Може статися, що ***Ь-а*** не ділиться націло на ***h,*** тобто точка ***b*** («• < пузлом табулювання, наприклад при ***a=1,*** 6=5, ***h=1.5.***

Якщо така ситуація небажана, задачу формулюють інакше: Днію відрізок ***[a,b] (a<Ь),*** функція ***f(x),*** натуральне ***n*** (кількість від- ii litів розбиття).

Іобудувати таблицю значень функції ***f(x): y^f(x^),*** де ***Xr-a+(b-a)i/n, I<). ï,...,n.***

Приклад. Протабулювати функцію ***y(x)=x^2+sm(x)*** на відрізку ((/,/> I. якщо задана кількість відрізків розбиття.

```
Vлi  x,y,a,b,h  :  Real;
     n:Byte;
tii"]in
     Write('Введіть a,b(a<b)');
     Readln(a,b);
     Write('Введіть  n>0');
     Readln(n);
     h:=(b-a)/n;  {обчислення значення кроку табулювання}
     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
KikJ.
```

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

```
V.ir  x,y,a,b,h,k:  Real;
  -qin
    Write('Введіть  a,b  (a<b):');  Readln(a,b);
    Write('Введіть крок h>0:');  Readln(h);
    Write('Введіть  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)
K.nd.
```

Приклад. Дано ***a,b (a<b),*** натуральне ***n,*** функція ***yfxj^x^5.*** Для i, ***a*** і ***ih (i-0,L...,n), h~(b-a)/n*** обчислити значення функції ***y(xj (i 0,1... n).***

```
' ir 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 цик лу, може сам бути оператором циклу або містити у собі операто циклу. Утворена конструкція називається вкладеним циклом.

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

```
Var   S:   Real;
        i,j:   Integer;
Begin
    S:=0;
    for i:=1 to 10 do
        for j:=1 to 5 do
            S:=S+1/(i+j*j);
   Writeln('S=*,S:6:3)
End.
```

Запитання для самоконтролю

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

Структури даних. Масиви

Особливістю мови Паскаль є вимога чіткого опису всіх нО г ктів, які використовуються в програмі. Так, блок описів профіти відокремлений від виконуваного блоку, у блоці описів відокремлені розділи опису констант, типів, змінних тощо. Кожний ***ч*** іемснт даних (константа чи змінна) повинен бути описаний перед використанням. Кожна підпрограма (процедура чи функція) має П\ їм 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 осо-Гшм, Паскаль вимагає чіткого структурування програми - щоб та чи інший опис знаходився у визначеному для нього місці.

Особливу роль відіграє структурування даних. У програмі шдаються правила обробки даних. Отже, поки не визначені самі піні, неможливо вдало розробити правила їх обробки.

Для спрощення написання і виконання програми окремі дані чисто буває зручно об'єднувати в певні структури. Від того, наскі-**ш**.ки вдало будуть вибрані ці структури, суттєво залежить ефективність програми.

В Паскалі та чи інша структура даних задається за допомо-і оіо певного типу даних.

Прості типи даних задають тривіальні структури даних - окреме значення.

Складніші структури даних задаються за допомогою складених (структурованих) типів.

Значення складеного типу в загальному випадку є нетривіа-п.ною структурою, тобто містить більш ніж одну компоненту. При цьому кожна компонента структури може бути значенням як просюго, так і складеного типу.

Найбільш уживаним складеним типом даних є регулярний піп, 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, **іа** якою можна звернутися до цього елемента.

Формат опису масиву у розділі опису типів:

Туре ім'ямасиву = Array[список типів індексів] of базонний_тип;

Array (масив), of(з) - службові слова, ім'я_масиву - ідентифік тор, тип індексів - діапазон (підмножина значень простого упоря дкованого типу).

Приклад.

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

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

Якщо базовим типом масиву є інший масив, утворюєтьс 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 безпосере ньо при описі змінних.

Елементи масиву Matr: Matr[l,l], Matr[2,3], Matr[4,4] (всьо 16 елементів)

Для звернення до окремого елемента масиву треба вказа ім'я масиву і в квадратних дужках індекси елемента. Елемент о новимірного масиву має один індекс, багатовимірного - стіль індексів, яка вимірність масиву.

Двовимірні масиви широко використовуються для подаг- матриць.

У загальному випадку матриця записується так:

$$\begin{matrix} a^{n} & a^{n} & \dots & a^{Xn} \\ '21 & '22 & & '2n \\ {}^{4}{}_{M1} & *_{T2} & & \end{matrix}$$

У ТР така матриця задається масивом типу Аггау[/..от, ***1..n]*** о 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, V<>W.

Решта операцій над масивами виконується поелементно.

Приклад. Надання значень елементам масиву:

```
,i)  V[l] :=10;  V[2]:=20;  V[3]:=45.7;
Г)   for i:=1 to 3 do V[i]:=0;
ії)  for i:=l to 3 do Readln(V[i]);
```

Приклад. Описи масивів.

```
Type
  Month = (January, February, March, April, May);
  DaysInMbntn = 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;
       Vectorl: Arrayf-10..10] of Byte;
Begin
   Days[January]:=31;
   Spring[April]:=30;
   Alpha['A']:='l';
   Vector[10]:=0.5;
   Matrix[1,1]:=4;
   Vectorl[-5]:=0
Knd.
```

Загальна схема розв'язування задач на опрацювання масивів:

I) введення масиву (надання елементам масиву початкових значень);

?.) опрацювання елементів масиву;

.<) виведення результатів.

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

Приклад. Дано масив *і 10* дійсних чисел. Підрахувати суму, іобуток і середнє арифметичне елементів масиву.

```
• Hist maxV=10;
i'ype mas = Array[1. .maxV] of Real;
```

```
Var  v: mas;          {масив}
     D,S,A: Real;     {добуток, сума, середнє арифметичне}
     i:Byte;          {лічильник}
Begin                                                          i
   {введення елементів масиву}                                 j
  Writeln('Введіть ', maxV, ' чисел:');                        I
  for i:=1 to maxV do                                          j
     Begin
        Write('V[',i,']=');
         Readln(v[i])
     End;
   {опрацювання елементів масиву}
  D:=l; S:=0; A:=0;
  for i:=l to maxV do
    Begin
      D:=D*v[i];
      S:=S+v[i]
    End;
  A:=S/maxV;
WritelnC Суш = ',S:4:1, 'добуток = ',D:4:1, 'сер. арифм. = '
A:4:l);              ' •
End.
```

Приклад. Дано масиви ***A, B*** (кожний з ***10*** цілих чисел). Побудувати масив С за правилом: с,— ***а? + Ь?, 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:=l to 10  do
      Begin
         Write('A[',i,']=');
          Readln(A[i])
      End;
   for i:=l to 10  do
      Begin
         Write('B[',i,']=');
          Readln(B[i])
      End;
  {формування масиву C}
  for i:=l to 10 do
     C[i]:=sqr(A[i])+sqr(B[i]);
  {виведення масиву C}
  Writeln('Масив C:');
  for i:=l to 10 do '
     Write(C[i]:3);
End.
```

Запитання для самоконтролю

1. *В чому полягає структуризація програм і даних у ТР?* \

.' Що таке тривіальні і нетривіальні структури даних?
I 1 Що називається масивом?
I ІЩо таке базовий тип масиву, індекс масиву?
\ Як описуються масиви у програмі на ТР?
(> If (о таке вимірність масиву? Як описуються багатовимірні масиви? 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.

```
ГУГ". mas=Array[1..15] of Real;
V.ii A: mas; N, i: Byte; Max: Real;
Ычііп
 Іпнедення масиву A]
Вч'Іin
  (or i:=l to 15  do
    Begin
      Write('A[ 1 ,i, ']=');
      Readln(A[i])
    End;
 I знаходження максимуму}
  Max:=A[1];
  lor i:=2 to 15 do
     if A[i]>Max then Max:=A[i];
 (знаходження кількості максимальних елементів}
  II: -0;
  lor  i:=l to 15 do
     if A[i]=Max  then N:=N+1;
  Wi Lteln('іуакс. ел-т ', Max:4:1, 'зустрічається у масиві ', N
         ' разів');
Ни I.
```

Приклад. Упорядкувати за зростанням масив з ***n*** дійсних чисел мі-годом "бульбашки".

```
' "lust n=10;
ГУГ"- mas=Array[1..n] of Real;
"и  a: mas;  c:Real;  i,j:Byte;
п' •• Iin
 Іпнедення масиву}
  lor i:=l to n do
     Begin
       Write('a[',i,']=');
       Readln(a[i,j])
     End;
 I /порядкування масиву}
  lor i:=l to n-1 do   {фіксація елемвнтів з 1-ї по n-1 позицію}
```

```
    for j:=i+l to n do {перегляд елементів, що стоять праворуч}
      if a[i]>a[j] then   {від зафіксованого}
        Begin
          c:=a[ij;        {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 ***x, y, z,*** - дійсні вектори заданої розмірності ***n.*** тому з векторів ***x, y*** і ***z,*** який має найбільш}' кількість від'єм* елементів (вважати, що такий вектор один), замінити всі дода елементи на їх куби, якщо це вектор ***x*** або вектор ***z,*** і на їх оберне значення, якщо це вектор ***y.***

```
Uses Crt;
Const n=5;
Type vector=Array[1..n] of Real;
Var x, y, z-.vector;
    i:Integer; max:Real;
    k:Array[1..3] of Real;
Begin
 {введення векторів}
  Writeln ('Введіть ',n,' елементів вектора x');
  for i:=l to n do
     Readln(x[i]);
  Writeln ('Введіть ',n, ' елементів вектора y');
  for i:=l to n do
  Readln(y[i]);
  Writeln ('Введіть ',n,' елементів вектора z');
  for i:=l to n do
     Readln(z[i]);
 {створення масиву з кількостей від"ємних елементів кожного векго
  for i:=1 to 3 do   k[i]:=0;
  for i:=1 to n do
    Begin
       if x[i]<0 then k[l]:=k[l]+l;
       if y[i]<0 then k[2]:=k[2]+l;
       if z[i]<0 then k[3]:=k[3]+1
    End;
 {знаходження максимальної кількості від'ємних елементів }
  max:=k[1];
  for i:=2 to 3 do
    if kfi]>max then max:=k[i];
 {перетворення векторів}
  if max=k[l] then {макс. кількість від'ємних елементів у векторі x
    for i:=1 to n do
```

```
      if. x[i]>0 then x[i]:=x[i]*x[i]*x[i];
  if mix- k[3] then { макс. кількість від'ємних елементів у векторі z}
    /' >) i:=1 to n do
      if z[i]>0 then z[i]:=z[i]*z[i]*z[i];
  it mix=k[2] then { макс. кількість від'ємних елементів у векторі y}
    Тої i:=1 to n do
      if y[i]>0 then y[i]:=1/y[i];
 Іиміч-дення перетворених векторів}
  Wi i teln ('нові елементи вектора x:');
  !"t i:=1 to n do
    Write(x[i]:5:1);
  Wi iteln;
  Wiiteln ('нові елементи вектора y:');
  f к i i:=1 to n do
    Write(y[i]:5:1);
  Wi iteln;
  Wi iteln ('нові елементи вектора z:');
  I'H i:=l to n do
    Write(z[i]:5:1);
  ii-peat until keypressed
Hud.
```

Приклад. Дано натуральне число *n,* цілі числа a^h a^2,..., a^n. Числа \ послідовності можуть повторюватись.

а) отримати всі числа, які входять у послідовність по одному разу;

б) з'ясувати, скільки чисел входять у послідовність по одному разу.

```
Пч...>;     Crt;
r,,list n=20;
v.ii a: Array[l..n] of Integer;
    i, j, P, S: Integer;
Hi-qin
  i'IrScr;
  Writeln ('Введіть ',n,' цілих чисел: ');
  lor i:=l to n do
   Readln(a[i]);
 la) }
   for i:=l to n do  {розгляд кожного елемента послідовності}
      Begin
         P:=0;      {кількість елементів, рівних і-му елементу}
        for j:=1 to n do
            if  a[i]=a[j]
               then P:=P+1;
             if P=l     {і-й елемент зустрічається 1 раз}
                then Write(a[i]:6)
      End;
 (6) }
  S:=0;          {кількість елементів, що зустрічаються 1 раз}
   for i:=1 to n do
      Begin
```

```
            P:=0;
            for j:=1 to n do
                if  a[i]=a[j]
                    then P:=P+1;
                if P=l then S:=S+1
        End;
   Writeln('Кількість елементів, що входять у послідовність
            раз, =',S);
  repeat until keypressed
End.
```

Приклад. Дано натуральне число ***n,*** цілі числа a^b *a^..., a^2 Ьі,...,Ь„* . Серед а, і ***bj*** немає рівних чисел.

а) чи правильно, що всі члени послідовності ***a*** входять у послід вність ***b***

б) чи правильно, що всі члени послідовності ***b*** входять у послід вність ***a?***

```
Uses Crt;
Const n=30;
Type vect_a=ArrayГ1..25] of Integer;
     vect_b=Array[1:.n] of Integer;
Var a:vect_a; b:vect_b;
    i,j:Byte; S:Integer; P:Boolean;
Begin
  ClrScr;
  Writeln ('Введіть 25 цілих чисел: ');
   for i:=1 to 25 do
      Read(a[i]);
  Writeln ('Введіть ',n,' цілих чисел: ');
   for i:=l to n do
      Read(b[i]); Writeln;
 {a)}
   S:=0;   {кількість членів 1-ї поел., що входять до 2-ї}
   for i:=1 to 25 do
       Begin
          P:=FALSE; {припущення, що і-й член 1-ї послідовності}
          for j:=l to n do          {не зустрічається у 2-й}
            if a[i]=b[j]           {спростування припущення}
               then P:=TRUE;
            if P thai S:=S+1 {киькісгь чгшв Ш пості., що ввдшь до 2-ї
       End;
   if S=25                 { всі члени 1-ї поел, входять до 2-ї
      then Writeln ('Правильно, що всі члени 1-ї послідовн
                    ті входять до 2-ї')
      else Writeln ('Неправильно, що всі члени 1-ї послідо
                    входять до 2-ї')
 (6) }
    S:=0;
   for i:=1 to n do
     Begin
```

```
        P:=FALSE;
        for j:=1 to 25 do
          if b[i]=a[j]
            then P:=TRUE;
          if P then S:=S+1
      End;
    if S=n
      then Writeln ('Правильно, що всі члени 2-ї послідовності
                     входять до 1-ї')
      else Writeln ('Неправильно, up всі члени 2-ї послідовності вхо-
                     дять до 1-ї')
    repeat until keypressed
Mud.
```

Приклад. Дано натуральне число ***n,*** цілі числа a_1, a_2,..., a_n. Всі члени послідовності з парними номерами, що передують першому ш порядком значенню тах(**o7,..**.,«„), домножити на max(a;,...,a„).

```
lines Crt;
<Y>nst n=20;
V.ir a: Array[l..n] of Integer;
    i, imax, max: Integer;
Mi'gin
  ClrScr;
  Writeln ('Введіть ',n,' цілих чисел: ');
  for i:=l to n do
   Readln(a[i]);
 (• знаходження максимального елемента і його індекса}
   max:=a[1]; imax:=1;
   for i:=2 to n do
     if a[i]>max
       then
          Begin
             max:=a[i]; imax:=i
          End;
 I перетворення послідовності}
   i:=2;
   while Kimax do
      Begin
         a[i]:=a[i]*max;
         i:=i+2
      End;
   Writeln('Перетворена послідовність:');
   for i:=l to n do
      Write (a[i]:4);
   repeat until keypressed
I'-.n.J.
```

Приклад. Дано квадратну матрицю розмірності ***4.*** Обчислити і іід матриці - суму елементів її головної діагоналі.

```
• • 'listn=4;
i'ype mas=Array[1..n, 1..n] of Real;
'її 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:=l to n do  {цикл по стовпцях}
         Read(a[i,j]);    {введення елемента рядка}
       Writeln;           {перехід до нового рядка}
    End;
 {знаходження сліда матриці}
  trace:=0;
    for i:=1 to n do
       trace:=trace + a[i,i];
  Writeln('Trace(A)=', trace:4:l)
End.
```

Приклад. Дано цілочисельну матрицю розміром *m m* . Знай 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[1,1J; imin:=l; jmin:=l;
  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 поря ку ***n (n<6).***

```
Type mas= Array[1..6,1..6] of Byte;
Var e:mas;  n,i,j:Byte;
Begin
  Write(1 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]:=I
              else e[i,j]:=0;
```

```
 I ниведення матриці E}
  Writeln('Матриця E:);
   I or i:=1 to n do
     Begin
       for j:=1 to n do
          Write(e[i,jj:2) ;
       Writeln;
     End;
»;IHI.
```

Приклад. Дано дійсне число ***n.*** Сформувати і надрукувати квадратну матрицю розмірності ***n,*** для якої $a_n=0$ при ***i=J*** і а,у=ї+2/ в іншому разі.

```
tin.-;; Crt;
i""ti.'-;t   n=5;
ïVI>c   matr=Array[l. .n, 1. .n]  of Real;
V.ij  a:  matr;
     i,j:Integer;
  1,in
   Irscr;
  Writeln('matrix');
  lor i:=l to n do
    Begin
      for j:=1 to n do
        Begin
          if i=j  then  a[i,j]:=0
                  else  a[i,j] :=i+2*j;
          Write(a[i,j]:5:0);
        End;
        Writeln;
     End;
  tupeat  until  keypressed
```

Приклад. Дано натуральне ***n.*** Отримати квадратну матрицю по-

ридк>' ***n:***

'1 ... 0 ... I

1 1 0 1 1

1 1 0 1

1 • • • o ...

```
n-..-.s  Crt;
> • 'listn=10;
'Jn   a:Array[1. .n, 1. .n]  of Integer;
     i,j:Byte;
h'-'l Ln
  '' LrScr;
 (формування  матриці}
  lor i:=l to n do
     for j:=1 to n do
```

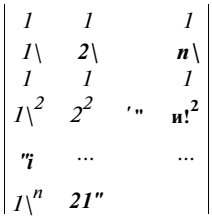
```
        if ((i>=j) and (i+j<=n+l)) or ((i<=j) and (i+j>=n+l))
            then a[i,j]:=1
            else a[i,jj :=0;
 {виведення матриці}
  for i:=l to n do
    Begin
        for j:=1 to n do
          Write(a[i,j]:2);
        Writeln;
    End;
  repeat until keypressed
End.
```

Приклад. Дано натуральне ***n***. Отримати квадратну матрицю по. рядку ***n:***

n	...	*0*	...	*0*
n-1	***n***	*0*	...	*0*
n-2	***n - 1***	***n***		
1	*2*	*3*		

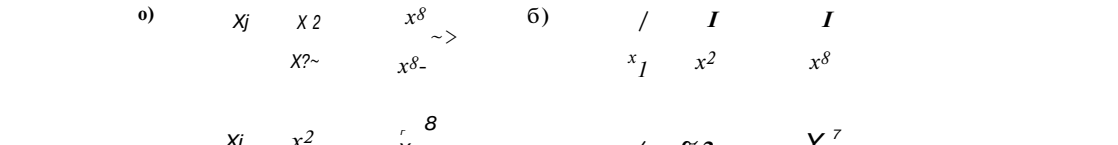
```
Uses Crt;
Const n=10;
Var a:Array[1..n, 1..n] of Integer;
    i,j:Byte;
Begin
  ClrScr;
 {формування матриці}
  for i:=l to n do
     for j:=1 to n do
        if  i>=j
            then a[i,j] :=n-i+j
            else a[i,j]:=0;
 {виведення матриці}
  for i:=l to n do
    Begin
        for j:=1 to n do
           Write(a[i,j] :2) ;
        Writeln;
    End;
  repeat until keypressed
End.
```

Приклад. Дано натуральне ***n***. Отримати квадратну матрицю **ncj** рядку ***n:***



```
u.-:es  Crt;
''onst n=10;
V.ir  a:Array[l. .n,1..n]  of  Real;
     v:Array[l. .n]  of  Real;
     i,j:Byte;  f:Real;
Hi'gin
  ClrScr;
 {формування допоміжного вектора v розмірності n, v[i]=l/i!}
  for  i:=l  to  n  do
       Begin
          f :=1;
          for  j:=1  to  i  do
              f:=f*j;
          v[i]:=l/f
       End;
 (формування  матриці}
  for j:=1  to  n  do
    for  i:=l  to  n  do
       Begin
           a[i,j]:=v[j];
           v[j]:=v[j]*v[j]
       End;
 (виведення  матриці]
  for  i:=l  to  n  do
    Begin
        for  j:=1  to  n  do
            Write(a[i,j]:2);
        Writeln;
    End;
  repeat  until  keypressed
Kid.
```

Приклад. Дано дійсні числа $x^{l}, x^{2},\ldots,x^{s}$. Отримати матрицю 8×8:



```
i i.-;es  Crt;
 'inst n=10;
v.i r  v, k:Array[1..n]  of  Real;
     a:Array[1..n,1..n]  of  Real;
```

```
        if ((i>=j) and (i+j<=n+l)) or ((i<=j) and (i+j>=n+l))
            then a[i,j]:=1
            else a[i,j]:=0;
  {виведення матриці}
   for i:=l to n do
     Begin
        for j:=1 to n do
          Write(a[i,j]:2);
        Writeln;
     End;
   repeat until keypressed
End.
```

Приклад. Дано натуральне ***n.*** Отримати квадратну матрицю п< рядку ***n:***

n	...	*0*	...	*0*
n -1	***n***	*0*		*0*
n-2	***n-1***	***n***		
1	*2*			

```
Uses Crt;
Const n=10;
Var a:Array[1..n,1..n] of Integer;
    i,j:Byte;
Begin
   ClrScr;
  {формування матриці}
   for i:=l to n do
      for j:=1 to n do
         if i>=j
            then a[i,j]:=n-i+j
            else a[i,j]:=0;
  {виведення матриці}
   for i:=l to n do
     Begin
        for j:=1 to n do
           Write(a[i,j]:2);
        Writeln;
     End;
   repeat until keypressed
End.
```

Приклад. Дано натуральне ***n.*** Отримати квадратну матрицю ію рядку ***n:***

$$\begin{vmatrix} 1 & 1 & & 1 \\ 1\backslash & 2\backslash & & n\backslash \\ 1 & 1 & & 1 \\ 1\backslash^2 & Z^2 & " & n\backslash^2 \\ "i & \cdots & & \cdots \\ .If & 2!" & & n\backslash". \end{vmatrix}$$

```
| | C r t ;
· · 4i:;t n=10;
V-ii  a:Array[ 1. .n, 1. .n]  of Real;
      v:Array[l..n]  of Real;
      i/j:Byte;  f:Real;

  i'IrScr;
 (формування допоміжного вектора v розмірності n, v[i]=l/i!}
  lor i:=l to n do
       Begin
          f:= 1;
          for j:=1 to i do
              f:=f*j;
          v[i]:=l/f
       End;
 I формування матриці}
  I or j:=1 to n do
    for i:=l to n do
       Begin
           a[i,j] :=v[j];
           v[j]:=v[j]*v[j]
       End;
 (виведення матриці)
  lor i:=l to n do
    Begin
        for j:=1 to n do
            Write(a[i,j]:2);
        Writeln;
    End;
  repeat until keypressed
Knd.
```

Приклад. Дано дійсні числа ***xj, x²,...,xˢ.*** Отримати матрицю ***8x8:***

x_j	$x->$	$*8$	б)	/	1	j
$x_{j\sim}$	$x?$	$x_{8\sim}$		x_1	x^2	x_8
x_j	x^2 ,	x_s^8		$X\ j\ \%2$		x^{s^7}

```
M:;es Crt;
•'onst n=10;
v.ir v, k:Array[1..n]  of Real;
     a:Array[1..n,1..n]  of Real;
```

```
    i,j:Byte;
Begin
  ClrScr;
  Writeln ('Введіть 8 дійсних чисел: ');
   for i:=l to 8 do
   Readln(v[i]);
 {формування допоміжного вектора k для збереження елементів v
   k:=v;
 {а)  формування матриці}
  for j:=1 to 8 do
      for i:=l to 8 do
          Begin
             a[i,j]:=k[j];
             ktj] :=k[j]*v[j'j
          End;
 {виведення матриці}
  for i:=l to n do
    Begin
        for j:=1 to n do
           Write(a[i,j]:2);
        Writeln;
    End;
 (б)  формування матриці}
  for j:=1 to 8 do
      Begin
        a[l,j]:=1;
      for i:=2 to 8 do
          Begin
             a[i,j]:=k[j);
             k[j]:=k[j]*v[j]
          End;
 {виведення матриці)
  for i:=l to n do
    Begin
        for j:=1 to n do
           Write(a[i,j]:2);
        Writeln;
    End;
  repeat until keypressed
End.
```

Приклад. Дано дійсну матрицю розміром ***mn.*** 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;
vectl=Array[1..m] of Real;      {для а) і в)}
vect2=Array[1..n] of Real;      {для б)}
Var a:mas; bl,b3:vectl; 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
       bl[i]:=a[i,1];
       for j:=2 to n do
           if a[i,j]>bl[i] then
             bl[i]:=a[i,j];
    End;
 (б) побудова масиву з сум ел-тів стовпців матриці}
  for j:=l to n do {цикл по стовпцях}
    Begin
       b2[j]:=0;
       for i:=l to m do
         b2[j]:=b2[j]+a[i,j];
    End;
 Iв) побудова масиву з середніх арифметичних рядків м-ці}
  for i:=1 to m do
    Begin
       Ь3[i]:=0;
       for j:=1 to n do
         Ь3[i]:=b3[i]+a[i,j];
       Ь3[i]:=Ь3[i]/п;
    End;
 {виведення результатів}
  Writeln('Макс. ел-ти рядків м-ці:');
  for i:=1 to m do Write(bl[i]:3);
  Writeln;
  Writeln('Суми ел-тів стовпців м-ці:');
  for i:=1 to n do Write(Ь2[i]:3);
  Writeln;
  Writeln('Середні арифм. рядків м-ці:');
  for i:=1 to m do Write(Ь3[i]:3) ;
  Writeln;
Knd.
```

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 розп всюдження практично у всіх мовах програмування.

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

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

У мові Паскаль для організації підпрограм використовуют процедури і функції.

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

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

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

Використання процедур і функцій забезпечує структуро^ ність програми, полегшує її написання і читання.

Процедури користувача

Процедура - це іменована група операторів, яка реалізує пе ну частину загальної задачі і викликається за іменем з будь-як позиції у головній програмі.

Формат опису процедури:

Procedure ім'япроцедури [(список формальних параметрів)]; - **з** головок процедури

```
розділи описів;
Begin
   оператори          тіло процедури
End;
```

Procedure (процедура) службове слово, ім'япроцедури - іде фікатор,
список формальних параметрів = формпарі: Типі, формпар Тип2, ...
розділ описів - опис локальних об'єктів процедури (за синтаксисо збігається 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 використовуються оператор виклику процедури.

Формат оператора виклику процедури:

їм и процедури[(список фактичних параметрів)];

і писок фактичних параметрів = факт_пар1, факт_пар2, ...

Приклад. Оператори виклику процедур:

Sort(al,bl); Gauss; Kvadr(14, 25, TRUE);

1 Іараметри використовуються для обміну даними між процеси рою і основною програмою.

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

1 Іараметри, які використовуються при виклику процедури, **Hit** шкаються **фактичними.**

Кількість, тип і порядок слідування фактичних параметрів нинішні бути такими, як у формальних параметрів. Параметри мо-*\ п. мати будь-який тип, але визначений заздалегідь. Процедура може взагалі не мати параметрів.

І Іри введенні у програму підпрограм виникає поділ змінних і Іч і пнів на глобальні і локальні.

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

Локальні змінні (константи, типи) - ті, які описані всередині ні чмрограми: або у списку параметрів (тільки змінні), або у розділ шч Const, Type, Var тіла підпрограми. За межами підпрограми лоті тьмі об'єкти недоступні.

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

І Іараметри, які передаються за значенням, називаються **параметрами-значеннями,** параметри, які передаються за посиланням, ші ишаються **параметрами-змінними.**

Параметри-значсння передаються з головної програми в проке і\ ру, але не повертаються з неї. Фактичний параметр, відповідний формальному параметру-значенню, може бути будь-яким вира и їм юго самого типу. Характерна риса параметрів-значень - зміна формальних параметрів не викликає зміни фактичних параметрів, міму параметри-значення використовуються в процедурах для пошипи їх аргументів (вхідних даних).

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

І Іараметри-змінні і передаються в процедуру, і повертаються і іні и основну програму. Фактичний параметр, відповідний паради-ір\ -змінній, може бути тільки іменем змінної. Характерна риса

параметрів-змінних - будь-яка зміна формального параметра озна чає зміну фактичного параметра. Параметри-змінні використову ються в процедурах для подання їх результатів, для того щоб р~ зультати могли бути передані і використані в основній програмі, списку формальних параметрів перед іменами параметрів-змі ставиться службове слово Var.

Правильно написана підпрограма "спілкується" з основно 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 а, ***Ь, с*** за фор му лою Герона, використавши процедуру.

```
Var a,b,c,S:Real;  {глобальні змінні}
Procedure Geron(x,y,z: Real; Var pi:Real);
 {x,y,z - параметри-значення, pi - 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:l);
End.
```

Приклад. Скласти процедуру обміну значеннями двох дійс 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 к" ристувач має можливість створювати необхідні йому функції і ви користовувати їх у своїх програмах.

Функція - це іменована група операторів, яка реалізує пев частину загальної задачі і має своїм результатом одне значе

простого типу. Ім'я функції використовується у виразах як операнд.

Формат опису функції:

Function Ім'яфункції [(список форм, пар-рів)]: тип_результату; - заголовок функції

розділи описів;
Begin
операторитіло функції
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 и 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 використовуються ім'я функції з указаним в дужках списком фактичних параметрів:

ім'яфункції [(список факт, пар-рів)];

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

Приклад. Обчислити значення виразу

$f(x) = x + x + 3 + e^{x+x+3} + \sin(x + x + 3)$ дляx=/,2,...,/(/

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

кіid.

Приклад. Знайти довжину вектора, заданого своїми координатами у «-вимірному просторі.

```
1'onst n=10;
гуре vector=Array[1..n] of Real;
```

```
Function VLength(v:vector):Real;
Var i:Integer; s:Real;
Begin
  s:=0;
  for i:=l to n do
    s:=s+sqr(v[i]);
    VLength:=sqrt(s);
End;
Var a:vector; i:Integer;
Begin
   for i:=l 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 вказує тип результату. В тілі функції повинен бути оператор, я присвоює імені функції певне значення.
2. Результати роботи. Результатом роботи процедури може бу довільна кількість значень довільних типів. Результатом ро ти функції є одне значення простого типу, вказаного в загол вку функції.
3. Використання. Процедура викликається у головній програмі допомогою окремого оператора. Функція викликається не докремлено, а як операнд виразу.

Правила локалізації

Програма на ТР має модульну структуру і може складатися* ряду вкладених один в одного блоків. Головна програма - це н' більший блок. Об'єкти, описані в цьому блоці, є глобальними і м жуть використовуватись у всіх вкладених блоках. Описані в об'єкти є локальними і недоступні у зовнішніх блоках.

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

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

Приклад. Дано *а, Ь, с* - довжини сторін деякого трикутни Знайти медіани трикутника, сторонами якого є медіани даного тр кутника. Довжина медіани, проведеної до сторони *а,* дорівнює

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

```
v.u  a,b,c:Real;                {сторони даного трикутника}
ml, m2, m3:Real;                {медіани даного трикутника}
mil, m22, m33:Real;             {медіани трикутника, утвореного
                                 з медіан даного}
 (•'unction med(x, y, z:Real) :Real;
IU'i) Ln
    med:=0.5*sqrt(2*x*x+2*y*y-z*z)
f'.nd;
и-'J in
    Readln(a, b, c) ;
    ml: =med(b, c, a) ;   {m^a}
    m2:=med(a, c,b) ;   {m^b}
    m3:=med(a,b,c);   {m^c }
    mil:=med(m2,m3,ml);
    m22:=med(ml,m3,m2);
    m33:=med(ml,m2,m3);
    Writeln(mll,m22,m33)
fcnd.
```

Приклад. Дано натуральні числа *n, m, k,* цілі числа *a[1],..., a[n], Ь/1/,...,Ь[m], c[1],...,c[1c].* Обчислити значення *F* за формулою */' inin(b[l],...,b[m]) + min{cflj,...,c[kj),* якщо *mm(a[1],...,a[n])>10,* і */' / +* (maxf*c[1],...,c[к]))*^{2} в протилежному випадку.

```
lімч-;   Crt;
I'niiHt   k=3;
        m=5;
        n=4;
Гу|н" mas=Array[1..20] of Integer;
y•ira,b,c:mas;
     F:Integer;
 Іпиедення масиву із заданої кількості чисел (к)}
|'mcedure Inp(var m:mas; к: Integer) ;
i/.ii i:Integer;
hi-,iin
  writein(1 Введіть 1,к, ' цілих чисел:');
  lor i:=1 to к do
    Read(m[i]);
  Writeln;
• и. I;
I   іислення максимуму в масиві із заданої кількості чисел}
tuuction Max(m:mas;k:Integer):Integer;
4
  n  i,tmp:Integer;
n.'1l n
  inip:=m[1];
  lor i:=2 to k do
    if  m[i]>=tmp
      then tmp:=m[i];
  M. їх: =tmp;
i п. I;
 ; • .i" іислення мінімуму в масиві із заданої кількості чисел}
```

```
Function Min(m:mas;к:Integer):Integer;
Var  i,tmp:Integer;
Begin
  tmp:=m[1];
  for i:=2 to к do
    if  m[i]<=tmp
      then tmp:=m[i];
  Min:=tmp;
End;
 {основна програма  }
Begin
  Clrscr;
  Inp(a,n);
  Inp(b,m);
  Inp(c,k);
  if  Min(a,n)>10
    thenF:=Min(b,m)+Min(c,k)
    else  F:=1+Sqr(Max(c,k));
  Writeln('F=',F:3);
  repeat  until  keypressed
End.
```

Приклад. Чотири вектори у 5-вимірному просторі задають своїми координатами. Обчислити скалярний добуток двох векторів, що мають найбільшу та найменшу норми (норму вектора визначити як найбільшу абсолютну величину координат вектора).

{ Координати всіх векторів зберігатимемо у вигляді матриці (тип vcct), координати одного вектора утворюють рядок матриці. Норми векторів зберігатимемо у масиві masnorm }

```
uses  Crt;
Const n=5;m=4;
Type vect=Array[1..m,1..n]  of Integer;
     masnorm=Array[1..m]  of Integer;
Var  v:vect;  norm:masnorm;
     i,imax, imin:Byte;
{введення вектора з указаним номером у масиві векторі
Procedure InpVect(var v:vect;k:Byte);
Var  i:Byte;
Begin
  Writeln ('Введіть ',n,' елементів ',k,'-го вектора: ')
  for i:=1 to n do
    Read(v[k,i]);
  Writeln
End;
 {виведення вектора з указаним номером  }
Procedure OutVect(v:vect;k:Byte);
Var  i:Byte;
Begin
  Writeln ('Елементи   ',k,'-го вектора:  ');
  for i:=1 to n do
    Write(v[k,i]:3);
```

```
  Writeln
Kud;
 (знаходження норми вектора з указаним номером }
Kunction Norma(Var v:vect;k:Byte):Integer;
Var i,max:Integer;
lu-gin
  max:=Abs(v[k,1]);
  for i:=2 to n do
    if Abs(v[k,i])>max
      then max:=Abs(v[k,i]) ;
  Norma:=max;
Kud;
 (знаходження номера максимального елемента в масиві
норм}
Procedure Max(v:masnorm;Var imax:Byte);
V.i r max,i:Integer;
hogin
  max:=v[1]; imax:=1;
  for i:=2 to m do
    if v[i]>max
      then
        Begin
          max:=v[i];
          imax:=i
        End;
Kud;
 I • інаходження номера мінімального елемента в масиві норм }
1'iocedure Min (v:masnorm;Var iminrByte);
v.i r min, i : Integer;
H'-qin
  itiin:=v[1] ; imin:=l;
  lor i:=2 to m do
    if v[i]<min
      then
        Begin
          min:=v[i];
          imin:=i
        End;
Kud;
 I • інаходження скалярного добутку векторів з указаними номерами]
function
  м LD (u: v e c t ; kl: B y t e ; v: v e c t ; k2 : B y t e ) : I n t e g e r ;
v.i t i, S : I n t e g e r ;
ll.-ij in
  :. :=0;
  lor i:=l to n do
    S:=S+u[kl,i]*v[k2,i];
  :;<:alD:=S;
i и. I;
 I. і с н о в н а  п р о г р а м а]
н ·. I Ln
  '' LrScr;
```

```
Function Min(m:mas;к:Integer):Integer;
Var i, trap:Integer;                                    ,
Begin
  tmp:=m[1] ;
  for i:=2 to к do
    if  m[i]<=tmp
      then tmp:=m[i];
  Min:=tmp;
End;
 {основна програма  }
Begin
  ClrScr;
  Inp(a,n);
  Inp(b,m) ;
  Inp(c,k) ;
  if  Min(a,n)>10
    then F:=Min(b,m)+Min(c,к)                           ;
    else F:=l+Sqr(Max(c,k));
  Writeln('F=*,F:3);
  repeat until keypressed
End.
```

Приклад. Чотири вектори у 5-вимірному просторі задаютьск своїми координатами. Обчислити скалярний добуток двох векторів) що мають найбільшу та найменшу норми (норму вектора визначи» ти як найбільшу абсолютну величину координат вектора).

{ Координати всіх векторів зберігатимемо у вигляді матриці (тип vect), координати одного вектора утворюють рядок матриці. Норми векторів зберігатимемо у масиві masnorm } '

```
uses Crt;
Const n=5;m=4;
Type vect=Array[1..m,1..n] of Integer;
     masnorm=Array[1..m] of Integer;
Var vrvect; normimsnorm;                                <
    i, imax,imin:Byte;
{введення вектора з указаним номером у масиві векторів}
Procedure InpVect(var v:vect;k:Byte);
Var i:Byte;
Begin
  Writeln ('Введіть ',n,' елементів ',k,'-го вектора: ');
  for i:=l to n do
    Read(v[k,i]);
  Writeln
End;
 {виведення вектора з указаним номером }
Procedure OutVect(v:vect;k:Byte);
Var i:Byte;
Begin
  Writeln ('Елементи  ',k,'-го вектора: ');
  for i:=l to n do
    Write(v[k,i]:3);
```

```
  Writeln
Knd;
 I знаходження норми вектора з указаним номером }
Function Norma(Var v:vect;k:Byte):Integer;
v.ir  i,max:Integer;
Megin
  max:=Abs(v[k,1]);
  for i:=2 to n do
    if Abs(v[k,i])>max
      then max:=Abs (v[k, i] ) ;
  Norma:=max;
Kud;
 I знаходження номера максимального елемента в масиві
норм}
I'tocedure Max (v:masnorm;Var imax:Byte);
v,ir max,i:Integer;

  max:=v[l]; imax:=l;
  for i:=2 to m do
    if  v[i]>max
      then
        Begin
          max:=v[i];
          imax:=i
        End;
Knd;
 (знаходження номера мінімального елемента в масиві норм }
Procedure Min(v:masnorm;Var imin:Byte);
Vur  min,i:Integer;
I.' 'gin
  min:=v[l]; imin:=l;
  for i:=2 to m do
    if v[i]<min
      then
        Begin
          min:=v[i];
          imin:=i
        End;
Knd;
 {знаходження скалярного добутку векторів з указаними номерами}
Function
• '.calD (u:vect;kl:Byte;v:vect;k2:Byte) :Integer;
Vur  i,S:Integer;
Begin
  S:=0;
  for i:=l to n do
    S:=S+u[kl,i]*v[k2,i];
  ScalD:=S;
Knd;
 (основна програма}
Begin
  ClrScr;
```

```
  {введення масиву координат векторів  }
   for i:=1 to m do InpVect(v,i);
    {формування масиву норм  }
   for i:=1 to m do norm[i]:=Norma(v,i);
  {знаходження номерів векторів з максимальною і мінімальною н.
   мами}
   Max(norm,imax);
   Min(norm, imin);
   Writeln('Вектор з максимальною нормою:');
   OutVect(v,imax);
   Writeln('Вектор з мінімальною нормою:');
   OutVect(v,imin);
  Writeln('Скалярний добуток цих векторів = ',ScalD(v,imax,v,imin))
   repeat until keypressed;
End.
```

Приклад. Три трикутники задані координатами своїх вершин площині. Визначити трикутник, що має найбільший периметр, обчислити його площу.

{Координати вершин кожного трикутника зберігатимемо у вигл матриці (тип koord), координати одної вершини утворюють рядо матриці.

Периметри трикутників зберігатимемо у масиві masper }

```
Uses  Crt;
Const m=3;n=2;
Type  koord=Array[1..m,1..n]  of Real;
      masper=Array[1..m]  of Real;
Var  a,b,c:koord;  p:masper;
     i,imax:Byte;max:Real;
 {введення координат вершин трикутника  }
Procedure  InpKoord(var v:koord);
Var  i:Byte;
Begin
   for i:=1 to m do
     Begin
       Write('x[',i,']= ');
       Readln(v[i,1]);
       Write('y[',i,']= ');
       Readln(v[i,2]);
     End;
     Writeln
End;
 {обчислення довжини сторони трикутника  }
Function Dov(xl,yl,x2,y2:Real):Real;
Begin
  Dov:=Sqrt(Sqr(x2-xl)+Sqr(y2-yl));
End;
 {обчислення периметра трикутника  }
Function  Perimetr(v:koord):Real;
Var  a,b,c:Real;
Begin
```

```
   .i:=Dov(v[l,l] ,v[l,2] ,v[2,l] ,v[2,2] ) .
   b:=Dov(v[l,1],v[l,2] , v[3, 1],v[3,2]) ,
   c:=Dov(v[2,1],v[2,2],v[3,l],v[3,2]
   I'erimetr: =a+b+c;
find;
 I знаходження максимального елемента в масиві периметрів і його
   номера}
1'iocedure MaxP(p.-masper;Var max:Real;var imax:Byte);
V.ir i:Integer;
hr-(jin
   inax:=p[l];  imax:=l;
   for i:=2 to m do
     if  p[i]>max
        then
          Begin
            max:=p[i] ;
            imax:=i
          End;
Knd;
 (обчислення площі трикутника }
function Pi(v:koord):Real;
v.ir  a, b, c,p : Real;
lw-qin
   n:=Dov(v[l,l],v[l,2],v[2,l],v[2,2])
   b:=Dov(v[l,l],v[l,2],v[3,l],v[3,2])
   c:=Dov(v[2,l],v[2,2],v[3,l],v[3,2])
   p:=(a+b+c)/2;
   PI:=Sqrt(p*(p-a)*(p-b)*(p-c)) ;
r,nd;
I основна програма }
Iw-gin
   ClrScr;
   Writeln('Введіть координати вершин першого трикутника[1]');
   CnpKoord(a);
   Writeln('Введіть координати вершин другого трикутника');
   inpKoord(b);
   Writeln('Введіть координати вершин третього трикутника');
   rnpKoord(c);
  I формування масиву периметрів трикутників}
   p[1]:=Perimetr(a);
   p[2]:=Perimetr(b);
   p[3]:=Perimetr(c);
   (знаходження максимального периметра і номера трикутника}
   MaxP (p, max, imax) ;
   (обчислення площі трикутника з максимальним периметром}
   case imax of
     1: Begin Writeln('Per 1 tr-ka=',p[l] :4:2,' Pl=',Pl(a) :4:2) End;
     2: Begin Writeln('Per 2 tr-ka=',p[2] :4:2, ' Pl=* ,P1 (b) :4:2)End;
     3: Begin Writeln ('Per 3 tr-ka=",p[3] :4:2, ' Pl=',Pl(c) :4:2)End;
   end;
   repeat until keypressed;
r.nd.
```

Рекурсія

Використання рекурсії є традиційною перевагою мови Паскаль. Під **рекурсією** розуміють виклик функції (процедури) з тіл 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-4)\, якщо ***n>0.***

Формула обчислення цілого степеня числа:

x"=1, якщо ***n=0;***

$x^{я}=x*x"\sim^{/}$, 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):Longlnt;
Begin
 if n=0 then Fact:=l
        else Fact:=n*Fact(n-1) ;
End;
Function IntPowerfx: Real; n: Word):Real;
Begin
  if n=0 then IntPower:=l
         else IntPower:=x*IntPower(x,n-1);
End;
```

Якщо у функцію передається ***n>0,*** відбувається наступне: запам'ятовуються відомі значення членів виразу гілки ***else*** (для факторіала 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 - при присвоювання гілки ***then),*** після чого алгоритм починає "розкручуватись" в інший бік, беручи з пам'яті "відкладені" значення. Спільки при цьому на кожному черговому кроці 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 виконана, 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 тому, що кожний "відкладений" виклик функції чи процедури - це черговий набір значень всіх локальних змінних цієї функції, розміщених у с теку. (У TP розмір стеку не може перевищувати 64K).

Незважаючи на наочність рекурсивних описів, в багатьох випадках такі 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;
v.it i: Word; m: Real;
M<-gin
  m:=l;
  for i:=l to n do m:=m*x;
  intPower:=m
F.nd;
```

Побічний ефект

Побічним ефектом називається виконання функцією дій, відмінних від обчислення її основного значення. Зокрема, це зміна іпачень глобальних змінних, виведення певних даних, не пов'язаних з призначенням функції тощо.

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

Приклад.

```
v.ir a,b: Real;
Function f(x:Real):Real;
Mi 'gin
   f:=2*x*x+0.5;
   a:=0; {побічний ефект - зміна значення глобальної змінної}
F.nd;
Hngin
   a:=3.14; b:=f(3);
   Writeln(a,b);
F.nd.
```

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

Основні оператори мови Паскаль. Оператори завершення роботи програми

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

Виклик Exit завершує роботу свого програмного блоку. Якщо виконається Exit у процедурі, то її виконання завершиться і відбудеться перехід до виконання оператора, наступного за оператором процедури. При цьому процедура поверне значення, які було обчислено на момент виконання Exit (якщо вона повинна була їх повернути). Виконання програми не перерветься. Якщо ж Exit виконається 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 п, який потім може бути проаналізований. Значення п=0 відповідає нормальному коду завершення. Виклик процедури Halt без параметра еквівалентний виклику Halt(0).

Запитання 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 використаннярекурсії.*
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 останніх 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
=; о ; <; >; <=; >=	операції відношення	'A' < 'B' -> TRUE

Таблиця 14. Функції для величин символьного типу

Назва	Призначення	Приклади
ORD(c)	порядковий номер символу **с** в заданому символьному наборі	ORD(A') = 65; ORD(#4) = 4
CHR(i)	символ, що відповідає числу і згідно з кодами ASCII	CHR(65)='A'
PRED(c)	символ, що передує символу **с**	PRED('B') = 'A'
SUCC(c)	символ, що є наступним за символом **с**	SUCC(*A')='B'
IJPCASE(c)	символ верхнього регістру, відповідний **V..'z'**	UPCASECa')='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 типу і = String I максимальна довжина рядка I;
1м'я_типу2 = String;
Var
змінна1: 1м'я_типу1;
змінна2: String**[МаКС.** довжина рядка];
змінна3: String;

String (рядок) - службове слово, максимальна довжина рядка - ціле число п, 0 < п < 255.

Опис String еквівалентний Slring[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
                            0 до 255 символів]
D: String[30]; {довжина B - від 0 до 20, D -від 0 до ЗО символів}
```

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

Рядки можна розглядати як масиви символів. Будь-який сим вол 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 Writahr(S[i]); {друкування рядка S посимвольно}
     Readln  {пауза до натиснення Enter}
End.
```

Символ рядка з номером 0 має код, який дорівнює кількос символів у рядку, тобто довжина рядка S завжди дорів 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 змг них надавати їм початкові значення, наприклад, порожній рядок " Для введення значень рядкових змінних використовується опер тор Readln, а не Read. Одним оператором можна ввести тільки од рядок.

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

Операції над рядками

Рядки можна порівнювати, присвоювати і з'єднувати (отр мувати суму, здійснювати конкатенацію, злиття).

Приклад.

```
Var
    SI, S2, S3 :String-;
Begin
    51 :=  'Вам ';
    52 :=  'привіт';
    53 :=  Si + S2;   {S3 = 'Вам привіт'}
```

```
    S3 :=  S3 + '!'   {S3 = 'Вам привіт!'}
Kiid.
```

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 більший за інший (його код більший), то рядок, що його містить, також вважа-(ться більшим незалежно від решти символів у рядках. Будь-який рядок більший "порожнього рядка".

Приклад.

```
abed' =  'abed'     (TRUE),
.ibcd' <> 'abede'   (TRUE),
• ibcd' > 'abcD'     (оскільки, 'd' > 'D'),
• ibcd' > 'abc'      (оскільки, 'd' >  ''),
• iBcd' < 'ab'       (оскільки, 'B' < 'b'),
  ' '  >   "         (оскільки, #32 > ") .
```

Для роботи з рядками у ТР реалізовано низку стандартних процедур і функцій (таблиця 16).

Таблиця 16. Процедури і функції для опрацювання рядків

Процедури і функції	Призначення, пояснення, приклади використання
1 ,cngth(S : String) : Byte	Видає довжину рядка S. Аналогічно діє Ord(S[0])
Concat(Sl, S2, Sn): Siring	Повергає конкатенацію або злиття рядків S1, S2,... ,Sn. Аналогічно діє операція "+": S3 :=Concat(Sl,S2); {те саме, що S3:=S1 +S2}
('i)ру(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 під Start до кінця рядка S, то повертається залишок рядка S від Start до кінця. а) SCopy:=Copy(AJBC***123',4,3); {SCopy = '***'} б) SCopy:=Copy('ЛВС',4,3); {SCopy ="} в) SCopy:=Copy('ABC***123\4,1 1); {SCopy = '***123'}
1 Vlete(Var S : String; Start, 1 i-ii : Tnteger)	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 змінюєтьс Якщо Len більше, ніж залишок рядка, вил' чається підрядок від Start і до кінця S. 8:='РЯДОК'; Delete(S,2,2); {S = 'POK'}
Lnsert(SubS : String; Var S : String; Start : Integer)	Вставляє в S підрядок SubS, починаючи позиції Start. Якщо змінений рядок S вияві ється надто довгим, то він автоматично с рочується з правого боку до оголошеної д вжини S. S:=' Ранок-вечір'; Ішел('день-\ S,7); {тепер S = 'Ранок-день-вечір'}
Pos(SubS , S : String) Byte	Шукає входження дідрядка SubS в рядок S> повертає номер першого символу SubS в або 0, якщо S не містить SubS. Недоліко' даної функції є те, що вона повертає н~ ближчу стартову позицію SubS в S від поч тку рядка, ігноруючи наступні, якщо вони Так, виклик P:=Pos('noo', 'Boonoonoonoos'); завершує ро ту, повернувши значення 4, хоча є ще 7 і 10.
Str(X [: Width [: dec]]; VarS: String)	Перетворює числове значення X в рядкове X може бути змінною або значенням ціл~ або дійсного типу. Можна задавати поля рмату, вказавши ширину моля для числа кількість десяткових знаків. Якщо число менше знаків, ніж дано в полі, воно вирів єгьея за правим краєм, порожнє місце запо нюється пропусками. Якщо задати п~ Width від'ємним, вирівнювання здійсню тиметься за лівим краєм, а залишки сти ються. Якщо формат з крапкою обмежує лькість знаків у дробовій частині, то во буде округлена при перетворенні. Значе~ самого числа при цьому не зміниться. Str(6.66:8:2, S); {S=' 6.66'} Str(б.бб:-8:2, S); {S='6.66'} Str(6.66:8:0, S); {S=' 7'}
Val(S : Suing; Var X; Var ErrCode : Integer)	Перетворює рядкове значення S (ря цифр) в значення числової змінної X. ***Я*** перетворення можливе, то змінна ErrCode інакше вона міститиме номер символу в S, якому відбулася помилка. Тип X повин 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;
hegin
      ClrScr;
      Writeln ('Введіть слово');  Readln(S);
       P:=TRUE;
       for i:=l to Length(S) div 2 do
            if S[i] <> S[Length(S)-i+l]  then P:=FALSE;
            if P then Writeln('паліндром')
               else Writeln('не паліндром')
Knd.
```

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

```
I'ype MasStr=Array[l..10] of String[8];
v.ir  Sl:MasStr; i:Integer;
H<-gin
  for i:=l to 10 do  Readln(Sl[i]);
  for i:=100 downto 1 do Writeln(SI[i])
Knd.
```

Приклад. Дано речення, яке містить *1-20* слів, у кожному слові / *5* літер, між словами - пропуски. Надрукувати слова речення у торотному порядку.

```
H.ues   Crt;
Type   Slovo = String[5];   {окреме слово}
       Rech = String;       {дане речення}
       MasSliv = Array[1. .20] of Slovo; { стверкваний масив слів}
v.ir  S:Slovo;  R:Rech; M: MasSliv;
 I Функція розбиття речення на слова}
Function Getword(S:Rech; Var N:Byte):Slovo;
v.ir Tmp: String;
lii-gin
  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 - елемэнта ма-
                         сиву слів}
       N:=N+1
     End;
  iietWord:=Tmp
Пi.);
"и  i:Byte;  {лічильник символів речення}
    N:Byte;{лічильник  слів}
```

```
    к: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.
```

Приклад. Дано рядок. Вивести двічі підряд кожну цифру, Щ входить v рядок. Наприклад, ввести *''ablc2d34e5",* вивес ***"abllc22d3344e5r***

```
Uses Crt;
Var s:String; i:Byte;
Begin
  ClrScr;
  Writeln('Введіть рядок:');
   Readln(s);
   i :=1;
   while i<=2*length(s) do
     Begin
       if pos(s[i],'0123456789')>0 {символ є цифрою}
          then
            Begin
              insert(s[i],s,i+1);
              i:=i+2
            End
          else i:=i+l
       End;
   Writeln('Перетворений рядок: ');
   Writeln(s);
  repeat until keypressed
End.
```

Приклад. Дано масив слів (до ***10*** слів). Вивести лише пер входження кожного слова.

```
Uses Crt;
Const n=5;
Type st=Array[1..n] of String;
Var s:st; i,j:byte; p:Boolean;
Begin
  ClrScr;
  Writeln('Введіть слова масиву:');
  for i:=l to n do Readln(s[i]);
    Writeln('Слова без повторень:');
    Writeln(s[1]); {перше входження першого слова}
    for i:=2 to n do {перегляд решти слів}
```

```
        Begin
          p:=TRUE;
          for j:=l to i-1 do {перегляд слів у розглянутій частині масиву}
            if s[i]=s[j] then p:=false; {таке слово вже було надруковано}
          if p then Writeln(s[i}); {слово знайдено вперше}
        End;
    repeat until keypressed
Knd.
```

Приклад. Дано масив слів. Вивести слова, літери яких упорядковані за алфавітом. Використати логічну функцію, яка визначає, чи є слово упорядкованим.

```
uses Crt;
I'onst n=15;
Type st=Array[l..n] of String;
v.ir s:st; i:byte;
Function ABC(str:String):Boolean;
v.ir i:Byte; p:Boolean;
Hi-gin
     p:=TRUE;
     for i:=l to Length(str)-1 do
         if str[i]>str[i+1]
              then p:=FALSE;
     ABC:=p
Knd;
M.-gin
   ClrScr;
   Writeln('Введіть слова масиву:');
   for i:=l to n do Readln(s[i]);
    Writeln('Упорядковані слова: ') ;
   (or i:=l to n do
       if ABC(s[ij) then Writeln(s[i]);
   repeat until keypressed
їм id.
```

Приклад. Дано масив слів. Вивести слова масиву, що містять мирну кількість літер «а». Використати функцію, значенням якої є кнькість літер «о» у слові.

```
·1i.-.'-;  Crt;
• "ii:;t n=15;
r-,'l"- st=Array[1..n] of String;
"•и s:st; i:Byte;
ким. tion CountO(str:String):Byte;
"•и i,K:Byte;
ld-.llП
     K:=0;
     for i:=l to Length(str) do
         if str[i]='o'
              then      K:=K+1;
     CountO:=K
```

```
End;
Begin
  ClrScr;
  Writeln('Введіть слова масиву: ') ;
  for i:=l to n do
     Readln(s[i]);
   Writeln('Слова з парною кількістю літер "o":');
  for i:=l to n do
      if CountO(s[i]) mod 2 = 0 then Writeln(s[i]);
  repeat until keypressed
End.
```

Приклад. Дано масив слів. Вивести слова масиву, що містять максимальну кількість літер «о». Використати функцію, значенням якої є кількість літер «о» у слові.

```
Uses Crt;
Const n=15;
Type st=Array[1..n] of String;
Var s:st; i, max:Byte;
Function CountO(str:String):Byte;
var i,K:Byte;          * .
Begin
     K:=0;
     for i:=l to Length(str) do
        if str[i]= 'o'
           then K:=K+1;
     CountO:=K
End;
Begin
  ClrScr;
  Writeln('Введіть слова масиву:');
  for i:=l to n do
    Readln(s[i]);
 {знаходження максимальної кількості літер "o" у словах }
  max:=CountO(s[1]) ;
  for i:=2 to n do
    if  CountO(s[i])>=max
        thenmax:=CountO(s[i]);
 {виведення слів з максимальною кількістю літер "o"}
  Wtiteln('Максимальна кількість літер "o" у словах=', max) ;
  Wtiteln('Слова з максимальною кількістю літер "o":');
  for i:=1 to n do
    if  CountO(s[i]) =max
      then Writeln(s[i]) ;
  repeat until keypressed
End.
```

Приклад. Дано масив слів. Вивести слова масиву, перша літе яких входить до слова не менше трьох разів. Використати функці значенням якої є кількість входжень першої літери у слово.

```
Uses Crt;
const n=15;
Type st=Array[1..n] of String;
Var s:st; i:Byte;
Function Countl(str:String):Byte;
var i,K:Byte;
Begin
     K:=l;
     for i:=2 to Length(str) do
         if str[i]=str[l]
           then K:=K+1;
     Countl:=K
Knd;
Begin
  ClrScr;
  Writeln('Введіть слова масиву:');
  for i:=l to n do
     Readln(s[i]);
   Writeln('Слова, перша літера яких входить до них не менше 3
            разів:');
  for i:=1 to n do
      if Countl(s[i])>=3
       then Writeln(s[i]);
  repeat until keypressed
Knd.
```

Приклад. Дано масив слів. Вивести слова, які зустрічаються в масиві не менше двох разів.

```
Uses Crt;
Const n=15;
i'ype st=Array[1..n] of String;
V.ir s:st; i:Byte; P:Byte; b:Boolean;
Ri -gin
  ClrScr;
  Writeln('Введіть слова масиву:');
  for i:=l to n do
    Readln(s[i]);
  for i:=l to n-1 do  {розгляд кожного слова масиву)
    Begin
      P:=1; {кількість входжень слова з номером i}
       for j:=i+l to n do
          if s[i]=s[j]
             then P:=P+1;
      b:=TRUE; {припущення, що таке слово ще не виводилось}
        for j:=1 to i-1 do
          if s[i]=s[j]
             then b:=FALSE;
          if b and (P>=2) {слово зустрічається не менше 2 ра-
                           зів, і воно}
             then Writeln(s[i]) {ще не виводилось}
```

```
    End;
     repeat until keypressed
End.
```

Запитання для самоконтролю

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 ; прикладах.*
5. *Як забезпечується доступ до символу за його кодом без використай 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 клавіатури?* * .
13. *Як перетворити маленькі латинські літери у великі?*
14. *Яка процедура призначена для вставки підрядка у рядок? Наведіт 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? Наве діть приклади її використання.*
17. *Як дізнатися, чи входить один рядок до складу іншого? Наведіть пр клади.*

Множини у мові Паскаль

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

Всі елементи множини повинні належати одному типу. Ц тип називається **базовим типом** множини. Базовим може бути бу який простий упорядкований тип. Він задається діапазоном або реліком значень. Область значень типу множина - набір всіх **і** множин, складених з елементів базового типу. Якщо базовий тип м 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'язмінної: Set of базовийтип;

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',
                                 'I',-,'9'}
 SetSpring = Set of (March, April, May); {множина з весняних
                                          місяців}
  SetGolosn = Set of  (*A', 'O', 'У, 'I', *Є', 'И', 'ї');
                          {множина з великих голосних літер}
```

В мові ТР дозволяється визначати множини, які мають не більше, ніж 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), Integer(-32768..32767), Word (0..65535), Lxmgmt(-2147483648..2147483647).

Для запису множин у ТР використовують квадратні дужки. І Іорожню множину записують як [].

Приклад.

```
Byte      [1,2,3,4,10,20,3O,40];
char      ['a', 'Ь', 'c'];
char       ['d'];
Spring := [April];
i)iap  = [1..4]; {те саме, що [1,2,3,4]}
Comp   = [1..4, 5,7,10..20];
mpty   = []; {порожня множина - підмножина будь-якої множини}
```

Порядок слідування елементів усередині дужок не має значення, не має значення і кількість повторення елементів. Повторні входження елементів ігноруються. Так, записи ***['a',Ь',Ь***','«'] і ['«','&'] рівносильні.

Приклад. Усі множини, які можна побудувати на базовому типі І 3.

v.ir **S : Set of 1. . 3;**

```
Begin
    {допустимі присвоювання}
    S:=[j;  S:=[1J;  S:=[2];  S:=[3J;  S;=[1,2J;  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	SI =S2	Результатом **є** логічне значення, дорівнює 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','Ь','c']
	Перевірка на нерівність	SIoS2	Результатом **є** логічне значення, дорівнює TRUE, якщо S1 і S2 відрізняються хоча б одним елементом, і FALSE у протилежному випадку. [1,2] ^**LI**];no[3J;[*a'..'c'Jo1'aVc']
	Перевірка на підмножину	S1<=S2	Результатом **є** логічне значення, дорівнює TRUE, якщо всі елементи S1 містяться і **в** S2 незалежно від їх порядку слідування, і FALSE у протилежному випадку. **Га**'.Ъ'] <= **га**'..'2'**1; П <= [41;** [1,2] <= [1,2,3]
	Перевірка на надмножину	S1>=S2	Результатом **є** логічне значення, дорівнює TRIJPI, якщо всі елементи S2 містяться в S1, і FALSE у протилежному випадку. ['x'..'z'] **>=Гу'];** [1.. 10] >=[!..5]; [5,7] >=[]
m	Перевірка входження елемента у множину	EE m [...] НИ in SI	Результатом **є** логічне значення, дорівнює TRUE, якщо значення Е належить базовому тину множини і входить у множину [...] (S1). іікщо множина не містить у собі значення К, то результатом *с* FALSE. 5 in [0..5]; 's' in ['a'..'z'l; not (7 in [9..20])
	Об'єднання множин	SI + S2	Результатом об'єднання є множина, отримана злиттям елементів цих множин і виключенням елементів, що повторюються. U,2]+[2,3,4j = [1,2,3,4]; ЦІ..4]} ['s'] + [] = ['s']

	різниця множин	SI -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]; [T,'2'] - ['879'] = [T','2']
*	Перетин множин	SI * 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]; fx'] *[] = []

Переваги типу множина:

компактність подання - один елемент множини займає один байт 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 в Турбо Паскаль не дозволяється виводити множини на екран і отримувати окремі елементи з множин. Введення множин можливе лише по елементах.

Робота з довільною множиною переважно відбувається за такою схемою:

перевірити всі елементи базового типу на входження в дану множину;

у разі входження вивести елемент на екран або певним чином опрацювати.

Приклад. Створити дві множини з маленьких латинських літер, 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;
Hegin
  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 o '.' 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'); WedSet(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 ['T','f','T','r','A','r','fi','i','Y','j','y','l
End.
```

Приклад. Ввести послідовність латинських літер (ознака закінчення введення - крапка). Вивести перші входження літер у послідовність, зберігаючи їх початковий взаємний порядок. (Наприкл 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.
```

Приклад. Ввести послідовність латинських літер, в кінці - крапка. Визначити кількість входжень у послідовність літер "й", "є", "с", "ґ".

```
Uses Crt;
Var let: Char; s:Integer;
Begin
  ClrScr;
  s:=0;
  Writeln('Вводьте латинські літери, в кінці - крапка:');
  Readln(let);
  while let <> '.' do
  Begin
    if let in ['a','e','C,'h']
      then s:=s+l;
    Readln(let)
  End;
 Writeln('Загальна кількість входжень a, e, c, h = ',s:3);
  repeat until keypressed
Knd.
```

Приклад. Підрахувати кількість різних цифр у десятковому запису натурального числа.

```
uses Crt;
Var sd: Set of 0..9;
    d: 0..9;
    n,k:Longint;
Begin
  ClrScr;
  Writeln('Введіть натуральне число:');
  Readln(n);
 (Виділення цифр числа справа наліво і запис їх у множину sd}
   sd:=[];
   repeat
     d:= n mod 10;
     sd:=sd+[d];
     n:=n div 10
   until n=0;
 I Підрахунок елементів у множині  sd]
   k:=0;
   for d:=0 to 9 do
     if d in sd
       then k:=k+l;
 Writeln('У числі ',k:2,' різних цифр');
```

```
 repeat until keypressed
End.
```

Приклад. Вивести в порядку зростання всі цілі числа від / до які можна подати у вигляді m^2+n , $m, n>=0$.

```
Uses Crt;
Var sl:Set of 1..25;
    m,n:Byte; p:1..25;
Begin
  ClrScr;
  sl:=[]; {утворення множини з шуканих чисел}
  for m:=0 to 5 do
    for n:=0 to 5 do
      sl:=sl+[m*m+n*n] ;
 {виведення отриманих чисел у порядку зростання}
  for p:=1 to 25 do
    if p in si
      then Writeln(p);
  repeat until keypressed
End.
```

Приклад. Ввести послідовність цифр, у кінці - крапка. Ко третю цифру записати до множини ***S1,*** решту - до множини Знайти перетин множин ***S1*** і ***S2.***

```
Uses Crt;
Var sl,s2: Set of '0'..'9';
    d: '0'..'9';
    i:byte;
Begin
  ClrScr;
  sl:=[]; s2:=[];
  i:=0;              {лічильник введених символів}
  Writeln('Вводьте цифри, в кінці - крапка:');
  Readln(d);
  while do '.' do
    Begin
      if d in ['0'..'9']
         then i:=i+l;
      if i mod 3 = 0
        then sl:=sl+[d]
        else s2:=s2+[d];
        Readln(d)
    End;
    Writeln('SI:'1);
    for d:='0' to '9' do
       if d in si
          then Write(d,' '1);
    Writeln;
    Writeln('S2:');
    for d:='0' to '9' do
```

```
    if d in s2
       then Write(d,' ');
  Writeln;
  Writeln('S=S1*S2:');
  for d:='0' to '9' do
    if d in s1*s2
       then Write(d,' ');
  Writeln;
  repeat until keypressed
F.nd.
```

Питання для самоконтролю.

1. Для чого призначений тип даних Set? Які його переваги і недоліки?

2. Як описується множинний тип засобами ТР?

.1 Які типи даних можуть використовуватись як базовий тип множини?

•І. 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.*

(і. Вкажіть операції над множинами у ТР, результатом яких є множина. Наведіть приклади їх використання.

/. Як здійснюється доступ до окремого елемента множини?

Х. 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 стосуються 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 передбачено тип даних запис, або комбінований тип.

Запис - це структурований тип даних, який 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 указаниям імен полів і типу кожного поля.

Формат опису типу запис:

```
Type
  Ім'ятипу = Record
               ім'яполя1: типполя1;

               ім'я_ raxiMN : тип_гк>ш^
             End;
```

Var

Ім'язмінної : Ім'ятипу;

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 ім'ям.

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 можна використовувати 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 := '25035КН*; M.Marka := 'ВАЗ-2108'; M.Probig := 16000; M.PIB := 'Ткачук П.П.'; Point.x :=12; Pointy := 24;

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

Приклад.

```
Read(M.Nomer, M.Marka, M.Probig, M.PIB);
Write(M.Nomer:?, 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.

У ряді задач зручно користуватися масивами із записів. їх можна описувати, зокрема, так:

Приклад.

```
Type
Person = Record
            PIB:  String[3O];
            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 і, здійснюватиметься за такими складеними іменами: L[i].PIB, L[i].Vik, L[i].Prof.

Складеш 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, використавши оператор 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,...,Rndo...
```

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:='Риженко';
     town:='Чернігів';
       With address do
          Begin
             street:='вул. Широка';
             house:=1;
             flat:=6
          End
   End;
```

Приклад. Описати комбінований тип, що містить відомості про людей: прізвище, місто проживання, адресу (назва вулиці, номер 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;
    Ы, Ь2, Ь3, 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
      Ы := Hum[i].Town <> Hum[j].Town;
      b2 := Hum[i].Address.Street = Hum[j].Address.Street;
      Ь3 := Hum[i].Address.House =
Hum[j].Address-House;
      b4 := Hum[i].Address.Flat  = Hum[j] .Address.Flat
      if Ы and b2 and Ь3 and b4 then
          Begin
              Writeln('За іронією долі');
              Writeln(Hum[i].Surname,' і ', Hum[j].Surname);
              Writeln ('живуть в різних містах за однаковими ад-
                       ресами')
          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 найвищого 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[l..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 humfi]1 do
       if sex='ж' then begin s:=s+height; nn:=nn+l 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:=l to n do
     if (humfi].зex='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:=l to n-1  do
      for j:=i+l  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, tl, t2: Longlnt;
Begin
 {переведення даних величин часу в секунди}
   tl:=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-tl);
 {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 ВНЗ: 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 сту-
                 дентів-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:');
tor 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[l] := C[l] +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[l]; {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. Варіантне частина формується за допомогою оператора Case. Він задає особливе поле запису - поле ознаки, яке визначає, який з варіантів і 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
           константавиборуї : (поле, ...: тип);

           константавиборуп : (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 за аналогією 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 : ( Cinal : 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 Cinal 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:

псі імена полів повинні відрізнятися одне від одного принаймні одним символом, навіть якщо вони зустрічаються у різних варіантах;

шпис 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 записуіться так: мітка: ();.

Запитання для самоконтролю

I. Для чого призначено тип даних Record? Які його особливості?

1 Як описується комбінований тип засобами ТР?

х Чи повинні всі поля запису мати однаковий тип?

I Як відбувається звертання до полів запису? Які дії можна виконувати з полями записів у ТР?

і *Які операції можна виконувати в ТР над змінними-записами вцілому?*

<> *Для чого призначено 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 повідомленнями необхідні джерело повідомлень, 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 повідомлень - тоді відбувається читання з файла, або приймачем - тоді відбувається запис даних 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 належать пристрої MS-DOS. Пристрої мають фіксовані імена, які можна використовувати у файлових операціях TP з текстовими файлами. Імена пристроїв і зауваження до них наведені у таблиці 18:

Таблиця 18. Стандартні пристрої M3-DQ3

ІМЯ	Пристрій	Пояснення
CON	Консоль (клавіатура і екран)	Введення з CON - це читання з клавіатури, а виведення в CON - це запис на екран
LPT 1 LPT2 LPT3	Паралельні порти (гину 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
COMI COM2	Послідовні порти (тину RS-232) номер 1..2 (якщо встано-	Імена файлів-пристроїв для введення-виведення даних через

	влені)	серійні порти комунікації
AUX	Синонім імені COMI	Файл-пристрій COMI
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 дає змогу програмісту не замислюватися технічними проблемами організації обміну даними, а програмувати сам потік даних. Різні фізичні файли мають різні механізми введення і виведення даних. Логічні ж файли стандартизують роботу з фізичними файлами, що дає змогу 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 типелементів;
Var Ім'язмінної: Ім'ятипу;
```

3. Нетипізовані файли (типу File).

Формат опису:

```
Var Ім'язмінної: File;
```

Для всіх типів файлів мінімальною одиницею даних, 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 ну-

ля. Буферна змінна на відміну від усіх інших змінних не може брати участь у виразах і операторах присвоювання.

Існують два способи доступу до елементів файла: послідовний і довільний (прямий). При **послідовному** способі доступу **по**шук 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 низка процедур (таблиця 19) і функцій (таблиця 20). При їх описі використаємо такі позначення:

F - файлова змінна;

Str - рядковий вираз;

P - змінні р1, р2, рп того самого типу, що й елементи змінної F;

п - 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 видаляється. Покажчик файла встановлюється 1 першу позицію (з номером 0), Фактично файл не містить жодного елемента, його лише підготовлено до завантаження
Reset (F)	Відкрити файл для читання	Виконання процедури забезпечу! встановлення покажчика файлі на перший елемент. Якщо процедура застосована до файла, що HQ існує, виникає помилка введення-виведення
Read(F, P)	Читати з файла	Відбувається чигання з дискового файла, визначеного файловою

		змінною F, значень p1, p2, ..., pn. Після завершення виконання процедури покажчик переміщується до наступного елемента
WriteOF, P)	Записати у файл	Змінні p1, p2, ..., pn записуються у дисковий файл, визначений змінною F. Після виконання процедури покажчик переміщується до наступного елемента
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 файла, призначеного змінній F. Якщо файл був відкритий, ніколи не слід виходити 3 програми, попередньо не закривши його
Erase(F)	Видалити файл	Виконання процедури викликає видалення файла, призначеного змінній F. Якщо здійснюється видалення відкритого файла, його треба попередньо закрити за допомогою процедури Close
Rename(F,Str)	Перейменувати файл	Виконання процедури викликає занесення у *каталог диска новою* імені файла, визначеного змінною F. Нове ім'я визначається значенням Str
Trancate(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, яке дорівнює номеру елемента, на жому встановлено в даний момент покажчик файла, відповідного змінній F. Відлік номера елемента починається з нуля
FileSize(F)	Визначити довжину файла	Функція повертає цілочисельне значення, же дорівнює 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 ({$1-}) для організації обробки помилок 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:

- дані подаються 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 ^{A}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 даних.
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
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 і записувати дані 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 Файловазмінна : Файловий тип; Ім'яфайла : 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 Файловазмінна : Text); - відкриває текстовий файл для зчитування з нього даних;
   б) Rewrite(Var Файловазмінна : Text); - відкриває файл для запису. Ця процедура служить для створення нового файла. Якщо файл з таким ім'ям вже існує, дані з нього будуть видалені;
   в) Append(Var Файловазмінна : Text); - відкриває текстовий файл для дописування в нього даних.

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

4) Для зчитування даних з текстового файла використовуються процедури:
   Read(Var Файловазмінна: Text; Зміннаї [, Змінна2,..., Змін**HaN]:** Char); - читання з файла даних символьного типу;
   Readln(V ar Файловазмінна: Text; Зміннаї: String); - читання **з** файла рядкових даних;
   Readln(Var Файловазмінна: Text); - перехід на новий рядок **у** файлі.

5) Для запису даних у текстовий файл використовуються проце- < Дури:
   Writc(Var Файловазмінна: Text; Зміннаї [, Змінна2,..., Змін**HaNJ:** Char); - запис до файла даних символьного типу;
   Writeln(Var Файловазмінна. Text; Зміннаї: String); - запис до файла рядкових даних;
   Writeln(Var Файловазмінна: Text); - запис у файл порожнього рядка.

6) Для визначення кінця рядка у файлі і кінця файла призначені функції:
   EOLn(Var Файловазмінна. Text): Boolean; - повертає TRUH, якщо досягнуто кінець рядка або кінець файла, інакше - FALSE

EOF(Var Файловазмінна: Text): Boolean; - повертає TRUE, якщо досягнуто кінець файла, FALSE в інших випадках.

7) Після опрацювання файла 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('Введіть 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).

І Іункти 3) і 4) повторюються необхідну кількість разів.

Приклад. Вивести на екран всі елементи файла, який створений 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('Введіть рядки.  zzz - закінчення введення ');
     Readln(S);
        While S <> 'zzz' do
             Begin
               Writeln(TexFile, S);
                 Readln(S)
              End;
    Close(TexFile)
End;
```

Приклад. Дано текстовий файл. Підрахувати кількість рядків файла.

```
Uses  Crt;
Var  f:Text;
     fn:String;  i:Byte;
Begin
  ClrScr;
  Write('Введіть ім"я файла:  ') ;
  Readln(fn);
  Assign(f,  fn) ;
  Reset(f);    i:=0;
```

```
  while not EOF(f) do
    Begin
      Readln(f);    i:=i+1;
    End;
  Close(f);
  Writeln('У файлі f ',i,' рядків');
  repeat until keypressed
End.
```

Приклад. Створити текстовий файл. Порівняти у ньому кількість рядків парної довжини і рядків, що містять літеру "к".

```
Var  f:Text;
     s:String;i,p,k:Byte;
Begin
  Assign(f,'proba.dat');
  Rewrite(f) ;
  Readln(s) ;
  while s<>'...' do
    Begin
      Writeln(f, s) ;
      Readln(s)
    End;
  Reset(f);
  p:=0;  k:=0;
  while not eof(f) do
    Begin
      Readln(f,s) ;
      if length(s) mod 2 = 0 then p:=p+1;
      if pos('k',s)>0 then k:=k+1
    End;
    if p>k then Writeln('рядків парної довжини більше');
        else
         if p<k then Writeln('рядків з літерою "к" більше');
           else Writeln('рядків з літерою "к" і рядків парної
                         довжини рівна кількість')
End.
```

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

```
Uses  Crt;
Var  fl,  f2  :  Text;
     FN1,  FN2  :  String[12];
     S:  String[60];
     1,  max  :  Byte;
Begin
  StvorTFile(f1);
  Write('Введіть ім"я файла для опрацювання:');
  Readln(FN1);
  Assign(fl,  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(fl)) do
      Begin
        Readln(fl, S);
        1 := Length(S);
        if 1 > max then max := 1
      End;
    Close(fl);
    Writeln('Максимальна довжина рядка = ', max);
    Writeln('Рядки, що мають максимальну довжину:');
      Reset(fl);
    While not (EOF(fl)) do
      Begin
        Readln(fl, S);
        l:=Length(S);
        if 1 = max then
          Begin
            Writeln(S);
            Writeln(f2, S);
          End;
      End;                    ' •
    Close(fl);  Close(f2);
End.
```

Приклад. Дано текстовий файл datal.txt. Деякі зі слів, які записані 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 TFill, Tfil2 : Text;
    St, Stl : String;
    i : Byte;
Begin
  Assign(Tfill, 'datal.txt');
  Assign(Tfil2, 'data2.txt');
  Reset(Tfill); Rewrite(Tfil2);
  Readln(Tfill, St); {читання першого рядка}
  While not (EOF(Tfill)) do
    Begin
      Readln(Tfill, Stl); {читання наступного рядка}
      if St{Length(St)] = '-' {останній символ - знак перенесення}
        then
```

```
        Begin
          Delete(St, Length(St),1);{вилучення знака
                                    перенесення}
          i:=1;
          While Stl[i] o " do i := i + 1; |знаходження
                "хвоста" слова у наступному рядку}
          St := St + Copy(Stl, 1, i); {дописування
                "хвоста" до попереднього рядка}
          Delete(Stl, 1, i) {вилучення "хвоста" з
                наступного рядка}
        End;
      Writeln(Tfil2, St); {запис виправленого рядка у новий
файл}
      St := Stl    {переприсвоювання рядків)
    End;
  Writeln(Tfil2, St); {запис останнього рядка у новий файл}
 Close(Tfill); Close(Tfil2)
End.
```

Опрацювання файлових помилок

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

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

Управління опрацюванням помилок здійснюється за допомогою директив компіляції:

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

{$1-} - режим перевірки правильності введення-виведення виключено. В цьому режимі при виникненні помилки виконання програми 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 $1+ . Цей ключ компіляції має локальну сферу впливу. Можна багаторазово включати і виключати режим, вставляючи у текст програми конструкції {$1+} і {$1-}, створюючи області з контролем введення-виведення і без нього.

Приклад.

```
{$1-}  {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

Часто буває зручно програми опрацювання файлів компілювати як виконувані файли-і здігускати 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(O), 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(O) поверне повне ім'я файла TURBO.EXE (з указаниям шляху до нього).

Щоб забезпечити зручність користування і "дружність" програми, слід робити так, щоб при відсутності параметрів при запусте)' програма запитувала їх з екрана. 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 є скінченним упорядкованим набором елементів певного типу. Елементи файла пронумеровані, нумерація починається з нуля.

Формат опису:

Type Ім'я_типу = File oftипелементів;

Var Ім'язмінної: 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 даних. Послідовний доступ до елементів типізованих файлів можливий, але неефективний.

Для роботи з типізованим файлом необхідно:

1) Визначити файлову змінну (змінну логічного файла):
   VarF : 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 Файловазмінна : Файловийтип; Ім'яфайла : 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 Файловазмінна : Файловий тип); - відкриває існуючий файл для зчитування, зміни і додавання нових даних;
   б) Rewrite(Var Файловазмінна : 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 вилучені.

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 на елемент з номером нуль. Насправді нумеруються не самі елементи файла, а межі між ними: пе« ред першим елементом - номер 0, між першим і другим - 1 і т.д.

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

4) Для зчитування даних з типізованого файла використовується процедура:
   Read(Var Файлзмінна: Файлтип, Змінна1 [, Зм2,..., 3MN|: Типелементів); - читання з файла даних, тип яких є базовим ти« пом 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 Файлзмінна: Файлтип, Змі [, Зм2,..., 3MN]: Типелементів); - 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 Файлзмінна: Файлтип): Longlnt;- 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 Файлзмінна: Файлтип): Longlnt; - 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. У самому кінці файла FilePos повертає число, яке дорівнює реальній кількості елементів у файлі, тобто FileSize
3. У решті випадків функція FilePos повертає значення, на 1 менше реального номера елемента, який готовий для читання або створення.

X) Для реалізації довільного доступу до відкритого файла передбачена процедура:

Seek(Var Файлзмінна : Файлтип, N: Longlnt); - встановлює файловий покажчик у необхідну позицію. В параметрі N має бути іаданий 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, l'ileSize(f)).
Дія доступу до останнього елемента файла - Seek(f, FileSize(f)-l).

") Для визначення кінця файла призначена функція:

EOF(Var Файлзмінна: Файлтип): Boolean; - повертає TRUE, якщо досягнуто кінець файла, FALSE в інших випадках (аналогічно текстовим файлам).

10) Після опрацювання файла необхідно закрити. Для цього, як і у випадку текстових файлів, призначена процедура
Close(Var Файлзмінна: Файлтип);

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 зчитаних даних;
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) зчитати 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 ело* мента. Тому для збереження відкоректованого елемента 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 за елементами 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* з парними номерами, які 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(Fil01d, comp);
        if (comp mod 3 = 0 ) and (comp mod 5 <> 0)
          then Write(FilNew, comp); {поповнення нового файла]
        i := i +2  {знаходження наступного парного номера елемента 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.
```

Приклад. Створити файл /із заданої кількості цілих чисел. Переписати у файл g всі елементи файла/ що є повними квадратами. Вивести вміст файла ***g*** на екран.

```
Uses Crt;
Var f,g:File of Integer;
    i,n:Byte;el:Integer;
Begin
  ClrScr;
  Write('Введіть кількість елементів файла f: ');
  Readln(n);
```

```
  Assign(f, 'num.dat') ;
  Assign(g,'sqr.dat');
  Rewrite(f);
  Writeln('Вводьте цілі числа - елементи файла:');
  for i:=1 to n do
      Begin
        Readln(el);
        Write(f,el)
      End;
  Reset(f);
  Rewrite(g);
  for i:=l to n do
    Begin
      Read(f,el);
      if  sqr(round(sqrt(el)))=el
          then Write(g,el)
    End;
   Seek(g,0);
   Writeln('Файл g:');
   for i:=l to FileSize(g)  do
      Begin
          Read(g,el);
          Writeln(el)
      End;
   Close(f);  Close(g);
  repeat until keypressed
End.
```

Нетипізовані файли

У TP введено особливий файловий тип, який фактично є узагальненим файловим типом. Нетипізований файл, як і типізований, складається з машинних представлень даних. Проте типізовані файли можуть містити дані заздалегідь оголошеного типу, а нетипізовані - довільні набори байтів незалежно від їх структури і природи.

Нетипізований файл - потужний засіб роботи з файлами, оскільки дає змогу маніпулювати з даними, не замислюючись про їх тип.

Введення-виведення у нетипізовані файли здійснюється спеціальними процедурами BlockRead і Block Write. 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:

Vnr Ім'язмінної: File;

І Іроцедури Assign і Close зберігають свій синтаксис і зміст.

І Іроцедури Reset і Rewrite можуть мати розширений синтаксис:

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

Kc\vrite([Var Файлзмінна : File [, Розмір_буфсра: Word]);

Розмірбуфера - розмір елемента (буфера файла) у байтах (за замовчуванням -128 байт).

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

Мінімальний блок, який можна записати або прочитати з файла, - 1 байт. Максимальний розмір блока не може перевищувати 64K.

Для зчитування і запису даних у нетипізований файл використовуються процедури:

BlockRead(Var Файлзмінна : File; Var Bufln; N : Word [; Var NIn : Word]);

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

Ці процедури здійснюють читання в змінну Bufln або запис зі змінної BufOut N блоків, які складаються з кількості байтів, які визначено у процедурі відкриття файла. При цьому повинна виконуватись умова N * Розмірбуфера < 64K. Необов'язковий параметр NIn повертає кількість блоків (буферів), зчитану поточною операцією BlockRead. Аналогічний параметр NOut після кожної операції запису показує кількість блоків (буферів), які записані в даний файл операцією Block Write.

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 значення параметриів NIn і NOut будуть дорівнювати цілій кількості 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: Longlnt;
 Begin
   if ParamCount<>2 then
     Begin
       Writeln('Потрібні 2 імені файлів!');
       Halt;
{$1-1
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 (NwONr) ;
Close(Fin);
Close(FOut)
End.
```

Запитання 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. *Уяке місце файла можна додавати нові елементи: на початок, у кінець, у 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. *У яке місце текстового файла можна додавати нові елементи: на початок, у 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 дані з файла і записувати в нього нові дані?*
19. *Як здійснюється створення нового файла? Наведіть приклади,*
20. *Як здійснюється опрацювання файла з послідовним доступом? Наведіть приюіади.*
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 задачу розбивають на окремі частини, кожну з яких поступово уточнюють. 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+bf*** при заданому ***n.***

$$[a + o) \quad = L^n a \quad o \quad + Ь_{„} a \quad o \quad + C_{„} a \quad o \quad + \ldots + L^n a \quad b \quad .$$

$$k\backslash(n\sim k)\backslash'$$

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(к:Integer) ;
Var i: Integer;
Begin
  for i:=0 to к 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:=l;
  for i:=l 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:=l;
   for i:=l 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 к do
    Begin
      Write(C(k,i));
      Write('a',  k-i);
      Write('Ь',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 TP.

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

Uses Ім'ямодуляї [, 1м'я_модуля2, ...];

(тут Uses (використовує) - службове слово)

Після цього можна використовувати вміст приєднаних бібліотек.

Переваги використання модулів:

1) можливість створення власних бібліотек процедур і функцій, які можна підключати до різних програм, не вимагаючи переробки останніх;
2) можливість створення програм великих розмірів (ні програма, ні модуль не можуть створити виконуваний код обсягом більше 64K, 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 приєднуватися 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 приєднанні 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, записуються тількислово End. У цьому розділі описуються дії, які мають виконуватись перед виконанням програми, до якої приєднано модуль. Найчастіше це надання змінним початкових значень. Більшість модулів не має блоку ініціалізації.

Приклад. Опишемо модуль, який містить функцію обчислення факторіала.

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

Щоб розроблений модуль можна було використати в інших програмах, слід відкомпілювати його на диск, встановивши у середовищі програмування опцію Compile - Destination - Disk.

У системі TP визначено ряд стандартних модулів. Всі вони, крім System, під'єднуються до програми директивою Uses:

System - автоматично поєднується до будь-якої програми: містить 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 даних на принтер.

Запитання 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 під'єднуються модулі до Паскаль-програм?*
4. *Яку структуру має модуль у ТР?*
5. *Як створити модуль у її Р-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.

Змінні, які розглядалися раніше, є **статичними.** При їх описі виділяється певний обсяг пам'яті, який не змінюється протягом виконання програми. Так, всі глобальні змінні і типізовані константи, оголошені в ТР-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, яка містить дані певного типу.

Формат опису типізованого вказівника:

Var Ідентифікатор: ^Ім'я_базового_типу;

Базовим може бути довільний тип мови ТР.

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 рг. Ця змінна має ім'я рг^Л. Спочатку вона не має значення. Надати значення змінній можна будь-яким зі способів, визначених у ТР, наприклад, за допомогою оператора присвоювання:
рг^А:=3.2;Зі змінною рг^Л 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 ТР.
pr:=Nil;

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

Dispose(Var p: Типізованийвказівник);

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

Приклад. Розглянемо дві змінні-вказівники:
Var i, j : ^Integer;
На початку роботи програми і і j ні на що не вказують:
i ®
j ®
Далі в і і j записуються адреси, на які вони будуть вказувати:
New(i); i О - Ю i^Л
New(j); j О - Ю j^А
Після цього цілим змінним i^Л і j^А можна надати значення:
i^А := 8; i 0->® i^л
j^А:=6; j О ^ Ф j^А
Якщо змінній i^А присвоїти значення j^А, результат виглядатиме так.
i^А:=j^А; i 0 - > ® i^А
j 0 - » © j^А
Якщо змінній і присвоїти значення j, результат буде таким:
i :=j; i О © - це значення недоступне до
\^кінця роботи програми, бо
j О—>© j^А 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;

End.Dispose(pSt); Dispose(pMas);
```

Нетипізовані вказівники використовуються тоді, коли тип даних у пам'яті, на яку буде здійснюватись посилання, не важливий. Для опису таких вказівників призначений тип Pointer. Його значення, як і значення типізовано вказівників, займають 4 байти.

Приклад. `Var p, pi : 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 помішується 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 саме значення, як у процедурі GetMem. Використовуючи цю процедуру, 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}
```

Приклад.

Дано 10 числових масивів, кожен з яких містить від 80 до 120 дійсних чисел. Вводячи послідовно кожен масив з клавіатури, розташувати числа цього масиву в порядку зростання і в упорядкованому вигляді вивести на екран (використовуючи вказівники, програму скласти так, щоб у кожний момент її виконання в оперативній пам'яті комп'ютера знаходилось не більше одного масиву).

```
Const maximum=120;
Type a=Array[1..maximum] of Real;
     pa=^a;
Var k, i, imin, j, size: Integer;
    temp: Real; p: pa;
Begin
  for k:=l to 10 do
    Begin
      New(p);
      Wrirein('Введіть кількість чисел у масиві');
      Readln(size);
      Wrireln('Введіть ', size, ' чисел');
      for i:=1 to size do
          Readln(p^[i]);
       for i:=l to size-1 do
        Begin
            imin:=i;
            for j:=i+l to size do
             if p^[j] < p^[imin]
                then imin:=j;
            temp:=p^[min];
            p^[min]:=p^[i];
            p^[i]:=temp
         End;
         Writeln;
          for i:=l to size do
         Write(p^[i]:13:6);
         Writeln;
         Dispose(p);
    End
End.
```

Запитання для самоконтролю

1. *У чому відмінність 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 належать список і дерево.

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

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

1^ введення

Т **^1** виведення

вершина

Рисі

Новий елемент додається у вершину стека. Вилучається елемент також з вершини стека. Отже, елемент, який останнім потрапив до стека, буде вилучено 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)

введення -> виведення

т т

кінець початок

Рис.2

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

Існують різні способи реалізації стека 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 масиву OneStack.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, оскільки описаний стек містить елементи дійсного типу. Оскільки неможливо занести елемент у повний стек, то спочатку слід перевірити, 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 заповнено')
  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, значення вказівника вершини стека зменшується на одиницю. Отже, якщо 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.ElemfOldStack.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+I елемент.

Оскільки початок і кінець черги постійно перемішуються в межах масиву, зручно організуватимасив у вигляді "кільця". Щоб обходити масив "по кільцю", використовується операція mod. Якщо Rear<QueueSize-l, він збільшується звичайним способом. Після того як Rear досягає значення QueueSize-I, він набуває значення 0. Це можна перевірити: (QueueSize-T)+l=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+l =6, б 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 перевірки на заповнення черги. Процедура AddQ перед перевіркою збільшує Rear, тим самим перевіряється, чи є більше одної вільної позиції між Front і Rear. Якщо є лиши 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.

Також слід зазначити, 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;
```

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

Процедура вилучення елемента зі списку:

```
Procedure DelList(var elem : pNode);
Var eleml: pNode;
Begin
   eleml:= elem^.next;
   Dispose(elem);
   elem:=eleml
End;
```

Приклад.

З рядків довжиною до 20 символів, що складаються з великих латинських літер, побудувати зв'язний список, в якому елементи розташовані в алфавітному порядку. Список вивести на екран.

Для створення впорядкованого списку використаємо процедуру AddList. Ця процедура проглядає наявні елементи списку, лексикографічно порівнює з ними рядок, що вводиться, знаходить для нього належне місце і за допомогою процедури Ins включає його до списку. Якщо у списку ще немає жодного елемента (список порожній), або рядок, що вводиться, «менший» за перший елемент списку, новий рядок вставляється в початок списку. Якщо ж після перегляду всіх рядків виявиться, що новий рядок «більший» за останній елемент списку, він буде вставлений в кінець списку.

```
Type Line=String[20];
      Link=^List;
      List=Record
             inf: Line;
             next: Link
               End;
Var name: Line;
      ptbase: Link;
Procedure AddList(newname: Line; Var base: Link);
   Var node, epos: Link;
   Procedure Ins(Var bas: Link);
        Begin
         node^.next:=bas;
         bas:=node;
         node:=Nil
        End; {Ins}
 Begin
     New(node);
     node^.inf:=newname;
     if (base=Nil) or (newname < base^.inf)
         then Ins(base)
      else Begin
             epos:=base;
             while (cpos^.next <> Nil) and
                    (cpos^.next^.inf < newname) do
                   epos:=cpos^.next;
```

```
                Ins(cpos^.next)
                 End
 End;  {AddList}
Procedure PrintList(Var base: Link);
    Var epos: Link;
     Begin
          epos:=base;
          while epos <> Nil do
                Begin
                     Writeln(cpos^.inf);
                      epos:=cpos^.next
                End
     End;  {PrintList}
Begin
    ptbase:=Nil;
    Writeln('Введіть список порядково;');
    Writeln('Для завершення введення наберіть ***');
     repeat
          Readln(name);
          if name <>  •>***'
               then AddList(name, ptbase)
     until name='***';
     PrintList(ptbase)
End.
```

Дерева

Якщо кожен вузол списку має два вказівники, така структура називається **бінарним деревом.** Початкова точка бінарного дерева називається кореневим вузлом. Кожен вузол дерева має дві гілки - праву і ліву. Виключенням є лише вершини найнижчого ярусу. Дерева зручно використовувати тоді, коли необхідно додавати і вилучати дані з певним чином упорядкованої структури. Наприклад, дерево на рис. 3 зберігає послідовність A C K D Y.

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 BaseONil 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 BaseONil do
    if Base^.Data=S then
      SearchTree:=TRUE
    else if Base^.Data>S then
              Base:=Base^.Left
            else Base:=Base^.Right;
End;
```

Запитання для самоконтролю

1. *Чим розрізняються статичні і динамічні структури даних?*
2. *Що таке стек, який принцип роботи стеку?*
3. *Як реалізується стеку TP? Наведіть приклади.*

4. *Що таке черга, який принцип роботи черги?*
5. *Як реалізується черга у ТР? Наведіть приклади.*
6. *Що таке список? Як здійснюються основні операції зі списками у ТР?*
7. *Що таке дерево? Як здійснюється опис і опрацювання дереву ТР?*

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

Монітор комп'ютера призначено для виведення на екран текстових і графічних даних. Монітор працює під управлінням спеціального апаратного пристрою - відеоадаптера, який може працювати в двох режимах - текстовому і графічному.

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

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

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

Режим екрана 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 ТР дає змогу працювати у графічному режимі. Для цього призначено 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 ТР 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 задания 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:

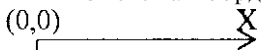


Рис.4.

Для визначення максимальних координат екрана використовують функції:

Function GetMaxX: Integer;
Function GetMaxY: Integer;

Існує поняття "графічний курсор" (ГК), який виконує ті самі функції, що і у текстовому режимі, але є невидимим. Для визна-

чення 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 екрана в графічному режимі призначено 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 можна визначити графічне вікно - прямокутну частина екрану, у верхній лівий кут якої переноситься початок системи координат. Для цього призначено процедуру:

Procedure SetViewPortfxl, yl, x2, y2: Integer; ClipMode:; Boolean);

(xl, yl)-(xl,y2) - діагональ вікна;

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

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

Procedure ClearViewPort;

Побудова основних графічних примітивів

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

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

Procedure Line(xl, yl, x2, y2: Integer); - будує відрізок, кінці якого мають координати ***(xlyl), (x2y2);***

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

Procedure LineRelfdx, 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 Line Style - тип ліній: 0 - звичайна лінія, 1 - точкова, 2 - пггрихпунктирна, 3 - пунктирна, 4 - тип ліній явно задається шаблоном;

параметр Pattern - "зразок": якщо LineStyle ф 4, то дорівнює 0, інакше необхідно задати шаблон з 16 бітів (1 - світиться, 0 - ні: 1100110011001100), перевівши у десяткову або шістнадцяткову систему числення;

параметр Thickness - товщина ліній: 1 - нормальна товщина (1 ігіксель), 3 - потовщена лінія.

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

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

Procedure SetBkColot(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^O;	чорний
B1ue=1;	синій
Green=2;	зелений
Cyan-^3;	блакитний
Red-4;	червоний
Magenta=5;	бузковий
Brown=6;	коричневий
LightGray^J;	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
While-15;	білий

Для зафарбовування замкнених областей попередньо встановити шаблон і колір заливання:

Procedure SelFillSlyle(Paltern: Word; Color: Word);

Pattern визначає вигляд шаблону заливки;

Color визначає колір шаблону.

Для параметра передбачені константи:

0 - заливання (суцільне) кольором фону;

1 - заливання (суцільне) поточним кольором;

2 - заливання лініями типу = = = = = =;

3 - заливання лініями типу ////////////////;

11 - заливання точками ::::::::::::::.

Procedure Rectanglefxl,yl,x2,y2: Integer); **-** малює прямокутник з діагоналлю (xl,yl) - (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[l..n] of PointType;

Для зображення кола використовується процедура

Procedure Circle(x,y: Integer; Radius: Word);
(x,y) - центр кола, Radius - радіус кола.

У модулі Graph є процедури малювання еліпсів, дуг, секторів. Відповідні процедури вимагають задания параметрів StartAngle і EndAngle, які позначають початковий і кінцевий кут дуги. Кути відміряються від додатного напрямку осі X проти годинникової стрілки у градусах.

Procedure Aгc(x, y: Integer; StartAngle, EndAngle, Radius: Word);- побудувати дугу кола радіуса Radius з центром (x, y) від кута StartAngle до EndAngle;

Procedure Ellipsefx, y: Integer; StartAngle, EndAngle, XRadius, YRadius: Word); - побудувати еліптичну дугу, XRadius і YRadius - розміри горизонтальної і вертикальної полуосей. Для зображення еліпса слід задати StartAngle=0, EndAngle=360.

Заливання областей зображення

Procedure Barfxl, yl, x2, y2: Integer); - побудувати прямокутник з діагоналлю (xl, yl)-(x2, y2), внутрішня область якого залита за поточним шаблоном;

Procedure Sectorfx, y: Integer; StartAngle, EndAngle, XRadius, YRadius: Word); **-** побудувати сектор еліпса, залитий за заданим шаблоном;

Procedure PieSlicefx, y: Integer; StartAngle, EndAngle, Radius: Word); **-** побудувати сектор кола поточного кольору, залитий за поточним шаблоном;

Procedure FillEllipsefx, y: Integer; XRadius, YRadius: Word); **-** побудувати еліпс поточного кольору, залитий за поточним шаблоном;

Procedure FillPoly(NumPoints: Word; Var PolyPoints); - заповнити багатокутник 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 OutTextXYfx, 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 SetTextStylefFont, 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 його в певному режимі в потрібне місце екрана. Так можна створювати рухомі зображення.

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

Function ImageSizefxl, yl, x2, y2: Integer): Word;

(xlyl)-(x2j>2) - діагональ прямокутника, який охоплює фрагмент.

Отже, обсяг зображення: Size = ImageSize(x/, ***yl, 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 Getlmage(xl,yl, x2, y2: Integer: Var Q);

xl, yl, x2, y2 мають ті самі значення, що у ImageSize; Q - змінна, яка займає область пам'яті розміром ImageSize. Це змінна P^{A}:

Getlmage(xl,yl,x2,y2, P^);

Для виведення зображення з буфера на екран використовується процедура:

Procedure Putlmagefxl, yl: Integer; Var Q; Mode: Word);

xl,yl - координати лівого верхнього кута нового положення фрагмента, Var Q вказується, як у Getlmage, Mode - режим виведення зображення (суміщення зображень фрагменту і фона). У модулі Graph визначено для параметра Mode такі константи:

CopyPut=0; заміщення (очищення перед малюванням)
XORPut=l; 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, то воно спочатку 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):
Putlmage(xl l, y 11, P^{A}, XORPut); Delay(20);
Putlmage(xl l, y 11, $P^{л}$, XORPut);

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

(xl, yl), (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 xl, x2, yl, 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');
xl:=l; yl:=l; x2:=r; y2:=r;
maxx:=GetMaxX; maxy:=GetMaxY;
sx:=xl; sy:=yl;
sxmove:=3; symove:=3;
SetColor(4); SetBkColor(0);
SetFillStyle(1,14);
Bar(xl,yl,x2,y2);
size:=ImageSize(xl,yl,x2,y2);
GetMem(p, size);
Getlmage(xl,yl,x2,y2,pA);
```

```
repeat
    PutImage(sx,sy,pA,XORPut) ;
    Delay(20);
    PutImage(sx, sy,pЛ, 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.
```

Запитання для самоконтролю

1. *Чим розрізняється робота в текстовому і графічному режимах роботи комп'ютера?*
2. *Що таке роздільна здатність?*
3. *Як реалізується графічний режиму 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?*

Лабораторна робота №1

ТЕМА: 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 такс 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 такс оператор?
6. Опишіть структуру Паскаль-програми.
7. Що таке коментар? Для чого призначені коментари?
8. Для чого призначений оператор присвоювання? Наведіть приклади його використання.
9. Опишіть дію операторів введення і виведення. Наведіть приклади.
10. Назвіть стандартні скалярні типи мови Паскаль. Які характеристики вони мають?
11. Що таке структурованийтип даних? Наведіть приклади.
12. Назвіть основні операції і функції для величин цілого типу. Наведіть приклади.
13. Назвіть основні операції і функції для величин дійсного типу. Наведіть приклади.

14. Назвіть основні операції і функції для величин логічного типу. Наведіть приклади.
15. Назвіть основні операції і функції для величин символьного типу. Наведіть приклади.
16. Вкажіть формат і опишіть дію умовного оператора. Наведіть приклади.
17. Вкажіть формат і опишіть дію оператора варіанту. Наведіть приклади.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1

1. Дано числа ***a, Ь*** і ***c*** (де ***a*** <> ***0***). Знайти дійсні корені рівняння

$$ax^4 + Ьx^2 + c=0$$

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/(Ь-c)*** , де ***a*** - значення двох рівних чисел, ***b*** - значення більшого, а с - 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 ***0(xl,yl),rl, 0(x2,y2),r2.*** Вивести, як розташовані ці кола (перетинаються чи ні, одне в іншому).

ВАРІАНТ 3

1. Дано три числа ***a,Ь,c.*** Якщо ці числа можуть розглядатись як сторони трикутника, то необхідно обчислити площу цього трикутника за формулою Гсрона. В іншому разі вважати 5=0.
2. Дано чотири числа. Якщо сума найменшого і найбільшого більша 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(xl,yl), B(x2,y2)*** і ***C(x3:y3).*** Якщо трикутник рівнобедрений, то вивести значення кута між рівними сторонами, інакше вивести значення його периметра.

ВАРІАНТ 5

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

$$f(x,y)=\begin{cases}\cos x+\sin y, & \text{якщо } (x,y) \text{ належить } D;\\ x\cdot y, & \text{якщо } (x,y) \text{ не належить } D.\end{cases}$$

2. Дано чотири числа ***Ь, 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

$$f(x,y)=\begin{cases}y-|x-1|, & \text{якщо } (x,y) \text{ належить } D;\\ x^2+y^2, & \text{у протилежному випадку.}\end{cases}$$

2. Три точки задані своїми координатами ***A(xl,yl), B(x2,y2)*** і ***C(x3,y3).*** Знайти ту з трьох точок, яка лежить найближче до початку координат. Координати інших точок замінити координатами середини відрізка, 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

$$f(x,y)=\begin{cases}x^2+y^2-1, & \text{якщо } (x,y) \text{ належить } D;\\ \dfrac{1}{|x+y|+1}, & \text{у протилежному випадку.}\end{cases}$$

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

ВАРІАНТ 8

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

$$J(x,y)=\begin{cases}x^5+e^y, & \text{якщо } (x,y) \text{ належить } D;\\ x^2+4y-5, & \text{у протилежному випадку.}\end{cases}$$

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

$$f(x,y)=\begin{cases} x-y, & \text{якщо } (x, y) \text{ належить } D;\\ x^2+y^2, & \text{у протилежному випадку.}\end{cases}$$

2. Дано точки ***A(xl,yl), 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

$$f(x,y)=\begin{cases} x^2+y^2, & \text{якщо } (x, y) \text{ належить } D;\\ x-y, & \text{у протилежному випадку.}\end{cases}$$

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

Лабораторна робота №2

ТЕМА: ЦИКЛІЧНІ ПРОГРАМИ (ЦИКЛИ WHILE, REPEAT)

МЕТА: Ознайомитись з операторами циклу з передумовою та післяумовою. Вивчити особливості використання кожного з операторів. Навчитися застосовувати оператори циклу в процесі програмування ітераційних процесів та табулювання 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,Ь]*** з кроком ***h.***

X	*F(x)*

Одне з завдань виконати з використанням циклу WHILE, інше - REPEAT.

ВАРІАНТ 1

1. $y = \sqrt[3]{4x^4} + б$.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„=sin(n).***

ВАРІАНТ 2

1. $y = yx^7 - x.$ Якщо обчислене значення *y* більше 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„-$in (n).***

ВАРІАНТ 4

1. $y - x + 7x + 10.$ 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„-sm(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 *к* членів послідовності, більших за *m*, та їх номери: *A»= 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 *к* членів послідовності, менших за *0:: A„=2-nsin(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 суму ряду із заданою точністю *eps,* вважаючи, що потрібна точність досягнута, якщо модуль наступного доданка менший *eps:* $\pounds \frac{(-1)'}{\text{fi}(i+1)(/+2)}$

ВАРІАНТ 8

1. $y = x^2 + лі\text{-}x + 6$. Виведення на екран припиняється, якщо значення *x* не належить ОДЗ. Подається відповідне повідомлення.
2. Обчислити суму ряду із заданою точністю *eps,* вважаючи, що потрібна точність досягнута, якщо модуль наступного доданка менший *eps:* $\pounds_{i=0} \frac{\text{Н У}}{4 + J}$

ВАРІАНТ 9

1. $y = x^2 + \text{-у/-}\, x + \sin(x) + 6$. Виведення на екран припиняється, якщо значення *x* не належить ОДЗ. Подається відповідне повідомлення.
2. Обчислити суму ряду із заданою точністю *eps,* вважаючи, що потрібна точність досягнута, якщо модуль наступного доданка менший *eps:* $\sqrt[\circ\circ]{\frac{(-1)'}{}}$.

ВАРІАНТ 10

1. $y = \frac{1}{x}$ь $\sqrt{x + 3}$ + б. Виведення на екран припиняється, якщо значення x не належить ОДЗ. Подається відповідне повідомлення.
2. Обчислити суму ряду із заданою точністю ***eps,*** вважаючи, що потрібна точність досягнута, якщо модуль наступного доданка менший ***eps:*** $\sum_{i<=0}^{\infty} \frac{(-1)^i}{2^i}$.

Лабораторна робота №3

ТЕМА: ЦИКЛИЧНІ ПРОГРАМИ (ЦИКЛ FOR)

МЕТА: Ознайомитись з оператором циклу з параметром. Вивчити особливості використання оператора. Засвоїти порівняльні характеристики всіх операторів циклу TP. 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

1. Що називається циклом у програмуванні? Які оператори мови TP призначені для організації циклів?
2. Вкажіть формат оператора циклу з параметром. Які існують модифікації цього оператора? Як вони виконуються?
3. Що називається кроком циклу?
4. Порівняйте особливості виконання трьох операторів циклу мови TP.
5. Що називається вкладеним циклом'/ Наведіть приклади.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1

1.Обчислити суму перших ***n*** членів послідовності:

$$V = 1 + \frac{1}{8} + \frac{1}{27} + \frac{1}{64} + \ldots.$$

2. Дано цілі числа ***a;, a², a³.*** Отримати цілочисельну матрицю ***B³ˣ³,*** для якої ***Ьг]=a²3a,, i,j = 1,2,3.*** Обчислити середнє арифметичне елементів побічної діагоналі цієї матриці.

ВАРІАНТ 2

1. Обчислити добуток перших ***n*** членів послідовності:

$$y \quad \frac{1}{2} + \frac{Л\ (i}{3)\ U} \quad \frac{Л\ f\ 7 + \pm\ )}{9 \quad 8 \quad 27^r}$$

2. Дано дійсні числа ***a^h...,a³, b^h...,b⁶.*** Отримати дійсну' матрицю ***Сбхз,*** для якої ***Ci}^a/(1 + \Ьi[).*** Обчислити суму додатних елементів цієї матриці.

ВАРІАНТ 3

1. Обчислити суму перших ***n*** членів послідовності:

$$y = 1 + \frac{3}{4} + \frac{5}{8} + \frac{7}{16} + \frac{9}{32} + \frac{11}{64} + \ldots.$$

2. Дано натуральне число ***n.*** Отримати Л*„ - цілочисельну матрицю, для якої я,у=ї+2/. Обчислити суму елементів, розташованих на головній діагоналі і вище.

ВАРІАНТ 4

1.Обчислити добуток перших ***n*** членів послідовності:

$$y \sim \frac{3}{1} \cdot \frac{4}{2} \cdot \frac{5}{3} \cdot \frac{6}{4} \cdot \frac{7}{5} \cdot \frac{8}{6} \cdot \frac{9}{7} \cdot \frac{10}{8} \cdots$$

2. Дано дійсну квадратну матрицю ***A^.*** Отримати дві квадратні матриці ***Bᵐ„, Cᵘⁿ,*** для яких ***bⁱᵗ =<*** $\begin{matrix} . \\ '\bullet' \end{matrix}$ ***.,*** $c^{if} = \begin{matrix} J \\ v \end{matrix}$ $\begin{matrix} < \\ \{ciji,j<i \end{matrix}$ $\begin{matrix} \\ \backslash -aij,j>i \end{matrix}$

ВАРІАНТ 5

1. Обчислити суму перших ***n*** членів послідовності:

$$y = \frac{1}{9} + \frac{1}{25} + \frac{1}{49} + \frac{1}{81} + \ldots.$$

2. Дано натуральне число ***n.*** З'ясувати, скільки додатних елементів містить матриця ***Aᵐⁿ,*** якщо ***ciij^sin((i²-f)/n).*** Матрицю вивести на екран.

ВАРІАНТ 6

1. Обчислити суму перших *n* членів послідовності:

$$y = 1 + \frac{2}{4} + \frac{3}{16} + \frac{4}{36} + \frac{5}{64} + \ldots$$

2. Дано цілочисельну матрицю A^{10n}. Отримати $b_j,\ldots b_{nn}$, де

ВАРІАНТ 7

1. Обчислити добуток перших и членів послідовності: $2\ \ 4\ \ 6$

2. Дано цілочисельну матрицю A^{mn}. Отримати $b^h\ldots,b^n$, де

$b^x = \|a^n\|$.

ВАРІАНТ 8

1. Обчислити добуток • перших *n* членів послідовності:

$$y = 1 \cdot \frac{}{2\ \ 3} \cdot \ldots$$

2. Дано цілочисельну матрицю A^{mn}. Отримати $b_1,\ldots,b_{nn}$, де

ВАРІАНТ 9

1 .Обчислити добуток перших *n* членів послідовності:

$$y \sim \ ' \ 2' \ \ 2\text{-}3' \ \ 2\text{-}3\text{-}4' \ \ 2\text{-}3\text{-}4\text{-}5''''$$

2.Дано цілочисельну матрицю A^{mn}. Отримати $b\ b^n$, де

$$\text{*} = 2\,K\,I\,-$$

ВАРІАНТ 10

1. Обчислити суму перших п членів послідовності.

$$y = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \ldots$$

2. Дано натуральне число *n.* З'ясувати, скільки від'ємних елементів містить матриця A^{mn}, якщо $a^4 = \cos(i^{2j} rj)$. Матрицю вивести на екран.

Лабораторна робота №4

ТЕМА: 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

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 TP? 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 TP.
10. Що таке рекурсія? Наведіть приклади використання рекурсії.
11. Що такс побічний ефект? Наведіть приклади.
12. Для чого призначені стандартні процедури Exit і Halt?

ВАРІАНТИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

ВАРІАНТ 1.

1. Дано дійсні числа ***a, b.*** Отримати и=тіп(а,6), ***v = mm(ab,a + b),*** тіп(и + v^2,3.14)

2. Дано дійсні числа ***s, t.*** Отримати

$$h(s,t) + max(h(s-t,st),h(s-t,s+t)) + h(\backslash,\backslash),$$

де $h(a,b) = \frac{}{1+b^2} + \frac{}{1+a^2}$.

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 $f(t,-2s,\backslash.\backslash 7) + f(2.2,t,s-t)$,

де $f(a,b,c) = \frac{2a-b - \sin e}{5+c}$.

ВАРІАНТ 3.

1. Чотирикутник (довільний) задається координатами вершин на площині. Знайти мінімальну і максимальну відстань між двома вершинами чотирикутника (розглянути і несуміжні вершини).

„ „ .„ . , $\frac{max(a,a+Ь) + max(a,Ь+c)}{1 + тах(я + 6c, 115)}$.

2. Дано дійсні числа ***a,Ь,c.*** Отримати

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 $я;,\ldots,я_{„}$, $b^1,\ldots,b^m$, $cj,\ldots, c^5$. Отримати

$$(mm(Ь^1,\ldots,Ь^m) + min(c^ь\ldots,c^5), якщз \quad тіп(я_ь\ldots,я_и) > 0;$$

$$[1 + (\max(cj,\ldots,c^5)) \quad \text{в протилежному випадку.}$$

ВАРІАНТ 5.

1. Три квадратні рівняння задаються своїми коефіцієнтами ***a, Ь, c.*** Визначити та надрукувати те з них, що має найбільший корінь.
2. Дано дійсні числа ***s, t, $a^0,\ldots,a^5$.*** Отримати $p(l)-p(t)$ і $p^2(s-t)-p(3)$, де $p\{x)-a^5x^5 + a^4x^4 + \ldots + a^0$.

ВАРІАНТ 6.

1. Три зведені квадратні рівняння задаються своїми коренями $x,, x^2$. Визначити та надрукувати те з них, що має найбільший додатний коефіцієнт при x (вивести рівняння в звичайному математичному вигляді).

„ „ . „ . „

2. Дано дійсні числа $щ, u^2, v^{lt}v^2, w^{lt}w^2$. Отримати $\frac{2u + 3uw}{2 + w - v}$, де u,v,w - комплексні числа $u^г+iu^2, v^t+iv^2, Wj+iw^2$. (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 ^-елементні дійсні вектори *x,yiz.* Обчислити величину (із, а) *- (Ь,с),* де *а* означає той з цих векторів, в якому найбільший мінімальний елемент (вважаючи, що такий вектор єдиний), *b* і с означають два інших вектори, 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.

о п *гл*

2. Дано дійсне число *y.* Отримати $\frac{1.7Д0.25) + 2Д1 + y)}{б-/(y^2-1)}$, де

$j\backslash x) = x + x^3 + x^5 + X^i$.

ВАРІАНТ 9.

1. Чотири вектори у б-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.?, *t.* Отримати *g(1.2,s) + g(t,s)-g(2s - l,st),*

де $g(a,b) = \frac{a^2+b^2}{a^2 + 2ab+3b^2 + 4}$

ВАРІАНТ 10.

1. Чотири трикутники задаються координатами своїх вершин на площині. Визначити трикутник, що має найменший периметр, та обчислити його площу.

o n -

2. Дано дійсне число у. Отримати $\frac{1.7t(0.25) + 2t(l + t)}{6\text{-}t(y^2\text{-}l)}$, де

$10 \quad {}_{x}2к$

$к=1 \quad 2к$

Лабораторна робота №5

ТЕМА: 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 текстових даних.

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 функції 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. 4и 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. Використовувати підпрограму розбиття речення на слова.
2. Завдання виконувати з максимальним використанням підпрограм.

Дано текст, в якому від *1* до ***ЗО*** слів, у кожному' слові від ***1*** до *8* літер, слова розділені пропусками.

ВАРІАНТ 1

1. Надрукувати всі слова тексту, що містять рівно три літери V.
2. Надрукувати всі симетричні слова тексту.
1. Надрукувати всі слова, що мають мінімальну довжину.
2. Порівняти кількість слів парної довжини і слів, що містять літеру '/f.

ВАРІАНТ З

1. Другу літеру кожного слова замінити на У.
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 одної літери, підрахувавши їх кількість.
2. Надрукувати слова, які входять в текст не менше двох разів.

ВАРІАНТ 6

1. Визначити 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 *'о'*.
2. У кожному слові останню літеру перенести на початок слова.

ВАРІАНТ 8

1. Порівняти кількість слів, що містять і не містять літеру *'т'*.
2. Надрукувати слова, довжина яких максимальна.

ВАРІАНТ 9

1. Порівняти кількість слів, що містять літеру *'к'* і слів, що складаються з чотирьох літер.
2. Кожне слово тексту надрукувати в зворотньому порядку.

ВАРІАНТ 10

1. Надрукувати всі слова, що мають довжину, яка дорівнює довжині останнього слова.
2. Перед кожним словом надрукувати кількість входжень в нього літери V.

Лабораторна робота №6

ТЕМА: ОПРАЦЮВАННЯ ЗАПИСІВ

МЕТА: Ознайомитись з можливостями мови Турбо Паскаль (ТР) в опрацюванні даних комбінованого типу. Вивчити особливості опрацювання записів у ТР. Закріпити вивчений матеріал у процесі створення 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 призначено 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 призначено 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 літаки 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 тари, 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) всі 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 номери 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 квитки. Визначити назви рейсів і місця призначення літаків, час польоту яких менший 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 найдорожчі 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 номери груп, що не мають занять у вівторок на 3-й парі.

Лабораторна робота №7

ТЕМА: 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 в процесі

створення власних нескладних програм опрацювання текстових файлів.

ОБЛАдаАННЯ: 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 достуй 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 вказаної користувачем кількості рядків або до введення вказаної 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 "я".
2. Дано текстовий файл *f* Переписати у файл *g* всі елементи файла *f* замінивши в них символи *0* на символи *1* і навпаки.

ВАРІАНТ 3

1. Дано текстовий файл, розбитий на рядки. Передостанню літеру кожного рядка замінити на *'m*
2. Дано текстовий файл /.' Записати в перевернутому вигляді рядки файла /у файл g. Порядок рядків у файлі ***g*** повинен збігатися з порядком рядків у файлі/.'

ВАРІАНТ 4

1. Дано текстовий файл, розбитий на рядки. Кожний рядок тексту надрукувати в зворотньому порядку.
2. Дано текстовий файл / 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 /. Переписати елементи файла /у файл g, дописуючи до початку кожного рядка літеру "о". Порядок рядків повинен бути збережений.

ВАРІАНТ 6

1. Дано текстовий файл, розбитий на рядки. Визначити кількість входжень у текст першого рядка.
2. Дано текстовий файл/ Переписати в файл ***g*** всі рядки ***з/,*** в яких друга літера збігається з передостанньою.

ВАРІАНТ 7

1. Дано текстовий файл, розбитий на рядки. Визначити номер рядка, що містить три літери "о".
2. Дано текстовий файл/, рядок ***s.*** Отримати всі рядки файла/ фрагментом яких є рядок ***s.***

ВАРІАНТ 8

1. Дано текстовий файл, розбитий на рядки. Порівняти кількість рядків, що містять літеру *'к'* і рядків, що складаються з чотирьох літер.
2. Дано текстовий файл / розбитий на рядки. Переписати в файл ***g*** всі рядки ***з/,*** в яких перша літера збігається з останньою.

ВАРІАНТ 9

1 Дано текстовий файл, розбитий на рядки. Останній рядок тексту надрукувати в зворотному порядку.
2. Дано текстовий файл/ 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, яка дорівнює довжині останнього рядка.
2. Дано текстовий файл/. У початок кожного рядка вставити його довжину. Результат помістити в файл *fl.*

Лабораторна робота № 8

ТЕМА: ОПРАЦЮВАННЯ ТИПІЗОВАНИХ ФАЙЛІВ

МЕТА: Засвоїти особливості опрацювання типізованих файлів у ТР. Вивчити стандартні процедури і функції опрацювання типізованих файлів. Закріпити вивчений матеріал в процесі створення 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 дані з файла і записувати в нього нові дані?
10. Як здійснюється створення нового файла? 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 здійснювати опрацювання даних, зчитуючи їх 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 армифмстичне значень елементів файла ***F.***

ВАРІАНТ 5

1. Дано файл /*, елементи якого - дійсні числа. Знайти:
   а) суму елементів файла ***F;***
   б) найменше із значень елементів файла ***F.***

ВАРІАНТ 6

1. Дано файл /", елементи якого - дійсні числа. Знайти:
   а) добуток першого та останнього елементів файла ***F.***

б) суму найбільшого та найменшого значень елементів файла *F*.

ВАРІАНТ 7

Дано файл *F*, елементи якого - цілі числа. Створити файл *G* з квадратів елементів файла *F*, файл *H* - з непарних елементів файла *F*.

ВАРІАНТ 8

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

Дано файл *F*, елементи якого - цілі числа. Знайти:

а) середнє арифметичне другого і передостаннього елементів файла *F;*

б) кількість елементів файла *F*, які кратні *9*.

ВАРІАНТ 10

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.***

Література

1. Абрамов С.А. и др. Задачи по программированию. - М.: Наука, 1988. - 224 с.
2. Абрамов В.Г., Трифонов Н.П., Трифонова Т.Н. Введение в язык Паскаль. - М.: Наука, 1988. - 320 с.
3. Бартків А.Б., Гринчишин Я.Т., Ломакович А.М., Рамський Ю.С. Turbo Pascal: Алгоритми і програми: чисельні методи в фізиці і математиці: Навч. посібник. - К.: Вища школа., 1992. - 247 с.
4. Брукшир, Дж., Гленн. Введение в компьютерные науки. Общий обзор, 6-е издание: Пер. с англ. - М.: Издательский дом "Вильямс", 2001.-688 с.

Верлань А.Ф., Апатова Н.В. Інформатика: Підруч. для учнів 10-11 кл. загальноосв. шк. - К.: Квазар-Мікро, 1998. -200 с.

Вирт Н. Алгоритмы и структуры даннях: Пер. с. англ.- М.: Мир. - 1989. - 360 с.

Вьюкова НИ., Галатенко В.А., Ходулев А.Б. Систематический подход к программированию. - М.: Наука, 1988. - 208 с.

Грогоно П. Программирование на языке Паскаль. - М.: Наука, 1982.

Жалдак М.1., Рамський Ю.С. Інформатика: Навч. Посібник / За ред. М.І.Шкіля. - К.: Вища школа, 1991. - 319 с.

10. Йенсен К., Вирт Н. Паскаль, руководство для пользователя и описание языка. - М.: Финансы и статистика, 1982.
11. Керман, Митчел, К. Программирование и отладка в Delphi. Учебный курс: Пер. с англ. - М.: Издательский дом «Вильяме», 2002. - 672 с.
12. 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.
13. Пильщиков В Н. Сборник упражнений по языку Паскаль. - М.: Наука, 1989. - 160 с.
14. Поляков Д.Б., Круглов И.Ю. Программирование в среде Турбо Паскаль (версия 5.5). - М.: изд-во МАИ, 1992. -576 с.
15. Следзінський І.Ф., Ломакович А.М., Рамський Ю.С, Зароський Р.1. Техніка обчислень і алгоритмізація. - К.: Вища школа, 1991.- 199 с.
16. Фаронов В.В. Программирование на персональных ЭВМ в среде Турбо-Паскаль. - 2-е изд. - М.: Изд-во МГТУ, 1992. - 448 с.
17. Фролов Г.Д., Кузнецов Э.И. Элементы информатики: Учеб. пособие для пед. ин-тов - М.: Высш.шк., 1989. - 304 с.
18. Шкиль Н.И., Жалдак М.И., Морзе Н.В., Рамскии Ю.С. Изучение языков программирования в школе. - к.: Рад. шк., 1988. - 368 с.

Зміст

Навчальне видання

Основи програмування

Дудик Михайло Володимирович
Рамський Юрій Савіянович
Цибко Ганна Юхимівна

Навчальний посібник для студентів вищих навчальних закладів фізико-математичних та індустріально-педагогічних спеціальностей

Рекомендовано Міністерством освіти і науки України

Підп. до друку 19.07.2005. Формат 60x84/16.
Папір офс. Гарнітура Тайме. Друк офс.
Ум. друк. арк. 7,34. Обл.-вид. арк. 6,34.
Тираж 300 прим. Зам. 42-07-05

Видавництво «Міленіум»
Свідоцтво про внесення суб'єкта видавничої справи
до державного реєстру видавців, виготівників
і розповсюджувачів видавничої продукції
ДК№535 від 19.07.2001 р.

м. Київ, вул. Ісаакяна, 18, оф. 101
Тел./факс 230-47-78
E-mail: info@millennium.net.ua
www.millennium.net.ua