ДРОГОБИЦЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА
Інститут фізики, математики, економіки та інноваційних технологій

Ірина Шаклеіна, Ольга Косовська

Програмування

Методичні рекомендації до виконання лабораторних робіт

(для студентів напряму підготовки Технологічна освіта
Спеціалізація: інформатика)

Дрогобич
2013

УДК.004(07)
ББК 32.973.2-018р.
Ш17

Рекомендовано до друку вченою радою Дрогобицького державного педагогічного університету імені Івана Франка як методичні вказівки до лабораторних робіт (протокол № 7 від 20.06.2013 р.)

Рецензенти:

Буров Є. В., кандидат технічних наук, професор кафедри «Інформаційні системи та мережі» національного університету «Львівська політехніка;

Козак Т. М., кандидат педагогічних наук, доцент кафедри інформатики та обчислювальної математики Дрогобицького державного педагогічного університету імені Івана Франка.

Шаклеіна І., Косовська О.

Ш17 **Програмування: методичні рекомендації до виконання лабораторних робіт**. – Дрогобич: Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка, 2013. – 132 с.

Посібник укладено відповідно до програми навчальної дисципліни «Програмування» для підготовки фахівців ОКР «Бакалавр» галузі знань: «0101 Педагогічна освіта» напряму підготовки «6.010103 Технологічна освіта. Спеціалізація: інформатика», затвердженої вченою радою Дрогобицького державного педагогічного університету імені Івана Франка (протокол № 7 від 20.06.2013).

УДК.004(07)
ББК 32.973.2-018р.

ЗМІСТ

ПЕРЕДМОВА

Завдання курсу "Програмування" полягає у вивченні сучасних підходів до розробки програмного забезпечення та створення ефективних комп'ютерних програм. Метою запропонованого курсу є навчити студентів розробляти ефективні комп'ютерні програми, застосовуючи основні принципи алгоритмізації та програмування; сформувати практичні навики роботи в Turbo Pascal та Delphi на базі теоретичних знань, отриманих при вивченні цього курсу.

Методичні рекомендації до виконання лабораторних робіт з курсу «Програмування» укладені на базі двосеместрового курсу, який читається для студентів інженерно-педагогічного факультету напряму підготовки «Технологічна освіта. Спеціалізація: інформатика». До посібника увійшли лабораторні роботи щодо програмування алгоритмів лінійної, розгалуженої та циклічної структури, роботи з масивами різної розмірності; використання підпрограм (процедур і функцій), файлів даних, множин та записів; вивчення принципів роботи з рядками символів та графікою. Передбачено ознайомлення з основами об’єктно-орієнтовного та візуального програмування на прикладі розробки додатків в середовищі Delphi. Усі лабораторні роботи мають однакову структуру: тема, мета, хід роботи, теоретичні відомості, що містять приклади виконання типових завдань з цієї теми та завдання для самостійного виконання. Теоретичні відомості охоплюють основний матеріал, який потрібно знати студенту для виконання завдань лабораторної роботи.

Під час підготовки до заняття студент повинен прочитати теоретичні відомості, спробувати самостійно виконати завдання, передбачені в лабораторній роботі. У деяких випадках при підготовці до роботи потрібно скористатися додатковою літературою, поданою наприкінці посібника.

Заняття з курсу проходять у три етапи: допуск до роботи, виконання роботи та захист роботи. Під час допуску студенти повинні показати мінімальний рівень підготовки, потрібний для виконання роботи. Після виконання роботи її потрібно захистити.

Середовище Turbo Pascal. Елементи та структура запису програми

МЕТА РОБОТИ: засвоєння основних дій для реалізації програм в середовищі Turbo Pascal: робота з редактором, створення файла програми; збереження файла програми; компіляція програми та запуск на виконання.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Середовище програмування Turbo Pascal призначене для підготовки текстів програм мовою Pascal та їхнього виконання. Для входу у середовище потрібно виконати команду turbo.exe. У верхньому рядку екрана буде розташоване головне меню, а в нижньому – опис деяких функціональних клавіш (рис.1).

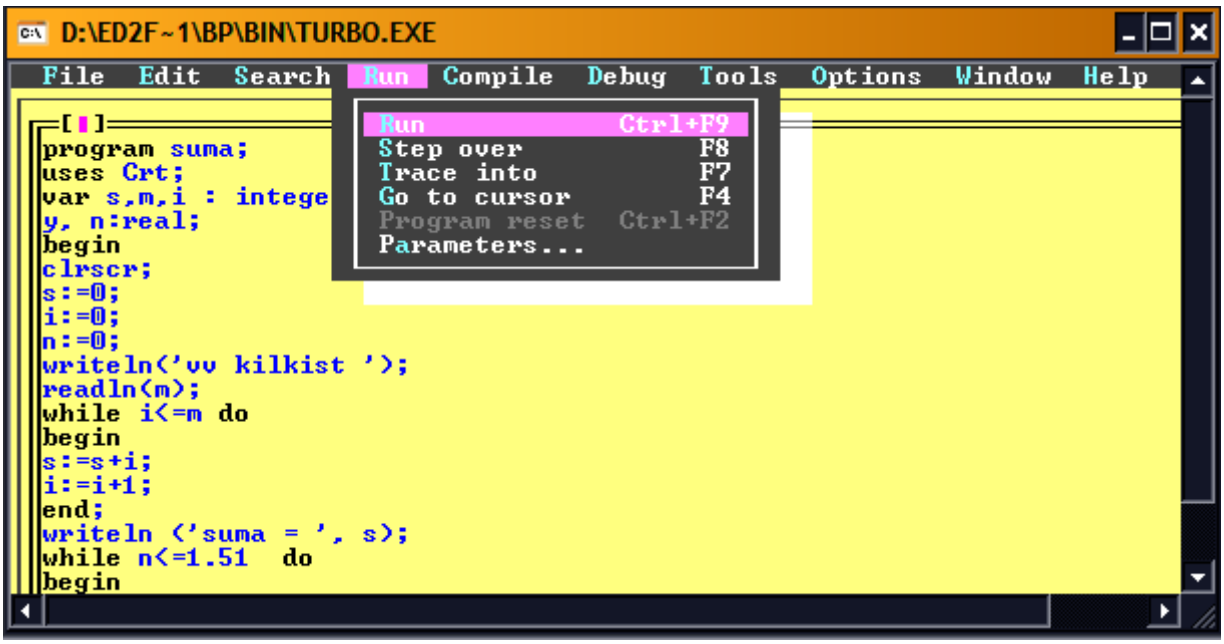


Рис.1. Головне вікно Turbo Pascal 7.0.

У розпорядженні користувача будуть такі пункти головного меню:

File – для роботи з файлами;

Edit – для редагування файлу;

Search – для відшукання чи заміни заданого фрагмента тексту;

Run – для виконання програми;

Compile – для компіляції програми та створення exe-файлу;

Debug – для налагодження програми;

Options – для конфігурування середовища;

Window – для конфігурування вікон і роботи з ними;

Help – для надання допомоги.

Потрібний пункт вибирають стрілками переміщення курсора або мишкою і натискають на клавішу вводу. Можна скористатися комбінацією клавіш Alt+<висвітлена буква>.

Розглянемо **основні етапи,** з яких складається сеанс роботи.

1. У пункті меню File обирають команду **New**, після чого середовище переходить у режим створення нового файлу з назвою NONAME00.PAS. У цьому режимі можна набирати текст програми.

2. Для полегшення уведення та виправлення очевидних помилок користуються традиційними прийомами редагування тексту; зокрема, такими комбінаціями клавіш для роботи з блоками (фрагментами) тексту:

Shift+стрілки – виокремити чи зняти виокремлення блоку тексту;

Ctrl+Insert – копіювати блок у буфер обміну;

Sbift+Del – перемістити блок з тексту у буфер;

Shift+Insert – вставити текст з буфера у місце, позначене курсором;

Ctrl+Del – вилучити виокремлений блок з тексту;

Ctrl+Y – вилучити рядок, де є курсор;

Ctrl+Q, Y – вилучити текст від курсора до кінця рядка;

Ctrl+N – вставити рядок;

Ctrl+PgUp – перейти до початку тексту;

Ctrl+PgDn – перейти до кінця тексту.

3. Якщо очевидних помилок немає, програму можна компілювати і виконати (пункт меню Run або за допомогою комбінації клавіш Ctrl+F9).

4. За наявності помилок у програмі система виведе відповідне повідомлення. Середовище перебуватиме у режимі редагування; курсор перебуватиме у рядку, де допущено помилку, або безпосередньо вказуватиме на позицію з помилкою. У верхньому рядку виводиться повідомлення про зміст помилки, що суттєво полегшує її виправлення.

5. Якщо синтаксичних помилок немає, програма буде виконана. Результати можна побачити у вікні результатів. Натиснувши після перегляду результатів на будь-яку клавішу, можна перейти у режим

редагування програми.

6. Щоб для заданого pas-файлу створити exe-файл потрібно виконати команду **Compile=>Destination=>Bisk**. Натискають на AU+F9 і в поточний каталог на диску буде записано exe-файл, який можна виконувати поза середовищем.

7. Щоб зберегти текст програми у файлі з розширенням pas, потрібно активізувати **File=>Save as** або **File=>Save** для зберігання файлу з поточним ім’ям.

8. Для закінчення сеансу роботи і виходу з середовища потрібно виконати **File=>Exit** або натиснути на клавіші Alt+x.

9. Для роботи з програмою, що є на диску, можна використати команду File => Open або натиснути на клавішу F3. Отримаємо діалогове вікно, в якому слід вибрати потрібний файл, після чого текст відповідної програми буде занесено у вікно редагування.

10. Вікон з програмами може бути декілька. Переходити від одної програми до іншої можна за допомогою клавіші F6. Щоб розкрити на весь екран чи згорнути вікно, використовують клавішу F5. Закрити активне вікно можна натиснувши на Alt+F3 або кліком мишки на значку прямокутника у рамці вікна.

11. Якщо потрібна додаткова інформація, можна за допомогою клавіши F1 отримати інформаційно-довідкові тексти про середовище і синтаксичні конструкції мови Pascal.

Коди типових помилок

Turbo Pascal 7.0 генерує два типи повідомлень про помилки: помилки компіляції і помилки виконання. ***Коди помилок компіляції,*** які найчастіше зустрічаються у процесі налагодження програми, такі:

- Unknown identifier – невідомий ідентифікатор;
- Duplicate identifier – повторення ідентифікатора;
- Syntax error – синтаксична помилка;
- Invalid file name – недопустиме ім'я файлу або вказано неіснуючий шлях;
- Type mismatch – невідповідність типів;
- Error in expression – помилка у виразі.

- Division by zero – ділення на нуль;
- Cannot Read or Write variables of this type – немає можливості зчитати або записати змінні цього типу.

Повідомлення про помилки виконання. Помилки виконання поділяються: на помилки вводу-виводу; критичні помилки; фатальні помилки. Ось деякі з них:

- Path not found – шлях не знайдено;
- File not open – файл не відкритий;
- File not open for input/output – файл не відкритий для введення/виведення;
- Unknown command – невідома команда;
- Arithmetic overflow error – помилка під час виконання математичної операції.

Таблиця кодів ASCII (American Standard Code for Information Interchange) складається з двох частин: базової з кодами від 0 до 127 і розширеної з кодами від 128 до 255. Коди від 0 до 32 зарезервовані для системних символів, коди від 128 до 175 і від 224 до 255 – для символів національних алфавітів, коди від 176 до 223 – для символів псевдографіки, за допомогою яких можна створити нескладні графічні зображення у текстовому режимі екрана. Перша чи перші дві цифри коду позначають номер рядка, а остання – номер стовпця, на перетині яких розташований відповідний символ. Наприклад, число 75 є кодом великої латинської літери К, а кириличній літері Л відповідає код 139.

Структура програми на мові Pascal

Програма – це послідовність команд, за допомогою яких записують алгоритм розв'язування задачі. На мові Pascal програма на складається з "заголовку", розділів описової частини та виконувальної частини. У "заголовку" програмі надається ім'я і перелічуються її параметри (за необхідності). Ім'я програмі надає користувач (програміст). Великі і малі букви в іменах програм рівноправні (імена MyName та myname позначають один і той самий об'єкт).

Описова частина може містити такі розділи:

Uses – приєднання бібліотек та модулів;

label – оголошення міток (позначок);

const – оголошення сталих;

type – опис типів;

var – оголошення змінних;

procedure – оголошення процедур користувача;

function – оголошення функцій

Після розділу описів йде розділ операторів (виконувальна частина), що починається із службового слова Begin і закінчується службовим словом End, після якого ставиться крапка. У цьому розділі задаються всі дії, які потрібно виконати для отримання результату. Розділювачем між конструкціями (командами) програми є символ ";" .

Отже, загальна структура програми така:

```
program <ім'я програми>;
   <розділи описової чатини>
 begin
   <розділ команд>
 end.
```

Заголовок та усі розділи, окрім останнього, є необов'язковими. У кінці програми завжди має стояти крапка.

Програма може містити коментарі – фрагменти тексту програми, взяті у фігурні дужки, які використовуються для пояснення роботи програми і не впливають на виконання команд.

Команди введення та виведення даних

Для введення і виведення даних у мові Pascal використовуються стандартні процедури введення і виведення Read і Write, що оперують стандартними послідовними файлами INPUT і OUTPUT.

Для введення даних використовуються оператори процедур введення:

```
read(<змінна 1>,...,<змінна n>)
readln(<змінна1>...,<змінна n>);
readln;
```

Значення початкових даних можуть відділятися один від одного пропусками і натисненням клавіш табуляції та Enter Команду readln без параметрів часто використовують для того, щоб затримати результати виконання програми на екрані. Щоб після цього перейти у режим редагування програми, потрібно натиснути на клавішу вводу.

Для виведення на екран повідомлень та результатів обчислень використовують команди:

```
write(<вираз1>,< вираз 2>,...,<вираз n>);
writeln(<вираз1> <вираз n>);
writeln;
```

У списку виведення можуть бути сталі, змінні або вирази. Змінні, що складають список виводу, можуть відноситися до цілого, дійсного, символьного або булевого типів. Як елемент списку виводу окрім імен змінних можуть використовуватися вирази і рядки. Команду writeln без параметрів використовують для того, щоб перейти на новий рядок під час виконання програми на екрані.

ХІД РОБОТИ

1. Розглянути основні команди головного меню середовища Turbo Pascal (File (файл), Edit (редагування), Search (пошук), Run (виконання), Compile (компіляція), Debug (налагодження), Tools (інструменти), Options (опції), Window (вікна), Help (допомога)).
2. Набрати текст програми для обчислення виразу $y=2^2*((a+b)/2)$:

```
program pr_1;
 uses crt;
 var a,b:integer;
     d,y:real;
begin clrscr;
  writeln(Введіть значення a,b');
  readln(a,b);
  c:=a+b
  d:=c/2;
  y =sqr(2)*d;
  writeln(y:5:2);
  readln
end
```

3. Запустити програму на компіляцію. Ідентифікувати та виправити помилки. Занотувати у звіт коди помилок.
4. Запустити програму на виконання. Занотувати у звіт результат роботи програми для різних значень введених величин a та b.
5. Зберегти програму під назвою «Pryklad1», скориставшись командою Save all меню File.
6. Набрати та проаналізувати текст програми 2, що дає змогу визначити час падіння каменя на поверхню Землі з висоти h.

```
program pr_2;
 uses crt;
 var g,t,h:real;
begin
 clrscr;
  g:=9.8;
  writeln('Введіть значення h');
  readln(h);
  t:=sqrt(2*h/g);
  writeln('час падіння t=',t);
  readln;
end.
```

7. Обчислити час падіння каменя для трьох довільних значень *h*. Змінити програму так, щоб значення прискорення вільного падіння *g* вводилося користувачем із клавіатури.
8. Зберегти текс програми власній папці у вигляді pas-файлу з назвою «Pryklad2». Створити для цього pas-файлу exe-файл.
9. Оформити звіт до лабораторної роботи.

ЛАБОРАТОРНА РОБОТА № 2

Основні типи даних та операції мови Turbo Pascal. Лінійні програми. Обчислення арифметичних виразів

МЕТА РОБОТИ: засвоєння принципів програмування лінійного обчислювального процесу, побудованого на використанні арифметичних виразів і операторів присвоєння.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Стандартні типи даних

Одним із найважливіших понять у програмуванні є **змінна** – поіменована ділянка оперативної пам'яті комп'ютера, де зберігається значення деякої величини. Змінна має назву (ім'я), значення, тип, які задаються користувачем. Змінні можуть бути описані у розділі опису змінних **Var** з відповідним атрибутом.

Тип змінної визначає її допустимі значення та операції, які можна над нею виконувати. До основних типів даних належать: числові цілі і дійсні (табл.1), символьний та логічний.

Цілочислені типи	
byte	0 … 255
shortint	-128 .. 127
word	0 .. 65535
integer	-32768 .. 32767
Дійсний типи	
real	2.9 10-39 – 1.7 10 38
single	1.5 10-45 – 3.4 10 38

Дійсні числа можна записувати у форматі з фіксованою крапкою, (наприклад, 5.04, –12.109), або у форматі з плаваючою крапкою (наприклад, 16.1E–3 (це 0.0161)).

До дійсних значень застосовні операції round і trunc. Перша породжує ціле значення, найближче до операнда (наприклад, round(4.12)=4), а друга – значення цілої частини аргумента (наприклад, trunc(3.62)=3).

У системі Turbo Pascal означено нульмісну функцію Pi (її значенням є число, близьке до числа π) й одномісні функції Frac і Int, застосовні лише до дійсних типів. Вони задають обчислення дробової частини й дійсного подання цілої частини свого аргументу. Наприклад, sin(pi/2)=1.0, frac(3.1415)=0.1415, int(3.1415)=3.0.

Символьний тип (char) – це один з базових типів мови, призначений для збереження та опрацювання одного символу. Множиною його значень є окремі символи (літери, цифри, знаки), впорядковані відповідно із розширеним набором символів ASCII-коду.

Значення символьних змінних та констант задаються у лапках, наприклад:

`Const Name1 = 'v';` – опис символьної константи,

`Name2:= 'n'` – надання значення символьній змінній.

Значення також можна задати, вказавши безпосередньо числовий ASCII-код, поставивши перед ним знак # (наприклад, #35, #102).

Для роботи з символьною інформацією у мові Pascal реалізовані функції перетворення: chr(N) – символ з кодом N, ord(S) – код символа S. Також застосовуються функції, що визначають наступний символ – succ(S), попередній символ – pred(S). Для цих функцій виконуються такі залежності:

```
succ(S) = chr(ord(S) +1);
pred (S) = chr(ord(S) -1).
```

Логічний тип (boolean) характеризується двома значеннями: false (хибність) та true (істинність). Прийнято, що в машинному коді true=1, false=0. В таблиці 2 наведено основні операції, що можна застосовувати до змінних логічного типу.

Таблиця 2. Логічні операції

Операнди		**Результати операцій**			
A	B	NOT A заперечення, інверсія	A OR B або, логічне додавання, диз'юнкція	A XOR B виключаюче або, сума за модулем	A AND B і, логічне множення, кон'юнкція
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
FALSE	TRUE		TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
TRUE	TRUE		TRUE	FALSE	TRUE

Крім булевих операцій, визначено операцію "порядковий номер" ord: ord(false)=0, ord(true)=1.

Нестандартні прості типи

Користувач може визначити свої типи даних у розділі type:

```
type
   <ім'я типу 1>=<опис типу 1>;
. . .
   <ім'я типу n>=<опис типу n>;
```

До простих нестандартних типів належать перелічуваний, діапазонний та рядковий типи. *Перелічуваний* тип утворюють з ідентифікаторів (імен користувача) шляхом їхнього об'єднання у список, який записують у круглих дужках:

```
type <ім'я типу>=(<значення 1>,<значення 2>, … <значення n>);
```

Наприклад, опишемо два перелічувані типи:

```
type  week = (mon, tue, wed, the, fri, sat, sun); - дні тижня;
      colors = (red, green, yellow, white); - кольори;
```

та оголосимо дві змінні − day та mycolor цих типів:

```
var day: week; mycolor: colors;
```
Цим змінним можна надати, наприклад, такі значення:
```
day:= fri, mycolor:= green, mycolor:= yellow.
```
Даними перелічуваного типу не можуть бути числові чи символьні значення. При роботі з даними цього типу можна застосовувати стандартні функції Succ, Pred, Ord. Succ (аргумент) повертає значення, наступне за елементом, вказаним як аргумент. Якщо наступного елемента немає (закінчилися), то це приводить до програмного переривання (помилки). Pred (аргумент) повертає попереднє значення з переліку констант, з яких складається цей тип. Ord (аргумент) повертає порядковий номер аргумента у списку констант, перелічних в описі у вигляді цілого числа. Елементи пронумеровані в порядку 0,1,2..., тобто перший елемент має номер 0.

Діапазонний тип – це звуження деякого базового упорядкованого типу. Його описують так:

```
type<ім'я типу>=<значення 1> .. <значення 2>;
```

Діапазонний тип задається діапазоном зміни констант усередині якого-небудь вже наявного перелічуваного або стандартного типу. Значення першої константи має бути менше значення другої константи діапазону.

```
Typ
   Dni = 1...31;
   Litera = 'a'...,'z',
Var
Rabdni,licdni, vuhod: dni;  {можуть набувати значень 1-31}
im,ident: litera ;          {можуть набувати значень 'A'-'Z'}
```

Вихід за межі діапазону викликає програмне переривання.

Рядковий тип даних (string). Значенням змінної рядкового типу може бути довільна послідовність, яка складається не більше, ніж з 255 символів. Змінні рядкового типу можна описати у розділі опису констант, типів або оголосити у розділі змінних. Наприклад,

```
const s = 'Bye!';
type t = string[10]; {у квадратних дужках – довжина рядка}
var top: t; w: string;
```

Арифметичні операції. Стандартні функції та процедури мови Паскаль

Арифметичні операції: + додавання, – віднімання, * множення; / – ділення. Арифметичні "і" – and і "або" – or проводять порозрядне булеве множення або складання цілих значень операндів.

Для правильного запису арифметичних виразів необхідно дотримуватися наступних правил:

1. Заборонена послідовна поява знаків двох операцій, тобто A + –B – невірно, A + (–B) – вірно.
2. Порядок виконання операцій: *, /, div, mod, and, or, shl, shr, – +. Порядок може бути порушений круглими дужками (частина виразу, взята в круглі дужки виконується насамперед).
3. Операції / і * мають однаковий пріоритет, наприклад, 18/2*3=27, а не 3.
4. Піднесення до степеня виконується за формулою $a^r = \exp(r*\ln(a))$ для додатніх *a* або організовується послідовне множення на *a* r разів. Для відємних *a* можливий тільки другий спосіб.

Основні стандартні функції та деякі процедури мови Паскаль описані в таблицях 3 та 4.

Таблица 3. Стандартні функції

Призначення функції	**Функція**	**Приклад**	
		Операція	**Результат**
Аргументи і результат типу REAL			
Піднесення до квадрату	sqr(R)	sqr(1.5)	2.25
Абсолютна величина	abs(R)	abs(−5.5)	5.5
Корінь квадратний	sqrt(R)	sqrt(2.25)	1.5
Синус	sin(R)	sin(3.1415926536)	0.0
Косинус	cos(R)	cos(3.1415926536)	1.0
Арктангенс	arctan(R)	arctan(1.0)	0.78539816340
Логарифм натуральний	ln(R)	ln(2.7182818285)	1.0
Експонента	exp(R)	exp(1.0)	2.7182818285
Пі	PI	PI	3.1415926536
Псевдовипадкове в інтервалі від 0 до n	random(n)	random(10)	Будь-яке число в інтервалі від 0 до 10
Дробова частина	frac(R)	frac(1.5)	0.5
Ціла частина	int(R)	int(1.5)	1.0
Аргументи типу REAL і результати INTEGER			
Ціла частина	trunc(R)	trunC(3.6)	3
Округлення до цілого	round(R)	round(3.6)	4
Залишок від ділення	R mod K	11mod5	1
Цілочисельне ділення	R div K	11div5	2

Аргументи типу CHAR			
Заміна малої літери латинської абетки на велику	upcase(x)	upcase(a)	A
ASCII –код символа	ord(x)	ord(K)	75

Таблиця 4. Стандартні процедури

Призначення процедури	**Процедура**	**Приклад**	
		Операція	Результат
Збільшує x на y	inc(x,y)	inc(1,9)	10
Збільшує x на 1	inc(x)	inc(9)	10
Зменшує x на y	dec(x,y)	dec(9,6)	4
Зменшує x на 1	dec(x)	dec(9)	8

В якості прикладу розглянемо програму для обчислення значення виразу $Y = 1 - \frac{1}{1 + (\frac{R * a}{k})^2} \Big/ \frac{e^2}{S^2}(1 + \frac{R * a}{L})^2$, де величини *R, a, S, L, K* вводяться з клавіатури.

```
Program Lin_pr1;
Var
   к,L:integer;
   R,a,S:real;
Begin
 writeln('введіть цілі к,L ');
 readln(к,L );
 writeln('введіть дійсні r,S,a ');
 readln(R,S,a );
 Y =',1-l/(l+sqr (r(a/k)) /exp(2) /sqr(s)(sqr(l+r (a/l);
 writeln('результат Y =', Y);
 readln;
end.
```

У результаті роботи програми спочатку буде виведено текст "результат Y=", а потім − значення виразу Y.

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. У кожній програмі треба передбачити команду виведення на

екран свого прізвища під результатами. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.

3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає в підборі таких значень аргумента, які давали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до цієї роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

1. Написати програму для визначення числа, отриманого виписуванням у зворотному порядку введеного користувачем деякого тризначного числа.
2. Подана сторона рівностороннього трикутника (вводиться користувачем). Потрібно програмно знайти площу цього трикутника.
3. З клавіатури вводиться сума нарахованої заробітної плати. З цієї суми необхідно утримати 12 % на прибутковий податок, 1 % на профспілковий податок, 1 % на пенсійний податок і додати 45 %. Отриману суму до видачі вивести на екран.
4. Написати програму для обчислення дробової частини середнього арифметичного трьох заданих додатніх чисел.
5. Відомо, що оплата за дитячий садок в місяць складає 150 грн (місяць – 22 дні). Розрахувати, скільки потрібно заплатити за місяць, якщо дитина була присутня *N* днів (кількість днів вводиться з клавіатури).
6. Написати програму, що обчислює добуток цифр введеного користувачем двозначного числа.
7. Обчислити периметр і площу прямокутного трикутника за введеним користувачем довжинами катетів.
8. Нехай *a* – деяке тризначне число, що вводиться з клавіатури. Написати програму для знаходження суми значень цифр даного числа *a* (наприклад, при *a*=123 це 6).

9. Ділянка лісу має форму рівнобічної трапеції. Обчислити програмно її периметр і площу за введеними користувачем довжинами сторін.
10. Ресторан закуповує щодня масло m_1 кг за 12.50 грн за кілограм, сметану m_2 кг за 8.40 грн, вершки m_3 кг за 9.10 грн. Визначити суми, потрібні для купівлі окремих продуктів, і загальну суму (значення m вводяться з клавіатури).
11. Визначити програмно і вивести на екран суму цифр введеного з клавіатури цілого двозначного числа a (наприклад, при a=83 це 11).
12. Тіло падає з прискоренням g. Визначити пройдений тілом шлях $h=gt^2/2$ після n-ої секунди падіння.
13. Визначити програмно і вивести на екран число одиниць у введеному користувачем натуральному числі n (n>9).
14. Телефонні розмови з трьома населеними пунктами коштують c_1, c_2, c_3 коп/хв. Розмови тривали t_1, t_2, t_3 хв відповідно. Яку суму нарахує комп'ютер до оплати за кожну окрему і всі розмови разом?
15. Написати програму для обчислення дробової частини середнього арифметичного трьох введених користувачем додатніх чисел.

Задача 2

Скласти програму для обчислення виразу вашого варіанту (величини a, b, c, d вводяться з клавіатури).

1	$10.3e^{-b}-\frac{2c-4\sin 3a}{3(b+5.7)}$	9	$1.3e^{2a}-\frac{tg3d+(2c+5)}{(b+a)(2b-1)}$
2	$\frac{(2c-5)\cos 2b}{(a+2)(b+1.7)}+6.3d$	10	$7.3d-\frac{(5c-b)\sin a}{(a-d)(b+5.7)}$
3	$\lvert\sin a\rvert-\frac{2c-4e^{-b}}{3(\cos d+5.7)}$	11	$5e^{-2c}-\frac{tgc+\sin a}{3(d+5.7b)}$
4	$10.3e^{-b}(a+d)-\frac{\sqrt{2c}-4c^2}{(b+3.7)}$	12	$\sqrt{10.3a^2}-\frac{c-4\sin 3a}{(d-5.7)e^{-2b}}$
5	$\frac{2c-4e^{-d}}{3(b+7)}+1.3\sin 3a$	13	$e^{-b}-\frac{c-4\cos 3a}{5(\sqrt{b+5.7}+c^2)}$
6	$\frac{2c-4\sqrt{2/\cos b}}{3}+\frac{2c}{(d+5.7)}e^{-b}$	14	$\sqrt{2c}\,tg2b-\frac{\sin a}{3(b+7)}$
7	$\cos a-\sin^2 b-\frac{\sqrt{b+5}}{3b+2d}$	15	$-\frac{2c-4\sin a}{3(b+a)}+3e^{-3a/b}$
8	$\sqrt{2c-4\sin 3a}+10e^{-b}\cos 2d$		

ЛАБОРАТОРНА РОБОТА № 3

Програмування алгоритмів розгалуженої структури

МЕТА РОБОТИ: вироблення практичних навичків використання умовного оператору та оператору вибору при розв'язуванні задач.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Умовній оператор IF

Для визначення дій, які виконуються тільки в разі виконання деякої умови, використовуються такі конструкції:

- `IF <умова> then <оператор> else <оператор>;` – повне розгалуження; використовується тоді, коли визначені різні дії в разі виконання та невиконання умови.
- `IF <умова> then <оператор>;` – неповне розгалуження; використовується тоді, коли визначені дії тільки у разі виконання деякої умови.

Нехай, наприклад, потрібно програмно запитати в користувача два різних числа. Далі перевірити, якщо 1-е число більше другого, вивести: "перше більше, ніж друге"; якщо 2-ге більше 1-го, то вивести: "друге більше, ніж перше". Для перевірки яке з введених чисел більше можна використати конструкцію If...Then...Else

```
If  A > B Then  Write (A, 'більш  ніж ',B, '!')
else  Write (B, 'більше ніж ',A,  '!');
```

Спочатку йде службове слово If; після нього стоїть перша логічна умова, в якій використовуються знаки порівняння. Якщо ця умова вірна, тобто A>B, то виконується дія, вказана після слова Then (перед словом Else немає крапки з комою). Якщо умова не вірна (A<B), то виконується дія, вказана після слова Else. Після цієї дії вже ставитися крапка з комою.

Розглянемо задачу про обчислення дійсних коренів квадратного рівняння $ax2+bx+c=0$ (за умови a відмінного від 0). Алгоритм виконання цього завдання є таким:

1) зчитати коефіцієнти *a, b, c*;

2) обчислити дискримінант $d=b*b-4*a*c$;

3) – якщо d>0, то обчислити $x_1=(-b-\text{sqrt}(d))/(2*a)$, $x_2=(-b+\text{sqrt}(d))/(2*a)$;

– якщо d=0, то обчислити x_1=*-b/(2*a)*

– інакше повідомити користувача, що рівняння не має корнів.

Задається три різні послідовності дій. Яка саме виконується, залежить від конкретних значень *a, b, c*.

У вигляді програми на мові Pascal цей алгоритм реалізується так:

```
program Rozg1;
var  a, b, c, x1, x2: real;
begin
  write('Введіть коефіцієнти квадратного рівняння a,b,c ');
  readln (a,b,c);    {припускаємо, що a<>0}
  d:=b*b-4*a*c;
  if d>0 then
  begin
   x1:=(-b+sqrt(d))/(2*a);
   x2:=(-b-sqrt(d))/(2*a);
  end
  else
  if d=0 then x1:=-b/(2*a );
    else writeln('рівняння коренів не має');
  readln;
end.
```

Оператори **if** можуть бути вкладені один в одного необмежену кількість разів; причому новий **if** може починатися як після слова **then**, так і після слова **else**. За допомогою логічних операції Or та And можна побудувати умови будь-якої складності. Наприклад, нехай потрібно знайти значення функції:

$$Z=\begin{cases}\dfrac{ax^2}{\sin x},\ якщо\ \ a>0\ \ та\ \ 0>x<0.5;\\ e^x,\ якщо\ \ a>0\ \ та\ \ 0\le 0.5x<3;\\ \dfrac{\sqrt{a}\quad\sqrt{x}}{\sin x},\ якщо\ \ a\le 0\ \ або\ \ x<3;\\ 0\ -\ в\ інших\ випадках.\end{cases}$$

Програма для реалізації цього завдання може бути такою:

```
Program  Rozg2;
Var  a,x:real;
begin
 writeln('введіть a, x');
 readln(a,x );
  if (a>0) and (x>0) and (x<=0.5) then
   writeln('z=',a*sqr(x)/sin(x)) else
    if (a>0) and (x>=0.5) and (x<3) then
     writeln('z=',exp(x)) else
       if (a<=0) or (x<=0) then
```

```
        writeln('z=',sqrt(a)*sqrt(x)/sin(x)) else
         writeln('z=0');
  readln;
end.
```

Оператор вибору варіантів case

Оператор case використовується для вибору одного з декількох варіантів подальшого ходу програми. Вибір необхідної послідовності операторів здійснюється під час виконання програми залежно від рівності значення змінної-селектора константі, що зазначена перед групою інструкцій. Загальний вигляд:

```
Case <вираз-селектор> of
 <список констант вибору 1>: <оператор1>;
 . . .
 <список констант вибору n>: <оператор n>;
Else <оператор> end;
```

Вираз-селектор може бути байтового, цілого, логічного, символьного, перераховуваного і інтервального типів. Список констант вибору складається з переліку констант, розділених комами, або з констант, заданих інтервалом, або з комбінацій переліку констант і інтервалів. Тип констант в списках Case повинен відповідати типу виразу-селектору. Значення констант у списках не повинні повторюватися.

Робота оператора: спочатку обчислюється вираз-селектор; потім отримане значення порівнюється з константами вибору, і, якщо значення виразу-селектора збігається з якою-небудь константою вибору, виконується оператор, що стоїть після списку констант, в якому є константа, співпадаюча із значенням виразу-селектору. Якщо значення виразу-селектору не збігається ні з однією з констант вибору, то виконується оператор, наступний за else. Else може в Case бути відсутнім – тоді у разі неспівпадіння констант оператор Case ігнорується.

Розглянемо деякі приклади використання селекторів:

1. *Селектор цілого типу, список констант представлені переліком констант.*

Нехай потрібно обчислити значення Z за введеним значенням змінної *i:*

$$Z=\begin{cases} i+10, \text{ якщо } i=1,\ 2,\ 5 \\ i+100, \text{ якщо } i=12,\ 16 \\ i+1000, \text{ якщо } i=31,\ 35,\ 46 \end{cases}$$

```
Program  Rozg3;
Var i,z:integer ;
begin
 writeln('введіть ціле i');
 readln (i);
   Case i of
   1,2,5: writeln ('i=',i ',  z=',i+10 );
   12,16: writeln ('i=',i ',  z=',i+100 );
   31,35,46: writeln ('i=',i ',  z=',i+1000 );
   else  writeln (неправильно задане i'); end ;
 readln ;
end .
```

2. *Селектор цілого типу – список констант представлений діапазоном.*

Нехай потрібно визначити, чи належить введене ціле число *i* якомусь з діапазонів [1;10], [11;100]або [101;1000],чи не належить. Програмна реалізація цього завдання може бути такою:

```
Program  Rozg4;
Var i:integer ;
begin
 writeln('введіть ціле i');
 readln(i);
 Case i of
  1 ..10:      writeln('число належиьть діапазону 1-10');
  11 ..100:    writeln('число належиьть діапазону 11-100');
  101 ..1000:  writeln('число належиьть діапазону 101-1000');
   else writeln('число є поза діапазоном 1-1000');
 end ;
 readln ;
end.
```

3. *Селектор знакового типу.*

Нехай потрібно ввести деякий шифр факультету і отримати інформацію, якому саме факультету він відповідає.

```
Program  Rozg5;
Var name:char ;
begin
  writeln('введіть шифр');
  readln(name);
  Case name of
   'P': writeln('інженерно-педагогічний факультет');
   'F': writeln('факультет фізичного виховання');
   'I': writeln('історичний факультет');
   'B': writeln('біологічний факультет');
      else writeln('шифр не відповідає жодному факультету');
  end;
  readln;  end.
```

Оператор безумовного переходу goto

За допомогою процедури Goto здійснюється заданий перехід у яке-небудь місце програми (зазвичай за деякими умовами).

Розглянемо програму, яка демонструє використання оператора goto. Програма зчитує символи з клавіатури, поки не зчитає символ "!" - знак оклику, що служитиме сигналом закінчення введення. Усі зчитані символи записуватимуться у рядок, який в кінці виконання програми виводиться на екран. Спершу запишемо алгоритм програми:

1. Читаємо символ з клавіатури;
2. Перевіряємо – якщо це не "!", то:
3. Додаємо введений символ в рядок;
4. Переходимо до першого пункту.
5. Інакше (символ = !): виводимо складений рядок на екран;
6. Завершуємо програму;

Для реалізації четвертого пункту використаємо процедуру Goto. Текст програми матиме вигляд:

```
Program Rozg6;
label  A;
var  C: Char;      {для символу, що зчитується з клавіатури}
     S: String;    {для запису всіх символів, що вводяться}
begin
  A:
  Write('Введіть символ: ');
  Readln(C);
   If C<>'!' Then
    begin
     S:=S+C;
     Goto A;
    end
    else  Write('Ви ввели наступні символи:',S);
  Readln;
end.
```

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених

програм. У кожній програмі треба передбачити команду виведення на екран свого прізвища під результатами. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.

3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які давали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до лабораторної роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

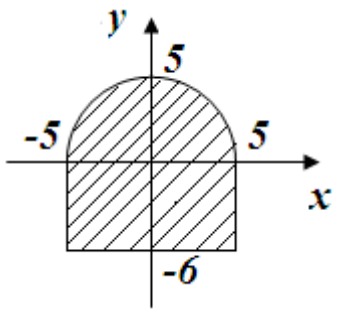
ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ №4

Задача 1

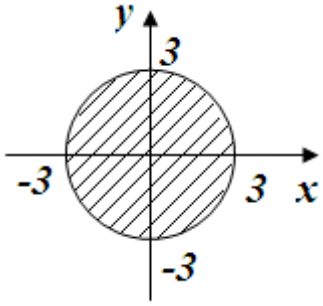
Скласти програму для реалізації таких завдань згідно з Вашим варіантом, використовуючи умовний оператор.

1. Ввести з клавіатури два числа; потім від 1-го відняти 2-е. Якщо результат від'ємний, вивести на екран відповідне повідомлення. У протилежному випадку на екран виводиться отриманий результат.
2. Ввести з клавіатури три числа; перевірити, чи сума перших двох більше за третє. Результати перевірки вивести на екран.
3. Написати програму для обчислення за цілим $N>3$ таких натуральних A і B, що $5A+2B=N$, причому $A+B$ мінімально.
4. За значеннями введених коефіцієнтів *a, b, c* визначити кількість коренів квадратного рівняння $ax^2+bx+c=0$ (брати до уваги випадок, коли $a=0$).
5. За значеннями введених коефіцієнтів *a, b, c* дослідити вигляд множини розв'язків нерівності $ax^2+bx+c>0$ (два інтервали, інтервал і т.і.).
6. За значеннями введених користувачем довжин трьох сторін трикутника визначити тип трикутника: рівносторонній, рівнобедрений, різнобічний.

7. Написати програму для перевірки, чи належить число, введене з клавіатури, інтервалу (–5;8).
8. За значеннями введених коефіцієнтів *a, b, c* обчислити та вивести та екран дійсні корені квадратного рівняння $ax^2+bx+c=0$ (врахувати випадок, коли $a=0$).
9. За введеним користувачем значенням вартості покупки, вивести сумму оплати з урахуванням знижки: знижка 5% надається, якщо сума покупки більше 700 грн, у 5% – якщо сума більше 1000 грн.
10. Ввести з клавіатури двозначне число. Визначити, яка з його цифр більше. Резуьтати перевірки вивести на екран.
11. За введеними координатами (x, y) точки визначити, чи належить вона заштрихованій області на малюнку:



12. Написати програму для перевірки, чи виконується нерівність $a<b<c$ для трьох дійсних чисел *a,b,c*, що вводяться з клавіатури.
13. Ввести з клавіатури декілька чисел. Знайти та вивести на екран суму тих чисел, які більше за пять.
14. Зясувати, чи є два дійсні числа A і B, введені користувачем, парними. Результати перевірки вивести на екран.
15. За введеними координатами (x, y) точки визначити, чи належить вона заштрихованій області на малюнку:



Задача 2

Скласти програму для розв'язання наведеного завдання, використовуючи команду case. Задати вхідні дані так, щоб вибір був з чотирьох-п’яти альтернатив. Використати, де треба, змінні типу integer для чисел, char для літер і string для прізвищ, назв тощо.

1. Ввести номер студента зі списку. Вивести його прізвище.
2. Ввести номер поїзда. Вивести назву пункту призначення.
3. Ввести номер дня тижня. Вивести його назву.
4. Ввести першу букву назви країни. Вивести назву континента.
5. Ввести номер місяця. Вивести назву пори року.
6. Ввести номер студента у списку. Вивести його ім'я.
10. Ввести номер місяця. Вивести назву місяця і номер кварталу.
11. Ввести першу букву назви країни. Вивести кількість населення і кількість міст цієї країни.
12. Ввести числовий код групи. Вивести повну назву групи і кількість студентів у ній.
13. Ввести число з діапазону 0..5. Вивести його написання двома мовами.
14. Ввести номер дня тижня. Вивести його назву і кількість пар.
15. Увести число з діапазону 5..9. Вивести його значення трьома мовами.

Задача 3.

Скласти програму для обчислення значення функції для Вашого варіанту (змінна x вводитьсяз клавіатури).

1. $y = 1 + 9x + \begin{cases} \ln\left|\sin x\right| + x^7 & , x \le 0 \\ \operatorname{ctg}\dfrac{\left|x+1\right|}{2} & , 0 < x \le 3 \\ 3x - x^5 & , x > 3 \end{cases}$

2. $y = \dfrac{1}{x} + 4 - \begin{cases} 0{,}65x + 8 & , x < 1 \\ \operatorname{arctg}\dfrac{x+6{,}1}{2} + e^x & , 1 \le x < 5 \\ \sqrt{1+\sqrt{x}} & , x \ge 5 \end{cases}$

3. $y = \dfrac{2}{x} + \left|x\right| + \begin{cases} 1 + 4x^2 & , x < 0 \\ \left(e^x + \left|x\right|\right)^2 & , 0 \le x \le 2 \\ 5\sin\left(x^2 + 1\right) & , x > 2 \end{cases}$

4. $y = 1 + x + \begin{cases} e^{\ln(2+2x)+2x} & , x \le 4 \\ \operatorname{ctg}\dfrac{1+x}{9} + 8x & , 4 < x \le 7 \\ 1 - 7x + x^2 - 2x^3 & , x > 7 \end{cases}$

5. $y=\dfrac{1}{|x+2|}+1-\begin{cases}7x^2+x-8 & ,\ x<1\\ \operatorname{ctg}\dfrac{x+4}{\sqrt{11}}+3 & ,\ 1\le x\le 4\\ \sqrt{1+|\cos^3 x|} & ,\ x>4\end{cases}$

6. $y=5e^{3x}-\begin{cases}3+\sin|x| & ,\ x\le -1\\ 2e^{\frac{x}{4}-1} & ,\ -1<x\le 3\\ 7-\sqrt{2}x^3 & ,\ x>3\end{cases}$

7. $y=x^2\sin\dfrac{x}{2}+\begin{cases}\operatorname{arctg}e^x & ,\ x\le -5\\ 1+\dfrac{x^3}{4} & ,\ -5<x\le 0\\ \ln|x|-\dfrac{x}{5} & ,\ x>0\end{cases}$

8. $y=2+6x+\begin{cases}\ln\cos x+x^5 & ,\ x\le 0\\ \operatorname{ctg}\dfrac{1+\ln x}{3} & ,\ 0<x\le 3\\ 12x-x^8 & ,\ x>3\end{cases}$

9. $y=2|x|^3-\begin{cases}5\cos 18x & ,\ x\le -0{,}1\\ \operatorname{arctg}\dfrac{x+2}{5} & ,\ -0{,}1<x<1{,}2\\ \operatorname{ctg}x+18 & ,\ x\ge 1{,}2\end{cases}$

10. $y=4{,}95x^2+\begin{cases}4+x^{-2} & ,\ x<-3{,}5\\ \operatorname{tg}\dfrac{3{,}5+x}{5} & ,\ -3{,}5\le x<1\\ \sin 3x-\cos x & ,\ x\ge 1\end{cases}$

11. $y=2|5-x|-\begin{cases}e^{|2+x|} & ,\ x\le -1\\ \sin^2\dfrac{1}{|2+x|} & ,\ -1<x<1\\ \dfrac{\cos^2 x}{1+|\sin x|} & ,\ x\ge 1\end{cases}$

12. $y=\dfrac{2+x}{x^2}+1+\begin{cases}x^3-2x^4 & ,\ x<0\\ \left(|x|+e^x\right)^3 & ,\ 0\le x\le 2\\ 4\cos\left(x^2-2\right) & ,\ x>2\end{cases}$

13. $y=2|x-5|-\begin{cases}\dfrac{\sin^2 x}{1+|\cos x|} & ,\ x<-1\\ \cos^2\dfrac{1}{|x+2|} & ,\ -1\le x\le 1\\ \ln|x+2| & ,\ x>1\end{cases}$

14. $y=|4x-1|+\begin{cases}x^7-2x & ,\ x<0\\ \operatorname{arctg}\dfrac{e^x+1}{8} & ,\ 0\le x<3\\ x^4+e^{x^2+3} & ,\ x\ge 3\end{cases}$

15. $y=x^3+2+\begin{cases}5x^8+x^6-x^2+3 & ,\ x<4\\ \operatorname{arctg}\left|\dfrac{x+3}{2}\right|+7x & ,\ 4\le x<7\\ \lg\left(2x+e^{5x+5}\right) & ,\ x\ge 7\end{cases}$

ЛАБОРАТОРНА РОБОТА № 4

Програмування алгоритмів циклічної структури. Табулювання функцій та обчислення сум

МЕТА РОБОТИ: вироблення практичних навичків використання операторів циклу при розв'язуванні задач.

ТЕОРЕТИЧНІ ВІДОМОСТІ**.**

Циклічні програми – програми, в яких реалізовано команди циклу. У циклічних програмах який-небудь алгоритм повторюється багато разів, при цьому один з параметрів змінюється. У мові Pascal передбачено три різновиди операторів циклу: цикл із передумовою **while**, цикл з післяумовою **repeat**, цикл із лічильником **for.** Також реалізована робота із вкладеними циклами.

1. ***Цикл із передумовою (цикл-«поки»)***

На мові Pascal оператор циклу з передумовою має вигляд:

```
while <умова> do <оператор>;
```

Робота оператора. Спочатку перевіряється умова (це вираз, який після обчислення приймає значення True або False), і, якщо вона вірна, то виконується тіло циклу; у протилежному випадку відбувається вихід з циклу. Тіло циклу – оператор, як правило, складений. Потрібно, щоб булевий вираз в тілі циклу на якомусь етапі став помилковим, інакше цикл ніколи не закінчиться.

Так, наприклад, для обчислення суми перших 100 натуральних чисел методом послідовного додавання можна скористатися циклом з передумовою:

```
m:=1; S: =0;
while m<=100 do
begin
  S:=S+m;
  m:=m+1;
end;
```

Розглянемо ще одне завдання – потрібно протабулювати функцію $y=sin(x)$ на проміжку [0; 3,1], з кроком h =0,1 і обчислити середнє

арифметичне значень функції більших, ніж 0,1 і менших за 0,6. Для реалізації даного завдання також використаємо цикл з передумовою:

```
program Cykl1;
var x, y, s, si, h, xk: real;
    n: integer;
begin
  x:= 0; xk:=3.1; h:=0.1; s:=0; n:=0;
   while x<=xk+h/2 do
   begin
    y:=sin(x);
    writeln(x:3:1, y:6:2);
     if (y >0.1) and (y<0.6) then
      begin
        s:=s+y; n:=n+1;
      end;
      x:=x+h;
    end;
      if n > 0 then
       begin
        si:=s/n; writeln('Середнє значення =', si);
       end
       else writeln('Таких значень немає, n=0');
 readln;
end.
```

2. Цикл із післяумовою (цикл-«до»)

Цикл, у якому тіло циклу виконується доти, поки умова, задана після тіла циклу, не стане правильною. Після того, коли умова стає правильною, робота циклу припиняється й керування передається операторові, наступному за оператором циклу.

На мові Pascal оператор циклу з післяумовою має вигляд:

```
repeat <оператор> until <умова >;
```

Робота оператора: спочатку виконується тіло циклу, потім перевіряється умова (булевий вираз), і, якщо вона помилкова, знову виконується тіло циклу. Вихід з циклу відбувається, коли умова стане істинною. Цикл зручний для організації великих структур даних, оформлення цілих блоків програми.

Прикладом практичного використання цьогоо циклу може виступати обчислення суми перших 100 натуральних чисел методом послідовного додавання:

```
m:= 0; S:= 0;
repeat
m:=m+1;
S:=S+m;
until m >=100;
```

Після слова repeat не ставиться крапка з комою, оскільки repeat є заголовком цілого блоку. Оператори усередині конструкції repeat-until не виділяються додатковими begin-end.

Використаємо цикл з післяумовою для табулювання функції $y=sin(x)$ на проміжку [–3,14; 3,14] з кроком h=0,2 і визначення максимального (max) та мінімального (min) значення функції на цьому проміжку.

```
program Cykl2;
var h, x, y, max, min: real;
begin
  h:=0,2; x:=-pi;
  max:=sin(x); min:=sin(x); {max i min - в першій точці}
    repeat
     y:=sin(x);
     writeln(x:7:2, y:7:2);
     if y>max then max:=y;    {Визначаємо максимум функції}
      if y<min then min:=y;   {Визначаємо мінімум функції}
       x:=x+h;
    until x>pi+h/2;           {умова виходу з циклу}
  writeln('max=', max:5:2, '  min=', min:5:2);
 readln;
end.
```

3. Цикл із лічильником

У мові Pascal реалізовано два оператори циклу з покроковою зміною аргументу: «цикл For-To» і «цикл For-DownTo».

а) «Цикл For-To» – цикл з кроком +1, застосовується, якщо початкове значення менше кінцевого значення. Оператор циклу має вигляд:

```
for <лічильник циклу>:=<початкове значення> to <кінцеве значення> do <оператор>;
```

Щоб продемонструвати, як працює цей цикл, запишемо реалізацію розглянутого раніше завдання – обчислення суми перших 100 натуральних чисел методом послідовного додавання:

```
S: =0;
for  m:=1  to  100 do
S:=S+m ;
```

Цикл починається словом for, після якого змінній привласнюється початкове значення. Далі йде слово to, після якого вказується кінцеве значення змінної. У кінці циклу, після команди do, вказуються всі дії, які мають бути зациклені. Якщо тіло циклу має складатися з декількох операторів, їх потрібно помістити в операторні дужки **begin - end.**

б) «Цикл For-DownTo» – цикл з кроком –1, застосовується, якщо початкове значення більше кінцевого значення. Оператор циклу має вигляд:

```
for <лічильник циклу>:=<початкове значення> downto <кінцеве значення> do <оператор>;
```

Використаємо оператори циклу з покроковою зміною аргументу для реалізацію наступного завдання: вивести на екран через пробіл всі літери латинської абетки спочатку від A до Z, а потім – у зворотному порядку від Z до A. В якості параметра цикла використаємо символьну змінну.

```
Program  Cykl3;
Var
  i:char;
begin
  for i:='a' to 'z' do
    write(' ',i);
    writeln;
  for i:='z' downto 'a' do
    write(' ',i);
 readln;
 end.
```

Під час реалізації циклу з покроковою зміною аргументу в Pascal необхідно заздалегідь знати про кількість повторень тіла циклу і пам’ятати про можливість зміни лічильника циклу тільки на 1 або –1. Початкове і кінцеве значення параметра циклу в загальному випадку є виразами. Тип цих виразів і тип параметра циклу повинні збігатися. У тілі циклу параметр циклу не повинен змінюватися. Також не можна за допомогою оператора переходу goto увійти до тіла циклу, минувши заголовок. Вийти з тіла циклу в програму можна за **if..goto**, не чекаючи повного перебору параметрів циклу.

Вкладені цикли

Оскільки тіло будь-якого циклу складається з операторів або складеного оператора, то в тілі циклу можна розташовувати інші оператори циклу. Вкладати цикли один в одного можна будь-яку кількість разів; необхідно лише пам'ятати, що кількість виконань самого внутрішнього тіла циклу при цьому зростатиме в геометричній прогресії.

Розглянемо наступне завдання: обчислити з точністю до e (e>0) кожну внутрішню суму, визначити кількість доданків, вивести проміжні результати:

$$\sum_{x=1}^{5}\sum_{k=1}^{\infty}(-1)^k x^{k+2}/(k+1)(k+2)$$

Математичне рішення: кожна внутрішня сума є сумою геометричної прогресії, знаменник якої: $q = b_n/b_{n-1} = -x/(k+1)(k+2)$. $B_0 = x^2/2$.

Текст програми для обчислення даних сум може бути таким:

```
program Cykl4;
var
   x,k,m:integer;
   e,s,d,l:real;
begin
  writeln('Введіть потрібну точність:');
  readln(e);
  for x:=1 to 5 do
    begin
      writeln('x=',x);
      k:=1;  l:=sqr(x)/2;
      repeat
      l:=-l*x/((k+1)*(k+2));
      d:=d+l;
      write(l:11:6);
      k:=k+1; m:=m+1;
      until abs(l)<e;
      writeln;
      s:=s+d;
      write('s',x,'=',d:0:7);
      writeln('  n',x,'=',k-1);
    end;
  Writeln;
  writeln('Кількість доданків:',m);
  Writeln('S=',s:0:8);
  readln;
end.
```

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. У кожній програмі треба передбачити команду виведення на екран свого прізвища під результатами. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.
3. Відкомпілювати, налагодити та запустити програми на виконання.

4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які давали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до лабораторної роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

1. Написати програму, яка виводить "стовпчиком" та в рядок всі цілі числа від 20 до 35.
2. Надрукувати "стовпчиком" квадрати всіх цілих чисел від 10 до *b* (значення *b* вводитьсяз клавіатури, причому *b*>10)
3. Скласти програму для знаходження суми квадратів всіх цілих чисел від 10 до *n*(значення *n* вводитьсяз клавіатури).
4. Вартість однієї одиниці деякого товару становить 20 гривень. Надрукувати таблицю вартості від 1 до *n* одиниць цього товару (значення *n* вводитьсяз клавіатури).
5. Знайти добуток всіх цілих чисел від 1 до *a* (значення *a* вводиться з клавіатури; 1<=*a*<=20).
6. Скласти програму, що друкує таблицю значень функції $y=\cos(2x)$ на відрізку [0;1] з заданим кроком *h* (значення *h* вводитьсяз клавіатури).
7. Написати програму, яка зчитує цілі числа з клавіатури і складає їх, доки не буде введено число 0.
8. Написати програму, яка виводить на екран 10 перших ступенів числа *a*. (значення *a* вводитьсяз клавіатури; 1<=*a*<=9).
9. Ввести з клавіатури число. Вивести на екран таблицю множення для нього.
10. Скласти програму, що друкує на екрані всі парні числа в діапазоні від 100 до 299.
11. Скласти програму, що виводить на екрані всі непарні числа в діапазоні від -25 до 25.

12. Написати програму для обчислення найбільшого загального дільника двох натуральних чисел A і C.
13. Організувати цикл, в якому змінна *i* міняється місцями від 3 до 27 з кроком 3.
14. Скласти програму для знаходження добутку всіх непарних чисел в діапазоні від 1 до *n* (значення *n* вводиться з клавіатури).
15. Побудувати таблицю відповідності між дюймами та сантиметрами, якщо 1 дюйм = 2,54 см. В якості початкового значення обрати 1 дюйм, кількість рядків у таблиці вводиться користувачем у режимі діалогу.

Задача 2

1. Протабулювати функцію *y=sin(x)* на проміжку [-1/10; 1/10] з кроком *h*=0,02. Результати обчислень вивести на екран у вигляді таблиці. Обчислити середнє арифметичне значень функції більших, ніж 0,3 і менших за 0,6.
2. Протабулювати функцію $y = x + \sqrt{1 + \sqrt{x}}$ на проміжку [0; 10] з кроком *h*=0,5. Результати обчислень вивести на екран у вигляді таблиці. Знайти мінімальне (min) значення функції на цьому проміжку.
3. Протабулювати функцію $y = \operatorname{ctg}\frac{x+1}{2}$ на проміжку [-0.5; 0.5] з кроком *h*=0,1. Результати обчислень вивести на екран у вигляді таблиці. Знайти максимальне (max) значення функції на цьому проміжку.
4. Протабулювати функцію $y = 5\sin\left(x^2 + 1\right)$ на проміжку [-1/2; 1/2] з кроком *h*=0,2. Результати обчислень вивести на екран у вигляді таблиці. Знайти максимальне (max) та мінімальне (min) значення функції на цьому проміжку.
5. Протабулювати функцію $y = \frac{\cos^2 x}{x}$ на проміжку [0; $\pi/4$] з кроком *h*=0,08. Результати обчислень вивести на екран у вигляді таблиці. Знайти мінімальне (min) значення функції на цьому проміжку.
6. Протабулювати функцію $y = 4\cos\left(x^2 - 2\right)$ на проміжку [-1; 1] з кроком *h*=0,02. Результати обчислень вивести на екран у вигляді таблиці.

Обчислити суму першого та останнього значень функції.

7. Протабулювати функцію $y = 2|x|^3 - \operatorname{ctg} x + 1$ на проміжку [1; 2] з кроком h=0,05. Результати обчислень вивести на екран у вигляді таблиці. Знайти кількість додатніх та від'ємних значень функції на цьому проміжку.
8. Протабулювати функцію $y = 4\sin 3x - \cos x$ на проміжку [-0.9; 0.9] з кроком h=0,2. Результати обчислень вивести на екран у вигляді таблиці. Обчислити кількість усіх додатніх значень функції на цьому проміжку.
9. Протабулювати функцію $y = e^x$ на проміжку [0; 5] з кроком h=0,5. Результати обчислень вивести на екран у вигляді таблиці. Обчислити середнє арифметичне значень функції більших, ніж 4 і менших за 6.
10. Протабулювати функцію $y = x^2 \sin\frac{x}{2}$ на проміжку [-1/5; 1/5] з кроком h=0,04. Результати обчислень вивести на екран у вигляді таблиці. Обчислити кількість та добуток усіх від'ємних значень функції на цьому проміжку.
11. Протабулювати функцію $y = \sqrt{1 + \cos^3 x}$ на проміжку [-1; 1] з кроком h=0,1. Результати обчислень вивести на екран у вигляді таблиці. Знайти мінімальне (min) значення функції на цьому проміжку та визначити значення аргумента, для якого воно досягається.
12. Протабулювати функцію $y = 5e^{3x} - 7 - \sqrt{2}x^3$ на проміжку [1; 3] з кроком h=0,06. Результати обчислень вивести на екран у вигляді таблиці. Знайти максимальне (max) та мінімальне (min) значення функції на цьому проміжку.
13. Протабулювати функцію $y = \cos\left(x + \frac{\pi}{4}\right)$ на проміжку [0; 1] з кроком h=0,1. Результати обчислень вивести на екран у вигляді таблиці. Обчислити добуток усіх додатніх значень функції на цьому проміжку.
14. Протабулювати функцію $y = 5\sin\left(x^2 + \frac{\pi}{4}\right)$ на проміжку [0; π/4] з кроком h=0,05. Результати обчислень вивести на екран у вигляді таблиці.

Обчислити середнє арифметичне значень функції більших, ніж 0.4 і менших за 0.6.

15. Протабулювати функцію $y = x^4 + e^{x^2+3}$ на проміжку [-1; 7] з кроком h=0,5. Результати обчислень вивести на екран у вигляді таблиці. Знайти максимальне (max) та мінімальне (min) значення функції на цьому проміжку.

Задача 3

Обчислити з точністю до e (e>0) кожну внутрішню суму, визначити кількість доданків та вивести проміжні результати:

$$\sum_{k=1}^{7}\sum_{n=1}^{\infty} k^{n-i} * \ln(i+1)/(n+2)(n+i),$$

де i – номер варіанту. Виконати програму двічі для різних значень точності.

ЛАБОРАТОРНА РОБОТА № 5

Опрацювання одновимірних масивів

МЕТА РОБОТИ: вироблення практичних навиків роботи з даними, представленими у вигляді одновимірних масивів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

У програмуванні, навіть при написанні найпростіших програм, виникає необхідність у великій кількості змінних. Зазвичай, вони різні за типами і за використанням, але бувають ситуації, коли потрібно оголосити велику кількість однотипних впорядкованих змінних. В таких випадках доцільно використовувати масиви.

Масив (array) – це скінчений набір елементів одного (базового) типу, які зберігаються в послідовно розташованих комірках оперативної пам'яті і мають спільну назву. Тип елементів, з яких складається масив (базовий тип), може бути як скалярним, так і структурованим.

Розрізняють одно- та багатовимірні масиви. Одновимірний масив – це вектор *a*1, *a*2, *a*3, *a*4, *a*5 ...*a*n, тобто лінійка величин. При розподілі пам'яті в описовій частині програми під масив резервується стільки місця, скільки передбачає вказана кількість елементів масиву, враховуючи тип елементів. Межі зміни індексів обов'язково повинні бути сталими величинами, а не змінними. У пам'яті комп'ютера елементи одновимірних масивів розташовано послідовно.

Роботу з масивами можна умовно поділити на три частини:

• формування масиву;

• опрацювання масиву;

• виведення масиву.

Опис та формування масивів

Опис масиву складається з імені масиву та службового слова Array, що означає "масив"; Загальний вигляд конструкції опису типу масиву такий:

```
array [<розмір>] of <назва базового типу>;
```

Розмір (кількість елементів) масиву найчастіше задають у вигляді діапазону (записується в квадратних дужках у вигляді двох чисел, розділених двокрапкою: початок..кінець) або назви деякого перерахованого типу даних.

Описати масив можна у розділі опису типів type, у розділі констант const, або у розділі оголошення змінних var. Назви типів масивів і змінних-масивів обирає користувач. Наприклад:

```
Type  mas=array [1..4] of real;     {опис через type}
      mymasyv= array [1..6] of real;
      day=(mon,tue,wed,the,fri,sat,sun);
const
     vydatky:mymasyv=(1.2,1,18,2.4,8.97,1.3);{масив-константа}
Var  a: array [1..100] of integer;  {прямий опис масиву}
     c: array [day] of char
     b: mas;
```

Формування значень елементів масиву можна виконувати у такий спосіб:

• уведення значень елементів масиву з клавіатури або з файла;

• формування значень випадковим чином, з використанням функції-генератора випадкових чисел Random;

• обчислення значень елементів масиву за формулою.

Доступ до елемента масиву здійснюється через назву масиву і номер (індекс) елемента, який записується у квадратних дужках. Тобто, для того, щоб звернутися у програмі до будь-якого елементу масиву, потрібно вказати ім'я змінної з номером елемента в квадратних дужках. Наприклад:

```
Mas[1]:=100;
Readln(Mas[2]);     {зчитуємо 2 елемент масису Mas}
Write(Mas[4]);      {виводимо 4 елемент масису Mas}
Mas[50]:=Mas[49]+Mas[1];
```

Опрацювання масивів

Всі операції, окрім копіювання (присвоєння конкретних значень, додавання, множення тощо), визначені лише над елементами масиву. До класичних задач роботи з масивами можна віднести:

• пошук заданого елемента в масиві;

• знаходження суми (добутку) елементів масиву;

• пошук максимального (мінімального) елемента в масиві;

• упорядкування масиву за ознакою (наприклад за зростанням або спаданням та ін.).

Задачі пошуку в масиві конкретних даних розв'язують методом сканування (перебирання) усіх елементів масиву за допомогою команду циклу (for, while або repeat) і умовної команди, де зазначають умову пошуку. Наприклад, створити масив з десяти цілих чисел і обчислити суму та середнє арифметичне всіх його елементів можна так:

```
Program  Mas1;
Var  i:integer;  s:real;
     a:array [1..10] of integer;
 begin
  for i:=1 to 10 do
   begin
     writeln('введіть значення величини a[',i ']');
      readln(a[i]);
     end ;                {введення елементів масиву}
    s:=0 ;
    for i:=1 to 10 do
     begin
      a[i]:=i; s:=s+a[i]  {знаходження суми елементів масиву}
     end;
    writeln('s=',s );
   readln;
 end.
```

Виведення елементів одновимірного масиву на екран можна виконувати в рядок `(For i:=1 to n do Write(mas[i]);)` або у стовпчик `(For i:=1 to n do Writeln(mas[i]))`.

У попередньому завданні елементи масиву формувалися шляхом введення з клавіатури. Розглянемо програму для обчислення середнього арифметичного перших 100 натуральних чисел. Для формування елементів використаємо функцію Random(A: Integer); яка повертає випадкове число з діапазону від нуля до A.

Для того, щоб використовувати цю функцію, необхідно ініціалізувати датчик випадкових чисел викликом процедури Randomize, яка записується на початку виконуваного блоку (забезпечує іншу послідовність випадкових чисел під час наступного виконання програми).

```
Program  Mas2;
var
  a: Array  [1..100] of  Integer ;
  s,s1: Real ; i: Byte ;
begin
 s:=0; s1:=0;
 Randomize;
 For i:=1 to 100 do
 a[i]:=Random(50); {заповнюємо масив випадковими числами до 50}
    For i:=1 to  100 do
       s:=s+a[i]; s1:=s/100;
  Write('Сума всіх елементів масиву: ', s);
  Writeln('Середнє арифметичне всіх чисел масиву: ', s1);
end .
```

Цей приклад демонструє створення масиву і доступ до його елементів.

Розглянемо ще одну задачу: утворити масив, елементи якого обчислюються за формулою $j=ln(k)-3$, де $k = 1..10$. З відємних елементів масиву побудувати інший масив і вивести його на екран. Якщо шуканих величин немає, вивести про це повідомлення. Програмна реалізація цього завдання на мові Pascal може бути такою:

```
program Mas3;
uses Crt;
var y,g: array [1..10] of real;
    k,n,i: integer;
begin
 clrscr;
 n:=0;                {Спочатку кількість елементів у g = 0}
 for k:=1 to 10 do
   begin
    y[k]:=ln(k)-3;
      if y[k]<0 then      {Перевіримо чи елемент від'ємний}
```

```
        begin
         n:=n+1;          {Збільшимо кількість елементів g}
         g[n]:= y[k];
        end;              {Знайдемо n-ий елемент}
      writeln('y(' ,k,>' ,y[k]:7:2);
   end;
  if n=0 then writeln('Масив у від"ємних елементів не має')
  else
    for i:=1 to n do        {Виведемо утворений масив g}
      writeln('g[' ,i ,']=' ,g[i]:7:2);
 readln
end.
```

Одним із найскладніших завдань при роботі з масивами є впорядкування елементів масиву. Для розв'язання цієї задачі існує декілька алгоритмів сортування: обмінне сортування («бульбашка»), сортування вибором, метод уставки, швидке сортування тощо.

Алгоритм сортування вибором:

1. Установити номер найбільшого елемента масиву.

2. Поміняти місцями найбільший і останній елементи.

3. Повторити пункт 1 і 2 над остачею масиву (без останнього елемента).

Застосовувати цей метод до елементів масиву, що залишилися, доки залишок не скоротиться до одного елемента. Аналогічно сортування вибором можна застосувати до найменшого елемента, міняючи його з першим. У результаті все одно отримаємо зростаючу (незменшувану) послідовність елементів масиву.

Алгоритм обмінного сортування («бульбашка»):

1. Порівняти два поруч розташованих елементи.

2. Якщо пара порушує потрібний порядок слідування, елементи міняють місцями.

Порівняння відбувається до кінця масиву. Обмін здійснюється доти, поки прохід у масиві не викличе жодного обміну.

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв'язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого

середовища розробки Turbo Pascal набрати тексти підготовлених програм. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.

3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які давали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до цієї роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

У наступних завданнях потрібно написати програми, що використовують одновимірні масиви елементів відповідного типу. Зазначені масиви формувати шляхом введення елементів з клавіатури.

1. Створити одновимірний масив з 10 елементів і знайти суму парних елементів і добуток непарних елементів.
2. Сформувати одновимірний масив з 12 елементів; обчислити та вивести на екран добуток всіх елементів масиву.
3. Створити одновимірний масив з 8 елементів. Обчислити добуток всіх парних елементів масиву.
4. Створити одновимірний масив, що містить 11 елементів. Обчислити суму всіх елементів, які більше за 5.
5. Створити одновимірний масив з 7 елементів. Визначити кількість елементів рівних 7. За відсутності шуканих даних, вивести відповідне повідомлення.
6. Створити одновимірний масив, що містить 9 елементів. Помножити всі елементи на 20. Вивести отримай масив на екран.
7. Утворити одновимірний масив з 15 елементів. Всі парні елементи замінити на 2, а непарні збільшити вдвічі. Вивести отриманий масив на екран.

8. Створити одновимірний масив з 12 елементів. Знайти мінімальний і максимальний елементи масиву.
9. Створити одновимірний масив з 6 елементів. Обчислити різницю між максимальним і мінімальним елементом масиву.
10. Створити одновимірний масив, що містить 9 елементів. Скопіювати всі його елементи в інший масив. Вивести отриманий масив на екран.
11. Написати програму, що формує масив з 10 чисел і друкує його на екрані в прямому і зворотному порядку.
12. Створити одновимірний масив, що містить 9 елементів. Поділити всі його елементи на 3, вивести отримай масив на екран.
13. Створити одновимірний масив з 10 елементів. Всі непарні елементи замінити на -1, а парні зменшити вдвічі. Вивести отриманий масив на екран.
14. Утворити одновимірний масив, що містить 9 елементів. Помножити всі його парні елементи на 3; вивести отриманий масив на екран.
15. Створити одновимірний масив з 11 елементів. Обчислити суму та різницю максимального і мінімального елементів масиву.

Задача 2

1. Утворити і вивести масив з елементами $a_n = \ln(n) - 2$, де $n = 1,...10$. Визначити порядковий номер найменшого з додатніх елементів масиву. За відсутності шуканих даних, вивести відповідне повідомлення.
2. Створити масив з 50 чисел та заповнити його випадковими числами. Визначити, скільки в ньому елементів, відмінних від останнього елементу. Вивести їхню кількість на екран.
3. Утворити масив з елементами $a_n = 2^n - 10$, де $n = 1,...15$. Вивести на екран спочатку всі додатні його елементи, а потім всі від'ємні. За відсутності шуканих даних, вивести відповідне повідомлення.
4. Утворити масив з 20 чисел. Заповнити його випадковими додатніми числами, після чого змінити в усіх чисел знак на протилежний. Вивести масив, що утворився, на екран.

5. Утворити і вивести масив з елементами $a_n = 3n^2 - 16n$, де $n = 1,...15$. Знайти номер першого входження деякого числа *x* в цей масив або вивести повідомлення, що такого елемента в отриманій послідовності чисел немає.
6. Дано масив, елементи якого визначаються за формулою $a_n = \frac{3n}{4} - 2$, де $n = 1,...20$. Вивести на екран порядковий номер і значення останнього від'ємного елемента масиву. За відсутності шуканих даних, вивести відповідне повідомлення.
7. Утворити шляхом введення з клавіатури масив з деякої кількості символів. Визначити, скільки в ньому символів `'a'`. Вивести інформацію на екран.
8. Утворити і вивести одновимірний масив a_n з 15 елементів, де $a_n = truns(\ln(n) - 2)$. Визначити, з якої кількості негативних чисел він починається. За відсутності шуканих даних, вивести повідомлення про це.
9. Утворити масив ***y*** з 30 чисел. Заповнити його випадковими додатніми числами. Побудувати масив ***g***, який складається з парних елементів масиву ***y***. Якщо шуканих величин немає, вивести відповідне повідомлення.
10. Утворити масив з 25 чисел. Заповнити його випадковими числами. Якщо на парному місці є елемент кратний 3, то замінити цей елемент на його квадрат. Отриману послідовність вивести на екран.
11. Утворити послідовність з елементів $a_n = \left(\frac{1}{3}\right)^n - 1$, де $n = 1,...15$. Визначити, чи є ця послідовність зростаючою; вивести повідомлення про це на екран.
12. Дано масив, елементи якого визначаються за формулою $a_n = truns(\frac{3n}{4} - 2)$, де $n = 1,...20$. Поміняти місцями мінімальний елемент масиву та елемент з номером m (значення m вводиться з клавіатури). Вивести отриманий масив на екран.
13. Утворити масив ***g*** з 35 чисел. Заповнити його випадковими цілими числами.Створити новий масив з елементів масиву ***g,*** значення яких кратні 4. Вивести отримай масив на екран.

14. Утворити і вивести масив з елементами $a_n = 3n^2 - 16n$, де $n = 1,...17$. Утворити новий масив, який відрізняється від початкового тим, що всі непарні елементи подвоєні. Вивести отримай масив на екран.
15. Створити масив, елементи якого визначаються за формулою $a_n = truns(\frac{3n}{5})$, де $n = 1, ...16$. Поміняти місцями мінімальний та максимальний елементи масиву. Вивести отримай масив на екран.

ЛАБОРАТОРНА РОБОТА № 6

Робота з двовимірними масивами

МЕТА РОБОТИ: вироблення практичних навиків роботи з даними, представленими у вигляді двовимірних масивів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Масив називається двовимірним, якщо для доступу до його елементів необхідно вказати значення двох індексів (дані можуть бути подані у вигляді таблиці – матриці). Перший індекс вказує номер рядка, а другий – номер стовпця, на перетині яких стоїть елемент.

Тривимірним масивом є тривимірна матриця або паралелепіпед величин, що складається з набору двомірних матриць (рис.1).

Лінійка тривимірних масивів складає чотиривимірний масив, матриця тривимірних масивів (паралелепіпедів величин) – п'ятимірний масив і так далі. Кількість розмірностей масиву не обмежена; проте слід пам'ятати про те, що кількість елементів в масиві росте в геометричній прогресії залежно від розмірності. Зупинимося детальніше на роботі з двовимірними масивами.

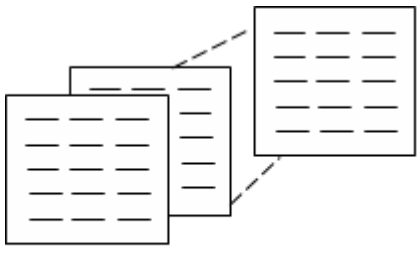


Рис. 1

Розглянемо приклади описів деяких масивів:

`const m1: array [1..2, 1..4] of integer=((4,3,5,3), (4,4,5,3));` – масив-константа m1, що має 2 рядки і 4 стовпці елементів. В даному масиві m1[l,l]=4, m1[l,2]=3..... m1[2,3]=5.

`var doba: array [0..23] of array [0..59] of integer;` – масив doba, який міститиме 24x60 елементів цілого типу.

`p: array [1..9, 1..9] of integer;` – двовимірний масив *p* розмірності 9 на 9 (містить 81 елемент). Спочатку вказується кількість рядків, потім – через кому – кількість стовпців.

Значення елементам масивів doba і *p* можна надати командою присвоєння двома способами, наприклад, так: `doba[16][30]:=5`, `doba[16,30]:=5`.

Розглянемо в якості прикладу роботи з двомірними масивами програму, яка створює масив і записує в нього у вигляді матриці таблицю множення (добуток двох потрібних чисел знаходиться на перетині рядка і стовпця, що починаються з цих чисел), після чого роздруковуєїї на екран.

```
program Mas2_1;
var
   Mas: Array [1..10, 1..10] of  Integer ;
   i, j: Byte ;
begin
     For i:=1 to 10 do
    For j:=1 to 10 do
      Mas[i,j]:=i * j; {формуємо елементи масиву}
   For i:=1 to 10 do
   begin
      For j:=1 to 10 do
        If  Mas[i,j]<10 then Write(Mas[i,j], '    ')
         else Write(Mas[i,j], '  ');  {виводимо масив на екран}
         Writeln;
    end;  Readln;
  end.
```

В останньому циклі можна було не використовувати оператор розгалуження і вивести масив наступним чином `write(Mas[i,j],' ')`, формуючи лише проміжки між елементами. Однак, у цьому випадку всі елементи створюваної таблиці, значення яких більше десяти, будуть дещо зміщені відносно інших.

Наступний приклад демонструє роботу з елементами двомірних масивів. Нехай маємо масив ***b***, елементи якого обчислюються за формулою $b_{ij}=i+j^2$, де $i, j = 1..5$. Потрібно утворити і вивести масив ***y***, який складається з

тих елементів масиву ***b***, значення яких більші, ніж 6. Обчислити кількість елементів масиву ***y***. Масив ***b*** вивести у вигляді матриці 5x5. Якщо шуканих даних немає, вивести про це повідомлення. Реалізувати це завдання можна так:

```
program Mas2_2;
type mas1= array [1..5, 1..5] of real;
     mas2= array [1..25] of real;
var b: mas1; y: mas2;
    i,j,k:integer;
begin
  k:=0;  {Спочатку кількість k елементів масиву y  рівна нулю}
  writeln('перший масив b');
   for i:=1 to 5 do begin
   for j:=1 to 5 do begin
     b[i,j]:=i+j*j;         {Обчислюємо елементи масиву b}
     write(b[i,j]:5:2);       {Виведемо елементи i-го рядка}
       if b[i,j]>16  then
           begin
            k:=k+1; y[k]:=b[i,j];
           end;              {Визначимо k-ий елемент масиву y)
        end;
      writeln; end;
   if k=0 then write('y масиві немає елементів >6')
    else begin
      writeln('утворений масив');
      for i:=1 to k do writeln(y[i]:5:2); {Виведемо масив y }
      writeln('Кількість k елементів масиву y =',k:2);
    end; readln
end.
```

У попередніх завданнях масиви формувалися шляхом обчислення значень елементів масиву за формулою. Розглянемо задачу, в якій двомірний масив формується шляхом введення його елементів з клавіатури.

Нехай потрібно написати програму, що формує квадратну матрицю шляхом введення елементів з клавіатури та виводить на екран елементи її головної діагоналі (рис.2.) Нехай індекс рядків *i*, а стовпців *j*, тоді на головній діагоналі лежать елементи, для яких $i = j$.

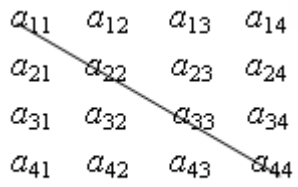


Рис. 2

```
program  Mas2_3;
 Var  i,j,k:integer;
      a:array [1..4,1..4] of integer;
      b:array [1..4] of integer;
 begin
  for  i:=1 to  4 do
  for  j:=1 to  4 do
    begin
```

```
      writeln('введіть a[',i,',',j ']');
      readln(a[i,j ]);
    end ;       {введення елементів масиву}
writeln('отриманий масив має вигляд ');
 for i:=1 to 4 do
    begin  writeln ;         {переводимо курсор на новий рядок}
      for  j:=1 to 4 do
      write (a[i,j], '  ');   {виведення елементів масиву}
  end ;
     k:=0 ;             {встановлюємо індекс для занесення b[1]}
     for  i:=1  to  4 do
     for  j:=1  to  4 do
    if  i=j  then
  begin
    k:=k+1;
    b[k]:=a[i,j];
  end ;          {шукаємо елементи головної діагоналі}
     writeln('на головній діагоналі лежать елементи:');
     writeln('a[1,1]=',b[1],'a[2,2]=',b[2],'a[3,3]=',b[3],
             'a[4,4]=',b[4]);
readln ;
end.
```

Під час виконання програми користувачу буде видано запити на введення 16 елементів матриці, потім буде виведений отриманий масив у вигляді:

a_{11} a_{12} a_{13} a_{14}

a_{21} a_{22} a_{23} a_{24}

a_{31} a_{32} a_{33} a_{34}

a_{41} a_{42} a_{43} a_{44}

після чого будуть знайдені елементи головної діагоналі, які будуть занесені в інший масив і надруковані у такому вигляді:

```
   на головній діагоналі лежать елементи:
a[1,1]= число1  a[2,2]= число2  a[3,3]= число3   a[4,4]= число4
```

Цю програму можна узагальнити – надати користувачу можливість самому обирати розмірність масиву, тобто вводити відповідну розмірність з клавіатури.

Одним з типових завдань при роботі з масивами є задача про відшукання в масиві елементу, що задовільняє деякій умові. Розглянемо на конкретному прикладі алгоритм знаходження в масиві розміром до 10x10 мінімального елементу та виведення рядка, що містить даний елемент.

```
program  Mas2_4;
  Var  i,j,min,m,n,k:integer ;
       a:array [1..10,1..10] of  integer ;
 begin
```

```
  writeln('ввести m,n');
  readln (m,n );
    for  i:=1  to  m do                {m – кількість рядків}
    for  j:=1  to  n do                {n – кількість стовпців}
    begin  writeln('ввести a[', i' ', j']');
       readln(a[i, j]);    {формування елементів масиву a}
    end ;
      for  i:=1  to  m do begin
       for  j:=1  to  n do
        write(a[i, j],'  '); writeln; {виведення масиву a}
          end ;
        min:=a[1, 1];
        k:=1; {приймаємо за мінімальний перший елемент масиву}
     for i:=1 to m do {заносимо в min елементи a, менші за min}
     for  j:=1  to n do begin
        if  a[i, j]<min  then
           begin
             min:=a [i, j]; k:=i;
           end ;
    end ;
 writeln('мінімальний елемент масиву ', min);
 writeln('рядок, що містить найменший елемент');
   for j:=1 to n do
   write(a[k,i], '  ');
     readln;
 end .
```

Розглянемо ще один приклад, який демонструє основні принципи опрацювання двовимірних та одновимірних масивів. Напишемо програму, що перемножує матрицю на вектор і виводить результат множення на екран. Нехай матриця може мати розмірність 5x7, відповідно вектор ***b*** може мати розмірність 7, а вектор ***c*** – розмірність до 5 елементів. Зауважимо, що для обчислення кожного елемента необхідно кожен елемент рядка масиву ***a*** помножити на відповідний за індексом елемент масиву ***b***, а потім всі ці добутки скласти. Отже, кількість елементів масиву ***c*** дорівнюватиме кількості рядків матриці ***a*** (і, відповідно, кількості елементів масиву ***b***).

```
Program  Mas2_5;
Var i,j,n,m:integer;
    a:array [1..5,1..7] of  integer;
    b:array [1..7] of  integer;
    c:array [1..5] of  integer;
begin
writeln ('ввести кількість рядків і стовпців');
 readln (n,m );
  for  i:=1  to  n do
  for  j:=1  to  m do
   begin  writeln ('ввести a[',i,',',j '] елемент матриці a');
     readln(a[i,j ]);
```

```
   end ;
     for  j:=1 to m do
     begin  writeln ('ввести b[',j '] елемент вектора b');
        readln (b[j]);
     end ;
   for i:=1  to n do   {множення матриці на вектор}
 begin  c [i]:=0;
   for  j:=1  to  m do
    c[i]:=c[i]+ a[i,j]*b[j]; end;
    writeln('масив a');
      for  i:=1  to n do
        begin  writeln ;          {почати новий рядок}
      for  j:=1  to m do
       write(' ',a [i,j ]);
        end ; writeln ;
    writeln('масив b');
         for  j:=1  to  m do
    write('  ',b [j]);               writeln ;
    writeln('результуючий масив c ');
         for  i:=1  to  n do
    write('  ',c [i]);     {виводимо результат}
   readln ;
end .
```

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає в підборі таких значень вхідних даних, які б давали змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до цієї роботи, в якому обов’язково навести алгоритми розв’язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Для реалізації наступних завдань потрібно написати програми, що використовують двовимірні масиви елементів відповідного типу.

1. Створити двовимірний масив розмірності 3х5 методом введення елементів з клавіатури. Замінити всі елементи кратні 5, нулями. Вказати кількість проведених замін. За відсутності шуканих даних, вивести відповідне повідомлення.
2. Утворити і вивести на екран двовимірний масив з елементами $a_{ij} = 2^i - 10j$, де $i, j = 1,...4$. Вивести на екран спочатку всі додатні його елементи, а потім всі від'ємні. За відсутності шуканих даних, вивести відповідне повідомлення.
3. Створити цілочисельну матрицю розмірності 3х5 з елементами від 0 до 100 за допомогою генератору випадкових чисел та вивести її на екран. Вивести на екран всі елементи другого рядка цієї матриці.
4. Створити матрицю з дійсних чисел розміром 3х3 шляхом введення елементів з клавіатури. Занести в окремий масив всі елементи цього масиву, значення яких більше за 5 та обчислити їхню суму. За відсутності шуканих даних, вивести відповідне повідомлення.
5. Дано двовимірний масив розмірності 3х5, елементи якого визначаються за формулою $a_{ij} = 3i^2 - 16j$, $i, j = 1,...5$. Поміняти місцями мінімальний та максимальний елементи масиву та вивести отриманий масив на екран.
6. Створити цілочисельний двовимірний масив розмірності 4х5 (елементи масиву вводити з клавіатури). Знайти суму елементів головної діагоналі. Елементи масиву, що дорівнюють нулю, замінити на 1. За відсутності шуканих даних, вивести відповідне повідомлення.
7. Утворити і вивести матрицю з елементами $a_{ij} = 3i^2 - 16j$, де $i, j = 1,...5$. Обчислити суму та різницю максимального і мінімального елементів масиву. Перевірити, чи є матриця симетричною щодо головної діагоналі.
8. Дано двовимірний масив розмірності 4х6, елементи якого визначаються за формулою $a_{ij} = \frac{3i}{4} - 2j^2$, де $i, j = 1,...4$. Знайти максимальний елемент масиву та вивести на екран рядок, що містить цей елемент.
9. Створити двовимірний масив розмірності 5х5 (елементи масиву вводити з клавіатури). Знайти кількість від'ємних елементів, що розміщені над головною діагоналлю.

10. Створити дійсний масив розмірності 3х6 шляхом введення елементів з клавіатури та вивести його на екран. Вивести номери стовпців, що містять від'ємні елементи.
11. Створити матрицю розмірності 7х3 (елементи масиву вводити з клавіатури). Отримати нову матрицю шляхом ділення всіх елементів цієї матриці на її найбільший за модулем елемент.
12. Створити цілочисельний двовимірний масив, що має 8 рядків і 10 стовпців за допомогою генератору випадкових чисел. Вивести на екран найбільший і найменший елементи цього масиву, вказавши на перетині яких рядків і стовпців перебувають ці елементи.
13. Створити цілочисельну матрицю розмірності 4х6. Визначити індекси першого нульового елементу матриці. Обхід матриці здійснювати за стовпцями. За відсутності шуканих даних, вивести відповідне повідомлення.
14. Сформувати та вивести на екран матрицю за таким правилом:

```
0 0 0 0 0
0 1 0 0 0
0 0 2 0 0
0 0 0 3 0
0 0 0 0 4
```

15. Дана дійсна матриця розміру 6 х 6. Знайти середнє арифметичне елементів кожного стовпця та середнє арифметичне елементів кожного рядка, що має парний номер.

Задача 2

1. Створити дійсну матрицю розмірності *n*х*m* (значення *m* та *n* вводиться з клавіатури) Знайти загальну суму елементів тільки тих стовпців, які мають хоч би один нульовий елемент. Якщо шуканих величин немає, вивести про це повідомлення
2. Утворити і вивести матрицю розмірності *n*х*m* з елементами $a_{ij} = sin(i+j/2)$ (значення *m* та *n* вводиться з клавіатури). З'ясувати скільки позитивних елементів містить даний масив. За відсутності шуканих даних, вивести відповідне повідомлення.

3. Створити та вивести на екран цілочисельну матрицю розмірності *n*x*m* за допомогою генератору випадкових чисел. Вивести номери рядків, що містять позитивних елементів більше, ніж негативних.
4. Створити дійсну матрицю розмірності *n*x*m* з елементами $a_{ij}= cos(i^2+j/2)$ (значення *m* та *n* вводиться з клавіатури). Поміняти місцями рядки з максимальним і мінімальним елементами.
5. Сформувати шляхом введення елементів з клавіатури дійсну квадратну матрицю розмірності *n*, вивести її на екран. Поміняти місцями елементи головної і бічної діагоналей матриці. Вивести на екран оновлену матрицю.
6. Утворити шляхом введення з клавіатури двовимірний масив ***y*** розмірності *n*x*m* (значення *m* та *n* вводиться з клавіатури). Побудувати масив ***g***, який складається з парних елементів масиву ***y***.
7. Сформувати та вивести на екран матрицю за таким правилом:

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

8. Створити дійсну матрицю розмірності *n*x*m* за допомогою генератору випадкових чисел та вивести її на екран. Знайти середнє арифметичне кожного рядка матриці та сформувати з цих значень вектор.
9. Дано натуральне число *n* (значення *n* вводиться з клавіатури). З'ясувати, скільки позитивних елементів містить матриця a_{ij}, елементи якої визначаються за формулою $a_{ij}=sin(i+j/2)$, де *i, j*=1..*n*. За відсутності шуканих даних, вивести відповідне повідомлення.
10. Створити цілочислену матрицю розмірності *n*x*m* (значення *m* та *n* вводиться з клавіатури) за допомогою генератора випадкових чисел. Знайти найбільший та найменший елементи масиву та поміняти їх місцями. Вивести масив на екран.
11. Створити цілочисельний двовимірний масив розмірності *n*x*m*, шляхом введення елементів з клавіатури. Розташувати всі елементи матриці за зростанням. Обхід матриці здійснювати за рядками.

12. Створити цілочисельний двовимірний масив розмірності *n*x*m*, шляхом введення елементів з клавіатури. Знайти добуток всіх від'ємних елементів матриці та замінити на нього всі діагональні елементи даної матриці
13. Ввести дійсну матрицю розмірності *n*x*m* (значення *m* та *n* вводиться з клавіатури) по рядках, а вивести по стовпцях.
14. Утворити дійсну квадратну матрицю порядку *n* (значення *n* вводиться з клавіатури), елементи якої визначаються за формулою $a_{ij} = 2i^2 - j^2$. З'ясувати, чи вірно, що найбільше із значень елементів головної діагоналі більше, ніж найменше із значень елементів бічної діагоналі.
15. Сформувати та вивести на екран двовимірний масив за таким правилом:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

ЛАБОРАТОРНА РОБОТА № 7

Робота з підпрограмами. Підпрограми-процедури. Використання підпрограм-функцій

МЕТА РОБОТИ: вироблення практичних навиків роботи з двома типами підпрограм: процедурами та функціями.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Автономна частина програми, яка реалізує визначений алгоритм і припускає звертання до нього із різних місць загальної програми, називається **підпрограмою**. Підпрограми призначені для реалізації алгоритмів опрацювання окремих частин деякої складної задачі. Розміщення підпрограм у програмі подібно до ієрархічного принципу побудови файлової системи.

Підпрограми поділяються на стандартні та підпрограми користувача. Стандартні підпрограми описувати не потрібно – вони містяться у стандартних модулях Crt, Dos, Graph тощо. Підпрограма користувача – це

поіменована група команд, яку створюють і описують у розділах **procedure** або **function** і до якої звертаються з будь-якого місця програми потрібну кількість разів.

В мові Pascal використовується два види підпрограм – підпрограми-процедури та підпрограми-функції, які відрізняються між собою структурою та способом виклику.

Всі процедури і функції користувача потрібно розмістити в розділах **procedure** і **function** описої частини програми. Своєю чергою, в їхніх розділах опису можна розміщувати процедури і функції 2-го рівня тощо. Наприкінці підпрогарми після end ставиться крапка з комою.

У зв'язку з тим, що кожна процедура або функція може містити свій розділ опису, виникає проблема локалізації імен змінних. Змінні і константи можна описувати у зовнішніх (незалежних) або внутрішніх (вкладених) блоках. Тому виникає два поняття змінних і констант: локальні і глобальні.

Локальні – існують у рамках того блока, де вони описані. Тобто за межами даного блока їх використовувати не можливо.

Глобальні – описані у зовнішньому блоці, але можуть використовуватися у внутрішньому блоці.

Наявність локальних і глобальних змінних дає змогу при написанні складних програм використовувати одні і ті ж імена у різних блоках, що полегшує взаємодію програмістів. Однак при написанні програм потрібно дотримуватися правил локалізації змінних.

Підпрограми-процедури мають таку структуру:

```
Procedure <ім'я> (список формальних параметрів);
      <розділ локальних даних>
 Begin
      <розділ виконавчих операторів>
 End;
```

Перший рядок складає заголовок процедури (ім'я процедури обирає програміст). У списку формальних параметрів описуються через крапку з комою параметри та інформація про їхній тип. Деякі параметри призначені для передачі даних в процедуру, інші для повернення результатів з процедури до тієї програмної одиниці, яка її викликала.

Наприклад, у процедурі:

```
Procedure R(x, y: real; k: integer; var s: real);
```

`x, y, k` – параметри-значення; `s` – параметр-змінна (використовується для повернення результату в програму, яка викликала цю процедуру), перед яким повинно стояти службове слово **var**. Параметри-значення – аргументи функції, а параметри-змінні – результати виконання процедури.

У розділі локальних даних (може бути відсутнім) описують ті дані, які використовуються тільки у самій процедурі (параметри циклів, робочі змінні та масиви, тощо). Всередині підпрограми записують послідовність операторів, які реалізують потрібний алгоритм. Зв'язок між окремими частинами програми здійснюється через списки формальних параметрів та за допомогою глобальних змінних. Глобальні дані описуються в головній програмі і не являються фактичними параметрами при виклику підпрограм.

Розглянемо просту програму: нехай потрібно розташувати в порядку зростання три цілі числа, введені з клавіатури. Напишемо програму з використанням процедури.

```
Program  Pidpr1;
Var  x,y,z: integer;
Procedure Obmin(Var a,b: integer);
 Var c: integer;      {c - незалежна локальна змінна}
 Begin
  if  a>b   then
   begin
    c:=a; a:=b; b:=c;   {міняємо місцями  a i b}
   end  ;
 end;
Begin  {переходимо до розділу операторів основної програми}
  Writeln('Введіть три числа');
  Readln(x,y,z);
   Obmin(x,y); {виклик процедури}
   Obmin(x,z);
   Obmin(y,z);
   Writeln('Числа в порядку зростання: ', x,y,z);
End.
```

При необхідності вийти із середини блока можна використавши стандартну процедуру Exit, яка має форму: Exit (аргумент), де аргумент – ім’я функції, процедури, програми, службове слово program. Якщо аргументом є ім’я програми або службове слово program, то виконання програми достроково завершується. Якщо аргументом є ім’я процедури або функції, то вони завершуються, і управління передається оператору, який стоїть після оператора виклику.

Розглянемо ще один приклад реалізації завдань з використанням

процедур. Потрібно за введеним значенням радіуса кола **R** визначити довжину кола **L** і площі круга **S**, обмеженого цим колом. Всі дії, пов'язані з обисленням параметрів та виведенням результатів, помістимо в процедуру:

```
Program Pidpr2;
var R: real;
    S, L: real;
Procedure krug(R: real; var S, L: real);
 const pi = 3.14;
 begin
   L:= 2*pi*R;
   S:= pi*sqr(R);
   writeln('довжина кола = ', L:6:2);
   writeln('площа кола = ', S:6:2);
 end;
begin
  writeln('введіть радіус круга');
  read(R);
  krug(R,L,S);    {виклик процедури}
end.
```

Підпрограми-функції використовуються для реалізації алгоритму та повернення в головну програму одного результату у вигляді імені функції. За своїм призначенням і способом побудови функції подібна до процедур. Ім'я функції також обирає користувач.

Підпрограма-функція має тільки один результат виконання, який позначається іменем функції і передається в основну програму (тому імені функції присвоюють необхідний тип даних). Функції можна викликати у середині математичного виразу.

Структура функції:

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

Типом функції може бути цілий, дійсний, логічний, символьний та рядковий тип **String.**

Відносно формальних параметрів, локальних та глобальних даних у функції діють такі ж самі обмеження та вимоги, що і в процедурах. Звернення до функції виконується з якого-небудь арифметичного виразу так,

як і до стандартних функцій типу *sin*(x), *ln*(x), тощо. Результат роботи функції передається в місце її виклику.

Особливості використання функцій

1. Функція при описі повинна отримувати свій тип (тип значення, що вона повертає в програму). Значення, що повертається функцією, задається простим присвоюванням значення імені функції усередині неї.

2. Як і стандартні функції Pascal, власні можуть бути використані усерединіпроцедур та всередині стандартних конструкцій.

Розглянемо використання підпрограми-функції для обчислення степеня. Нехай потрібно обчислити значення виразу $z = x^y + y^x$.

```
Program Pidpr3;
Var x,y,z: Real ;
Function Step(a:real; b:real):real;
 Begin
  If a<=0 then
   Begin
     Writeln('перевірте дані');
     Halt(0);
    End;
        If b=0 then Step:=1
          else Step:=exp(b*ln(a));
  End;
Begin
 Writeln('введіть x та y');
 Readln(x,y);
  Z:=Step(x,y)+Step(y,x);  {виклик функції}
  Writeln(z);
End.
```

Отже, якщо процедури і функції розробляються для багаторазового виклику, краще їх використовувати з використанням параметрів, щоб досягнути максимальної незалежності від тієї програми, де вони використовуються.

Методи звертань до процедур і функцій

Існує 3 варіанти звертання до процедур і функцій: із зовнішнього блоку; із сусідньої функції; із самої функції (рекурсія).

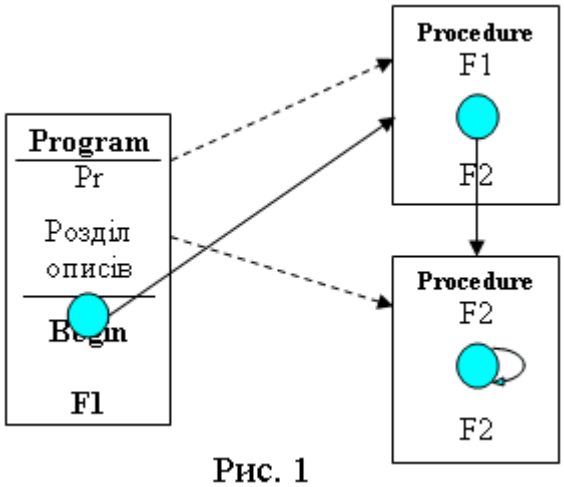


Рис. 1

На рис. 1 умовно подано:

1) звертання до процедури **F1** із тіла програми **Pr;**

2) звертання до процедури **F2** із тіла процедури **F1**;

3) звертання до процедури **F2** всередині самої себе – рекурсія.

Отже, **рекурсія** – це алгоритмічна конструкція, де підпрограма викликає сама себе. Рекурсія дає змогу записувати циклічний алгоритм, не використовуючи команду циклу. Розглянемо для прикладу рекурсивну функцію обчислення суми цілих чисел від *a* до *b*:

```
function Suma(a,b:integer):integer;
begin
if a=b then Suma:= a              {це стоп-умова рекурсії}
else Suma:= b + Suma(a, b-1)      {це неявний цикл}
end;
```

Використовуючи рекурсію, потрібно правильно складати стоп-умови, які забезпечують закінчення циклічних обчислень.

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв'язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які давали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до лабораторної роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Програмно реалізувати поставлені завдання, використовуючи підпрограми-процедури.

1. Ввести з клавіатури два числа, знайти сумму квадратів цих чисел та вивести результат на екран.

2. Визначити суму мінімального та максимального з чотирьох введених користувачем дійсних чисел. Вивести на екран мінімальне і максимальне число та отриману суму.
3. Розташувати введені з клавіатури чотири цілих числа в порядку зростання та в порядку спадання. Вивести на екран отримані послідовності.
4. Написати програму знаходження добутку та частки максимального і мінімального з трьох введених користувачем цілих чисел. Вивести на екран отримані результати.
5. Визначити, яка з введених користувачем трьох довільних літер латинського алфавіту розташована ближче до початку в алфавіті.
6. Написати програму визначення числа, отриманого виписуванням у зворотному порядку заданого тризначного числа
7. Сформувати шляхом введення з клавіатури одновимірний масив. Використовуючи підпрограму-процедуру подвоїти всі його елементи. Вивести утворений масив на екран.
8. Обчислити добуток та частку цифр введеного з клавіатури тризначного числа.
9. Знайти найбільший загальний дільник п’яти введених з клавіатури чисел.
10. Скласти програму для обчислення значення виразу C= n!(n-m)! для натуральних n і m.
11. Використовуючи процедури, знайти суму максимального і мінімального елементів двомірного масиву розмірності 3x4 (масив формувати випадковим чином).
12. Визначити та вивести на екран суму та різницю цифр введеного користувачем чотиризначного числа.
13. За введеними з клавіатури значеннями довжин сторін трикутника обчислити його площу та величини кутів.
14. Визначити та вивести на екран довжини сторін трикутника та його площу за координатами його вершин (вводяться з клавіатури).
15. Сформувати шляхом введення з клавіатури масив розмірності 3x3. Використовуючи підпрограму-процедуру, видалити рядок, в якому міститься мінімальний елемент матриці. Вивести отриманий масив на екран.

Задача 2

Програмно реалізувати поставлені завдання, використовуючи підпрограми-функції.

1. Створити функцію для обчислення *tg*(*x*) та обчислити значення виразу *c*=5*tg*(*x*)+1/3*ctg*(*x*)+*tg*(*2x*).

16. Дані дійсні числа *s*, *t* (вводяться з клавіатури). Визначити та вивести на екран значення виразу *v* = *f*(*t*, 2*s*, 1.17)+ *f*(2.2, *t*, *s*-*t*), де

$$f(a,b,c)= (2*a-b-sin(c))/(5 + c).$$

2. Написати програму для обчислення подвоєнного квадрату площі трикутника за довжинами його сторін. У вигляді функції оформити обчислення площі трикутника за формулою Герона (*S*=*sqrt*(*p**(*p*-*a*)*(*p*-*b*)*(*p*-*c*)), де *a*, *b*, *c* - сторони трикутника, *p* – його півпериметр).

3. Обчислити значення виразу: $b = \dfrac{f_1(x^2) + f_2{}^2(y+1)}{1+1/3 f_3(z)}$, де $f_1(x) = x+256.4$; $f_2(y) = y+256.4$; $f_3(z) = z+256.4$.

4. Скласти підпрограму функцію для обчислення факторіалу і використати її для обчислення факторіалу будь-якого введеного з клавіатури натурального числа.

5. За введеними користувачем координатами вершин, встановити, чи є цей трикутник рівностороннім. Як функцію оформити обчислення довжини сторони трикутника.

6. Обчислити значення виразу: $b = \dfrac{\sqrt{s^3} - 3s}{e^s + 1/3}$, де параметр *s* дорівнює добутку коренів квадратного рівняння $ax^2+bx+c=0$ (коефіцієнти *a, b, c* вводяться з клавіатури).

7. Написати програму для обчислення периметру трикутника, заданого координатами вершин. В якості функції оформити обчислення довжини сторони трикутника.

8. Використовуючи підпрограму-функцію для обчислення степеня, скласти програму для обчислення значення виразу $b = x^y + y^x + x\,y$ (значення x та y вводяться з клавіатури).

9. Скласти підпрограму функцію для обчислення факторіалу і використати її для обчислення значення виразу F=*m*!–*k*! для натуральних *n* і *k* (вводяться користувачем).
10. Використовуючи підпрограми-функції, визначити мінімальний та максимальний елемент одновимірного масиву, їхню суму та добуток (елементи масиву вводити з клавіатури).
11. Написати функцію для обчислення *ctg*(*x*) та обчислити значення виразу $a=tg^2(x)+ctg^3(x)+1/5tg(x)$.
12. За допомогою підпрограми-функції визначити, чи є число простим. Знайти та вивести на екран всі прості числа від 1 до заданого N.
13. Трикутник заданий координатами вершин (вводяться користувачем). Обчислити периметр цього трикутника та вивести результати.
14. Обчислити значення виразу: $b=\frac{s^2+2s}{1+1/3\sqrt{s}}$, де параметр *s* дорівнює сумі коренів квадратного рівняння $ax^2+bx+c=0$ (коефіцієнти *a, b, c* вводяться з клавіатури).
15. За введеними користувачем координатами вершин, встановити, чи є цей чотирикутник паралелограмом. В якості функції оформити обчислення довжини сторони трикутника.

ЛАБОРАТОРНА РОБОТА № 8

Опрацювання рядків символів (string). Основні процедури для роботи з рядками

МЕТА РОБОТИ: Вироблення практичних навиків роботи з даними, представленими у вигляді рядків символів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Рядковий тип даних узагальнює поняття символьних масивів, дає змогу динамічно змінювати довжину рядка. Змінна такого типу може зберігати будь-які символи. Кількість символів в рядку (довжина рядка) може динамічно змінюватися від 0 до 255.

Для визначення рядкових даних використовується ідентифікатор String, за яким у квадратних дужках вказується максимальна довжина рядка. Якщо значення не вказане, то за замовчуванням довжина рядка встановлюється 255 байт. Наприклад: `Line: string[80]; Line1: string;`

Отже, важлива різниця між рядками і символьними масивами в тому, що рядки можуть динамічно змінювати свою довжину. Для рядкових змінних пам'ять виділяється відповідно максимального значення, а використовується лише частина, яка реально зайнята символами рядка у цей час, тобто для N символів виділяється N+1 байт пам'яті, із яких N байт призначено для зберігання символів рядка, а 1 байт – для значення поточної довжини цього рядка (рис.1).

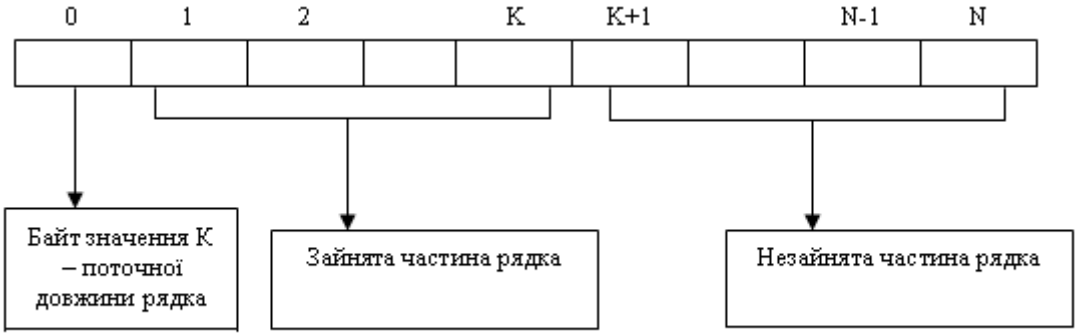


Рис.1

Змінну рядкового типу визначають у розділі опису типів або безпосередньо у розділі опису змінних. Рядкові дані також можна використовувати у програмі як константи. Наприклад:

```
const address='пл. Соборна, 1';
type Line = string [125];
var S1: Line;
    S2: string[50];
```

У разі присвоювання рядковій змінній виразу з довжиною більшою ніж максимально допустима для даної змінної, символи за межами максимальної довжини не використовуються (вилучаються). Переривання виконання програми у цьому випадку не відбувається.

До основних операцій, визначених над рядковими даними можна віднести: присвоювання (:=); з'єднання (+); порівняння (<,<=,>,>=,=,<>).

Порівняння двох рядків здійснюється зліва направо до перших різних символів. "Більшим" вважається символ, який розташований в алфавіті далі (він має більший номер у таблиці кодів комп'ютера ASCII). Рядки рівні, якщо мають ту саму довжину, і в її межах відповідні символи однакові.

Для того, щоб визначити довжину літерного рядка, введеного користувачем, використовується стандартна функція Length(аргумент), яка визначає точну кількість символів в рядку, назва якого вказана в якості аргумента. Як результат роботи вона повертає довжину рядка. Розглянемо приклад використання цієї функції:

```
Program  String1;
var  S: String ;
begin
   Write ('Введіть рядок: ');
   Readln (S);
  Write ('Довжина введеного рядка: ', Length (S));
  Readln ;
end .
```

У результаті роботи програми на екран буде виведено натуральне число, що дорівнює кількості символів у введеному користувачем довільному рядку (включаючи пробіли).

Доступ до окремих символів рядка

Іноді потрібно перевіряти окремі символи в рядку. Наприклад, перевірити, чи є у введеному рядку пробіли. Щоб реалізувати звернення до певного символу рядка, потрібно вказати ім’я змінної-рядка з номером символу в ній, вказаним в квадратних дужках. Наприклад, для привласнення змінній літерного типу C 1-го символу рядка S; потрібно виконати команду C:= S[1];

Розглянемо наступне завдання: запитати у користувача рядок і перевірити його на наявність пробілів. Як відповідь вивести кількість пробілів. Для реалізації потрібний буде лічильник, який зберігатиме кількість пробілів; тобто потрібно перевіряти за чергою всі символи веденого рядка, визначаючи, чи поточний символ є пробілом.

Для того, щоб перевірити всі символи, потрібно визначити кількість символів в рядку та в циклі провести порівняння. При цьому поточним символом буде значення циклу:

```
Program  String2;
var  C, I: Byte; S: String;
begin
  Write('Введіть рядок:');
  Readln(S);
  C:=0;
  For I:= 1 to Length (S) do
   If S[I]=' ' then C:=C+1;
  Write ('Кількість пропусків:', C);
  Readln;
```

```
end.
```

Використовуючи аналогічний цикл, можна порахуваті кількість слів у заданому реченні.

Основні методи та процедури роботи з рядками

Для обробки даних типу String використовують спеціальні стандартні підпрограми. Розглянемо деякі з них:

1**) function Copy (S: String; Index: Integer; Count: Integer): String;**– створює з наявного рядка підрядок, тобто виділяє якусь частину наявного рядка і повертає його як результат своєї роботи. Результат цей може бути привласнений змінній, використаний у виразі або в процедурі (наприклад, виведений на екран). Параметри цієї функції:

S: String; – задає початковий рядок, тобто той рядок, зякого "вирізається" частина.

Index: Integer; – номер першого символу, починаючи з якого відбуватиметься виділення рядка (якщо цей параметр більше реальної довжини рядка, то функція поверне порожній рядок).

Count: Integer; – кількість символів, які будуть виділені, починаючи з номера, що задається параметром Index: Integer.

Розглянемо роботу функції Copy на конкретному прикладі. Нехай маємо рядок S:= 'informatyka'. Наприклад, потрібно виділити 5 символів, починаючи з третього, тобто отримати рядок: 'forma' Для реалізації потрібно використати функцію Copy з такими парметрами: S1:= Copy(S, 3, 5).

2**) procedure Delete (S: String; Index: Integer; Count: Integer); –** видаляє зрядка певну його частину. Параметри:

S: String; – початковий рядок, в якому відбуватиметься видалення.

Index: Integer; – номер першого символу, що видаляється (якщо це число більше, ніж довжина рядка, символи не видаляються).

Count: Integer; – число символів, що видаляються. Якщо символом в рядку не вистачає, то видаляються ті, що залишилися.

Нехай є рядок S:= 'Це перший рядок'. Потрібно видалити з нього слово 'перший'. Використовуючи процедуру Delete (S, 4, 7); отримуємо: 'Це рядок'

3) procedure Insert (Source: string; S: string; Index: Integer); – вставляє підрядок в рядок. При цьому якщо довжина нового рядка перевищує 255 символів, то зайві символи відкидаються. Розглянемо параметри процедури:

Source: String; – підрядок, який поміщається в рядок.

S: String; – початковий рядок, в який буде вставлено рядок Source.

Index: Integer; – позиція в початковому рядку S, починаючи з якої буде вставлено підрядок. Наприклад, маємо два рядки: S:= 'Це рядок', S1:= 'перший'. Використовуючи процедуру Insert(S1, S, 3), отримаємо S:= 'Це перший рядок';

Розглянемо таку задачу: ввести два рядки. Перевірити: якщо з'єднати перший рядок з другим – чи не перевищить довжина 255 символів? Якщо ні, то вставляємо в 1-й рядок 2-й і виводимо результат на екран:

```
Program  String3 ;
var  S, S1 : String ;
     I: Integer ;
begin
 Write('Введіть 1-й рядок: ');
 Readln(S);
 Write('Введіть 2-й рядок: ');
 Readln(S1);
 If Length(S)+Length(S1)<=255 then
   begin
     Write('Введіть позицію вставки: ');
     Readln(I);
     Insert(S1 , S, I);
     Write(S);
   end
    else  Write('Дуже довгі рядки!');
Readln ;
end .
```

4) function Pos (Substr, S: String): Byte; – призначена для пошуку підрядка в рядку. Вона повертає номер символу, з якого починається шуканий рядок, або нуль, якщо такого підрядка немає. Параметри:

Substr: String; – шуканий рядок, який фукнция намагається знайти.

S: String; – рядок, в якому проводитися пошук.

Наприклад, нехай потрібно ввести два рядки і перевірити, чи міститься в першому рядку другий. Якщо міститься, то вивести його позицію. Програмна реалізація цього завдання може бути такою:

```
Program  String4;
var S, S1: String;
    I: Integer;
begin
  Writeln('Введіть 1-й  рядок: ');
  Readln(S);
  Writeln('Введіть 2-й  рядок: ');
  Readln(S1);
  I:=Pos(S1 ,S);
```

```
    If  I <> 0 then
     Writeln('Такий рядок є! Позиція: ', I)
     else  Writeln('Немає такого рядка!');
    Readln;
  end.
```

Якщо ввести в якості першого рядка символи '12345', а другого – '234', то шукана позиція дорівнюватиме 2.

5) function Concat (S1, S2 ....., Sx: String): String; – це об'єднання декількох рядків в один (абсолютний аналог операції "+"). Параметри:

S1, S2 ....., Sx: String; – список рядків, які будуть складені.

6) procedure Str(X [:width[:decimals]]; st: String) – перетворює число X (real або integer) в рядок символів st. Необовязкові параметри width і decimals (якщо вони присутні) задають формат перетворення: загальну ширину поля для представлення числа X (width) та кількість символів дробової частини (decimals), якщо X: real.

Є два способи опрацювання даних типу string. Можна опрацьовувати весь рядок як єдине ціле, застосовуючи до нього функції та процедури, або розглядати рядок як масив, складений з елементів-символів і опрацьовувати його за правилами роботи з елементами масиву.

Розглянемо таку задачу: нехай задано деякий рядок. Визначити довжину рядка. Вивести на екран друге слово цього рядка.

```
program String5;
var i,k,m,n1,n2: integer;
    S: string
begin
  Writeln('Введіть рядок: ');
  Readln(S);
   m:=0;
  writeln('Довжина рядка',length(S));{визначаємо довжину рядка}
  for i:=1 to length(S) do   {переглядаємо всі символи рядка}
    if S[i]= ' ' then        {шукаємо пропуск}
    begin
     m:=m+1; if m=1 then n1:=i; {визначаємо номери пропусків}
    if m=2 then n2:=i
    end;            {виводимо слово між двома пропусками}
  for i:=n1+1 to n2-1 do write(S[i]);
 readln
 end.
```

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі.

2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.
3. Відкомпілювати, налагодити та запустити програми на виконання.
6. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які давали б змогу отримати проміжні і кінцеві результати.
4. Оформити звіт до лабораторної роботи, в якому обов'язково навести тексти програм та результати їхнього виконання.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Ввести з клавіатури власні прізвище, ім'я та по батькові як одне дане типу рядок. Визначити довжину отриманого рядка. Виконати такі завдання застосовуючи до рядка функції та процедури.

1. Визначити, скільки літер '*a*' є у цьому рядку.
2. Вивести три літери – свої ініціали через крапку.
3. Усі літери 'і' в рядку продублювати та вивести отриманий рядок на екран
4. Вивести на екран власне прізвище та ім'я через пробіл.
5. Визначити довжину прізвища та вивести прізвище на екран.
6. Вилучити усі літери 'а' та 'о' з цього рядка. Вивести отриманий рядок на екран.
7. Визначити кількість голосних літер у цьому рядку.
8. Вивести на екран найдовше слово в рядку.
9. Встановити, чи починається хоч би одне слово в рядку з літери 'і'?
10. Вивести власне ім'я у стовпчик.
11. Вивести ім'я та кількість літер у третьому слові.
12. Вивести на екран тільки ті слова, в яких немає літери "а".
13. Визначити та вивести на екран найкоротше слово рядка.
14. Кожну літеру імені продублювати. Вивести отриманий рядок на екран.

15. Вивести довжину найкоротшого слова в рядку.

Задача 2

1. Заданий рядок символів. Видалити з нього слово “line”, якщо воно є. У протилежному випадку вивести відповідне повідомлення.
2. Ввести з клавіатури деякий рядок. Вилучити з нього пробіли та літери *a*. Вивести отриманий рядок на екран.
3. Заданий деякий символьний рядок. Замінити всі послідовності символів 'on' на 'online' і вивести новий рядок (якщо шуканої послідовності в рядку немає, то вивести відповідне повідомлення).
4. Ввести з клавіатури деякий символьний рядок. Визначити, чи є у цьому рядку два будь-яких однакових символи, і вивести відповідне повідомлення
5. Заданий рядок символів. Впорядкувати його так, щоб слова починалися з літер за абеткою.
6. Ввести з клавіатури декілька назв міст. Вивести на екран назви міст, що містять парну кількість символів.
7. Ввести з клавіатури три прізвища. Визначити, яке з них найдовше. Вивести прізвища за мірою зменшення символів в них, починаючи з найдовшого.
8. Дано символьний рядок і слово, що складається з чотирьох символів. Визначити, чи є у цьому рядку всі літери зазначеного слова.
9. Заданий рядок символів. Переставити його слова в протилежному порядку
10. Ввести з клавіатури два символьні рядки. Вивести на екран слова, які зустрічаються в обох рядках.
11. Нехай задано деякий рядок. Визначити довжину рядка. Вивести на екран перше слово цього рядка
12. Ввести з клавіатури деякий символьний рядок. Впорядкувати його так, щоб слова в рядку розташовувалися за мірою зменшення кількості літер в них (найдовше слово – на першому місці, найкоротше – останнє).

13. Зчитати з клавіатури деякий рядок. Закодувати його кодом Цезаря, замінивши кожну літеру на четверту за нею літеру з алфавіту. Вивести ориманий рядок на екран.
14. Подано рядок, що містить текст. Записати його у зворотному порядку.
15. Нехай задано деякий рядок. Визначити довжину рядка та кількість слів у ньому. Вивести на екран останнє слово цього рядка.

ЛАБОРАТОРНА РОБОТА № 9

Використання записів (record). Оператор приєднання (with). Масиви записів

МЕТА РОБОТИ: вироблення практичних навиків роботи з даними типу record; ознайомлення з особливостями використання записів з варіантою частиною та масивів записів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Досить часто потрібно представити деякі дані як складові частини іншої логічної одиниці. Так, наприклад, інформацію про номер будинку, назву вулиці і місто зручно згрупувати в єдине ціле і назвати адресою, а об'єднану інформацію про день, місяць і рік народження – датою. У мові Pascal для представлення сукупності різнорідних даних використовують комбінований тип запис. Зазвичай запис містить сукупність різнотипних атрибутів, що налжать до одного об'єкта.

Запис – структурований тип даних. Записи є неоднорідними невпорядкованими структурами з прямим доступом до компонентів. Компоненти запису називають полями запису. У одному полі дані мають один і той же тип, а в різних полях можуть мати різні типи. Поля запису можуть мати будь-який тип (окрім файлу); зокрема, самі можуть бути записами. Опис записів виконують двома способами: у розділі типів та змінних або у розділі змінних. Загальний вид опису типу record:

```
type t=record
  n11,n12,…,n1n:type 1 ;
```

```
   ......................
   nk1,nk2,…,nkn:type k ;
end;
```

де *nij* – ідентифікатори полів; *type i* – типи полів. Так, наприклад, інформацію про деякого студента можна описати так:

```
Type  adr=record     {опис запису у розділі типів}
      nom_bud: Integer ;
         vulycia, misto: String[20];
         kraina: String[20];
      end;
      data= record
         month: 1..12;
         day: 1..31;
         year: integer;
 end;
Var  student: Record;            {прямий опис запису}
     pryzv: string[25];
 ocinky: array [1..5] of integer;
 adres: adr;
     end ;
    data_nar, data_vst: data;    {оголошення змінної-запису}
```

Розглянемо детальніше описаний в прикладі тип data і зміну data_nar, що належить цьому типу. Змінна data_nar описана як запис, що складається з трьох полів: month, day і year. Кожне поле містить відповідні дані: ціле число в межах від 1 до 12 (номер місяця), ціле число від 1 до 31 (число), ціле число (рік).

До кожного елемента запису можна звернутися, використовуючи складене ім'я, яке має наступну структуру: `<ім'я змінної>.<ім'я поля>`. Наприклад, щоб записати в data_nar дату 12.01.1985, потрібно виконати такі команди:

```
data_nar.month:=1;  data_nar.day:=12;  data_nar.year:=1985;
```

Якщо поле є своєю чергою записом, то складене ім'я подовжується, наприклад:

```
student.adres.kraina:='Україна';
student.adres.misto:='Львів';
```

Кожне поле запису можна розглядати як звичайну змінну, яку можна надрукувати або використовувати в розрахунках. Водночас, запис може використовуватися як єдине ціле. Окрім дій над окремими полями записів можна виконувати операції над всім записом. Наприклад, наступний оператор привласнення встановлює рівність значень записів `data_nar` і `data_vst`:

```
  data_nar:=data_vst;
```

Це привласнення еквівалентно наступній послідовності операторів:

```
data_nar.Day:=data_vst.Day;
data_nar.Month:=data_vst.Month;
data_nar.Year:=data_vst.Year;
```

Для змінних одного типу можна перевірити виконання відношення рівності або нерівності ("=", "<>").

Масиви записів

Оскільки на тип компонентів масиву не накладається обмежень, то можна використовувати масив, компонентами якого є записи. Приклад опису такого масиву:

```
Var Vidom: Array[l..25] of data;
    Grupa: Array[l..13] of student;
```

Щоб звернутися до деякого поля певного запису масиву, слід вказати ім'я масиву, індекс запису, що цікавить, і ім'я необхідного поля. Наприклад, наступний оператор друкує вміст поля Year запису data[3]:

```
Write(data[3].Year);
```

Для введення/виведення масиву записів використовують цикли. Щоб ввести або вивести запис, треба ввести/вивести відповідні поля, наприклад:

```
Readln(student.pryzv);
For i:=l to 5 do Read(Student.Ocinku[i]);
Writeln(Student.pryzv);
For i:=l to 5 do write(Student.Ocinku[i]);
```

Оператор with

У деяких програмах, що містять велику кількість звернень до одного і того ж поля, необхідність дотримання всіх правил перерахування індексів і імен полів при складанні посилань приводить до одноманітного повторення. Щоб полегшити виконання багатократних посилань на поля структур, використовують оператор With.

Він дає змогу, один раз вказавши ім'я змінної типу "запис" після слова with, працювати в межах одного оператора (простого або складеного) з іменами полів як із звичайними змінними, тобто не писати громіздких складених імен.

Для цього треба визначити область дії конструкції, наприклад:

```
with  <ім'я змінної> do
begin
<оператори> ...
End;
```

У межах оператора, розташованого усередині оператора With, до полів вказаної змінної можна звертатися просто за іменем. Наприклад, для

занесення дати народження у попередньому прикладі досить виконати оператори:

```
with  student.adres  do
begin
  nom_bud:= 5;
  vulycia:='Шевченко';
  misto: ='Львів';
end.;
```

Якщо у програмі використовується декілька записів з однаковими іменами полів, треба бути обережним з вживанням With, щоб не сталося плутанини. Неприпустимим є використання вкладених операторів With, в яких виникає неоднозначність конструкції. Також, потрібно уважно підходити до використання вкладених операторів With, застосування яких може привести до помилок і до втрати наочності структури програми.

Розглянемо програмну реалізацію наступного завдання: у масиві зберігаються дані про студентів: курс, прізвище, ім'я, група. Вивести список студентів, які вчаться на другому курсі.

```
Program record1;
Uses  Crt;
Type  student=record
        kurs: integer;
        Prizv, imia: string[15];
        grupa: string;
      end ;
Var I, n: integer;
    st: array [1..40] of student;
Procedure Poshuk;
Begin
  for i:=1  to n do
    if  st[i].kurs=2
      then  with  st[i] do
        writeln (Prizv:10, imia:10, grupa:6);
End
Begin
  writeln('Введіть кількість учнів');
   read(n);
  for  i:=1  to  n do
    begin
    writeln('Введіть групу та прізвище' i,'студента');
      with  st[i] do
        begin
          readln(grupa,Prizv);
          write ('Введіть курс студента (тільки число)');
          read (kurs);
        end;
    end;
  writeln('Студенти 2-го курсу:');
  Poshuk;
```

```
  Readln;
End.
```

Розглянемо ще один приклад використання записів при реалізації завдань. Відомості про деталі, що зберігаються на складі, містять такі атрибути: назва, кількість, вартість однієї деталі. Вивести інформацію про деталь, сумарна вартість для якої максимальна.

```
program  record2;
type  detal= record
        a:string [30];
        kil,vart: integer;
      end;
var  a: array [0..99] of detal;
     n,i,max: integer;
begin
 write('Кількість деталей? ');
 readln(n);
 for i:=0 to n-1 do
  With a[i] do
   begin
    write('Інформація про ', i ' деталі: ');
    readln(a,kil,vart);
   end ;
 max:= 0;
 for i:= 1 to n-1 do
  if a[max].kil*a[max ].vart<a[i].kil*a[i].vart then  max:=i;
  writeln('Шукана  деталь:',  a[max].a,'вартістю',a[max].vart,
'у кількості', a[max].kil);
end.
```

Записи з із варіантами

При визначенні типу запису в нього можна включати варіантну частину. Це означає, що різні змінні, хоч і належать до одного типу, можуть мати різні структури.

Варіантна частина запису починається вибором case і слідує за загальною частиною; після її закінчення в записі не можуть з'являтися ніякі інші поля, тому case закривається службовим словом end. Будь-який запис може мати тільки одну варіантну частину, яка повинна розміщуватися в кінці запису (після фіксованої частини). Проте, усередині якого-небудь варіанта, своєю чергою, може бути присутньою інша варіантна частина, вкладена в першу.

При записі варіанта (списків елементів) обов'язкова наявність круглих дужок, навіть якщо в них нічого не вкладено. Наприклад, хай необхідно задати інформацію про деяку людину, вказавши прізвище і рік народження, а

також, якщо це чоловік, то повідомити, чи військовозобов'язаний він і яку має спеціальність, а якщо це жінка, то вказати, чи заміжня вона і скільки має дітей.

```
Type  pol= (m,w );
      people=record
         fam:string [20];
         godro:1900 ..2007;
         case  mw:pol  of
           m: (voen: boolean; spec: string[15]);
           w: (merry: boolean; child: byte);
      end;
var   p1, p2: people;
```

Всі імена елементів мають бути різними, навіть якщо вони зустрічаються в різних варіантах. До елементів варіантної частини можна звертатися так само, як до елементів фіксованої частини запису.

```
P1.mw:=m;   p1.voen:=true;   p2.child:=2;
```

У процесі виконання програми в структуру запису включається той варіант, елементам якого у цей момент було привласнено значення. Як тільки якому-небудь елементу іншого варіанта привласнюється деяке значення, у структуру запису залучається цей варіант, а елементи попереднього варіанта стають невизначеними.

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень введених даних, які давали б змогу отримати проміжні і кінцеві результати.

5. Оформити звіт до лабораторної роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Скласти програму для реалізації такого завдання.

1. Сформувати записи, що мають таку структуру:

```
type дата=record число:1..31;
          місяць:1..12;
          рік:1900..1996;
       end ;
          анкета=record
               прізвище:string ;
               стать:(ч, ж);
               дата народження:дата;
           end;
```

Вивести на екран список людей, що народилися в заданому місяці.

2. Сформувати записи, що мають таку структуру :

```
type студент= record
             прізвище: string;
             номер.гр : string;
            оцінка1, оцінка2, оцінка3: integer;
       end;
```

Вивести на екран список людей, що мають за іспит оцінку 2.

3. Сформувати записи, що мають таку структуру:

```
type студент= record
          прізвище, імя:string;
          стать:(ч, ж);
           курс:1..5;
       end;
```

Вивести на екран список студентів чоловічої статі, які вчаться на даному курсі.

4. Сформувати записи, що мають таку структуру:

```
type власник=record
          прізвище:string;
          марка авто, реєстр.номер:string;
          рік випуску:1900..2005;
       end;
```

Вивести на екран прізвища власників і номери автомобілів цієї марки.

5. Сформувати записи, що мають таку структуру:

```
type  книга=record
            автор, назва, видавництво:string;
            кількість сторінок:integer;
            ціна: real;
        end;
```

Вивести на екран назви книг певного автора, виданих в цьому видавництві.

6. Сформувати записи, що мають таку структуру:

```
type пассажир=record
          прізвище, імя:string;
          номер рейсу:string;
          кількість речей, загальна вага:integer;
         end;
```

Вивести на екран інформацію про кількість речей і загальну вагу кожного рейса.

7. Сформувати записи, що мають таку структуру:

```
type спортсмен=record
           прізвище, країна:string;
           ріст:150..220;
           вік, результат:ineger;
        end;
```

Вивести на екран інформацію про спортсменів даного віку з результатом, не гірше данного.

8. Сформувати записи, що мають таку структуру:

```
type  спортсмен=record
            прізвище:string;
            країна, тренер:string;
            вік:ineger;
         end;
```

Вивести на екран інформацію про наймолодшого спортсмена, що займається у цього тренера;

9. Сформувати записи, що мають таку структуру:

```
type предмет=record
          назва:string;
          кількість годин:integer;
          викладач, кафедра:string ;
          форма звітності:(залік, іспит);
       end;
```

Вивести на екран повну інформацію про список предметів, для яких формою звітності є іспит.

10. Сформувати записи, що мають таку структуру:

```
Type  предмет=record
             назва:string;
             кількість годин:integer;
             викладач, кафедра:string;
            форма звітності:(залік, іспит);
          end;
```

Вивести на екран повну інформацію про викладачів цієї кафедри.

11. Сформувати записи, що мають таку структуру:

```
type  предмет=record
            назва:string;
            кількість годин:integer;
            викладач, кафедра:string;
            форма звітності:(залік, іспит);
        end;
```

Вивести на екран повну інформацію про список предметів, що читаються цією кафедрою.

12. Сформувати записи, що мають таку структуру:

```
  type  викладач=record
             прізвище:string;
            предмет, факультет:string;
             кількість годин:integer;
            end;
```

Вивести на екран повну інформацію про список предметів, які читає цей викладач.

13. Сформувати записи, що мають таку структуру

```
 type викладач=record
             прізвище:string;
             предмет, факультет:string;
             кількість годин:integer;
          end;
```

Вивести на екран повну інформацію про викладача, що має найбільше навантаження

14.Сформувати записи, що мають таку структуру:

```
type предмет=record
            назва:string;
            викладач, кафедра:string;
            форма звітності:(залік, іспит);
         end;
```

Вивести на екран повну інформацію про список предметів, для яких формою звітності є залік.

15. Сформувати записи, що мають такуструктуру:

```
type викладач=record
              прізвище:string;
              предмет, факультет, кафедра:string;
              кількість годин:integer;
          end;
```

Вивести на екран повну інформацію про список предметів, які читаються на цій кафедрі.

Задача 2.

1. Створити масив учнів класу і вивести на екран прізвища та імена у вигляді таблиці.
2. Відомі дані про 6 співробітників фірми: прізвище, вік, і відношення до військової служби. Вивести на екран прізвища всіх військовозобов'язаних співробітників.
3. Дано назви 6 міст і країн, серед них є міста, що знаходяться в Італії. Вивести на екран їх назву.
4. Дано назви 10 країн і частин світу. Надрукувати на екрані всі країни, що знаходяться в Азії.
5. Із зведеної відомості трьох студентів (порядковий номер, ініціали і 5 оцінок). Визначити кількість відмінників.
6. З відомості 5 студентів з їхніми оцінками (порядковий номер, ініціали і три оцінки) визначити середній бал кожного студента.
7. Необхідно заповнити відомості про 5 студентів (прізвище, дата народження, адреса, курс і група), а потім вивести ці відомості на екран.
8. Необхідно заповнити відомості про 6 студентів деякого факультету. Вивести з загального списку прізвища студентів 2-го курсу.
9. Відомі дані про 6 співробітників фірми: прізвище, імя, вік, і відношення до військової служби. Вивести на екран прізвища всіх співробітників, вік яких більше 35 років.
10. Дано назви 7 річок із зазначенням областей і країн, в яких вони перебувають. Вивести на екран назви річок, які є на Україні.
11. Відомі дані про 8 студентів деякої групи: прізвище, імя, вік, і середній бал. Вивести на екран прізвища всіх студентів, средній бал яких більше за 4,5.
12. Дано відомості про 10 студентів деякого факультету. Вивести з загального списку прізвища студентів, які вчаться в одній групі.

ЛАБОРАТОРНА РОБОТА № 10

Використання файлів даних (file). Загальні засоби для роботи з різними типами файлів (текстові, типізовані та нетипізовані файли)

МЕТА РОБОТИ: ознайомлення з особливостями використання типізованих, текстових та не типізованих файлів та вироблення практичних навиків роботи з файловими типами даних.

ТЕОРЕТИЧНІ ВІДОМОСТІ

У практиці програмування часто зустрічаються завдання, що розв’язуються із застосуванням малозмінних у часі даних. До них належать, наприклад, завдання бухгалтерського обліку, оптимального планування та інші. Введення даних з клавіатури при кожному розрахунку є нераціональним або просто неможливим. Саме для цього в мовах програмування реалізована концепція файлів, що дає змогу після створення набору інформації запам'ятати її на зовнішньому носії і звертатися до неї безпосередньо з оброблювальних програм за кожної потреби.

Файл – це впорядкована послідовність однотипних компонентів, розташованих на зовнішньому носії. Файли призначені тільки для зберігання інформації, а її обробка здійснюється програмами. Файл можна також вважати структурованим типом даних, що складається з послідовності компонент одного і того ж типу однакової довжини і структури. Досить часто компонентами файлу є записи. Використання файлів доцільне у випадках:

- довготривалого зберігання даних;
- доступу різних програм до одних і тих же даних;
- обробки великих масивів даних, які неможливо розмістити в оперативній пам'яті комп'ютера.

Файли бувають стандартними та створеними користувачем. Кожен файл має своє ім'я, яке зареєстроване у відповідній директорії. Введення і виведення даних здійснюється через буфер – область у пам'яті, що

виділяється для кожного файлу. Механізм буферизації дає змогу більш швидко й ефективно обмінюватися інформацією з зовнішніми пристроями.

У мові Pascal визначені три типи файлів: текстові файли, типізовані та нетипізовані. Всі файли мають бути описані в програмі або в розділі змінних Var, або в розділі типів Type. Змінну файлового типу описують одним з трьох способів:

`file of <базовий тип>` – типізований (вказаний тип компоненти);

`file of text` – текстовий файл;

`file of file` – нетипізований файл.

Новий файл створюється шляхом додавання записів в кінець порожнього файлу (файлу, що не містить жодного елементу). Довжина файлу, тобто кількість елементів, при визначенні файлу не задається. Розглянемо приклади опису файлових змінних:

```
var
   f1: file of char;
   f2: file of integer;
   f3: file;
```

Файлові змінні, описані в програмі, називають логічними файлами. Всі основні процедури і функції, що забезпечують введення-виведення даних, працюють тільки з логічними файлами. Фізичний файл повинен бути зв'язаний з логічним до виконання процедур відкриття файлів.

Ім'я файлу має бути унікальним і таким, що складається з власного імені і необов'язкового розширення – типу файлу, що містить три символи і відокремленого від основного імені крапкою. Розширення, як правило, вказує в мнемонічній формі на вміст файлу: pas, exe, txt тощо. У деяких випадках тип файлу привласнюється автоматично операційною системою або використовуваним пакетом.

Під читанням файлу розуміють введення даних із зовнішнього файлу, що перебуває на диску, в оперативну пам'ять машини. Запис у файл – виведення результатів роботи програми з оперативної пам'яті на диск у файл.

Робота з файлами виконується за допомогою таких процедур і функцій:

Assign(f, FileName) – пов'язує файлову змінну f з фізичним файлом, повнеім'я якого задане в рядку FileName. Встановлений зв'язок діятиме до кінця роботи програми, або доки не буде зроблено перепризначення. Після зв'язку файлової змінної з дисковим ім'ям файлу в програмі потрібно вказати

напрям передачі даних (відкрити файл). Залежно від цього напряму говорять про читання з файлу або запису у файл.

Reset(f) – відкриває для читання файл, з яким пов'язана файлова змінна f. Якщо вказаний файл не знайдений, процедура завершується з повідомленням про помилку.

Rewrite(f) – відкриває для запису файл, з яким пов'язана файлова змінна f. Якщо вказаний файл вже існував, то всі дані з нього знищуються.

Close(f) – закриває відкритий до цього файл з файловою змінною f. Виклик процедури Close необхідний при завершенні роботи з файлом. Якщо з якоїсь причини процедура Close не буде виконана, файл буде створений на зовнішньому пристрої, однак вміст останнього буфера не буде перенесений в нього.

EOF(f): boolean – повертає значення True, коли при читанні досягнутий кінець файлу. Це означає, що вже прочитаний останній елемент у файлі або файл після відкриття опинився порожній.

Rename(f, NewName) – дає змогу перейменувати фізичний файл на диску, пов'язаний з файловою змінною f. Перейменування можливе після закриття файлу.

Erase(f) – знищує фізичний файл на диску, який був пов'язаний з файловою змінною f. Файл до моменту виклику процедури Erase має бути закритий.

IOResult – використовується для пошуку помилок, які виникають при роботі з файлами. Повертає ціле число, відповідне коду останньої помилки введення-виведення. При нормальному завершенні операції функція поверне значення 0. Значення функції IOResult необхідно привласнювати якій-небудь змінній, оскільки при кожному виклику функція обнуляє своє значення.

Типізовані файли

Типізовані файли – це файли, що складаються з нумерованої послідовності об'єктів (записів) будь-якого типу (окрім типу "файл"). З такими файлами можна працювати в режимі прямого доступу, при якому виконується безпосереднє звертання до будь-якого запису файлу. Доступ до компонентів файлу здійснюється за їхніми порядковими номерами. Компоненти нумеруються, починаючи з нуля. Після відкриття файлу

покажчик (номер поточної компоненти) стоїть на його початку (на нульовому компоненті). Кожне читання або запис призводить до зрушення покажчика до наступного компонента.

Розглянемо основні процедури і функції обробки типізованих файлів:

Write(f, список змінних) – процедура записує у файл f всю інформацію із списку змінних.

Read(f, список змінних) – процедура читає з файлу f компоненти у вказані змінні. Типи файлових компонент і змінних повинні збігатися. Якщо буде зроблена спроба читання неіснуючих компонент, то відбудеться помилкове завершення програми.

Seek(f, n) – процедура зміщує покажчик файлу f на n-ту позицію, забезпечуючи прямий доступ до потрібного елемента.

FileSize(f): longint – функція повертає кількість компонент у файлі f.

FilePos(f): longint – функція повертає порядковий номер поточного компоненту файлу f.

Truncate(f) – процедура відсікає кінець файлу, починаючи з поточної позиції включно.

Для додавання записів в кінець файлу використовуються процедури:

```
Readln(a);
Seek(f, filesize(f));
Write(f,a);
```

Покажчик встановлюється за кінець файлу, оскільки нумерація записів починається з нуля. Після чого за допомогою Write можна додавати записи. Відкривати файл можна тільки процедурою Reset (f).

В якості прикладу роботи з файловими змінними розглянемо програмну реалізацію такого завдання: створити файл записів, що містить деяку інформацію про комп’ютери та вивести вміст файлу на екран.

```
Program File1;
Type comp = record   {тип запису, що містить х-ки комп’ютера}
       Typ: string[15];
      Hdd, ram: real;
     End;
Filecomp=file of comp;
Var f1: filecomp ;
    i, n: integer ;
    c1: comp;
begin
writeln(‘введіть кількість комп’ютерів‘);
readln(n);
```

```
  assign(f1,'file1'); {пов'язати з файлом file1 змінну f1}
    rewrite(f1);      {відкриваємо файл для запису даних}
    for i:=1 to n do
      begin
       writeln('введіть марку комп'ютера');
       readln(c1.typ);    {записати пропозиції у файл}
       writeln('введіть обсяги hdd та ram');
       readln(c1.hdd, c1.ram);
      end;
    close(f1 );       { закрити файл для запису }
  writeln(' тип процесора   обсяг вінчестера      ram  ' );
  reset(f1);        {відкриваємо файл для зчитування даних}
  for i:=1 to n do   {виводимо вміст файлу на екран}
    begin
     read(f1,c1); {зчитуємо дані з файлу}
     writeln(c1.typ,c1.hdd,c1.ram);
    end;   readln
 end.
```

Отже, для створення файлу послідовного доступу необхідно:

- оголосити файлову змінну;
- «прив'язати» файл до фізичного носія інформації (привласнити файлу ім'я) (Assign);
- Відкрити новий файл (Rewrite);
- підготувати інформацію для введення в компоненту файлу, тобто сформувати запис для введення у файл в якості компоненти;
- записати у файл компоненту (Write);
- повторити пункти 4 і 5 необхідну кількість разів;
- закрити створений файл (Close).

Для доступу до компонентів послідовного файлу (наприклад, для перегляду вмісту файлу на екрані або для обробки компонент в цілях залучення їх як фрагменти в програму обробки яких-небудь даних) потрібно:

- привласнити файлу ім'я (Assign);
- відкрити вже існуючий файл (Reset);
- вважати поточну компоненту з файлу в робоче місце пам'яті (як правило, типу запис із структурою компоненти) (Read);
- виконати обробку інформації (наприклад, вивести на екран поля запису);
- закрити файл (Close).

Додавання нових записів у файл послідовного доступу виконується шляхом запису компонент в кінень файлу. Маркер кінця файлу

переноситься, на фізичному носієві при цьому має бути вільний простір. Для запису нових компонент в кінці наявного файлу необхідно:

- привласнити файлу ім'я (Assign);
- відкрити вже наявний файл (Reset);
- встановити покажчик файлу за останньою компонентою (Seek(FV, File sise(FV)):
- створити у спеціально виділеній області пам'яті нову компоненту;
- записати нову компоненту у файл (write);
- закрити файл (Close).

Текстові файли

Текстові файли – файли на диску, що складаються з символів ASCII. Текстові файли є файлами з послідовним доступом. У будь-який момент часу доступний тільки один запис файлу. Інші записи стають доступними лише в результаті послідовного просування по файлу. Текстові файли внутрішньо розділені на рядки, довжини яких різні. Для розділення рядків використовується спеціальний маркер кінця рядка.

Для роботи з текстовими файлами використовуються згадані вище функції і процедури assign, rewrite, reset, close, read, readln, write, writeln, seekeof, проте єі специфічні функції і процедури:

Append(fv) – відкрити текстовий файл і встановити покажчик на маркер кінця файлу. За допомогою цієї процедури у текстовий файл можна дописувати дані.

Eoln(fv) – повернути true, якщо покажчик файлу досяг маркера кінця рядка, інакше, повернути значення false.

Seekeoln(fv) – функція, аналогічна попередній, але покажчик проходить всі пропуски і знаки табуляції, повертає true досягши маркера кінця рядка.

Seekeof(fv) – повертає true, якщо покажчик файлу розміщений на маркері кінця файлу.

Організація текстового файлу проводиться за схемою:

- оголошується файлова змінна текстового типу;
- привласнюється файлу ім'я (assign);
- відкривається файл (rewrite);
- готується рядок – компонента для запису у файл;

- записується рядок – компонента у файл (writeln);
- повторюються пункти 4 та 5 потрібну кількість разів;
- закривається файл.

Роботу з тектовими файлами розглянемо на наступному прикладі: створити текстовий файл, в який записати три пропозиції. Прочитати цей файл, вивести його вміст на екран. Визначити довжину кожного речення.

```
 Program  File2;
 Var f1: text;
     st: string; n: byte;
 begin
 assign(f1,'file1.txt'); {пов'язати з файлом file1.txt  файлову змінну f1}
 rewrite(f1);   {створити новий файл  з ім'ям file1.txt }
 writeln(f1, 'Дуже корисно вивчати');   {записати пропозиції у файл}
 writeln(f1, 'всім  студентам ');
 writeln(f1, 'мови програмування');
 close(f1);        {закрити файл для запису}
 reset(f1);        {відкрити файл для читання}
 while not  eof(f1) do  {поки не кінець файлу  f1}
    begin
    readln(f1, st);   {читаємо рядок з файлу f1}
    writeln(st);
    n:=length(st);     {визначаємо довжину рядка}
    writeln('довжина =',n);
    end;
 close(f1);     {закрити файл  для читання}
 end.
```

Робота з нетипізованими файлами

Нетипізовані файли – це послідовність компонент довільного типу; дають змогу записувати на диск довільні ділянки пам’яті ЕОМ і зчитувати їх з диска в пам'ять. Основні процедури та функції для роботи з ними:

Rewrite(f, BufSize) – відкриття файлу, що не типізується. Параметр BufSize задає число байтів, що прочитуються з файлу або записуються в нього за одне звернення. Якщо BufSize не вказаний, то за замовчуванням він приймається рівним 128.

BlockRead(f, X, Count, QuantBlock) – зчитування даних з файлу, що не типізується. Ця процедура здійснює за одне звернення читання в змінну X кількості блоків, задане параметром Count, при цьому довжина блоку дорівнює довжині буфера. Значення Count не може бути менше 1.

Необов'язковий параметр QuantBlock повертає число блоків, прочитаних поточною операцією BlockRead.

BlockWrite(f, X, Count, QuantBlock) – запис даних у файл, що не типізується. Ця процедура здійснює за одне звернення запис із змінної X кількості блоків, задане параметром Count, при цьому довжина блоку дорівнює довжині буфера. Необов'язковий параметр QuantBlock повертає число блоків, записаних успішно поточною операцією BlockWrite.

Для файлів, що не типізуються, можна використовувати процедури Seek, FIlePos і FileSize, аналогічно відповідним процедурам типізованих файлів

Розглянемо прилад роботи з файлами цього типу. Нехай маємо текстовий файл Id.Dat, що містить числові значення дійсного типу по два числа у кожному рядку – значення аргументу і функції відповідно. Кількість пар чисел не більш 50. Розглянемо програму, що читає файл, значення аргумента і функції записує в одномірні масиви, підраховує їхню кількість, виводить на екран дисплея і записує у типізований файл Rd.Dat.

```
Program File3;
  var rArg, rF: Array[1..50] of  Real;
       inf: Text;
      outf: file of Real;
       n,I: Integer:
begin
 Assign(inf, 'ID.DAT');  {пов'язати з файлом ID.DAT змінну inf}
 Assign(outf, 'RD.DAT'); {пов'язати з файлом RD.DA змінну outf}
    Reset(inf);      {відкрити файл для читання }
 Rewrite(outf); {відкриваємо файл для запису даних}
 n:= 0
while not EOF(inf) do
    begin
      n:=n+1;
      Readln(inf, rArg[n],rF[n]);
    end;
for I:=1 to n do
   begin
      Writeln(I:2, rArg[ I ]:8:2 rF[I]:8:2);
      Write(outf, rArg[I], rF[I]);
   end;
close(outf);      {закрити файл  для читання}
end.
```

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв'язку задачі.

2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які давали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до лабораторної роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Скласти програму для реалізації такого завдання. Створити файл, помістити туди 15 елементів дійсного типу. Зчитати їх із файлу і виконати завдання, що відповідає номеру Вашого варіанта:

1. Вивести на екран список додатніх елементів. Якщо вказаних елементів немає, вивести відповідне повідомлення.
2. Вивести кількість додатніх елементів. Якщо вказаних елементів немає, вивести відповідне повідомлення.
3. Вивести кількість елементів, котрі більші 15. Якщо вказаних елементів немає, вивести відповідне повідомлення.
4. Вивести на екран максимальний елемент файлу;
5. Вивести всі додатні елементи. Якщо вказаних елементів немає, вивести відповідне повідомлення.
6. Вивести на екран суму всіх елементів файлу.
7. Вивести на екран добуток додатніх елементів. Якщо вказаних елементів немає, вивести відповідне повідомлення.

8. Вивести суму елементів, більших за 6. Якщо вказаних елементів немає, вивести відповідне повідомлення.
9. Вивести на екран перший і восьмий елемент.
10. Вивести всі елементи файлу на екран.
11. Вивести суму квадратів додатніх елементів. Якщо вказаних елементів немає, вивести відповідне повідомлення.
12. Вивести різницю першого і останнього елемента;
13. Вивести на екран квадрат десятого елемента.
14. Вивести всі елементи, які більші від 15. Якщо вказаних елементів немає, вивести відповідне повідомлення.
15. Вивести всі елементи, які за модулем більші за 14. Якщо вказаних елементів немає, вивести відповідне повідомлення.

Задача 2

Засобами текстового редактора створити файл, що містить інформацію про особисті дані 10 студентів. Скласти програму, використовуючи тип даних text для виведення вмісту даних на екран та пошуку даних у файлі. Критерій пошуку обрати самостійно.

ЛАБОРАТОРНА РОБОТА № 11

Множини (set). Операції над множинами. Пошук даних у множині

МЕТА РОБОТИ: ознайомлення з особливостями використання множиного типу даних та вироблення практичних навичків роботи з множинами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Опис зміних множиного типу

Множини – це набори однотипних логічно зв'язаних один з одним об'єктів, що належать деякому базовому типу. Характер зв'язків між об'єктами не контролюється середовищем Turbo Pascal. Кількість елементів,

що належать множині, може мінятися в межах від 0 до 256. Саме цим множини відрізняються від масивів і записів.

Приналежність змінних до множинного типу може бути визначена так:

```
Type
 <ім'я типу >= set of <ім'я базового типу>;
Var
 <ідентифікатор,... >:<ім'я типу>;
```

або безпосередньо розділі опису змінних:

```
Var
  <ім'я змінної множини>: set of <тип елементів>;
```

Як базові типи можуть використовуватися перелічувані типи даних, символьний і байтовий типи або діапазонні типи на їхній основі.

Наприклад:

```
type
   Colour= (white, blue, red, green);
   ColourSet= set of Colour;
   CharSet= set of 'A'..'Z'
Var
   C1: ColourSet;
   kod: CharSet ;
   day: set of 0..30
```

У цьому випадку значенням змінної day може бути будь-яка цифра від 0 до 30, значенням змінної kod будь-яка літера від A до Z. В описі можна вказувати не лише межі значень, а також їхній перелік:

```
M3: set of 'A', 'R', 'C';
```

Константи множинного типу записуються у вигляді взятої в квадратні дужки послідовності елементів або інтервалів базового типу, розділених комами, наприклад: ['A', 'C'] – символи латинського алфавіту; [1, 3, 5, 7, 9] – множина непарних чисел; [ ] – порожня множина (множина, що не містить жодного елемента). Порядок перераховування елементів базового типу в константах не має значення.

Множина охоплює набір елементів базового типу, всі підмножини цієї множини, а також порожню підмножину. Якщо базовий тип, на якому будується множина, має K елементів, то число підмножин, що належать до неї, рівне 2K . Порожня множина є елементом всіх типів множин.

Нехай є змінна P інтервального типу:

```
Var   P: 1..3;
```

Ця змінна може набувати три різні значення – 1 або 2, або 3. Змінна T множинного типу

```
var T: Set of 1..3;
```
може набувати вісім різних значень:

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

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

Для присвоювання імені множини змінних значень використовують конструкцію виду: `<ім'я змінної множини>:= <список>;`

Дві множини вважаються еквівалентними тоді і тільки тоді, коли всі їхні елементи однакові (порядок входження елементів в множині не має значення). Якщо всі елементи однієї множини належать також і в іншу, кажуть про включення першої множини в другу. Порожня множина включається у будь-яку іншу.

Операції над множинами

Всі операції, визначені над множинами, можна поділити на теоретико-множинні операції та операції відношень. До т**еоретико-множинних** операцій належать "+" (об'єднання), "*" (перетин), "–" (різниця). Розглянемо ці операції детальніше. Нехай маємо дві множини A:= ['A', 'B'], B:= ['A', 'K'].

Об'єднанням двох множин є множина, що містить всі елементи і першої і другої множини:

```
C:=A+B;                {C=['A', 'B', 'K']}
```

Перетином двох множин є множина, що формується з елементів, які одночасно належать двом множинам:

```
S:=A*B;                {S=['A']}
```

Різницею множин є множина, що формується з елементів першої множини, які не належать до другої множини:

```
R:=A-B;                {R=['B']}
```

До **операцій відношень** належать: належність елемента множині; рівність, нерівність; належність множини множині.

Визначити чи належить елемент множині можна за допомогою службового слова in. Першим операндом, розміщеним зліва від слова in, є вираз базового типу (тобто типу, якому повинні належати всі члени множини). Другий операнд, який перебуває справа від in, повинен мати множинний тип. Наприклад:

```
Red in [red, white, blue, green];     {результат True}
```

```
8 in [0..3, 6, 9]    {результат False}
```

Зазначену операцію зручно використовувати для спрощення перевірок. Наприклад, умова:

```
if (ch = 'a') or (ch = 'b') or (ch = 'k') or (ch = 'y') then S;
```

значно спрощується завдяки використанню цієї операції:

```
if ch in ['a', 'b', 'x', 'y'] then S;
```

Також над множинами визначені операції перевірки на рівність (“=”) або нерівність (“<>”) двох множин та належності одної множини іншій (“<=” – перевірка належності множини лівого операнда правому; “=>” - перевірка належності множини правого операнда лівому). Результатом операцій в усіх випадках є True або False. Наприклад:

```
3 in [2, 3, 4, 6];          {результат True}
[2, 4] <= [2, 3, 4, 6];     {результат True}
[ ] = ['A', 'B'];           {результат False}
```

Особливість введення та виведення множини

Ввести та вивести множини за допомогою операторів read та write неможливо. Послідовне введення елементів множини можна виконати за допомогою циклічного оператора з передумовою. Внутрішнім оператором циклу є конструктор, який формує множину. Нариклад, формуємо множину з елементів, введених з клавіатури. Нехай ознакою закінчення формування буде крапка.

```
MN:=[ ];
M:= ' ';  {можна присвоювати будь-який символ}
While M <> '.' do
Begin
  Writeln('ввести елемент множини');
  Read(M);
  MN:= MN+[M];
End;
```

Виведення елементів множини можна виконувати шляхом перевірки наявності елемента в множині і його виводу, якщо він міститься в множині. Нехай потрібно вивести на екран елементи множини

```
C: set of 1..30;
```

тоді фрагмент циклічного перебору має вигляд:

```
For i:= 1 to 30 do
If i in C then Writeln(i);
```

Розглянемо наступну задачу: описати множину M[1..15]; зробити її порожньою, після чого заповнити множину 10 елементами шляхом введення елементів з клавіатури. Алгоритм виконання цього завдання є наступним.

1. У розділі опису змінних опишемо множину цілих чисел від 1 до 15. Змінну X цілого типу використовуватимемо для зчитування числа-кандидата в множину, цілу змінну *i* використовуємо для підрахунку кількості введених чисел.

2. Робимо множину M порожньою, застосувавши операцію ініціалізації множини M:=[ ].

3. У циклі Repeat організовуємо заповнення множини елементами. Операцію додавання елемента в множину запишемо оператором привласнення M:=m+[X]. Перед додаванням елементу виконаємо перевірку на допустимість (0 < X < 50).

4. При виконанні умови X in M виведемо повідомлення про те, що число X вже належить множині.

Програмна реалізація цього завдання може бути такою:

```
 Program Set1;
 Var   M: set of 1..50;
   X, i: integer;
Begin
 M:=[ ]; i:=1;
 repeat
 write('Введіть ',i,'-й елемент множини');
 readln(X);
  if (X in M)
   then writeln(X, ' вже міститься в множині')
    else
     if (X<1) or (X>50)
      then
        writeln('Введено недопустиме значення ', X)
       else
         begin
           writeln(X, 'поміщено в множину');
           M:= M+[X]; i:= i+1;
         end;
  until i>10;
End.
```

При використанні у програмах даних множинного типу виконання операцій відбувається над бітовими рядками даних. Кожному значенню множинного типу в пам'яті ЕОМ відповідає один двійковий розряд. Якщо множина містить деякий елемент, в “відповідальному” за нього біті зберігається 1, якщо не містить – зберігається 0. Наприклад, множина

['A','B','C','D'] представлена в пам'яті ЕОМ бітовим рядком 1 1 1 1. Підмножини цієї множини представлені рядками:

```
['A','B','D']          1 1 0 1
['B','C']              0 1 1 0
['D']                  0 0 0 1
```

Процедури Include та Exclude

Додатково до розглянутих операцій можна використовувати такі процедури:

- **Include (S,I)** – залучає новий елемент I в множину S. Тип елементу I має відповідати типу елементів множини.
- **Exclude(S,I)** – вилучає елемент I з множини S. Параметри звернення такі ж, як у процедури Include.

Зазначені процедури оптимізовані для роботи з одиночними елементами множини, тому відрізняються високою швидкістю виконання.

Розглянемо в якості прикладу роботи з множинами таку задачу: виділення з перших N натуральних чисел всіх простих чисел.

```
Program Set3;
Var   M: set of byte;
     i, k, N: Integer;
Begin
 Writeln('Введіть розмір проміжка (до 255)');
 Readln(N);
 M := [2..N];
 For k := 2 to N div 2 do
   For i := 2 to N do
     If (i mod k=0) and (i<>k)
       Then
         M := M-[i];
 For i := 1 to N do
   If i in M
     Then
       Write(i:3);
 Readln;
 End.
```

Розглянутий алгоритм можна спростити, якщо брати до уваги такі факти:

- якщо з множини M видалені всі елементи, що діляться на 2, то не потрібно перевіряти, чи діляться числа, що залишилися, на 4, 6, 8, 10, і так далі;
- коли програма перевіряє подільність, наприклад, на 50, то вона перевіряє і числа до 50, що не має сенсу.

Розглянуту програму не можна використовувати для довільного N, оскільки в будь-якій множині не може бути більше 256 елементів.

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв'язку задачі
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. Якщо в умові немає конкретних даних, то їх треба задати на власний розсуд, керуючись змістом задачі.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати. Розробка контрольного прикладу полягає у підборі таких значень вхідних даних, які дали б змогу отримати проміжні і кінцеві результати.
5. Оформити звіт до цієї роботи, в якому обов'язково навести алгоритми розв'язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Скласти програму для реалізації наступного завдання. Сформувати довільну множину, що складається з N+5 елементів певного типу, де N – номер Вашого варіанту. Утворити для цієї множини три довільні підмножини (всі елементи множин вводити з клавіатури). Вивести на екран елементи, які є у всіх підмножинах одночасно та елементи, які не містить жодна підмножина. За умови відсутності вказаних елементів, вивести відповідне повідомлення. Визначити та вивести на екран елементи, які:

1) містяться у другій підмножині але не містяться в третій підмножині.
2) належать першій і другій підмножині одночасно.

3) містяться в другій підмножині, але не містяться у першій підмножині.
4) містяться в другій і в третій підмножині одночасно.
5) є в головній множині і не належать першій підмножині.
6) містяться в першій і другій підмножині, але не містяться у третій.
7) містяться в другій підмножині, але не містяться в третій.
8) Належать до першої і третьої підмножини, але не містяться у другій.
9) є в головній множині і не належать до другої підмножини.
10) містяться в першій підмножині, але не містяться у третій.
11) є в головній множині і не входять в третю підмножину.
12) містяться в другій і третій підмножині, але не містяться в першій.
13) належать до першої підмножини, але не містяться в другій і третій.
14) містяться в третій підмножині, але не належать до другої.
15) містяться в третій підмножині, але не належать до першої.

Задача 2

Ввести символьний рядок. Для кожного символа визначити, чи він є літерою латинського алфавіту, цифрою або іншим знаком. Результати перевірки вивести на екран.

ЛАБОРАТОРНА РОБОТА № 12

Робота з графікою в середовищі Turbo Pascal. Основні процедури та функції модуля для графічних побудов.

МЕТА РОБОТИ: вироблення практичних навиків роботи з процедурами та функціями модуля Graph (виведення точок, ліній, замкнутих фігур, робота з текстом та фрагментами зображень).

ТЕОРЕТИЧНІ ВІДОМОСТІ

Ініціалізація графічного режиму

До складу Turbo Pascal належить бібліотека графічних підпрограм, яка забезпечує керування режимами різних адаптерів дисплеїв. Вона містить 80

графічних процедур та функцій, а також стандартних констант і описів типів даних.

Для виконання графічних дій потрібно ініціалізувати графічний режим роботи дисплейного адаптера. Настройка графічних процедур на роботу з графічним адаптером досягається методом підключення відповідного графічного драйвера.

Графічні можливості адаптерів визначаються загальною кількістю пікселів (роздільна здатність) та кількістю кольорів (відтінків) кожного пікселя. Графічний екран дисплея складається з точок, які називаються *пікселями.* Кількість точок на екрані може буті різною; положення кожної точки зображення задане координатами (x,y). Координати – цілі числа, які задають номери колонки і рядка растру і не залежать від фізичного розміру екрана. Осі координат напрямлені так: горизонтальна вісь X напрямлена зліва направо; вертикальна вісь Y напрямлена згори вниз; верхній лівий кут має координати (0,0).

Керування графічним режимом забезпечується модулем Graph.tpu. У цьому модулі зібрані стандартні графічні процедури та функції, які використовуються для графічних побудов. Модуль Graph містить також широкий набір процедур та функцій роботи з відеорежимами.

Для ініціалізації графічного режиму використовують процедура *InitGraph;* завершує графічний режим процедура *CloseGraph.* Структура графічної Pascal-програми практично не відрізняється від стандартної структури (рис.1). Загальний вигляд програми з використанням Graph має такий вигляд:

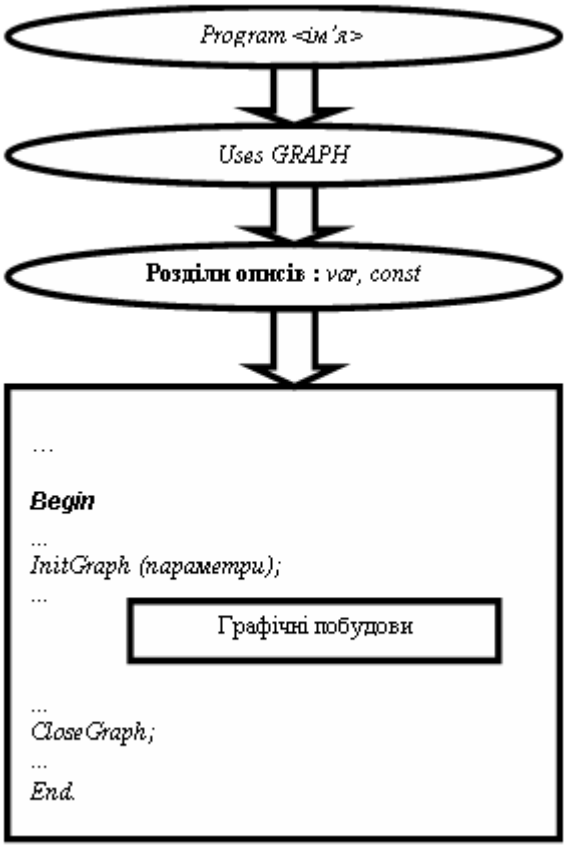


Рис.1.

```
Program Graph1;
Uses Graph;
var
```

```
  Driver, Mode: integer;
 begin
  Driver:=Detect; {визначення номера встановленого в системі драйвера}
  initGraph(Driver, Mode, ' '); {ініціалізація графічного режиму}
 …
 {графічні дії}
 ….
 CloseGraph;
 end.
```

У модулі *Graph* реалізований механізм визначення помилок та видачі повідомлень про них на екран за допомогою функцій **GraphResult** і **GraphErrorMsg**.

Функція **GraphResult:integer** повертає 0, якщо остання графічна операція виконалася без помилок, або число в діапазоні –14.. –1 при наявності помилок. Наприклад:

- ✓ GrOk =0; {немає помилок}
- ✓ GrInitGraph = –1;{не ініціалізований графічний режим}
- ✓ GrNotDetect = –2; {не визначений тип драйвера}
- ✓ GrFileNotFind = –3; {не знайдений графічний драйвер}
- ✓ GrInvalidDriver = –4; {неправильний тип драйвера}
- ✓ GrNoLoadMem = –5; {немає пам'яті для розміщення драйвера}
- ✓ GrNoScanMem = –6; {немає пам'яті для проглядання областей}
- ✓ GrNoFloodMem = –7. {немає пам'яті для зафарбовування областей}

Функція **GraphErrorMsg(Code: integer): string** повертає значення типу string, в якому відповідному коду помилки надається текстове повідомлення. Code – код помилки, який повертається функцією *GraphResult.*

Тому запропоновану програму можна модифікувати, залучивши наступне розгалуження:

```
  InitGraph(Driver, Mode, ' '); {ініціалізація графічного режиму}
 if graphresult= 0 then
 begin
   <текст конкретної  програми з графічними командами>
   CloseGraph;
 end;
  else writeln ('відбулася помилка при ініціалізації графіки');
```

що дає змогу за потреби вивести користувачу відповідне повідомлення.

Дві лапки в Initgraph мають сенс лише тоді, коли графічна програма перебуває у тому ж каталозі, що і бібліотека Graph. Інакше в лапках повинен бути зазначений шлях – вказівка, де розташована бібліотека Graph.

Процедури і функції бібліотеки Graph

Функції:

- **GetMaxX** та **GetMaxY** повертають значення розміру екрана уздовж осі OX та осі OY;
- **Getcolor** – повертає значення поточного кольору;
- **Getcolor(x,y)** – повертає значення кольору точки (x,y);
- **GetX** и **GetY** повертають поточні координати покажчика;
- **ImageSize(X1,Y1,X2,Y2)** – повертає розмір пам'яті в байтах, необхідний для розміщення прямокутного фрагмента зображення (наприклад, вікна з малюнком).

Процедури:

- **InitGraph (Driver, Mode, ' ')** – задає графічний режим. У лапках потрібно зазначити шлях до драйвера, якщо він пербуває в іншому каталозі, ніж файл turbo.exe.
- **SetColor (Color)** – встановлює поточний колір для ліній і символів, що виводяться.
- **Setbkcolor (Color)** – встановлює колір фону вікна. Кольори задаються числами від 1 до 15 або відповідними англійськими назвами.
- **SetFillStyle (Fill, Color)** – встановлює стиль Fill і колір заповнення Color. Значення Fill задається числами від 0 до 12 і визначає стиль заповнення (наприклад 0 – заповнення фоном; 1 – суцільне заповнення і т.і.).
- **SetViewPort (X1, Y1, X2, Y2, Clipon)** – встановлює прямокутне вікно на графічному екрані, X1, Y1 – координати лівого верхнього кута; X2, Y2 – координати нижнього правого кута вікна. Якщо вираз Clipon має значення true і елементи зображення не поміщаються у вікні, то вони відсікаються, якщо – false, то відсікання ігнорується.
- **MoveTo (X, Y)** – встановлює нове поточне положення покажчика.
- **MoveRel (Ox, Dy)** – встановлює зміни нових координат покажчика відносно старих.

- **ClearDevice** – очищує графічний екран, покажчик встановлюється у лівий верхній кут, екран заповнюється кольором, заданим процедурою SetBkColor.
- **PutPixel (X, Y, Color)** – виводить на екран точку кольору Color.
- **Line (X1, Y1, X2, Y2)** – малює лінію між точками (X1, Y1) та (X2, Y2).
- **LineTo (X, Y)** – малює лінію від поточної точки до точки (X, Y).
- **SetLineStyle (Typi, Pattern, Thick)** – задає вигляд ліній. Тут Typi – тип лінії; Pattern – зразок лінії; Thick – товщина лінії. Тип лінії задається константою з таблиці:

Const	Значення	Характеристика
SolidLn	0	Безперервна
PottedLn	1	Пунктирна
CenterLn	2	Штрих-пунктирна
DashedLn	3	Штрихова
UserBitLn	4	Задана

Параметр *Pattern* тільки для ліній типу *UserBitLn* і може приймати значення від 1..65536, тобто 2 байта кожен біт (із 16 біт слова) може приймати значення 0 або 1 (піксель не світиться або світиться). Отже параметр *Pattern* задає відрізок ліній, довжиною 16 пікселів. Цей шаблон періодично повторюється за всією довжиною лінії.

Параметр Thich приймає два значення: Thick=l – товщина лінії в 1 піксель. Thick=2 – товщина лінії в 3 пікселі.

- **Rectangle(X1, Y1, X2, Y2)** – малює прямокутник з використанням поточного кольору і поточного стилю ліній, із задами координатами діагонально протилежних вершин.
- **DrawPoly(N, P)** – малює довільну ламану лінію. N – кількість точок зламу, включаючи крайні; P – змінна типу PointType. При малюванні використовуються поточний колір і поточний стиль ліній.
- **Circle(X, Y, R)** – малює коло з центром X, Y і радіусом R.
- **Arc(X, Y, Beg A, End A, R)** – малює дугу кола Beg A і End A, відповідно, початковий і кінцевий кути дуги.

- **Ellipse(X, Y, Beg A, End A, RX, RY)** – малює дугу еліпса з центром X, Y; Beg A, End A – початковий і кінцевий кут, RX і RY – горизонтальний і вертикальний радіуси.
- **Bar (X1, Y1, X2, Y2)** – малює зафарбований прямокутник.
- **Bar3D (X1, Y1, X2, Y2, Depth, Top)** – малює тривимірне зображення паралелепіпеда і замальовує його передню грань. Depth – глибина третього виміру, Top=true – верхня грань викреслюється, false – ні.
- **FillPoly (n, Coords);** – обводить лінією і замальовує замкнутий багатокутник; n – кількість вершин, Coords – змінна типу PointType, що містить координати вершин.
- **FillEllipse(X, Y, Rx, Ry)** – обводить лінією і замальовує еліпс.
- **Sector(X, Y, Beg A, End A, Rx, Ry)** – малює і замальовує сектор еліпса.
- **PieSlice(X, Y, Beg A, End A, R)** – малює і замальовує сектор кола.
- **FloodPill(X, Y, Color)** – замальовує довільну замкнуту лінію (точки з'єднання повинні збігатися) поточним стилем і поточним кольором. X, Y – координати точки усередині фігури, Color – колір обмежень лінії.
- **GetImage(X1, Y1, X2, Y2, Buf)** – поміщає в пам'ять копію прямокутного фрагмента зображення. Buf – змінна, куди буде поміщена копія відеопам'яті зфрагментом зображення.
- **PutImage(X, Y, Buf, Mode)** – виводить в задане місце екрану копію фрагмента зображення, раніше поміщену в пам'ять процедурою GetImage. X, Y – координати лівого кута того місця на екрані, куди буде скопійований фрагмент зображення. Buf – змінна, звідки береться фрагмент зображення. Mode – спосіб копіювання.

При відображенні тексту у графічному режимі слід пам’ятати, що всі дії виконуються тільки з рядковими константами і змінними, тому вся чисельна інформація повинна перетворюватися у символьну. Розглянемо основні засоби модуля *Graph* для виведення текстової інформації:

а) процедура **OutText(text:string)** – виводить на екран рядок тексту, починаючи з поточного розміщення графічного курсору;

б) процедура **OutTextXY(x, y, ‘text’)** – виводить рядок тексту починаючи з указаних координат;

в) процедура **SetTextStyle(Font, Direct, size: word)** – встановлює стиль тексту. Параметри: код шрифту, код орієнтації символів, розмір символів.

Розглянемо приклади програм, що демонструють основні прийоми роботи з графікою та практичне використання процедур і функцій модуля Graph.

Приклад 1. Позиціонування графічного курсору та визначення його координат.

```
  Program Graph2;
 Uses Graph;
  var Driver, Mode: integer;
 begin
   Driver:= Detect;
   InitGraph(Drive, Mode, ' ');{ініціалізація графічного режиму}
   if GraphResult <> 0 then
     begin
       writeln ('помилка');
       halt(1);
     end;
 MoveTo(GetMax Xdiv2, GetMax Ydiv2); {встановлюємо нове поточне
положення покажчика}
 OutTextXY(GetX, GetY, 'курсор по центру');
 MoveRel(-Get X div 2, -GetY div 2);  {встановлюємо зміни нових
координат покажчика}
 OutText('Курсор переміщено');
 readln;
 CloseGraph;
 end.
```

Приклад 2. Наступна програма демонструє виведення різних простих об'єктів – кіл, дуг, еліпсів, прямокутників, паралелепіпедів тощо.

```
  Program Graph3;
 Uses Graph;
 var Driver,Mode,Radius,i,Width,y0,y1,y2,x1,x2:integer;
 Begin
 Driver:=detect;                {визначення номера драйвера}
   InitGraph  (Driver,Mode,  '  ');  {ініціалізація  графічного
режиму}
   if  GraphResult  =0 then
   begin
    for  radius:=1  to  10  do {малюємо різними кольорами 10
концентричних кіл з центром посередині екрана}
     begin
      setcolor(r div 10);
```

```
      Circle(320, 240,Radius*10); readln ;
     end;
    rectangle(220,  140,  420,340);  {описуємо навколо кіл
прямокутник}
    ClearDevice;          {очищуємо екран}
    Ellipse(200,200,0,360,30,50); {малюємо замкнутий еліпс}
      readln ;
    Ellipse(300,200,0,180,50,30);  {малюємо півеліпса}
         readln ;
          Width:=10 ;
          for  i:=1  to  5 do
   Bar(10+i*Width,300+i*10,20+i*Width,400);{п'ять прямокутників}
         readln ;
   Bar3D(210,300,250,360,10,TopOn);{паралелепіпед  з  верхньою
гранню}
        readln ;
     CloseGraph ;          {закриваємо Graph}
         end
      else  Halt(1);       {відкрити режим Graph не вдалося}
   End.
```

Приклад 3. Розглянемо використання засобів модуля Graph для побудов графіків функцій. Нехай потрібно вивести на екран графік функції *y=2cos(x)+1*. Програмна реалізація цього завдання може бути такою:

```
 program Graph4;
 uses graph;
 var i,y:integer;
     driver, mode: Integer;
     a,b:integer;
     k,l:real;
 function f(x:real):real;           {вихідна функція}
 begin
   f:=2cos(x)+1;
 end;
 begin
  a:=10;   b:=240;
  k:=0.1;  l:=200;
  driver:=detect;      {визначення номера драйвера}
  initgraph(driver,mode,'');   {ініціалізація графічного режиму}
   if graphresult=0 then
   begin
     y:=-trunc(f(0)*l);    {визначення значення координати для
першої точки графіку}
     moveto(a,y+b);       {перенесення пера в початок графіку}
     for i:=0 to 600 do
     begin
      y:=-trunc(f(i*k)*l);   {визначення проміжного значення Y}
```

```
      putpixel(i+a,y+b,8);  {виведення наступної точки графіку}
      lineto(i+a,y+b);        {малювання відрізка до наступної
точки}
     end;
     CloseGraph  ;       {закриваємо Graph}
   end ;
  else writeln('помилка при ініціалізації графіки');
 end;  readln;
 end.
```

ХІД РОБОТИ

1. Згідно з індивідуальним завданням розробити алгоритм розв’язку задачі.
2. Підготувати програму реалізацію на мові Pascal розроблених алгоритмів. Засобами вбудованого текстового редактора інтегрованого середовища розробки Turbo Pascal набрати тексти підготовлених програм. Якщо в умові немає конкретних даних, то їх потрібно задати на власний розсуд, керуючись змістом задачі.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати.
5. Оформити звіт до лабораторної роботи, в якому обов’язково навести алгоритми розв’язку задач, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Скласти програму для реалізації наступного завдання. Вигадати власний логотип (або емблему) та реалізувати його графічно, використовуючи основні графічні побудови, наведені в теоретичних відомостях. На рисунку обов’язково розмістити власне прізвище та ім’я.

Задача 2

Нарисувати графік функції згідно вашого варіанту. У результаті експериментів зобразити графік на екрані якнайкраще.

1. y=sin(x)+l на проміжку [0; 2π], табулюючи функцію з кроком h=0,2.
2. y=cos(2x) на проміжку [0; 2π], табулюючи функцію з кроком h=0,l.
3. y=2(x^2)+1 на проміжку [-3; 3], табулюючи функцію з кроком h=0,2.
4. y=2sin(x) на проміжку [0; 2π], табулюючи функцію з кроком h=0,15.
5. y=2cos(x) на проміжку [0; 2π], табулюючи функцію з кроком h=0,l.
6. y=2sin(x) на проміжку [0; 2π], табулюючи функцію з кроком h=0,15.
7. y=$\sqrt{2x+4}$ на проміжку [0; 3], табулюючи функцію з кроком h=0,2.
8. y=cos(x+2) на проміжку [0; 2π], табулюючи функцію з кроком h=0,l.
9. y=sin(3x)+l на проміжку [0; 2π], табулюючи функцію з кроком h=0,3.
10. y=sin(0,5x) на проміжку [0; 2π], табулюючи функцію з кроком h=0,2.
11. y=2cos(0,5x) на проміжку [0; 2π], табулюючи функцію з кроком h=0,l.
12. y=5sin(x) –2 на проміжку [0; 2π], табулюючи функцію з кроком h=0,2.
13. y= cos(x^2) на проміжку [0; 2π], табулюючи функцію з кроком h=0,2.
14. y=sin(x^2)+l на проміжку [0; 2π], табулюючи функцію з кроком h=0,3.
15. y=cos(x^2) –2 на проміжку [0; 2π], табулюючи функцію з кроком h=0,l.

ЛАБОРАТОРНА РОБОТА № 13

Основні прийоми роботи в середовищі візуального програмування Delphi. Робота з об’єктами Form, Label, Button, Image, Edit

МЕТА РОБОТИ: ознайомлення з середовищем візуального програмування Delphi та вироблення навичків роботи з об'єктами: форма (Form), текстове поле (Label), зображення (Image), кнопка (Button); особливостями побудови прикладних програм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Середовище візуального програмування Delphi – це графічна автоматизована оболонка над об'єктно-орієнтованою версією мови Pascal

(Object Pascal). Структурною одиницею в даному випадку є візуальний об'єкт, який називається компонентом. Процес розробки програм автоматизується завдяки можливості додавати стандартні компоненти на форму (у програму) та змінювати їх властивості, не вносячи вручну змін до програмного коду.

Технологія роботи у середовищі Delphi базується на ідеях об'єктно-орієнтованого та візуального програмування. Ідея об'єктно-орієнтованого програмування полягає в інкапсуляції (об'єднанні) даних і засобів їхнього опрацювання (методів) у тип, який називається класом. Конкретною змінною певного класу є об'єкт. Прикладами об'єктів можуть бути елементи керування у вікні: кнопки, списки, текстові поля тощо.

Всю сукупність файлів, потрібних для створення програми називають: **проект** або **Delphi-проект**. Проект Delphi складається з декількох пов’язаних файлів. Деякі з них створюються при написанні програми і побудові форм, інші при компіляції проекту. Для ефективної роботи з проектами Delphi необхідно знати призначення кожного файлу.

1. Файл проекту (файл з розширенням .dpr) – текстовий файл програми на мові Object Pascal, яка створює необхідні форми та передає управління головній формі проекту. Не можна змішувати поняття **Delphi-проекту** (сукупності файлів) та **файлу проекту** (одного файла з розширенням **.dpr**)

2. Файли форм (файли з розширенням .dfm) – призначені для збереження результатів візуальної розробки програми.

3. Файли модулів (файли з розширенням .pas) – містять тексти окремих модулів проекту.

4. Файли ресурсів (файли з розширенням .res)

5. Файли відкомпільованих модулів (файли з розширенням .dcu) – містять код, утворений в результаті компіляції тексту відповідного **.pas –** файла модуля.

6. Файли конфігурації та опцій проекту (файли з розширенням .cfg та .dof) – зберігають настройки середовища Delphi та індивідуальні настройки цього проекту.

7. **Файли резервних копій: файли з розширенням .~****

Файли відкомпільованих модулів, конфігурації та резервних копій можна знищувати – система Delphi створює їх автоматично. При копіюванні проекту вказані файли також не потрібні.

Для запуску проекту на виконання потрібно виконати команду Прикладного Меню Run → Run або натиснути кнопку  "Run" на Панелі Інструментів, або натиснути клавішу F9 клавіатури. Для збереження Delphi-проекту необхідно виконати команду File→Save Project As, після чого задати імена модулів та ім'я проекту.

Головне меню редактора містить пункти: File, Edit, Search, View, Project, Run, Component, Database, Tools, Help. Меню ***File*** містить стандартні команди для роботи з файлами проекту.

За допомогою команд меню ***Edit*** можна вирівнювати компоненти відносно сітки та між собою, змінювати розмір вибраного компонента та масштабувати візуальні компоненти. Меню ***Search*** містить стандартні команди пошуку та заміни фрагмента тексту. У меню ***View*** розташовані команди візуалізації елементів середовища. Меню ***Project*** містить команди компіляції та перевірки синтаксису програми. Меню ***Run*** містить команди налагодження та запуску програми. Меню ***Component*** використовують для створення та інсталяції нових компонентів. Меню ***Database*** містить команди виклику інструментів бази даних. У меню ***Tools*** перебувають команди для завдання параметрів середовища.

Панель інструментів служить для розташування кнопок інструментів. На ній можуть міститися кнопки всіх згаданих команд.

Робота з консольними додатками

Окрім додатків, що використовують графічні інтерфейси, Delphi пропонує можливість створення простих програм у стані MS-DOS – консольні додатки. Для створення консольного додатку необхідно виконати: File → New → Other (рис. 1.).

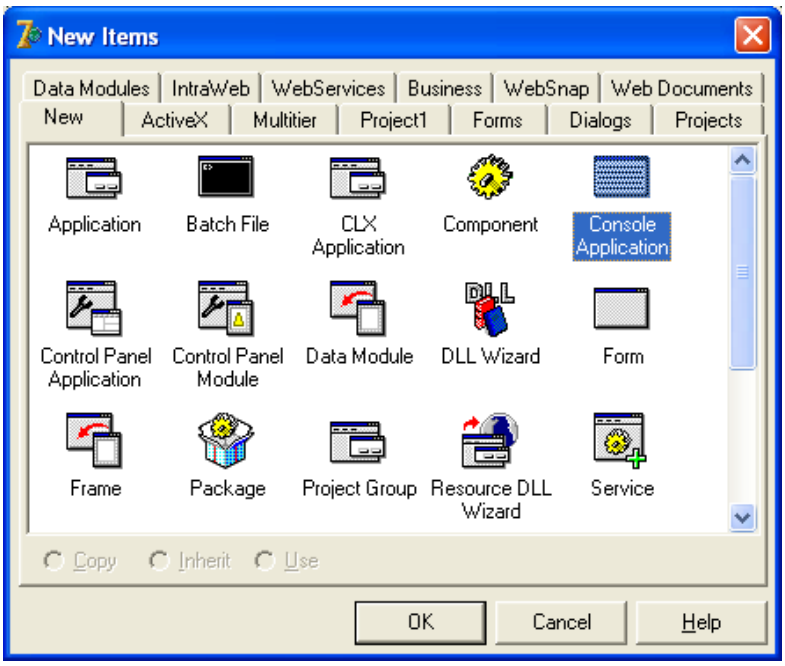


Рис. 1. Вибір консольного додатку

На вкладці, що відкриється, слід обрати варіант Console Application, після чого автоматично відкриється вікно редактора коду. Структура консольної програми є такою ж самою, як і структура програм у середовищі Pascal.

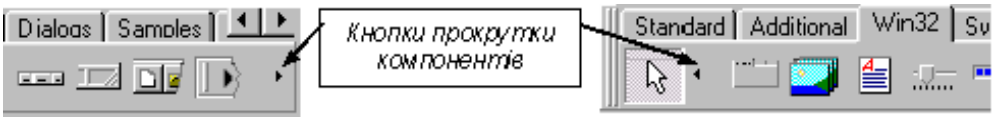
Робота з компонентами

Компоненти Delphi є основою для створення прикладних програм. З погляду користувача, компоненти – це об'єкти, за допомогою яких можна створити інтерфейс програми або додати невізуальні елементи. З погляду розробника, компоненти – це код на мові Object Pascal, організований у вигляді класів, які визначають стан та поведінку для елементів, що використовуються.

Компоненти можуть бути візуальними та невізуальними. Перші відображаються на екрані під час виконання програми, призначені для організації діалогу з користувачем. Це різні кнопки, списки, текстові поля, зображення тощо. Невізуальні компоненти призначені для доступу до системних ресурсів комп'ютера. Компоненти розташовуються на формі – спеціальному компоненті, який володіє властивостями вікна Windows.

Для вибору компонента і розміщення його на формі необхідно: 1) перейти на потрібну закладку Палітри Компонентів; 2) клацнути мишкою по потрібному компоненту; 3) клацнути мишкою по формі – там, де потрібно розмістити цей компонент. Якщо на цій закладці Палітри Компонентів

знаходиться більше компонентів, ніж поміщається на екрані, то автоматично виводяться кнопки прокрутки компонентів. Для перегляду тих компонентів, які у цей момент невидимі, необхідно скористатися цими кнопками прокрутки компонентів:



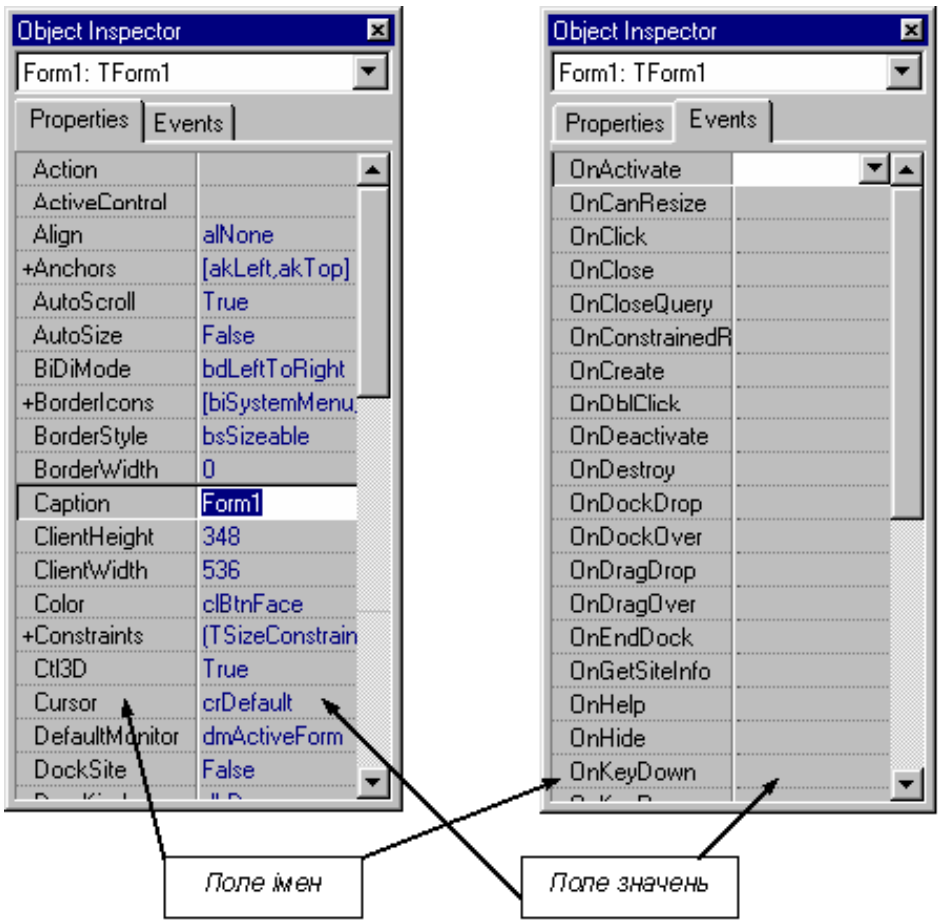
Для розташування на формі декількох однойменних об’єктів потрібно натиснути клавішу Shift і клацнути спочатку на його піктограмі, а потім у вікні форми. Вибраний компонент можна переміщати на формі, а також змінювати розміри, перетягуючи його маркери. Для переходу на потрібну закладку Палітри Компонентів потрібно клацнути по ній.

Майбутнє вікно Windows-програми створюється в Конструкторі Форм. Візуальне конструювання програми полягає в розміщенні потрібних елементів керування на формі, – ці елементи управління будуть так само відображатися і при виконанні програми. При виконанні програми кожній формі відповідає своє вікно, яке має такий самий вигляд, що і форма цього вікна (тобто, ті самі елементи керування, розміщені так само, як було задано при конструюванні форми). Отже, форма служить проектом майбутнього вікна програми.

Робота з Інспектором об'єктів

За допомогою інспектора об'єктів можна вказувати характеристики компонентів і визначати зв’язок між подіями та підпрограмами-реакціями на них. Вікно інспектора об'єктів містить список компонентів поточної форми, а також дві закладки: властивостей (Properties) та подій (Events). За допомогою закладки Properties встановлюються значення нетипових характеристик для компонента, закладка Events містить інформацію, яка описує поведінку цього компонента.

Закладка властивостей цього вікна складається з двох стовпців: лівий містить назви властивостей компонентів, правий – їхні значення. Властивості можуть бути простими або комплексними (складаються з набору інших властивостей, позначені символом "+", наприклад, + Font).



Закладка подій також має два стовпці. У лівому відображаються імена стандартних подій, на які об'єкт може реагувати, а в правому – імена процедур, які будуть реалізовувати реакцію на подію. Кожній стандартній події відповідає процедура, яка з'являється після подвійного клацання мишею у правому стовпці. У цей момент у вікно тексту програми додається шаблон базового коду (процедури), який треба заповнити.

Робота з редактором коду

Під час проектування форми в Delphi зазвичай потрібно писати код для реакції на деякі події. При натисканні кнопки миші в формі або на компоненті, Windows посилає додатку повідомлення про цю подію. Реакція Delphi полягає в отриманні повідомлення та викликі відповідної процедури для відклику на подію. Інформацію про події, що доступні для форми або компонента можна отримати з вкладки Events вікна Object Inspector при виборі відповідного елемента.

Додавання нових компонентів у вікно або застосування методу до них веде до змін у вікні редактора коду (поява нових змінних, заготовок базового коду відповідних процедур тощо). Наприклад, якщо додати на форму кнопку

і два рази клікнути мишкою по доданому компонету, Delphi активізує вікно кода з таким текстовим фрагментом:

```
procedure TForm1.Button1Click(Sender: Tobject);
begin
…
end;
```

Слово procedure повідомляє компілятор про початок процедури. Між командними словами `begin i end` (в тілі процедури) потрібно помісти опис дій, що мають відбутись в результаті виконання процедури. Наприклад, розташований в даній процедурі код `Form1.Color:= clTeal;` призведе до зміни кольору фону форми при натисканні на кнопку.

Компонент Form

Форма в Delphi являє собою видиме вікно Windows і є основною частиною практично будь-якої програми. Терміни ”форма” і ”вікно” є синонімами. На формі розміщуються візуальні компоненти, які утворюють інтерфейсну частину програми, і системні (невізуальні) компоненти. Для роботи з формою призначена компонента `Form` класу `TForm`. Зі створення форми починається конструювання програми. Розглянемо основні властивості форми:

Властивість	**Опис властивості**	**Приклади значень**
ActiveControl	Для задання активного об'єкта (фокуса) у формі	Buttonl, Edit2
AutoScroll	Наявність у формі смуг прокручування	True, False
BorderStyle	Можливість змінювати розміри вікна	bsSizeable (вікно з довільними розмірами), bsDialog, bsNone (вікно з фіксованими розмірами)
Width,Height	Ширина і висота вікна у пікселях	503, 224 (числове значення)
Font	Шрифт	Комплексна властивість, задається у діалоговому вікні
HorizScrollBar VertScrollBar	Параметри смуг прокручування	Комплексна властивість
Icon	Задаємо піктограму, яка буде в заголовку форми під час виконання програми	(None) - стандартна піктограма для Delphi, або завантажена з певного файлу *.ico
Name	Ім'я форми	Forml (ідентифікатор)

Caption	Заголовок форми	Довільний рядок символів
Color	Колір фону форми	clGreen, clInfoBk (перелічуваний тип) або $004525B1 (числове значення – задається у діалоговому вікні)
Cursor	Вигляд вказівника миші на формі під час виконання проекту	crDrag, crCross, crHelp, crArrow (перелічуваний тип)
Enabled	Доступність для дій об'єктів у формі під час виконання	True, False
Left, Top	Координати лівого верхнього кутка вікна у пікселях	200, 108 (числове значення)
Position	Розміщення і розміри вікна у момент запуску програми	poScreenCenter, poDesigned
WindowState	Стан вікна у момент запуску програми	wsNormal, wsMaximized, wsMinimized

Властивість Name забезпечує доступ до імені форми, тобто до імені змінної, за допомогою якої ми можемо керувати формою. Ім‘я форми (тобто, значення властивості Name) повинне задовільняти вимогам мови Object Pascal: це повинна бути послідовність літер латинського алфавіту (причому компілятор не розрізняє великих та малих букв), символу “_” (підкреслення) та цифр. Починатися ім’я повинне із літери латинського алфавіту або символа “_” (підкреслення).

Властивість Name – літерного типу. Встановлення значень для всіх властивостей літерного типу виконується подібним чином, для цього потрібно:

1) виділити в Інспекторі Об’єктів відповідну властивість (наприклад, клацнувши мишкою по ній);
2) в полі значень відредагувати текст;
3) натиснути клавішу Enter на клавіатурі.

Компонент Label

Об'єкт Label використовують для створення текстових полів (написів) у вікні програми. Вміст поля визначається значенням властивості Caption. Окрім аналогічних до наведених у попередній таблиці властивостей Width, Height, Font, Color, Name, Caption, Cursor, Enabled, Left, Top, він володіє такими властивостями:

Властивість	Опис властивості	Приклади
Align	Вирівнювання поля відносно об'єкта, що його містить (форми)	alBottom, alClient, alLeft, alNone, alTop
Alignment	Вирівнювання тексту в межах поля	taCenter, taLeftJustify,
AutoSize	Приведення меж поля до границь тексту	True, False
Visible	Видимість об'єкта	True, False
Wordwrap	Перенесення слів тексту у новий рядок	True, False

Компонент Image

Об'єкт Image використовують для додавання графічних об'єктів з файлів типу *.bmp, *.emf, *.ico, *.wmf у форму. Окрім відомих властивостей Align, Width, Height, Name, Cursor, Enabled, Left, Top, Visible, можна виділити наступні:

Властивість	Опис властивості	Приклади
Center	Вирівнювання малюнка до центру відносно поля, що його містить	True, False
Picture	Ім'я графічного файлу	Задається у діалоговому вікні
Stretch	Приведення розміру зображення до заданих розмірів об'єкта	True, False
AutoSize	Приведення розміру об'єкта до реальних розмірів зображення	True, False

Компонент Button

Об'єкт Button використовують для створення кнопок на формі. Кнопки мають такі властивості: Visible, Width, Height, Font, Color, Name, Caption, Cursor, Enabled, Left, Top та інші.

Компонент Edit

В компоненті Edit текст, що вводиться (виводиться), міститься у властивості Text, яку можна встановлювати в процесі проектування або задавати програмно. Вирівнювання тексту, як в текстових полях (Label), неможливе. Введення даних з поля введення здійснюється звертанням до властивості Text.

ХІД РОБОТИ

1. Виконати програмну реалізацію в середовищі Delphi поставлених завдань згідно з Вашим варіантом.
2. Відкомпілювати, налагодити та запустити програми на виконання.
3. Протестувати програми і проаналізувати отримані результати.
4. Оформити звіт до лабораторної роботи, в якому обов’язково навести алгоритми виконання завдань, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Задача 1

Ралізувати свій варіант завдання лабораторної роботи 11 (“Рядкі символів та дії з ними”) в середовищі Delphi за допомогою консольного проекта. Зберігти у власній папці exe-файл програми.

Задача 2

1. Створіть новий проект. Змініть колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я». Змініть значення властивостей Font (шрифт) цього текстового поля на такі:

Font: Times New Roman Cyr;
Font Style: Bold;
Size: 16
Color: Purpure.

Аналогічно вставте у форму ще декілька текстових полів з вашими даними. Розташуйте об’єкти якнайкраще. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Поекспериментуйте з властивістю Visible (видимість) зображення.

2. Створіть новий проект. Змініть колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище , Ім’я», встановіть значення властивостей Font (шрифт) цього текстового поля за власним вподобанням. Визначте випливаючу підказку для вказаного компонента. Додайте на форму кнопку та поле для введення даних (об’єкт типу Edit). Створіть для кнопки власну підпрограму – реакцію на подію: натискання будь-якої клавіші

(OnKeyPress), яка встановлює колір фону компонента Edit. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image.

3. Створіть новий проект. Змініть заголовок форми. Вставте у форму текстове поле (об’єкт типу Label) та кнопку. Створіть для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює напис мітки «Прізівище , Імя». Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Поекспериментуйте з властивістю Visible (видимість) зображення. Програмно змініть розміри форми.

4. Створіть новий проект. Змініть розміри форми за власним вподобанням. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище , Імя». Встановіть для компонента Label колір фону та змініть форму курсора. Аналогічно вставте у форму кнопку та поле для ввдення даних (об’єкт типу Edit). Розташуйте об’єкти якнайкраще. Створіть для кнопки власну підпрограму – реакцію на подію: натискання будь-якої клавіші (OnKeyPress), яка встановлює напис «введіть дані» в компоненті Edit. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Визначити випливаючу підказку для вказаного компонента.

5. Створіть новий проект. Змініть колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Імя». Встановіть для компонента Label колір фону Аналогічно вставте у форму ще декілька текстових полів з вашими даними. Розташуйте об’єкти якнайкраще. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image Поекспериментуйте з властивістю Visible (видимість) зображення

6. Створіть новий проект. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я» та кнопку. Створити для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює напис заголовка вікна форми. Поекспериментуйте з доступністю кнопки (доступна або ні) Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image1

7. Створіть новий проект. Змініть заголовок форми форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я». Змініть значення властивостей Font (шрифт) цього текстового поля на такі:

Font: Ariel;

Font Style: Bold;

Size: 20

Color: Red.

Додайте на форму кнопку. Створіть для кнопки власну підпрограму – реакцію на подію: натискання будь-якої клавіші (OnKeyPress), яка змінює колір фону форми. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image.

8. Створіть новий проект. Змініть колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я». Встановіть для компонента (Label) колір фону та змініть форму курсора Аналогічно вставте у форму ще декілька текстових полів з вашими даними та кнопку. Розташуйте об’єкти якнайкраще. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Визначити випливаючу підказку для вказаного компонента.

9. Створіть новий проект. Встановіть вигляд курсора для форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Імя» та кнопку. Додайте об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Встановити оптимальні розміри зображення. Створити для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює видимість або невидимість зображення.

10. Створіть новий проект. Змініть розміри форми за власним вподобанням. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Імя». Аналогічно вставте у форму кнопку та поле для ввдення даних (об’єкт типу Edit). Розташуйте об’єкти якнайкраще. Створіть для кнопки власну підпрограму – реакцію на подію: натискання будь-якої клавіші (OnKeyPress), яка встановлює напис «введіть дані» компонента Edit та колір фону для нього. Вставте у форму об'єкт типу Image (зображення). Вставте свою

фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Визначте нову форму курсора для даного об’єкта.

11. Створіть новий проект. Змініть заголовок та колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з написом «Прізвище, Ім’я» та кнопку. Створити для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює розміри вікна форми. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image Поекспериментуйте з властивістю Visible (видимість) зображення.

12. Створіть новий проект. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я» та кнопку. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Створіть власну підпрограму – реакцію на подію: клацання по формі (OnClick), яка встановлює заголовок форми. Створити для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює зміну кольору мітки.

13. Створіть новий проект. Змініть колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я» змініть значення властивостей Font (шрифт) цього текстового поля на такі:

Font: Times New Roman Cyr;
Font Style: Bold;
Size: 18
Color: Red.

Додайте на форму кнопку. Створіть для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює доступність мітки. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image.

14. Створіть новий проект. Змініть розміри форми за власним вподобанням. Вставте у форму текстове поле (об’єкт типу Label) з текстом «Прізвище, Ім’я». Встановіть для компонента Label колір фону та змініть форму курсора. Аналогічно вставте у форму кнопку та поле для ввдення даних (об’єкт типу Edit). Розташуйте об’єкти якнайкраще. Створіть для кнопки власну підпрограму – реакцію на подію: натискання будь-якої клавіші (OnKeyPress), яка встановлює напис «введіть дані» в компоненті Edit. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію

за допомогою властивості Picture (ілюстрація) об'єкта Image. Визначити випливаючу підказку для вказаного компонента.

15. Створіть новий проект. Змініть заголовок та колір фону форми. Вставте у форму текстове поле (об’єкт типу Label) з написом «Прізвище, Ім’я» та кнопку. Створити для кнопки власну підпрограму – реакцію на подію: клацання по кнопці (OnClick), яка встановлює розміри вікна форми. Аналогічно вставте у форму ще декілька текстових полів з вашими даними. Розташуйте об’єкти якнайкраще. Вставте у форму об'єкт типу Image (зображення). Вставте свою фотографію за допомогою властивості Picture (ілюстрація) об'єкта Image. Визначте нову форму курсора для даного об’єкта.

ЛАБОРАТОРНА РОБОТА № 14

Робота в середовищі візуального програмування Delphi. Програмування кнопок. Опрацювання подій

МЕТА РОБОТИ: вироблення практичних навичків програмування кнопок та опрацювання подій, закріплення навичків роботи з компонентами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Windows є середовищем, що керується подіями (event-driver). **Подія (Event) –** це те, що відбувається під час роботи програми. До подій належать переміщення вікна, кліки кнопками миші, натиснення клавіш клавіатури, тощо. У Delphi кожній події привласнено ім'я. Наприклад, клацання кнопкою миші – це подія OnClick, подвійне клацання мишею подія OnDblClick. Список подій, на які реагує компонент, наведений на укладці Events вікна інспектора об’єктів. Ім’я події одночасно слугує її описом.

Так як і властивості, багато подій є спільними для різних об’єктів. Найбільш поширеною є подія OnClick, яка відбувається у момент клікання

мишкою – вона визначена майже для всіх візуальних компонент. У таблиці приведені деякі події та умови їхнього настання.

Подія	**Умова настання**
OnClick	При клацанні кнопкою миші
OnDblClick	При подвійному клацанні кнопкою миші
OnMouseDown	При натисненні кнопки миші
OnMouseUp	При відпуску кнопки миші
OnMouseMove	При переміщенні миші
OnKeyPress	При натисненні клавіші клавіатури
OnKeyDown	При натисненні клавіші клавіатури. Події OnKeyDown і OnKeyPress — це події, що чергуються, які відбуваються до тих пір, поки не буде відпущена утримувана клавіша (у цей момент відбувається подія OnKeyUp)
OnKeyUp	При відпусканні натиснутої клавіші клавіатури
OnCreate	При створенні об'єкту (форми, елементу управління). Процедура обробки цієї події зазвичай використовується для ініціалізації змінних, виконання підготовчих дій
OnPaint	При появі вікна на екрані на початку роботи програми, після появи частини вікна, яка, наприклад, була закрита іншим вікном, і в інших випадках
OnEnter	При отриманні елементом управління фокусу
OnExit	При втраті елементом управління фокусу

Кожен компонент спроектований так, щоб реагувати на певні події. У Delphi реакція на подію реалізується як процедура обробки події. Отже, для того, щоб програма виконувала деяку роботу у відповідь на дії користувача, програміст повинен написати процедуру обробки відповідної події.

Слід звернути увагу на те, що значну частину обробки подій бере на себе компонент. Тому програміст повинен розробляти процедуру обробки події тільки в тому випадку, якщо реакція на подію відрізняється від стандартної або не визначена. Наприклад, якщо по умові завдання обмежень на символи, що вводяться в поле Edit, немає, то процедуру обробки події

OnKeyPress писати не треба, оскільки під час роботи програми буде використана стандартна (прихована від програміста) процедура обробки цієї події. Методику створення процедур обробки подій розглянемо на прикладі процедури обробки події OnClick для командної кнопки Обчислити.

Щоб приступити до створення процедури обробки події, треба спочатку у вікні Object Inspector вибрати компонент, для якого створюється процедура обробки події. Потім в цьому ж вікні потрібно перейти на вкладку Events (Події), на якій перераховані події, що може сприймати компонент.

У лівій колонці вкладки Events перелічені імена подій, які може сприймати вибраний компонент (об'єкт). Якщо для події визначена (написана) процедура обробки події, то в правій колонці поряд з ім'ям події виводиться ім'я цієї процедури. Для того, щоб створити функцію обробки події, потрібно два рази клацнути мишею в полі імені процедури обробки відповідної події. У результаті цього відкриється вікно редактора коду, в яке буде доданий шаблон процедури обробки події, а у вікні Object Inspector поряд з ім'ям події з'явиться ім'я функції його обробки.

Наприклад, якщо додати на форму кнопку і два рази клікнути мишкою по доданому компонету, Delphi активізує вікно кода з таким текстовим фрагментом:

```
procedure TForm1.Button1Click(Sender: Tobject);
begin
end;
```

Слово **procedure** повідомляє компілятор про початок процедури. Delphi привласнює функції обробки події ім'я, яке складається з двох частин. Перша частина імені ідентифікує форму, що містить об'єкт (компонент), для якого створена процедура обробки події. Друга частина імені ідентифікує сам об'єкт і подію. У нашому прикладі ім'я форми – Form1, ім'я командної кнопки – Button1, а ім'я події – Click. Параметр Sender означає обєкт, що є джерелом події.

Крім того, при цьому в розділі `type` з'явився запис про відповідну процедуру:

```
type
TForm1 = class(TForm)
Button1: Tbutton;
procedure TForm1.Button1Click(Sender: Tobject);
```

У вікні редактора коду між командними словами `begin` і `end` (в тілі процедури) потрібно помісти опис дій, що мають відбутися в результаті виконання процедури.

Засоби введення та виведення даних

Одним з першочергових питань при написанні програм є отримання вхідних даних та виведення результатів роботи програми. Програма може отримати вихідні дані з поля введення, вікна введення, діалогового вікна або з файлу.

Вікно введення – це стандартне діалогове вікно, яке з'являється на екрані в результаті виклику функції InputBox. Значення функції InputBox – рядок, який ввів користувач. У загальному вигляді інструкція введення даних з використанням функції InputBox виглядає так: Змінна: = InputBox ('текст заголовка вікна введення', ' текст інформаційного повідомлення', 'значення за замовчуванням'). Наприклад, за допомогою наступної процедури можна отримати вихідні дані для програми знаходження площі трикутника за формулою Герона:

```
 procedure TForm1.Button1Click(Sender: TObject);
 var
 s: string;
 begin
 s:=InputBox('Площа трикутника','Введіть довжину сторони
a','0');
 Edit1.Text:= s;
 end;
```

Поле введення – це компонент Edit, в якому дані, що вводяться або виводяться, є значенням властивості Text. Всі дані, введені за допомогою компонента Edit є символьними величинами, тому для роботи з дійсними або цілими числами потрібно скористатися функціями: `StrToFloat` та `StrToInt` відповідно (для зворотніх перетворень використовуються `FloatToStr` та `IntToStr`).

Для виведення результатів роботи програми можна використовувати вікно повідомлення або поле виводу (компонент Label).

Вміст компоненту Label визначається значенням властивості Caption, змінити яке можна як під час розробки форми програми, так і під час роботи програми. Властивість Caption символьного типу, тому для того, щоб під час роботи програми вивести в поле мітки числове значення, потрібно

перетворити число в рядок за допомогою функції `FloatToStr` або `IntToStr`. Наприклад:

```
Label2.Caption:= 'Площа трикутника:'+ #13+ 'S='+FloatToStr(S);
```

Для виведення інформації також можна скористатися компонентом Panel, основним призначення якого є групування компонентів у вікні форми. Цей компонент має властівості BorderStyle (рамка), BevelInner, BevelOuter, BevelWidth, BorderWidth, які надають ширші можливості оформлення тексту.

Вікна повідомлень використовуються для привертання уваги користувача. За допомогою вікна повідомлення програма можна вивести інформацію про помилку у вихідних даних або дати запит на підтвердження виконання деякої операції.

Вивести на екран вікно з повідомленням можна за допомогою процедури: ShowMessage або функції MessageDlg. Процедура ShowMessage виводить на екран вікно з текстом і командної кнопкою ОК. У загальному вигляді інструкція виклику процедури ShowMessage виглядає так: ShowMessage (const Msg: String). Наприклад `ShowMessage ('Рівняння не має коренів').`

Наступна процедура дає змогу перевірити дані введені в компонент TEdit, і вивести повідомлення користувачу, якщо замість латинських символів використана кирилиця:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if Key in ['A'..'я'] then
  begin
    Key:=#0;
    ShowMessage('Перейдіть на англійську розкладку');
   end;
end;
```

Користувач побачить попередження з тим же заголовком, що його має проект. Вікно буде містити одну кнопку "ОК", що відповідає тільки за закриття повідомлення, тому з нею не можна пов'язати якусь функцію.

Приклад 1. Задача знаходження суми скінченого ряду

Розглянемо розробку додатку для обчислення суми нескінченого ряду

$$S = \sum_{i=1}^{n} \left(\frac{i}{i+1} \prod_{k=i}^{n+1} \frac{i}{k+i}\right).$$

Реалізація відповідних процедур при виконанні проекту здійснюватиметься за законами алгоритму. Для реалізації цього проекту розташовуємо на формі такі компоненти:

Компоненти	**Властивість**	**Значення**
Label1	Caption	ОБЧИСЛЕННЯ СУМИ РЯДУ
Label2	Caption	Введіть числове значення n
Label3	Caption	Сумма ряду S=
Edit 1	Text	'0'
Edit2	Text	'0'
Button1	Caption	Очистка вікон (Edit)
Button2	Caption	ОБЧИСЛИТИ
Button3	Caption	ВИХІД

Розмістивши відповідні компоненти та встановивши зазначені властивості, отримаємо форму, наведену на рис. 1.

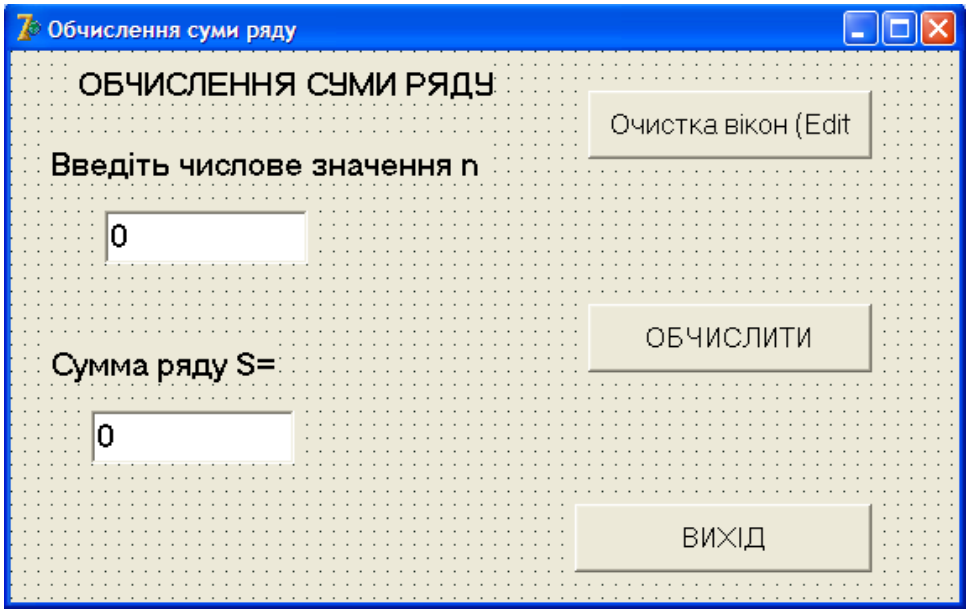


Рис.1. Вигляд головної форми проекту для обчислення суми числового ряду.

Для створеня цього додатку потрібно запрограмувати кнопки так:

Кнопка «Очистка вікон (Edit)»

```
procedure TForm1.Button1Click(Sender: TObject);
  begin
    Edit1.Clear;
    Edit2.Clear;
 end;
```

Кнопка «Вихід»

```
procedure TForm1.Button3Click(Sender: TObject);
begin
```

```
  Close;
end;
```

Кнопка «Обчислити»

```
procedure TForm1.Button2Click(Sender: TObject);
var s,p:real;
    n,i,k:longint;
begin
 n:=StrToInt(Edit1.Text);
 S:=0;
 for i:=1 to n do
 begin
  p:=1;
  for k:=i to n+1 do
  p:=p*i/(k+i);
  s:=s+i/(i+1)*p;
 end;
  Edit2.Text:=FloatToStr(s);
end;
```

Результуюча форма проекту при заданому з клавіатури n=100 показує значення суми заданого числового ряду S.

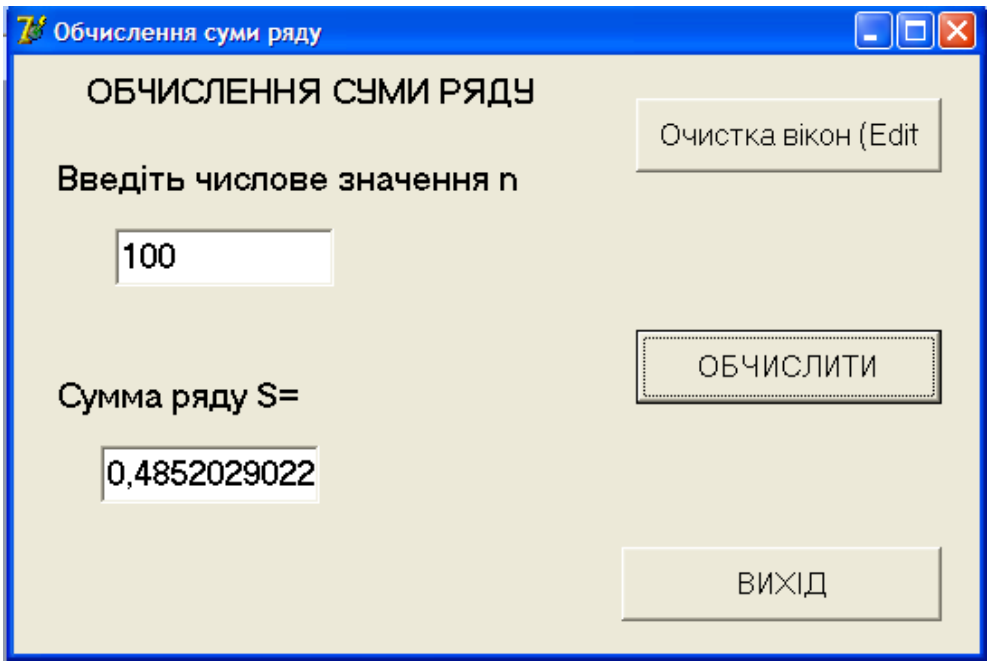


Рис. 2. Результат роботи програми.

Приклад 2. Задача знаходження площі трикутника

Розглянемо розробку додатку для обчислення площі трикутника за трьома сторонами. Сторони трикутника вводяться за допомогою компонента Edit. Для реалізації цього проекту розташовуємо на формі такі компоненти:

Компоненти	Властивість	Значення
Label1	Caption	Введіть довжину першої сторони a =
Label2	Caption	Введіть довжину другої сторони b =
Label3	Caption	Введіть довжину третьої сторони c =
Label3	Caption	результат
Edit 1	Text	''
Edit2	Text	''
Edit3	Text	''
Button1	Caption	Обчислити площу трикутника
Button2	Caption	Вихід

Програмуємо відповідні кнопки:

Кнопка «Обчислити площу трикутника»

```
procedure TForm1.Button1Click (Sender: TObject);
var  a, b, c, p, s: real;
begin
if(((edit1.Text  ='')  or  (edit2.Text  ='''))  or  (edit3.Text
='''))   then   label4.Caption:=('Введіть   всі   3   сторони
трикутника')
else begin
   a:= StrToFloat(edit1.Text);
   b:= StrToFloat(edit2.Text);
   c:= StrToFloat(edit3.Text);
if (((a = 0) or (b = 0)) or (c = 0)) then
label4.Caption:= ('такого трикутника не існує')
else
begin
 p:= (a + b + c) / 2;
 s:= sqrt (p * (p-a) * (p-b) * (p-c));
 if s = 0 then
label4.Caption:=('такого трикутника не існує')
else
label4.Caption:= ('Площа трикутника ='+ FloatToStr(s));
 end;
 end;
end;
end.
```

Кнопка «Вихід»

```
procedure TForm1.Button3Click(Sender: TObject);
begin
 ShowMessage('Ви дійсно хочете вийти з проекту');
 Close;
end;
```

Зовнішній вигляд програми подано на рис. 3

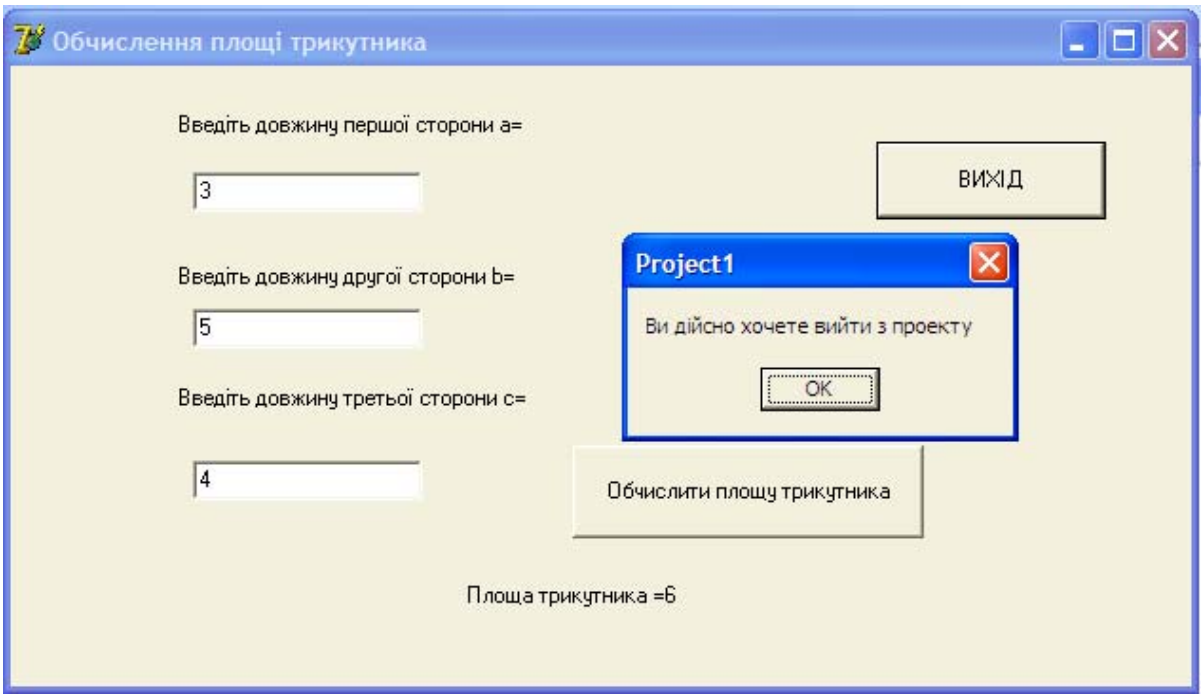


Рис. 3. Результат роботи програми для обчислення площі трикутника.

ХІД РОБОТИ

1. Згідно із індивідуальним завданням розробити алгоритм розв'язку задачі.
2. Виконати програмну реалізацію в середовищі Delphi поставлених завдань згідно з Вашим варіантом.
3. Відкомпілювати, налагодити та запустити програми на виконання.
4. Протестувати програми і проаналізувати отримані результати.
5. Оформити звіт до цієї роботи, в якому обов'язково навести алгоритми виконання завдань, тексти програм та результати виконання програми.

ІНДИВІДУАЛЬНІ ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

Створіть новий проект. Розташуйте на формі компоненти, потрібні для реалізації завдання Вашого варіанта та запрограмуйте відповідні кнопки. Запустіть проект на виконання та протестуйте роботу програми. Збережіть у власній папці exe-файл програми.

1. Обчислити суму, різницю, добуток та частку двох введених чисел.

2. Обчислити дробову частину середнього арифметичного трьох введених користувачем додатніх чисел.
3. Знайти корені квадратного рівняння $ax^2+bx+c=0$ за введеними коефіцієнтами *a, b, c*.
4. Обчислити суму числового ряду $S=\sum_{i=2}^{m}\frac{2i-11.45}{3i+\cos(4i+1)}$ для будь-якого введеного значення m .
5. Визначити вид трикутника за трьома довжинами його сторін (рівносторонній, рівнобедрений і не рівносторонній, різнобічний).
6. Обчислити значення функції

$$y=\frac{1}{x}-\begin{cases}0,65x+8 & ,\ x<1\\ 6,1+e^{x} & ,\ 1\le x<5\\ \sqrt{1+\sqrt{x}} & ,\ x\ge 5\end{cases}$$

для будь-якого значення x, що вводиться користувачем.

7. Визначити вид чотирикутника за значеннями довжин чотирьох його сторін та одного кута (квадрат, ромб, прямокутник, паралелограм, довільний чотирикутник).
8. За заданими значеннями змінних x, y, z, обчислити значення арифметичного виразу $u=tg^2(x+y)-e^{y-z}\sqrt{\cos x^2+\sin z^2}$.
9. Обчислити значення функції

$$y=\frac{1}{x}-\begin{cases}0,65x+8 & ,\ x<1\\ 6,1+e^{x} & ,\ 1\le x<5\\ \sqrt{1+\sqrt{x}} & ,\ x\ge 5\end{cases}.$$

для будь-якого значення x, що вводиться користувачем.

10. Обчислити периметр і площу прямокутного трикутника за заданими катетами.

11. За введеними значеннями *a, b, c* обчислити значення виразу $\frac{2c-2\sin(2a)}{3(b+9)(c^2-1)}+10e^{-2ab}$.

12. Обчислити суму скінченого ряду $S=\sum_{k=3}^{m}\frac{3k+4.5}{k^3+e^k}$ для будь-якого введеного значення *m.*

13. Обчислити цілу частину середнього арифметичного трьох введених користувачем цілих чисел.

14. За заданими значеннями змінних *a, b, c*, обчислити значення арифметичного виразу $n=e^{a^2}+tg(b-2c)$.

15. Обчислити суму цифр введеного користувачем трьохзначного числа.

Зразок оформлення звіту

Звіт до лабораторної роботи №2
з дисципліни «Програмування»
на тему: «Опрацювання рядків символів (string). Основні процедури для роботи з рядками»
студента групи ТТІ-21 Петренка Петра

Мета: вироблення практичних навиків роботи з даними, представленими у вигляді рядків символів.

Варіант 1

Завдання 1. Ввести з клавіатури власні прізвище, ім'я та по батькові як одне дане типу рядок. Визначити довжину отриманого рядка. Визначити, скільки літер '*a*' є в даному рядку.

Алгоритм виконання завдання 1

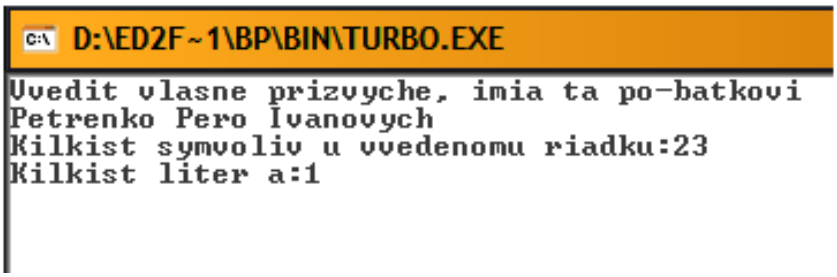
1. **…**
2. **…**

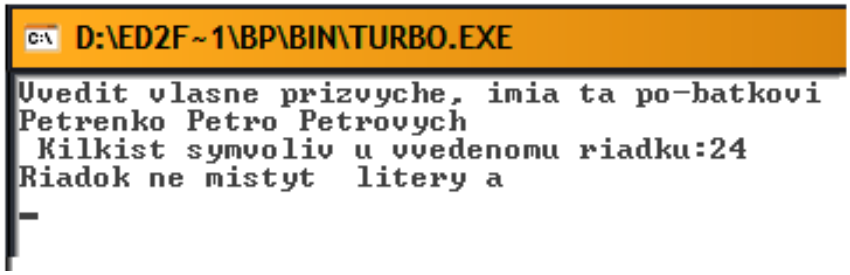
   . . .

Програмна реалізація завдання 1

```
Program  St;
var  C, I: Byte;
     S: String;
begin
  Writeln('Vvedit vlasne prizvyche, imia ta po-batkovi');
  Readln(S);
  C:=0;
  For I:= 1 to Length (S) do
   If S[I]='a' then C:=C+1;
   Writeln(' Kilkist symvoliv u vvedenomu riadku:', Length(S));
    If c=0 then
     Writeln('Riadok ne mistyt danoi litery')
      Else Writeln(' Kilkist liter a: ', C);
  Readln;
end.
```

Результат виконання програми





Завдання 2. Заданий рядок символів. Видалити з нього слово “line”, якщо воно є. В протилежному випадку вивести відповідне повідомлення.

Алгоритм виконання завдання

1. …
2. …

. . .

Програмна реалізація завдання 2

. . .

Результат виконання програми

. . .

Висновок:….

Література

1. Семотюк В. Програмування у середовищі Turbo Pascal, – Львів, 2000. – 238 с.
2. Моргун О. М. Довідник з Turbo Pascal для студентів. – М. : Діалектика, 2006. – 608с.
3. Глинський Я.М., Анохін В.Є., Ряжська В.А. Паскаль. Turbo Pascal і Delphi. Навчальний посібник. 10-те вид. – Львів: СПД Глинський, 2009. – 192 с.
4. Фаронов В.В. Turbo Pascal Найбільш повне керівництво в оригіналі. – Видавництво "ОМД Груп", 2003. – 1054 с.
5. Шпак Ю.А. Программирование в Turbo Pascal. Переход к Delphi. – Издательство "МК-Пресс", 2006. – 416 с.
6. Культин Н. Программирование в Turbo Pascal 7.0 и Delphi. – СПб. : БХВ-Петербург, 2012. – 390 с.
7. Зеленяк О. П. Современный задачник по Турбо Паскалю. – Издательство: ДМК Пресс, 2010. – 312 с.
8. Рубанцев В. Самоучитель Delphi в примерах, играх и программах. От простых приложений, решения задач и до программ. – Наука и Техника, 2011. – 672 с.
9. Шпак Ю.А. Turbo Pascal 7.0 на прикладах. – Видавництво "Юніор", 2003. – 498 с.
10. Рапаков Г. Г., Ржеуцкая С. Ю. Turbo Pascal для студентов и школьников. – БХВ-Петербург, 2007. – 352 с.
11. Климова Л. М. Pascal 7.0. Практическое программирование. Решение типовых задач. – Издательство: КУДИЦ-Образ, 2003. – 528 с.

12. Моргун О. М. Решение задач средствами языка Turbo Pascal 7.0. – Видавництво "Юніор", 2002. – 216 с.
13. Глинський Я. М. Інформатика: 10-11 класи: навч. посібник у 2 ч. – Ч. 1. Алгоритмізація і програмування. – Львів: СПД Глинський, 2011. – 256 с.
14. Галисеев Г.В. Компоненті в Delphi7. Профессиональная работа. – М. : Издательский дом «Вильямс», 2004. – 342 с.
15. Присяжнюк Т.А., Жуковський С.С. Структурне програмування мовою Pascal (лабораторний практикум). Методичний посібник для студентів фізико-математичного факультету. – Житомир: Видавництво ЖДУ, 2010. – 71 с.

Навчальне видання

Ірина Шаклеіна, Ольга Косовська

Програмування

Методичні рекомендації до виконання лабораторних робіт

Редакційно-видавничий відділ
Дрогобицького державного педагогічного університету
імені Івана Франка

Головний редактор
Ірина Невмержицька
Редактор
Леся Грабинська
Технічний редактор
Наталя Кізима
Коректор
Наталя Кізима

Здано до набору 10.08.2013 р. Підписано до друку 11.11.2013 р. Формат 60x90/16. Папір офсетний. Гарнітура Arial. Наклад 330 прим. Ум. друк. арк. 8,25. Зам 225.

Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка (Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції ДК №2155 від 12.04.2005 р.) 82100, Дрогобич, вул. І.Франка, 24, к. 42, тел. 2-23-78.